ALGORITMOS EVOLUTIVOS CELULARES CON RATIO VECINDARIO-POBLACIÓN DINÁMICA

Realizado por:
Bernabé Dorronsoro Díaz

Dirigido por:
Enrique Alba Torres

Departamento:
LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN

Titulación:
E.T.S.I. Informática

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS EN INFORMÁTICA

UNIVERSIDAD DE MÁLAGA

Málaga, Mayo de 2002

UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS EN INFORMÁTICA

Reunido el tribunal examinador en el día de la fecha, constituido por:
Presidente Dº/Dº_____
Secretario Dº/Dª_____
Vocal Dº/Dª _____
Para juzgar el Proyecto Fin de Carrera titulado:

Algoritmos Evolutivos Celulares con Ratio Vecindario-Población Dinámica.
Del alumno Dº/Dª Bernabé Dorronsoro Díaz_____
Dirigido por Dº/Dª Enrique Alba Torres_____

ACORDÓ POR _____ OTORGAR LA CALIFICACIÓN DE

_____

Y PARA QUE CONSTE, SE EXTIENDE FIRMADA POR LOS COMPARECIENTES DEL TRIBUNAL, LA PRESENTE DILIGENCIA

Málaga, a de de

El Presidente El Secretario El Vocal

Fdo: Fdo: Fdo:

Índice general

Índice de Tablas

Índice de figuras

Índice de Algoritmos

Capítulo 1

Planteamiento y Objetivos

En este primer capítulo se describirá una breve introducción al proyecto que nos ocupa, una sección en la que se detallan los objetivos perseguidos durante este trabajo, las fases que se han seguido durante el trabajo y una última sección en la que se explica la estructura que se le ha dado a esta memoria.

1.1. Introducción

El uso de algoritmos evolutivos (EAs) en problemas de optimización ha sido especialmente importante en la actual década [BFM97]. La utilización de este tipo de algoritmos resulta muy provechosa en terrenos de optimización matemática, diseño, optimización con restricciones, problemas con ruido y problemas donde existe una elevada epistasis[1]. La elevada complejidad y aplicabilidad de estos algoritmos ha promovido la aparición de novedosos modelos nuevos de optimización y búsqueda. Los EAs celulares son uno de estos nuevos modelos surgidos en donde la necesidad de investigación y nuevos análisis se considera prioritario ([JS96],[MS91]). Un EA celular dispone la población de soluciones en forma de cuadrícula toroidal y aprovecha esta topología bidimensional para dotar a la búsqueda de un aislamiento por distancia y una diversidad en las regiones del espacio de búsqueda que son de mayor magnitud que en EA's tradicionales. Sobre todo, la capacidad de exploración es muy útil, debido al trabajo paralelo de los vecindarios; al mismo tiempo, es especialmente simple añadir búsqueda local a cada uno de los puntos de búsqueda, generando así nuevos algoritmos híbridos. La relación entre la forma de la rejilla y la del vecindario de cada solución se denomina *ratio* del algoritmo celular. Esta ratio influye de manera definitiva en el tipo de búsqueda llevado a cabo. Una línea de interés abierta recientemente consiste en estudiar de qué forma ocurre este fenómeno y cómo pueden proponerse nuevos criterios que permitan modificar esta ratio para una optimización más eficiente [AT00]. Estos criterios girarán entorno a las

[1] Problema con epistasis es aquel en el que existe una intensa correlación entre los valores optimizados

velocidades relativas de mejora en la adecuación media de la población y la velocidad de disminución de la entropía en dicha población, distinguiendo además distintos tipos de actividades cuando estos criterios son alcanzados en distintos momentos de la búsqueda. En concreto, se propone en este trabajo extender los resultados descritos en [AT00] a nuevos problemas de interés, tanto en el dominio de la optimización combinatoria (problema del corte máximo, problema de tarea con espera mínima, etc.) como en otras disciplinas tradicionales de optimización. Los resultados del análisis pueden ofrecerse en términos del mantenimiento de la diversidad (entropía media de la población) o escenarios de evolución del mejor individuo (off-line) [CTTS99]. Esto ayuda de forma conjunta a caracterizar el funcionamiento de los EAs celulares y su aplicación a problemas reales. La relación con otros trabajos similares, así como la interpretación exhaustiva y cuidadosa de los resultados puede arrojar mucha luz sobre futuros algoritmos para este dominio, así como sobre el funcionamiento de los EAs en concreto.

1.2. Objetivos del Proyecto

Como principal objetivo, se propone el estudio de distintos criterios de modificación de la ratio de un EA celular. En particular, se plantea la investigación de los siguientes puntos de interés:

- Estudio de los resultados existentes relacionados, operadores y algoritmos usados para resolver problemas similares a los ya tratados en [AT00].
- Estudio de la influencia de la forma de la cuadrícula sobre la velocidad de seguimiento del óptimo para varios problemas.
- Propuesta de criterios de detección de estancamiento de la búsqueda. Análisis de las acciones que tomar según estos criterios y de la forma en que hacerlo durante la búsqueda. Estos criterios girarán principalmente en torno a la velocidad de avance de la adecuación media y de la velocidad de descenso de la entropía en la población.
- Análisis de resultados de los distintos criterios por problema.
- Implementación del proyecto en forma de biblioteca C++ ([Stro91] y [Schi99]) para su posterior reutilización y comparación con software similar.

1.3. Fases en la Realización del Proyecto

La realización de este proyecto consta de las siguientes fases:

1. Estudio de los EAs y sus variantes celulares.

2. Estudio de las características de los problemas ya abordados y de los nuevos definidos en el proyecto.

3. Implementación de un sistema que permita fácilmente alcanzar los objetivos propuestos.

4. Estudios preliminares sobre simplificaciones de los objetivos del proyecto.

5. Análisis detallado de técnicas y problemas. Extracción de conclusiones.

Por tanto, la primera tarea que realizaremos será el estudio de todo lo relacionado don algoritmos evolutivos y, en particular, con los celulares; este estudio comprenderá la búsqueda y lectura de artículos científicos publicados sobre el tema, páginas web, libros, etc. Tras esto, el siguiente paso natural consistirá en el estudio de una serie de problemas que se han propuesto para evaluar el comportamiento del algoritmo genético utilizado, además de su implementación en forma de librerías implementadas en C++ y que complementen las ya disponibles en la implementación con la que trabajaremos. Tras esto nos resta la ejecución de los diferentes problemas y la extracción de conclusiones a partir de estos resultados.

1.4. Organización de la Memoria

El documento que aquí se presenta se ha estructurado en 9 capítulos. El *Capítulo 1* es una introducción al trabajo realizado durante la vida del proyecto.

En el *Capítulo 2* se realiza una breve descripción teórica de la Computación Evolutiva en general, haciendo especial hincapié en el caso de los Algoritmos Genéticos Celulares (cGAs), que es el tipo de algoritmo con el que se ha trabajado mayoritariamente en el proyecto.

En el *Capítulo 3* se explica en qué consiste cada uno de los problemas que serán objeto de estudio en este trabajo.

La representación que utilizaremos para estos problemas, junto con sus funciones de evaluación y los esquemas de correspondientes a sus soluciones se describen en el *Capítulo 4*.

Los primeros resultados aparecen en el *Capítulo 5*; en él se detallan los resultados obtenidos en el estudio de los problemas utilizando tanto rejillas fijas como rejillas dinámicas con cambio preprogramado, estudiando los resultados desde dos puntos de vista diferentes: el comportamiento de cada problema con las distintas rejillas y el comportamiento de cada rejilla con los distintos problemas.

El objetivo de los *capítulos 6 y 7* es obtener un único criterio que se pueda aplicar a los problemas, tanto utilizando el valor de adecuación de los individuos como el de la entropía de la población, obteniendo buenos resultados. En el primero de estos dos capítulos se describen cuatro criterios auto-adaptativos, todos ellos basados en el valor de adecuación de los individuos; a estos criterios, ampliados también al caso del uso de la entropía, se les realiza un análisis comparativo en el siguiente capítulo, decidiendo cuales de ellos se utilizarán para estudiar el comportamiento de los problemas con estos criterios.

En el *Capítulo 8* se estudian los resultados proporcionados por el mejor de los criterios (obtenidos del estudio de los capítulos 6 y 7) aplicado a la entropía y al valor de adecuación, y empezando por la rejilla más rectangular y por la de ratio más intermedio. En total 4 criterios, que se compararán entre ellos y con otro criterio adicional que introducimos, mezcla de los criterios basados en la entropía y en el valor medio de adecuación y que comienza su ejecución utilizando la rejilla de ratio más intermedio.

Por último, en el *Capítulo 9* se establecen de manera resumida las conclusiones que se desprenden del estudio realizado y las propuestas para futuras investigaciones.

En los apéndices de A a H aparecen las tablas con los resultados obtenidos para todos los problemas con todos los tipos de rejilla y criterios utilizados durante el desarrollo del problema. También se encuentran las tablas correspondientes a pruebas adicionales que han sido necesario realizar durante el trabajo, como el estudio comparativo de diferentes criterios, con el fin de elegir el más apropiado para el posterior estudio de los problemas, o el estudio previo de diferentes valores de ε para tratar de descubrir los más representativos para utilizar después en los problemas. Tras los apéndices se muestra una lista con las principales referencias utilizadas durante la realización del proyecto.

Capítulo 2

Introducción a la Computación Evolutiva

Hace unos treinta años, varios investigadores coincidieron en desarrollar, independientemente cada uno, la idea de utilizar algoritmos basados en el modelo de la evolución orgánica como un intento de resolver las tareas de adaptación y optimización duras sobre computadores. Hoy en día, debido a su gran robustez y amplia aplicabilidad, y también a la disposición de gran potencia computacional, e incluso de hardware paralelo, el campo de investigación resultante, el de la computación evolutiva, recibe firmemente una creciente atención por parte de investigadores de muchas disciplinas.

2.1. Computación Evolutiva

El marco de la Computación Evolutiva [BFM97] constituye un enfoque para la resolución de problemas de búsqueda de valores óptimos utilizando modelos computacionales basados en procesos evolutivos.

Las técnicas de Computación Evolutiva se basan en hechos observados en la naturaleza, en la que los individuos adquieren características propias y habilidades sin apenas ser conscientes de ello a través del mecanismo de la evolución natural. Este mecanismo se da en casi todos los organismos a través de dos procesos primarios:

- **Selección Natural:** Determina cuáles serán los organismos que se reproducirán. Si un sujeto no muestra idoneidad morirá sin reproducirse.

- **Reproducción:** Mediante la reproducción se garantiza la descendencia del sujeto, además de la mezcla y recombinación de los genes de los sujetos que se reproducen (que, teóricamente son los mejores individuos).

Las implementaciones informáticas que se rigen por estos modelos se conocen como *algoritmos evolutivos* [BH90] y su propósito genérico consiste en guiar una búsqueda estocástica haciendo evolucionar a un conjunto de estructuras y seleccionando de modo iterativo las más adecuadas. De manera que se puede decir que los EAs están diseñados de forma que imitan los procesos de evolución y selección presentes en la naturaleza, mecanismos que fomentan la aparición de estructuras orgánicas complejas y bien adaptadas a su entorno.

No es extraño utilizar modelos presentes en la naturaleza para resolver problemas en el entorno informático; prueba de ello son la existencia de técnicas como el Recocido Simulado o *Simulated Annealing* [LA87], que se basa en el proceso físico de enfriamiento de un sólido, en el que se van reorganizando sus partículas hasta conseguir llegar a estados de máxima estabilidad, o las Redes Neuronales [MR86], en las que se trata de imitar el comportamiento de las neuronas biológicas, responsables de la conducta inteligente.

La computación evolutiva destaca por realizar un tratamiento numérico de la información, a diferencia de otros dominios como la *Inteligencia Artificial (AI)*. Podemos encuadrar a los EAs como una subclase de las técnicas conocidas como *Soft Computing* o Inteligencia Computacional [Zade94]; según esta clasificación, la Computación Evolutiva, junto a las Redes de Neuronas Artificiales y la Lógica Difusa o *Fuzzy Logic* [Zade65] forman la tripleta de campos de trabajo, junto con algunas otras técnicas novedosas, donde se busca la solución de problemas a través de técnicas algorítmicas basadas en la representación numérica del conocimiento.

2.2. Algoritmos Evolutivos

Como ya se dijo en la sección anterior, en la naturaleza existen infinitud de organismos adaptados a la supervivencia prácticamente en cualquier ecosistema (o, lo que es lo mismo, en casi cualesquiera condiciones). Esto sorprende si consideramos que el medioambiente está en continua evolución y sufriendo cambios constantemente (incluso por la actuación de factores externos a él), lo que propicia la extinción de algunas especies y la evolución de otras, que se adaptan a la nueva situación. Además, es de especial interés observar que esta evolución no se realiza conscientemente por parte de los individuos. Los EAs surgen para tratar de imitar este fenómeno, en el que los

individuos que forman la población evolucionan sin ser conscientes de ello hacia sujetos mejores. Además, son capaces de adaptarse a las nuevas condiciones que se puedan dar debido a posibles cambios del ecosistema en el que se encuentran [BRYS99].

En el año 1995, Jones definió a los EAs de la forma siguiente [Jone95]: *Los algoritmos mantienen un conjunto de soluciones potenciales a un problema. Dichas soluciones son utilizadas para producir nuevas soluciones potenciales mediante la aplicación de una serie de operadores. Dichos operadores actúan sobre algunas soluciones que han sido seleccionadas por su bondad con respecto al problema atacado. Este proceso se repite hasta que se alcanza un cierto criterio de terminación.*

En este capítulo se presentará brevemente la visión clásica de los EAs, partiendo de una breve reseña sobre los fundamentos biológicos, mostrando más tarde una descripción cualitativa de los EAs y las familias en las que se han concentrado su estudio. De entre estas familias se estudiará más a fondo a los *algoritmos genéticos* y, dentro de estos, a los *algoritmos genéticos celulares*.

2.2.1. Bases Biológicas de los Algoritmos Evolutivos

Como ya se explicó en la Sección 2.1, los dos procesos primarios en los que se basan los EAs son la *Selección Natural* y la *Reproducción*. El primero de estos procesos se corresponde con los estudios desarrollados por Darwing en su *Teoría de la Selección Natural* [Darw59], mientras que el segundo se puede encontrar estudiado en los trabajos de Mendel sobre *Herencia Genética* [Mend65]. De ambos trabajos se pueden extraer una serie de puntos que nos serán muy útiles si los tenemos en cuenta:

- La evolución es un proceso que no opera sobre organismos, sino que actúa sobre sus cromosomas, que son unos instrumentos orgánicos que contienen codificada toda la información referente al individuo. Estos cromosomas y, por tanto, la información que contienen, son heredados por individuos de otra generación mediante los procesos reproductivos.

- Durante la etapa de reproducción se da el proceso evolutivo; cuando se genera un individuo nuevo, éste hereda la información de los cromosomas de sus progenitores mediante *recombinación*. Asimismo, también se pueden dar en este proceso casos de mutación de genes, que propician la variabilidad del material genético de los individuos, y que pueden ser determinantes de la adaptación del individuo ante posibles cambios del (o en el) entorno en el que se encuentre.

- Mediante la selección natural se permite que los mejores individuos (aquellos con información en su cromosoma más adecuada para el medio en el que se encuentran) tengan más facilidad

para reproducirse que aquellos individuos que están menos adaptados al medio; propiciando, de esta manera, la extinción de estos últimos.

Estudiando estos puntos parece lógica la existencia de multitud de investigaciones que traten de trasladar estas características al campo de la algoritmia, intentando conseguir sistemas con comportamientos similares. Profundizando un poco en esta clase de estudios, nos podemos dar cuenta de que los procesos que se dan en la naturaleza no persiguen ningún fin, son simplemente procesos en los que los individuos van interaccionando, reaccionando, entre sí, mientras que los sistemas desarrollados por el hombre sí que van persiguiendo un objetivo final. Además, es deseable que este objetivo se encuentre de la manera más rápida y eficiente posible. Por tanto, podemos pensar en construir sistemas inspirados en la evolución natural intentando reproducir lo más fielmente posible los principios naturales (se da en el área de la *Vida Artificial* [Lang89]), o en utilizar dichos principios como inspiración, de manera que los modifiquemos de la manera precisa para obtener sistemas eficientes en el desempeño de la tarea deseada. Este último enfoque es mucho más práctico que el primero en el sentido de que nos permite encauzar el desarrollo de la evolución hacia generaciones siempre mejores (o iguales) que las anteriores. En este enfoque es en el que debemos buscar el origen de los EAs.

2.2.2. Conceptos Generales

Los EAs son útiles para la solución de problemas NP[1] [GJ79], ya que trabajan rápidamente en grandes espacios de búsqueda gracias a que son capaces de operar sobre varias soluciones simultáneamente. Para resolver un problema NP las únicas técnicas seguras consisten en ir evaluando exhaustivamente todas las posibilidades hasta encontrar una solución.

Las principales características de los EAs se pueden resumir en que:

- Buscan a partir de una población de distintas soluciones.
- Utilizan una función de adaptación para realizar la búsqueda.
- Utilizan reglas de transición probabilísticas, no reglas deterministas.
- No necesitan grandes requisitos matemáticos para la solución de problemas de interés.
- Se trata de EAs que pueden manejar cualquier tipo de funciones objetivo, y cualquier tipo de restricciones definidas en espacios de búsqueda discretos, continuos o mixtos.
- Son algoritmos de funcionamiento intrínsecamente paralelo, con todas las ventajas que ello supone, como la ejecución multiprocesador o multicomputador.

[1] Un problema NP es aquel en el que no es posible asegurar que se pueda encontrar una solución en tiempo polinomial

- Tienen mecanismos para disminuir la posibilidad de caer en máximos locales.
- Las soluciones encontradas dependen de unos valores aleatorios que se generan al comenzar.
- Introducen un cierto grado de aleatoriedad en la búsqueda de la solución.
- No siguen ningún camino de búsqueda predeterminado por el espacio de soluciones.

Los principales conceptos que utilizaremos al hablar de EAs se explican a continuación:

- **Cromosoma**: Se trata de una cadena finita de valores definidos en un rango determinado cada uno. Contiene información codificada de una posible solución.
- **Gen**: Es cada uno de los elementos que forman la cadena del cromosoma.
- **Alelo**: Es cada uno de los valores (uno o más) del gen.
- **Individuo** o **genotipo**: Cada individuo (ver Figura 2.1) contendrá una posible solución al problema. Está formado por uno o más cromosomas.
- **Valor de adecuación** o **fitness**: Valor numérico que se asigna a cada individuo para poder utilizar un criterio capaz de evaluar la proximidad del mismo a una solución óptima.
- **Dominancia**: Función definida para devolver un único cromosoma a partir de todos los cromosomas que contiene el individuo. Es necesaria en poblaciones en las que los individuos están formados por dos o más cromosomas, cuya aplicación es previa para poder obtener así el valor de adecuación de los individuos.
- **Población** o **fenotipo**: Se trata de una expresión (decodificación) del genotipo en el dominio del problema.
- **Mutación**: Consiste en el cambio del valor de un alelo por otro valor aleatoriamente elegido.
- **Vecindario**: Conjunto de individuos vecinos a uno dado, es decir, que están situados próximos a él en la población según una topología espacial dada.
- **Reproducción**: La reproducción consiste en elegir varios individuos para generar descendientes a partir de ellos.
- **Convergencia**: Este término fue introducido por De Jong en su tesis doctoral [Jong75]. La población converge si el 95 % de los individuos que la forman comparten el mismo valor para cada uno de sus genes.

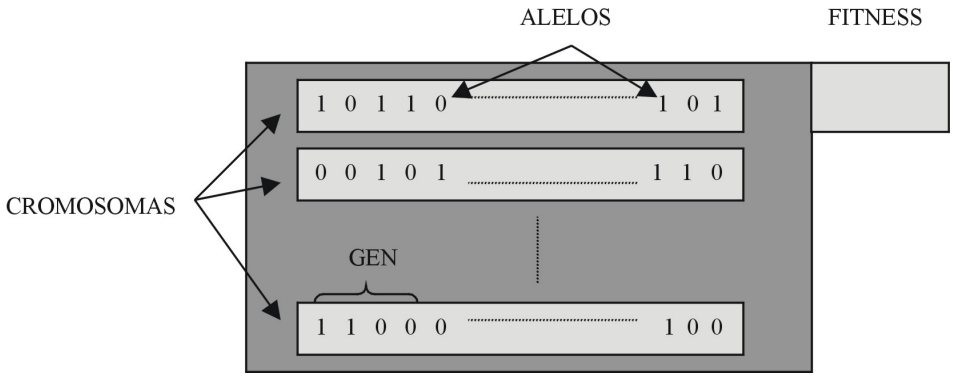

Figura 2.1: Estructura genética de un individuo típico en EAs

2.2.3. Estructura de un Algoritmo Evolutivo

Un EA es un proceso iterativo que opera sobre un conjunto P de individuos que forman una población, cada uno de los cuales tiene uno o más cromosomas almacenando toda su información; de manera que este(os) cromosoma(s) pueda(n) utilizarse como representación de una posible solución al problema considerado. Por tanto, cada individuo se convierte en una potencial solución al problema tratado. Mediante un proceso adicional de codificación/decodificación (ρ) podemos obtener la información que se codifica en los cromosomas que los individuos contienen. Esta población puede ser generada inicialmente de forma aleatoria o bien utilizando algún heurístico como ayuda.

El EA cuenta con una función (*función de adecuación* o *función de fitness*) para asignar a cada individuo de la población un *valor de adecuación* (o *valor de fitness*), que es un valor que depende de la información contenida en los cromosomas de cada individuo. Este valor de *fitness* da una idea comparativa y cuantitativa de lo mejor o peor que es un individuo en relación a los demás. Este es el valor por el que el EA regirá su estrategia de búsqueda.

A grandes rasgos, el funcionamiento de un EA se podría explicar como sigue: Partimos de un conjunto de individuos -o genotipos ($\mathcal{G}$)- que forman la población, también llamada fenotipo ($\mathcal{F}$). A partir del fenotipo obtenemos el conjunto de cromosomas de todos los individuos que lo forman. Sobre los genotipos que componen la población se realizan las operaciones de selección (σ), reproducción (ω_x), mutación (ω_m) y reemplazo (ψ), que se van realizando de forma repetitiva en cada una de las iteraciones hasta que se cumple un cierto criterio de terminación.

Mediante estas operaciones se promociona que los individuos más aptos de la población se irán reproduciendo y, por tanto, irán transmitiendo su información, con mayor facilidad que los menos aptos, de manera que la población irá evolucionando globalmente hacia individuos más aptos. El

operador de mutación emula a la mutación genética introduciendo un cierto grado de aleatoriedad para evitar que el algoritmo caiga en soluciones locales.

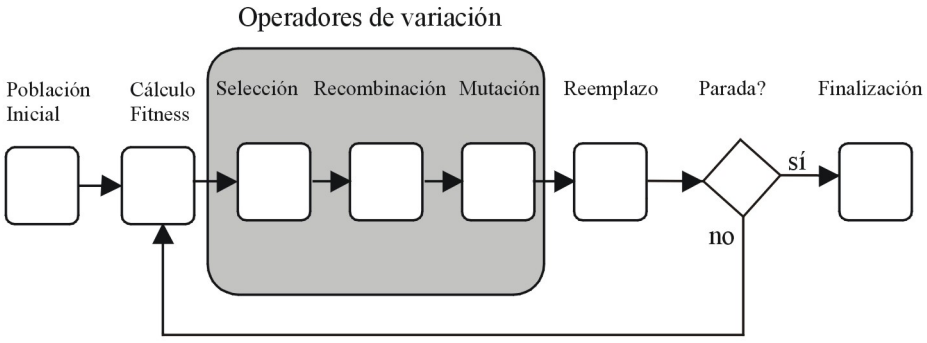

Figura 2.2: Estructura general de un EA

De esta manera conseguimos integrar en nuestro modelo los principios de Selección Natural y Herencia Genética de los que se habló en la Sección 2.2.1.

Como vemos en la Figura 2.2, el procedimiento estándar que sigue un EA durante su ejecución se rige por los siguientes pasos:

- Genera una población inicial de individuos de forma aleatoria.
- Asigna un valor a cada individuo generado mediante una función objetivo para el cálculo del *fitness*, con el fin de determinar la proximidad o lejanía de dicha solución con respecto al óptimo.
- Elige los individuos progenitores de acuerdo a su valor de adecuación mediante algún mecanismo de selección definido para el problema.
- Los individuos progenitores se recombinan o se mutan para formar los individuos de la nueva generación.
- Estos nuevos individuos se insertan en la población, bien sustituyendo la anterior población por completo o bien conviviendo con sus progenitores.

Este procedimiento, que se corresponde con la creación de una nueva generación, se repite hasta que se cumpla un cierto criterio de parada. Podemos ver el pseudocódigo correspondiente a un EA en el Algoritmo 1.

Algoritmo 1 Pseudocódigo de un Algoritmo Evolutivo

1: $t \leftarrow 0$; {Contador de generaciones}
2: $P[0] \leftarrow INICIALIZAR\_POBLACION()$;
3: **while** $NOTCRITERIO\_FINALIZACION(P[t])$ **do**
4: $EVALUAR\_POBLACION(P[t])$; {Evaluación de la población inicial}
5: $P'[t] \leftarrow SELECCIONAR\_PROGENITORES(P[t])$;
6: $P''[t] \leftarrow APLICAR\_OPERADORES\_REPRODUCCION(P'[t])$; {Nuevas soluciones}
7: $P[t+1] \leftarrow REEMPLAZAR(P[t], P''[t])$; {Población de la generación siguiente}
8: $t \leftarrow t+1$;
9: **end while**
10: $SOL \leftarrow MEJOR(P[t]), \forall t$

2.2.4. Tipos de Algoritmos Evolutivos

De entre las familias más importantes que han surgido del estudio de los EAs podemos destacar a los GAs [Mich92], los Sistemas Clasificadores [Gold89], las Estrategias de Evolución [Rech73], la Programación Evolutiva [Mich92] y el paradigma de Programación Genética [Koza92]. Estos sistemas se diferencian principalmente en la estructura de los individuos y en el tipo de tratamiento al que se les somete, que ejerce una influencia directa sobre el funcionamiento de los operadores de recombinación y mutación.

2.3. Algoritmos Genéticos

Los GAs, desarrollados principalmente por Holland durante los años 60, son algoritmos de búsqueda aleatoria basados en el modelo de la evolución biológica ([Gold89], [Holl75]). Holland llamó a esta técnica *planes reproductivos*, pero tras la publicación de su libro en el año 1975 [Holl75] se hizo popular con el nombre de *algoritmo genético (GA)*.

Según la definición dada, fácilmente podremos deducir que forman parte del campo de la computación evolutiva. Estos algoritmos tienen en cuenta el proceso de aprendizaje colectivo en una población de individuos, cada uno de los cuales representa un punto de búsqueda en el espacio de soluciones potenciales de un problema de optimización dado. La población evoluciona hacia regiones del espacio de búsqueda cada vez mejores por medio de procesos aleatorios como son la selección, la mutación y la recombinación. El mecanismo de selección favorece que, cuando se vaya a crear la nueva población, los individuos con mayor valor de adecuación (los más próximos a la solución) se reproduzcan con más frecuencia que los que cuentan con menores valores de *fitness*.

La recombinación permite la mezcla de la información de los padres cuando se le pasa a sus descendientes; mientras que la mutación introduce un cierto factor de innovación en la información de los individuos de la población. Normalmente, la población inicial se inicializa con valores aleatorios; además, el proceso de evolución se detiene tras haber realizado un número predefinido de iteraciones. Con esta descripción informal podemos obtener una primera aproximación a un GA simple, que se representa en el Algoritmo 2.

Algoritmo 2 Pseudocódigo de un Algoritmo Genético Simple

1: $t \leftarrow 1$; {Contador de generaciones}
2: $P[0] \leftarrow INICIALIZAR\_POBLACION()$;
3: $FITNESS\_P[0] \leftarrow EVALUAR\_POBLACION(P[0])$; {Evaluación de la población inicial}
4: **repeat**
5: $\quad Q[t] \leftarrow SELECCIONAR\_PROGENITORES(P[t-1])$;
6: $\quad Q[t] \leftarrow REPRODUCIR\_PROGENITORES(Q[t])$; {Creación de nuevas soluciones}
7: $\quad Q[t] \leftarrow MUTAR(Q[t])$; {Mutación de las nuevas soluciones}
8: $\quad FITNESS\_Q[t] \leftarrow EVALUAR\_POBLACION(Q[t])$; {Evaluación de nuevas sols}
9: $\quad P[t] \leftarrow REEMPLAZAR(P[t-1], Q[t], FITNESS\_P[t-1], FITNESS\_Q[t])$;
10: $\quad FITNESS\_P[t] \leftarrow EVALUAR\_POBLACION(P[t])$;
11: $\quad t \leftarrow t+1$;
12: **until** $CRITERIO\_FINALIZACION(P[t])$
13: $SOL \leftarrow MEJOR(P[t]), \forall t$

John Koza propone la siguiente definición de GA [Koza92]: *Es un algoritmo matemático altamente paralelo que transforma un conjunto de objetos matemáticos individuales con respecto al tiempo usando operaciones modeladas de acuerdo al principio darwiniano de reproducción y supervivencia del más apto, y tras haberse presentado de forma natural una serie de operaciones genéticas de entre las que destaca la recombinación sexual. Cada uno de estos objetos matemáticos suele ser una cadena de caracteres (letras o números) de longitud fija que se ajusta al modelo de las cadenas de cromosomas, y se les asocia con una cierta función matemática que refleja su aptitud.*

La aplicación más común de los GAs es la de resolver problemas de optimización; pero no todos los problemas pueden ser apropiados para ser resueltos mediante esta técnica. Para que un problema pueda ser resuelto utilizando GAs debe cumplir una serie de características, de entre las que destacaremos:

- El espacio de posibles soluciones del problema (su espacio de búsqueda) debe estar delimitado dentro de un cierto rango.

- Debe poderse definir una función de evaluación que nos indique el grado de validez de una posible solución dentro de un rango numérico.
- Las soluciones deben poder codificarse de una forma que resulte aceptable de implementar.

La función de aptitud se corresponde con la función objetivo o función de *fitness* del problema de optimización. El GA sólo es capaz de maximizar; por tanto, para el caso en que necesitemos minimizar una función debemos usar el recíproco de la función maximizada. Una buena función de adecuación debe ser capaz de fomentar la reproducción de las buenas soluciones premiándolas para que sean elegidas más fácilmente durante la reproducción y penalizando las malas para que se extingan.

El proceso de evaluación calcula el valor de adecuación para cada individuo de la población. Los GAs tienen un especial interés porque han demostrado una aplicabilidad ampliamente satisfactoria en un gran número de aplicaciones prácticas. A continuación realizaremos un estudio más profundo sobre este tipo de algoritmos.

2.3.1. Especificación Formal de un Algoritmo Genético

Un GA puede representarse formalmente mediante la siguiente tupla de ocho elementos [Holl75]:

$$GA = (P^i, \mu, l, s, \rho, \omega, f, t), \tag{2.1}$$

Donde:

- Tamaño de la población: $\mu \in \mathcal{N}$
- Longitud del cromosoma del individuo: $l \in \mathcal{N}$
- Población en la generación i: $P^i = (a_1^i, \ldots, a_\mu^i) \in I^\mu; I = 0, 1^l$
- Operador de selección: $s : I^\mu \to I^\mu$
- Función de determinación del operador: $\rho : I \to \Omega$
- Conjunto de operadores genéticos: $\Omega \subseteq \omega : I \times I^\mu \to P \to I^2$
- Función de evaluación: $f : I \to \mathcal{R}$
- Criterio de terminación: $t : I^\mu \to 0, 1$

[2] $P = p : I \to [0, 1]$representa el conjunto de las distribuciones de probabilidad sobre I

A esto habría que añadir la posibilidad de que la función ρ pueda ser el resultado de varios operadores genéticos [BH90]; por ejemplo, al individuo a_i^1, obtenido de la generación inicial por 'crossover', puede sufrir una mutación, siendo entonces su operador genético:

$$\omega_i^1 = \omega_m \circ \omega_c, \tag{2.2}$$

Por tanto, describiremos la transición de una generación t a otra $t+1$ de la siguiente manera:

$$\begin{aligned} s(P^t) &= (P^t)' && (2.3)\\ &= ((a_1^t)', \ldots, (a_\mu^t)')\rho((a_i^t)') \\ &= \overline{\omega}_i^t \, \forall i \in 1, \ldots, \mu \; a_i^{t+1} \\ &= \overline{\omega}_i^t((a_1^t)', P^t), \forall i \in 1, \ldots, \mu && (2.4)\\ P^{t+1} &= ((a_1^{t+1})', \ldots, (a_\mu^{t+1})') && (2.5) \end{aligned}$$

2.3.2. Tamaño de la Población

Uno de los parámetros cuyo valor hay que plantearse al diseñar el GA es el relacionado con el tamaño de la población. Es intuitivo pensar que el uso de poblaciones pequeñas nos hace correr el riesgo de no poder cubrir adecuadamente el espacio de búsqueda en la generación inicial de soluciones aleatorias; mientras que el uso de poblaciones demasiado grandes nos puede acarrear problemas relacionados con el excesivo coste computacional.

En [GR87] se realiza un estudio teórico sobre el tamaño de la población, obteniéndose la conclusión de que el tamaño óptimo de la población para cadenas de una determinada longitud, con codificación binaria, crece exponencialmente con el tamaño de la cadena; lo que quiere decir que cuanto más larga sea la longitud del cromosoma de los individuos con más lentitud convergirá el GA y, por tanto, se precisará una población mayor.

2.3.3. Población Inicial

La población inicial se suele crear generando cadenas aleatorias para representar la información de los individuos. Hay muy pocos trabajos en los que se hayan realizado estudios acerca de la posibilidad de generar la población inicial mediante algún tipo de regla heurística. Los resultados existentes reflejan que esta técnica puede acelerar la convergencia del GA. El problema es que en muchos casos esta convergencia es tan rápida que supone una desventaja, ya que propiciaría la convergencia hacia óptimos locales.

2.3.4. Generación

Cada generación está representada por la población en un tiempo determinado. Durante la ejecución de un GA se crea un número determinado de generaciones como máximo, número que se corresponde con uno de los parámetros del algoritmo. Este parámetro sirve para detener el algoritmo cuando no pueda converger a una solución, y su valor se suele determinar de forma empírica.

2.3.5. Función de Evaluación

Dos factores importantes en el comportamiento de un GA son la función de evaluación (o de adecuación) utilizada y la codificación elegida para la solución (Sección 2.3.7).

El propósito de la función de evaluación consiste en asignar un valor de fitness o aptitud a los individuos generados por el algoritmo. Dicho valor nos indica el grado de adaptación al medio, que no es más que la calidad de la solución representada. En los casos más triviales, la función de evaluación puede coincidir exactamente con la función objetivo que se plantea optimizar como resultado de resolver el problema. Nos interesa construir funciones de evaluación que verifiquen que para dos individuos que se encuentren cercanos en el espacio de búsqueda, sus respectivos valores en las funciones objetivo estén próximos.

Una dificultad con la que nos podemos encontrar al trabajar con EAs es la existencia de gran cantidad de óptimos locales en el espacio de búsqueda generado por la función de adecuación, o también con la posibilidad de que el óptimo global esté muy aislado.

Al diseñar una función de evaluación debemos tener en cuenta que no todos los individuos tienen por qué representar soluciones factibles; nuestra función de evaluación debe ser capaz de penalizar a aquellos individuos que representen soluciones no factibles. Al encontrarnos en este caso podemos tomar dos alternativas, como son reparar las soluciones para hacerlas factibles o que las representaciones y los operadores siempre generen soluciones factibles.

2.3.6. Selección

Un aspecto muy importante en el diseño de GAs es la especificación de cómo seleccionar a los individuos a la hora de la reproducción. El propósito de la selección consiste en dar mayor probabilidad de ser elegidos a los individuos cuyos valores de adecuación sean más altos, con la esperanza de que sus descendientes puedan incrementar este valor. Las principales funciones de

selección que se conocen se detallan a continuación [JS97]:

- *Ruleta*, según la cual cada individuo tiene una mayor probabilidad de ser seleccionado cuanto mayor sea su valor de adecuación.
- *Selección aleatoria*, en la que todos los individuos tienen la misma probabilidad de ser seleccionados (independientemente de su valor de adecuación).
- *Selección por torneo*, en la que se toman dos individuos y se elige el que tenga mayor valor de adecuación.

2.3.7. Codificación

Un GA trabaja con individuos que representan soluciones potenciales al problema. Lo primero que debemos decidir a la hora de diseñar un GA es una codificación del espacio de soluciones. En problemas de optimización combinatoria es usual codificar la solución como una permutación de valores donde lo que importa es el orden de ciertos elementos. Estos problemas se les conoce a menudo como *problemas basados en el orden*.

Debe notarse que este tipo de codificación representa el valor de una posible solución asignada al individuo.

2.3.8. Operador de Recombinación

Un operador de recombinación crea uno o más descendientes con información genética obtenida a partir de dos padres[3]. El propósito de la recombinación consiste en la combinación de posiciones de individuos diferentes para formar nuevos; obviamente, es deseable que los descendientes tengan características que los hagan mejores que sus progenitores. Existe un parámetro, la tasa de recombinación (ω_c), que indica la probabilidad de reproducirse para el operador. Una elección típica del valor de este parámetro es $\omega_c = 0.6$ [Jong75] o superior. La recombinación funciona eligiendo dos individuos de la población como padres $\overrightarrow{x} = (x_1, \ldots, x_l)$ y $\overrightarrow{y} = (y_1, \ldots, y_l)$, escogiendo un punto de cruce $\chi \in 1, \ldots, l-1$ de forma aleatoria, e intercambiando entre los dos individuos todos los bits que aparecen tras la posición χ para formar dos individuos nuevos:

$$\overrightarrow{x}' = (x_1, \ldots, x_{\chi-1}, x_\chi, y_{\chi+1}, \ldots, y_l) \tag{2.6}$$

$$\overrightarrow{y}' = (y_1, \ldots, y_{\chi-1}, y_\chi, x_{\chi+1}, \ldots, x_l) \tag{2.7}$$

[3]El operador de recombinación no está limitado en este sentido y de manera general crea d descendientes a partir de p padres, aunque normalmente se cruza una pareja para producir uno o dos descendientes.

Este operador de recombinación de un punto ha servido de estándar durante muchos años. De cualquier forma, puede ser generalizado a m puntos intercambiando los valores de los cromosomas de los progenitores cada dos puntos alternativamente [Jong75].

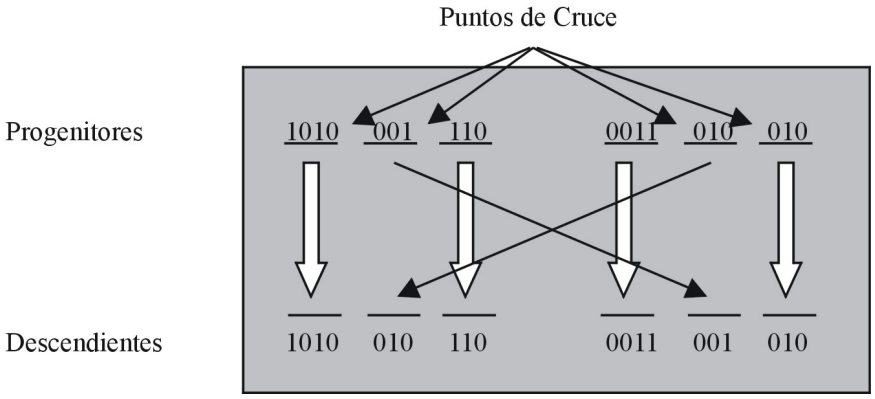

Figura 2.3: Operador de recombinación basado en dos puntos

2.3.9. Operador de Mutación

Otro operador de variación que se utiliza en GAs es la mutación. Siguiendo nuevamente una analogía con la Genética, un operador de mutación puede alterar parte de la cadena cromosómica de un individuo, modificando de manera aleatoria el valor de uno o más de sus genes[4]. En el contexto de los EAs, tradicionalmente se le asigna un papel muy importante a este operador. Produce diversidad en la población, evitando la convergencia en óptimos locales, permitiendo que se creen nuevas soluciones y aumentando, por consiguiente, la componente de exploración de la búsqueda.

