UNIVERSIDAD DE MURCIA

FACULTAD DE INFORMÁTICA

TESIS DOCTORAL

Modelado y Autooptimización de Metaheurísticas e Hiperheurísticas Parametrizadas Paralelas Aplicadas a Problemas de Optimización en Ciencias e Ingeniería

José Matías Cutillas Lozano

2014

UNIVERSIDAD DE MURCIA

DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS

TESIS DOCTORAL

Modelado y Autooptimización de Metaheurísticas e Hiperheurísticas Parametrizadas Paralelas Aplicadas a Problemas de Optimización en Ciencias e Ingeniería

José Matías Cutillas Lozano

2014

Director:
Domingo Giménez Cánovas

Resumen

En este trabajo se estudia la aplicación de esquemas parametrizados paralelos de metaheurísticas e hiperheurísticas a problemas de optimización en ciencias e ingeniería. Un objetivo a conseguir es la aplicación eficiente de estos métodos, por lo que es necesario el uso de modelos que permitan su autooptimización durante la ejecución a través de la selección adecuada de parámetros característicos del sistema computacional y del paradigma de paralelismo empleado.

La utilización de un esquema parametrizado de metaheurísticas permite aplicar fácilmente diferentes metaheurísticas a problemas de optimización, simplemente modificando algunos parámetros metaheurísticos. Además, puesto que muchos de estos problemas tienen una elevada carga computacional se hace indispensable la introducción de paralelismo en el esquema. Así, se consideran dos paradigmas que pueden ser complementarios: paralelismo local de memoria compartida y paralelismo global de paso de mensajes. El uso de algoritmos paralelos persigue un objetivo claro: la reducción del tiempo de ejecución, suponiendo un enfoque diferente para la resolución de los problemas de optimización.

Debido a que obtener una buena metaheurística para un problema de optimización concreto puede ser un proceso costoso, se aporta también como novedad el desarrollo de hiperheurísticas basadas en esquemas metaheurísticos parametrizados, entendidas como algoritmos de más alto nivel cuya finalidad es la selección automática de la mejor metaheurística para un problema o conjunto de problemas dados. Como la estructura de las hiperheurísticas se basa en el esquema parametrizado de metaheurísticas, se puede considerar en este caso la misma metodología de modelado y autooptimización, pero a un nivel superior de abstracción. La aplicación de las hiperheurísticas basadas en un esquema metaheurístico paralelo con un modelo teórico del tiempo de ejecución permite una selección automática de los parámetros de paralelismo óptimos, dando como resultado algoritmos eficientes tanto en calidad de los resultados como en rapidez en alcanzarlos.

Abstract

In this work the application of parallel parameterized schemes of metaheuristics and hyperheuristics to optimization problems in science and engineering is studied. One goal is the efficient application of these methods, so it is necessary to use models that allow method auto-tuning during the execution through the proper selection of the characteristic parameters of parallelism and of the computer system used.

The use of a parameterized metaheuristic scheme allows the easy application of different metaheuristics to optimization problems, simply by modifying some metaheuristic parameters. Furthermore, since many of these problems have a high computational cost, the introduction of parallelism in the scheme is indispensable. Thus, we consider two complementary paradigms: local shared-memory parallelism and global message-passing parallelism. The use of parallel algorithms pursues a clear objective: to reduce the execution time by assuming a different approach for solving the optimization problems.

Because getting a good metaheuristic for a specific optimization problem can be a costly process, the development of hyperheuristics based on parameterized metaheuristic schemes is an advance. A hyperheuristic can be understood as an algorithm of higher level whose purpose is the automatic selection of the best metaheuristic for a given problem or set of problems. As the hyperheuristic structure is based on the parameterized metaheuristic scheme, the same modeling and auto-tuning methodology can be considered in this case, but at a higher level of abstraction. Applying the hyperheuristics based on a parallel metaheuristic scheme with a theoretical model of the execution time allows automatic selection of optimal parallelism parameters, resulting in an efficient algorithm both in quality of the results and in speed of achieving them.

Índice general

1. Introducción **1**
- 1.1. Introducción . . . 1
  - 1.1.1. Metaheurísticas . . . 2
  - 1.1.2. Hiperheurísticas . . . 2
  - 1.1.3. Esquemas parametrizados de metaheurísticas . . . 3
  - 1.1.4. Metaheurísticas paralelas y autooptimización . . . 4
- 1.2. Objetivos . . . 5
- 1.3. Metodología . . . 7
- 1.4. Herramientas computacionales . . . 8
  - 1.4.1. El hardware . . . 8
  - 1.4.2. El software . . . 9
- 1.5. Estructura de la memoria . . . 9

2. Herramientas algorítmicas y problemas de optimización **13**
- 2.1. Herramientas algorítmicas . . . 13
  - 2.1.1. Metaheurísticas . . . 14
  - 2.1.2. Hiperheurísticas . . . 17
  - 2.1.3. Esquemas unificados parametrizados de metaheurísticas . . . 19
  - 2.1.4. Metaheurísticas paralelas y autooptimización . . . 20
- 2.2. Problemas de optimización . . . 22
  - 2.2.1. El problema de consumo de electricidad en la explotación de pozos de agua (PCEPA) . . . 22
  - 2.2.2. El problema de determinación de las constantes cinéticas de una reacción química (PCOCI) . . . 27
  - 2.2.3. El problema de selección de parámetros metaheurísticos óptimos (POPME) . . . 33
- 2.3. Conclusiones . . . 33

3. El esquema parametrizado de metaheurísticas: ampliación y aplicaciones **35**
- 3.1. El esquema parametrizado de metaheurísticas . . . 35
  - 3.1.1. Clasificación de las combinaciones de metaheurísticas . . . 38
- 3.2. Inclusión de la Búsqueda Tabú en el esquema parametrizado . . . 40

3.3. Aplicación del esquema metaheurístico al problema PCEPA 42
3.4. Aplicación del esquema metaheurístico al problema PCOCI 52
3.5. Conclusiones 64

4. Hiperheurísticas basadas en esquemas metaheurísticos parametrizados 67
4.1. Hiperheurísticas basadas en esquemas metaheurísticos parametrizados 68
4.2. Aplicación de hiperheurísticas a la optimización de metaheurísticas . 71
4.2.1. Aplicación de hiperheurísticas a PCEPA 74
4.2.2. Aplicación de hiperheurísticas a PCOCI 91
4.3. Conclusiones 97

5. Modelado y autooptimización de metaheurísticas e hiperheurísticas paralelas en memoria compartida 99
5.1. Esquemas parametrizados en memoria compartida de metaheurísticas e hiperheurísticas 100
5.1.1. Esquemas parametrizados en memoria compartida 100
5.2. La metodología de modelado y autooptimización 102
5.3. Aplicación de la metodología a PCEPA y POPME 106
5.4. Conclusiones 113

6. Modelado y autooptimización de metaheurísticas paralelas en memoria distribuida 115
6.1. Esquemas parametrizados de metaheurísticas en memoria distribuida 116
6.1.1. Esquemas parametrizados en memoria distribuida 117
6.2. Aplicación del esquema metaheurístico parametrizado en memoria distribuida a PCEPA 119
6.2.1. Aplicación del esquema metaheurístico paralelo en sistemas computacionales homogéneos a PCEPA 120
6.2.2. Aplicación del esquema metaheurístico paralelo en un sistema computacional heterogéneo a PCEPA 131
6.3. Autooptimización de metaheurísticas aplicadas a PCEPA en sistemas homogéneos 140
6.4. Conclusiones 150

7. Conclusiones y trabajos futuros 153
7.1. Conclusiones 153
7.2. Resultados y entorno de trabajo 156
7.3. Trabajos futuros 160

A. Modelos de tiempos 163
A.1. Modelo de tiempos para memoria compartida 163
A.2. Modelo de tiempos para paso de mensajes 165

Bibliografía **167**

Índice de tablas

3.1. Clasificación de las diferentes combinaciones/hibridaciones de las metaheurísticas básicas usadas en los experimentos. 39
3.2. Valores de los parámetros para diferentes combinaciones de metaheurísticas aplicadas al problema PCEPA. 43
3.3. Valores de las variables del problema PCEPA considerados en los experimentos. 44
3.4. Valor de la función de fitness, C_e (€), obtenida para diferentes combinaciones de metaheurísticas para diferentes tamaños del problema PCEPA. 45
3.5. Tiempos de ejecución (en segundos) obtenidos para las diferentes combinaciones de metaheurísticas presentadas en la figura 3.1 para diferentes series del problema PCEPA. 49
3.6. Indicador común ($IC = 10^6/(f \cdot t)$) para las diferentes combinaciones de metaheurísticas presentadas en las figuras 3.1 y 3.2 para diferentes series del problema PCEPA. 50
3.7. Valores de los parámetros para diferentes combinaciones de metaheurísticas aplicadas al problema PCOCI. 55
3.8. Valores de las variables del problema PCOCI consideradas en los experimentos. 56
3.9. Valor de la función de fitness obtenida para diferentes combinaciones de metaheurísticas para diferentes series del problema PCOCI. 57
3.10. Tiempos de ejecución obtenidos para las diferentes combinaciones de metaheurísticas presentadas en la figura 3.7 para diferentes series del problema PCOCI. 60
3.11. Indicador común ($IC = 10/(f \cdot t)$) para las diferentes combinaciones de metaheurísticas presentadas en las figuras 3.7 y 3.8 para diferentes series del problema PCOCI. 61

4.1. Valores de los parámetros hiperheurísticos utilizados para la selección de metaheurísticas. Hiperheurística híbrida (Hhi), Hiperheurística híbrida reducida (Hre) e Hiperheurística basada en un Algoritmo Genético (Hge). 72

4.2. Valores de los parámetros para las cuatro metaheurísticas puras y para las metaheurísticas híbridas (Mhi1 y Mhi2) consideradas. 73

4.3. Límites inferiores y superiores de los parámetros metaheurísticos para la selección de individuos por medio de la hiperheurística. 74

4.4. Valores de las variables para los distintos tamaños de PCEPA considerados en los experimentos. 75

4.5. Comparativa de los valores de fitness (en euros), tiempos de ejecución (en segundos) e indicador común ($IC = 10^9/(f \cdot t)$) para las tres configuraciones hiperheurísticas consideradas obtenidos al ejecutar una instancia del problema PCEPA de tamaño 10-3. Implementación paralela con ComGru como función de combinación y fitness calculado como Fit1P1E. 76

4.6. Comparativa de los valores medios de fitness para diferentes instancias de PCEPA, obtenidos con dos implementaciones diferentes de la función de combinación para dos configuraciones de hiperheurísticas. 77

4.7. Valores de los parámetros metaheurísticos obtenidos al aplicar las hiperheurísticas híbrida (Hhi), híbrida-reducida (Hre) y genética (Hge) a una instancia de tamaño 10-3 de PCEPA (Fit1P1E) y a varias instancias (FitVPVE2). 78