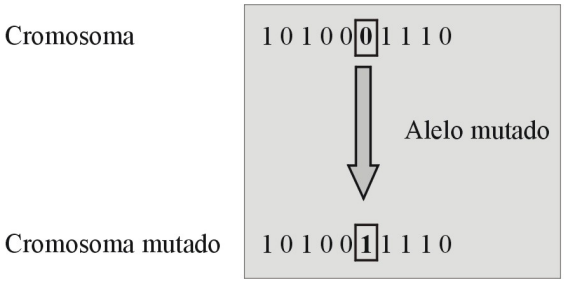

Figura 2.4: Operador de mutación sobre un cromosoma

[4]Un proceso de mutación puede crear ocasionalmente mejores o peores individuos. Los procesos de selección se encargarán de mantener las soluciones más aptas

2.3.10. Tipos de Algoritmos Genéticos

Entre los campos principales dentro de los GAs estructurados, las dos herramientas más ampliamente conocidas son los GAs distribuidos [AT99] y los GAs celulares [CTTS98]. La diferencia entre estos dos tipos de algoritmos radica en la forma en la que explotan a la población. En el caso de los algoritmos distributivos, la población se divide en subpoblaciones que son tratadas como islas independientes en cada una de las cuales se ejecuta una instancia distinta del algoritmo; estas instancias se comunican entre sí mediante unas cadenas de conexión. Por otro lado, en el caso de los algoritmos celulares, una única instancia del GA opera sobre la población en forma de vecindarios; es decir, para cada individuo, realiza las operaciones correspondientes entre él y sus vecinos. En la Figura 2.5 podemos ver tres tipos de GAs: el panmíctico (ver Algoritmo 2 en la Sección 2.3), el distribuido y el celular.

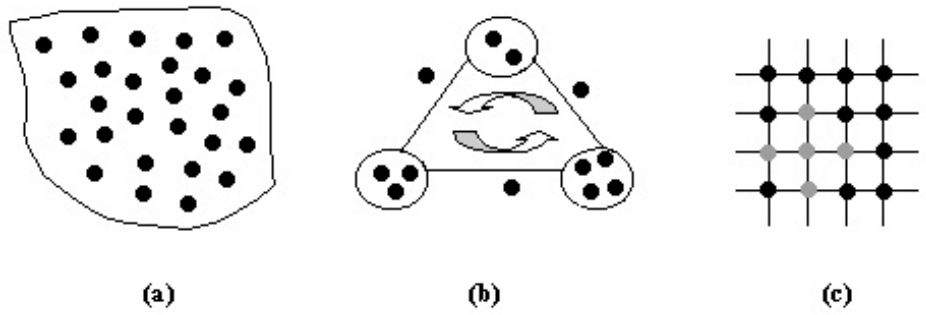

Figura 2.5: Distribución de población: GAs panmíctico(a), distribuido(b) y celular(c)

La ventaja entre el algoritmo celular y el panmíctico es que el celular no está centralizado, de manera que el ciclo reproductivo se realiza sobre cada individuo y sus vecinos; como, a su vez, dicho individuo pertenece a varios vecindarios, esto permite que se produzca poco a poco una difusión de las mejores soluciones a través de la población.

2.4. Algoritmos Genéticos Celulares

Éste es el tipo de algoritmos con el que hemos trabajado en este trabajo. Los algoritmos celulares tienen gran importancia por varias razones:

- Están sumergidos en una estructura espacial interna que permite que la diversidad del valor de adecuación y del genotipo perdure durante un mayor número de ciclos.
- Algunos trabajos establecen las ventajas de utilizar EAs celulares para tareas complejas de optimización (alta eficacia y un reducido número de pasos) en relación a otros EAs [Balu93] [MSB91] [AT00].

- La similitud entre GAs y autómatas celulares [Toma93] [Whit93], sus aplicaciones potenciales, y su posibilidad de ser implementados en máquinas SIMD[5] los hacen dignos de estudio.

2.4.1. Caracterización de un Algoritmo Genético Celular

El Algoritmo 3 muestra el código de un cGA. En él se observa el uso de operadores de selección aplicados a la vecindad. El código usa también otros operadores de variación como DPX1, operador de recombinación que crea un nuevo individuo con un cromosoma obtenido a partir de los padres, y un operador de mutación. Antes de ser insertado en la población, el algoritmo calcula el fitness o valor de aptitud del individuo.

Algoritmo 3 Pseudocódigo de un cGA simple

```
1:  for s ← 1 to MAX_STEPS do
2:    for x ← 1 to WIDTH do
3:      for y ← 1 to HEIGHT do
4:        n_list ← COMPUTE_NEIGH(POS(x,y)); {Vecindario del individuo en (x.y)}
5:        parent1 ← LOCAL_SELECT(n_list); {Aplica selección descentralizada}
6:        parent2 ← LOCAL_SELECT(n_list);
7:        REC(Pc,n_list[parent1],n_list[parent2],aux_ind); {Recombinación de 1 solo punto}
8:        MUTATE(Pm,aux_ind); {Aplica mutación con cierta probabilidad Pm}
9:        aux_ind.fit ← FIT(aux_ind); {Evalúa el nuevo individuo}
10:       INSERT_NEW_IND(POS(x,y),aux_ind,aux_pop); {Inserta nuevo ind. en pob. auxiliar}
11:       pop ← aux_pop; {Siguiente generación}
12:     end for
13:   end for
14: end for
```

La población se estructura en una rejilla (o *grid*) 2D con forma toroidal. Lo primero que hace el algoritmo es obtener el vecindario definido para el individuo que ocupa la posición (x, y) (línea 4); se seleccionan los padres de entre los vecinos de dicho individuo (líneas 5 y 6). En la línea 7 se realiza la operación de recombinación de los padres seleccionados previamente y se obtiene un único hijo[6]; posteriormente se le aplica una posible mutación (según un valor estadístico) a este hijo (línea 8). Después se calcula el valor de adecuación de este nuevo individuo (línea 9) y se inserta en una población auxiliar (línea 10) (puede insertarse siempre o sólo si tiene un valor de adecuación superior al del individuo en estudio -en caso contrario se insertaría dicho individuo-).

[5]SIMD (Single Instruction Multiple Data): Son máquinas capaces de procesar cada instrucción varias veces con datos distintos simultáneamente

[6]En este pseudocódigo se ha considerado un solo descendiente, pero, generalmente, se pueden obtener varios descendientes de la operación de recombinación

Por último, se sustituye la antigua población por la recientemente obtenida (línea 13).

2.4.2. Selección Descentralizada y Modelo de Vecindario

En secciones anteriores se sugirió el uso de un operador de selección descentralizada en un cGA o de grano fino, que actúe considerando únicamente los individuos vecinos a uno dado.

De manera general, el operador de selección es el encargado de guiar la búsqueda estocástica que realiza un GA. Al aplicarlo a un grupo de individuos, dicho operador devolverá uno de entre ellos que destaque sobre los demás. En tales condiciones, el operador seleccionará normalmente un individuo con un grado de adaptación, aptitud o fitness superior a los demás individuos de su vecindario.

En diversos estudios se han definido vecindarios de varios tamaños, como, por ejemplo el de 5 individuos, denominado comúnmente NEWS (North, East, West, South), que considera el individuo central y los inmediatamente superior, inferior, izquierdo y derecho; existen estudios sobre otros vecindarios de tamaños mayores [JS96] [JS97], como es el caso de 9, 13 o compactos de 25, el usar un vecindario de menor radio hace que las soluciones se extiendan más lentamente por la población, induciendo una menor presión selectiva global[7] y manteniendo mayor diversidad genética[8] que al usar vecindarios mayores.

2.4.3. La Ratio Vecindario/Población

Dos parámetros muy importantes para guiar la búsqueda en cGAs son la topología de la población y el vecindario definido en ella. Existen muchos artículos que han estudiado el comportamiento de GAs según los valores de estos parámetros [JS96], y también sobre la combinación de valores de ambos [AT00]. Siguiendo la línea trazada por este segundo tipo de estudios, intentaremos definir un parámetro cuyo valor varíe en función de la topología de la población y la definición del vecindario; manteniendo una relación biunívoca. Es decir, un determinado ratio se corresponde con una población y vecindario únicos, y viceversa.

A este parámetro, que define una función biunívoca con la relación entre la topología de la

[7]Por presión selectiva entendemos la proporción del mejor individuo presente en la población en la siguiente generación. Una presión selectiva alta indica que en pocas generaciones el mejor individuo se extenderá por toda la población

[8]La diversidad de genotipos hace referencia a la mayor o menor composición de individuos que representen soluciones diferentes en una población. Existen muchas formas de medirla, una de ellas puede ser la *entropía*, medida utilizada en teoría de la información. Mantener cierta diversidad de genotipos es fundamental en un algoritmo genético, para evitar quedar atrapado en un óptimo local

población y la forma del vecindario, se le conoce como *ratio*. En la sección 2.4.4 se comenta la importancia que tiene en el estudio de cGAs este nuevo parámetro.

Por tanto, la relación entre el vecindario y la rejilla de la población nos permite introducir la presión selectiva que deseemos. Como sabemos que la ratio es una relación entre ambas estructuras, nos damos cuenta de la importancia que tiene la ratio en el estudio de algoritmos genéticos, ya que modificando su valor podemos elegir entre ejercer una mayor o menor presión selectiva. Sobre el estudio de este comportamiento del cGA con distintos problemas versará el presente trabajo.

2.4.4. El Cambio Dinámico de la Ratio en un cGA

El cGA se comporta de manera muy distinta durante su ejecución en función de la ratio vecindario/población utilizada. En [AT00] podemos encontrar un completo estudio acerca del comportamiento del algoritmo genético en función de esta ratio. De este estudio se desprende que la reducción de la ratio significa reducir la intensidad de selección global en la población, promoviendo la exploración. De esta manera se espera una mayor diversidad presente en el algoritmo que mejore los resultados en problemas difíciles, como los multimodales o los epistáticos. Por otro lado, la búsqueda desarrollada dentro de cada vecindario está guiando hacia la explotación de la población por parte del algoritmo; por tanto, incrementando la ratio logramos ejercer una explotación local sobre los individuos de la población. La única manera de cambiar de la exploración a la explotación en un cGA durante la búsqueda con una complejidad mínima y sin introducir nada nuevo en el algoritmo consiste en cambiar la ratio existente entre la población y el vecindario.

2.5. Caracterización del cGA Utilizado

El cGA utilizado en este trabajo está implementado en la biblioteca desarrollada en C++ por el Dr. Enrique Alba durante su tesis doctoral [Alba99]. A continuación se describirán algunos detalles sobre los parámetros y operadores utilizados en esta implementación.

2.5.1. Población y Vecindario

Al crear la población inicial se asigna a cada individuo un valor aleatorio, que será considerado como una posible solución. Durante todo el trabajo se ha considerado que la población está formada por 400 individuos, que están dispuestos en una malla (o grid) toroidal. Se utilizarán vecindarios de 5 individuos (el individuo a estudiar y sus cuatro vecinos) que siguen el patrón de diseño NEWS

(North-East-West-South), ya explicado en la Sección 2.4.2.

La población se implementa como una única lista de μ (Número de individuos que forman la población) individuos $P = \{a_0, \ldots, a_{\mu-1}\}$, donde cada individuo a_i se aloja *virtualmente* en la posición (x,y) de una rejilla de tamaño $w \times h$, siendo:

$$X(j) = j \mod w \quad Y(j) = j \div w \quad h = \mu/w \quad j \in \{0, \ldots, \mu - 1\} \tag{2.8}$$

Por otro lado, el individuo que se encuentra en la posición (x, y) de la malla está almacenada en la lista de la población:

$$pos(x, y) = y * w + x \tag{2.9}$$

Por último, el vecindario NEWS se definirá de la siguiente manera:

$$v(j) = news(j) = \{n, e, w', s\} \tag{2.10}$$

$$\begin{array}{ll} n = pos(X(j), --_{mod}(Y(j), h)) & e = pos(++_{mod}(X(j), w), Y(j)) \\ w' = pos(--_{mod}(X(j), w), Y(j)) & s = pos(X(j), ++_{mod}(Y(j), h)) \end{array} \tag{2.11}$$

Las funciones incremento y decremento acotado utilizadas en las ecuaciones 2.11 se definen a continuación:

$$++_{mod}(i, k) = (i + 1) mod\, k \quad --_{mod}(i, k) = i > 0?(i - 1) : k \tag{2.12}$$

2.5.2. Operadores Genéticos

El operador de selección elige a dos padres distintos para limitar en lo posible las redundancias con un coste mínimo (casi nulo). La elección de los padres del individuo se realiza entre los vecinos (según la estructura NEWS) de dicho individuo.

El operador de recombinación utilizado es el conocido como *DPX (Distance Preserving Crossover)*, que trata de generar un hijo que esté a la misma distancia de sus dos padres. Su objetivo es intentar que las distancias entre hijo y padre 1, hijo y padre 2, y padre 1 y padre 2 sean iguales.

Por último, el operador de mutación cambia el valor de un alelo del individuo según un cierto valor probabilístico, llamado *probabilidad de mutación*.

El reemplazo elimina el individuo considerado sólo si el nuevo individuo obtenido tiene un valor de *fitness* mejor; en caso contrario se mantendrá el individuo antiguo en la población. De esta manera evitamos retroceder en nuestra búsqueda del óptimo global (solución) con un bajo coste.

2.5.3. Cambiando la Forma de la Rejilla

Como se explicó en la Sección 2.5.1, la población está formada por una única lista de individuos sobre la cual se implementan una serie de operaciones que nos permiten tratar dicha lista como una malla toroidal de individuos. Los individuos que forman dicha lista no cambian de posición nunca, en ella sólo pueden insertarse o eliminarse individuos debido a los efectos del operador de recombinación.

Por tanto, al cambiar la forma de la malla que organiza la disposición de población de $x \times y$ a $x' \times y'$, los x' primeros individuos de la lista formarán la primera fila de la malla, los siguientes x' formarán la segunda fila, y así hasta completar las y' filas. Mientras que antes del cambio de rejilla, la malla que forma la población tenía y filas la primera de las cuales estaba formada por los x primeros elementos de la lista, la segunda por los x siguientes, y así sucesivamente hasta completar la malla entera.

2.5.4. Definición de la Ratio

El concepto de ratio y su utilidad ya fue presentado con anterioridad en la Sección 2.4.3. Ahora pasaremos a dar la definición matemática de la ratio utilizada en el cGA utilizado. El ratio del vecindario con respecto a la población viene dado por la ecuación:

$$Ratio = \frac{Radio_{Vecindario}}{Radio_{Grid}}, \tag{2.13}$$

La definición que se utilizará para evaluar el radio, tanto el del grid como el del vecindario será [AT00]:

$$rad = \sqrt{\frac{\sum (x_i - \overline{x})^2 + \sum (y_i - \overline{y})^2}{n^*}}, \overline{x} = \frac{\sum_{i=1}^{n^*} x_i}{n^*}, \overline{y} = \frac{\sum_{i=1}^{n^*} y_i}{n^*} \tag{2.14}$$

La fórmula elegida para obtener el radio mide la dispersión de n^* patrones, y destaca por ofrecer siempre valores distintos para rejillas distintas. Ya que otras posibles medidas del radio como por ejemplo el radio del círculo más pequeño que comprenda el rectángulo que forma la rejilla. Según vemos en la Ecuación 2.14, como el vecindario que utilizaremos durante todo nuestro estudio es el mismo (NEWS), la única manera posible de modificar la ratio vecindario/población será modificar el radio de la población; de manera que cuanto más estrecha sea la rejilla, mayor será su radio y, por tanto, menor será la ratio.

2.5.5. Cambio Dinámico de la Ratio

El objetivo que perseguiremos durante nuestro estudio consistirá en ir modificando esta ratio para ir moviéndonos poco a poco hacia la exploración del algoritmo por toda la población o la

explotación exhaustiva de la misma, según sea necesario.

Existen tres posibilidades que se pueden utilizar para incrementar el ratio. éstas son: modificar el radio del vecindario, modificar el radio del grid o realizar ambas modificaciones. Aquí nos centraremos en la modificación del radio del grid, aunque será interesante estudiar la tercera opción más adelante.

Por tanto, para aumentar el ratio, nuestro objetivo será el de disminuir el radio del grid. Esto se conseguirá redistribuyendo en un nuevo grid de distinta forma al inicial los elementos que forman parte de la población; y, para disminuir dicho ratio, tendremos que aumentar el radio del grid.

Lo que queda por estudiar será cuándo (según sean rejillas fijas, dinámicas preprogramadas o dinámicas auto-adaptativas), y en base a qué parámetros, se debe realizar dicho cambio. De entre las posibles métricas en las que nos podemos apoyar para ver los efectos de la utilización de las diferentes rejillas, nos centraremos en la diversidad de genotipos y en el valor que la función de fitness asocia a cada individuo de la población.

Capítulo 3

Descripción de los Problemas Estudiados

A continuación se especifican los problemas seleccionados para el estudio que se ha realizado durante el proyecto. Entre estos problemas se han incluido los descritos en [AT00], que han sido resueltos utilizando el mismo GA y siguiendo los mismos principios que los aquí utilizados, pero sólo en los casos de rejillas fijas y dinámicas preprogramadas. En este trabajo se ampliará ese estudio al caso de las rejillas dinámicas auto-adaptativas y con todos los problemas descritos a continuación.

3.1. Problema del Diseño de Código Corrector de Errores (Error Code Design Problem - ECC)

El problema consiste en la construcción de un código binario asignando palabras de código a un alfabeto de forma que resulte mínima la longitud de los mensajes transmitidos, a la vez que se les intenta proveer de la máxima capacidad posible de detección y/o corrección de eventuales errores que puedan aparecer durante la transmisión del mensaje.

La resolución del problema entraña una gran dificultad en el sentido de que ambas restricciones (mínima longitud de las palabras y máxima capacidad de detección/corrección de errores) se contraponen, ya que cuanto mayor sea la capacidad de detección/corrección de errores mayor debe ser la redundancia de la información incluida en las palabras del código y, por tanto, mayor será la longitud de dichas palabras. Por tanto, se hace necesario establecer un compromiso entre la longitud de las palabras del código y la capacidad de detección/corrección de errores.

- Instancia del problema: Se utilizará una tripleta (n, M, d). El significado de los elementos que la componen se detalla a continuación:
  - n: Longitud de las palabras de código
  - M: Número de palabras de código
  - d: Mínima distancia de Hamming entre dos palabras cualesquiera del código.
- Solución Factible: Un conjunto de M palabras de código binario de longitud n distintas entre sí.
- Función objetivo: La suma de las distancias de Hamming entre las palabras que forman el código:

$$f(C) = \sum_{i=1}^{M} \sum_{\substack{j=1 \\ i \neq j}}^{M} \frac{1}{d_{ij}^2} \tag{3.1}$$

  siendo d_{ij} la distancia mínima distancia de Hamming entre las palabras i y j del código C.

- Solución óptima: El objetivo del problema consiste en obtener M palabras binarias de longitud n; con la propiedad de que la distancia de Hamming que exista entre ellas sea máxima, de forma que sea posible la detección o incluso la corrección de errores en el máximo número posible de bits de las palabras de código. Es decir, la solución óptima será aquella que minimice la ecuación 3.1.

Cuanto mayor sea el valor asignado a d, mayor será la tolerancia a errores del código resultante y , por tanto, mejor será la solución encontrada.

Es importante comprender que dado un M fijo, conforme se va haciendo mayor el valor de n, el espacio de búsqueda de las posibles palabras del código va creciendo exponencialmente.

Formalmente, podríamos representar el problema mediante la función:

$$f : C \times M \mapsto d, \tag{3.2}$$

donde $C \in \mathcal{B}^n$; sabiendo que $\mathcal{B} \in \{0, 1\}$.

3.2. Modulación en Frecuencia de Sonidos (Frequency Modulation Sounds - FMS)

El problema de los modulación en frecuencia de sonidos está propuesto en [TGCF97] como una tarea realmente dura que consiste en ajustar un modelo general $y(t)$ en una función de sonido

básica $y_0(t)$. El objetivo es minimizar la suma de errores cuadrados dados por

- Instancia del problema: Un vector de 6 números reales ($\vec{x} = (a_1, w_1, a_2, w_2, a_3, w_3)$) y las ecuaciones 3.3 y 3.4 que se corresponden con las funciones transformada y objetivo, respectivamente.

$$y(t) = a_1 \sin(w_1 t\theta + a_2 \sin(w_2 t\theta + a_3 \sin(w_3 t\theta))) \tag{3.3}$$

$$y_0(t) = 1.0 \sin(5.0 t\theta - 1.5 \sin(4.8 t\theta + 2.0 \sin(4.9 t\theta))) \tag{3.4}$$

- Solución Factible: Seis números reales.

- Función objetivo: La suma de los errores cuadrados de la diferencia de las funciones 3.3 y 3.4:

$$E_{FMS}(\vec{x}) = \sum_{t=0}^{100} (y(t) - y_0(t))^2 \tag{3.5}$$

- Solución óptima: Encontrar seis parámetros que asignen cero al valor del error cuadrado medio expresado en la Ecuación: 3.5. Es decir, que ajuste al máximo a la función $y(t)$ el objetivo $y_0(t)$.

- Solución aceptable: Cuando el error caiga por debajo del valor 0.01 se considerará que se ha encontrado una solución aceptable.

Tanto 3.3 como 3.4 tienen el mismo valor para el parámetro θ, que se corresponde con la expresión $\theta = \frac{2\pi}{100}$. El problema resultante es una función multimodo altamente compleja con una fuerte epistasis y con mínimo valor en $E_{FMS} = 0.0$.

3.3. Problema del Corte Máximo de un Grafo (Maximum Cut - MAXCUT)

Sea un grafo ponderado[1] $G = (V, E)$, con V el conjunto de vértices del grafo y E el conjunto de ejes que lo forman, el problema del corte máximo de un grafo consiste en dividir G en dos subgrafos disjuntos $G_0 = (V_0, E_0)$ y $G_1 = (V_1, E_1)$, de forma que $V_0 \cap V_1 = \emptyset$, $V_0 \cup V_1 = V$ y de manera que la suma de los pesos de los ejes que unen los vértices de cada uno de los subgrafos entre sí sea máxima.

Es decir, el problema trata de dividir el grafo inicial G en 2 grafos G_1 y G_2, de manera que $V_0 \cap V_1 = \emptyset$ y $V_0 \cup V_1 = V$ haciendo máxima, además, la función $\sum_{i,j=1}^{n} e_{ij} * w_{ij}$ sabiendo que $e_{ij} = 1$

[1] Grafo ponderado es aquel que tiene asignados pesos en sus ejes

si existe el eje que une los vértices i y j y w_{ij} es el peso de dicho eje.

Más formalmente, podemos definir el problema del corte máximo de un grafo como sigue [Stin87]:

- Instancia del problema: Un grafo ponderado $G = (V, E)$, donde $V = \{1, \ldots, n\}$ es el conjunto de vértices y E es el conjunto de ejes. w_{ij} es el peso del eje (i, j). Asumiremos que $w_{ij} = w_{ji}$ y que $w_{ii} = 0 \forall\ i \in 1, \ldots, n$.

- Solución factible: Un conjunto C conteniendo todos los ejes que poseen un extremo perteneciente a V_0 y el otro a V_1, donde $V_0 \cap V_1 = \emptyset$ y $V_0 \cup V_1 = V$. En otras palabras, V_0 y V_1 forman una partición de V.

- Función objetivo: La suma de los pesos de los ejes pertenecientes a C:

$$W = \sum_{|i,j| \in C} w_{ij} \tag{3.6}$$

- Solución óptima: Un corte que optimice el peso de la función de la Ecuación 3.6.

Podemos definir la función MAXCUT como sigue

$$f : P \times E \mapsto W, \tag{3.7}$$

Se puede comprobar consultando [Karp72] que este problema es NP-completo[2].

3.4. Problema Engañoso Masivamente Multimodal (Massively Multimodal Deceptive Problem - MMDP)

secc:MMDP El problema engañoso masivamente multimodal ha sido diseñado expresamente para resultar difícil de resolver para los GAs [GDH92]. Posee dos características que hacen a los problemas complejos de resolver por mediante estos algoritmos: el problema engañoso (*deceptive*) y el de masivamente multimodal (*massively multimodal*).

- Instancia del problema: El problema MMDP está formado por k subproblemas, de 6 bits cada uno.

- Solución factible: La compuesta de k cadenas de 6 bits cada una.

[2]NP es el tipo de problema de decisión para el que la solución propuesta ante una entrada puede ser comprobada en tiempo polinómico para ver si se trata realmente de una solución. NP-completo es todo problema que posea transformación inversa desde su expresión equivalente en forma polinómica

- Función objetivo: La suma de los valores de los k subproblemas que componen el problema:

$$F_{MMDP}(\overrightarrow{subpr}) = \sum_{i=1}^{k} fitness_{subpr_i} \tag{3.8}$$

- Solución óptima: Aquella que maximiza el valor de la ecuación 3.8, que se corresponde con el valor k.

La solución óptima se corresponde con el caso en el que todo subproblema está compuesto de cero o seis unos. Todos los subproblemas contribuyen al fitness según el número de unos que posean (ver Figura 3.1), como se muestra en la Tabla 3.1.

Cada uno de los k problemas que forman el MMDP posee la que característica de ser engañoso, ya que todos ellos constan de dos máximos globales y un atractor engañoso (punto ℓ en la Figura 3.1). Por tanto, cada una de los k subproblemas que utilizaremos poseen un total de $\binom{6}{3} + 2 = 20 + 2 = 22$ óptimos; de los cuales tan sólo 2 serán globales. Y teniendo en cuenta el problema completo (con los k subproblemas), el número de máximos locales es realmente grande (unos 22^k), de los cuales sólo 2^k son soluciones globales.

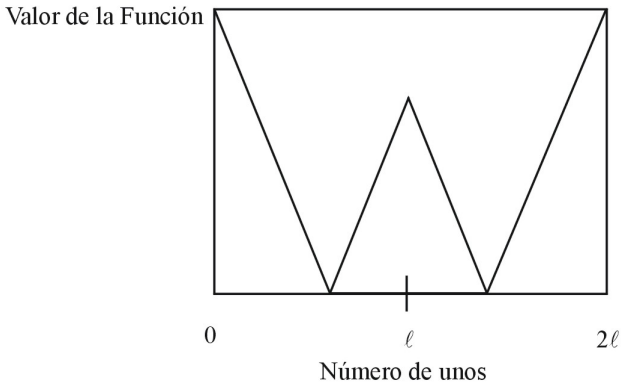

Figura 3.1: Construcción intuitiva de una función engañosa bipolar

3.5. Problema de Tarea con Espera Mínima (Minimum Tardy Task Problem - MTTP)

El problema de tarea con espera mínima (MTTP) consiste en un problema de organización de tareas. Se puede comprobar consultando [Karp72] que este problema, al igual que MAXCUT, es NP-Completo. La definición formal del problema se presenta a continuación [Stin87]:

Número de Unos	Valor de la Función ($fitness_{subpr}$)
0	1.000000
1	0.000000
2	0.360384
3	0.640576
4	0.360384
5	0.000000
6	1.000000

Tabla 3.1: Valores que puede tomar cada subproblema de MMDP

- Instancia del problema:
  - Tareas: $1, 2, \ldots, n$; con $i > 0$
  - Longitudes: $l_1, l_2, \ldots, l_n$; con $l_i > 0$
  - Límites de tiempo: $d_1, d_2, \ldots, d_n$; con $d_i > 0$
  - Pesos: $w_1, w_2, \ldots, w_n$; con $w_i > 0$
- Soluciones: Una planificación g definida en

$$S \subseteq T/g : S \mapsto \mathcal{Z}^+ \cup \{0\}/\forall i, j \in S : \tag{3.9}$$

  1. Si $g(i) < g(j) \Rightarrow g(i)+l_i \leq g(j)$, lo que nos asegura que una tarea no se puede planificar antes de que se complete la anterior.
  2. $g(i) + l_i \leq d_i$, lo que nos asegura que toda tarea se completa antes de llegar a su límite de tiempo.

- Función objetivo: Se corresponde con la suma de los pesos de las tareas no planificadas:

$$W = \sum_{i \in T-S} w_i \tag{3.10}$$

- Solución óptima: Se corresponderá con la planificación S que posea el mínimo W.

El subconjunto S del conjunto de tareas T es una planificación factible si es posible pensar en ella como una lista de tareas ordenadas en orden creciente según su tiempo límite de vida, sin violar en ningún caso este tiempo [Stin87]. Si no estuvieran en este orden, para comprobar la corrección de la planificación sería necesario desarrollar un paso previo que ordenara las tareas según el orden creciente de sus límites de tiempo de vida y renombrarlas en la forma $d_1 \leq d_2 \leq \ldots \leq d_n$.

3.6. El Problema de las P Cimas (P-PEAKS)

El problema de las P cimas es un generador de problemas propuesto en [JPS97]. Un generador de problemas es una tarea fácilmente parametrizable que tiene un nivel de dificultad ajustable, permitiéndonos obtener instancias con dificultad tan alta como queramos. Además, utilizar un generador de problemas elimina la posibilidad de ajustar a mano los algoritmos para un problema particular; por tanto nos proporciona una mayor justicia a la hora de comparar algoritmos. Con un generador de problemas evaluamos los algoritmos en un número elevado de instancias de problemas aleatorios distintos entre sí en cada ejecución, incrementándose así el poder predictivo de los resultados para la clase problema. El generador de problemas estudiado en [AT00] es el P-PEAKS [JPS97].

- Instancia del problema: Consistirá en P cadenas de longitud N bits cada una.
- Solución factible: Cualquier cadena de N bits.
- Función objetivo: La mínima distancia de hamming entre la cadena y la cima más próxima a ésta:

$$f_{P-PEAKS}(\vec{x}) = \min_{i=1}^{P}\{HammingD(\vec{x}, Peak_i)\} \tag{3.11}$$

- Solución óptima: Se corresponde con el caso en el que la función 3.11 valga cero.

Las P cadenas aleatorias que tendremos en la instancia del problema representan la localización de las P cimas en el espacio de búsqueda.

Para evaluar una cadena arbitraria de bits, primero se localiza el pico más cercano en el espacio de Hamming. Entonces el *fitness* de la cadena de bits es el número de bits que la cadena tiene en común con esa cima cercana, y dividido por N (ver Ecuación 3.11). Los problemas con un número mayor o menor de cimas poseen una más o menos elevada epistasis, respectivamente.

Se ha incluido este problema en nuestro trabajo por su interés en el estudio de la mutación y el emparejamiento en GAs. Consideremos un problema simple 2-PEAKS [Kenn98], con óptimos en las cadenas $000\ldots000$ y $111\ldots111$. Los individuos con el 50 % de 1's y 0's serán los que posean un menor valor de adecuación. La mutación de cualquier individuo con elevado valor de adecuación en cualquier cima tenderá a mantener al individuo en dicha cima, pero alejándolo del valor del óptimo global para esa cima. Por otro lado, la recombinación produce resultados algo diferentes, dependiendo de la localización de los padres: si los dos padres están en la misma cima, es de esperar que el descendiente estará con una alta probabilidad en la misma cima; pero si los padres se encuentran en cimas distintas, lo más probable es que el hijo se encuentre en un valle entre ambas cimas, donde el valor de adecuación es bastante bajo.

Por tanto, podemos tener la hipótesis de que el emparejamiento puede dañar el rendimiento del algoritmo. ¿Qué sucede si hay más de una cima? Parece lógico pensar que el emparejamiento será aún más deteriorante, ya que el emparejamiento de individuos de diferentes cimas es más probable que produzca aún peores descendientes hasta que la población converja a una cima. De aquí el interés que produce este problema en el estudio de la mutación y el emparejamiento.

3.7. Satisfacción de Cláusulas Lógicas (Satisfiability Problem - SAT)

Antes de definir el problema, debemos introducir algunas definiciones:

- Literal: Variable booleana x_i o su negación $\overline{x_i}$
- x_i (o $\overline{x_i}$) se satisface respecto a una entrada $a = (a_1, a_2, \ldots, a_n)$ si $a_i = 1$ (o $a_i = 0$).
- Cláusula: Es una disyunción de literales.

A continuación pasaremos a describir el problema.

3.7.1. El Problema SAT

Una vez aclaradas las definiciones anteriores, podemos definir el problema de satisfacción de cláusulas lógicas (SAT) como sigue: dada una función lógica $f : \mathcal{B}^n \rightarrow \mathcal{B} = \{0, 1\}$, encontrar una asignación de variables (si existe) $x = (x_1, x_2, \ldots, x_n) \in \mathcal{B}^n$, tal que $f(x) = 1$. Asumimos que f viene dada en su forma normal conjuntiva sin que ello suponga una pérdida de generalidad[3]: $f = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ donde C_i es una disyunción de l_i ($i \in \{1, \ldots, k\}$) literales.

Por tanto, dada una fórmula lógica expresada en su forma normal conjuntiva $f = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ en la que cada C_i, con $i \in \{1, \ldots, m\}$, está formada por la disyunción de k literales $C_i = l_1 \vee l_2 \vee \ldots \vee l_k$, el problema SAT trata de encontrar una asignación de variables $x = (x_1, \ldots, x_n)$ de forma que satisfaga a la fórmula f; o, más formalmente, de forma que $f(x) = 1$. Mientras que en el caso del problema kSAT además debe cumplirse que dicha asignación de variables satisfaga todas y cada una de las cláusulas que forman f; es decir, $C_i(x) = 1 \backslash \forall\, i \in \{1, \ldots, m\}$.

El problema SAT ha recibido mucha atención por parte de la comunidad científica debido al hecho de que cualquier problema del tipo de problemas NP puede ser transformado en tiempo polinomial a un problema SAT equivalente (teorema de Cooke); mientras que la transformación

[3]Cualquier fórmula lógica puede ser expresada en su forma normal conjuntiva (CNF) y disyuntiva (DNF)

inversa en tiempo polinomial puede no existir. Al tipo de problemas que sí poseen esta transformación inversa se les denomina NP-completos. El problema SAT fue el primero para el que se demostró su pertenencia a la clase de problemas NP [Cook71].

Al igual que en muchos otros tipos de problemas, en la función SAT aparece una característica llamada *transición de fase*, cuya presencia en el problema implica que al variar un parámetro del mismo éste pasa de ser fácilmente resoluble a ser muy difícil de resolver. La zona de transición en la que el problema pasa de tener fácil solución a tener solución difícil se denomina *región sensible* o *región crítica*. En la Figura 3.2 se puede apreciar esta región crítica sombreada para el problema SAT con 200 variables.

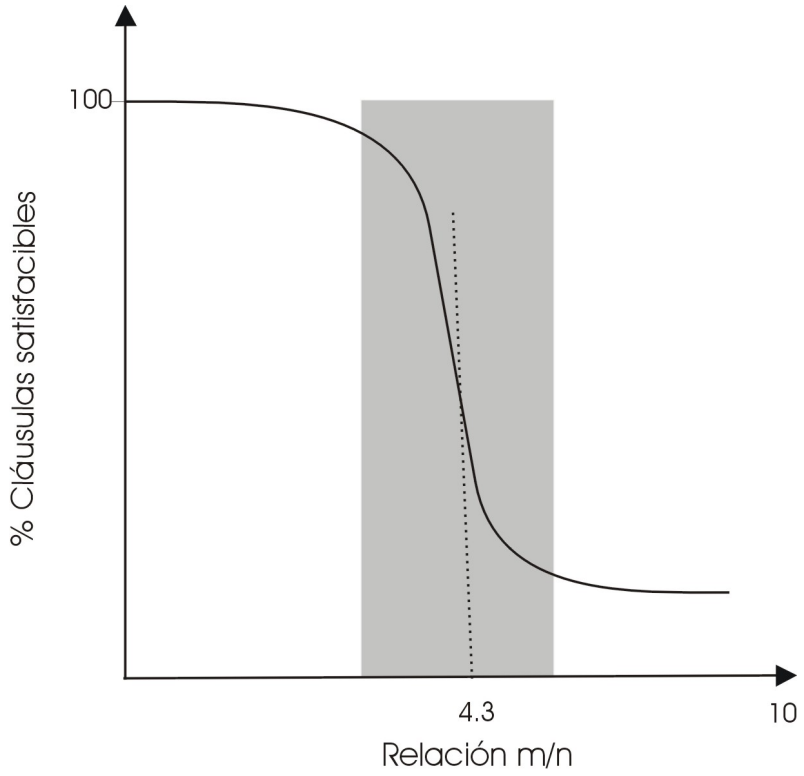

Figura 3.2: Porcentaje de satisfactibilidad y dificultad del problema (normalizada de 0 a 100) para el problema 3-SAT con 200 variables [CD96]

Muchos estudios como [Kiho95], [Smi97], [Hay97] o [SC95]han analizado la fase de transición presente en SAT y de ellos se desprende que la dificultad de encontrar solución estriba en función de la tasa $\phi = \frac{m}{n}$, siendo m el número de cláusulas que componen la función y n el de variables distintas que existen en el problema. En concreto, para problemas 3SAT, el incremento de la dificultad comienza cuando $\phi \simeq 4.25$. Para $\phi < 4.25$ el problema es de fácil solución: es satisfacible con

una alta probabilidad y existen multitud de asignaciones de variables que llevan a una solución; mientras que si $\phi > 4.25$ el problema resulta difícil de resolver.

Para que kSAT encuentre solución necesariamente la fórmula debe ser satisfacible. Según el objetivo de búsqueda del problema que queramos utilizar podemos encontrar distintas variantes al problema kSAT, como son MAX-kSAT y COUNT-kSAT (en adelante, siempre utilizaremos $k = 3$, y simplificaremos estos nombres a MAXSAT y COUNTSAT)[FJW00].

3.7.2. El Problema MAXSAT

El problema MAX-kSAT puede definirse como sigue [BCCG00]:

- Instancia: Un conjunto $X = \{x_1, \ldots, x_n\}$ de variables lógicas y una colección $C = \{C_1, \ldots, C_m\}$ de cláusulas con exactamente k literales.
- Solución: Una asignación de verdad a las variables de X.
- Medida: Número de cláusulas satisfechas por la asignación de verdad.

Consideremos el caso en que $k = 3$, es decir, las cláusulas que forman el problema están formadas por exactamente tres variables.

El objetivo del problema MAXSAT consiste en encontrar una asignación de variables $X = (x_1, \ldots, x_n)$ de forma que satisfaga el máximo número posible de cláusulas. Se puede ver en [GJ79] que el problema MAXSAT es un problema duro incluso utilizando cláusulas de tan sólo dos literales.

En este estudio utilizaremos el tipo de cláusulas presentado en [Papa94], que se puede definir mediante los siguientes puntos:

- Variables utilizadas: x_i, con $1 \leq i \leq n$.
- Descripción de las cláusulas (cláusulas de Horn): $x_i \vee \overline{x_j} \vee \overline{x_k}$, con $(i, j, k) \in \{1, \ldots, n\}^3$, siendo $i \neq j \neq k \neq i$.

Utilizando este tipo de cláusulas resulta sencillo comprobar la evolución que va realizando el algoritmo durante su ejecución, ya que la única solución que satisface todas las cláusulas es la que asigna unos a todas las variables.

3.7.3. El Problema COUNTSAT

Para el estudio que realizaremos, vamos a transformar MAXSAT en un problema polinomial [FJW00]

$$COUNTSAT_n : \{0,1\}^n \rightarrow \mathcal{Z}, \tag{3.12}$$

donde $COUNTSAT_n(x)$ se corresponde con el número de cláusulas que satisfacen la entrada x. De esta manera, el objetivo de nuestro GA será encontrar la entrada x que maximice el valor de la función COUNTSAT.

- Instancia del problema: Una cadena de n bits, que se corresponden con las n variables lógicas.
- Solución factible: Cualquier asignación de valores lógicos sobre dicha cadena.
- Función objetivo: La función que cuenta el número de cláusulas que se satisfacen con los valores lógicos asignados a las variables en la cadena de n bits.

$$F_{COUNTSAT_n}(\vec{x}) = \sum_{1 \leq i \leq n} x_i + \sum_{1 \leq i \leq n} \sum_{\substack{1 \leq j \leq n \\ j \neq i}} \sum_{\substack{1 \leq k \leq n \\ k \neq i, k \neq j}} (1 - (1 - x_i)x_j x_k) \tag{3.13}$$

- Solución óptima: Se corresponde con el caso que maximice el valor de la función 3.13.

Por tanto, dada una secuencia de cláusulas $C_1, C_2, \ldots, C_m$ sobre las variables $x_1, x_2, \ldots, x_n$, COUNTSAT devuelve el número de cláusulas que satisface la entrada a. En nuestro caso, cada cláusula C_i será una cláusula de Horn de tres literales (presentada en [Papa94]).

El problema que estudiaremos tratará de encontrar la entrada que haga máximo el valor devuelto por COUNTSAT; el cual, como es fácil ver, será el correspondiente a la entrada que asigne a todas las variables el valor 1.

Capítulo 4

Representación de los Problemas y Funciones de Evaluación

A continuación se describirá la representación de la solución que se ha utilizado en los individuos para cada problema; así como la función de evaluación utilizada en cada caso.

El capítulo está estructurado de forma que en la Sección 4.1 se detallan los parámetros de implementación comunes a todos los problemas estudiados y en las restantes secciones, de la 4.3 a la 4.8, se van estudiando estos problemas de manera individual; describiendo los parámetros específicos que utilicen.

4.1. Parámetros Comunes Utilizados en la Resolución de los Problemas

Los parámetros utilizados para resolver los problemas siguen, por lo general, y mientras no se indique lo contrario, las mismas pautas que las establecidas en [AT00] y que se especifican a continuación:

- Tamaño de la población: 400 individuos
- Selección de los dos padres: Roulette Wheel [1]
- Tipo de emparejamiento: DPX1[2]
- Probabilidad de emparejamiento: $p_c = 1.0$

[1] Ruleta Rusa; descrita en la Sección 2.3.6

[2] Partiendo de dos padres obtenemos un hijo con la porción más larga del mejor padre.

- Tipo de reemplazo: Replace if better[3]
- Valores de los alelos: $\{0, 1\}$
- Probabilidad de mutación: $p_m = \frac{1}{L}$; Siendo L la longitud del genotipo.

A continuación se pasará a describir, para cada uno de los problemas estudiados, los detalles de implementación: La codificación de la solución empleada en los individuos y la función de evaluación que aplicaremos a dichas soluciones.

4.2. Satisfacción de Cláusulas Lógicas (COUNTSAT)

El problema COUNTSAT es un problema especialmente difícil de resolver para un GA debido a la existencia de un máximo local de valor muy próximo al máximo global y del que es muy difícil salir, ya que al estar lejanos los máximos global y local, una vez que el algoritmo se aproxime al máximo local, tendrá que dar un salto muy grande para salir de ese máximo local y así poder encontrar el global. Todo esto se puede ver claro en la Figura 4.1, extraída de [FJW00], donde se reflejan los valores que toma la función COUNTSAT con 40 variables con la entrada extendida al rango $[-10, 45]$. El máximo local se corresponde con el valor de la función para la entrada en la que todas las variables toman valor 0, mientras que el máximo global se dará con la entrada formada por 40 unos.

Para decidir el número de literales que utilizaremos durante el estudio del problema se ha decidido ejecutar el problema con distintos números de variables y siempre bajo una misma rejilla, la de 20×20, y con la misma probabilidad de mutación la definida como $p_m = \frac{1}{L}$. Se han realizado pruebas de ejecución para instancias de 20, 23, 25, 27 y 30 variables distintas. Los resultados obtenidos con estas pruebas se detallan en la Tabla 4.1.

Variables	Núm. Medio de Evaluaciones	Tiempo Medio	Soluciones Encontradas
20	22340.43 ± 17114.63	8.71 ± 6.52	23.33 %
23	27534.33 ± 10413.14	13.33 ± 5.13	10 %
25	24059 ± 0	11 ± 0	3.33 %
27	62956 ± 4536.8	30.5 ± 2.12	6.67 %
30	0	0	0 %

Tabla 4.1: Estudio del problema COUNTSAT con distintos números de variables

[3]El hijo reemplaza a su padre sólo si su valor de adecuación es mejor.

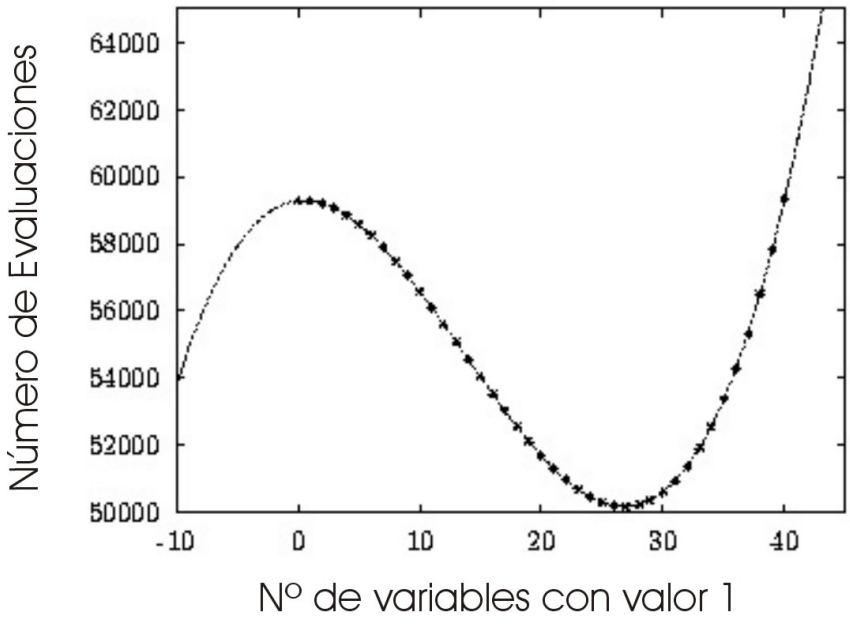

Figura 4.1: Función COUNTSAT para individuos de longitud 40

Vistos estos resultados se ha decidido implementar el problema con 20 variables, ya que es el caso que da mayores porcentajes de soluciones encontradas. Una vez realizada esta elección se ha resuelto el problema utilizando distintas probabilidades de mutación con el fin de intentar incrementar el porcentaje de soluciones encontradas. La tabla 4.2 muestra los resultados obtenidos por el problema según la probabilidad de mutación utilizada.

Estudiando los resultados de la Tabla 4.2 se ha decidido utilizar la probabilidad de mutación de 0.025, que coincide con el valor de $\frac{1}{2*L}$ para el caso de $L = 20$ variables.

La función que el problema $COUNTSAT_n(x)$ trata de maximizar es:

$$F_{COUNTSAT_n}(\vec{v}) = \sum_{1 \leq i \leq n} x_i + \sum_{1 \leq i \leq n} \sum_{\substack{1 \leq j \leq n \\ j \neq i}} \sum_{\substack{1 \leq k \leq n \\ k \neq i, k \neq j}} (1 - (1 - x_i) x_j x_k) \tag{4.1}$$

Según se puede comprobar en [FJW00], podemos obtener a partir de 4.1 una fórmula que nos permita evaluar $COUNTSAT_n(x)$ en orden lineal ($O(n)$ frente a $O(n^3)$ de 4.1). Esto lo logramos ya que, gracias a la simetría podemos obtener una descripción más simple de $COUNTSAT_n(x)$

Pr. Mutación	N. Medio de Evaluaciones	Tiempo Medio	Soluciones Encontradas
0.0	29235.55 ± 24100.37	11.36 ± 9.28	36.67 %
0.01	25963.75 ± 31919.25	11.67 ± 14.31	40 %
0.015	19336.11 ± 8496.51	7.97 ± 3.39	30 %
0.02	24095.45 ± 10009.68	9.45 ± 3.96	36.67 %
0.025	24326.33 ± 16321.27	9.67 ± 6.43	40 %
0.03	16403.55 ± 13037.69	6.45 ± 5.28	36.67 %
0.035	36203.57 ± 20559.11	14.29 ± 8.3	23.33 %
0.05	25512.63 ± 14232.37	10 ± 5.53	26.67 %
0.075	16540.25 ± 15727.43	6.75 ± 5.91	13 %
0.1	24059 ± 16932.46	10.67 ± 7.64	10 %
0.15	0	0	0 %

Tabla 4.2: Estudio del problema COUNTSAT con distintos valores de probabilidad de mutación

definida en $s = \sum_{i=1}^{n} x_i$ de la forma: $COUNTSAT_n^* : 0, \ldots, n \rightarrow \mathcal{R}$. Como $(1 - (1 - x_i)x_j x_k) = 1 - x_j x_k + x_i x_j x_k$, $COUNTSAT_n^*(\vec{x})$ se puede escribir como:

$$COUNTSAT_n^*(x) = s + n(n-1)(n-2) - 2(n-2) \sum_{1 \leq j < k \leq n} x_j x_k + 6 \sum_{1 \leq i < j < k \leq n} x_i x_j x_k \quad (4.2)$$

Tras definir $s = x_1 + \ldots + x_n$, de la Ecuación 4.2 obtenemos la función que se utilizará durante la ejecución del problema:

$$COUNTSAT_n^*(s) = s + n(n-1)(n-2) - 2(n-2) \begin{pmatrix} s \\ 2 \end{pmatrix} + 6 \begin{pmatrix} s \\ 3 \end{pmatrix} \quad (4.3)$$

que, como se ha comentado antes, es evaluable en tiempo lineal.

Una vez conocida esta función, vemos que tomará su valor de adecuación óptimo cuando todas las variables tomen el valor 1. Por tanto $s = x_1 + \ldots + x_n = \overbrace{1 + \ldots + 1}^{n} = n$ y, según la Ecuación 4.3,

$$\begin{aligned} COUNTSAT_n^*(s) &= n + n(n-1)(n-2) - 2(n-2) \begin{pmatrix} n \\ 2 \end{pmatrix} + 6 \begin{pmatrix} n \\ 3 \end{pmatrix} \\ &= n + n(n-1)(n-2) - 2(n-2)\frac{n!}{2!(n-2)!} + 6\frac{n!}{3!(n-3)!} \\ &= n + n(n-1)(n-2) - \frac{n!}{(n-3)!} + \frac{n!}{(n-3)!} \\ &= n + n(n-1)(n-2) \end{aligned} \quad (4.4)$$

Los parámetros que se utilizarán para el estudio del comportamiento del problema con las distintas rejillas se detallan a continuación:

- Función de adecuación: Ecuación 4.3
- Solución óptima: según la Ecuación 4.4, $fitness = 20 + 20 * 19 * 18 = 6860$
- Longitud del genotipo: 20 alelos (número de variables)
- Probabilidad de Mutación: $p_m = \frac{1}{2L} = 0.025$

En la Figura 4.2 podemos ver un esquema de la estructura del cromosoma de los genotipos. Está formado por tantos alelos como el número de variables que utilicemos, cada uno de los cuales tomará el valor binario que se le da a su variable correspondiente en la asignación de verdad utilizada como entrada al problema.

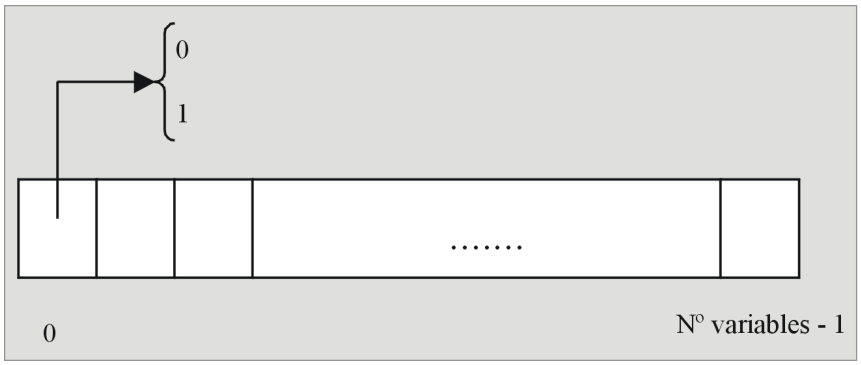

Figura 4.2: Estructura de un individuo para el problema COUNTSAT

Este problema es especialmente complicado de resolver para un GA debido a la existencia de un máximo local ($a_i = 0 \forall i$) de valor $n * (n-1) * (n-2)$, que es muy próximo al máximo global $n * (n-1) * (n-2) + n$, y del que, como se puede apreciar en la Figura 4.1, es muy difícil que el algoritmo pueda salir.

Una entrada aleatoria tiene aproximadamente $\frac{n}{2}$ unos; siendo n el número de variables. Según se aprecia en la Figura 4.1, los cambios locales sobre esta entrada decrementando el número de unos conducen a mejores individuos, mientras que los cambios incrementando el número de unos decrementan el valor de la función de adecuación; a menos que sean cambios muy bruscos.

Por tanto, es de esperar que el EA tienda a encontrar rápidamente el máximo local correspondiente al caso en el todas las variables contienen el valor cero y encuentra dificultad para obtener

la cadena formada por unos en todas sus posiciones, que se corresponde con la entrada para la que la función de adecuación del problema obtiene su máximo global.

4.3. Problema del Diseño de Código Corrector de Errores (ECC)

La instancia de este problema se basa en la descrita en [CFW98], y se detalla a continuación:

- Longitud de las palabras de código: $n = 12$
- Número de palabras de código: $M = 24$
- Función de adecuación:

$$E_{ECC}(C) = \frac{1}{\sum_{i=1}^{M} \sum_{\substack{j=1 \\ i \neq j}}^{M} \frac{1}{d_{ij}^2}} \tag{4.5}$$

  siendo d_{ij} la distancia mínima de Hamming entre las palabras i y j del código C.

- Solución óptima: $fitness = 0.0674$.
- Longitud del genotipo: 288 alelos binarios $(12 * 24)$.

Se observa que este problema no es fácil de resolver para un GA debido a que el espacio de búsqueda a explorar es de $C_{24}^{2^{12}} = \begin{pmatrix} 2^{12} \\ 24 \end{pmatrix} \simeq 10^{87}$ posibles soluciones.

En la Figura 4.3 se detalla la estructura del genotipo utilizado en los individuos para resolver el problema. En ella podemos ver que en el único cromosoma que representa al individuo están representadas todas las palabras que formarán el código. De esta manera, si el código está formado por n palabras de longitud M, necesitaremos un cromosoma de $n * M$ alelos para representarlo. Es obvio que cada alelo se corresponderá con un único y determinado bit de una palabra concreta del código buscado.

4.4. Modulación en Frecuencia de Sonidos (Frequency Modulation Sounds - FMS)

Cada parámetro real se codifica con 32 bits en el rango $[-6.4, +6.35]$. Debido a que la solución del problema consiste en minimizar la ecuación 3.5 y los GAs sólo realizan maximizaciones, la

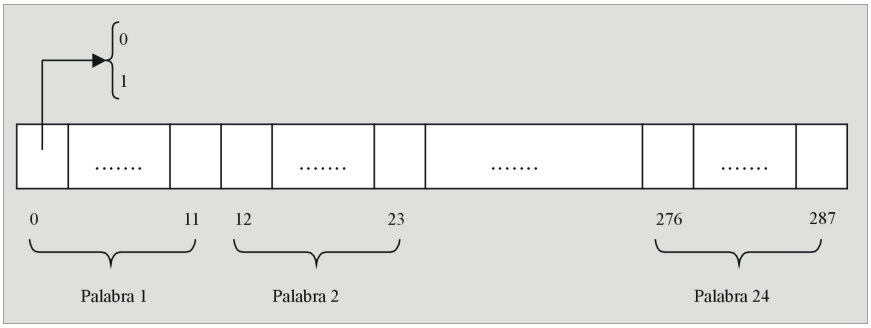

Figura 4.3: Estructura de un individuo para el problema ECC

función de evaluación que utilizaremos vendrá dada por la inversa de $E_{FMS}(\vec{x})$:

$$F_{FMS}(\vec{x}) = \begin{cases} \frac{1}{E_{FMS}(\vec{x})} & \text{Si } E_{FMS}(\vec{x}) \neq 0 \\ 100 & \text{En otro caso} \end{cases} \tag{4.6}$$

Se le ha asignado el valor máximo 100 a la función ya que el algoritmo parará cuando el error caiga por debajo de 0.01, en cuyo caso $F_{FMS}(\vec{x}) = \frac{1}{E_{FMS}(\vec{x})} = \frac{1}{0.01} = 100$.

En la Figura 4.4 se puede ver la estructura utilizada para representar el problema en los individuos.

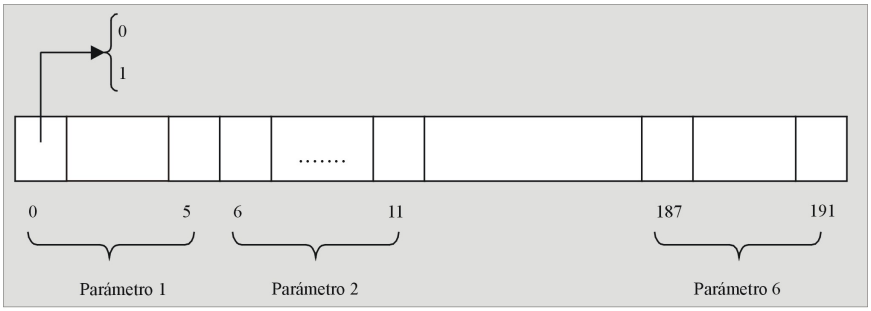

Figura 4.4: Estructura de un individuo para el problema FMS

4.5. Problema del Corte Máximo de un Grafo (MAXCUT)

Para representar el problema, utilizaremos una cadena binaria de longitud n en la que cada bit se corresponderá con un vértice del grafo, de manera que si el bit correspondiente a la posición i (x_i) es 0, el vértice i pertenecerá al conjunto V_0, mientras que si dicho bit tiene valor 1, el vértice i formará parte del conjunto V_1.

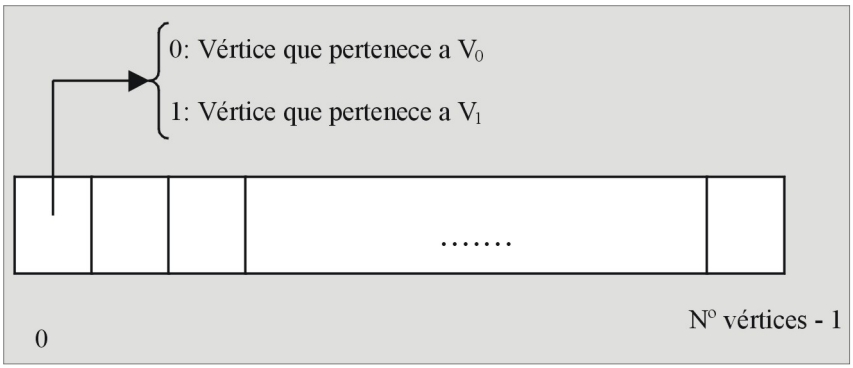

Figura 4.5: Estructura de un individuo para el problema MAXCUT

Se puede apreciar en la Figura 4.5 cómo es el esquema del genotipo que se ha utilizado para resolver el problema. Está formado por un cromosoma cuyos alelos toman valores binarios; el número de alelos que componen el cromosoma se corresponde con el número de vértices que forman el grafo. De esta manera, cada alelo representará a un vértice y su valor indicará si el vértice al que representa pertenece a un subgrafo o al otro.

La función que el algoritmo tratará de maximizar se detalla a continuación [KBH94]:

$$F_{MAXCUT}(\vec{x}) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij}[x_i(1-x_j) + x_j(1-x_i)] \tag{4.7}$$

Durante el estudio del problema se utilizarán tres instancias distintas [KBH94], cada una de las cuales se corresponde con una de las tres secciones que siguen.

4.5.1. Grafo Disperso de 20 Vértices

Las características del problema se describen a continuación:

- Instancia utilizada: cut20.01.dat; descrito en [KBH94].
- Función de adecuación: la Ecuación 4.7.
- Solución óptima: $fitness = 10.119812$.
- Longitud del genotipo: 20 alelos (número de vértices del grafo).

4.5.2. Grafo Denso de 20 Vértices

El grafo utilizado en esta sección está, al igual que el anterior, obtenido de [KBH94]. Los parámetros utilizados para este grafo son:

- Instancia utilizada: cut20.09.dat.
- Función de adecuación: la Ecuación 4.7.
- Solución óptima: $fitness = 56.740064$.
- Longitud del genotipo: 20 alelos (número de vértices del grafo).

4.5.3. Grafo de 100 Vértices

El grafo estudiado en esta ocasión es un grafo escalable introducido en [KBH94]. La creación de dicho grafo es bastante sencilla. Se disponen los vértices en dos columnas. Cada vértice se conecta mediante un eje de peso 1 con los vecinos de su columna (excepto los dos primeros vértices de cada columna -el primer eje-, que se conectan con peso igual a 0.5). Los vértices de distintas columnas se conectan en cruz con ejes de peso 10. En la Figura 4.6 se puede ver el grafo generado para 10 nodos obtenido según se ha explicado.

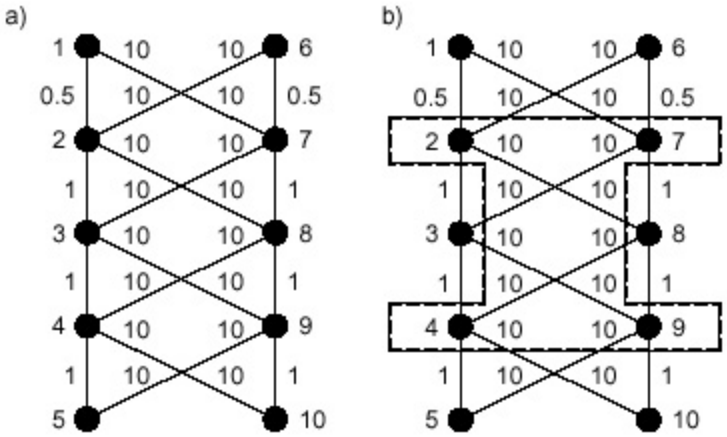

Figura 4.6: Grafo generado de 10 vértices: Estructura -a)- y corte máximo -b)-

Los parámetros utilizados en la resolución del problema son:

- Instancia utilizada: cut100.dat.
- Función de adecuación: la Ecuación 4.7.
- Solución óptima:

$$fitness = 21 + 11 * (n - 4), \tag{4.8}$$

con $n \geq 4$ (n: número de nodos del grafo)

- Longitud del genotipo: 100 alelos (número de vértices del grafo).

La partición óptima para este grafo será la cadena formada por la repetición durante 50 veces de la secuencia '01', o su complemento. Y su valor de adecuación será, según la Ecuación 4.8, de $21 + 11 * (100 - 4) = 1077$.

4.6. Problema Engañoso Masivamente Multimodal (Massively Multimodal Deceptive Problem - MMDP)

En [AT00] se ha utilizado para resolver el problema una instancia considerablemente grande de $k = 40$ subproblemas. Utilizando dicha instancia, la estructura de cada individuo será como aparece en la Figura 4.7.

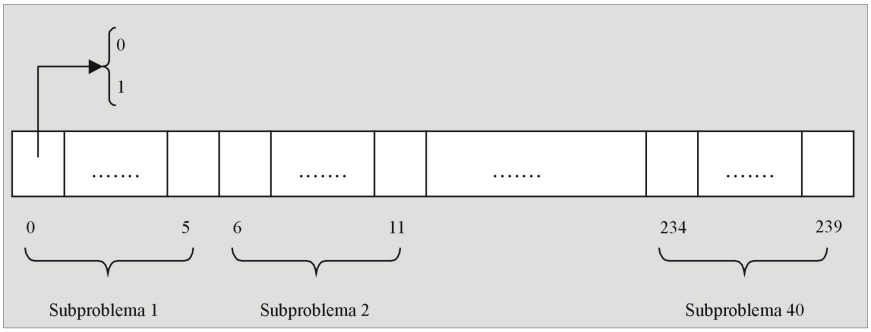

Figura 4.7: Estructura de un individuo para el problema MDDP

La función de evaluación utilizada en la implementación de este problema es la misma que la descrita en la Sección 5.6,y se corresponde con la siguiente:

$$F_{MMDP}(\overrightarrow{subpr}) = \sum_{i=1}^{k} fitness_{subpr_i} \tag{4.9}$$

4.7. Problema de Tarea con Espera Mínima (MTTP)

Para resolver este problema representaremos la planificación de tareas S como un vector $x = (x_1, x_2, \ldots, x_n)$, con $x_i \in \{0, 1\}$, en el que si $x_i = 1$ la tarea $i \in S$, y si $x_i = 0$, entonces la tarea i no se encontrará planificada en S.

En la Figura 4.8 podemos ver la representación del genotipo de los individuos que se usará en este problema. Cada individuo estará formado por un cromosoma que constará de tantos bits como tareas tenga que planificar el problema. De esta manera, cada bit se corresponderá con una determinada tarea de forma que el primer bit del cromosoma represente a la tarea con menor límite máximo de tiempo (también llamado *deadline*) y el último bit simbolice a la tarea con mayor *deadline*. Para realizar esta asignación necesitaremos tener las tareas ordenadas según su límite máximo de tiempo; de no ser así, habría que introducir un paso previo al algoritmo que las ordene. Una vez realizada esta asignación entre alelos y tareas le podremos asignar dos posibles valores a cada alelo: 1 si su tarea correspondiente se incluye en la planificación o 0 si finalmente no se incluye.

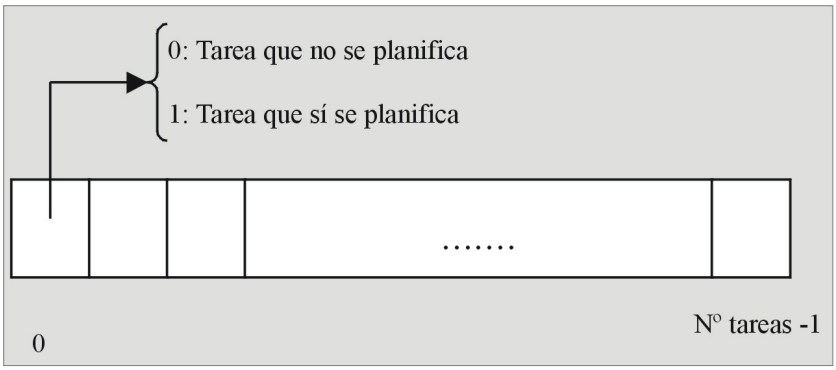

Figura 4.8: Estructura de un individuo para el problema MTTP

La función de adecuación y los conjuntos de tareas que utilizaremos para estudiar este problema se pueden encontrar en [KBH94]. A continuación se presenta la función de adecuación utilizada:

$$F_{MTTP} = \frac{1}{\sum_{i \in T-S} w_i} \tag{4.10}$$

siendo S el conjunto de tareas seleccionadas para la planificación y T el conjunto de todas las tareas existentes.

A continuación se detallan los parámetros utilizados en cada una de las tres instancias estudiadas.

4.7.1. MTTP con 20 Tareas

Los parámetros utilizados en el estudio de este problema son:

- Instancia utilizada: mttp20.dat, descrito en [KBH94].
- Función de adecuación: la Ecuación 4.10.
- Solución óptima: $fitness = \frac{1}{41} = 0.02439$.

- Longitud del genotipo: 20 alelos (número de tareas a planificar).

4.7.2. MTTP con 100 Tareas

Los parámetros utilizados para resolver el problema son:

- Instancia utilizada: mttp100.dat, obtenido utilizando una instancia de problema escalable que se presenta en [KBH94].
- Función de adecuación: la Ecuación 4.10.
- Solución óptima: $fitness = \frac{1}{200} = 0.005$.
- Longitud del genotipo: 100 alelos (número de tareas a planificar).

4.7.3. MTTP con 200 Tareas

Al igual que en el caso de MTTP de 100 tareas, el fichero utilizado para obtener las tareas, se obtiene utilizando la instancia de problema escalable que se presenta en [KBH94]. Los parámetros utilizados son:

- Instancia utilizada: mttp200.dat.
- Función de adecuación: la Ecuación 4.10.
- Solución óptima: $fitness = \frac{1}{400} = 0.0025$.
- Longitud del genotipo: 200 alelos (número de tareas a planificar).

4.8. El Problema de las P Cimas (P-PEAKS)

En [AT00] se han utilizado instancias de $P = 100$ cimas de $N = 100$ bits cada una, lo que representa un nivel medio-alto de epistasis. Podemos ver la estructura que utilizaremos para representar la solución en los individuos en la figura 4.9.

La función de evaluación utilizada en este problema se corresponde con la ecuación 4.11:

$$F_{P-PEAKS}(\vec{x}) = \frac{1}{N} \max_{i=1}^{P} \{N - HammingD(\vec{x}, Peak_i)\} \tag{4.11}$$

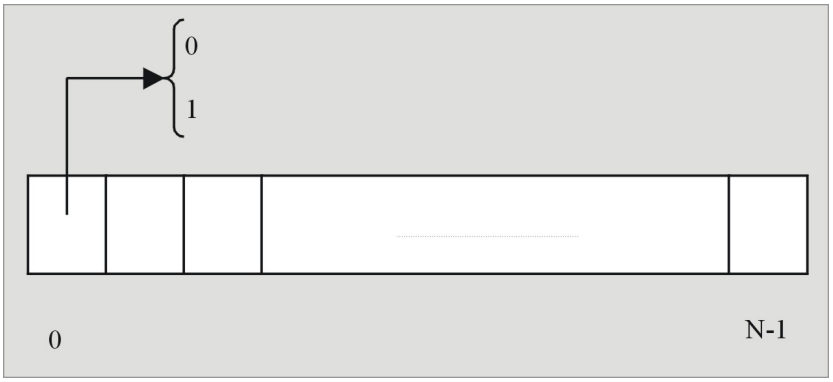

Figura 4.9: Estructura de un individuo para el problema P-PEAKS

Capítulo 5

Resultados Experimentales Utilizando Rejillas Fijas y Dinámicas Preprogramadas

En este capítulo se describirán los resultados obtenidos por un cGA al realizar pruebas con cambio preprogramado sobre los problemas descritos en [AT00] y otros más que se han añadido con el fin de ampliar dicho estudio (ver Capítulo 3). Se han empleado las rejillas estudiadas en [AT00]: *Square, Rectangular* y *Narrow* (rejillas fijas), además de las siguientes combinaciones de éstas: *SqNar (Square/Narrow)* y *NarSq (Narrow/Square)*. Todas ellas aparecen definidas en la Sección 4.1.

El capítulo está estructurado de forma que en las secciones de la 5.2 a la 5.8, se van estudiando los resultados obtenidos para cada uno de los problemas con las distintas rejillas utilizadas. En la Sección 5.9 se analizarán los mismos resultados, pero en este caso fijándonos en otro punto de vista: se comparará el comportamiento de todos los problemas para cada una de las rejillas en lugar de estudiar el comportamiento de las distintas rejillas para cada problema. Por tanto, cada una de las subsecciones que forman la Sección 5.9 se realizará una comparación sobre los resultados obtenidos por todos los problemas para cada una de las rejillas utilizadas. Por último, en la Sección 5.10 se presenta una tabla resumen con todos los resultados obtenidos a lo largo de todas las secciones.

5.1. Detalles de la Implementación

La condición de finalización del algoritmo es encontrar una solución o llegar a ejecutar 1002900 evaluaciones sin encontrar solución alguna. Estas evaluaciones a las que nos referimos se definen como el número de ejecuciones de la función de adecuación que se realizan durante la resolución del problema. Es una medida que tomamos para analizar el coste de resolver el problema.

Se ha resuelto cada problema con cinco tipos distintos de ratios entre la población y el vecindario. Tres de ellos serán ratios estáticos y los otros dos dinámicos, que cambiarán la forma de la rejilla que aloja a la población, manteniéndose la forma del vecindario constante[1] en todo momento.

Las rejillas utilizadas para resolver los problemas de forma estática tienen las formas: 20×20 (*Square*), 10×40 (*Rectangular*) y 4×100 (*Narrow*); mientras que en el caso de rejillas dinámicas preprogramadas se han utilizado los cambios de 20×20 a 4×100 (en adelante nos referiremos a esta rejilla como *SqNar*) y de 4×100 a 20×20 (en adelante *NarSq*). Para elegir el momento de la ejecución en el que se cambia la rejilla se ha seguido el criterio expuesto en [AT00], según el cual se conmuta de ratio a la mitad de la media de evaluaciones necesaria para resolver el problema con una rejilla *Square*.

La rejilla *Square* (de 20×20 individuos) se caracterizan por realizar una elevada explotación de la población, mientras que la *Narrow* (de 4×100) centra más la búsqueda en la exploración de dicha población. Por otro lado la rejilla *Rectangular* (de 10×40) representa un compromiso entre ambas.

Todos los problemas se han ejecutado treinta veces en cada una de las rejillas. Para las estadísticas sólo se tendrán en cuenta los resultados en los que se ha encontrado una solución; sólo se tendrán en cuenta en el caso del cálculo del porcentaje de éxito.

A continuación se pasará a describir, para cada uno de los problemas estudiados, los resultados obtenidos y la interpretación que podemos sacar de dichos resultados.

5.2. Satisfacción de Cláusulas Lógicas (COUNTSAT)

Los valores que hemos obtenido al resolver este problema están representados en la Tabla 5.1. Como en todos los problemas anteriores, se han comparado estos resultados aplicando funciones de análisis estadístico; obteniendo los resultados de la Tabla A.8.

[1] El vecindario está formado por 4 vecinos, los situados al Norte, Este, Oeste y Sur (NEWS) del individuo

Rejilla	Núm. Medio Evaluaciones	Tiempo Medio	Soluciones Encontradas
Square	24326.33 ± 16321.27	9.67 ± 6.43	40 %
Rectangular	25214.82 ± 13076.29	9.77 ± 5.04	56.66 %
Narrow	21736.54 ± 11403.45	8.33 ± 4.38	*80* %
SqNar	22637.27 ± 4688.44	8.50 ± 5.26	36.67 %
NarSq	19720.91 ± 6426.30	7.73 ± 2.51	73.33 %

Tabla 5.1: Resultados obtenidos con COUNTSAT

Se aprecia en los resultados de las tablas 5.1 y A.8 que, tanto la rejilla *Square* como *SqNar* dan muy malos resultados en términos del porcentaje de soluciones encontradas. Esto es debido a que al hacer una explotación exhaustiva de la población es fácil que el algoritmo caiga en el máximo local; correspondiente al caso en el que todas las variables tienen el valor cero. Además de este máximo local es muy difícil que pueda salir el cGA aunque disponga de mecanismos como la mutación y, como se da en el caso de la rejilla dinámica preprogramada, cambie a una rejilla más orientada a la exploración.

En cambio, si nos fijamos en el número medio de evaluaciones o en el tiempo medio invertido para encontrar la solución, se aprecia que no existen diferencias realmente significativas entre los distintos tipos de rejilla; aunque los mejores valores son los obtenidos por la rejilla dinámica preprogramada *NarSq* que, además se da que es la segunda rejilla con mejor porcentaje de soluciones encontradas por detrás de *Narrow*.

También conviene destacar lo elevado que resultan ser los valores de las desviaciones estándar del número medio de evaluaciones para las tres rejillas fijas, que suponen casi el 50 % del valor medio de evaluaciones; mientras que en el caso de las rejillas dinámicas preprogramadas estos valores son bastante menores. Esto nos hace pensar que en el caso de utilizar las rejillas fijas el coste de encontrar la solución depende en gran medida del valor aleatorio que se asigne inicialmente a los individuos. Por el contrario, en el caso de las rejillas dinámicas preprogramadas, el coste de encontrar la solución es mas uniforme en las distintas ejecuciones; independientemente del valor que se les asigne a los individuos en el inicio.

La solución óptima se corresponde con la cadena que asigna a sus 20 variables el valor 1; es decir, la cadena: 11111111111111111111.

5.3. Problema del Diseño de Código Corrector de Errores (ECC)

Podemos ver en la Tabla 5.2 el número medio de evaluaciones utilizado por el cGA para encontrar la solución óptima, junto con la desviación estándar de estos valores, el tiempo empleado para encontrar la solución y su desviación estándar, y el porcentaje de ejecuciones en las que se encontró solución.

Con el fin de obtener una idea comparativa y cuantificada de la diferencia de utilizar una rejilla u otra en la resolución del problema aplicamos un análisis estadístico sobre el número de evaluaciones realizadas para cada una de las rejillas durante cada una de las ejecuciones en las que se encontró solución. Aplicando esta función a dichos resultados obtenemos los datos de la Tabla A.1[2].

Rejilla	Núm. Medio Evaluaciones	Tiempo Medio(seg)	Soluciones Encontradas
Square	159102.40 ± 26566.10	691.43 ± 115.84	100 %
Rectangular	144332.30 ± 28735.89	625.30 ± 124.40	100 %
Narrow	151724.00 ± 25067.11	658.23 ± 112.52	100 %
SqNar	163018.90 ± 28792.33	809.90 ± 143.28	100 %
NarSq	175891.00 ± 36339.49	764.70 ± 158.50	100 %

Tabla 5.2: Resultados obtenidos con ECC

Viendo estos resultados se aprecia que el problema posee un considerable grado de epistasis y de multimodalidad; ya que los mejores resultados se obtienen al utilizar las rejillas más *Rectangulares*. La rejilla cuadrada (*Square*) es la más lenta de las tres rejillas fijas; esto es debido a que el algoritmo se detiene mucho explorando los máximos locales.

Por otro lado, sorprende que la rejilla *NarSq* sea la peor de todas, cuando a priori se podía pensar que pudiera ser la más adecuada. La causa está en que en la primera fase de la ejecución (rejilla *Narrow*) se crean muchos máximos locales, máximos que son luego estudiados exhaustivamente al usar la rejilla *Square*, demorando mucho la aparición de un máximo global.

De todas maneras, es necesario destacar que el algoritmo encontró solución al problema el 100 % de las veces, para cualquier tipo de rejilla utilizada.

Los mejores resultados se obtienen con las rejillas fijas más rectangulares y, particularmente,

[2]En negrita están los resultados significativos (≤ 0.05)

con la rejilla *Rectangular*, debido a que aunque se aproxime con mayor lentitud que *Narrow* a la zona del máximo global, realiza una explotación más exhaustiva de los individuos, lo que le permite encontrar más fácilmente la solución una vez se haya acercado a la zona del espacio de valores en la que se encuentra el máximo global de la función.

Una solución óptima para códigos de 24 palabras de longitud 12 tiene un valor de *fitness* de 0.0674, y está ilustrada en [CFW98].

Por último, se puede apreciar en los resultados obtenidos que para este problema no es deseable la utilización de rejillas dinámicas preprogramadas, ya que resultan ser siempre más lentas que las estáticas.

5.4. Modulación en Frecuencia de Sonidos (Frequency Modulation Sounds - FMS)

Debido a la dificultad de resolución de este problema, el algoritmo para cuando el valor del error cae por debajo de 10^{-2}. Los resultados se detallan en las tablas 5.14 y A.10. Según estas tablas podemos comprobar que los mejores resultados son los obtenidos por la rejilla *NarSq*.

Para este problema, de alta epistasis, es deseable utilizar rejillas de ratio pequeño (es decir, estrechas), debido a que acelera en gran medida la búsqueda y sale rápidamente de los óptimos locales. El cambio dinámico preprogramado de la ratio *SqNar* es el caso que peores resultados ofrece.

5.5. Problema del Corte Máximo de un Grafo (MAXCUT)

Para el estudio completo del problema se ha decidido investigar su comportamiento para tres tipos de grafos distintos: uno de gran dimensión (100 vértices) y dos de tamaño más reducido (20 vértices); uno de los cuales tiene gran densidad de ejes, mientras que la densidad de ejes del otro es bastante más baja.

5.5.1. Grafo Disperso de 20 Vértices

El grafo que vamos estudiar en esta sección se trata de un grafo disperso[3], que ha sido obtenido aleatoriamente.