4.8. Cociente del fitness respecto al mejor fitness, para varios tamaños del PCEPA, para cuatro metaheurísticas puras (GRASP, Búsqueda Tabú, Algoritmo genético y Búsqueda Dispersa), para metaheurísticas híbridas seleccionadas por las hiperheurísticas para el tamaño de problema 10-3 con dos fitness (Fit1P1E y FitVPVE2), y para hiperheurísticas directamente aplicadas a instancias de cada tamaño de problema. 81

4.9. Cociente del fitness respecto al mejor fitness, para varios tamaños del PCEPA, para cuatro metaheurísticas puras (GRASP, Búsqueda Tabú, Algoritmo genético y Búsqueda Dispersa), para metaheurísticas híbridas seleccionadas por las hiperheurísticas para el tamaño de problema 20-3 con dos fitness (Fit1P1E y FitVPVE2), y para hiperheurísticas directamente aplicadas a instancias de cada tamaño de problema. 83

4.10. Comparativa de la media de los tiempos de ejecución (en segundos) para las tres configuraciones hiperheurísticas consideradas al ejecutar tres instancias del problema PCEPA de tamaños 10-3 y 20-3 en paralelo. 83

4.11. Medias de fitness (f), tiempo (t) e indicador común ($IC = 10^5/(f \cdot t)$) para los seis tamaños de problema PCEPA obtenidas al aplicar la metaheurística seleccionada por las hiperheurísticas con el fitness calculado como Fit1P1E y FitVPVE2 para los tamaños de problema 10-3 y 20-3. 85

4.12. Medias de fitness (f), tiempo (t) e indicador común ($IC = 10^5/(f \cdot t)$) en la tabla 4.11 para cada hiperheurística y cálculo del fitness, media de cada hiperheurística con los dos métodos de cálculo del fitness (últimas tres filas) y para los dos métodos de cálculo del fitness cuando se usan en las tres hiperheurísticas (última columna). 85
4.13. Valores de los parámetros metaheurísticos obtenidos al aplicar las hiperheurísticas híbrida (Hhi), híbrida-reducida (Hre) y genética (Hge) a tres instancias de PCEPA variando la instancia en cada iteración en una misma ejecución (FitVP1E). 86
4.14. Cociente del fitness respecto al mejor fitness, para varios tamaños de PCEPA, para las cuatro metaheurísticas puras, para las metaheurísticas híbridas seleccionadas por las hiperheurísticas con problemas variables en una ejecución (FitVP1E), y para hiperheurísticas directamente aplicadas a instancias de cada tamaño de problema. 87
4.15. Comparativa de las medias de fitness (f), tiempo (t) e indicador común ($IC = 10^5/(f \cdot t)$) obtenidas al aplicar las mejores metaheurísticas alcanzadas con Fit1P1E, FitVPVE2 and FitVP1E para varias configuraciones hiperheurísticas a instancias de tamaños 20-6, 40-3 y 40-6 del problema PCEPA. 88
4.16. Comparativa del fitness obtenido al ejecutar las tres configuraciones hiperheurísticas directamente a las seis instancias diferentes de PCEPA, con el fitness obtenido al aplicar una de las mejores metaheurísticas híbridas (Mhi1) a esos problemas. 89
4.17. Resultados del test de Wilcoxon para dos muestras con un nivel de significación $\alpha = 0.05$, para los grupos de tamaños de problema que mostraron diferencias significativas en sus respectivos subgrupos algorítmicos. 91
4.18. Resultados del test de Wilcoxon para dos muestras con un nivel de significación $\alpha = 0.05$, para comparar los grupos M-M y M-P con el grupo de referencia M-H. 91
4.19. Valores de los parámetros metaheurísticos obtenidos al aplicar las hiperheurísticas híbrida-reducida (Hre) y genética (Hge) a tres instancias de PCOCI, variando la instancia en cada iteración en la misma ejecución (FitVP1E). 93
4.20. Cociente del fitness respecto al mejor fitness, para varios tamaños de PCOCI, para las cuatro metaheurísticas puras, para las metaheurísticas híbridas seleccionadas por las hiperheurísticas con problemas variables en una ejecución (FitVP1E), y para hiperheurísticas directamente aplicadas a instancias de cada problema. 94
4.21. Comparativa de las medias de fitness (f), tiempo (t) e indicador común ($IC = 10^5/(f \cdot t)$) obtenidas al aplicar las mejores metaheurísticas alcanzadas con FitVP1E para dos configuraciones hiperheurísticas a las instancias S4, S5 y S6 del problema PCOCI. 94

4.22. Comparativa del fitness obtenido al ejecutar dos configuraciones hiperheurísticas directamente a las tres instancias de test de PCOCI, con el fitness obtenido al aplicar una de las mejores metaheurísticas híbridas (Mhi2) a esos problemas. 95

4.23. Resultados del test de Wilcoxon para dos muestras con un nivel de significación $\alpha = 0.05$, para los grupos de tamaños de problema considerados. 96

5.1. Comparativa del speed-up experimental más alto (exp), el speed-up modelado (mod) y el obtenido en los experimentos al usar la metodología de autooptimización (exp-auto), para $NEIIni = 100$ y 500, en la rutina paralela de un nivel de `Inicializar`, al aplicar el esquema metaheurístico (EM) 107

5.2. Comparativa del speed-up experimental más alto (exp), el speed-up modelado (mod) y el obtenido en los experimentos al usar la metodología de autooptimización (exp-auto), para otras combinaciones de parámetros en la rutina paralela de dos niveles de `Inicializar`, al aplicar el EM a PCEPA en *Ben*. 107

5.3. Comparativa del speed-up experimental más alto (exp), el speed-up modelado (mod) y el obtenido en los experimentos al usar la metodología de autooptimización (exp-auto), para $NEIIni = 20$ y 100, en la rutina paralela de un nivel de `Inicializar`, al aplicar el HEM a PCEPA en *Ben*. 108

5.4. Comparativa del speed-up experimental más alto (exp), el speed-up modelado (mod) y el obtenido en los experimentos al usar la metodología de autooptimización (exp-auto), para dos combinaciones de $NEIIni$, $PEMIni$ y $IMEIni$ (c1: 50,50,1; c2: 100,50,1), en la rutina paralela de dos niveles de `Inicializar`, al aplicar el HEM a PCEPA en *Ben*. 108

5.5. Comparativa del speed-up obtenido con el número máximo de hilos disponible (max), con la mitad del número máximo de hilos disponible ($\frac{max}{2}$), el speed-up experimental más alto (exp) y el experimental obtenido con nuestra metodología de autooptimización (exp-auto) y usando el número de hilos óptimo obtenido a partir del modelo (mod), para $NEIIni = 100$ y 500, en la rutina paralela de un nivel de `Inicializar`, al aplicar el EM a PCEPA en *Ben*. 109

5.6. Comparativa del speed-up obtenido con el número máximo de hilos disponible (max), con la mitad del número máximo de hilos disponible ($\frac{max}{2}$), el speed-up experimental más alto (exp) y el experimental obtenido con nuestra metodología de autooptimización (exp-auto) y usando el número de hilos óptimo obtenido a partir del modelo (mod), para dos combinaciones de $NEIIni$, $PEMIni$ y $IMEIni$ (c1: 100,50,10; c2: 500,100,5), en la rutina paralela de dos niveles de `Inicializar`, al aplicar el EM a PCEPA en *Ben*. 110

5.7. Comparativa del speed-up obtenido con el número máximo de hilos disponible (max), con la mitad del número máximo de hilos disponible ($\frac{max}{2}$), el speed-up experimental más alto (exp) y el experimental obtenido con nuestra metodología de autooptimización (exp-auto) y usando el número de hilos óptimo obtenido a partir del modelo (mod), para $NEIIni = 20$ y 100, en la rutina paralela de un nivel de `Inicializar`, al aplicar el HEM a PCEPA en *Ben*. 110

5.8. Comparativa del speed-up obtenido con el número máximo de hilos disponible (max), con la mitad del número máximo de hilos disponible ($\frac{max}{2}$), el speed-up experimental más alto (exp) y el experimental obtenido con nuestra metodología de autooptimización (exp-auto) y usando el número de hilos óptimo obtenido a partir del modelo (mod), para dos combinaciones de $NEIIni$, $PEMIni$ y $IMEIni$ (c1: 50,50,1; c2: 100,50,1), en la rutina paralela de dos niveles de `Inicializar`, al aplicar el HEM a PCEPA en *Ben*. 110

5.9. Valores de las constantes del modelo para todas las funciones al aplicar el EM a PCEPA en *Saturno*. 111

5.10. Valores de las constantes del modelo para todas las funciones al aplicar el HEM a PCEPA en *Saturno*. 111

5.11. Valores de los parámetros metaheurísticos usados para los experimentos de autooptimización al aplicar el EM (metaheurísticas m1 y m2) y el HEM (hiperheurísticas h1 y h2) a PCEPA en *Saturno*. 112

5.12. Valores de los parámetros de paralelismo para cuatro combinaciones de parámetros metaheurísticos en *Saturno*. Número óptimo de hilos del primer nivel de paralelismo para todas las funciones al aplicar el EM a PCEPA (metaheurísticas m1 y m2); el valor del parámetro del segundo nivel se fijó a 1. Número óptimo de hilos (niveles uno y dos de paralelismo) para todas las funciones al aplicar el HEM a PCEPA (hiperheurísticas h1 y h2). 113

5.13. Speed-ups para varias combinaciones de parámetros metaheurísticos al aplicar todas las funciones del EM (metaheurísticas m1 y m2 en las tablas 5.11 y 5.12) y todas las funciones del HEM (hiperheurísticas h1 y h2 en las tablas 5.11 y 5.12) a PCEPA. Valores experimentales obtenidos con el número máximo de hilos disponible (max), con la mitad del número máximo de hilos disponible ($\frac{max}{2}$) y valores obtenidos con autooptimización (exp-auto), en *Saturno*. 114