[3]Grafo disperso es aquel que tiene una baja densidad de ejes

Los resultados obtenidos en la ejecución del EA para este problema se detallan en la Tabla 5.3.
La mejor solución al problema con la instancia cut20.01.dat viene dada por la cadena: 00101010110000011110.

Los resultados de comparar estos valores mediante un análisis estadístico se dan en la Tabla A.2.

Rejilla	Núm. Medio de Evaluaciones	Tiempo Medio	Soluciones Encontradas
Square	18364.80 ± 8135.60	7.73 ± 3.44	100 %
Rectangular	18899.47 ± 9615.20	7.90 ± 4.06	100 %
Narrow	17375.67 ± 7142.58	7.30 ± 3.04	100 %
SqNar	15851.87 ± 8841.27	7.47 ± 4.24	100 %
NarSq	24206.03 ± 10228.13	10.07 ± 4.30	100 %

Tabla 5.3: Resultados obtenidos con MAXCUT20.01

Se puede observar que el mejor resultado de entre los que utilizan rejilla fija se corresponde con la rejilla de forma 4x100, frente a las otras dos, que desprenden resultados muy parecidos. Esto se debe a que al ser disperso el grafo, el cambio en el valor de un alelo del genotipo puede suponer un cambio muy importante en la función de adecuación del individuo, lo que nos indica que existe para este tipo de grafos un cierto grado de epistasis, aunque es bajo.

Por otro lado, vemos que el mejor resultado corresponde a la rejilla *SqNar* lo que contrasta con el hecho de que el peor resultado corresponda, al igual que en el caso del problema ECC, con el otro tipo de rejilla dinámica preprogramada utilizado, el *NarSq*, que, además, es significativamente peor que cualquiera de los otros tipos de rejilla utilizados.

5.5.2. Grafo Denso de 20 Vértices

A continuación se detallan en la Tabla5.4 los valores correspondientes al número medio de evaluaciones para este problema.

Para esta instancia, la mejor solución viene representada por la cadena: 00110101000110101001

Tras comparar estos resultados utilizando un análisis estadístico obtenemos la Tabla A.3. Fijándonos en las tablas 5.4 y A.3 podemos ver que el mejor resultado obtenido en este caso es el correspondiente a la rejilla de forma *Rectangular*, mientras que el peor resultado es el correspondiente a la rejilla *Square*. Cabe destacar que no hay diferencias significativas entre los resultados obtenidos con las distintas rejillas con la instancia cut20.09; aunque que en el caso del grafo dis-

perso con la rejilla *NarSq* se obtienen valores significativamente peores que con las demás. Otro hecho destacable es que, una vez más, el algoritmo encuentra siempre solución.

Rejilla	Núm. Medio de Evaluaciones	Tiempo Medio	Soluciones Encontradas
Square	41703.00 ± 17104.00	17.70 ± 7.33	100 %
Rectangular	36610.30 ± 17106.65	15.50 ± 7.23	100 %
Narrow	37679.63 ± 14234.67	15.77 ± 5.93	100 %
SqNar	42825.80 ± 19916.92	17.93 ± 8.37	100 %
NarSq	40045.53 ± 17741.38	16.70 ± 7.43	100 %

Tabla 5.4: Resultados obtenidos con MAXCUT20.09

Vemos que el número medio de evaluaciones realizado para encontrar solución en el caso del grafo denso es significativamente mayor al necesario en el caso del grafo disperso; hasta el punto de ser mayor del doble en todos los casos excepto en el de la rejilla dinámica preprogramada *NarSq*.

5.5.3. Grafo de 100 Vértices

En la Tabla 5.5 se detallan los resultados obtenidos al resolver el problema con los distintos tipos de rejillas.

La mejor solución que podemos obtener con esta instancia se corresponde con la siguiente cadena de bits: 1110000011011110110100110101011101110100100010001
10101011010110101000010100101111001001000110100100 1.

Al aplicar el análisis estadístico sobre estos resultados obtenemos la Tabla A.4, en la que se aprecia que la mejor solución obtenida en este caso se corresponde con la rejilla intermedia (*Rectangular*), al igual que en el caso de cut20.09. Por el contrario, la peor solución, y con diferencia, es la obtenida por la rejilla *SqNar*.

Si nos fijamos en el número medio de evaluaciones, cabe destacar que la rejilla *Rectangular* es la que ofrece un mejor resultado. Además, el caso maxcut100 coincide con la instancia cut20.09 en el sentido de que la rejilla fija *Rectangular* ofrece un mejor número medio de evaluaciones que todas las la rejillas dinámicas. Por otro lado, *SqNar* es la rejilla con la que se realiza un mayor número medio de evaluaciones.

Fijándonos en el tiempo, observamos que la rejilla que mejor se comporta es, al igual que en el

Rejilla	Núm. Medio Evaluaciones	Tiempo Medio	Soluciones Encontradas
Square	256826.10 ± 105398.00	562.13 ± 230.82	50 %
Rectangular	232141.50 ± 97322.54	516.18 ± 216.71	36.67 %
Narrow	357101.30 ± 249547.70	788.71 ± 552.11	56.67 %
SqNar	415788.80 ± 250513.80	938.12 ± 566.60	56.67 %
NarSq	284057.40 ± 209751.70	633.00 ± 468.44	46.67 %

Tabla 5.5: Resultados obtenidos con MAXCUT100

caso del número medio de evaluaciones, la *Rectangular*, mientras que le peor es *SqNar*.

En cuanto al porcentaje de soluciones encontradas, la que ha encontrado un porcentaje menor a lo largo de las treinta ejecuciones es la rejilla *Rectangular*. Por otro lado, las rejillas que han obtenido mayores valores en este sentido son la *Rectangular* y la *SqNar*. El que tengan un porcentaje mayor de soluciones encontradas nos hace pensar que sean menos propensos a caer en máximos locales (o tienen una mayor facilidad para salir de éstos).

En este problema se observa que los porcentajes de soluciones encontradas en las ejecuciones sobre cualquiera de las rejillas son muy bajos, rondando el 50 %, o incluso menos en algunos casos; lo que nos da una idea del grado de complejidad que llega a representar el problema para el cGA cuando aumenta el tamaño y/o complejidad del grafo. Comparando los resultados obtenidos para los grafos de 20 vértices con los obtenidos para el de 100, nos podemos hacer una idea del crecimiento de la complejidad del problema al incrementar el tamaño del grafo estudiado; frente al grafo cut20.01, que tiene un número 5 veces menor de vértices, el grafo cut100 necesita entre las 62 veces de la rejilla *NarSq* y las 125 de la de *SqNar* el tiempo que utiliza cut20.01. Por tanto, mientras que el número de vértices del grafo se multiplica por 5, el tiempo de ejecución llega a multiplicarse hasta por un valor comprendido entre 61 y 125, dependiendo del tipo de rejilla en el que nos fijemos. Esto refleja un crecimiento bastante rápido del tiempo de ejecución con respecto a la complejidad de la entrada.

5.6. Problema Engañoso Masivamente Multimodal (Massively Multimodal Deceptive Problem - MMDP)

Los resultados obtenidos se detallan en las tablas 5.14 y A.9. Podemos esperar que el cambio dinámico de la ratio no sea deseable en todos los problemas (Teorema NFL) [WM97]. Un ejemplo de obtención de pobres resultados utilizando ratios dinámicas es el del MMDP: tanto en SqNar

como en NarSq ha sido necesario detener el algoritmo al llegar al millón de evaluaciones sin haber podido encontrar solución. Para problemas no epistáticos, como es el caso que nos ocupa, parece que la mejor opción (inicialmente) es el uso de rejillas fijas con elevado ratio. Por tanto, podemos concluir que para problemas engañosos es más eficiente el uso de rejillas que ejerzan una elevada presión sobre los individuos de la población.

Cualquier solución óptima a este problema está formada por P cadenas de 6 bits; cada una de las cuales tendrá sus 6 bits con el mismo valor lógico (los 6 con el valor 0 o los 6 con el valor 1).

5.7. Problema de Tarea con Espera Mínima (MTTP)

Los resultados obtenidos se presentan a continuación en 3 secciones, cada una de las cuales resolverá instancias de dificultad creciente [KBH94], que se han llamado mttp20 (mttp20.dat), mttp100 (mttp100.dat) y mttp200 (mttp200.dat); y que están formados por grupos de 20, 100 y 200 tareas respectivamente.

5.7.1. MTTP con 20 Tareas

Los resultados obtenidos para este problema se detallan en la Tabla 5.6. Hay varias posibles soluciones óptimas al problema; una de ellas se corresponde con la cadena: 01001110111011111111.

Aplicando los análisis estadísticos sobre estos resultados obtenemos los datos de la Tabla A.5.

Rejilla	Núm. Medio de Evaluaciones	Tiempo Medio	Soluciones Encontradas
Square	18017.27 ± 7790.60	7.00 ± 3.06	100 %
Rectangular	17923.70 ± 6283.40	6.87 ± 2.40	100 %
Narrow	19514.33 ± 4644.70	7.47 ± 1.81	100 %
SqNar	15838.50 ± 5779.60	6.77 ± 2.52	100 %
NarSq	19420.77 ± 6326.08	7.53 ± 2.49	100 %

Tabla 5.6: Resultados obtenidos con MTTP20

Vemos que en este caso el mejor resultado nos lo proporciona la rejilla *SqNar*, aunque a priori podríamos haber pensado en la rejilla *NarSq* como la mejor que, curiosamente, es junto a la de 4×100, la peor de las rejillas utilizadas. De cualquier forma, exceptuando los resultados obtenidos por la rejilla *SqNar* no hay en el resto de las rejillas resultados significativamente distintos. Además, para cualquiera de las rejillas estudiadas, los tiempos de ejecución del algoritmo son bastante bajos.

Llama la atención que la rejilla que da los mejores resultados (y, además, con diferencia) sea aquella en la que a priori se pueda pensar como la menos adecuada, ya que en primer término realiza una explotación exhaustiva y local del espacio de soluciones para después pasar a realizar una exploración general de este espacio. Esto nos hace pensar que inicialmente los valores que toman las funciones de adecuación de los individuos son parecidos en magnitud.

El algoritmo encuentra solución en todas sus variantes y ejecuciones.

5.7.2. MTTP con 100 Tareas

En la Tabla 5.7 podemos ver los resultados que se han obtenido tras las ejecuciones del algoritmo.

Aplicando los análisis estadísticos sobre los valores obtenidos en la ejecución del algoritmo construimos la Tabla A.6.

Rejilla	Núm. Medio Evaluaciones	Tiempo Medio	Soluciones Encontradas
Square	294613.70 ± 64594.97	435.50 ± 96.13	100 %
Rectangular	274162.70 ± 44105.24	434.33 ± 71.13	100 %
Narrow	303636.20 ± 48156.21	507.00 ± 80.35	100 %
SqNar	284829.30 ± 70811.24	426.33 ± 105.36	100 %
NarSq	303061.40 ± 59103.52	454.73 ± 88.73	100 %

Tabla 5.7: Resultados obtenidos con MTTP100

Para este caso el mejor resultado coincide con el obtenido para la rejilla *Rectangular*, mientras que los peores resultados son los obtenidos con las rejillas *Narrow* y *NarSq*, que ofrecen resultados casi idénticos. En este caso también se ha encontrado solución en todas las ejecuciones realizadas.

Fijándonos en el número medio de evaluaciones necesario para encontrar la solución podemos comparar esta instancia con mttp20 y vemos que en el caso de mttp100 se dobla aproximadamente al número de evaluaciones empleado con la entrada de 20 tareas; aunque el número de tareas utilizadas en la instancia se quintuplica. En cambio, según el tiempo medio empleado en encontrar la solución, en mttp100 es del orden de 60 veces el necesario en el caso de mttp20, luego el tiempo necesario para realizar una evaluación se incrementa en un orden de 30 al quintuplicar el número de tareas a planificar; este incremento se debe a un aumento en el coste de evaluación de la función de adecuación. Con estos datos podemos imaginar que en el caso de mttp200, que estudiaremos a continuación, los tiempos de ejecución se incrementarán aún más. Por tanto, estaremos ante un

problema realmente costoso de resolver.

Al igual que en el caso de mttp20, son otra vez las rejillas *Rectangular* y *SqNar* las que ofrecen los mejores resultados; aunque en este caso la única rejilla con resultados significativamente mejores a las demás es *Rectangular*.

5.7.3. MTTP con 200 Tareas

Los resultados de las distintas ejecuciones realizadas sobre este problema se recopilan en la Tabla 5.8.

Aplicando el análisis estadístico sobre los valores obtenidos en la ejecución del algoritmo construimos la Tabla A.7.

Rejilla	Núm. Medio Evaluaciones	Tiempo Medio	Soluciones Encontradas
Square	673598.80 ± 118540.50	1924.57 ± 339.81	100 %
Rectangular	688850.20 ± 118191.10	2174.26 ± 454.54	100 %
Narrow	566999.60 ± 72818.58	1625.50 ± 208.40	100 %
SqNar	521419.30 ± 74260.20	1474.03 ± 208.65	100 %
NarSq	572132.40 ± 74941.82	1612.93 ± 211.73	100 %

Tabla 5.8: Resultados obtenidos con MTTP200

El mejor resultado de los obtenidos es el que proporciona la rejilla de *SqNar*; aunque *NarSq* ofrece buenos resultados también. Hay que destacar que el número medio de evaluaciones es bastante elevado para cualquiera de las rejillas, sobre todo para las estáticas, lo que indica que se trata de un problema bastante duro; pero el GA encuentra soluciones en el 100 % de las ejecuciones que realiza.

Al igual que en los dos casos anteriores, la rejilla de *SqNar* obtiene muy buenos resultados (los mejores en este caso), pero sorprende que *Rectangular*, que ha dado buenos resultados en las dos instancias anteriores (mttp20 y mttp100), ofrezca para el caso del mttp200 los peores resultados de todos. Además de la rejilla de *SqNar*, también dan buenos resultados las rejillas *Narrow* y *NarSq* que curiosamente, en los casos de mttp20 y mttp100 eran las que daban peores resultados. A partir de estos datos podemos pensar que ya en el caso de 200 tareas el grado de epistasis del problema aumenta hasta llegar a un nivel ya bastante considerable respecto al que había en los casos de 20 y 100 tareas.

Por último, destacar que, tal y como se anunciaba en la sección anterior, tanto el número medio de evaluaciones como el tiempo medio de ejecución aumentan en mttp200 con respecto al caso de mttp100, aunque en menor medida que el cambio experimentado entre mttp100 y mttp20: los tiempos de ejecución se han triplicado, o incluso cuadriplicado en algún caso, mientras que en el caso de la comparación entre mttp20 y mttp100 se multiplicaron por un orden de 60 unidades.

5.8. El Problema de las P Cimas (P-PEAKS)

Los resultados obtenidos con este problema están en las tablas 5.14 y A.11. Al igual que en el caso de FMS, es deseable utilizar rejillas de bajo ratio, que acelera la búsqueda permitiendo salir rápidamente de los máximos locales. Se aprecia que en el caso de P-PEAKS, el efecto de cambiar el ratio durante la ejecución incrementa la eficiencia del algoritmo.

El resultado de una cadena óptima será la cadena que esté formada por la misma secuencia de valores que cualquiera de los picos que se generan aleatoriamente al comenzar la ejecución.

5.9. Estudio del Comportamiento de las Rejillas

En esta sección se estudiarán los resultados presentados las secciones anteriores, pero en este caso desde el punto de vista de los resultados obtenidos por todos los problemas para cada una de las rejillas en lugar de estudiar el comportamiento del cGA utilizando las distintas rejillas para cada problema.

5.9.1. Rejilla Cuadrada: *Square*

En la Tabla 5.9 podemos ver los resultados obtenidos por cada uno de los problemas cuando utilizan la rejilla *Square* para buscar la solución.

El problema COUNTSAT destaca por el bajo porcentaje de soluciones encontradas, que se debe a la existencia de un máximo local muy próximo en magnitud al máximo global, de manera que una vez que el algoritmo se acerque a dicho máximo, le será muy difícil salir de él. Cabe destacar lo importantes que son las desviaciones típicas obtenidas para este problema, esto es debido a la gran desigualdad que hay entre los valores de las distintas ejecuciones.

Por el contrario, con los problemas ECC, MAXCUT20.01 y MAXCUT20.02 se encuentra solución en todas las ejecuciones.

Problema	Núm. Medio de Evaluaciones	Tiempo Medio	Soluciones Encontradas
COUNTSAT	24326.33 ± 16321.27	9.67 ± 5.26	40 %
ECC	159102.4 ± 26566.1	691.43 ± 115.84	100 %
FMS	612112.1 ± 189988.9	888.78 ± 277.73	90 %
MAXCUT20.01	18364.8 ± 8135.6	7.73 ± 3.44	100 %
MAXCUT20.09	41703.0 ± 17104.0	17.7 ± 7.33	100 %
MAXCUT100	256826.1 ± 105398.1	562.13 ± 230.82	50 %
MMDP	170637.9 ± 19139.93	599.36 ± 166.91	93.33 %
MTTP20	18017.27 ± 7790.6	7.0 ± 3.06	100 %
MTTP100	294613.7 ± 64594.97	435.5 ± 96.13	100 %
MTTP200	673598.8 ± 118540.5	1924.57 ± 339.81	100 %
P-PEAKS	50458.1 ± 4295.84	162.6 ± 13.9	100 %

Tabla 5.9: Resultados obtenidos con la rejilla *Square*

También resulta bajo el porcentaje de soluciones encontradas en el caso de MAXCUT100, que sólo encuentra solución en el 50 % de las ejecuciones debido al gran número de máximos locales que tiene, lo que hace que el algoritmo pierda mucho tiempo en explorarlos. Al igual que en el caso del problema COUNTSAT, MAXCUT100 obtiene desviaciones medias bastante importantes en sus valores.

El problema MMDP obtiene solución en casi todas sus ejecuciones, siendo su porcentaje del 93.33 %.

Toda las instancias de MTTP consiguen encontrar solución en el 100 % de las ejecuciones. Cabe destacar el notable incremento de la complejidad al aumentar el número de variables consideradas en cada instancia: al incrementar el número de variables de 20 a 100 (se multiplica por 5), el número de evaluaciones aumenta en un factor de 16 veces, mientras que el tiempo medio se multiplica por 62. Por otro lado, al cambiar el número de variables utilizadas en la instancia de 100 a 200, el número medio de evaluaciones también se duplica, y el tiempo medio se cuadruplica.

En cuanto a FMS cabe decir que posee un porcentaje del 90 % de soluciones encontradas.

Por último, sólo cabe decir que P-PEAKS consigue encontrar solución en todas sus ejecuciones.

5.9.2. Rejilla Rectangular: *Rectangular*

Comparando los valores de las tablas 5.10 y 5.9 se observa que al disminuir el ratio de la población con respecto al vecindario, la evolución del porcentaje de soluciones encontradas se comporta de manera distinta en los problemas MAXCUT100 y MDDP que en el caso de los problemas COUNTSAT y FMS; mientras que los demás problemas mantienen el 100 % de soluciones encontradas.

El problema COUNTSAT ve incrementado su porcentaje de soluciones encontrado desde un 40 % hasta el 56.66 %.

Tanto ECC como las dos instancias más pequeñas de MAXCUT (cut20.01 y cut20.09) continúan manteniendo el 100 % de las soluciones encontradas.

El caso del problema MAXCUT100 es distinto: el porcentaje de soluciones encontradas baja de un 50 % hasta un 36.67 %; en contraste con el caso de COUNTSAT, en el que dicho porcentaje subió.

Como en MAXCUT100, en MMDP se disminuye el porcentaje de soluciones encontradas del 93.33 % al 86.67 %.

Al igual que con *Square*, también con esta rejilla obtenemos el 100 % de soluciones encontradas en todas las instancias de MTTP.

FMS aumenta ligeramente su porcentaje de soluciones encontradas de un 90 % a un 93.33 %.

Por último, P-PEAKS se mantiene en el 100 % de soluciones encontradas.

Se puede concluir comparando estos resultados con los de la rejilla *Square* que con la rejilla *Rectangular* se disminuye el número medio de evaluaciones realizadas en los problemas que posean un elevado número de máximos locales en su espacio de búsqueda de soluciones, como es el caso de los problemas MAXCUT y MMDP.

5.9.3. Rejilla Estrecha: *Narrow*

Como se puede comprobar en la Tabla 5.11, todos los problemas que poseían un 100 % de soluciones encontradas (ECC, MAXCUT20.01, MAXCUT20.09, cualquier instancia de MTTP y P-PEAKS) continúan manteniendo dicho porcentaje. Por otro lado, en los casos en los que no se encontraba solución en todas las ejecuciones se observan comportamientos distintos.

Problema	Núm. Medio de Evaluaciones	Tiempo Medio	Soluciones Encontradas
COUNTSAT	25214.82 ± 13076.29	9.77 ± 5.04	56.66 %
ECC	144332.3 ± 28735.89	625.3 ± 124.4	100 %
FMS	594093.9 ± 143506.4	852 ± 214.18	93.33 %
MAXCUT20.01	18899.47 ± 9615.2	7.9 ± 4.06	100 %
MAXCUT20.09	36610.3 ± 17106.65	15.5 ± 7.23	100 %
MAXCUT100	232141.5 ± 97322.54	516.18 ± 216.71	36.67 %
MMDP	237003.4 ± 77210.15	648.5 ± 274.12	86.67 %
MTTP20	17923.7 ± 6283.4	6.87 ± 2.4	100 %
MTTP100	274162.7 ± 44105.24	434.33 ± 71.13	100 %
MTTP200	688850.2 ± 118191.1	2174.26 ± 454.54	100 %
P-PEAKS	50364.6 ± 5372.35	160.77 ± 17.39	100 %

Tabla 5.10: Resultados obtenidos con la rejilla *Rectangular*

Comparando la rejilla *Narrow* con la utilizada en el apartado 5.9.2 el problema COUNTSAT sigue aumentando el número de soluciones encontradas conforme va disminuyendo la ratio de la población con respecto al vecindario, resolviendo en este caso un 80 % de las ejecuciones realizadas.

En cambio, el problema MAXCUT100 ve ahora incrementado el porcentaje de soluciones encontradas desde el 36.67 % hasta un 56.67 %, aproximadamente el que poseía en el caso de la rejilla *Square*.

FMS disminuye ligeramente su porcentaje de soluciones encontradas desde un 93.33 % a un 86.67 % y MMDP lo mantiene en el 86.67 % del caso de la rejilla *Rectangular*.

Estos comportamientos nos confirman que la rejilla *Narrow* es apropiada para aquellos problemas altamente epistáticos y en aquellos para los que se presenten máximos locales muy próximos en valor a los máximos globales.

También en este caso, al igual que en *Square* y *Rectangular*, tanto las tres instancias estudiadas del problema MAXCUT como el problema COUNTSAT poseen desviaciones típicas bastante importantes en el número medio de evaluaciones realizadas.

La mejora obtenida al utilizar esta rejilla con el problema COUNTSAT es tan importante que incluso llega a obtener un mejor rendimiento que el problema MTTP20, que se ejecutaba en menor

tiempo en los casos de las rejillas *Square* y rectangular. Este hecho obtiene una mayor relevancia cuando, además, observamos que esta rejilla ofrece una mejora en las tres instancias del problema MTTP en comparación con los otros problemas.

Por último, destacar que, como era de esperar, el problema MMDP continúa empeorando su rendimiento con respecto a los otros problemas al estrechar la rejilla que distribuye a la población.

Problema	Núm. Medio de Evaluaciones	Tiempo Medio	Soluciones Encontradas
COUNTSAT	21736.54 ± 11403.45	8.33 ± 4.38	80 %
ECC	151724.0 ± 25067.11	658.23 ± 112.52	100 %
FMS	593519.1 ± 167334	807.73 ± 273.67	86.67 %
MAXCUT20.01	17375.67 ± 7142.578	7.3 ± 3.04	100 %
MAXCUT20.09	37679.63 ± 14234.67	15.77 ± 5.93	100 %
MAXCUT100	357101.3 ± 249547.7	788.71 ± 552.11	56.67 %
MMDP	331719.6 ± 86401.57	823.5 ± 256.24	86.67 %
MTTP20	19514.33 ± 4644.7	7.47 ± 1.81	100 %
MTTP100	303636.2 ± 48156.21	507 ± 80.35	100 %
MTTP200	566999.6 ± 72818.58	1625.5 ± 208.4	100 %
P-PEAKS	48653.6 ± 3590.23	147.33 ± 11.14	100 %

Tabla 5.11: Resultados obtenidos con la rejilla *Narrow*

5.9.4. Rejilla Dinámica Preprogramada Cuadrada/Estrecha: *SqNar*

Con esta rejilla obtenemos los peores porcentajes de solución hallados con cualquiera de las rejillas anteriores para el caso del problema COUNTSAT, que es del 36.67 % frente al 40 %, que era el peor de los obtenidos en las anteriores rejillas.

Como hasta ahora, los problemas ECC, MAXCUT20.01, MAXCUT20.09, P-PEAKS y cualquiera de las instancias de MTTP mantienen porcentajes del 100 % en las soluciones encontradas.

En el caso del problema MAXCUT100 se mantiene el porcentaje obtenido con la rejilla *Narrow*, que es el mayor encontrado en las rejillas estudiadas hasta ahora.

Destaca que para este tipo de rejilla el problema MMDP no encuentra nunca solución antes de las 1002900 evaluaciones (ver Tabla 5.12), mientras que sí lo hacía con las rejillas fijas.

Por último, el problema FMS obtiene un porcentaje de soluciones del 93.33 %, el mismo que en el caso de *Rectangular*, que coincide con el mayor porcentaje encontrado para este problema utilizando las rejillas estudiadas hasta ahora.

También coinciden los resultados de esta rejilla con los anteriores en el hecho de que los problemas MAXCUT y COUNTSAT son los que cuentan con las desviaciones típicas más importantes.

Son de resaltar los buenos resultados que da el problema MTTP (tanto con 20 como con 100 o 200 variables) con esta rejilla, ya que se reduce el número medio de evaluaciones, mientras que en el resto de los problemas este número aumenta en el orden de un 10 % o un 15 %; como era de esperar a priori, ya que esta rejilla comienza realizando una explotación exhaustiva y local para pasar luego a una exploración más general del espacio de búsqueda.

Al utilizar esta rejilla vemos que tanto para el problema P-PEAKS como para FMS se ve incrementado significativamente (del orden de 3 ó 4 veces) el tiempo medio de ejecución, mientras que el número medio de evaluaciones permanece aproximadamente constante.

Problema	Núm. Medio de Evaluaciones	Tiempo Medio	Soluciones Encontradas
COUNTSAT	22637.27 ± 4688.44	8.5 ± 5.26	36.67 %
ECC	163018.9 ± 28792.33	809.9 ± 143.28	100 %
FMS	692205.2 ± 195807.8	2489.71 ± 657.02	93.33 %
MAXCUT20.01	15851.87 ± 8841.269	7.47 ± 4.24	100 %
MAXCUT20.09	42825.8 ± 19916.92	17.93 ± 8.37	100 %
MAXCUT100	415788.8 ± 250513.8	938.12 ± 566.6	56.67 %
MMDP	1002900.0	4656.9 ± 789	0 %
MTTP20	15838.5 ± 5779.6	6.77 ± 2.52	100 %
MTTP100	284829.3 ± 70811.24	426.33 ± 105.36	100 %
MTTP200	521419.3 ± 74260.2	1474.03 ± 208.65	100 %
P-PEAKS	57769.7 ± 4814.23	483.5 ± 40.17	100 %

Tabla 5.12: Resultados obtenidos con la rejilla *SqNar*

5.9.5. Rejilla Dinámica Preprogramada Estrecha/Cuadrada: *NarSq*

Al igual que en todas las rejillas estudiadas, ECC, MAXCUT20.01, MAXCUT20.09, MTTP20, MTTP100, MTTP200 y P-PEAKS continúan manteniendo el 100 % de soluciones encontradas.

COUNTSAT, encuentra el segundo mayor porcentaje de soluciones de todas las rejillas estudiadas, con un 73.33 % , ya que el máximo encontrado es el 80 %, que se corresponde con la rejilla *Narrow*.

Como en el caso *SqNar*, el problema MMDP continúa sin obtener solución en ninguno de los 30 intentos.

En MAXCUT100 se reduce el porcentaje de soluciones encontradas, con respecto a SqNar, de un 56.67 % a un 46.67 %.

En cambio, y sólo en el caso de esta rejilla, el problema FMS encuentra solución en el 100 % de las veces.

Cabe destacar que en este caso las instancias del problema MAXCUT continúan teniendo desviaciones típicas bastante importantes en el número medio de evaluaciones, pero, en cambio, el problema COUNTSAT ve reducidos de manera notable estos valores; de hecho, para este problema se incrementa el rendimiento al utilizar esta rejilla para distribuir a la población.

Por otro lado, al igual que en la otra rejilla dinámica preprogramada, el problema MMDP no es capaz de encontrar en ninguna ocasión solución antes de 1002900 ejecuciones.

En cuanto al problema MAXCUT, es importante destacar el hecho de que para las instancias más simples (las dos de 20 variables), los resultados son similares a los de la otra rejilla dinámica preprogramada, pero en cambio los resultados obtenidos por la instancia de 100 variables son mucho mejores.

Al igual que en la otra rejilla dinámica preprogramada, los problemas FMS y P-PEAKS ven incrementado notablemente el tiempo medio de ejecución, mientras que el número medio de evaluaciones permanece en los mismos valores que en el caso de las rejillas fijas.

5.10. Resumen del Capítulo

En la Tabla 5.10 se recopila la información obtenida durante todo el capítulo referente al nómero medio de evaluaciones y el tiempo medio utilizado por los diversos problemas para obtener la solución. Las medias están obtenidas sobre un total de 30 ejecuciones para cada programa.

Problema	Núm. Medio de Evaluaciones	Tiempo Medio	Soluciones Encontradas
COUNTSAT	19720.91 ± 6426.3	7.73 ± 2.51	73.33 %
ECC	175891.0 ± 36339.49	764.7 ± 158.5	100 %
FMS	575353.8 ± 167489.3	2486.33 ± 674.6	100 %
MAXCUT20.01	24206.03 ± 10228.13	10.07 ± 4.3	100 %
MAXCUT20.09	40045.53 ± 17741.38	16.7 ± 7.43	100 %
MAXCUT100	284057.4 ± 209751.7	633 ± 468.44	46.67 %
MMDP	1002900.0	2942.6 ± 7.46	0 %
MTTP20	19420.77 ± 6326.08	7.53 ± 2.49	100 %
MTTP100	303061.4 ± 59103.52	454.73 ± 88.73	100 %
MTTP200	572132.4 ± 74941.82	1612.93 ± 211.73	100 %
P-PEAKS	48012.1 ± 4376.8	452.43 ± 37.03	100 %

Tabla 5.13: Resultados obtenidos con la rejilla *NarSq*

Prob\Crit	*Square*	*Rectangular*	*Narrow*	*SqNar*	*NarSq*
COUNTSAT	24326.33; 9.67	25214.82; 9.77	21736.54; 8.33	22637.27; 4688.44	**19720.91; 7.73**
ECC	159102.40; 691.43	**144332.30; 625.30**	151724.00; 658.23	163018.90; 28792.33	175891.00; 764.70
FMS	612112.10	594093.90	593519.10	692205.2	**575353.80**
MAXCUT20.01	18364.80; 7.73	18899.47; 7.90	17375.67; 7.30	**15851.87; 7.47**	24206.03; 10.07
MAXCUT20.09	41703.00; 17.70	**36610.30; 17106.65**	37679.63; 14234.67	42825.80; 17.93	40045.53; 16.70
MAXCUT100	256826.10; 562.13	**232141.50; 516.18**	357101.30; 788.71	415788.80; 938.12	284057.40; 633.00
MMDP	**170637.90**	237003.40	331719.60	1002900.00	1002900.00
MTTP20	18017.27; 7.00	17923.70; 6.87	19514.33; 7.47	**15838.50; 6.77**	19420.77; 7.53
MTTP100	294613.70; 435.50	**274162.70; 434.33**	303636.20; 507.00	284829.30; 426.33	303061.40; 454.73
MTTP200	673598.80; 1924.57	688850.20; 2174.26	566999.60; 12625.50	**521419.30; 1474.03**	572132.40; 74941.82
P-PEAKS	50458.10	50364.60	48653.60	57769.70	**48012.10**

Tabla 5.14: Resumen del Núm. med. de evaluaciones y tiempo med. para cada problema con rejillas fijas y dinámicas con cambio preprogramado

Capítulo 6

Selección del Criterio Dinámico Auto-Adaptativo a Utilizar

Las estrategias existentes para el control de parámetros en EAs se pueden clasificar como sigue [Bäck01]:

- *Control dinámico de parámetros*: Los valores de los parámetros se modifican de acuerdo a una estrategia planificada por el diseñador del EA.
- *Control adaptativo de parámetros*: Los nuevos valores de los parámetros de control se obtienen por un mecanismo de retroalimentación que monitoriza la evolución y premia o penaliza los valores de los parámetros en función de si han causado un incremento o deterioramiento al valor de la función objetivo.
- *Control auto-adaptativo de parámetros*: Los valores de los parámetros de control se desarrollan por el EA mediante la aplicación de operadores evolutivos a los parámetros de control de una manera similar a las representaciones de soluciones.

Una vez estudiados los problemas utilizando las rejillas fijas y dinámicas preprogramadas, podemos pensar en desarrollar un criterio que vaya modificando la forma de la rejilla que compone la población automáticamente durante toda la ejecución según la evolución que vayan sufriendo los individuos que la forman. A este tipo de criterio lo llamaremos *criterio auto-adaptativo* o *rejilla dinámica auto-adaptativa*. Dejándonos llevar por esta idea se realizará en este capítulo el estudio de varios criterios de este tipo, a los que nombraremos auto-adaptativos, utilizando para ello dos posibles parámetros:

- Valor de adecuación medio de los individuos, que supone una medida de calidad de las soluciones representadas por los individuos en un instante.

- Valor de la entropía de la población, que representa una medida de la diversidad de soluciones representadas por todos los individuos de la población.

A continuación se proponen cuatro criterios auto-adaptativos muy parecidos, con el fin de estudiar su comportamiento para realizar la elección del criterio con el que estudiaremos el comportamiento de todos los problemas en el capítulo siguiente.

Estos cuatro criterios se basan en el estudio de la media del valor de fitness de los individuos que forman la población y comienzan utilizando la rejilla de menor ratio posible (la más rectangular), cambiando su forma automáticamente según se va resolviendo el problema. Siendo $\overline{f}_t$, $\overline{f}_{t-1}$, $\overline{f}_{t+1}$ la media de los fitness de los individuos de la población en el tiempo t, el inmediatamente anterior y el inmediatamente siguiente (ver Figura 6.1), los cuatro criterios se detallan a continuación en las secciones que siguen.

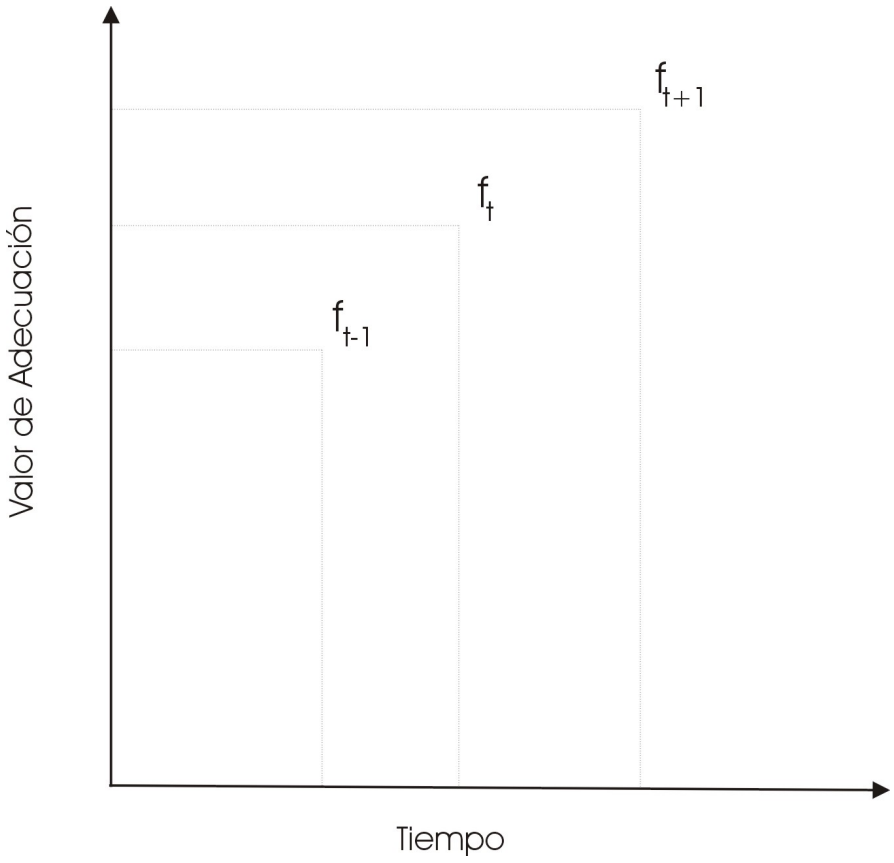

Figura 6.1: Posibles valores de la función de adecuación en distintos tiempos

Cuando cualquiera de los criterios alcanza una rejilla extrema, bien la *Square* o bien la más estrecha, y el siguiente cambio de rejilla necesita rebasar ese límite (cambiar a una rejilla más cuadrada estando ya en la cuadrada o a una más rectangular estando ya en la más estrecha), el

criterio se mantiene en dicha rejilla extrema, es decir, sin realizar ningún cambio.

Todos los criterios se basan en la velocidad de cambio del valor del parámetro estudiado: el fitness medio de los individuos o la entropía de la población, según sea el caso.

6.1. Criterio AvgFitεR

La evaluación de este criterio se describe en los siguientes puntos:

1. Comenzar con la rejilla más rectangular
2. Si $(\overline{f}_{t+1} - \overline{f}_t) < \varepsilon(\overline{f}_t - \overline{f}_{t-1})$ entonces usar siguiente rejilla más cuadrada
3. Si $(\overline{f}_{t+1} - \overline{f}_t) > (1-\varepsilon)(\overline{f}_t - \overline{f}_{t-1})$ entonces usar la siguiente rejilla más rectangular

Básicamente, lo que hace el criterio es lo siguiente: Sean t+1, t y t-1 los tiempos correspondientes a la generación actual, la anterior y la dos veces anterior. Si el cambio del valor medio de adecuación en el tiempo t+1 con respecto a su valor en el tiempo t es menor que la diferencia de este valor en el tiempo t con respecto a su valor en el tiempo t-1, multiplicado por un cierto valor de ε, se cambia la rejilla actual por la inmediatamente más cuadrada; si la actual es la rejilla cuadrada, entonces no se hace nada. En caso de que este cambio sea mayor que la diferencia en el valor medio de adecuación en las generaciones t y t-1, se cambiará por la siguiente rejilla inmediatamente más rectangular, siempre que no nos encontremos ya utilizando la más estrecha.

6.2. Criterio AvgFitrR

El comportamiento del criterio AvgFitrR se describe a continuación:

1. Comenzar con la rejilla más rectangular
2. Si $(\overline{f}_{t+1} - \overline{f}_t) < (1+\varepsilon)(\overline{f}_t - \overline{f}_{t-1})$ entonces usar siguiente rejilla más cuadrada
3. Si $(\overline{f}_{t+1} - \overline{f}_t) > (1-\varepsilon)(\overline{f}_t - \overline{f}_{t-1})$ entonces usar la siguiente rejilla más rectangular

En el caso de este criterio siempre habrá cambio de rejilla; a menos que nos encontremos en una de las rejillas extremas y no sea posible realizar el cambio necesario. Este criterio cambiará a la siguiente rejilla más rectangular siempre que se cumpla que la diferencia entre el valor medio de adecuación de las generaciones t+1 y t sea mayor que la misma diferencia, pero en el caso de las generaciones t y t-1.