6.1. Valores de los parámetros metaheurísticos usados en los experimentos de memoria distribuida. 121
6.2. Comparativa de los speed-ups mejores (y número de procesos con los cuales se han obtenido) y de los fitness obtenidos con los esquemas de paso de mensajes (MD, implementaciones 1 y 2) y de memoria compartida para dos combinaciones metaheurísticas (m1 y m2 en la tabla 6.1) aplicadas a PCEPA 50-6 en *Saturno*. 122
6.3. Resultados numéricos medios relacionados con la figura 6.3. Fitness medio obtenido al variar $NEMPar$ para varios valores de $NGMPar$ y variando $NGMPar$ para varios valores de $NEMPar$ en *Saturno* para las combinaciones metaheurísticas m8 y m9 de la tabla 6.1 para el tamaño de problema 50-6 de PCEPA. 126
6.4. Resultados numéricos medios relacionados con la figura 6.5. Tiempo (en segundos) medio obtenido al variar $NEMPar$ para varios valores de $NGMPar$ y variando $NGMPar$ para varios valores de $NEMPar$, en *Saturno*, para las combinaciones metaheurísticas m8 y m9 de la tabla 6.1 para el tamaño de problema 50-6 de PCEPA. 128
6.5. Número de procesos lanzados para tres combinaciones metaheurísticas (m3, m4 y m5 en la tabla 6.1) aplicadas a PCEPA 50-6 en el sistema heterogéneo *Saturno*(sat) + *Marte*(mar) + *Mercurio*(mer) + *Luna*(lun), con las técnicas de mapeo: *Cores Sin Sobrecarga* (CSS), *Sobrecarga Sin Balancear* (SSB), *Sobrecarga Totalmente Balanceada* (STB), y *ySTB*. 133
6.6. Speed-up alcanzado cuando se varía el número de procesos de STB al multiplicarlo por varios factores de reducción (0.1 a 0.9), en el sistema heterogéneo *Saturno* + *Marte* + *Mercurio* + *Luna* al aplicar el esquema metaheurístico a PCEPA 50-6. 136
6.7. Fitness alcanzado cuando se varía el número de procesos de STB al multiplicarlo por varios factores de reducción (0.1 a 0.9) en el sistema heterogéneo *Saturno* + *Marte* + *Mercurio* + *Luna* al aplicar el esquema metaheurístico a PCEPA 50-6. 136
6.8. $IC = \frac{10^6}{f \cdot t}$ alcanzado cuando se varía el número de procesos de STB al multiplicarlo por varios factores de reducción (0.1 a 0.9) en el sistema heterogéneo *Saturno* + *Marte* + *Mercurio* + *Luna* al aplicar el esquema metaheurístico a PCEPA 50-6. 139

6.9. Valores específicos de las constantes secuenciales del sistema *Saturno* y de los parámetros metaheurísticos considerados para el modelo en la ecuación 6.4. Funciones: Gen-Ini, generación de elementos en la inicialización; Mej-Ini, Mej-Ref y Mej-Com, mejora de elementos en la inicialización, mejora del conjunto de referencia y del de combinaciones, respectivamente; Com, combinación; Div-Ref y Div-Com, diversificación de los conjuntos de referencia y combinaciones; Inc, función de inclusión de elementos. 142

6.10. Comparativa del speed-up experimental más alto (exp), el speed-up modelado (mod) y el obtenido en los experimentos al usar la metodología de autooptimización (exp-auto), para las metaheurísticas m6 y m8 de la tabla 6.1, al aplicar el esquema metaheurístico a PCEPA 50-6 en *Saturno*. 144

6.11. Valores específicos de las constantes secuenciales del sistema *Marte + Mercurio* y de los parámetros metaheurísticos considerados para el modelo en la ecuación 6.4. Funciones: Gen-Ini, generación de elementos en la inicialización; Mej-Ini, Mej-Ref y Mej-Com, mejora de elementos en la inicialización, mejora del conjunto de referencia y del de combinaciones, respectivamente; Com, combinación; Div-Ref y Div-Com, diversificación de los conjuntos de referencia y combinaciones; Inc, función de inclusión de elementos. 146

6.12. Comparativa del speed-up experimental más alto (exp), el speed-up modelado (mod) y el obtenido en los experimentos al usar la metodología de autooptimización (exp-auto), para las metaheurísticas m6 y m8 de la tabla 6.1, al aplicar el esquema metaheurístico a PCEPA 50-6 en el clúster homogéneo *Marte + Mercurio*. 147

6.13. Comparativa del speed-up experimental más alto (exp), el speed-up modelado (mod) y el obtenido en los experimentos al usar la metodología de autooptimización (exp-auto), para las metaheurísticas m6, m8 y m9 de la tabla 6.1, al aplicar el esquema metaheurístico a PCEPA 50-6 en el clúster heterogéneo *Júpiter + Luna + Saturno + Marte + Mercurio*. 149

A.1. Valores específicos de las constantes del sistema y de los parámetros metaheurísticos para cada función considerada con un nivel de paralelismo. Con Gen-Ini representando la generación del conjunto inicial de elementos dentro de `Inicializar`. 164

A.2. Valores específicos de las constantes del sistema y de los parámetros metaheurísticos para cada función considerada con dos niveles de paralelismo. Con mejora en la inicialización (Mej-Ini), mejora en el conjunto de referencia (Mej-Ref), mejora en el conjunto de combinaciones (Mej-Com) y diversificación (Div). 165

A.3. Valores específicos de las constantes secuenciales del sistema y de los parámetros metaheurísticos considerados en la ecuación A.11. Funciones: Gen-Ini, generación de elementos en la inicialización; Mej-Ini, Mej-Ref y Mej-Com, mejora de elementos en la inicialización, mejora del conjunto de referencia y del de combinaciones respectivamente; Com, combinación; Div-Ref y Div-Com, diversificación de los conjuntos de referencia y combinaciones; Inc, función de inclusión de elementos. 166

Índice de figuras

1.1. Estructura de nuestra hiperheurística basada en un esquema metaheurístico parametrizado. La hiperheurística selecciona los parámetros metaheurísticos (PM) óptimos para un problema concreto. 3

3.1. Valor de la función de fitness, C_e (€), obtenida para diferentes combinaciones de metaheurísticas para diferentes tamaños del problema PCEPA. 46
3.2. Tiempos de ejecución obtenidos para las diferentes combinaciones de metaheurísticas presentadas en la figura 3.1 para diferentes series del problema PCEPA. 48
3.3. Indicador común ($IC = 10^6/(f \cdot t)$) para las diferentes combinaciones de metaheurísticas presentadas en las figuras 3.1 y 3.2 para diferentes series del problema PCEPA. 48
3.4. Valores de fitness obtenidos por diferentes metaheurísticas en iteraciones sucesivas para el tamaño de problema PCEPA 50-6. 51
3.5. Tiempos de ejecución obtenidos por las diferentes metaheurísticas de la figura 3.4 en iteraciones sucesivas para la serie S5 del problema PCEPA. 52
3.6. Ejemplo de solución metaheurística para una instancia reducida de prueba del problema PCEPA. 53
3.7. Valor de la función de fitness obtenida para diferentes combinaciones de metaheurísticas para diferentes series del problema PCOCI. 58
3.8. Tiempos de ejecución obtenidos para las diferentes combinaciones de metaheurísticas presentadas en la figura 3.7 para diferentes series del problema PCOCI. 59
3.9. Indicador común ($IC = 10/(f \cdot t)$) para las diferentes combinaciones de metaheurísticas presentadas en las figuras 3.7 y 3.8 para diferentes series del problema PCOCI. 59
3.10. Valores de fitness obtenidos por diferentes metaheurísticas en iteraciones sucesivas para la serie S5 del problema PCOCI. 62
3.11. Tiempos de ejecución obtenidos por las diferentes metaheurísticas de la figura 3.10 en iteraciones sucesivas para la serie S5 del problema PCOCI. 62

3.12. Evolución con el tiempo de (a) el pH y (b) la concentración de protones. Valores experimentales y calculados. 63

3.13. Evolución de (a) la concentración de especies carbonatadas con el tiempo y (b) la concentración de especies acéticas y calcio. Valores calculados en la simulación de la reacción. 64

4.1. Cociente del fitness respecto al mejor fitness, para varios tamaños del PCEPA, para cuatro metaheurísticas puras (GRASP, Búsqueda Tabú, Algoritmo genético y Búsqueda Dispersa), para metaheurísticas híbridas seleccionadas por las hiperheurísticas para el tamaño de problema 10-3 con dos fitness (Fit1P1E y FitVPVE2), y para hiperheurísticas directamente aplicadas a instancias de cada tamaño de problema. 80

4.2. Cociente del fitness respecto al mejor fitness, para varios tamaños del PCEPA, para cuatro metaheurísticas puras (GRASP, Búsqueda Tabú, Algoritmo genético y Búsqueda Dispersa), para metaheurísticas híbridas seleccionadas por las hiperheurísticas para el tamaño de problema 20-3 con dos fitness (Fit1P1E y FitVPVE2), y para hiperheurísticas directamente aplicadas a instancias de cada tamaño de problema. 82

4.3. Cociente del fitness respecto al mejor fitness, para varios tamaños de PCEPA, para las cuatro metaheurísticas puras, para las metaheurísticas híbridas seleccionadas por las hiperheurísticas con problemas variables en una ejecución (FitVP1E), y para hiperheurísticas directamente aplicadas a instancias de cada tamaño de problema. 87

4.4. Resumen estadístico de las medias de fitness obtenidas al aplicar diferentes algoritmos metaheurísticos a varios tamaños del problema PCEPA (10-x, 20-x y 40-x). Se han considerado cuatro conjuntos de algoritmos en las figuras (a) y (b): la aplicación directa de las tres configuraciones hiperheurísticas (H), el conjunto de metaheurísticas obtenidas a partir de las hiperheurísticas (M-H) con computación del fitness agrupado en Fit1P1E y FitVPVE2, la aplicación de las mejores metaheurísticas no seleccionadas automáticamente (M-M), y el conjunto de las cuatro metaheurísticas puras (M-P). La figura (c) es igual pero con un conjunto más (M-H') que computa el fitness como FitVP1E. 90

4.5. Cociente del fitness respecto al mejor fitness, para varios tamaños de PCOCI, para las cuatro metaheurísticas puras, para las metaheurísticas híbridas seleccionadas por las hiperheurísticas con problemas variables en una ejecución (FitVP1E), y para hiperheurísticas directamente aplicadas a instancias de cada problema. 93

4.6. Resumen estadístico de las medias de fitness obtenidas al aplicar diferentes algoritmos metaheurísticos a varios tamaños del problema PCOCI (S4, S5 y S6). Se han considerado cuatro conjuntos de algoritmos: la aplicación directa de las configuraciones hiperheurísticas Hge y Hre (H), el conjunto de metaheurísticas obtenidas a partir de las hiperheurísticas (M-H) con computación del fitness FitVP1E, la aplicación de las mejores metaheurísticas no seleccionadas automáticamente (M-M), y el conjunto de las cuatro metaheurísticas puras (M-P). 96

5.1. Speed-ups teórico y experimental variando el número de hilos del primer y segundo nivel de paralelismo en una rutina paralela de dos niveles al aplicar la hiperheurística basada en el esquema metaheurístico (HEM) 105

5.2. Speed-ups teórico y experimental variando el número de hilos para tres valores del parámetro $NEIIni$ en la rutina paralela de un nivel de `Inicializar` al aplicar el EM a PCEPA en *Ben*. 109

6.1. Speed-ups obtenidos con los esquemas de paso de mensajes al variar el número de procesos, para dos combinaciones metaheurísticas (m1 y m2 en la tabla 6.1) aplicadas a PCEPA 50-6 en *Saturno*. 122

6.2. Resumen estadístico del speed-up y del fitness obtenidos al aplicar dos combinaciones metaheurísticas a PCEPA 50-6 en paralelo, en *Saturno*. Se han considerado tres grupos: paso de mensajes con dos implementaciones de las funciones básicas del esquema (MD1 y MD2), y memoria compartida (MC). 124