6.3. Criterio AvgFitsR

El criterio AvgFitsR funciona de la siguiente manera:

1. Comenzar con la rejilla más rectangular
2. Si $(\overline{f}_{t+1} - \overline{f}_t) < (1+\varepsilon)(\overline{f}_t - \overline{f}_{t-1})$ entonces usar siguiente rejilla más cuadrada
3. En otro caso si $(\overline{f}_{t+1} - \overline{f}_t) > (1-\varepsilon)(\overline{f}_t - \overline{f}_{t-1})$ entonces usar la siguiente rejilla más rectangular

Al igual que en el caso del criterio AvgFitrR, siempre que el cambio requerido sea posible (si no nos salimos de las rejillas extremas) habrá cambio de rejilla; pero, en este caso, dicho cambio se realizará hacia la siguiente rejilla más cuadrada, siempre que se cumpla que la diferencia entre el valor medio de adecuación de las generaciones t+1 y t sea menor que la misma diferencia, pero en el caso de las generaciones t y t-1.

La diferencia entre los criterios $AvgFitrR$ y $AvgFitsR$ estriba en que como ambas condiciones se solapan para cualquier valor mayor que cero de ε (es decir se cumplen las dos condiciones del criterio), en estos casos el criterio $AvgFitrR$ se decide por utilizar la siguiente rejilla más rectangular, mientras que el criterio $AvgFitsR$, al utilizar la sentencia Én otro caso si'se decidirá por la siguiente más cuadrada, ya que al cumplir la primera condición del criterio ya no evalúa la segunda.

6.4. Criterio AvgFitR

Por último, el criterio AvgFitR funciona como sigue:

1. Comenzar con la rejilla más rectangular
2. Si $(\overline{f}_{t+1} - \overline{f}_t) < (1+\varepsilon)(\overline{f}_t - \overline{f}_{t-1})$ entonces usar siguiente rejilla más cuadrada
3. Si $(\overline{f}_{t+1} - \overline{f}_t) > (2-\varepsilon)(\overline{f}_t - \overline{f}_{t-1})$ entonces usar la siguiente rejilla más rectangular

Estos tres puntos vienen a decir que si la diferencia entre los valores medios de adecuación de los individuos entre las generaciones t+1 y t crece en un valor igual o mayor al asignado a ε el criterio decide elegir la siguiente rejilla más rectangular (si estamos ya en la más estrecha no hace nada); mientras que si esta diferencia decrece en un valor mayor o igual a ε, se decide elegir la siguiente rejilla más cuadrada.

Capítulo 7

Elección del Mejor Criterio Auto-Adaptativo

El estudio de estos criterios se ha realizado sobre los problemas MAXCUT100 y MTTP200, que se han escogido como los más representativos ya que son instancias que, teóricamente, deben darnos resultados fiables, aunque se trata de una apuesta algo arriesgada que ha sido necesario adoptar debido a la gran envergadura del proyecto. Tras realizar este estudio, se repetirá en la Sección 7.2, pero en éste caso sólo con los dos criterios que han dado mejor resultado en la Sección 7.1, y aplicados al caso de la entropía de la población. De nuevo somos conscientes del riesgo que corremos al hacer tal simplificación en nuestro estudio.

Para comparar los criterios se han ejecutado 10 veces los problemas MAXCUT100 y MTTP200 para distintos valores de ε y se han obtenido las medias de estos resultados. Es conveniente destacar que los valores reflejados en las tablas son obtenidos como la media de tan solo 10 ejecuciones, por lo que los resultados pueden no ser muy fiables.

Los posibles cambios de rejilla se realizan cada 10 pasos del algoritmo, o lo que es lo mismo, cada 4000 evaluaciones, ya en que cada paso se realiza una evaluación por cada uno de los 400 individuos. En las tablas presentadas a continuación se especifica la tasa del número de cambios de rejilla realizados por el algoritmo durante todas las ejecuciones con respecto al número total de cambios estudiados (número de veces que se evalúa el criterio). Para calcular las veces que el algoritmo estudia la realización de un posible cambio, bastará con dividir el número medio de evaluaciones por 4000.

7.1. Estudio Según la Media del Fitness

Los valores de ε utilizados en el estudio han sido elegidos cuidadosamente como los más representativos tras realizar multitud de pruebas (ver Apéndice B).

7.1.1. Problema MAXCUT100

En las tablas C.1, C.2, C.3 y C.4 podemos ver los resultados obtenidos por el problema al resolverlo por cada uno de los cuatro criterios.

En ellas vemos que el mejor resultado (según el número medio de evaluaciones) obtenido al utilizar cualquiera de los criterios mejora los obtenidos en el capítulo anterior para las rejillas fijas y dinámicas preprogramadas. En cambio, si nos fijamos en el tiempo medio empleado en resolver el problema, tan sólo el criterio AvgFitsR con $\varepsilon = 0.05$ mejora dichos resultados; aunque este resultado pertenece a la única ejecución en la que se obtuvo solución de los diez intentos realizados. En cuanto al porcentaje medio de cambios de rejilla por ejecución, todas las pruebas desprenden valores similares, exceptuando el caso del criterio AvgFitsR con $\varepsilon = 0.05$, que es del orden de diez veces mayor que en los demás casos. Por último, si nos fijamos en el porcentaje de soluciones alcanzadas, se aprecia que el máximo porcentaje de cada tabla es superior al máximo obtenido con cualquiera de las rejillas fijas o dinámicas preprogramadas; al igual que el peor resultado en el caso de AvgFitεR.

Es lógico que los criterios auto-adaptativos tengan una media de tiempo superior a la de los casos de rejilla fija y dinámica preprogramada (a pesar de realizar un menor número medio de evaluaciones), ya que realizan varios cambios de rejilla y multitud de evaluaciones del criterio durante la resolución del problema frente a uno y ninguno de los casos de rejilla dinámica preprogramada y fija, respectivamente.

De entre los criterios estudiados en esta sección, el que mejores resultados ofrece tanto en el número medio de evaluaciones como en el tiempo medio de ejecución y en el porcentaje de soluciones encontradas es AvgFitεR, aunque, por otro lado, este criterio posee unas altas desviaciones estándar en sus valores. Otro criterio que ofrece buenos resultados y con desviaciones estándar más bajas es AvgFitR, que destaca especialmente en los resultados que obtiene cuando ε adopta los valores 0.05 y 0.15.

Como conclusión, observamos que, para el caso del problema MAXCUT, los criterios AvgFitεR

y AvgFitR obtienen valores parecidos, con mejores (más pequeñas) desviaciones estándar para AvgFitR y mejores porcentajes de solución para el criterio AvgFitεR.

Al realizar tan solo 10 ejecuciones, se considera que es mucho más fiable y significativo el valor obtenido en las desviaciones estándar que el del porcentaje de soluciones encontradas; por tanto, concluimos que el mejor criterio de los utilizados con le problema MAXCUT100 es **AvgFitR**.

7.1.2. Problema MTTP200

En esta sección se realizará el mismo estudio que el de la Sección 7.1.1 pero resolviendo en este caso el problema MTTP200. Se considera que los resultados de la comparación obtenidos en esta sección son más significativos por tratarse de un problema más costoso de resolver, pero que se resuelve casi siempre, y que, además, obtiene unos valores con desviaciones estándar mucho menores que las obtenidas con MAXCUT100; lo que quiere decir que la diferencia entre los resultados obtenidos en todas las ejecuciones está comprendida en un rango de valores más estrecho.

Comparando los resultados obtenidos en las tablas C.5, C.6, C.7 y C.8 con los de las tablas de Apéndice A (que contiene las tablas con los resultados de utilizar las rejillas fijas y dinámicas preprogramadas) se observa, en referencia al número medio de evaluaciones realizado, que el criterio AvgFitεR ofrece bastante peores resultados que cualquiera de los casos presentados en dicho apéndice. Por otro lado, el criterio AvgFitrR obtiene resultados del orden de los de las tablas del apéndice para todos sus ε, con la salvedad del caso $\varepsilon = 0.3$, que cuenta con un número medio de evaluaciones bastante mayor. También cuenta el criterio AvgFitsR con un caso (aquel en el que $\varepsilon = 0.05$) con pobres resultados en este campo, pero el resto de resultados resultan ser bastante buenos, ya que el peor de ellos es mejor que el mejor de los de las tablas del Apéndice A. Por último, el criterio AvgFitR da unos resultados claramente mejores que el resto de criterios y, por supuesto, que las tablas del apéndice mencionado. Si nos fijamos en el tiempo medio, vemos que las comparaciones desprenden los mismos resultados: AvgFitεR es el criterio con peores valores, AvgFitrR tiene valores del mismo orden que las tablas correspondientes a las rejillas fijas y dinámicas preprogramadas, salvo en los casos $\varepsilon = 0.3$ y, a diferencia del caso del estudio del número medio de evaluaciones, también $\varepsilon = 0.25$. AvgFitsR ofrece resultados del orden de los de las tablas del Apéndice A salvo en el caso $\varepsilon = 0.05$, que da un valor bastante peor. Por último, y como en el caso del estudio según el número medio de evaluaciones, el criterio AvgFitR es el que mejores resultados da; aunque en este caso, la diferencia con respecto a los de las rejillas del apéndice mencionado no es tan notable. En cuanto al porcentaje de cambio s de rejilla, cabe destacar que en todos los criterios salvo en el caso de AvgFitrR son bastante bajos; del orden del 4 % en los casos de

AvgFitεR y AvgFitsR y del 10 ó 15 % en el caso de AvgFitR. En este campo es de destacar la gran diferencia entre el valor de $\varepsilon = 0.05$ y el resto en el criterio AvgFitrR. En cuanto al porcentaje de soluciones encontradas, cabe destacar que, a diferencia del caso de las rejillas fijas y dinámicas preprogramadas, se dan tres casos en los que no se hallan el 100 % de las soluciones. Estos casos se corresponden con los valores $\varepsilon = 0.05$ y $\varepsilon = 0.15$ del criterio AvgFitεR (con un 40 % y un 80 % de soluciones encontradas, respectivamente) y el valor $\varepsilon = 0.05$ del criterio AvgFitsR (con un 70 %).

Cabe destacar los malos resultados obtenidos por el criterio AvgFitεR, incluso en el porcentaje de soluciones encontradas. Esto se debe a que este criterio tiene cierta tendencia a cambiar la rejilla por una más cuadrada y, como se aprecia en la tabla 5.8, los mejores resultados de este problema se obtienen con rejillas rectangulares y, en especial, la rejilla dinámica preprogramada que cambia de la rejilla *Square* a la estrecha.

Es especialmente destacable que los resultados obtenidos con este problema y el criterio AvgFitR para cualquier ε de los utilizados mejora al mejor de los obtenidos con las distintas rejillas fijas y dinámicas preprogramadas ya estudiadas. Por otro lado, destaca que el tiempo medio de ejecución utilizado para resolver el problema, en proporción al número medio de evaluaciones, no es tan bajo con respecto a los resultados disponibles en el Apéndice A; esto se debe al coste adicional que supone comprobar (y realizar si se da el caso) los cambios de rejilla.

Los dos criterios que demuestran una cierta tendencia a evolucionar hacia rejillas más cuadradas (Ver Capítulo 8), AvgFitεR y AvgFitsR, realizan con este problema un número de cambios de rejilla bastante inferior al de los otros dos criterios. Este hecho no es de extrañar si pensamos en que, como se ha especificado con anterioridad, este problema ofrece mejores valores con rejillas rectangulares. En concreto, en caso del criterio AvgFitsR con los valores de $\varepsilon = 0.15$, $\varepsilon = 0.25$ y $\varepsilon = 0.3$ los seis cambios de rejilla que realizan se corresponden con las seis primeras comprobaciones y se realizan siempre hacia la siguiente rejilla de mayor ratio (más cuadrada), hasta que se alcanza la rejilla de mayor ratio, manteniéndose la población en esa disposición hasta el final del algoritmo.

Sorprende el elevado número de cambios de rejilla que se da en la Tala C.6 (salvo en el caso de $\varepsilon = 0.05$), que es del orden de entre 6 y 10 veces mayor que el realizado por el criterio AvgFitR.

Por último, y a propósito de la observación realizada en el párrafo anterior, comentaremos que el cambio de rejilla no parece ser muy costoso, ya que, por ejemplo, el criterio AvgFitR con $\varepsilon = 0.3$ (ver Tabla C.8) y AvgFitsR (Tabla C.7) con $\varepsilon = 0.15$ tienen aproximadamente el mismo número medio de evaluaciones y el mismo tiempo medio de ejecución; mientras que el número medio de

cambios de rejilla es más o menos el triple en el caso de AvgFitR. Se puede ver aun más claro en el caso de los criterios AvgFitrR con $\varepsilon = 0.3$ (76.5 cambios de rejilla y 2405.60 segundos) y AvgFitsR con $\varepsilon = 0.05$ (1.57 cambios y 2317.00 segundos) en las tablas C.6 y C.7.

Con los datos aportados por estas tablas, conluimos que el mejor criterio, debido a que ofrece los valores más bajos tanto en número medio de evaluaciones como en tiempo medio de ejecución, es el **AvgFitR** (criterio basado en la media de los valores de fitness de los individuos de la población, y que comienza utilizando la rejilla más rectangular).

A continuación se realizará el mismo estudio pero tomando como referencia en este caso el valor de la entropía (ver [CTTS98]) de la población. Este valor puede resultar interesante debido a que nos ofrece una medida de la diversidad existente entre los individuos que componen la población.

7.2. Estudio Según la Entropía de la Población

En esta sección completaremos el estudio de la Sección 7.1 ampliando sus resultados mediante la comparación de los dos criterios más representativos de los estudiados tomando como medida la entropía de la población en lugar de utilizar la media del valor de fitness de los individuos. El estudio se realizará para los problemas MAXCUT100 y MTTP200.

La entropía se obtiene mediante la siguiente Fórmula 7.1 [CTTS98] y proporciona una medida de la diversidad de genotipos de la población.

$$H(x) = \sum_{y \in \Gamma} q_x(\gamma) \log \frac{1}{q_x(\gamma)}, \tag{7.1}$$

Para no extendernos demasiado en este estudio, se presentarán las tablas correspondientes a los resultados de los dos mejores criterios estudiados en la Sección 7.1 pero trasladados al caso de la entropía. Se trata, por tanto, de los criterios:

- PopHεR:
  1. Comenzar con la rejilla más rectangular
  2. Si $(H_{t+1} - H_t) < \varepsilon(H_t - H_{t-1})$, donde H es la entropía de la población, entonces usar siguiente rejilla más cuadrada
  3. Si $(H_{t+1} - H_t) > (1 - \varepsilon)(H_t - H_{t-1})$ entonces usar la siguiente rejilla más rectangular
- PopHR:

1. Comenzar con la rejilla más rectangular
2. Si $(H_{t+1} - H_t) < (1 + \varepsilon)(H_t - H_{t-1})$, donde H es la entropía de la población, entonces usar siguiente rejilla más cuadrada
3. Si $(H_{t+1} - H_t) > (2 - \varepsilon)(H_t - H_{t-1})$ entonces usar la siguiente rejilla más rectangular

7.2.1. Problema MAXCUT100

Para este problema, vemos que los criterios basados en la entropía de la población realizan un mayor número medio de evaluaciones que sus homólogos de la sección anterior, al igual que ocurre si nos fijamos en el tiempo medio de ejecución. Además, se observa que realizan bastantes más cambios de rejilla que los basados en el valor de adecuación de los individuos. Por otro lado, dentro de los criterios basados en la entropía se puede apreciar que el criterio PopHR ofrece resultados bastante mejores que PopHεR tanto en número medio de evaluaciones como en tiempo medio.

Por tanto, fijándonos tanto en el número medio de evaluaciones como en el tiempo medio de ejecución, podemos concluir que los criterios basados en el valor de adecuación ofrecen mejores valores que los basados en la entropía: Por un lado, PopHεR es peor que AvgFitεR para cualquier valor de ε aunque, por otro lado, PopHR es mejor que AvgFitR para los valores de $\varepsilon = 0.25$ y 0.3 donde tiene, además, porcentajes de solución algo mayores.

Si nos fijamos en los tiempos medios de ejecución, podremos apreciar que los criterios basados en la entropía son siempre más lentos que los basados en el valor de adecuación. Esto es debido a que es necesario un procesamiento extra en cada evaluación del algoritmo para calcular la entropía, mientras que la media de los valores de adecuación de los individuos es un dato que cualquier GA necesita para las operaciones que va realizando en cada paso y, por tanto, se calcula siempre.

7.2.2. Problema MTTP200

Viendo las tablas C.11 y C.12 observamos que para el problema MTTP200 el mejor resultado lo proporciona el criterio PopHεR, aunque tanto PopHεR como PopHR ofrecen peores resultados que el criterio AvgFitR, tanto en número medio de evaluaciones como en el tiempo medio empleado para encontrar la solución.

Por otro lado, destacaremos que en el criterio PopHR se realizan muchos más cambios de rejilla que en el criterio PopHεR.

Como nuestro objetivo consiste en conseguir un buen criterio robusto que funcione bien en todos los problemas, utilizaremos el criterio **AvgFitR** para el caso en el que utilicemos el valor de adecuación medio de los individuos y su homólogo para el caso de la entropía **PopHR**, a pesar de que este último criterio funcione peor que PopHεR para este problema.

Hechas todas estas comparaciones, decidimos utilizar los criterios AvgFitR y PopHR para el estudio de los problemas.

Capítulo 8

Estudio del Comportamiento de los Problemas con los Criterios Seleccionados

Una vez elegido el mejor criterio para todos los problemas, pasaremos a realizar un estudio más completo del comportamiento de dicho criterio. En este capítulo se estudiarán los resultados de este criterio para los mismos valores de ε empleados en el Capítulo 6, pero en este caso se realizarán 30 ejecuciones para obtener valores más fiables y se ampliarán los problemas estudiados a los siguientes: COUNTSAT, ECC, FMS, MAXCUT (con las instancias cut20.01, cut20.09 y cut100), MMDP, MTTP (con 20, 100 y 200 variables) y P-PEAKS.

Además, se estudiará para todos estos problemas el comportamiento del criterio basado en el fitness y en la entropía y, para cada uno de estos dos, comenzando por la rejilla más rectangular (AvgFitR y PopHR) y por la rejilla de ratio más intermedio (AvgFitC y PopHC).

El objetivo de este capítulo consiste en comprobar qué valores de ε funcionan mejor con cada problema de los arriba mencionados.

8.1. Estudio del Criterio AvgFitR

Partiendo de las tablas D.2, D.5, D.6 D.4, D.7, D.8, D.9, D.10, D.11 y D.1 estudiaremos qué valores de ε producen mejores resultados para cada uno de los problemas:

- Problema **COUNTSAT** (Tabla D.1): Con el problema COUNTSAT se obtiene un menor tiempo de ejecución cuando utilizamos $\varepsilon = 0.3$ que en el caso en el que $\varepsilon = 0.15$, que es el caso que ofrece el menor tiempo de ejecución y mayor porcentaje de soluciones alcanzadas.

- Problema **ECC** (Tabla D.2): En este problema se puede apreciar que la mejor solución obtenida según el número medio de evaluaciones es $\varepsilon = 0.05$, mientras que si consideramos el tiempo medio de ejecución la mejor solución será $\varepsilon = 0.15$. El porcentaje de soluciones encontradas es del 100 % en todos los casos.

- Problema **FMS** (Tabla D.3: La mejor solución según el número medio de evaluaciones será la correspondiente a $\varepsilon = 0.3$, mientras que según el tiempo medio de ejecución será $\varepsilon = 0.05$. Ambos casos poseen un porcentaje de soluciones del 90 %, que, además es el máximo porcentaje encontrado.

- Problema **MAXCUT20.01** (Tabla D.5): Los mejores resultados para este problema se obtienen cuando $\varepsilon = 0.05$; tanto teniendo si tenemos en cuenta el número medio de evaluaciones como el tiempo medio de ejecución o el porcentaje de soluciones encontradas.

- Problema **MAXCUT20.09** (Tabla D.6): En este caso, según el número medio de evaluaciones, el mejor resultado se corresponde con $\varepsilon = 0.05$; mientras que si tenemos en cuenta sólo el tiempo medio de ejecución nos quedaríamos con $\varepsilon = 0.3$ como el que proporciona mejores valores. En cuanto al porcentaje de soluciones encontradas, en todos los casos se obtiene el 100 % de las soluciones.

- Problema **MAXCUT100** (Tabla D.4): En este caso cabe destacar que si nos fijamos en el tiempo medio de ejecución, el valor de $\varepsilon = 0.3$ es el que ofrece un mejor resultado, pero no se considera muy significativo debido a la alta desviación estándar que presenta y al bajo porcentaje de soluciones encontradas. Por otro lado, el mejor resultado según el tiempo medio se corresponde con el caso en que $\varepsilon = 0.25$, y si tomamos como baremo el porcentaje de soluciones encontradas, el mejor ε es el de valor igual a 0.15.

- Problema **MMDP** (Tabla D.7): El mejor resultado teniendo en cuenta tanto el número medio de evaluaciones como el tiempo medio de ejecución se corresponde con el valor $\varepsilon = 0.3$; el mejor porcentaje de soluciones se obtiene al utilizar $\varepsilon = 0.25$.

- Problema **MTTP20** (Tabla D.10): El menor número de evaluaciones lo conseguimos con $\varepsilon = 0.15$, mientras que el menor tiempo medio nos lo proporciona $\varepsilon = 0.25$. En cuanto al porcentaje de soluciones encontradas, en todos los casos se obtiene el 100 %.

- Problema **MTTP100** (Tabla D.10): Según el número medio de evaluaciones, el mejor resultado se obtiene con $\varepsilon = 0.3$, mientras que si tenemos en cuenta solamente el tiempo medio, este valor cambia a $\varepsilon = 0.25$.

- Problema **MTTP200** (Tabla D.10): En este caso, el mejor resultado tanto según el número medio de evaluaciones como el Tiempo medio de ejecución o el Porcentaje de soluciones encontradas coincide en el mismo valor de ε, el correspondiente a 0.3. Es de destacar la similitud que se da para este problema entre los valores obtenidos para cualquiera de los valores de ε empleados.

- Problema **P-PEAKS** (Tabla D.11): Al igual que ocurre con ECC, en el caso del problema P-PEAKS nos encontramos con diferentes elecciones de ε dependiendo del parámetro que utilicemos en la comparación. Para este problema vemos que la mejor solución puede ser $\varepsilon = 0.15$ si consideramos el tiempo medio o $\varepsilon = 0.25$ si lo que tenemos en cuenta es el número de evaluaciones; como en el caso del problema ECC, todos los porcentajes de solución encontrados son del 100 %.

Una vez elegidos los valores de ε mejores para cada problema y el criterio AvgFitR, procederemos a comparar los resultados obtenidos utilizando estos valores de ε con los ya estudiados en el Capítulo 5 para el caso de las rejillas fijas y dinámicas preprogramadas: Es necesario reseñar que los problemas FMS, MMDP y P-PEAKS sólo se podrán comparar con respecto al número medio de evaluaciones, ya que es el único dato que aparece en el artículo [AT00]. El problema COUNTSAT ofrece mejores resultados con el criterio AvgFitR que con cualquiera de las rejillas dinámicas preprogramadas y estáticas tanto en el número medio de evaluaciones como en el tiempo medio empleado y en el porcentaje de soluciones alcanzadas. Se aprecia que el problema ECC da peor resultado que las rejillas fijas (*Rectangular*, *Narrow* y *Square*) tanto en el número medio de evaluaciones como en el tiempo medio, pero mejora los casos de las dinámicas preprogramadas tanto en tiempo medio como en número de evaluaciones. Por otro lado, el problema FMS ofrece resultados similares a *Rectangular* y *Narrow*, siendo ligeramente peor que *NarSq* y mejor con respecto a los casos *Square* y *SqNar*. Los resultados obtenidos con MAXCUT100 son algo peores que los resultados de las rejillas *Square* y *Rectangular*, pero mejor que el resto de rejillas fijas y dinámicas preprogramadas; todo ello según el número medio de evaluaciones y el tiempo medio. Si tomamos como referencia el porcentaje de soluciones encontradas, las rejillas *Narrow* y *SqNar* obtienen resultados un poco mejores que los del criterio AvgFitR. En cuanto al problema MMDP el número de evaluaciones obtenido es bastante menor que cualquiera de los que ofrecen las rejillas fijas. Este hecho cobra singular importancia al ver que el algoritmo no es capaz de encontrar solución alguna antes de 1002900 evaluaciones cuando utiliza cualquiera de las dos rejillas dinámicas preprogramadas estudiadas. Los valores obtenidos con MTTP200 y AvgFitR son mejores que cualesquiera de los obtenidos con las rejillas fijas y dinámicas preprogramadas, tanto tomando como referencia el número medio de evaluaciones como el tiempo medio; además en todos los casos (rejillas fijas, dinámicas preprogramadas y dinámica auto-adaptativa) el porcentaje de soluciones es del 100 %. En cuanto al problema P-PEAKS el número medio de evaluaciones realizadas por el

criterio es sólo inferior al obtenido con la rejilla *SqNar*, siendo bastante mayor que en las rejillas estáticas y que en la otra dinámica preprogramada. Por último, .

Se desea destacar que los resultados de los problemas MTTP200 y COUNTSAT son mejores que cualquiera de las rejillas fijas y dinámicas preprogramadas; pero no sólo para el valor de ε que hemos seleccionado, sino también para cualquiera de los estudiados.

En resumen, diremos que, según el número medio de evaluaciones realizadas, ECC mejora con AvgFitR los resultado de las rejillas dinámicas preprogramadas, pero empeora los de las rejillas estáticas, MAXCUT100 mejora los resultados de la rejilla *Narrow* y las dinámicas preprogramadas, MMDP, MTTP200 y COUNTSAT ofrecen mejores resultados que cualquiera de las rejillas fijas y dinámicas preprogramadas y P-PEAKS sólo mejora el valor obtenido con la rejilla *SqNar*. Si nos fijamos en el tiempo medio, ECC empeora con respecto a las rejillas fijas, pero obtiene mejores resultados que al utilizar las rejillas dinámicas preprogramadas, MAXCUT100 es mejor para el caso de las rejillas dinámicas preprogramadas y la *Narrow*, y MTTP200 y COUNTSAT obtienen menores valores que cualquiera de las rejillas dinámicas preprogramadas y estáticas. Por último, según el porcentaje de soluciones encontradas, tanto ECC como MTTP200 obtienen el 100 % de las soluciones en todos los casos; con MAXCUT100 conseguimos un porcentaje menor al utilizar AvgFitR que en el caso de las rejillas *Narrow* y *SqNar*. En cambio, COUNTSAT mejora con respecto a todas las rejillas fijas y dinámicas preprogramadas al usar AvgFitR.

Por último, destacaremos que, en general, para todos los valores hallados, se obtienen desviaciones estándar menores en el caso de utilizar AvgFitR, por lo que los valores obtenidos tras realizar las distintas ejecuciones son más uniformes en este caso.

8.2. Estudio del Criterio AvgFitC

En esta sección se estudiarán los mismos problemas que en la Sección 8.1 y utilizando el mismo criterio; pero en este caso se comenzará la ejecución del algoritmo con la rejilla intermedia entre la más rectangular y la más cuadrada, es decir, la que posea un ratio más cercano a la media entre los ratios de la rejilla más estrecha y de la más cuadrada. Los resultados obtenidos se detallan en las tablas E.2, E.4, E.5, E.6, E.7, E.8, E.9, E.10, E.11 y E.1.

A continuación se detallan para cada problema las mejores soluciones según tres parámetros: el número medio de evaluaciones, el tiempo medio empleado durante las ejecuciones y el porcentaje

de soluciones encontradas durante las 30 ejecuciones.

- Problema **COUNTSAT** (Tabla E.1): En el problema COUNTSAT, cuando utilizamos $\varepsilon = 0.3$ se obtienen valores menores tanto en tiempo de ejecución como en número medio de evaluaciones realizadas, aunque también obtiene los peores resultados en cuanto al porcentaje de soluciones encontradas. Por otro lado, si nos fijamos en el porcentaje de soluciones encontradas, el valor de ε que ofrece mejores resultados es 0.25.

- Problema **ECC** (Tabla E.2): La mejor solución obtenida con ECC utilizando el criterio AvgFitC tanto según el número medio de evaluaciones como el tiempo medio de ejecución la mejor solución es la obtenida con $\varepsilon = 0.15$.

- Problema **FMS** (Tabla E.3): Tanto según el número medio de soluciones encontradas como según el tiempo medio empleado en cada ejecución, el ε que ofrece mejores resultados es el de valor 0.05. Aunque, según el porcentaje de soluciones encontradas, $\varepsilon = 0.25$ ofrece resultados levemente mejores (un 80 % frente al 73.33 % del caso $\varepsilon = 0.05$).

- Problema **MAXCUT20.01** (Tabla E.4): Según cualquiera de los tres parámetros en que nos fijamos, el mejor resultado es el proporcionado por $\varepsilon = 0.15$.

- Problema **MAXCUT20.09** (Tabla E.5): Como en el caso de MAXCUT20.01 existe un valor de ε que ofrece los mejores resultados en cualquiera de los tres parámetros de estudio. Dicho valor es 0.25.

- Problema **MAXCUT100** (Tabla E.6): En este caso la mejor solución es la correspondiente a $\varepsilon = 0.05$ según el tiempo medio de ejecución y el número medio de evaluaciones. Por otro lado, el valor de ε que nos lleva a obtener un mejor porcentaje de soluciones encontradas es 0.15.

- Problema **MMDP** (Tabla E.7): Para este problema es claro que las mejores soluciones las da $\varepsilon = 0.25$, tanto según el número medio de evaluaciones como el tiempo medio de ejecución y el porcentaje de soluciones encontradas.

- Problema **MTTP20** (Tabla E.8): Con $\varepsilon = 0.15$ obtenemos el menor número medio de evaluaciones realizadas, mientras que con $\varepsilon = 0.05$ conseguimos el menor tiempo medio de ejecución. En cuanto al porcentaje de soluciones encontradas, en todos los casos se encuentra el 100 %.

- Problema **MTTP100** (Tabla E.9): En este caso $\varepsilon = 0.05$ consigue un menor número de evaluaciones que el resto, mientras que, si nos fijamos en el tiempo medio de ejecución, el menor valor se corresponde con el caso en el que $\varepsilon = 0.15$.

- Problema **MTTP200** (Tabla E.10): En este caso, el mejor resultado según el número medio de evaluaciones coincide con el que se obtiene con $\varepsilon = 0.05$, mientras que según el Tiempo medio de ejecución el mejor resultado sería el obtenido con $\varepsilon = 0.15$. Es de destacar, al igual que en el caso del estudio de este problema con AvgFitR, la similitud que se da entre los valores obtenidos para cualquiera de los valores de ε empleados.

- Problema **P-PEAKS** (Tabla E.11): En el caso del problema P-PEAKS nos encontramos con que la mejor solución es la correspondiente a $\varepsilon = 0.15$ independientemente del parámetro que utilicemos en la comparación (tiempo medio y número de evaluaciones).

Si comparamos los resultados de este criterio con los obtenidos en los estudios anteriores (Capítulo 5 y Sección 8.1), nos fijamos que con COUNTSAT se han obtenido resultados un poco peores (según los tres parámetros: número medio, tiempo y porcentaje de soluciones) que los del ε seleccionado para el criterio AvgFitR; esto es lógico si pensamos que el criterio AvgFitR funcionará mejor en los problemas que obtengan mejores resultados al utilizar la rejilla *Rectangular* (que son, precisamente, MTTP200 y COUNTSAT) debido a que con este criterio empiezan por la rejilla de ratio intermedio y tienen que evolucionar hacia la rejilla de menor ratio (la más estrecha), mientras que con AvgFitR al empezar directamente por esa rejilla no tienen que hacer esta evolución. En lo que respecta al número medio de evaluaciones el problema ECC obtiene resultados muy parecidos al caso en el que se utilizó el criterio AvgFitR; pero, tanto para el número medio de evaluaciones como para el tiempo medio empleado, el menor valor obtenido con AvgFitC es menor que el obtenido con el criterio AvgFitR, pero sigue ofreciendo peores resultados que las rejillas *Rectangular* y *Narrow* (narrow). Con FMS obtenemos peores resultados que con AvgFitR y, además, de todas las rejillas fijas y dinámicas con cambio preprogramado, sólo mejoramos el valor obtenido por *SqNar*. Al igual que con ECC, también con el problema MAXCUT100 se ha conseguido obtener valores mejores al aplicar el criterio AvgFitC en lo referente al número medio de evaluaciones y al porcentaje de soluciones encontradas, pero, en cambio, esto no sucede si nos fijamos en el tiempo medio empleado. Además, en este caso sí hemos conseguido un resultado en cuanto al número medio de evaluaciones mejor que el obtenido en los casos en que se usaron rejillas fijas y dinámicas preprogramadas. El problema MMDP obtiene un peor resultado que para el caso del criterio AvgFitR, aunque sigue siendo mejor que los obtenidos en los casos de rejillas fijas y dinámicas preprogramadas. Con MTTP200 se han obtenido resultados un poco peores que los obtenidos con el criterio AvgFitR. El problema P-PEAKS ofrece prácticamente los mismos resultados con ambos criterios, aunque el ε seleccionado con este criterio ofrece valores algo más bajos que los obtenidos para el caso que se eligió como mejor con el criterio AvgFitR. Por último,

8.3. Estudio del Criterio PopHR

En esta sección se realizará un estudio similar al realizado por el criterio AvgFitR, pero utilizando en esta ocasión la entropía de la población como valor a interpretar por el criterio. Por tanto, en este caso, la rejilla se modificará en función de la diversidad existente entre los individuos.

Los resultados que en este caso hemos tomado como más satisfactorios son los correspondientes a $\varepsilon = 0.3$ en los problemas ECC y MMDP, $\varepsilon = 0.25$ en el caso de MAXCUT100, $\varepsilon = 0.05$ en MTTP200 y $\varepsilon = 0.15$ en COUNTSAT y P-PEAKS. Estas conclusiones se desprenden de los siguientes razonamientos:

- Problema **COUNTSAT** (Tabla F.1): Con COUNTSAT se obtienen valores bastante menores con $\varepsilon = 0.15$ tanto en el número medio de evaluaciones como en el tiempo medio, aunque también logra el menor porcentaje de soluciones encontradas; el mayor se corresponde con $\varepsilon = 0.3$, que tiene el excelente valor del 96.67 % de soluciones encontradas, que es un valor realmente bueno para este problema.

- Problema **ECC** (Tabla F.2): En este caso, el mejor valor es distinto en función del parámetro que tomemos como referencia: si usamos el número medio de evaluaciones el menor valor se corresponde con $\varepsilon = 0.3$; pero si nos fijamos en el tiempo medio es menor el valor obtenido con $\varepsilon = 0.25$.

- Problema **FMS** (Tabla F.3): En este problema, el mejor número medio de evaluaciones y el mejor porcentaje de soluciones encontradas los obtenemos con $\varepsilon = 0.25$, mientras que con $\varepsilon = 0.3$ conseguimos el mejor tiempo de ejecución medio.

- Problema **MAXCUT20.01** (Tabla F.4): El mejor caso, tanto si nos fijamos en el número medio de evaluaciones como en el tiempo medio de ejecución o el porcentaje de soluciones encontradas, se corresponde con el caso $\varepsilon = 0.05$.

- Problema **MAXCUT20.09** (Tabla F.5): Al igual que en el caso de MAXCUT20.01, todos los parámetros estudiados coinciden en que el caso en el que se obtienen mejores resultados es $\varepsilon = 0.05$.

- Problema **MAXCUT100** (Tabla F.6): Tanto en el caso del número medio de evaluaciones como en el del tiempo medio empleado en la ejecución, los mejores resultados se obtienen al utilizar $\varepsilon = 0.25$; por otro lado, si tomamos como baremo el porcentaje de soluciones encontradas, el mejor resultado cambiará al caso en el que $\varepsilon = 0.3$.

- Problema **MMDP** (Tabla F.7): El mejor resultado se corresponde con el caso en el que $\varepsilon = 0.3$ tanto en número medio de evaluaciones como en tiempo medio. Si nos guiamos por

el porcentaje de soluciones encontradas, el mejor valor se corresponde con $\varepsilon = 0.05$.

- Problema **MTTP20** (Tabla F.8): El menor valor en el número medio de evaluaciones lo proporciona $\varepsilon = 0.15$, pero en el tiempo medio es el obtenido por $\varepsilon = 0.05$.

- Problema **MTTP100** (Tabla F.9): Tanto según el número medio de evaluaciones como el tiempo medio o el porcentaje de soluciones, el mejor resultado lo da $\varepsilon = 0.05$.

- Problema **MTTP200** (Tabla F.10): Como en los problemas estudiados arriba (salvo el ECC), los mejores valores tanto en número medio de evaluaciones como en tiempo medio coinciden en el mismo valor de ε : 0.05. En este caso los valores para los distintos ε no son ya tan parecidos como en los casos de los criterios basados en el fitness.

- Problema **P-PEAKS** (Tabla F.11): También en este caso coinciden los mejores valores según cualquiera de los parámetros en el mismo valor de ε : 0.15.

Para el caso del problema COUNTSAT, el resultado obtenido es peor que el que ofrece AvgFitR, pero similar al de AvgFitC. En cuanto al problema ECC, este criterio da excelentes resultados, ya que su mejor resultado es el mejor de los obtenidos hasta ahora, ya sea según el número medio de evaluaciones o el tiempo medio de ejecución. El problema FMS logra mejorar con este criterio cualquier resultado obtenido hasta ahora -tanto en el caso de las rejillas fijas como en las dinámicas con cambio preprogramado y en las dinámicas auto-adaptativas-, según el número medio de evaluaciones y el porcentaje de soluciones encontradas. En cuanto a los demás problemas, los resultados obtenidos con MAXCUT100 son similares a los que ofrecen los criterios basados en fitness. Los resultado de MMDP se encuentran entre los de los criterios AvgFitR y AvgFitC. MTTP200 obtiene mejores resultados que las rejillas dinámicas preprogramadas, pero peores que los casos de criterios auto-adaptativos basados en el valor de adecuación de los individuos. Es de destacar la mejoría que supone utilizar este criterio con P-PEAKS, ya que, además, sus resultados están ya a la altura de los mejores resultados obtenidos en [AT00].

Es de destacar que, como confirmaremos con los resultados de la Sección 8.4, al introducir la entropía como baremo de estudio en el criterio el número de cambios de rejilla realizados aumenta considerablemente en los problemas ECC y MTTP200.

Los porcentajes de soluciones encontradas aumentan tanto en MAXCUT100 como en COUNTSAT con respecto a todos los estudios realizados hasta ahora. Concretamente, MAXCUT100 obtiene en todos los casos solución en más del 50 % de las ejecuciones; y COUNTSAT encuentra solución en el 96.67 % de las ejecuciones para el caso en que $\varepsilon = 0.3$.

Los tiempos de ejecución se ven incrementados con respecto al número de evaluaciones realizadas debido a que obtener la entropía de la población en cada paso del GA supone un coste de procesamiento extra, porque en el caso del valor medio de adecuación de los individuos, el algoritmo siempre calcula automáticamente este valor como uno de sus datos estadísticos.

8.4. Estudio del Criterio PopHC

En este caso se estudiará el comportamiento de los distintos problemas cuando se utiliza el criterio auto-adaptativo basado en la entropía de la Sección 8.3, pero comenzando en este caso por la rejilla de ratio más intermedio.

A continuación estudiaremos los resultados obtenidos por los estudios realizados sobre este criterio (ver tablas del Apéndice G):

- Problema **COUNTSAT** (Tabla G.1): El mejor resultado para COUNTSAT, tanto si nos fijamos en el número medio de evaluaciones como en el tiempo medio, se obtiene cuando $\varepsilon : 0.15$; por otro lado $\varepsilon : 0.05$ es el caso que mayor número de soluciones encuentra durante las 30 ejecuciones.

- Problema **ECC** (Tabla G.2): Tanto en función del número medio de evaluaciones como del tiempo medio el menor valor obtenido se da cuando $\varepsilon = 0.3$.

- Problema **FMS**(Tabla G.3): El mejor valor de ε es 0.3 en el número medio de evaluaciones; en el tiempo medio se obtienen valores similares con $\varepsilon = 0.3$ y $\varepsilon = 0.25$, mientras que según el porcentaje de soluciones encontradas el mejor ε es 0.25.