6.3. Evolución del fitness con el número de elementos a migrar ($NEMPar$) para varias frecuencias de migración ($NGMPar$) en *Saturno* para las combinaciones metaheurísticas m8 y m9 de la tabla 6.1 para el tamaño de problema 50-6 de PCEPA. 126

6.4. Representación gráfica de los valores medios de fitness de la tabla 6.3. (a) Fitness medio obtenido al variar $NEMPar$ para varios valores de $NGMPar$ y (b) variando $NGMPar$ para varios valores de $NEMPar$. En *Saturno* para las combinaciones metaheurísticas m8 y m9 de la tabla 6.1 para el tamaño de problema 50-6 de PCEPA. . . . 127

6.5. Evolución del tiempo de ejecución (en segundos) con el número de elementos a migrar ($NEMPar$) para varias frecuencias de migración ($NGMPar$) en *Saturno* para las combinaciones metaheurísticas m8 y m9 de la tabla 6.1 para el tamaño de problema 50-6 de PCEPA. . . 128

6.6. Representación gráfica de los valores medios de tiempo de la tabla 6.4. (a) Tiempo medio en segundos obtenido al variar $NEMPar$ para varios valores de $NGMPar$ y (b) variando $NGMPar$ para varios valores de $NEMPar$. En *Saturno* para las combinaciones metaheurísticas m8 y m9 de la tabla 6.1 para el tamaño de problema 50-6 de PCEPA. . . 129

6.7. Evolución del speed-up con el número de procesos (p) en *Marte* + *Mercurio* para las combinaciones metaheurísticas m6, m7, m8 y m9 de la tabla 6.1 para varios tamaños de PCEPA. 130

6.8. Evolución del fitness con el número de procesos (p) en *Marte* + *Mercurio* para las combinaciones metaheurísticas m6, m7, m8 y m9 de la tabla 6.1 para varios tamaños de PCEPA. 131

6.9. Speed-up alcanzado al aplicar el esquema metaheurístico a PCEPA 50-6 en el sistema heterogéneo *Saturno* + *Marte* + *Mercurio* + *Luna*: (a) con las tres configuraciones básicas de procesos heterogéneos de la tabla 6.5 y las tres metaheurísticas consideradas en los experimentos heterogéneos y (b) cuando se varía el número de procesos de STB al multiplicarlo por varios factores de reducción (0.1 a 0.9). 135

6.10. Fitness alcanzado al aplicar el esquema metaheurístico a PCEPA 50-6 en el sistema heterogéneo *Saturno* + *Marte* + *Mercurio* + *Luna*: (a) con las tres configuraciones básicas de procesos heterogéneos de la tabla 6.5 y las tres metaheurísticas consideradas en los experimentos heterogéneos y (b) cuando se varía el número de procesos de STB al multiplicarlo por varios factores de reducción (0.1 a 0.9). 137

6.11. $IC = \frac{10^6}{f \cdot t}$ alcanzado al aplicar el esquema metaheurístico a PCEPA 50-6 en el sistema heterogéneo *Saturno* + *Marte* + *Mercurio* + *Luna*: (a) con las tres configuraciones básicas de procesos heterogéneos de la tabla 6.5 y las tres metaheurísticas consideradas en los experimentos heterogéneos y (b) cuando se varía el número de procesos de STB al multiplicarlo por varios factores de reducción (0.1 a 0.9). 138

6.12. Resumen estadístico de $IC = \frac{10^6}{f \cdot t}$, para las tres configuraciones básicas de procesos heterogéneos de la tabla 6.5 y las tres metaheurísticas consideradas (m3, m4 y m5 en la tabla 6.1) en la aplicación del esquema metaheurístico a PCEPA 50-6. 139

6.13. Tiempos experimentales de computación t_{cmp}, comunicaciones t_{rec} y t_{dif}, y de ordenación t_{ord} en función del número de procesos p al aplicar la metaheurística m7 a PCEPA en *Saturno*. Valores medios de cuatro series de ejecuciones con valores de $NEMPar$ correspondientes para cada valor de p a un 100 %, 75 %, 50 % y 25 % de migración. 141

6.14. Evolución en *Saturno* frente a p de (a) los tiempos de computación t_{cmp} y de recogida de elementos en el maestro t_{rec}, y (b) t_{rec} por separado para apreciar mejor la forma de la curva en la zona comprendida entre p=6 y p=24. 143

6.15. Evolución en *Saturno* de los tiempos experimentales y modelados de computación t_{cmp}, de recogida de elementos en el maestro t_{rec} y total $t_{total} = t_{cmp} + t_{rec}$, al aplicar la metodología de autooptimización a (a) y (b) la metaheurística m6 y (c) y (d) la metaheurística m8 de la tabla 6.1. 145
6.16. Evolución en el clúster homogéneo *Marte + Mercurio* de los tiempos experimentales y modelados de computación t_{cmp}, de recogida de elementos en el maestro t_{rec} y total $t_{total} = t_{cmp} + t_{rec}$, al aplicar la metodología de autooptimización a (a) y (b) la metaheurística m6 y (c) y (d) la metaheurística m8 de la tabla 6.1. 147
6.17. Tiempos experimentales de computación t_{cmp}, comunicaciones t_{rec} y t_{dif}, y de ordenación t_{ord} en función del número de procesos p al aplicar la metaheurística m7 a PCEPA en el clúster heterogéneo *Júpiter + Luna + Saturno + Marte + Mercurio*. Valores medios de cuatro series de ejecuciones con valores de $NEMPar$ correspondientes para cada valor de p a un 100 %, 75 %, 50 % y 25 % de migración. 148
6.18. Evolución en el clúster heterogéneo *Júpiter + Luna + Saturno + Marte + Mercurio* frente a p de (a) los tiempos de computación t_{cmp} y de recogida de elementos en el maestro t_{rec}, y (b) t_{rec} por separado para apreciar mejor la forma de la curva en la zona comprendida entre p=6 y p=24. 149
6.19. Evolución en el clúster heterogéneo *Júpiter + Luna + Saturno + Marte + Mercurio* de los tiempos experimentales y modelados de computación t_{cmp}, de recogida de elementos en el maestro t_{rec} y total $t_{total} = t_{cmp} + t_{rec}$, al aplicar la metodología de autooptimización a (a) y (b) la metaheurística m6, (c) y (d) la metaheurística m8 y (e) y (f) la metaheurística m9 de la tabla 6.1. 151

Índice de algoritmos

1. Cálculo de la función de fitness para un individuo en la optimización de PCOCI. 32
2. Esquema general de metaheurísticas. 36
3. Esquema parametrizado de metaheurísticas. 37
4. Esquema parametrizado en memoria compartida de metaheurísticas. . 101
5. Esquema parametrizado de metaheurísticas en memoria distribuida. Modelo de Islas(`S,ParamPar`). 118
6. Esquema Metaheurístico Secuencial($\texttt{S}_\texttt{i}$`,NGMPar`). 118

Capítulo 1

Introducción

Se introducen en este capítulo los conceptos que constituyen los fundamentos del trabajo desarrollado en esta tesis. Se hace una revisión de las técnicas metaheurísticas e hiperheurísticas y se justifica su idoneidad como métodos de resolución de problemas de optimización con elevado coste computacional. Se justifica además el uso de esquemas parametrizados paralelos de metaheurísticas como herramientas eficaces para la aplicación de estos métodos de optimización. Se presentan también una serie de objetivos a alcanzar en el presente trabajo, acompañados de la metodología empleada para conseguirlos y las herramientas computacionales que se usarán para tales fines.

1.1. Introducción

Algunos problemas prácticos no pueden ser abordados mediante métodos exhaustivos para encontrar una solución óptima o una solución de buena calidad. Para su resolución se utilizan métodos que pueden presentar un cierto componente aleatorio, denominados heurísticas o metaheurísticas [64, 71, 119, 128, 131]. Normalmente para encontrar una metaheurística adecuada para un problema concreto es necesario experimentar con varias metaheurísticas, lo cual puede ser costoso en tiempo de ejecución.

El uso de esquemas parametrizados de metaheurísticas facilita la aplicación de distintas metaheurísticas con solo variar algunos parámetros en el esquema [6, 7]. En el presente trabajo, se pretende describir el uso del esquema metaheurístico así como la manera de incluir nuevas metaheurísticas en él, con la idea de ilustrar al lector en la forma de abordar el problema de la adaptación del algoritmo a nuevas configuraciones específicas para su aplicación a problemas de optimización.

También como novedad, se pretende reutilizar el esquema metaheurístico para el desarrollo de hiperheurísticas destinadas a obtener de manera automática la mejor metaheurística para un problema o conjunto de problemas dados.

Además, se usará paralelismo dentro del esquema como una herramienta útil para reducir el tiempo de ejecución de los algoritmos mediante la creación de mo-

delos temporales y su posterior optimización mediante la selección automática de parámetros.

1.1.1. Metaheurísticas

Las heurísticas son métodos de búsqueda incompletos que no ofrecen garantías de éxito y quizás conllevan un componente aleatorio, y son comunmente aplicadas a problemas de optimización cuando no existe un método exacto de resolución, cuando existe un método exacto que consume mucho tiempo para ofrecer la solución óptima, cuando existen limitaciones de tiempo, o como paso intermedio para obtener una solución inicial para la aplicación de otra técnica [119, 128].

El término metaheurística a menudo aparece en la literatura como un término general [64, 71, 131]. Algunos autores reservan el término heurística para el procedimiento de decisión aplicado a un solo paso en la búsqueda y aplican el término metaheurística a estrategias de control global. Una metaheurística satisfactoria requiere normalmente de experimentación y desarrollo de varios métodos y su adaptación a cada problema particular, lo cual es un proceso computacionalmente costoso y de carácter no general.

Las metaheurísticas presentan una serie de inconvenientes cuando son aplicadas a un problema específico: el número de parámetros a ajustar para un buen rendimiento algorítmico puede requirir bastante tiempo, dificultad para entender dichos parámetros si no se está acostumbrado a estos métodos, y suelen ser soluciones específicas para instancias de problema concretas, con resultados no óptimos si se aplican a instancias diferentes de aquellas sobre las que se ha experimentado. Para eliminar estos inconvenientes se introduce el concepto de hiperheurística.