- Problema **MAXCUT20.01** (Tabla G.4): En este caso, los mejores resultados nos los dan los valores de ε extremos: 0.05 según el número medio de evaluaciones y 0.3 si consideramos el tiempo medio de ejecución.

- Problema **MAXCUT20.09** (Tabla G.5): Como en el caso en el que se utilizó la rejilla PopHR, los mejores resultados según cualquiera de los tres parámetros en los que nos fijamos, lo da $\varepsilon = 0.05$.

- Problema **MAXCUT100** (Tabla G.6): Tanto en el caso del número medio de evaluaciones como en el del tiempo medio empleado en la ejecución y el porcentaje de soluciones encontradas, los mejores resultados se obtienen al utilizar $\varepsilon = 0.25$.

- Problema **MMDP** (Tabla G.7): Para este problema, el mejor valor según el número medio de evaluaciones coinciden con el valor $\varepsilon : 0.15$, mientras que el menor tiempo medio lo obtenemos con $\varepsilon = 0.05$; también $\varepsilon : 0.15$ ofrece un porcentaje de soluciones encontradas algo mejor.

- Problema **MTTP20** (Tabla G.8): Con $\varepsilon = 0.15$ obtenemos el menor número medio de evaluaciones, mientras que el menor tiempo medio se obtiene al utilizar $\varepsilon = 0.25$.

- Problema **MTTP100** (Tabla G.9): Tanto según el número medio de evaluaciones como según el tiempo medio de ejecución, los mejores valores los proporciona $\varepsilon = 0.05$.

- Problema **MTTP200** (Tabla G.10): El mejor resultado en el caso del número medio de evaluaciones se corresponde con el caso en el que $\varepsilon = 0.05$, mientras que fijándonos en el tiempo medio el mejor caso lo da $\varepsilon = 0.15$.

- Problema **P-PEAKS** (Tabla G.11): El caso $\varepsilon : 0.05$ ofrece los menores valores en cuanto al número medio de evaluaciones, pero si tenemos en cuenta el tiempo medio el ε elegido será 0.3.

Como podemos ver en las tablas correspondientes (ver apéndices D, E, F y G), para todos los problemas estudiados existe otro criterio auto-adaptativo de los antes estudiados que obtiene mejores resultados que el que nos ocupa en esta sección. Es más, salvo los problemas ECC y P-PEAKS, que tienen mejores resultados con este criterio que con los dos basados en el fitness, cualquiera de los demás problemas obtiene peores resultados con PopHC que con cualquiera de los demás criterios auto-adaptativos.

8.5. Estudio del Criterio FitHC

En este caso se estudiará el comportamiento de los distintos problemas cuando se utiliza el criterio auto-adaptativo basado en la entropía de la población y el fitness de los individuos conjuntamente, comenzando por la rejilla de ratio más intermedio.

El criterio FitHC se puede describir como sigue:

1. Comenzar con la rejilla de ratio más intermedio

2. Si $((H_{t+1} - H_t) < (1+\varepsilon)(H_t - H_{t-1}))$ y $((\overline{f}_{t+1} - \overline{f}_t) < (1+\varepsilon)(\overline{f}_t - \overline{f}_{t-1}))$ entonces, si existe, usar siguiente rejilla más cuadrada

3. Si $((H_{t+1} - H_t) > (2-\varepsilon)(H_t - H_{t-1}))$ y $((\overline{f}_{t+1} - \overline{f}_t) > (2-\varepsilon)(\overline{f}_t - \overline{f}_{t-1}))$ entonces, si existe, usar la siguiente rejilla más rectangular

A continuación estudiaremos los resultados obtenidos por los estudios realizados sobre este criterio (ver tablas del Apéndice H):

- Problema **COUNTSAT** (Tabla H.1): El mejor resultado para COUNTSAT, tanto si nos fijamos en el número medio de evaluaciones como en el tiempo medio, se obtiene cuando $\varepsilon : 0.3$; por otro lado $\varepsilon : 0.15$ es el caso que mayor número de soluciones encuentra a lo largo de las 30 ejecuciones.

- Problema **ECC** (Tabla H.2): Tanto en función del número medio de evaluaciones como del tiempo medio el menor valor obtenido se da cuando $\varepsilon = 0.15$.

- Problema **FMS** (Tabla H.3): Los mejores valores obtenidos según el número medio de evaluaciones, el porcentaje de soluciones encontradas y el porcentaje de soluciones encontradas corresponden a $\varepsilon = 0.25$.

- Problema **MAXCUT20.01** (Tabla H.4): Los mejores resultados tanto según el número medio de evaluaciones como el tiempo medio de ejecución o el porcentaje de soluciones encontradas se obtienen con el mismo valor de ε: 0.3.

- Problema **MAXCUT20.09** (Tabla H.5): Si consideramos el número medio de evaluaciones, obtendremos $\varepsilon = 0.05$ como el caso que ofrece mejores resultados, mientras que si nos fijamos en el tiempo medio de evaluación los mejores resultados vendrán dados por $\varepsilon = 0.15$.

- Problema **MAXCUT100** (Tabla H.6): El menor número medio de evaluaciones lo proporciona $\varepsilon = 0.15$, el menor tiempo medio, $\varepsilon = 0.25$; y el mayor porcentaje de soluciones encontradas, $\varepsilon = 0.3$.

- Problema **MMDP** (Tabla H.7): Para este problema, todos los parámetros estudiados (el número medio de evaluaciones, el menor tiempo medio y el mejor porcentaje de soluciones) nos lo proporciona el mismo valor: $\varepsilon : 0.15$.

- Problema **MTTP20** (Tabla H.8): El menor número medio de evaluaciones lo proporciona $\varepsilon = 0.05$, mientras que el menor tiempo medio lo obtiene el caso $\varepsilon = 0.3$.

- Problema **MTTP100** (Tabla H.9): Según el número medio de evaluaciones, el mejor resultado es el proporcionado por $\varepsilon = 0.15$, mientras que $\varepsilon = 0.3$ es el caso que menor tiempo medio obtiene.

- Problema **MTTP200** (Tabla H.10): El mejor resultado en el caso del número medio de evaluaciones coincide con el mejor valor del el tiempo medio: $\varepsilon = 0.3$.

- Problema **P-PEAKS** (Tabla H.11): El caso $\varepsilon : 0.05$ ofrece el menor valor en cuanto al número medio de evaluaciones, pero si tenemos en cuenta el tiempo medio el ε elegido será 0.3.

Como podemos ver en las tablas correspondientes (ver apéndices D, E, F, G y H), para todos los problemas estudiados existe otro criterio auto-adaptativo de los antes estudiados que obtiene mejores resultados que el que nos ocupa en esta sección, salvo para el caso del problema MTTP200, que reduce ligeramente el número de evaluaciones en un 1.7 % con respecto al obtenido con Avg-FitR (que era el mejor hasta ahora), aunque el tiempo de ejecución se mantiene en los mismos valores en ambos casos.

8.6. Conclusión Sobre el Comportamiento de los Problemas con los Criterios Dinámicos Auto-Adaptativos

De los estudios realizados en las secciones anteriores, deducimos que no es posible sacar una conclusión de un único valor de ε que podamos utilizar para cualesquier criterio y problema y obtener resultados satisfactorios. Por ello, se ha decidido elaborar una serie de tablas, una por cada uno de los parámetros obtenidos (Número medio de evaluaciones, Tiempo medio empleado en encontrar la solución y porcentaje de soluciones encontradas), (tablas 8.4) en las que se detallan los valores de ε que ofrecen los mejores resultados para cada criterio y problema estudiados.

Recordaremos qué significa cada uno de los nombres que se le han dado a los criterios:

- **AvgFitR**: Criterio auto-adaptativo basado en el fitness que comienza por la rejilla más rectangular ($2 \times \frac{|Poblacion|}{2}$):
  1. Comenzar con la rejilla más rectangular
  2. Si $(\overline{f}_{t+1} - \overline{f}_t) < (1+\varepsilon)(\overline{f}_t - \overline{f}_{t-1})$ entonces, si existe, usar siguiente rejilla más cuadrada
  3. Si $(\overline{f}_{t+1} - \overline{f}_t) > (2-\varepsilon)(\overline{f}_t - \overline{f}_{t-1})$ entonces, si existe, usar la siguiente rejilla más rectangular
- **AvgFitC**: Criterio similar a AvgFitR, pero en este caso comienza por la rejilla cuyo ratio tenga el valor más intermedio de todos los ratios posibles de la población con respecto al vecindario:
  1. Comenzar con la rejilla con ratio más intermedio
  2. Si $(\overline{f}_{t+1} - \overline{f}_t) < (1+\varepsilon)(\overline{f}_t - \overline{f}_{t-1})$ entonces, si existe, usar siguiente rejilla más cuadrada
  3. Si $(\overline{f}_{t+1} - \overline{f}_t) > (2-\varepsilon)(\overline{f}_t - \overline{f}_{t-1})$ entonces, si existe, usar la siguiente rejilla más rectangular
- **PopHR**: Criterio auto-adaptativo basado en la entropía de la población que comienza por la rejilla más rectangular ($2 \times \frac{|Poblacion|}{2}$):

1. Comenzar con la rejilla más rectangular
2. Si $(H_{t+1} - H_t) < (1 + \varepsilon)(H_t - H_{t-1})$ entonces, si existe, usar siguiente rejilla más cuadrada
3. Si $(H_{t+1} - H_t) > (2 - \varepsilon)(H_t - H_{t-1})$ entonces, si existe, usar la siguiente rejilla más rectangular

- **PopHR**: Criterio auto-adaptativo basado en la entropía de la población que comienza por la rejilla con ratio vecindario/población más intermedio:

  1. Comenzar con la rejilla de ratio más intermedio
  2. Si $(H_{t+1} - H_t) < (1 + \varepsilon)(H_t - H_{t-1})$ entonces, si existe, usar siguiente rejilla más cuadrada
  3. Si $(H_{t+1} - H_t) > (2 - \varepsilon)(H_t - H_{t-1})$ entonces, si existe, usar la siguiente rejilla más rectangular

- **FitHC**: Criterio auto-adaptativo basado en la entropía de la población y en el valor de adecuación de los individuos de la población, que comienza por la rejilla con ratio vecindario/población más intermedio:

  1. Comenzar con la rejilla de ratio más intermedio
  2. Si $((H_{t+1} - H_t) < (1+\varepsilon)(H_t - H_{t-1}))$ y $((\overline{f}_{t+1} - \overline{f}_t) < (1+\varepsilon)(\overline{f}_t - \overline{f}_{t-1}))$ entonces, si existe, usar siguiente rejilla más cuadrada
  3. Si $((H_{t+1} - H_t) > (2-\varepsilon)(H_t - H_{t-1}))$ y $((\overline{f}_{t+1} - \overline{f}_t) > (2-\varepsilon)(\overline{f}_t - \overline{f}_{t-1}))$ entonces, si existe, usar la siguiente rejilla más rectangular

Partiendo de los datos reflejados en las tablas 8.1, 8.2 y 8.3, trataremos de obtener una tabla (la Tabla 8.4) única que asigne a cada problema y para cada criterio un solo valor de ε, que se adoptará como el mejor de los de las tres tablas estudiadas. No se ha definido un criterio exacto a la hora de decidirnos por uno u otro valor de ε en caso de que no coincidan para todas las tablas, pero se ha convenido darle algo más de preferencia al número medio de evaluaciones que al tiempo medio o el porcentaje de soluciones encontradas. A continuación se realiza un estudio de los valores de ε para cada problema y criterio que no coinciden en las tres tablas:

- Problema **COUNTSAT**:

  - Criterio **AvgFitR** (Tabla D.1): Se ha elegido como mejor resultado el obtenido por $\varepsilon = 0.15$ porque posee mejor porcentaje de soluciones encontradas, menor número medio de evaluaciones realizadas y sólo un poco mayor tiempo medio de ejecución con respecto a $\varepsilon = 0.3$.

Problema\Criterio	AvgFitR	AvgFitC	PopHR	PopHC	FitHC
COUNTSAT	0.15	0.3	0.15	0.15	0.3
ECC	0.15	0.15	0.3	0.3	0.15
FMS	0.3	0.05	0.25	0.3	0.25
MAXCUT20.01	0.05	0.15	0.05	0.05	0.3
MAXCUT20.09	0.05	0.25	0.05	0.05	0.05
MAXCUT100	0.25	0.05	0.25	0.25	0.15
MMDP	0.3	0.25	0.3	0.15	0.15
MTTP20	0.15	0.15	0.15	0.15	0.3
MTTP100	0.3	0.05	0.05	0.05	0.3
MTTP200	0.3	0.05	0.05	0.05	0.3
P-PEAKS	0.05	0.15	0.15	0.05	0.05

Tabla 8.1: Mejores valores de ε para cada problema y criterio según el número medio de evaluaciones

Problema\Criterio	AvgFitR	AvgFitC	PopHR	PopHC	FitHC
COUNTSAT	0.3	0.3	0.15	0.15	0.3
ECC	0.15	0.15	0.25	0.3	0.15
FMS	0.05	0.05	0.3	0.3	0.3
MAXCUT20.01	0.05	0.15	0.05	0.3	0.3
MAXCUT20.09	0.3	0.25	0.05	0.05	0.15
MAXCUT100	0.3	0.05	0.25	0.25	0.25
MMDP	0.3	0.25	0.3	0.05	0.15
MTTP20	0.25	0.05	0.05	0.25	0.05
MTTP100	0.25	0.15	0.05	0.05	0.15
MTTP200	0.3	0.15	0.05	0.15	0.3
P-PEAKS	0.15	0.15	0.15	0.3	0.3

Tabla 8.2: Mejores valores de ε para cada problema y criterio según el tiempo medio de ejecución

- Criterio **AvgFitC** (Tabla E.1): Se utilizará $\varepsilon = 0.3$ como el que ofrece mejores resultados, ya que la diferencia en el número total de evaluaciones es bastante significativa con el caso $\varepsilon = 0.25$, que sólo obtiene el mejor resultado en el Porcentaje de soluciones encontradas que, debido a que se hacen 30 ejecuciones, es un valor no del todo fiable, ya que puede variar bastante de unas pruebas a otras.

Problema\Criterio	AvgFitR	AvgFitC	PopHR	PopHC	FitHC
COUNTSAT	0.15	0.25	0.3	0.05	0.15
ECC	Todos	Todos	Todos	Todos	Todos
FMS	0.05; 0.3	0.25	0.25	0.25	0.3
MAXCUT20.01	Todos	Todos	Todos	Todos	Todos
MAXCUT20.09	Todos	Todos	Todos	Todos	Todos
MAXCUT100	0.25	0.3	0.25	0.25	0.3
MMDP	0.25	0.25	0.05	0.15	0.15; 0.3
MTTP20	Todos	Todos	Todos	Todos	Todos
MTTP100	Todos	Todos	Todos	Todos	Todos
MTTP200	Todos	Todos	Todos	Todos	Todos
P-PEAKS	Todos	Todos	Todos	Todos	Todos

Tabla 8.3: Mejores valores de ε para cada problema y criterio según el porcentaje de soluciones encontradas

- Criterio **PopHR** (Tabla F.1): Estamos ante el mismo caso que en el punto anterior, pero ahora con los valores de ε 0.15 y 0.3. Por la misma razón expuesta anteriormente se elige como mejor resultado el obtenido con $\varepsilon = 0.15$.

- Criterio **PopHC** (Tabla G.1): Otra vez nos encontramos en la misma situación que en los dos puntos anteriores; en este caso con los valores $\varepsilon = 0.15$ y $\varepsilon = 0.05$; actuando en consecuencia con los razonamientos expuestos en dichos puntos, se adopta como mejor solución la proporcionada por $\varepsilon = 0.15$.

- Criterio **FitHC** (Tabla H.1): En este caso sucede como en los tres puntos anteriores. El valor $\varepsilon = 0.3$ consigue los menores número medio de evaluaciones y tiempo medio, mientras que $\varepsilon = 0.15$ encuentra el mayor número de soluciones; actuando en consecuencia con los razonamientos expuestos en puntos anteriores, se adopta como mejor solución la proporcionada por $\varepsilon = 0.3$.

- Problema **ECC**

  - Criterio **AvgFitR** (Tabla D.2): Como mejor solución se ha optado por elegir $\varepsilon = 0.15$ porque mejora notablemente el tiempo de ejecución con respecto a $\varepsilon = 0.05$, mientras que sólo empeora un poco en el número de evaluaciones medio realizado.

- Criterio **PopHR** (Tabla F.2): Para ambos valores de ε el porcentaje de soluciones encontradas es del 100 %, luego debemos decidir qué epsilon seleccionar de los que aparecen las tablas correspondientes al número medio de evaluaciones y al tiempo medio para este problema y criterio. Se ha resuelto utilizar $\varepsilon = 0.3$ como mejor solución porque los valores que ofrece tienen desviaciones estándar bastante más bajas que en cualquiera de los demás casos.

- Problema **FMS**:

  - Criterio **AvgFitR** (Tabla D.3): $\varepsilon = 0.05$ es un 39.63 % mejor que $\varepsilon = 0.3$ en el tiempo medio y sólo un 9.93 % peor en el número medio de evaluaciones. Como en el porcentaje de soluciones encontradas son equivalentes, optamos por utilizar $\varepsilon = 0.05$ como el que mejores resultados ofrece.

  - Criterio **PopHR** (Tabla F.3): La diferencia entre $\varepsilon = 0.25$ y $\varepsilon = 0.3$ es más o menos del mismo porcentaje en el número medio de evaluaciones (21.92 % a favor de $\varepsilon = 0.25$) y el tiempo medio (26.33 % a favor de $\varepsilon = 0.3$). Como damos más importancia al número medio de evaluaciones y, además, en este caso el mayor porcentaje de soluciones encontradas se corresponde con $\varepsilon = 0.25$, elegiremos este valor como el mejor de los ε estudiados con este criterio.

  - Criterio **FitHC** (Tabla H.3): Al igual que en el caso anterior $\varepsilon = 0.25$ ofrece los mejores resultados en cuanto al número medio de evaluaciones que el porcentaje de soluciones encontradas, pero en este caso optaremos por seleccionar $\varepsilon = 0.3$ como el mejor porque la diferencia en el porcentaje de soluciones es muy baja (un 6 % de diferencia) y la diferencia entre ambos en el tiempo medio de ejecución es el doble de la diferencia en el número medio de evaluaciones a favor de $\varepsilon = 0.3$ (32.71 % en el tiempo medio a favor de $\varepsilon = 0.3$ frente al 13.98 % a favor de $\varepsilon = 0.25$ en el número medio de evaluaciones).

- Problema **MAXCUT20.01** y criterio **PopHC** (Tabla G.4): A pesar de que la diferencia en el número medio de evaluaciones es tan solo del 4.3 % a favor de $\varepsilon = 0.05$ y en el tiempo medio es del 10.29 % a favor del caso $\varepsilon = 0.3$, se ha seleccionado como mejor valor de ε al primero, ya que, al ser una instancia de problema tan simple, la diferencia en el tiempo medio es solamente algo más de medio segundo, y en el número medio de evaluaciones, es de una generación y media.

- Problema **MAXCUT20.09**:

  - Criterio **AvgFitR** (Tabla D.6): En este caso, la diferencia en el número de evaluaciones es del 17.99 % a favor de $\varepsilon = 0.05$ frente al 2.86 % a favor de $\varepsilon = 0.3$ en el caso del tiempo medio; por tanto se seleccionarán como mejores los resultados obtenidos por $\varepsilon = 0.05$.

  - Criterio **FitHC** (Tabla H.5): Como en el caso anterior, la diferencia en el número medio de evaluaciones es mayor que en el tiempo medio de ejecución (aunque en menor medida que en el caso anterior: 11.34 % frente al 5.59 %, respectivamente); por tanto, seleccionaremos como mejor valor a $\varepsilon = 0.05$.

- Problema **MAXCUT100**:

  - Criterio **AvgFitR** (Tabla D.4): El valor del tiempo medio obtenido en este caso no se considera muy significativo por poseer una elevada desviación estándar que presenta y al bajo porcentaje de soluciones encontradas. Por tanto, si no tenemos en cuenta el valor $\varepsilon = 0.3$, los mejores resultados tanto en el número medio de evaluaciones como en el tiempo medio se corresponden con el caso en que $\varepsilon = 0.25$, aunque según el porcentaje de soluciones encontradas, el mejor ε es el de valor igual a 0.15.

  - Criterio **AvgFitC** (Tabla E.6): En este caso los mejores valores para el número medio de evaluaciones y el tiempo medio de ejecución coinciden en $\varepsilon = 0.05$. La discordia se encuentra en el porcentaje de soluciones encontradas, pero se elegirá $\varepsilon = 0.05$ como valor de ϵ que ofrece mejores resultados ya que la diferencia entre los valores de ambos ε es mucho más significativa en los casos del número medio de evaluaciones y en el tiempo medio que en el porcentaje de soluciones encontradas.

  - Criterio **FitHC** (Tabla H.6): En este caso, la diferencia en el número medio de evaluaciones es del 19.75 % a favor de $\varepsilon = 0.15$, y la diferencia en al tiempo medio es del 6.09 % a favor $\varepsilon = 0.25$. La diferencia entre los porcentajes de soluciones no es demasiado significativa; por tanto, elegiremos como mejor caso a $\varepsilon = 0.15$.

- Problema **MMDP**:

  - Criterio **AvgFitR** (D.7): El porcentaje de soluciones encontradas en el caso de $\varepsilon = 0.3$ (proporciona el menor valor en número medio de evaluaciones y tiempo medio de ejecución) es muy cercano al mejor, por tanto, seleccionaremos $\varepsilon = 0.3$ como el mejor para

este criterio y problema.

- Criterio **PopHR** (F.7): El parámetro que obtiene distinto ε a los otros dos el referente al porcentaje de soluciones encontradas, aunque la diferencia no es grande: 96.67 % con $\varepsilon = 0.05$ frente al 90 % del caso $\varepsilon = 0.3$; por lo que utilizaremos éste último como representante de este problema con este criterio.

- Criterio **PopHC** (Tabla G.7): Como la diferencia entre los tiempos medios de ambas soluciones es muy grande, utilizaremos 0.05 como mejor valor de ε para este problema.

- Problema **MTTP20**:

  - Criterio **AvgFitR** (Tabla D.8): Elegimos como mejor ε al valor 0.15, ya que es un 27.03 % mejor en el número de evaluaciones y sólo un 18.12 % peor en el tiempo medio de ejecución que $\varepsilon = 0.25$.

  - Criterio **AvgFitC** (Tabla E.8): En este caso, $\varepsilon = 0.15$ es un 13.96 % mejor en el número medio de evaluaciones que $\varepsilon = 0.05$ que, por otra parte, es un 19.15 % mejor en el tiempo medio de ejecución. Debido a la poca diferencia existente entre ambos valores, optamos por seleccionar 0.15 como el valor de ε que ofrece mejores resultados.

  - Criterio **PopHR** (Tabla G.8): En este caso, la diferencia en el número medio de evaluaciones es del 19.51 % a favor de $\varepsilon = 0.15$, más del doble que la diferencia en el tiempo medio.

  - Criterio **PopHC** (Tabla F.8):También en este estudio obtenemos, a favor de $\varepsilon = 0.15$, una diferencia bastante más notable en el número medio de evaluaciones que en el tiempo medio.

  - Criterio **FitHC** (Tabla H.8): En este caso, la diferencia entre $\varepsilon = 0.05$ y $\varepsilon = 0.3$ en el tiempo medio de ejecución es del 22.43 % a favor del primero, el doble que en el número medio de evaluaciones, por lo que lo escogeremos como el mejor de los dos.

- Problema **MTTP100**:

- Criterio **AvgFitR** (Tabla D.9): La diferencia en el número medio de evaluaciones es del 18.95 % a favor de $\varepsilon = 0.3$, el triple que en el caso del tiempo medio de ejecución.

- Criterio **AvgFitC** (Tabla E.9): En este caso la diferencia es abismal: diferencia del 28.45 % en el número medio de evaluaciones en favor de $\varepsilon = 0.05$ con respecto a $\varepsilon = 0.15$ frente al 1.08 % del segundo con respecto al primero.

- Criterio **FitHC** (Tabla H.9): Aquí se nos presenta un caso similar al del punto anterior: diferencia del 15.41 % en el número de evaluaciones para $\varepsilon = 0.3$ frente al 0.36 % en el tiempo medio para $\varepsilon = 0.15$.

- Problema **MTTP200**:

  - Criterio **AvgFitC** (Tabla E.10): En este caso se ha elegido como mejor solución un ε distinto a los dos que aparecen en las tablas: $\varepsilon = 0.3$. Esto es porque, fijándonos en el número de evaluaciones, el caso en el que $\varepsilon = 0.05$ es mejor que $\varepsilon = 0.15$, pero muy similar a $\varepsilon = 0.3$, con una diferencia entre ambas de 1800 evaluaciones, que no es muy significativa en las magnitudes que nos movemos (unas 500000 evaluaciones). Por otro lado, si nos fijamos en el tiempo medio empleado en encontrar solución, el caso de $\varepsilon = 0.15$ es mejor que $\varepsilon = 0.05$, y en este caso el mejor resultado también tiene un valor similar a $\varepsilon = 0.3$ (con una diferencia de 9 segundos en magnitudes de unos 1500). Por esto, se decide que, aunque no obtenga el mejor de los resultados en ninguno de los dos casos, teniendo en cuenta ambos casos a la vez, es el mejor resultado porque se aproxima mucho al mejor valor obtenido tanto según el número medio de evaluaciones como según el tiempo medio de ejecución.

  - Criterio **PopHC** (Tabla G.10): Como la diferencia entre ambos resultados en cuanto al tiempo medio no es grande, decidimos utilizar como mejor resultado para este problema el correspondiente a $\varepsilon = 0.05$.

- Problema **P-PEAKS**:

  - Criterio **AvgFitR** (Tabla D.11): Como mejor resultado hemos elegido $\varepsilon = 0.25$ porque es más significativa la diferencia en el número de evaluaciones que en el tiempo medio de ejecución, que es de tan solo 1.3 segundos en una magnitud de unos 170 segundos.

- Criterio **PopHC** (Tabla G.11): En este caso es mucho más significativa la diferencia en el tiempo medio de ejecución que la existente en el número medio de evaluaciones entre los ϵ con 0.05 y 0.3. Por tanto, adoptaremos éste último epsilon como el que proporciona mejores valores.

- Criterio **FitHC** (Tabla H.11): En este caso, como en el anterior, es mucho más significativa la diferencia en el tiempo medio de ejecución (del $15,9\,\%$) que la existente en el número medio de evaluaciones (del $5.23\,\%$) entre los ϵ con 0.05 y 0.3, también como en el caso anterior. Por tanto, adoptaremos éste último epsilon como el que proporciona mejores valores.

En la tabla 8.4 aparecen los valores de ε que se ha ido considerando en los puntos anteriores que consiguen mejores valores en general (teniendo en cuenta el número medio de evaluaciones, el tiempo medio y el porcentaje de soluciones encontradas). En ella se ha marcado en negrita el criterio que da mejor resultado (teniendo en cuenta los tres parámetros también) para cada problema.

En la Tabla 8.5 se han detallado los valores del número medio de evaluaciones y tiempo medio de ejecución de los criterios y problemas descritos en la Tabla 8.4. Vemos que el criterios auto-aptativo que funciona mejor en un mayor número de problemas es el primero de los que se estudió, el AvgFitR, que obtiene los mejores resultados para los problemas MMDP, MTTP200 (conjuntamente con FitHC) y COUNTSAT. El criterio PopHR es el mejor en los problemas ECC y P-PEAKS, mientras que con MAXCUT100 funciona mejor AvgFitC. Son de destacar los malos resultados obtenidos con PopHC ya que no consigue ser mejor para ningún problema.

Si comparamos los resultados de la Tabla 8.5 con los de las tablas correspondientes a las rejillas fijas y dinámicas preprogramadas del Apéndice A, comprobamos que hemos conseguido, utilizando criterios auto-adaptativos, mejorar considerablemente los mejores resultados que se consiguieron con dichas rejillas; salvo en el caso del problema P-PEAKS, en el que se obtiene un valor similar. Esta mejora llega hasta el punto de que, en el caso del problema MTTP20, el peor resultado obtenido para cualquiera de los ε y criterio estudiados resulta ser mejor (en cuanto al número medio de evaluaciones) que el mejor de entre los obtenidos con cualquiera de las rejillas fijas y dinámicas con cambio preprogramado.

Estudiando la Tabla 8.5 nos damos cuenta de que los problemas que poseen un mayor nivel de epistasis funcionan mejor con los criterios basados en la entropía de la población; mientras que los problemas con elevada multimodalidad (MMDP), con máximos locales muy próximos a

los globales (de los que es muy difícil salir si se cae) (COUNTSAT) o los problemas que son muy duros de resolver computacionalmente (MTTP200 y MAXCUT100) funcionan mejor con criterios basados en el fitness de los individuos. Por último, cabe destacar lo bien que funciona el criterio FitHC (obtiene el mejor resultado en número medio de evaluaciones) con un problema tan duro computacionalmente como MTTP200.

Problema\Criterio	AvgFitR	AvgFitC	PopHR	PopHC	FitHC
COUNTSAT	**0.15**	0.3	0.15	0.15	0.3
ECC	0.15	0.15	**0.3**	0.3	0.15
FMS	0.05	0.05	0.25	0.3	0.3
MAXCUT20.01	0.05	0.15	**0.05**	0.05	0.3
MAXCUT20.09	0.05	0.25	0.05	**0.05**	0.3
MAXCUT100	0.25	**0.05**	0.25	0.25	0.15
MMDP	**0.3**	0.25	0.3	0.05	0.15
MTTP20	0.15	0.15	0.15	**0.15**	0.05
MTTP100	0.3	**0.05**	0.05	0.05	0.3
MTTP200	**0.3**	0.3	0.05	0.05	**0.3**
P-PEAKS	0.25	0.15	**0.15**	0.3	0.3

Tabla 8.4: Mejores valores de ε para cada problema y criterio

Prob\Crit	AvgFitR	AvgFitC	PopHR	PopHC	FitHC
COUNTSAT	**13825.48; 6.92**	16039.00; 6.43	16018.95; 6.20	17178.68; 6.79	19931.06; 7.88
ECC	159530.20; 714.87	153983.00; 694.23	**140095.00; 746.70**	148409.10; 668.60	167590.30; 790.30
FMS	663624.30; 122893	676631.80; 1258.00	**566354.20; 1790.25**	660591; 1247.82	684873.60; 1307.54
MAXCUT20.01	15036.50; 7.23	15303.83; 6.60	**12590.40; 6.17**	13953.80; 6.83	15985.53; 6.77
MAXCUT20.09	28216.03; 15.37	26251.13; 11.30	26705.60; 13.20	**24246.13; 12.00**	32720.60; 16.63
MAXCUT100	266664.00; 601.64	**226263.30; 633.92**	264069.30; 626.12	299078.20; 732.33	252906.60; 831.46
MMDP	**147032.30; 512.38**	167334.80; 580.67	155111.70; 549.85	234111.40; 976.00	152321.70; 547.54
MTTP20	7684.83; 10.10	7577.90; 9.40	7390.77; 9.10	**7257.10; 9.37**	8633.87; 7.47
MTTP100	199790.60; 436.17	**176011.30; 411.07**	261557.90; 483.70	225628.30; 443.60	205818.90; 401.93
MTTP200	484366.90; 1377.27	493977.50; 1456.30	520202.90; 1510.73	535721.60; 1769.27	**476133.00; 1397.13**
P-PEAKS	54695.40; 179.70	54254.30; 176.17	**48292.77; 156.20**	48653.67; 156.30	54441.43; 180.23

Tabla 8.5: Núm. med. de evaluaciones y tiempo med. de los mejores valores de ε para cada problema y criterio

A continuación se presentan las gráficas que muestran la evolución del cambio de las rejillas para el mejor criterio en las instancias más grandes de cada uno de los problemas (marcados en negrita en la Tabla 8.5). Concretamente están representadas en cada gráfica tres ejecuciones de las treinta que se han hecho: la mejor, la peor y una ejecución típica (el que aparece en negrita en la Tabla 8.5).

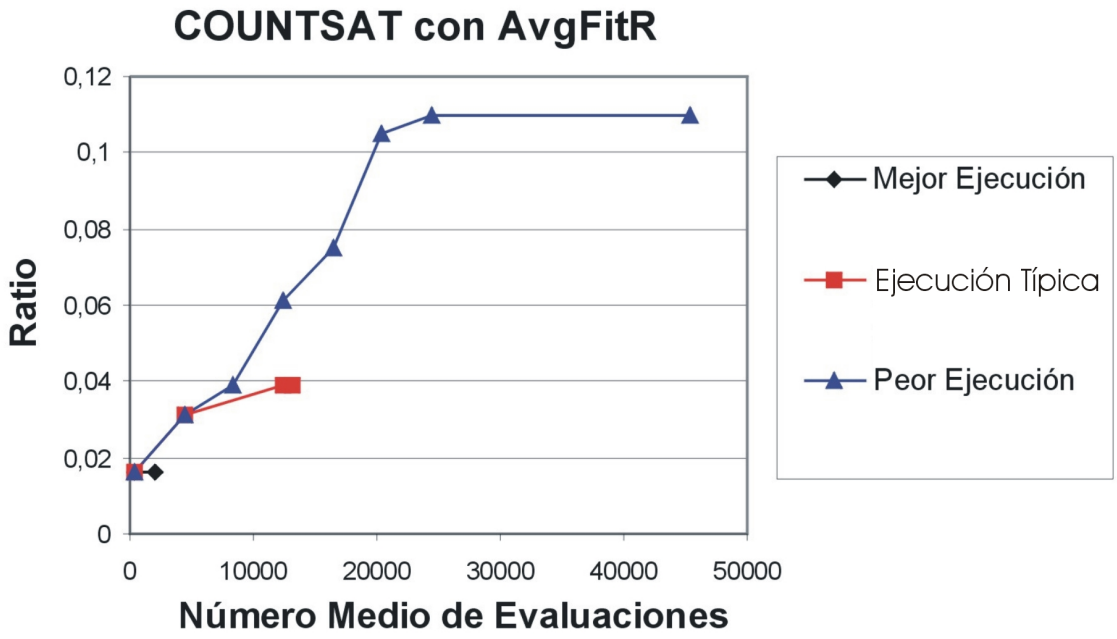

Figura 8.1: Cambios de rejilla en tres ejecuciones de COUNTSAT con AvgFitR

En la Figura 8.1 se aprecia una tendencia a evolucionar siempre hacia rejillas de mayor ratio por parte del algoritmo. Esto nos indica que se produce una constante mejora del *fitness* medio a lo largo de las generaciones que va realizando el cGA. Existe en el caso de COUNTSAT una gran diferencia en el número de evaluaciones realizadas entre la mejor ejecución y la peor. Esto se debe a que en los casos en los que algún individuo de la población se inicializa cerca de la solución el algoritmo evoluciona rápidamente hacia dicha solución, llegando a ella en muy pocas generaciones. Pero si ningún individuo se inicializa cerca de la solución, el algoritmo evolucionará normalmente hacia el máximo local y, debido a las características de esta función (ver Sección 4.2, le será muy difícil abandonar dicho máximo globel para poder alcanzar el máximo global.

Observando las tablas de los apéndices F y G, podemos darnos cuenta del elevado número de cambios de rejilla que se producen. El problema ECC no es una excepción. La presencia de tantos cambios de rejilla nos indica que la entropía de la población decrece muy poco en muchas ocasiones, o incluso se mantiene o hasta crece.

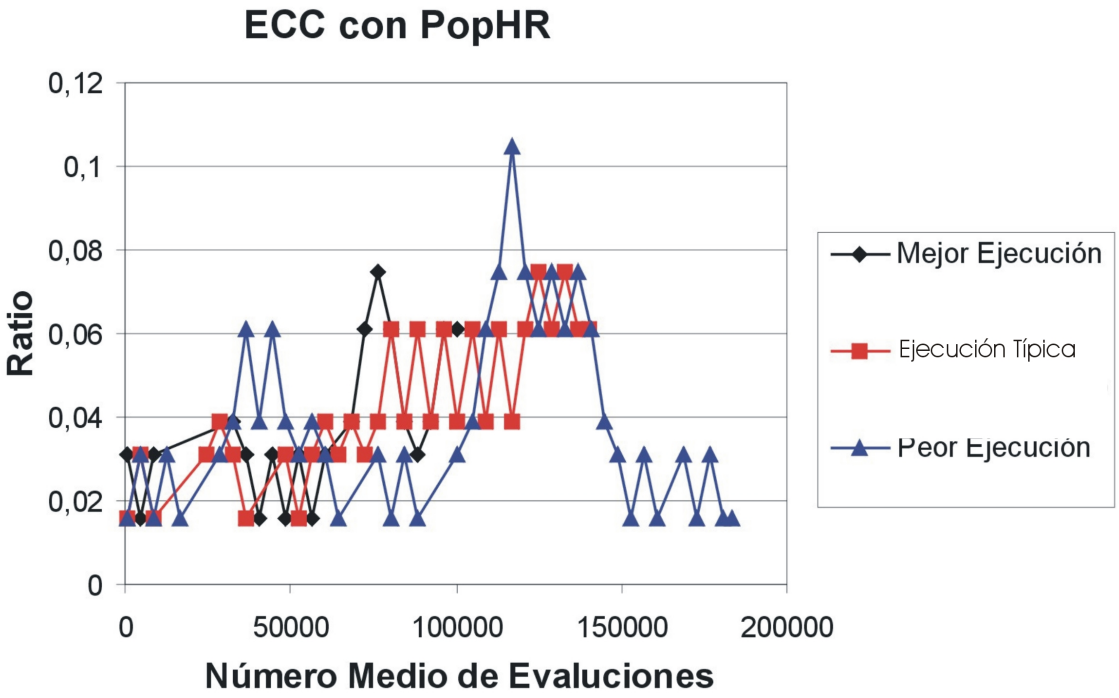

Figura 8.2: Cambios de rejilla en tres ejecuciones de ECC con PopHR

Vemos también en la Figura 8.2 que las tres ejecuciones representadas están comprendidas entre las 100000 evaluaciones y cerca de las 200000 un rango de valores mucho menor que en el caso de COUNTSAT que hemos visto anteriormente.

Observando la Figura 8.3 podemos confirmar con este caso lo que se comentó en el problema anterior (ECC con PopHR), ya que, al utilizar un criterio basado en la entropía, volvemos a obtener un gran número de cambios de rejilla. Este problema encuentra, con respecto a los demás estudiados, una grandísima dificultad para encontrar la solución. Podemos ver que este problema se acentúa con el hecho de que no hay ninguna tendencia fija en el cambio de las rejillas: durante una ejecución se puede pasar de la rejilla de ratio 0.016, la más cuadrada, a la 0.11, la más estrecha, y de ésta volver a la 0.016 para después seguir evolucionando también sin ningún patrón aparente; son casos claros de este fenómeno la mejor ejecución y la más próxima al caso medio.

En el caso de la Figura 8.4 vemos que existe una clara tendencia de evolución hacia la rejilla de ratio 0.11. Por otro lado, se observa que al pasar de las 300000 evaluaciones (sólo en el peor de los tres casos representados se llega a tal número) el algoritmo cambia mucho entre la rejilla 0.11 y 0.105. Esto es debido a que llega un momento en el que el valor de la media de los *fitness* de los individuos que forman la población se queda prácticamente estable entre las distintas generaciones.