1.1.2. Hiperheurísticas

La idea básica de una hiperheurística es construir nuevos algoritmos para la resolución de problemas combinando, de algún modo, (meta)heurísticas conocidas de manera que se permita a cada una compensar las carencias de las otras [31, 32]. Deben entenderse como heurísticas para elegir otras heurísticas. Son métodos que trabajan con un espacio de búsqueda de heurísticas [107, 109]. En este sentido, difieren de la mayoría de las aplicaciones de las metaheurísticas, que normalmente trabajan en el espacio de las soluciones. Se puede seleccionar, generar o combinar diferentes (meta)heurísticas o componentes de (meta)heurísticas para resolver un problema de optimización dado de diferentes formas. Normalmente, hay una clara separación entre la hiperheurística de alto nivel y el conjunto de (meta)heurísticas de bajo nivel o sus componentes (figura 1.1). Supuestamente, hay una barrera de dominio entre ellas cuyo propósito es darle a la hiperheurística un nivel mayor de abstracción. Esto aumenta aún más el nivel de generalidad de la hiperheurística mediante la aplicación de su estructura a nuevos problemas cambiando simplemente el conjunto de (meta)heurísticas de bajo nivel. La barrera permite solo el flujo de

información independiente del problema desde el nivel inferior al superior. Entre esta información están los valores de la función de bondad (*fitness*), coste o penalización, medidos a través de una función de evaluación que indique la calidad de la solución. Las (meta)heurísticas o sus componentes son los elementos específicos del dominio del problema en una estructura hiperheurística, por lo que tienen acceso a toda la información importante de las soluciones candidatas. El trabajo de la estrategia de alto nivel es guiar la búsqueda de manera inteligente y adaptarla de acuerdo con el éxito o el fracaso de las (meta)heurísticas o combinaciones de (meta)heurísticas de bajo nivel durante el proceso de búsqueda, con el objetivo de reusar el mismo enfoque para resolver distintos problemas [11, 19, 33, 85, 86].

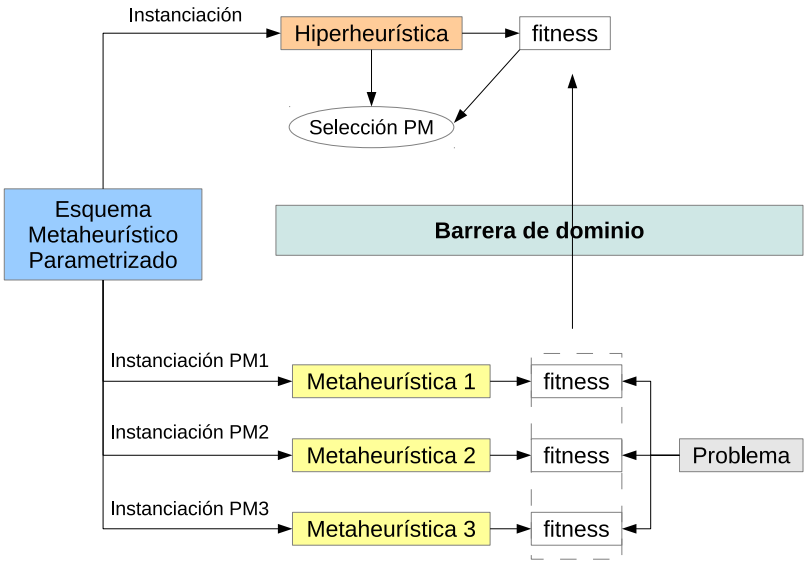


Figura 1.1: Estructura de nuestra hiperheurística basada en un esquema metaheurístico parametrizado. La hiperheurística selecciona los parámetros metaheurísticos (PM) óptimos para un problema concreto.

1.1.3. Esquemas parametrizados de metaheurísticas

El uso de un esquema parametrizado de metaheurísticas nos permite elegir fácilmente entre diferentes tipos de metaheurísticas o combinaciones de metaheurísticas simplemente seleccionando valores adecuados para los parámetros en el esquema [6, 7]. El esquema se ha aplicado con éxito en diferentes campos, para obtener Modelos de Ecuaciones Simultáneas satisfactorios a partir de un conjunto de valores de las variables [6], para un problema de asignación de tareas a procesadores con tareas independientes y restricciones de memoria [5] y para el problema p-hub [7].

Como novedad en el presente trabajo, se ha aplicado el esquema a la optimización del consumo eléctrico en la explotación de pozos de agua [61], para optimizar los valores de las constantes cinéticas de un modelo de reacción química [54] y en la búsqueda de la mejor combinación de parámetros metaheurísticos como problema de optimización aplicable a los dos anteriores [56, 58, 59].

En los trabajos iniciales se consideraban tres metaheurísticas puras: Búsqueda Voraz Adaptativa y Aleatoria (GRASP), Búsqueda Dispersa (SS) y Algoritmos Genéticos (GA), y en este trabajo se estudia la inclusión en el esquema de una nueva metaheurística, tomando como ejemplo la Búsqueda Tabú (TS).

No obstante, y como otra aportación de esta tesis, se puede dar un paso más y facilitar la implementación de hiperheurísticas utilizando el esquema unificado parametrizado. Así, la hiperheurística podrá seleccionar la combinación de parámetros metaheurísticos más adecuada para cada problema concreto, definiéndose así, como se verá en el siguiente capítulo, un nuevo problema de optimización a resolver desde un nivel superior de abstracción. Es importante optimizar la función de fitness y también, si es posible, minimizar el tiempo de ejecución. Por lo tanto, nuestra hiperheurística trata de satisfacer dos objetivos a la vez: la selección automática de la mejor metaheurística y la reducción del tiempo de ejecución de la hiperheurística y de la metaheurística resultante.

La metodología es aplicable a un conjunto grande de problemas de optimización, pero se han elegido algunos problemas concretos de test para comprobar su eficacia: un problema de consumo de electricidad en la explotación de pozos de agua (PCEPA), un problema de determinación de constantes cinéticas en un reacción química (PCOCI) y el propio problema de selección de la mejor metaheurística para resolver los dos anteriores (POPME). La resolución de los distintos problemas se aborda aplicándoles directamente metaheurísticas mediante el esquema general parametrizado [54, 61] y también se usa el esquema en la aplicación de las hiperheurísticas. Los resultados de ambas estrategias se pueden comparar para ver si las hiperheurísticas, o las metaheurísticas obtenidas a partir de éstas, consiguen mejorar los valores de fitness alcanzados con la aplicación directa del esquema metaheurístico y en tiempos de ejecución moderados.

1.1.4. Metaheurísticas paralelas y autooptimización

Aunque el esquema metaheurístico sea eficiente, su uso para resolver instancias de problema grandes causa incrementos significativos en el tiempo de ejecución. Basándonos en las posibilidades ofrecidas por las arquitecturas modernas de hardware, la aplicación de estrategias de computación de altas prestaciones a metaheurísticas es una interesante opción para reducir el tiempo de ejecución [3, 21]. Así, se han desarrollado un gran número de estrategias paralelas que pueden ser aplicadas a diferentes metaheurísticas [2, 27, 28, 45, 124].

En el presente trabajo, la paralelización de diferentes metaheurísticas se lleva a cabo a través del esquema unificado parametrizado de metaheurísticas, y por tanto

las diferentes metaheurísticas que se obtienen a partir del esquema se paralelizan de un modo unificado. Se han utilizado dos paradigmas de paralelización diferentes pero complementarios entre sí: por un lado, paralelismo de memoria compartida a nivel local en cada máquina, y por otro, paralelismo global de paso de mensajes aprovechando la memoria distribuida y la interconexión entre distintos nodos dentro de un mismo clúster. Además, es posible la combinación de los dos modelos de programación citados dando lugar a algoritmos paralelos híbridos que exploten las ventajas del paralelismo de memoria compartida y de paso de mensajes; aunque no se mostrarán resultados combinando los dos modelos pues los resultados obtenidos de esa forma no mejoran a los alcanzados con la utilización de un único paradigma de paralelismo.

Paralelizar el esquema reduce el tiempo de ejecución, pero tener una rutina paralela no asegura que sea usada correctamente, y el tiempo de ejecución de la rutina paralela puede estar lejos del óptimo (o incluso ser mayor que el tiempo secuencial) si el número de hilos y procesos usado en la aplicación de la rutina no es el apropiado.

El problema de autooptimización de rutinas secuenciales y paralelas ha sido estudiado en diferentes campos [26, 35, 47, 48, 70, 83, 84, 91, 138]. Se considera en el presente trabajo, por primera vez, la aplicación de las metodologías de autooptimización a metaheurísticas parametrizadas en memoria compartida y distribuida, siendo válidas las técnicas de autooptimización para las diferentes metaheurísticas obtenidas a partir del esquema. La misma metodología puede ser aplicada a hiperheurísticas que usen el mismo esquema metaheurístico para la selección satisfactoria de metaheurísticas. Si el esquema se desarrolla con autooptimización, se pueden reducir los tiempos de experimentación con una buena selección del número de hilos o procesos a usar en las diferentes partes del esquema paralelo.

1.2. Objetivos

Como objetivo general de esta tesis, se pretende mejorar y comprobar la utilidad de un algoritmo metaheurístico parametrizado paralelo para la resolución de problemas de optimización de distinta naturaleza.

Este objetivo principal se puede subdividir en una serie de objetivos parciales:

- La aplicación de un esquema unificado parametrizado de metaheurísticas a varios problemas de optimización utilizados como test para comprobar la idoneidad de dicho esquema en cuanto a la bondad de las soluciones alcanzadas.

- Se parte de las siguientes metaheurísticas puras: Búsqueda Voraz, Algoritmos Genéticos y Búsqueda Dispersa. Se plantea la metodología de introducción de una nueva metaheurística, utilizándose como caso de prueba la Búsqueda Tabú, cuya integración en el esquema metaheurístico posibilita la creación de un mayor número de metaheurísticas híbridas.

- Debido al tamaño que pueden llegar a tener los problemas de optimización a resolver y a su elevado coste computacional, se plantean varios subobjetivos de paralelismo:

  - La paralelización del esquema con dos subobjetivos claros: por un lado reducir el tiempo de ejecución y por otro ayudar a mejorar la efectividad de la propia metaheurística.
  - Para obtener una óptima paralelización del código se debería aplicar una estrategia de autooptimización que permita elegir los parámetros de paralelismo más adecuados para cada plataforma de cómputo. El método es diferente para cada tipo de paralelismo por lo que los subobjetivos también lo son:
    - En memoria compartida, se persigue obtener un modelo del tiempo de ejecución de las distintas rutinas del esquema metaheurístico como función de los parámetros metaheurísticos, de paralelismo y de las constantes del sistema particular estudiado. Dicho modelo nos permitirá fijar los parámetros de paralelismo óptimos (número de hilos de ejecución) de cada rutina para cada sistema concreto como función de los parámetros de la metaheurística.
    - En el caso de paralelismo de paso se mensajes se construirá un modelo global que considere el número total de procesos puestos en marcha, el coste de su inicialización y de las comunicaciones entre procesos. En este caso el modelo incluirá parámetros que reflejen el volumen y frecuencia de las comunicaciones entre procesos, valores que a su vez influyen en la bondad de los resultados obtenidos, por lo que la optimización del tiempo de ejecución influye en el resultado final de la metaheurística con que se trabaja. Por ejemplo, si la frecuencia de migración de elementos es baja o si se comunican pocos elementos entre procesos el tiempo de ejecución se reduce, pero a costa de alcanzar normalmente peores valores del fitness. Esto no pasaba en MC pues el número de hilos no influía en el fitness. Por otro lado, en cada proceso se consideran los modelos de tiempo secuenciales o de memoria compartida.

- Además, puesto que encontrar la mejor metaheurística supone un gran esfuerzo y numerosos experimentos variando los parámetros metaheurísticos, se desarrolla el concepto de hiperheurística con el objetivo de seleccionar de manera automática el conjunto óptimo de parámetros metaheurísticos para cada problema o conjunto de problemas. Dicha hiperheurística se basa en el mismo esquema unificado parametrizado de metaheurísticas, pero a un nivel superior de abstracción. Por ello, son aplicables aquí también las mismas técnicas de paralelización, las cuales tendrán un efecto mayor sobre la reducción del tiempo de ejecución debido a la mayor carga computacional de las hiperheurísticas.

1.3. Metodología

La metodología utilizada nos permite abordar de forma satisfactoria cada uno de los objetivos propuestos. Se puede hacer una división en varias técnicas y métodos para alcanzar los resultados deseados. Por un lado, tenemos toda la metodología referente al desarrollo y aplicación del esquema unificado de metaheurísticas que, además, es aplicable sin grandes modificaciones a la hiperheurística. Por otro lado, está la metodología de autooptimización aplicable en el ámbito del paralelismo de dichos esquemas. Se detallan estas ideas a continuación:

- Primeramente, para poder aplicar las diferentes metaheurísticas de manera sencilla mediante el esquema unificado, se ha ampliado el esquema metaheurístico con la introducción de una nueva metaheurística y se han adaptado sus funciones a los problemas de optimización considerados.

- Puesto que uno de los objetivos es desarrollar una hiperheurística que sirva como algoritmo de selección inteligente de las mejores metaheurísticas para problemas dados, se ha utilizado un método de desarrollo y aplicación basado en el mismo esquema metaheurístico, adaptándolo a las nuevas funciones.

- Para contrastar la utilidad de las metaheurísticas e hiperheurísticas estudiadas, se ha seguido una metodología basada en la experimentación sobre varios problemas usados como test.

- Se ha paralelizado el esquema utilizando los paradigmas de memoria compartida y paso de mensajes.

- Como el objetivo es reducir el tiempo de ejecución al máximo, en el esquema paralelo, se ha aplicado a cada rutina independientemente la metodología de autooptimización para obtener los parámetros de paralelismo óptimos. El proceso que se detallará para memoria compartida y para paso de mensajes en los capítulos 5 y 6 se divide en tres etapas siguiendo las ideas de [34, 35, 49]:

    - Diseño: Las rutinas se desarrollan junto con su tiempo de ejecución teórico. Se obtiene un modelo del tiempo de ejecución para cada rutina básica en memoria compartida y un modelo global que incluye términos de computación y de comunicaciones en memoria distribuida.
    - Instalación: Se estiman los valores de los parámetros dependientes del sistema particular en el que se ha instalado el esquema parametrizado paralelo.
    - Ejecución: En tiempo de ejecución se estiman los parámetros de paralelismo para cada rutina básica (número de hilos en memoria compartida) y los parámetros paralelos globales (número de procesos y de elementos a migrar y su frecuencia para paso de mensajes) como función de los

parámetros metaheurísticos (o hiperheurísticos) y de las constantes del modelo calculadas en la fase anterior.

- En todos los casos se ha contrastado el modelo teórico con los resultados experimentales alcanzados.
- Para evitar resultados excesivamente dependientes del sistema computacional elegido, se ha experimentado en varias máquinas con estructuras y configuraciones diferentes.

1.4. Herramientas computacionales

Se detallan a continuación los sistemas computacionales utilizados para llevar a cabo los experimentos tanto secuenciales como usando paralelismo de memoria compartida y de paso de mensajes en cada uno de los nodos del conjunto, así como el software usado: lenguaje de programación y librerías.

1.4.1. El hardware

Los sistemas computacionales utilizados se caracterizan por ser agrupaciones o clusters de nodos multiprocesadores con un número variable de procesadores por nodo. Según la capacidad computacional de los nodos dentro de un clúster, hablaremos de sistemas heterogéneos u homogéneos. La implementación del paralelismo (global) de paso de mensajes fue posible gracias a la interconexión entre nodos mediante una red. Cada nodo está constituido a su vez por un sistema de memoria compartida entre los cores que lo constituyen, lo que hace posible el paralelismo a nivel local. Los sistemas utilizados son:

- Red del Grupo de Investigación en Computación Científica y Programación Paralela de la Universidad de Murcia (PCGUM) [113]. Este sistema está compuesto por cinco nodos de computación, dos de ellos (*Marte* y *Mercurio*) con un procesador hexa-core AMD Phenom II X6 1075T a 3 GHz por nodo y con 15 y 8 GB de RAM respectivamente, otro (*Saturno*) con cuatro procesadores Intel hexa-core NEHALEM-EX EC E7530 a 1.87 GHz (24 cores en total) con 32 GB de memoria compartida, otro (*Júpiter*) con dos hexa-cores (12 cores) Intel Xeon E5-2620 a 2.00GHz y 32 GB de RAM, y finalmente (*Luna*) con un procesador quad-core Intel Core 2 Quad Q6600a 2.4 GHz con 4 GB de memoria. Los nodos están conectados mediante una red de 100 Mb/s. Dependiendo de la configuración de nodos considerada, tenemos un sistema homogéneo (*Marte+Mercurio*) o heterogéneo (*Saturno+Marte+Mercurio+Luna*).
- Supercomputador *Altix* del Barcelona Supercomputing Center [16]. Es una máquina con arquitectura cc-NUMA, memoria compartida y una red de interconexión Numalink 5 a 15GB/s. Dispone de 96 procesadores Nehalem-EX Intel(R) Xeon(R) CPU E7-8837 8-core a 2.67 GHz y 1.5 TB de memoria RAM.

- Supercomputador *Ben Arabí* del Murcia Supercomputing Center [105]. Se utilizó el nodo central (*Ben*) de memoria compartida, que es un HP Integrity Superdome SX2000 con 128 núcleos del procesador Intel Itanium-2 dual-core Montvale a 1.6Ghz y 1.5 TB de memoria RAM.

1.4.2. El software

La implementación de los algoritmos metaheurísticos e hiperheurísticos utilizados en este trabajo se ha hecho en el lenguaje de programación C++. Además, la introducción de paralelismo en los algoritmos ha seguido dos paradigmas complementarios: paralelismo en memoria distribuida implementada con la ayuda de la librería de paso de mensajes MPI (Message Passing Interface) [8, 108, 117, 129] y la interfaz de programación de aplicaciones OpenMP para el paralelismo en memoria compartida [8, 38, 39, 117, 133].

MPI es un estándar que define la sintaxis y la semántica de las funciones y procedimientos contenidas en una biblioteca de paso de mensajes diseñada para ser usada en programas que exploten la existencia de múltiples procesadores. El paso de mensajes es una técnica empleada en programación concurrente para aportar sincronización entre procesos y permitir la exclusión mutua. Su principal característica es que no precisa de memoria compartida, por lo que es muy importante en la programación de sistemas distribuidos.

OpenMP es una API que nos permite añadir concurrencia a las aplicaciones mediante paralelismo con memoria compartida. Se basa en la creación de hilos de ejecución paralelos compartiendo las variables del proceso padre que los crea. Consta de tres componentes complementarios: un conjunto de directivas de compilador, una librería de funciones en tiempo de ejecución, y un número limitado de variables de entorno. Se ha elegido la especificación para C++ y su implementación en Unix.

Además de considerar ambos paradigmas de programación por separado, se contempla, en un futuro, una hibridación teniendo en cuenta todos los parámetros de paralelismo simultáneamente, con el objetivo de aprovechar al máximo el paralelismo que ofrecen los sistemas computacionales considerados.

1.5. Estructura de la memoria

Se presenta aquí un resumen de los contenidos de esta tesis agrupados por capítulos. En el capítulo 1 se han introducido los conceptos fundamentales que se van a desarrollar en capítulos posteriores. Así, se resumen las técnicas y métodos de resolución de problemas y se presentan una serie de objetivos a alcanzar. Se puede encontrar la metodología empleada para conseguir dichos objetivos y las herramientas computacionales que se usarán para tales fines.

En el capítulo 2 se analizan las diferentes herramientas algorítmicas usadas en este trabajo para resolver los problemas que se plantean. Concretamente, se explica

el concepto de metaheurística, las razones que nos llevan a elegirlas como métodos de resolución idóneos para nuestros problemas, y su estructura y funcionamiento. También se introduce el concepto de hiperheurística, como un método de búsqueda de más alto nivel orientado a la selección automática de metaheurísticas. Seguidamente se presentan brevemente los esquemas unificados parametrizados de metaheurísticas destacando sus ventajas como herramienta de aplicación sencilla de metaheurísticas e hiperheurísticas a problemas de optimización. Por último, se describen los problemas de optimización que se usarán como casos de prueba para analizar la idoneidad de los métodos de resolución propuestos.

El capítulo 3 se centra en la aplicación del esquema unificado parametrizado de metaheurísticas a los dos problemas de optimización propuestos de forma secuencial. Se detalla la estructura interna del esquema, se enumeran las funciones básicas que lo constituyen, así como los parámetros metaheurísticos con los que se invocan. Se explica también un ejemplo de inclusión de una nueva metaheurística en el esquema (Búsqueda Tabú), y la facilidad con la que se aplica con la introducción de cuatro nuevos parámetros. Se presentan resultados de fitness alcanzados por las difetentes metaheurísticas y combinaciones analizadas, pudiéndose extraer conclusiones sobre el mejor método para cada problema.

En el capítulo 4 se va un paso más allá y se analiza la implementación de hiperheurísticas basadas en el esquema unificado parametrizado de metaheurísticas. Se verá en este caso cómo ahora el espacio de búsqueda es el propio espacio de metaheurísticas, que a su vez actúan sobre el espacio de los problemas. Se pueden comparar en este punto los resultados alcanzados con el método automático de selección de metaheurísticas con aquellos obtenidos mediante la aplicación directa de metaheurísticas a los problemas.

El capítulo 5 aborda el tema del paralelismo en memoria compartida, ampliándose el esquema unificado a un esquema parametrizado paralelo. Esto implicará la introducción de otro tipo de parámetros (paralelos) en el esquema para controlar los niveles y la intensidad del paralelismo establecido. La utilización de este paradigma de programación tiene como objetivo la reducción del tiempo de ejecución de los algoritmos. Se tratará de refinar todavía más el paralelismo con la inclusión de la metodología de autooptimización orientada a minimizar el tiempo de ejecución. Así, se procederá a desarrollar modelos del tiempo de ejecución para cada una de las rutinas que forman el esquema unificado parametrizado paralelo, para posteriormente determinar las constantes características de dichos modelos en los sistemas donde van a ser ejecutados (proceso de instalación). Finalmente, se procederá a la ejecución de las metaheurísticas con unos parámetros de paralelismo optimizados para el sistema concreto y dependientes de los parámetros metaheurísticos y de las constantes del modelo. La metodología de autooptimización será evaluada en varios sistemas y utilizando varios niveles de paralelismo para poder sacar conclusiones lo más generales posibles.