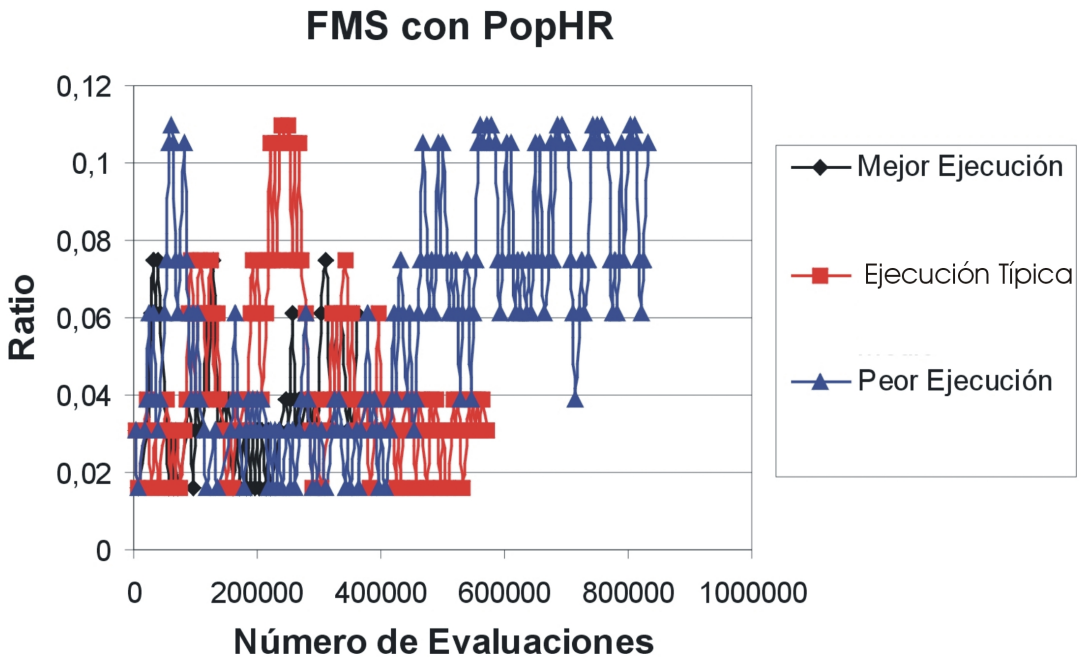

Figura 8.3: Cambios de rejilla en tres ejecuciones de FMS con PopHR

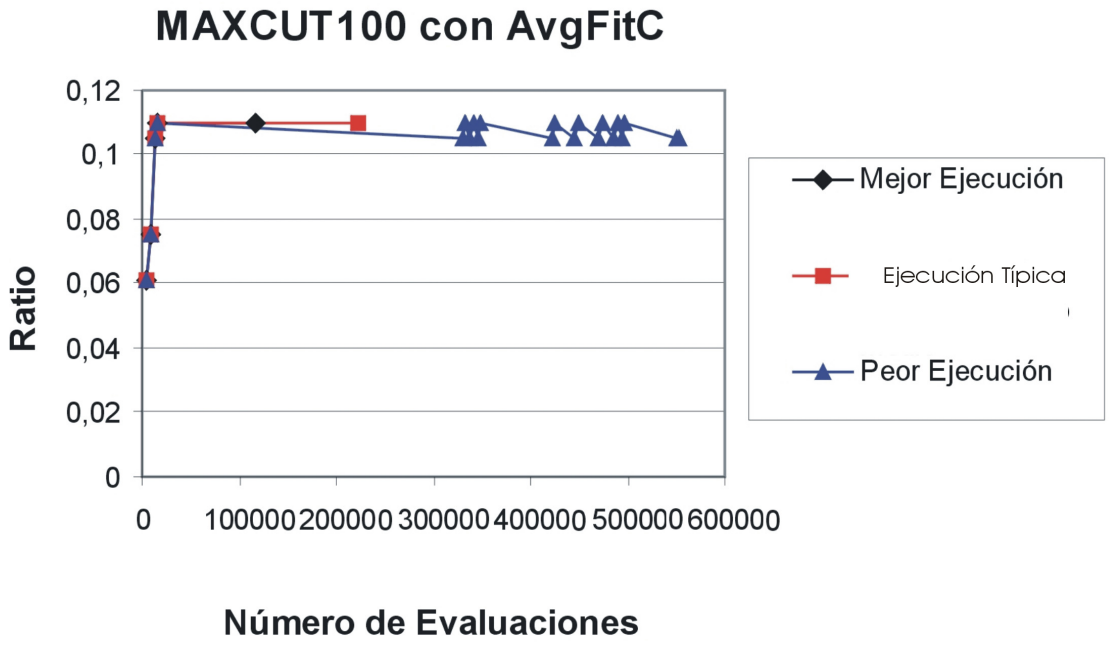

Figura 8.4: Cambios de rejilla en tres ejecuciones de MAXCUT100 con AvgFitC

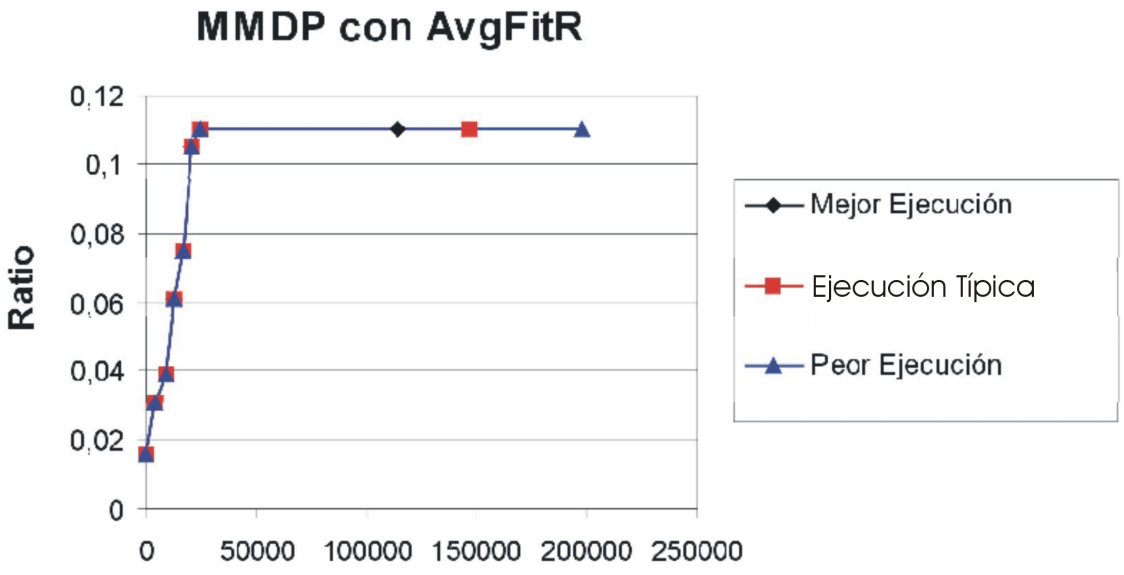

Figura 8.5: Cambios de rejilla en tres ejecuciones de MMDP con AvgFitR

En la Figura 8.5 vemos que se aprecia una clara tendencia de crecimiento en cada generación del valor de medio de los *fitness* de los individuos. Hay una rápida evolución hacia las rejillas de mayor ratio hasta que se alcanza el extremo (la rejilla más cuadrada, de ratio 0.11) y no puede continuar incrementando el ratio en las distintas generaciones.

En cualquiera de las tres ejecuciones representadas en la Figura 8.5 se aprecia el mismo comportamiento en cuanto a la evolución de las rejillas: se produce una rápida evolución hacia la rejilla de ratio 0.11 y, una vez alcanzada dicha rejilla, el algoritmo permanece utilizándola hasta el final de la ejecución; lo que indica que la media de los valores de *fitness* de los individuos permanece creciendo hasta el final de la ejecución.

El problema representado en la Figura 8.6 es el MTTP con la instancia de 200 tareas (mttp200.dat). En ella vemos que se produce una evolución hacia la rejilla cuadrada; pero esta evolución es menos rápida en los casos de la mejor evolución y la más próxima al caso medio. Aunque, como vemos, las tres evaluaciones alcanzan rápidamente la rejilla de ratio 0.11 y terminan su ejecución en ella.

Por último, en la Figura 8.7, vemos que en los casos de la peor y la mejor ejecución, el algoritmo

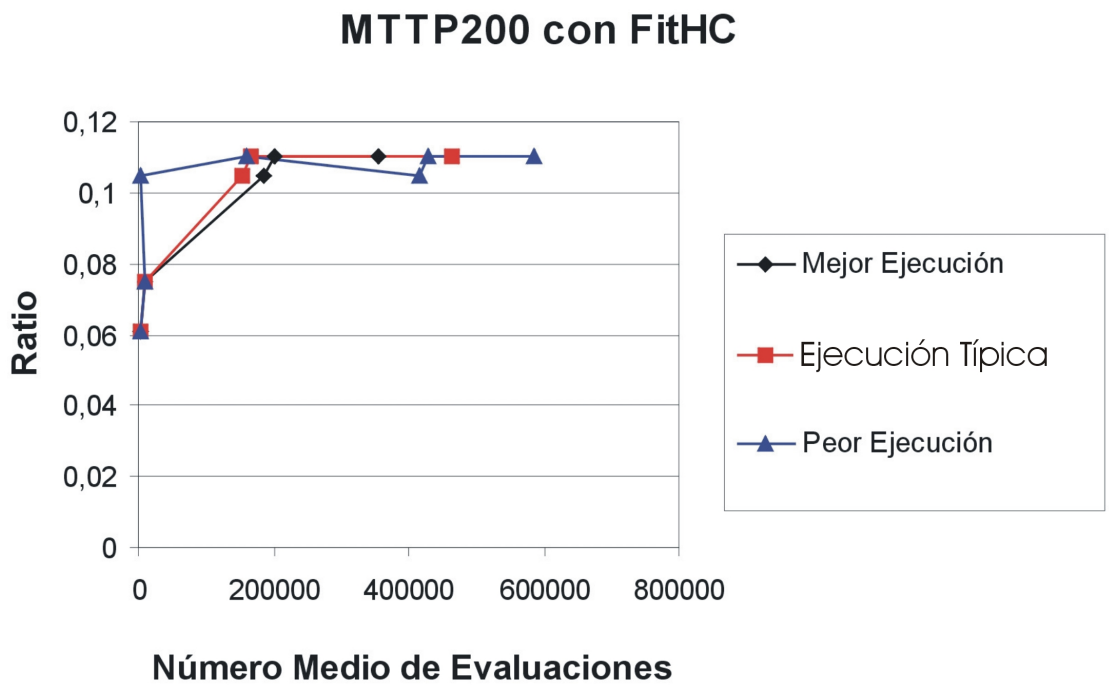

Figura 8.6: Cambios de rejilla en tres ejecuciones de MTTP200 con FitHC

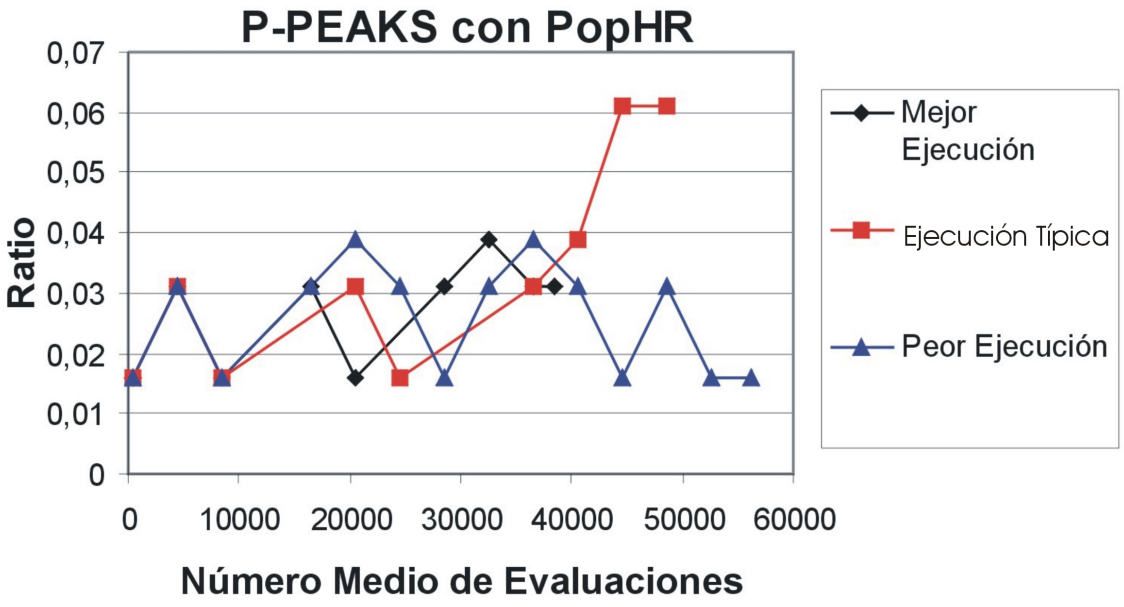

Figura 8.7: Cambios de rejilla en tres ejecuciones de P-PEAKS con PopHR

utiliza siempre las rejillas más estrechas, terminando la ejecución en las rejillas de ratio 0.016 y 0.031, respectivamente. Por otro lado, la ejecución más próxima al caso medio termina en una rejilla intermedia, la de ratio 0.061. Debido a la altísima epistasis que tiene este problema, la rejilla con la que termina la ejecución del mismo es muy próxima a la rejilla con la que comenzó, ya que no se puede adivinar una cierta tendencia en la evolución de los individuos cuando nos fijamos en el valor de la entropía de la población.

Capítulo 9

Conclusiones y Trabajo Futuro

9.1. Conclusiones

En este trabajo se ha realizado el estudio del comportamiento de un cGA en función de la ratio existente entre el radio de la población y el del vecindario. Para ello se han propuesto una serie de problemas de características diversas, y se han resuelto utilizando diversos tipos de rejillas. Estas rejillas son tanto fijas, que no cambian durante toda la ejecución, como dinámicas preprogramadas, que cambian una sola vez y en un momento concreto de la ejecución, o con rejillas dinámicas auto-adaptativas, que van cambiando automáticamente de tamaño a lo largo de la evolución según los valores de ciertos parámetros que se vayan obteniendo durante la ejecución. La novedad que supone este último caso en el estudio de los cGAs es lo que dota de mayor interés a este trabajo.

Tras la realización de este estudio hemos podido observar que, gracias a la introducción de las rejillas dinámicas auto-adaptativas en el campo de los cGAs, se han conseguido mejorar los resultados existentes en los trabajos publicados hasta la fecha, que sólo utilizaban rejillas fijas o dinámicas con cambio preprogramado, o bien trataban de optimizar el algoritmo mediante el estudio de otros parámetros distintos de la ratio vecindad/población. Por otro lado, hemos conseguido obtener un modelo de GA más robusto y fácil de emplear, debido a que ya no es necesario considerar el parámetro correspondiente a la forma de la rejilla a utilizar. Esto es importante ya que este parámetro, como hemos visto a lo largo de todo este estudio, es, en muchos casos, absolutamente determinante en todos los aspectos del rendimiento del algoritmo (tiempo, obtención de soluciones, coste computacional...); por lo que supone un gran problema la elección correcta de su valor, siendo necesaria para ello la realización de múltiples pruebas experimentales.

Cualquiera de los criterios auto-adaptativos estudiados en este trabajo obtienen mejores resultados (en lo que respecta al número medio de evaluaciones) que el mejor de los obtenidos con las

rejillas fijas y dinámicas de cambio preprogramado en la mayoría de los problemas; mientras que para el resto de los problemas obtiene valores parecidos.

9.2. Comparación de los Criterios Dinámicos Auto-Adaptativos con los de Rejilla Fija y Dinámica Preprogramada

En las tablas 9.1 y 9.2 se muestra un resumen de todos los valores obtenidos durante el presente estudio. Son de especial utilidad si nos proponemos la comparación de los valores obtenidos al introducir la novedad de utilizar rejillas dinámicas auto-adaptativas frente a los obtenidos con las rejillas fijas y dinámicas con cambio preprogramado. Se han resaltado en negrita los mejores valores encontrados para cada problema durante todo el presente estudio. Además, se han utilizado números en cursiva para destacar, para cada problema, los valores de la tabla 9.1 que resultan ser mejores que el mejor de la tabla 9.2. La comparación se ha realizado teniendo en cuenta solamente el número medio de evaluaciones realizadas durante la ejecución del problema, ya que se ha considerado como el parámetro más representativo para realizar dicha comparación. En dichas tablas vemos que todos los problemas obtienen mejor (o similar) resultado con cualquier criterio auto-adaptativo que con el mejor de los casos de las rejillas fijas y dinámicas con cambio preprogramado; excepto en los casos de ECC y P-PEAKS con AvgFitR, AvgFitC y FitHC, y MAXCUT100 con todos los criterios excepto con AvgFitC. No se ha encontrado ningún criterio que dé para cualquier problema siempre un mejor valor que el obtenido con las rejillas fijas y preprogramadas, pero cabe destacar que PopHR y PopHC lo consiguen en todos los problemas salvo en MAXCUT100; y que cada uno de los demás criterios no lo consiguen sólo en dos o tres de todos los problemas estudiados.

Prob\Crit	AvgFitR	AvgFitC	PopHR	PopHC	FitHC
ECC	159530.20; 714.87	153983.00; 694.23	***140095.00; 746.70***	*148409.10; 668.60*	167590.30; 790.30
FMS	663624.30; 122893	676631.80; 1258.00	***566354.20; 1790.25***	660591; 1247.82	684873.60; 1307.54
MAXCUT20.01	*15036.50; 7.23*	*15303.83; 6.60*	***12590.40; 6.17***	*13953.80; 6.83*	*15985.53; 6.77*
MAXCUT20.09	*28216.03; 15.37*	*26251.13; 11.30*	*26705.60; 13.20*	***24246.13; 12.00***	*32720.60; 16.63*
MAXCUT100	266664.00; 601.64	***226263.30; 633.92***	264069.30; 626.12	299078.20; 732.33	252906.60; 831.46
MMDP	***147032.30; 512.38***	*167334.80; 580.67*	*155111.70; 549.85*	*234111.40; 976.00*	*152321.70; 547.54*
MTTP20	*7684.83; 10.10*	*7577.90; 9.40*	*7390.77; 9.10*	***7257.10; 9.37***	*8633.87; 7.47*
MTTP100	*199790.60; 436.17*	***176011.30; 411.07***	*261557.90; 483.70*	*225628.30; 443.60*	*205818.90; 401.93*
MTTP200	*484366.90; 1377.27*	*493977.50; 1456.30*	*520202.90; 1510.73*	*535721.60; 1769.27*	***476133.00; 1397.13***
P-PEAKS	54695.40; 179.70	54254.30; 176.17	***48292.77; 156.20***	*48653.67; 156.30*	54441.43; 180.23
COUNTSAT	***13825.48; 6.92***	*16039.00; 6.43*	*16018.95; 6.20*	*17178.68; 6.79*	*19931.06; 7.88*

Tabla 9.1: Núm. med. de evaluaciones y tiempo med. de los mejores valores de ε para cada problema y criterio

Prob\Crit	*Square*	*Rectangular*	*Narrow*	*SqNar*	*NarSq*
ECC	159102.40; 691.43	**144332.30; 625.30**	151724.00; 658.23	163018.90; 28792.33	175891.00; 764.70
FMS	612112.10	594093.90	593519.10	692205.20	**575353.80**
MAXCUT20.01	18364.80; 7.73	18899.47; 7.90	17375.67; 7.30	**15851.87; 7.47**	24206.03; 10.07
MAXCUT20.09	41703.00; 17.70	**36610.30; 17106.65**	37679.63; 14234.67	42825.80; 17.93	40045.53; 16.70
MAXCUT100	256826.10; 562.13	**232141.50; 516.18**	357101.30; 788.71	415788.80; 938.12	284057.40; 633.00
MMDP	**170637.90**	237003.40	331719.60	1002900.00	1002900.00
MTTP20	18017.27; 7.00	17923.70; 6.87	19514.33; 7.47	**15838.50; 6.77**	19420.77; 7.53
MTTP100	294613.70; 435.50	**274162.70; 434.33**	303636.20; 507.00	284829.30; 426.33	303061.40; 454.73
MTTP200	673598.80; 1924.57	688850.20; 2174.26	566999.60; 12625.50	**521419.30; 1474.03**	572132.40; 74941.82
P-PEAKS	50458.10	50364.60	48653.60	57769.70	**48012.10**
COUNTSAT	24326.33; 9.67	25214.82; 9.77	21736.54; 8.33	22637.27; 4688.44	**19720.91; 7.73**

Tabla 9.2: Núm. med. de evaluaciones y tiempo med. para cada problema con las rejillas fijas y dinámicas con cambio preprogramado

9.3. Trabajo Futuro

En cuanto al trabajo futuro, el siguiente paso natural de estudio sería considerar la posibilidad de la realización de nuevos criterios para la auto-adaptación del tamaño de la rejilla, así como considerar otros posibles parámetros indicadores de la evolución del algoritmo para tener en cuenta en dichos criterios (además de la entropía de la población y del fitness de los individuos). También es interesante un futuro estudio sobre la repercusión que pueda tener la forma de redistribuir por la rejilla de la población a los individuos que la forman tras cambiar la forma de ésta. En este trabajo este cambio está descrito en la Sección 2.5.3, pero sería interesante comprobar los efectos que pueden repercutir al redistribuir los individuos en la rejilla siguiendo ciertos criterios:

- Intentar mantener los vecindarios lo más agrupados posible: esto da como resultado que casi ningún vecindario original se mantenga completo tras el cambio.
- Dar preferencia a los mejores individuos a la hora de mantener los vecindarios: esto propiciaría la formación de nuevas poblaciones en las que los individuos con mejor valor de adecuación mantienen sus vecindarios intactos o casi intactos. Mientras que los vecindarios formados por peores individuos se deshacen. Esto fomenta continuar con la búsqueda por los caminos que han dado mejores resultados hasta el momento y la creación de nuevos caminos utilizando los individuos que no estaban cerca de la solución.

Otro campo interesante para futuros estudios consiste en la ampliación de los problemas estudiados a otros que posean características distintas y/o de corte real, como por ejemplo problemas de optimización dinámica, cuyas funciones de evaluación cambian con el tiempo, o el problema de enrutado de vehículos (o VRP).

Apéndice A

Análisis Estadístico sobre los Resultados de las Rejillas Fijas y Dinámicas preprogramadas

En este apéndice se presentan las tablas con los valores producidos tras realizar el análisis estadístico sobre los resultados obtenidos durante las ejecuciones de los problemas ECC, MAX-CUT100, MTTP100, MTTP200 y COUNTSAT utilizando las rejillas fijas *Square* (de ratio 0.11), *Rectangular* (ratio 0.75) y *Narrow* (ratio 0.31) y de las rejillas dinámicas preprogramadas que cambian de la rejilla *Square* a la *Narrow* (*SqNar*) y de la *Narrow* a la cuadrada (*NarSq*). Este cambio se realiza cuando se supera durante la ejecución el número de evaluaciones correspondiente a la mitad del número medio de evaluaciones realizado con la rejilla cuadrada por el mismo problema.

En las tablas aparece detallado para cada rejilla el número medio de evaluaciones necesario para encontrar la solución (en el caso en que se encuentre) durante 30 ejecuciones, el tiempo medio en segundos empleado en encontrar dicha solución y el porcentaje de ejecuciones que encontraron solución.

p-values	*Square*	*Rectangular*	*Narrow*	*SqNar*	*NarSq*
Square	-	**0.04321**	0.2731	0.5861	**0.04605**
Rectangular	**0.04321**	-	0.2928	**0.01466**	**4.527e-4**
Narrow	0.2731	0.2928	-	0.1106	**4.172e-3**
SqNar	0.5861	**0.01466**	0.1106	-	0.1341
NarSq	**0.04605**	**4.527e-4**	**4.172e-3**	0.1341	-

Tabla A.1: Análisis estadístico de los resultados sobre ECC

p-values	*Square*	*Rectangular*	*Narrow*	*SqNar*	*NarSq*
Square	-	0.817	0.6187	0.2567	**0.01757**
Rectangular	0.817	-	0.4889	0.2064	**0.04289**
Narrow	0.6187	0.4889	-	0.4658	**4.155e-3**
SqNar	0.2567	0.2064	0.4658	-	**1.298e-3**
NarSq	**0.01757**	**0.04289**	**4.155e-3**	**1.298e-3**	-

Tabla A.2: Análisis estadístico de los resultados sobre MAXCUT con el grafo cut20.01

p-values	*Square*	*Rectangular*	*Narrow*	*SqNar*	*NarSq*
Square	-	0.2536	0.3263	0.8156	0.714
Rectangular	0.2536	-	0.7934	0.2	0.4483
Narrow	0.3263	0.7934	-	0.2548	0.5712
SqNar	0.8156	0.2	0.2548	-	0.5703
NarSq	0.714	0.4483	0.5712	0.5703	-

Tabla A.3: Análisis estadístico de los resultados sobre MAXCUT con el grafo cut20.09

p-values	*Square*	*Rectangular*	*Narrow*	*SqNar*	*NarSq*
Square	-	0.5435	0.1449	**0.02595**	0.6493
Rectangular	0.5435	-	**0.07638**	**0.01234**	0.3967
Narrow	0.1449	**0.07638**	-	0.4987	0.3688
SqNar	**0.02595**	**0.01234**	0.4987	-	0.1110
NarSq	0.6493	0.3967	0.3688	0.1110	-

Tabla A.4: Análisis estadístico de los resultados sobre MAXCUT con el grafo cut100

p-values	*Square*	*Rectangular*	*Narrow*	*SqNar*	*NarSq*
Square	-	0.9593	0.3706	0.224	0.4469
Rectangular	0.9593	-	0.2698	0.1862	0.3616
Narrow	0.3706	0.2698	-	**8.808e-3**	0.9482
SqNar	0.224	0.1862	**8.808e-3**	-	**0.02571**
NarSq	0.4469	0.3616	0.9482	**0.02571**	-

Tabla A.5: Análisis estadístico de los resultados sobre MTTP20

p-values	*Square*	*Rectangular*	*Narrow*	*SqNar*	*NarSq*
Square	-	0.1582	0.5422	0.5782	0.5992
Rectangular	0.1582	-	**0.01641**	0.4871	**0.0369**
Narrow	0.5422	**0.01641**	-	0.2346	0.9672
SqNar	0.5782	0.4871	0.2346	-	0.2836
NarSq	0.5992	**0.0369**	0.9672	0.2836	-

Tabla A.6: Análisis estadístico de los resultados sobre MTTP100

p-values	*Square*	*Rectangular*	*Narrow*	*SqNar*	*NarSq*
Square	-	0.6196	**1.158e-4**	**2.759e-7**	**2.403e-4**
Rectangular	0.6196	-	**1.533e-5**	**3.129e-8**	**3.334e-5**
Narrow	**1.158e-4**	**1.533e-5**	-	**0.01961**	0.7888
SqNar	**2.759e-7**	**3.129e-8**	**0.01961**	-	**0.01084**
NarSq	**2.403e-4**	**3.334e-5**	0.7888	**0.01084**	-

Tabla A.7: Análisis estadístico de los resultados sobre MTTP200

p-values	*Square*	*Rectangular*	*Narrow*	*SqNar*	*NarSq*
Square	-	0.8772	0.6286	0.7854	0.3652
Rectangular	0.8772	-	0.3833	0.6139	0.1261
Narrow	0.6286	0.3833	-	0.8454	0.4602
SqNar	0.7854	0.6139	0.8454	-	0.4944
NarSq	0.8244	0.6733	0.5908	0.4944	-

Tabla A.8: Análisis estadístico de los resultados sobre COUNTSAT

p-values	*Square*	*Rectangular*	*Narrow*	*SqNar*	*NarSq*
Square	-	**0.0177**	**5.12e-06**	**2.2e-16**	**2.2e-16**
Rectangular	**0.0177**	-	**0.0181**	**2.2e-16**	**2.2e-16**
Narrow	**5.12e-06**	**0.0181**	-	**2.2e-16**	**2.2e-16**
SqNar	**2.2e-16**	**2.2e-16**	**2.2e-16**	-	1.0
NarSq	**2.2e-16**	**2.2e-16**	**2.2e-16**	1.0	-

Tabla A.9: Análisis estadístico de los resultados sobre MMDP

p-values	*Square*	*Rectangular*	*Narrow*	*SqNar*	*NarSq*
Square	-	0.6818	0.7128	0.1089	0.4115
Rectangular	0.6818	-	0.9904	**0.0397**	0.6536
Narrow	0.7128	0.9904	-	**0.0665**	0.7087
SqNar	0.1089	**0.0397**	**0.0665**	-	**0.0164**
NarSq	0.4115	0.6536	0.7087	**0.0164**	-

Tabla A.10: Análisis estadístico de los resultados sobre FMS

p-values	*Square*	*Rectangular*	*Narrow*	*SqNar*	*NarSq*
Square	-	0.9409	**0.0829**	**2.852e-08**	**0.0212**
Rectangular	0.9409	-	0.1531	**3.514e-07**	**0.0531**
Narrow	**0.0829**	0.1531	-	**8.03e-12**	0.4966
SqNar	**2.852e-08**	**3.514e-07**	**8.03e-12**	-	**1.061e-12**
NarSq	**0.0212**	**0.0531**	0.4966	**1.061e-12**	-

Tabla A.11: Análisis estadístico de los resultados sobre P-PEAKS

Apéndice B

Resultados Obtenidos con las Rejillas Dinámicas Auto-Adaptativas con Todos los ε Empleados

A continuación se detallan las tablas que se han utilizado para seleccionar los ε más representativos para el estudio del comportamiento de los distintos problemas con los criterios basados en la entropía, en el fitness y el conjugado de ambos. Todas las tablas contienen el número medio de evaluaciones necesario para encontrar la solución óptima durante 30 ejecuciones y el tiempo medio empleado para encontrarla expresado en segundos, ambos valores están acompañados de su desviación estándar. Además se incluyen el porcentaje medio de cambios de rejilla por evaluación y el porcentaje total de soluciones encontradas.

ε	N Medio de Evaluaciones	Tiempo Medio(seg)	Comprobaciones de Rejilla	Soluciones
0.05	206168.62 ± 156521.00	550.26 ± 296.12	12.42 %	60 %
0.1	2632601.20 ± 136482.34	603.42 ± 310.83	9.67 %	70 %
0.15	245210.50 ± 128066.10	574.63 ± 301.08	9.95 %	80 %
0.2	362187.26 ± 186257.65	875.24 ± 498.26	9.75 %	60 %
0.25	340849.00 ± 199759.20	854.50 ± 500.95	9.96 %	60 %
0.3	296739.00 ± 153597.30	690.11 ± 357.78	9.96 %	90 %
0.4	367805.10 ± 242243.30	845.44 ± 556.66	9.92 %	90 %

Tabla B.1: MAXCUT100 y AvgFitεR en 10 ejecs con todos los ε estudiados

ε	N Medio de Evaluaciones	Tiempo Medio(seg)	Comprobaciones de Rejilla	Soluciones
0.01	385360.00 ± 310770.40	881.25 ± 715.22	6.48 %	40 %
0.05	290082.40 ± 121767.60	674.60 ± 280.36	7.17 %	50 %
0.1	202905.00 ± 58656.67	534.50 ± 154.89	9.45 %	40 %
0.15	202905.00 ± 58656.67	534.50 ± 154.89	9.82 %	40 %
0.2	356812.21 ± 263026.00	793.35 ± 534.21	9.91 %	60 %
0.25	342052.00 ± 226485.00	787.11 ± 521.07	9.96 %	90 %
0.3	367649.20 ± 236372.90	878.00 ± 589.25	9.97 %	60 %
0.4	274951.30 ± 180907.90	634.44 ± 420.23	9.53 %	90 %

Tabla B.2: MAXCUT100 y AvgFitrR en 10 ejecs con todos los ε estudiados

ε	N Medio de Evaluaciones	Tiempo Medio(seg)	Comprobaciones de Rejilla	Soluciones
0.01	342987.70 ± 204699.10	790.33 ± 479.77	9.67 %	30 %
0.05	192479.00 ± 0.00	439.00 ± 0.00	9.67 %	10 %
0.1	296324.35 ± 195423.21	680.63 ± 453.26	9.60 %	30 %
0.15	274817.70 ± 176119.10	676.00 ± 430.97	9.96 %	30 %
0.2	351269.23 ± 265384.16	765.06 ± 584.08	9.95 %	50 %
0.25	327856.60 ± 245987.50	749.00 ± 575.64	9.93 %	50 %
0.3	319653.30 ± 122425.60	742.57 ± 303.99	9.96 %	70 %
0.4	170825.00 ± 0.00	395.00 ± 0.00	9.35	10 %

Tabla B.3: MAXCUT100 y AvgFitsR en 10 ejecs con todos los ε estudiados

ε	N Medio de Evaluaciones	Tiempo Medio(seg)	Comprobaciones de Rejilla	Soluciones
0.005	334433.00 ± 160072.90	774.67 ± 368.66	9.67 %	30 %
0.01	391976.50 ± 215438.60	986.25 ± 541.88	10.31 %	80 %
0.025	324185.20 ± 237112.90	756.22 ± 554.97	9.74 %	90 %
0.05	225427.80 ± 97369.68	527.50 ± 227.63	10.86 %	60 %
0.075	313480.80 ± 121090.70	786.25 ± 304.84	9.50 %	40 %
0.1	274326.20 ± 86321.04	603.53 ± 183.25	9.87 %	40 %
0.15	255837.00 ± 74170.19	591.50 ± 171.27	9.96 %	40 %
0.2	395126.08 ± 196223.62	902.35 ± 442.23	9.84 %	60 %
0.25	370332.50 ± 185560.10	858.83 ± 429.79	9.97 %	60 %
0.3	298543.50 ± 223720.80	589.50 ± 673.87	10.09 %	20 %

Tabla B.4: MAXCUT100 y AvgFitR en 10 ejecs con todos los ε estudiados

ε	N Medio de Evaluaciones	Tiempo Medio(seg)	Comprobaciones de Rejilla	Soluciones
0.1	516126.10 ± 74157.77	791.10 ± 116.76	99.67 %	100 %
0.2	432958.70 ± 111822.70	667.00 ± 173.83	98.24 %	100 %
0.3	433840.90 ± 76219.18	693.90 ± 123.33	97.35 %	100 %
0.4	439294.50 ± 121702.80	714.10 ± 200.60	99.56 %	100 %

Tabla B.5: MTTP100 y AvgFitεR en 10 ejecs con todos los ε estudiados

ε	N Medio de Evaluaciones	Tiempo Medio(seg)	Comprobaciones de Rejilla	Soluciones
0.01	271315.60 ± 43163.86	452.70 ± 71.12	99.60 %	100 %
0.05	330102.20 ± 64207.55	490.70 ± 101.39	96.24 %	100 %
0.1	331706.20 ± 55994.62	488.00 ± 82.96	97.33 %	100 %
0.2	396989.00 ± 62555.71	588.20 ± 93.40	98.90 %	100 %
0.3	389049.20 ± 82916.40	577.60 ± 127.23	96.20 %	100 %
0.4	414191.90 ± 95725.95	648.90 ± 144.37	98.34 %	100 %

Tabla B.6: MTTP100 y AvgFitrR en 10 ejecs con todos los ε estudiados

ε	N Medio de Evaluaciones	Tiempo Medio(seg)	Comprobaciones de Rejilla	Soluciones
0.01	249300.70 ± 56015.76	445.50 ± 101.02	95.65 %	100 %
0.05	282142.60 ± 44282.70	418.40 ± 65.86	98.26 %	100 %
0.1	381269.80 ± 67441.64	684.40 ± 120.92	94.50 %	100 %
0.2	237872.20 ± 28420.94	355.20 ± 46.52	97.22 %	100 %
0.3	264779.30 ± 60797.05	391.90 ± 89.76	95.93 %	100 %
0.4	252027.50 ± 49097.44	375.80 ± 75.36	92.75 %	100 %

Tabla B.7: MTTP100 y AvgFitsR en 10 ejecs con todos los ε estudiados

ε	N Medio de Evaluaciones	Tiempo Medio(seg)	Comprobaciones de Rejilla	Soluciones
0.005	274042.40 ± 58626.27	420.00 ± 86.16	98.24 %	100 %
0.01	221992.60 ± 60060.88	385.90 ± 104.76	98.75 %	100 %
0.025	260448.50 ± 53887.81	395.00 ± 82.36	94.85 %	100 %
0.05	246694.20 ± 57641.43	376.30 ± 88.44	99.37 %	100 %
0.075	269791.80 ± 54115.95	409.10 ± 81.89	96.80 %	100 %
0.1	267385.80 ± 61232.41	408.10 ± 96.18	98.27 %	100 %
0.15	257280.60 ± 41848.21	395.10 ± 68.95	94.06 %	100 %
0.2	268268.00 ± 360036.73	410.40 ± 57.78	96.98 %	100 %
0.25	257922.20 ± 35135.17	390.40 ± 53.08	94.22 %	100 %
0.3	293089.90 ± 30857.29	448.90 ± 43.28	97.72 %	100 %

Tabla B.8: MTTP100 y AvgFitR en 10 ejecs con todos los ε estudiados

ε	N Medio de Evaluaciones	Tiempo Medio(seg)	Comprobaciones de Rejilla	Soluciones
0.005	22283.14 ± 11405.28	8.71 ± 4.31	92.35 %	70 %
0.01	17643.00 ± 6390.89	7.57 ± 2.64	93.86 %	70 %
0.025	24259.50 ± 121673.56	9.83 ± 4.88	95.33 %	60 %
0.05	13862.14 ± 2708.43	5.57 ± 0.98	92.86 %	70 %
0.075	15988.88 ± 7333.05	6.13 ± 2.80	93.50 %	80 %
0.1	16497.29 ± 7517.83	6.57 ± 2.99	93.57 %	70 %
0.15	22455.00 ± 13411.04	9.00 ± 5.29	95.06 %	50 %
0.2	19096.63 ± 7201.87	7.50 ± 2.83	94.13 %	80 %
0.25	16529.11 ± 8592.96	6.89 ± 3.62	93.44 %	90 %
0.3	28298.14 ± 33529.98	10.86 ± 13.07	96.43 %	60 %

Tabla B.9: COUNTSAT y AvgFitR en 10 ejecs con todos los ε estudiados

Apéndice C

Resultados del Estudio de los Distintos Criterios Auto-Adaptativos

En este Apéndice se detallan las tablas con los valores medios obtenidos durante 10 ejecuciones en el estudio de los criterios auto-adaptativos presentados en las secciones 7.1 y 7.2. Todas las tablas contienen el número medio de evaluaciones y el tiempo medio empleado en resolver el problema junto con sus desviaciones estándar, y sólo se tienen en cuenta los valores obtenidos en los casos en los que se encuentra solución. Además de estos valores, también se detallan el porcentaje medio de cambios de rejilla realizados en cada ejecución y el porcentaje total de soluciones encontradas durante todas las ejecuciones.

Estas tablas se han realizado para los problemas MAXCUT100 y MTTP200 con los criterios AvgFitεR, AvgFitrR, AvgFitsR, AvgFitR, PopHεR y PopHR, y con los valores de $\varepsilon = 0.05, 0.15, 0.25y0.3$.

ε	N Medio de Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	206168.62 ± 156521.00	550.26 ± 296.12	12.42 %	60 %
0.15	245210.50 ± 128066.10	574.63 ± 301.08	9.95 %	80 %
0.25	340849.00 ± 199759.20	854.50 ± 500.95	9.96 %	60 %
0.3	296739.00 ± 153597.30	690.11 ± 357.78	9.96 %	90 %

Tabla C.1: MAXCUT100 y AvgFitεR (10 ejecuciones)

ε	N Medio de Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	290082.40 ± 121767.60	674.60 ± 280.36	7.17 %	50 %
0.15	202905.00 ± 58656.67	534.50 ± 154.89	9.82 %	40 %
0.25	342052.00 ± 226485.00	787.11 ± 521.07	9.96 %	90 %
0.3	367649.20 ± 236372.90	878.00 ± 589.25	9.97 %	60 %

Tabla C.2: MAXCUT100 y AvgFitrR (10 ejecuciones)

ε	N Medio de Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	192479.00 ± 0.00	439.00 ± 0.00	9.67 %	10 %
0.15	274817.70 ± 176119.10	676.00 ± 430.97	9.96 %	30 %
0.25	327856.60 ± 245987.50	749.00 ± 575.64	9.93 %	50 %
0.3	319653.30 ± 122425.60	742.57 ± 303.99	9.96 %	70 %

Tabla C.3: MAXCUT100 y AvgFitsR (10 ejecuciones)

ε	N Medio de Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	225427.80 ± 97369.68	527.50 ± 227.63	10.86 %	60 %
0.15	255837.00 ± 74170.19	591.50 ± 171.27	9.96 %	40 %
0.25	370332.50 ± 185560.10	858.83 ± 429.79	9.97 %	60 %
0.3	298543.50 ± 223720.80	589.50 ± 673.87	10.09 %	20 %

Tabla C.4: MAXCUT100 y AvgFitR (10 ejecuciones)

ε	N Medio de Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	878790.50 ± 99223.80	2533.50 ± 289.21	0.68 %	40 %
0.15	859642.80 ± 127588.20	2525.38 ± 412.88	0.70 %	80 %
0.25	865036.20 ± 75086.12	2502.50 ± 216.64	1.85 %	100 %
0.3	788365.00 ± 108320.00	2300.20 ± 324.54	2.44 %	100 %

Tabla C.5: MTTP200 y AvgFitεR (10 ejecuciones)

ε	N Medio de Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	535003.80 ± 70242.88	1522.71 ± 200.33	4.49 %	100 %
0.15	651383.40 ± 60557.40	1933.80 ± 181.04	78.80 %	100 %
0.25	665458.50 ± 81679.47	2598.50 ± 295.69	51.69 %	100 %
0.3	803803.50 ± 86894.49	2405.60 ± 355.41	38.07 %	100 %

Tabla C.6: MTTP200 y AvgFitrR (10 ejecuciones)

ε	N Medio de Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	801999.00 ± 143311.70	2317.00 ± 414.18	0.78 %	70 %
0.15	484447.10 ± 133395.90	1378.10 ± 380.98	4.95 %	100 %
0.25	502291.60 ± 83303.87	1439.10 ± 233.03	4.78 %	100 %
0.3	493568.30 ± 90340.45	1859.31 ± 341.72	4.86 %	100 %

Tabla C.7: MTTP200 y AvgFitsR (10 ejecuciones)

ε	N Medio de Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	495421.10 ± 49811.32	1470.03 ± 153.28	11.20 %	100 %
0.15	501115.30 ± 66449.47	1485.07 ± 197.09	11.73 %	100 %
0.25	489980.90 ± 80728.09	1395.60 ± 236.07	12.68 %	100 %
0.3	484366.90 ± 73697.25	1377.27 ± 217.25	14.20 %	100 %

Tabla C.8: MTTP200 y AvgFitR (10 ejecuciones)

ε	N Medio de Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	358593.30 ± 163333.20	1053.75 ± 479.20	40.58 %	80 %
0.15	303475.80 ± 153390.10	712.00 ± 360.98	43.76 %	50 %
0.25	488417.00 ± 238926.00	1148.00 ± 562.34	53.64 %	20 %
0.3	320097.30 ± 403069.00	754.75 ± 953.44	62.79 %	40 %

Tabla C.9: MAXCUT100 y PopHεR (10 ejecuciones)

ε	N Medio de Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	295822.40 ± 136104.30	892.57 ± 411.03	91.76 %	70 %
0.15	446011.30 ± 299391.80	1234.50 ± 829.65	93.27 %	40 %
0.25	277825.20 ± 115985.50	654.83 ± 273.26	90.46 %	60 %
0.3	249020.00 ± 118732.20	688.20 ± 330.89	83.21 %	50 %

Tabla C.10: MAXCUT100 y PopHR (10 ejecuciones)

ε	N Medio de Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	507264.00 ± 47338.01	1549.00 ± 206.61	9.31 %	100 %
0.15	514081.00 ± 63130.58	1509.80 ± 187.41	13.54 %	100 %
0.25	518933.10 ± 36813.19	1523.70 ± 108.71	17.11 %	100 %
0.3	487975.90 ± 50360.40	1437.40 ± 149.18	17.62 %	100 %

Tabla C.11: MTTP200 y PopHεR (10 ejecuciones)

ε	N Medio de Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	542792.60 ± 75407.72	1583.80 ± 211.16	82.83 %	100 %
0.15	549810.10 ± 54720.61	1601.50 ± 169.39	69.55 %	100 %
0.25	636586.50 ± 83649.76	1857.80 ± 251.01	53.10 %	100 %
0.3	624035.20 ± 50649.90	1806.80 ± 150.00	48.97 %	100 %

Tabla C.12: MTTP200 y PopHR (10 ejecuciones)

Apéndice D

Resultados Obtenidos con el Criterio Auto-Adaptativo AvgFitR

A continuación se presentan las tablas correspondientes a las ejecuciones de todos los problemas estudiados utilizando el criterio AvgFitR; criterio basado en el fitness descrito en la Sección 7.1, que comienza la ejecución del algoritmo utilizando la rejilla más rectangular posible ($2 \times \frac{|Poblacion|}{2}$).

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	17520.96 ± 8298.10	7.52 ± 3.51	87.44 %	76.67 %
0.15	13825.48 ± 10339.61	6.92 ± 4.20	84.48 %	83.33 %
0.25	20427.72 ± 11490.69	7.83 ± 4.42	91.44 %	60 %
0.3	15036.50 ± 8364.89	5.63 ± 3.38	86.46 %	80 %

Tabla D.1: COUNTSAT y AvgFitR (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	156896.90 ± 27638.81	832.53 ± 146.91	17.85 %	100 %
0.15	159530.20 ± 26402.04	714.87 ± 118.63	17.55 %	100 %
0.25	166440.70 ± 20681.10	756.93 ± 92.97	17.14 %	100 %
0.3	165197.60 ± 26773.96	955.63 ± 154.35	17.75 %	100 %

Tabla D.2: ECC y AvgFitR (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	663624.30 ± 151024.70	1228.93 ± 274.60	36.63 %	90 %
0.15	681114.20 ± 159513.40	1262.71 ± 288.20	36.85 %	80 %
0.25	700722.40 ± 130938.70	1292.84 ± 242.53	43.93 %	83.33 %
0.3	597726.60 ± 150723	2035.78 ± 630.97	41.34 %	90 %

Tabla D.3: FMS y AvgFitR (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	349355.90 ± 215948.10	804.07 ± 513.45	99.28 %	46.67 %
0.15	300642.10 ± 170415.30	698.13 ± 396.33	99.16 %	50 %
0.25	266664.00 ± 79912.97	601.64 ± 231.00	99.14 %	36.67 %
0.3	298543.50 ± 223720.80	589.50 ± 673.87	98.48 %	20 %

Tabla D.4: MAXCUT100 y AvgFitR (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	15036.50 ± 10625.00	7.23 ± 5.14	100 %	100 %
0.15	18164.30 ± 8290.70	7.87 ± 3.64	100 %	100 %
0.25	19130.58 ± 8355.95	8.13 ± 3.60	100 %	100 %
0.3	16787.53 ± 7467.75	7.27 ± 3.18	100 %	100 %

Tabla D.5: MAXCUT20.01 y AvgFitR (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	28216.03 ± 17216.28	15.37 ± 9.48	76.98 %	100 %
0.15	39056.40 ± 17900.03	17.03 ± 7.97	68.62 %	100 %
0.25	42398.07 ± 17800.11	18.17 ± 7.67	66.98 %	100 %
0.3	34404.80 ± 16377.78	14.93 ± 7.30	77.55 %	100 %

Tabla D.6: MAXCUT20.09 y AvgFitR (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	172351.90 ± 108519.20	796.92 ± 512.05	19.73 %	86.67 %
0.15	170362.30 ± 85763.52	713.65 ± 359.51	18.97 %	86.67 %
0.25	158808.80 ± 24533.08	676.24 ± 135.28	17.63 %	96.97 %
0.3	147032.30 ± 17984.91	512.38 ± 60.12	19.04 %	80 %

Tabla D.7: MMDP y AvgFitR (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	10090.83 ± 3444.84	8.80 ± 2.96	91.17 %	100 %
0.15	7684.83 ± 1961.90	10.10 ± 2.51	100 %	100 %
0.25	10531.93 ± 2745.96	8.27 ± 2.29	100 %	100 %
0.3	8005.63 ± 2488.66	10.17 ± 2.93	100 %	100 %

Tabla D.8: MTTP20 y AvgFitR (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	232726.00 ± 35977.90	438.63 ± 67.89	14.04 %	100 %
0.15	253778.50 ± 47212.03	418.13 ± 77.84	14.61 %	100 %
0.25	246493.70 ± 42176.01	408.17 ± 70.86	15.47 %	100 %
0.3	199790.60 ± 31559.79	436.17 ± 68.85	19.42 %	100 %

Tabla D.9: MTTP100 y AvgFitR (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	495421.10 ± 49811.32	1470.03 ± 153.28	11.12 %	100 %
0.15	503695.10 ± 56051.37	1491.57 ± 169.39	11.97 %	100 %
0.25	491504.70 ± 66676.04	1446.77 ± 197.83	13.57 %	100 %
0.3	484366.90 ± 73697.25	1377.27 ± 217.25	14.20 %	100 %

Tabla D.10: MTTP200 y AvgFitR (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	53385.47 ± 3578.50	184.30 ± 12.78	52.45 %	100 %
0.15	54895.90 ± 6542.95	178.47 ± 21.61	51.01 %	100 %
0.25	54695.40 ± 4413.64	179.70 ± 14.52	51.19 %	100 %
0.3	55644.43 ± 4967.00	184.30 ± 16.19	50.32 %	100 %

Tabla D.11: P-PEAKS y AvgFitR (30 ejecuciones)

Apéndice E

Resultados Obtenidos con el Criterio Auto-Adaptativo AvgFitC

En este apéndice se encuentran las tablas correspondientes a las ejecuciones de todos los problemas estudiados utilizando el criterio AvgFitC; criterio basado en el fitness descrito en la Sección 7.1, que comienza la ejecución del algoritmo utilizando la rejilla de ratio más intermedio.

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	16124.93 ± 11497.08	7.14 ± 5.08	79.63 %	46.67 %
0.15	16894.47 ± 14880.96	6.87 ± 6.15	75.76 %	50 %
0.25	19859.05 ± 8308.19	7.95 ± 3.42	87.01 %	63.33 %
0.3	16039.00 ± 9431.86	6.43 ± 3.65	87.29 %	46.67 %

Tabla E.1: COUNTSAT y el criterio AvgFitC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	160786.60 ± 21340.22	928.60 ± 123.61	12.44 %	100 %
0.15	153983.00 ± 24707.86	694.23 ± 111.28	12.99 %	100 %
0.25	169822.50 ± 34630.04	766.13 ± 155.24	12.18 %	100 %
0.3	163099.10 ± 25104.13	734.10 ± 112.56	12.75 %	100 %

Tabla E.2: ECC y el criterio AvgFitC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	676631.80 ± 157686.80	1258.00 ± 291.45	39.42 %	73.33 %
0.15	689146.10 ± 159657.80	1281.33 ± 305.34	40.35 %	70 %
0.25	687530.20 ± 117807.30	1297.38 ± 230.96	45.38 %	80 %
0.3	716132.70 ± 168205.10	1331.13 ± 307.55	46.07 %	76.67 %

Tabla E.3: FMS y el criterio AvgFitC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	15624.63 ± 9045.74	7.50 ± 4.27	91.39 %	100 %
0.15	15303.83 ± 7800.66	6.60 ± 3.46	98.54 %	100 %
0.25	17161.80 ± 11747.55	7.40 ± 5.16	91.60 %	100 %
0.3	20369.80 ± 10752.95	8.80 ± 4.73	84.44 %	100 %

Tabla E.4: MAXCUT20.01 y el criterio AvgFitC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	30849.27 ± 10182.74	16.07 ± 5.35	62.63 %	100 %
0.15	36490.00 ± 15600.13	15.80 ± 6.74	57.77 %	100 %
0.25	26251.13 ± 12123.08	11.30 ± 5.28	72.07 %	100 %
0.3	32052.27 ± 17533.96	13.87 ± 7.67	68.26 %	100 %

Tabla E.5: MAXCUT20.09 y el criterio AvgFitC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	226263.30 ± 116486.00	633.92 ± 323.03	11.63 %	40 %
0.15	346555.50 ± 233780.40	810.00 ± 546.80	21.93 %	43.33 %
0.25	320548.40 ± 166590.90	789.69 ± 413.11	18.88 %	53.33 %
0.3	304037.20 ± 192887.70	697.20 ± 443.65	20.43 %	50 %

Tabla E.6: MAXCUT100 y el criterio AvgFitC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	185261.00 ± 95603.82	770.81 ± 398.51	14.60 %	70 %
0.15	182771.50 ± 81137.22	1211.50 ± 551.00	14.05 %	80 %
0.25	167334.80 ± 34522.71	580.67 ± 120.15	13.00 %	90 %
0.3	190822.70 ± 122350.40	667.70 ± 416.97	17.40 %	76.67 %

Tabla E.7: MMDP y el criterio AvgFitC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	8807.63 ± 2803.01	7.60 ± 2.46	94.01 %	100 %
0.15	7577.90 ± 2673.14	9.40 ± 2.99	91.32 %	100 %
0.25	10291.33 ± 3117.42	7.97 ± 2.43	100 %	100 %
0.3	7952.17 ± 2376.05	9.87 ± 2.99	100 %	100 %

Tabla E.8: MTTP20 y el criterio AvgFitC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	176011.30 ± 25616.84	411.07 ± 60.33	12.57 %	100 %
0.15	245985.80 ± 40514.99	406.63 ± 67.10	11.55 %	100 %
0.25	247469.50 ± 53530.09	412.10 ± 88.63	12.77 %	100 %
0.3	244141.20 ± 46334.47	420.13 ± 80.70	14.42 %	100 %

Tabla E.9: MTTP100 y el criterio AvgFitC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	492186.40 ± 75765.23	1525.50 ± 235.46	9.43 %	100 %
0.15	508146.20 ± 51246.20	1447.73 ± 148.05	10.49 %	100 %
0.25	506020.90 ± 63341.05	1453.60 ± 201.54	12.67 %	100 %
0.3	493977.50 ± 61737.00	1456.30 ± 228.40	12.37 %	100 %

Tabla E.10: MTTP200 y el criterio AvgFitC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	55804.83 ± 5088.76	184.53 ± 17.09	35.84 %	100 %
0.15	54254.30 ± 5036.25	176.17 ± 16.59	36.86 %	100 %
0.25	54535.00 ± 4807.39	177.53 ± 16.28	36.67 %	100 %
0.3	55283.53 ± 4624.84	191.60 ± 15.99	36.18 %	100 %

Tabla E.11: P-PEAKS y el criterio AvgFitC (30 ejecuciones)

Apéndice F

Resultados Obtenidos con el Criterio Auto-Adaptativo PopHR

A continuación se presentan las tablas correspondientes a las ejecuciones de todos los problemas estudiados utilizando el criterio PopHR; criterio basado en la entropía de la población descrito en la Sección 7.2, que comienza la ejecución del algoritmo utilizando la rejilla más rectangular posible ($2 \times \frac{|Poblacion|}{2}$).

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	21797.36 ± 12992.28	10.04 ± 6.04	99.45 %	83.33 %
0.15	16018.95 ± 13990.90	6.20 ± 5.67	98.62 %	66.67 %
0.25	20171.04 ± 10348.20	7.91 ± 4.18	99.23 %	76.67 %
0.3	20892.48 ± 12692.21	8.28 ± 4.93	97.64 %	96.67 %

Tabla F.1: COUNTSAT y el criterio PopHR (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	148462.60 ± 25502.90	719.13 ± 137.39	83.60 %	100 %
0.15	152713.20 ± 25835.86	845.20 ± 143.62	90.71 %	100 %
0.25	148369.00 ± 27608.31	657.97 ± 122.71	82.15 %	100 %
0.3	140095.00 ± 20777.81	746.70 ± 108.72	76.61 %	100 %

Tabla F.2: ECC y el criterio PopHR (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	766189.70 ± 140504.40	1404.85 ± 250.64	92.90 %	66.67 %
0.15	751641.80 ± 119874.70	1372.05 ± 218.39	90.19 %	63.33 %
0.25	566354.20 ± 128538.50	1790.25 ± 599.37	90.71 %	93.33 %
0.3	725386.90 ± 152656.20	1318.90 ± 279.33	89.59 %	63.33 %

Tabla F.3: FMS y el criterio PopHR (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	12590.40 ± 7166.81	6.17 ± 3.40	100 %	100 %
0.15	17482.60 ± 9918.29	7.37 ± 4.17	98.38 %	100 %
0.25	18645.50 ± 9244.77	7.90 ± 3.90	97.18 %	100 %
0.3	15557.80 ± 7738.08	6.83 ± 3.45	99.50 %	100 %

Tabla F.4: MAXCUT20.01 y el criterio PopHR (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	26705.60 ± 10439.07	13.20 ± 5.16	93.91 %	100 %
0.15	32199.30 ± 14050.26	15.70 ± 6.89	96.52 %	100 %
0.25	39350.47 ± 14368.14	16.80 ± 6.00	90.47 %	100 %
0.3	31958.70 ± 14111.30	13.53 ± 6.00	93.50 %	100 %

Tabla F.5: MAXCUT20.09 y el criterio PopHR (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	349194.80 ± 165809.11	1057.00 ± 503.33	92.00 %	53.33 %
0.15	419645.50 ± 232141.00	1162.94 ± 644.12	90.85 %	53.33 %
0.25	264069.30 ± 135993.40	626.12 ± 330.48	87.95 %	56.67 %
0.3	282627.60 ± 127293.10	783.67 ± 353.19	84.18 %	70 %

Tabla F.6: MAXCUT100 y el criterio PopHR (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	197719.70 ± 110560.90	742.59 ± 416.71	87.62 %	96.67 %
0.15	238626.10 ± 98768.04	998.40 ± 412.93	80.26 %	83.33 %
0.25	270043.90 ± 49153.66	1131.82 ± 210.67	68.08 %	93.33 %
0.3	155111.70 ± 14915.18	549.85 ± 52.41	18.05 %	90 %

Tabla F.7: MMDP y el criterio PopHR (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	9181.90 ± 2399.89	8.37 ± 2.25	100 %	100 %
0.15	7390.77 ± 2115.83	9.10 ± 2.45	100 %	100 %
0.25	10919.57 ± 3428.06	8.67 ± 2.78	100 %	100 %
0.3	8313.07 ± 1780.71	10.17 ± 2.17	100 %	100 %

Tabla F.8: MTTP20 y el criterio PopHR (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	261557.90 ± 50632.33	483.70 ± 92.07	78.04 %	100 %
0.15	312484.90 ± 49679.03	502.77 ± 80.11	61.79 %	100 %
0.25	329246.70 ± 63992.46	529.90 ± 104.86	48.47 %	100 %
0.3	323459.00 ± 56914.68	539.77 ± 95.11	44.15 %	100 %

Tabla F.9: MTTP100 y el criterio PopHR (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	520202.90 ± 74079.50	1510.73 ± 214.10	82.25 %	100 %
0.15	573161.70 ± 62431.99	1667.20 ± 189.21	66.56 %	100 %
0.25	640944.00 ± 84587.34	1862.00 ± 248.05	54.84 %	100 %
0.3	662370.80 ± 76047.54	1916.63 ± 221.78	49.70 %	100 %

Tabla F.10: MTTP200 y el criterio PopHR (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	49803.20 ± 4927.46	171.97 ± 17.28	87.78 %	100 %
0.15	48292.77 ± 4473.64	156.20 ± 14.60	76.45 %	100 %
0.25	49174.97 ± 3835.41	158.57 ± 12.57	71.83 %	100 %
0.3	48773.97 ± 4393.35	157.70 ± 14.62	73.56 %	100 %

Tabla F.11: P-PEAKS y el criterio PopHR (30 ejecuciones)

Apéndice G

Resultados Obtenidos con el Criterio Auto-Adaptativo PopHC

En este apéndice se encuentran las tablas correspondientes a las ejecuciones de todos los problemas estudiados utilizando el criterio PopHC; criterio basado en la entropía de la población descrito en la Sección 7.2, que comienza la ejecución del algoritmo utilizando la rejilla de ratio más intermedio.

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	23237.90 ± 15529.87	9.19 ± 6.23	98.35 %	70 %
0.15	17178.68 ± 10325.49	6.79 ± 4.60	99.67 %	63.33 %
0.25	19880.16 ± 10875.01	7.74 ± 4.21	98.71 %	63.33 %
0.3	19648.00 ± 13000.43	7.83 ± 5.27	98.29 %	60 %

Tabla G.1: COUNTSAT y el criterio PopHC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	153354.80 ± 21890.01	955.80 ± 138.65	92.26 %	100 %
0.15	153702.30 ± 26817.56	692.37 ± 121.18	85.88 %	100 %
0.25	156602.90 ± 23002.15	710.10 ± 102.01	83.60 %	100 %
0.3	148409.10 ± 17577.99	668.60 ± 79.63	79.43 %	100 %

Tabla G.2: ECC y el criterio PopHC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	680328.90 ± 171317.40	1259.08 ± 315.08	91.89 %	80 %
0.15	688293.20 ± 155855.70	1275.41 ± 276.73	91.54 %	90 %
0.25	673679.00 ± 177126.70	1243.32 ± 322.29	92.10 %	93.33 %
0.3	660591.80 ± 159243.10	1247.82 ± 317.70	92.40 %	73.33 %

Tabla G.3: FMS y el criterio PopHC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	13953.80 ± 8139.68	6.83 ± 4.07	100 %	100 %
0.15	15544.43 ± 10076.41	6.53 ± 4.31	100 %	100 %
0.25	18311.33 ± 10042.59	7.77 ± 4.32	100 %	100 %
0.3	14582.03 ± 9592.54	6.13 ± 4.11	100 %	100 %

Tabla G.4: MAXCUT20.01 y el criterio PopHC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	24246.13 ± 9908.30	12.00 ± 5.00	100 %	100 %
0.15	32600.30 ± 14393.36	13.77 ± 6.10	100 %	100 %
0.25	32480.00 ± 11790.24	14.07 ± 5.14	100 %	100 %
0.3	34164.20 ± 18626.80	14.50 ± 7.98	100 %	100 %

Tabla G.5: MAXCUT20.09 y el criterio PopHC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	336933.40 ± 291081.10	931.71 ± 809.15	91.14 %	56.67 %
0.15	358153.70 ± 234045.70	858.69 ± 563.14	90.55 %	43.33 %
0.25	299078.20 ± 248670.10	732.33 ± 619.62	89.31 %	60 %
0.3	307245.20 ± 214742.50	741.30 ± 541.13	84.49 %	33.33 %

Tabla G.6: MAXCUT100 y el criterio PopHC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	234111.40 ± 121700.20	976.00 ± 507.28	87.63 %	93.33 %
0.15	216566.70 ± 40909.01	1440.76 ± 283.09	80.83 %	96.67 %
0.25	297021.20 ± 90834.64	1036.44 ± 316.98	73.48 %	90 %
0.3	300704.40 ± 97529.30	1046.22 ± 340.07	68.28 %	90 %

Tabla G.7: MMDP y el criterio PopHC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	8460.10 ± 2683.49	7.47 ± 2.53	100 %	100 %
0.15	7257.10 ± 1987.92	9.37 ± 2.40	100 %	100 %
0.25	9088.33 ± 2133.19	7.03 ± 1.75	100 %	100 %
0.3	7698.20 ± 2179.64	9.80 ± 2.99	100 %	100 %

Tabla G.8: MTTP20 y el criterio PopHC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	225628.30 ± 52389.24	443.60 ± 101.92	76.41 %	100 %
0.15	308662.10 ± 43327.70	514.90 ± 72.55	64.67 %	100 %
0.25	311288.60 ± 53339.15	523.33 ± 89.56	51.18 %	100 %
0.3	293985.50 ± 38272.03	576.83 ± 75.94	42.82 %	100 %

Tabla G.9: MTTP100 y el criterio PopHC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	535721.60 ± 62544.52	1769.27 ± 206.44	82.63 %	100 %
0.15	595871.60 ± 60997.79	1739.17 ± 177.19	68.20 %	100 %
0.25	654765.20 ± 70138.14	1912.07 ± 213.88	55.45 %	100 %
0.3	651343.30 ± 85408.89	1905.10 ± 252.38	50.30 %	100 %

Tabla G.10: MTTP200 y el criterio PopHC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	48266.03 ± 4155.97	173.00 ± 15.10	94.48 %	100 %
0.15	49642.80 ± 4708.80	158.00 ± 14.95	94.03 %	100 %
0.25	49054.67 ± 4242.91	158.27 ± 13.87	82.93 %	100 %
0.3	48653.67 ± 4231.14	156.30 ± 13.58	81.97 %	100 %

Tabla G.11: P-PEAKS y el criterio PopHC (30 ejecuciones)

Apéndice H

Resultados Obtenidos con el Criterio Auto-Adaptativo FitHC

Este apéndice contiene las tablas correspondientes a las ejecuciones de todos los problemas estudiados utilizando el criterio FitHC; criterio combinado basado en la entropía de la población y en el valor de adecuación de los individuos, que aparece descrito en la Sección 8.5. Con este criterio se comienza la ejecución del algoritmo utilizando la rejilla de ratio más intermedio.

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	21676.59 ± 11937.30	8.35 ± 4.50	50.01 %	56.67 %
0.15	23903.06 ± 17337.99	9.17 ± 6.80	54.89 %	60 %
0.25	21602.88 ± 13601.00	8.25 ± 5.37	64.81 %	26.67 %
0.3	19931.06 ± 11866.62	7.88 ± 4.48	66.03 %	56.67 %

Tabla H.1: COUNTSAT y el criterio FitHC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	177281.10 ± 27036.08	822.30 ± 125.32	11.28 %	100 %
0.15	167590.30 ± 21187.27	790.30 ± 100.92	11.93 %	100 %
0.25	167857.60 ± 23239.30	790.93 ± 107.72	12.08 %	100 %
0.3	170504.20 ± 31351.91	802.27 ± 151.07	11.89 %	100 %

Tabla H.2: ECC y el criterio FitHC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	765463.40 ± 175703.20	1429.50 ± 336.84	18.32 %	60 %
0.15	726591.00 ± 163574.60	1357.45 ± 302.83	18.99 %	66.67 %
0.25	589129.70 ± 177741.90	1943.08 ± 999.69	21.49 %	86.67 %
0.3	684873.60 ± 133639.90	1307.54 ± 259.49	23.56 %	80 %

Tabla H.3: FMS y el criterio FitHC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	16132.57 ± 7877.47	7.80 ± 3.93	66.20 %	100 %
0.15	19781.67 ± 8868.60	8.33 ± 3.75	64.10 %	100 %
0.25	21078.23 ± 10927.52	8.87 ± 4.61	66.42 %	100 %
0.3	15985.53 ± 9533.54	6.77 ± 4.07	66.81 %	100 %

Tabla H.4: MAXCUT20.01 y el criterio FitHC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	32720.60 ± 10460.83	16.63 ± 5.31	50.98 %	100 %
0.15	36904.37 ± 17403.68	15.70 ± 7.47	46.61 %	100 %
0.25	37131.60 ± 18015.82	16.00 ± 7.76	46.32 %	100 %
0.3	41208.43 ± 16696.41	17.40 ± 7.03	45.33 %	100 %

Tabla H.5: MAXCUT20.09 y el criterio FitHC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	319489.10 ± 217282.80	889.33 ± 608.92	10.43 %	50 %
0.15	252906.60 ± 108984.50	831.46 ± 351.13	9.62 %	43.33 %
0.25	315151.60 ± 277533.80	780.75 ± 683.58	13.43 %	40 %
0.3	341625.90 ± 181722.50	1184.69 ± 835.07	12.81 %	53.33 %

Tabla H.6: MAXCUT100 y el criterio FitHC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	154309.70 ± 38308.96	558.33 ± 137.17	13.25 %	90 %
0.15	152321.70 ± 13247.49	547.54 ± 47.13	13.13 %	93.33 %
0.25	182955.30 ± 134052.20	737.33 ± 492.15	14.39 %	80 %
0.3	167516.80 ± 103601.10	1098.36 ± 650.00	13.64 %	93.33 %

Tabla H.7: MMDP y el criterio FitHC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	8633.87 ± 3193.68	7.47 ± 2.80	92.66 %	100 %
0.15	7858.60 ± 1831.87	9.97 ± 2.16	96.71 %	100 %
0.25	10451.73 ± 2827.54	7.97 ± 2.13	100 %	100 %
0.3	7604.63 ± 1779.31	9.63 ± 2.30	100 %	100 %

Tabla H.8: MTTP20 y el criterio FitHC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	255061.70 ± 49357.66	416.97 ± 81.97	7.95 %	100 %
0.15	243325.80 ± 54418.99	400.50 ± 89.36	8.83 %	100 %
0.25	249140.30 ± 41085.84	409.60 ± 68.17	8.56 %	100 %
0.3	205818.90 ± 37150.95	401.93 ± 72.91	10.55 %	100 %

Tabla H.9: MTTP100 y el criterio FitHC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	501596.50 ± 68683.89	1951.57 ± 263.61	6.40 %	100 %
0.15	497132.10 ± 65651.17	1784.80 ± 663.25	5.39 %	100 %
0.25	497813.80 ± 77552.76	1455.70 ± 226.66	5.79 %	100 %
0.3	476133.00 ± 61471.64	1397.13 ± 184.96	6.22 %	100 %

Tabla H.10: MTTP200 y el criterio FitHC (30 ejecuciones)

ε	N Medio Evaluaciones	Tiempo Medio(s)	Porcent. Cambios de Rejilla	Soluciones
0.05	51594.33 ± 6381.63	214.30 ± 26.37	38.22 %	100 %
0.15	54802.33 ± 6281.79	180.43 ± 20.85	36.50 %	100 %
0.25	54347.87 ± 6202.20	189.63 ± 20.54	36.29 %	100 %
0.3	54441.43 ± 5313.28	180.23 ± 17.61	36.00 %	100 %

Tabla H.11: P-PEAKS y el criterio FitHC (30 ejecuciones)

Bibliografía

[Alba99] E. Alba. *Análisis y Diseño de Algoritmos Genéticos Paralelos Distribuidos*. Tesis Doctoral, Universidad de Málaga, Marzo 1999.

[AT00] E. Alba, and J.M. Troya. *Cellular Evolutionary Algorithms: Evaluating the Influence of Ratio*, Proceedings of the Parallel Problem Solving from Nature VI, Schoenauer M. et al. (eds.)Springer-Verlag, pp. 29-38, 2000.

[AT99] E. Alba, and J.M. Troya. *A Survey of Parallel Distributed Genetic Algorithms. Complexity*, John Wiley & Sons 4(4):31-52, 1999.

[Balu93] S. Baluja. *Structure an Performance of Fine-Grain Parallelism in Genetic Search*, In: Proceedings of the 5th International Conference on Genetic Algorithms, Forrest S. (ed.). Morgan Kaufmann. pp. 151-162, 1993.

[BCCG00] A. Bertoni, P. Campadelli, M. Carpentieri, and G. Grossi. *A Genetic Model: Analysis and Aplication to MAXSAT*. Evolutionary Computation, Volume 8, Number 3, pp. 291-309, 2001.

[BFM97] T. Bäck, D. Fogel, and Z. Michalewicz. *Handbook of Evolutionary Computation*, Oxford University Press. 1997.

[BH90] T. Bäck, and F. Hoffmeister. *Adaptative Search by Evolutionary Algorithms*. Diciembre de 1990.

[BHK94] T. Bäck, J. Heitkötter, and S. Khuri. *An Evolutionary Approach to Combinatorial Optimization Problems*, In: Proceedings of the 22nd Annual ACM Computer Science Conference, ed. D. Cizmar, pp. 66-73. ACM Press, New York, 1994.

[BRYS99] A. E. BRYSON. *Dynamic Optimization*. Addison-Wesley. 1999.

[Bäck01] T. Bäck. *Introduction to the Special Isue*, Introducción al número 9(2) de la revista Evolutionary Computation.

[CD96] J. M. Crawford, and L. D. Auton. *Experimental Results on the Crossover Point in Random 3SAT*. Supported by th Air Force Office of Scientific Research an by ARPA/Rome Labs, August 1996.

[CFW98] H. Chen, N. S. Flann, and D. W. Watson. *Parallel genetic simulated annealing: a massively parallel SIMD algorithm.* IEEE transactions parallel and distributed systems, vol. 9, number 2, pp. 805 811, February 1998.

[Cook71] S. Cook. *The Complexity of Theorem Proving Procedures*. In Proceedings of the Third Simposium of the ACM on the Theory of Computing, pages 151-158, ACM Press, New York,1971.

[CTTS98] M. Capcarrère, A. Tettamanzi, M. Tomassini, and M. Sipper. *Studying Parallel Evolutionary Algorithms: The Cellular Programming Case*. Proceedings of the Parallel Problem Solving from Nature V, 1998.

[CTTS99] M. Capcarrère, A. Tettamanzi, M. Tomassini, and M. Sipper. *A Statistical Study of a Class of Cellular Evolutionary Algorithms*. In Voigt H. M., Ebeling W., Rechenberg I., Schwefel H. P. (eds.), Proceedings of the Fourth Parallel Problem Solving from Nature, Springer-Verlag, pp. 236-244, 1998.

[Darw59] C. Darwin. *On the Origin of Species by Means of Natural Selection*. John Murray, Londres. 1859.

[FJW00] S. Froste, T. Jansen, and I. Wegener. *A Natural and Simple Function Wich is Hard For All Evolutionary Algorithms*, Collaborative Research Center 531, Çomputational Intelligence", University of Dortmund.

[GDH92] D. E. Goldberg, K. Deb, and J. Horn. *Massively Multimodality, Deception and Genetic Algorithms*. In: Proceedings of the PPSN II, Männer R., Manderick B. (eds). North-Holland, pp. 37-46, 1992.

[GJ79] M. R. Garey, and D. S. Johnson. *Computers and Interactibility. A Guide to the Theory of NP-Completeness*. W. H. Frrman Company, 1979.

[Gold89] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, Reading, MA, 1989.

[GR87] D. E. Goldberg, and J. T. Richardson. *Genetic Algorithms with Sharing for Multimodal Function Optimization*. Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms an their Applications. pp. 41-49, 1987.

[Hay97] B. Hayes. *Can't get no satisfaction*. Informe técnico. http://www.amsci.org/amsci/issues/Comsci97/compsci9703.html, Marzo 1997.

[Holl75] J. H. Holland. *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[Jone95] T. Jones. *Evolutionary Algorithms, Fitness Landscapes and Search*. Unpublished doctoral dissertation, University of New Mexico, Albuquerque, New Mexico. 1995

[Jong75] K. A. De Jong. *An Analysis of the Behaviour of a Class of Genetic Adaptative Systems*, Tesis Doctoral, University of Michigan. 1975.

[JPS97] K. A. Jong, M. A. Potter, and W. M. Spears. *Using Problem Generators to Explore the Effects of Epistasis*. In: Proceedings of the 7th International Conference of Genetic

Algorithms, T. Bäck (ed.). Morgan Kaufmann, pp. 338-345, 1997.

[JS96] K. De Jong, and J. Sarma. *An Analysis of the Effects of Neighborhood Size and Shape on Local Selection Algorithms*. In Voigt H. M., Ebeling W. Rechengerg I., Schwefel H. P. (eds.) Proceedings of the Fourth Parallel Problem Solving from Nature. Springer-Verlag, pp. 236-244, 1996.

[JS97] K. De Jong, and J. Sarma. *An Analysisof Local Selection Algorithms in a Spatially Structured Evolutionary Algorithm*. In: Proceedings of the 7th International Conference on Genetic Algorithms, Bäck T. (ed.). Morgan Kaufmann, pp. 181-186, 1997.

[Karp72] R. M. Karp. *Reducibility among Combinatorial Problems*. In R.E. Miller and J.W.Thatcher, editors,*Complexity of Computer Computation*, pp. 85 103. Plenum, New York, 1972.

[KBH94] S. Khuri, T. Bäck, and J.Heitkötter. *An Evolutionary Approach to Combinatorial Optimization Problems.* Proceedings of the 22^{nd} Annual ACM Computer Science Conference, pp. 66 73, ACM Press, NY, 1994.

[Kiho95] Kihong Park. *A Comparative Study of Genetic Search*. Proceedings of the sixth International Conference of Genetic Algorithm. Morgan Kaufmann Publishers, pp.512-519, Julio 1995.

[Kenn98] J. Kennedy, and W. M. Spears. *Matching Algorithms to Problems: an Experimental Test of the Particle Swarm and some Genetic Algorithms on the Multimodal Problem Generator*. In: Proc. 1998 IEEE World Congress on Computational Intelligence, pp. 74–77, Anchorage: Alaska, May 1998.

[Koza92] J. R. Koza. *Genetic Programming. On the Programming of Computers by Means of Natural Selection*, The MIT Press, 1992.

[LA87] P. van Laarhoven, and E. H. L. Aarts. *Simulated Annealing: Theory and Applications.* Kluwer Academic Publishers. 1987.

[Lamp99] L. Lamport. *LaTeX: A Document Preparation System. User's Guide and Reference Manual*. Addison-Wesley, 10th ed., 1999.

[Lang89] C. G. Langton. *Artificial Life*, Artificial Life 1 (pp. 1-47), Langton C.G.(ed.), Addison-Wesley, Santa Fe NM, 1989.

[Mend65] G. Mendel. *Versuche über Pflanzen-Hybriden*. Verhandlungen des Naturforschedes Vereines in Brünn 4. 1865.

[Mich92] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1992.

[MS91] B. Manderick, and P. Spiessens.*A Massively Parallel Genetic Algorithm*. In Belew R.K., Booker L.B. (eds.) Proceedings of the Fourth International Conference on Genetic Algorithms. Morgan Kaufmann, San Mateo, California, pp. 279-286, 1991.

[MSB91] H. Mühlenbein, M. Schomisch, and J. Born. *The Parallel Genetic Algorithm as Function Optimizer*, In: Proceedings of the 4th International Conference on Genetic Algorithms, Belew R.K., Booker L.B.(eds.). Morgan Kaufmann. pp 271-278, 1991.

[MR86] J. L. McLelland, and D. E. Rumelhart et al. *Parallel Distributed Processing*. MIT Press. 1986.

[Papa94] C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[Rech73] I. Rechenberg. *Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien der Biologischen Evolution*. Fromman-Holzboog Verlag, Sttutg, 1973.

[SC95] R. Schrag, and J. M. Crawford. *Implicates and Prime Implicates in Random 3SAT*. Informe técnico http://www.cirl.uoregon.edu/crawford/papers/papers.html, Abril 1997.

[Schi99] H. Schildt. *C ++ : Manual de referencia*. Osborne/McGraw-Hill. 1999.

[Smi97] B. M. Smith. *The Fase Transition in Constraint Satisfaction Problems: a closer Look at the Mushy Region*. Informe técnico http://www.scs.leeds.ac.uk/bms, 1997.

[Stin87] D. R. Stinson. *An Introduction to the Design and Analysis of Algorithms*. The Charles Babbage Research Center, Winnipeg, Manitoba, Canada, 2nd edition, 1987.

[Stro91] B. Strouptrup. *The C++ Programming Language (2nd Edition)*. Addison-Wesley, 1991.

[TGCF97] S. Tsutsui, A. Ghosh, D. Corne, and Y. Fujimoto. *A Real Coded Genetic Algorithm with an Explorer and Exploiter Populations*. In: Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA), Bäck T. (ed.). Morgan Kaufmann, pp. 238-245, 1997.

[Toma93] M. Tomassini. *The Parallel Genetic Cellular Automata: Aplication to Global Function*, In: Proceedings of the International Conference on Artificial Neural Nets and GAs, Albretch R.F., Reeves C.R., Steele N.C. (eds.). Springer-Verlag. pp 385-391, 1993.

[Whit93] D. Whitley. *Celular Genetic Algorithms*, In: Proceedings of the 5th International Conference on Genetic Algorithms, Forrest S. (ed.). Morgan Kaufmann. pp 658, 1993.

[WM97] D. H. Wolpert, and W. G. Macready. *No Free Lunch Theorems for Optimization*. IEEE Transactions on Evolutionary Computation 1, pp 67-82. 1997.

[Zade65] A. L. Zadeh. *Fuzzy Sets*. Inf. Control 8, 338-353. 1965.

[Zade94] A. L. Zadeh. *Fuzzy Logic, Neural Networks, and Soft Computing*. CACM 37(3): 77-84. 1994.