En el capítulo 6 se hará uso del paralelismo en memoria distribuida con algunas modificaciones en nuestra metodología. Se ampliará el esquema parametrizado

paralelo, con la introducción de parámetros paralelos para controlar el número de procesos a poner en marcha en cada sistema. Al igual que en el caso de memoria compartida, la utilización del paradigma de programación de paso de mensajes tiene como objetivo la reducción de los tiempos de ejecución de los algoritmos. Se harán estudios en sistemas homogéneos comparando los resultados con aquellos alcanzados en memoria compartida. También se analizará el comportamiento del esquema en sistemas heterogéneos pero siempre con una asignación de datos homogénea y poniendo en marcha un número de procesos en cada nodo proporcional a las prestaciones de este. Como en el capítulo anterior, se introducirá la metodología de autooptimización orientada a optimizar el tiempo de ejecución. Así, se procederá a desarrollar otros modelos del tiempo que tendrán en cuenta los procesos distribuidos en los sistemas computacionales considerados, así como la frecuencia y el volumen de las comunicaciones.

El capítulo 7 recogerá las principales conclusiones obtenidas a lo largo de esta tesis, junto con los resultados destacados y los proyectos y aportaciones realizadas gracias a este trabajo. También se proponen posibles vías futuras de investigación.

Se presenta un capítulo adicional (Apéndice A) donde se describe el desarrollo matemático llevado a cabo para obtener los parámetros de paralelismo óptimos a partir de los modelos de tiempos.

Capítulo 2

Herramientas algorítmicas y problemas de optimización

Se detallan en este capítulo las herramientas algorítmicas utilizadas para encontrar soluciones aceptables a los problemas de optimización planteados. Se va a presentar una clasificación de estos algoritmos y una descripción de su uso dentro de nuestra metodología. Como se comentó en la introducción, algunos tipos de problemas son inabordables por métodos exhaustivos, lo que hace necesario introducir técnicas de resolución aproximadas conocidas como heurísticas o metaheurísticas. Además, se puede ir un paso más allá al considerar algoritmos de control hiperheurísticos, con un nivel superior de abstracción, que permiten automatizar todavía más el proceso de búsqueda de soluciones satisfactorias. La aplicación de estos métodos se facilita con la introducción de esquemas unificados parametrizados de metaheurísticas, que permiten la experimentación con varios métodos simplemente cambiando los valores de algunos parámetros. Se detallan también los problemas utilizados como test para comprobar la idoneidad de nuestra metodología: un problema de minimización de costes eléctricos en la explotación de pozos de agua, un problema de optimización de las constantes cinéticas de un modelo de reacción química, y el propio problema de obtener los parámetros óptimos para las metaheurísticas aplicables a los dos problemas anteriores.

2.1. Herramientas algorítmicas

Se presentan primeramente las diferentes herramientas algorítmicas utilizadas para resolver los problemas propuestos. Se pretende dar una idea resumida y contextualizada de los algoritmos de optimización secuenciales y paralelos utilizados en este trabajo. Se empieza describiendo el concepto de metaheurística, sus ventajas como métodos de resolución adecuados para nuestros problemas, y su estructura y funcionamiento. Seguidamente se desarrolla el concepto de hiperheurística, como un método de búsqueda de más alto nivel orientado a la selección automática de metaheurísticas. Por último se abordan los esquemas unificados parametrizados

de metaheurísticas, destacando sus ventajas como herramientas de aplicación de múltilpes metaheurísticas e hiperheurísticas a problemas de optimización.

2.1.1. Metaheurísticas

La optimización en el sentido de encontrar la mejor solución, o al menos una solución lo suficientemente buena, para un cierto problema es un campo de vital importancia en ciencias e ingeniería. Debido a la gran importancia de los problemas de optimización, a lo largo de la historia de la Investigación Operativa se han desarrollado múltiples métodos para tratar de resolverlos. Destacan las metaheurísticas [64, 71, 131], especialmente útiles para la mayoría de los problemas combinatorios atractivos y de interés, donde los métodos exactos no son apropiados excepto para problemas de tamaño pequeño. Así pues, las metaheurísticas han surgido como técnicas adecuadas para desarrollar algoritmos de aproximación. Las metaheurísticas reúnen métodos e ideas de diferentes campos, como inteligencia artificial, matemáticas y biología. El punto más importante a su favor es su fácil e inmediata aplicabilidad a problemas complejos.

Podemos destacar ciertas propiedades fundamentales que caracterizan a este tipo de métodos:

- Las metaheurísticas son estrategias o plantillas generales que guían el proceso de búsqueda, cuyo objetivo es una exploración eficiente del espacio de búsqueda para encontrar soluciones (casi) óptimas.
- Son algoritmos no exactos, y generalmente no deterministas, que pueden incorporar mecanismos para evitar regiones no prometedoras del espacio de búsqueda.
- El esquema básico de cualquier metaheurística tiene una estructura predefinida y se hace uso del conocimiento del problema que se trata de resolver, en forma de heurísticos específicos que son controlados por una estrategia de más alto nivel.
- Una metaheurística satisfactoria requiere normalmente de experimentación y desarrollo de varios métodos y de su adaptación a cada problema particular, lo que es un proceso computacionalmente costoso y de carácter no general.
- Los elementos característicos de una metaheurística son, por una parte la función de fitness que permite asignar a una solución alcanzada un valor que refleja la bondad del resultado obtenido, y por otro lado tenemos los individuos que forman el conjunto de soluciones candidatas de un problema y que, codificados de alguna manera, constituyen el conjunto de referencia o población.

Las metaheurísticas resultarán eficaces para determinados problemas de optimización en la medida en que presenten un balance entre exploración (diversificación)

y explotación (intensificación). La exploración es necesaria para identificar partes del espacio de búsqueda con soluciones de alta calidad. La explotación es importante para intensificar la búsqueda en áreas prometedoras donde se ha explorado con insistencia. Las principales diferencias entre las metaheurísticas existentes se dan en la manera en que tratan de conseguir este balance. Existen muchos criterios de clasificación para las metaheurísticas, pero los más importantes tienen que ver con las características que definen el patrón de búsqueda que siguen, el uso de memoria, el tipo de exploración del vecindario usado o el número de soluciones que se consideran a través de las diferentes iteraciones del algoritmo. Suele tomarse como clasificación más diferenciadora la que distingue entre metaheurísticas basadas en trayectoria y metaheurísticas basadas en población. Así, las metaheurísticas basadas en trayectoria están más orientadas a la explotación mientras que las metaheurísticas basadas en población están más orientadas a la exploración.

En este trabajo, vamos a considerar cuatro metaheurísticas básicas para realizar los experimentos: Algoritmo Genético y Búsqueda Dispersa (basados en población), y Búsqueda Tabú y Búsqueda Voraz Adaptativa y Aleatoria (basados en trayectoria). Describiremos brevemente las características principales de cada una de ellas:

- Los Algoritmos Genéticos (*Genetic Algorithms, GA*) [15, 72, 103] son métodos basados en poblaciones de elementos que usan un conjunto de referencia para evolucionar estos elementos en sucesivas iteraciones y construir nuevas soluciones a partir de una población inicial de soluciones aleatorias. Los elementos o individuos son codificados en cadenas de números (binarios, enteros, reales, etc) y a cada uno se le asigna una función de bondad relacionada con el problema de optimización considerado. La evolución se consigue seleccionando, combinando y diversificando los elementos iniciales hasta conseguir nuevas soluciones mejoradas. Se trabaja iterativamente aplicando sucesivamente estos tres operadores hasta satisfacer algún criterio de terminación y se puede añadir una búsqueda local para mejorar algunos elementos.

  Se considera a J. Holland [77] como el creador (junto a sus colaboradores) de los Algoritmos Genéticos durante las décadas de 1960 y 1970.

- La Búsqueda Dispersa (*Scatter Search, SS*) [74, 87, 88] es un método evolutivo que usa un conjunto de referencia para combinar elementos y construir nuevas soluciones a partir de una población inicial dispersa de soluciones generadas aleatoriamente, pero que también puede incluir soluciones optimizadas previamente mediante búsquedas locales. Se elige un subconjunto a partir de este conjunto de referencia cuyos elementos se combinan para conseguir soluciones de partida a las que aplicar un procedimiento de mejora iterativo. Las principales diferencias con los GA son que los tamaños de las poblaciones son más reducidos y que no se considera diversificación dentro del proceso de búsqueda.

  Las ideas iniciales fueron propuestas por Glover en 1977 como una heurística para programación entera y se basó en estrategias para la creación de reglas

de decisión compuestas [87].

- La Búsqueda Tabú (*Tabu Search, TS*) es un procedimiento de búsqueda local cuya característica distintiva es el uso de memoria adaptativa como estrategia de resolución del problema [73, 75, 137].

  El origen del algoritmo se basa en las ideas presentadas por F. Glover de prohibir temporalmente soluciones muy parecidas a las últimas soluciones del recorrido.

  Sus características clave son: fuertemente basado en el uso de memoria, procesos sistemáticos (se minimizan los componente aleatorios), procesos de aprendizaje implícitos y explícitos, la memoria adaptativa permite encontrar un compromiso entre intensificación y diversificación, permite movimientos de empeoramiento para escapar de óptimos locales, y emplea mecanismos de reinicio para mejorar la habilidad de exploración-explotación del espacio de búsqueda [130]. Las dos últimas características se consiguen a través de la memoria a corto plazo (o basada en lo reciente) y la memoria a largo plazo (o basada en lo frecuente).

- La Búsqueda Voraz Adaptativa y Aleatoria (*Greedy Randomized Adaptive Search Procedure, GRASP*) [68, 121, 122] de Feo y Resende es un procedimiento iterativo multiarranque donde cada iteración GRASP consiste en una fase de construcción, donde se construye una solución factible, seguida de un procedimiento de búsqueda local que trata de encontrar una solución localmente óptima. La fase de construcción de GRASP es básicamente un algoritmo aleatorio voraz. Repetidas aplicaciones del procedimiento de construcción originan diversas soluciones de partida para la búsqueda local.

Además de las metaheurísticas puras, es interesante considerar algún tipo de combinación o hibridación entre ellas que permita aprovechar las ventajas de cada una en un único método [22, 23, 65, 81, 106, 118, 120]. Una propuesta bastante conocida es el uso de métodos de búsqueda local dentro de métodos basados en población. Así, muchas de las aplicaciones metaheurísticas actuales hacen uso de procedimientos de búsqueda local para refinar las soluciones generadas. Esto es así porque la mayor fortaleza de los métodos basados en población es su capacidad de exploración, mientras que los métodos basados en búsqueda local (o trayectoria) tienen una gran capacidad de intensificación, esto es, de encontrar rápidamente las mejores soluciones en el vecindario de elementos dados. Al comienzo de la búsqueda generalmente se intenta capturar una visión global del espacio de soluciones y, normalmente, despúes se aplican iterativamente operaciones más simples y dependientes del problema para alcanzar soluciones nuevas, focalizando la inspección de regiones prometedoras del espacio de búsqueda. En resumen, los métodos basados en población son buenos para identificar áreas prometedoras del espacio de búsqueda en las cuales los métodos de búsqueda local pueden determinar fácimente las mejores soluciones.

A pesar del gran potencial de las metaheurísticas como algoritmos de optimización, como se dijo en la introducción, éstas presentan una serie de inconvenientes cuando son aplicadas a un problema específico, como la dependencia del problema, la dificultad para entender los parámetros metaheurísticos, o la propia estructura del algoritmo. Para tratar de resolver estas dificultades y automatizar un poco más el proceso de búsqueda de soluciones satisfactorias, se introduce el concepto de hiperheurística.

2.1.2. Hiperheurísticas

Como se comentó en la introducción, podemos definir las hiperheurísticas como algoritmos de optimización de alto nivel cuya finalidad es encontrar las mejores metaheurísticas para problemas de optimización dados [29, 31, 110, 111]. Puesto que tienen una estructura similar a la de las metaheurísticas pueden implementarse usando la misma metodología pero a un nivel de abstracción superior [30, 107, 109, 112].

La literatura contiene muchas clasificaciones de hiperheurísticas dependiendo de su estructura, su manera de abordar el problema o de si su propósito es resolver un sólo problema de optimización o un conjunto de problemas [32, 37]. Hay dos maneras principales de aplicar una hiperheurística:

- Selección de (meta)heurísticas (metodologías para la selección de (meta)heurísticas existentes).
- Generación de (meta)heurísticas (metodologías para generar nuevas (meta)heurísticas a partir de componentes de (meta)heuristicas ya existentes).

Una clasificación independiente de hiperheurísticas se basa en:

- La naturaleza del espacio de búsqueda de heurísticas.
- La fuente de realimentación durante el proceso de búsqueda.

Las hiperheuristicas se pueden usar para seleccionar o generar heurísticas constructivas (que procesan una solución o soluciones parciales y construyen una solución o soluciones completas), o heurísticas perturbativas (que operan con soluciones completas). La diferente selección de algoritmos determina la naturaleza del espacio de búsqueda de heurísticas.

Una dirección de investigación nueva de hiperheurísticas híbridas podría incluir una combinación de metodologías de selección y generación de heurísticas, o una combinación de heurísticas constructivas y perturbativas. Nuestra metodología hiperheurística permite el manejo fácil de metaheurísticas de cualquier categoría (incluyendo categorías híbridas) que definen la naturaleza del espacio de (meta)heurísticas. Así, asignando los rangos apropiados de parámetros para nuestras metaheurísticas, la hiperheurística puede manejar metaheurísticas constructivas y perturbativas, o una combinación de ellas.

Las ideas de optimización (meta)heurística aparecen frecuentemente como métodos para la configuración automática de algoritmos con diferentes enfoques: métodos para establecer los valores de los parámetros algorítmicos; instanciación de posibles algoritmos mediante la aplicación repetida de reglas definidas por una gramática [99]; y estrategias que describen el proceso de instanciación de gramáticas en términos de parámetros [80, 98]. Algunas de estas estrategias están basadas en modelos y pueden explorar el espacio de soluciones de forma eficiente descartando los elementos menos prometedores desde un punto de vista estadístico [79]. Los parámetros algorítmicos pueden ser numéricos o categóricos; en este último caso permitiendo la combinación de diferentes componentes heurísticos para formar un algoritmo específico para un problema particular. Las heurísticas pueden afinarse después con otros parámetros de tipo numérico, creándose así una estructura algorítmica flexible y altamente configurable.

Nuestro enfoque proporciona una configuración algorítmica que incluye parámetros de tipo numérico que son usados desde un conjunto de funciones básicas dentro de una estructura común de metaheurísticas. Se diferencia de otros métodos basados en programación genética guiada por gramáticas, donde el código algorítmico se modifica (optimiza) o partes de él (funciones) se instancian mediante parámetros categóricos.

Además, el uso de un esquema hiperheurístico parametrizado nos permite abordar el problema de construcción desde un punto de vista híbrido de selección-generación. Esto significa que se pueden aplicar partes de metaheurísticas a una o varias instancias del problema, y cambiar o añadir nuevas partes en cada iteración de acuerdo con su idoneidad, o se pueden manejar o modificar directamente metaheurísticas completas. Nuestra hiperheurística puede seguir ambas estrategias simultáneamente seleccionando automáticamente los parámetros metaheurísticos más adecuados en cada paso de la búsqueda.

Una hiperheurística puede no usar aprendizaje, o usar aprendizaje on-line (adquiriendo realimentación del proceso de búsqueda mientras se trabaja con una instancia), o aprendizaje off-line (con realimentación adquirida a través de entrenamiento con un conjunto de instancias). Una hiperheurística que combina la selección aleatoria simple de (meta)heurísticas con un método de aceptación de la mejora y movimientos de igual calidad es un ejemplo de uso de un enfoque sin aprendizaje [12, 111]. Si una hiperheurística incorpora un mecanismo para guiar adaptativamente el proceso de búsqueda y posibilita el enfoque para tomar decisiones informadas sobre seleccionar o generar una (meta)heurística de bajo nivel, entonces se trata de una hiperheurística con aprendizaje. Las técnicas de aprendizaje máquina son comunmente usadas en las hiperheurísticas. Por ejemplo, el aprendizaje por refuerzo (basado en la recompensa/penalización) es utilizado como un método de aprendizaje on-line para la selección de (meta)heurísticas. En nuestra metodología, el aprendizaje on-line se usa cuando la hiperheurística se aplica a una instancia individual del problema (con la selección de las mejores metaheurísticas en tiempo de ejecución) y el aprendizaje off-line aparece cuando se aplica la hiperheurística a un conjunto de

instancias y las metaheurísticas así obtenidas se aplican a instancias distintas.

2.1.3. Esquemas unificados parametrizados de metaheurísticas

La parametrización de algoritmos se utiliza en diferentes campos y con diferentes objetivos. Se pueden seleccionar parámetros para reducir tiempos de ejecución [89, 138] o para seleccionar una metaheurística o hibridación satisfactoria [9, 20, 80]. En algunos casos, se consideran parámetros de configuración o adaptación en procesos automáticos [79, 82]. En nuestro caso, se usarán parámetros dentro de un esquema unificado para seleccionar metaheurísticas (o hiperheurísticas) y para reducir tiempos de ejecución. La ventaja de usar un esquema unificado es que se pueden adaptar de manera sencilla las distintas metaheurísticas (secuenciales o paralelas) a los problemas de optimización dados, simplemente modificando algunos de los parámetros.

Se puede trabajar a un nivel de abstracción alto generalizando el concepto de parametrización mediante un esquema unificado parametrizado [7]. Así, se proporcionan operadores específicos cuando son necesarios, y el esquema se adapta a diferentes dominios de problema cuando así se requiere. Los operadores de bajo nivel se pueden compartir, y las implementaciones se pueden reutilizar entre los distintos métodos. Este esquema algorítmico general está absolutamente abierto, y la búsqueda de un buen método es guiada a través de la selección apropiada de los valores de los parámetros. El método permite generar metaheurísticas puras, como Algoritmos Genéticos o Búsqueda Dispersa, o incluso métodos híbridos combinando varias de ellas simplemente introduciendo diferentes valores de los parámetros durante su invocación. Una característica bastante importante de este enfoque es la posibilidad de producir nuevos algoritmos que no son exactamente metaheurísticas puras o híbridas. Así, es posible generar nuevas metaheurísticas que se pueden considerar intermedias entre varios métodos puros pero con características diferentes a éstos. Esta característica admite la generación eventual de un conjunto de métodos (incluyendo las metaheurísticas clásicas) que se pueden usar en varios escenarios diferentes. Se ha constatado que este esquema unificado parametrizado consigue un buen balance entre facilidad de uso, rendimiento, y portabilidad a nuevos dominios de problema. Por lo tanto, la parametrización unificada constituye, por sí sola, una metodología para el desarrollo rápido que puede ser usada en un amplio rango de problemas de optimización. Además, este método puede constituir la base de marcos metaheurísticos de alto nivel.

En el capítulo 3 se describirá el esquema parametrizado de metaheurísticas utilizado en este trabajo. Se analizarán cada una de las funciones básicas que lo constituyen así como los parámetros característicos de cada una de ellas.

2.1.4. Metaheurísticas paralelas y autooptimización

La aplicación del esquema unificado parametrizado de metaheurísticas ha demostrado ser una estrategia eficiente para resolver problemas de optimización. No obstante, el elevado tamaño de muchos de estos problemas hace que se incrementen considerablemente los tiempos de ejecución empleados en su resolución. La aplicación de técnicas de computación de altas prestaciones a metaheurísticas es una interesante opción para reducir los tiempos de ejecución [3, 43, 93, 132]. Así, se han desarrollado un gran número de estrategias paralelas que pueden ser aplicadas a diferentes metaheurísticas [2, 41, 97, 131]. Cada estrategia obedece o se enmarca en una arquitectura computacional diferente, siguiendo un modelo de programación específico para dicha arquitectura [135].

Los principales modelos de computación son los siguientes:

- *Memoria Compartida*. Todos los procesos, que pueden operar independientemente, pueden acceder a toda la memoria como un espacio de direcciones global. Así, los cambios en una posición de memoria realizados por un proceso son visibles por el resto de procesos. En nuestro caso, se utilizarán máquinas computadoras con un acceso a memoria no uniforme (NUMA) [125].

  En este modelo un único proceso puede poner en marcha varios hilos de ejecución o tareas de manera concurrente. Desde la perspectiva de la programación no son más que una biblioteca de subrutinas que son llamadas desde el código fuente paralelo.

- *Memoria Distribuida*. Los sistemas con memoria distribuida requieren de una red de comunicaciones para conectar la memoria entre procesadores. Los procesadores tienen su propia memoria local. Las direcciones de memoria de un procesador no mapean otros procesadores, por lo que éstos trabajan independientemente y los cambios en la memoria local de un procesador no afectan la memoria de los otros, no aplicándose aquí el concepto de coherencia de caché.

  Cuando un procesador necesita acceder a los datos de otro procesador, es tarea del programador definir explícitamente como y cuando comunicar los datos. La sincronización entre tareas es también una responsabilidad del programador. La red de interconexión varía en los diferentes sistemas, pero puede ser tan simple como una Ethernet [3, 139].

- *Memoria Híbrida Compartida-Distribuida*. Los sistemas más grandes y rápidos actuales emplean ambas arquitecturas de memoria compartida y distribuida. El componente de memoria compartida puede ser una máquina de memoria compartida o una GPU. La componente de memoria distribuida es la puesta en red de múltiples máquinas de memoria compartida o GPUs, que solo trabajan con su propia memoria (no con la de las otras máquinas). Por tanto, se requiere de una red de comunicaciones para mover los datos de una máquina a otra [18, 126].

En el presente trabajo, la paralelización de diferentes metaheurísticas se lleva a cabo a través del esquema unificado parametrizado de metaheurísticas y, por tanto, las diferentes metaheurísticas que se obtienen a partir del esquema se paralelizan de un modo unificado. Se han utilizado dos paradigmas de paralelización diferentes pero complementarios entre sí: por un lado, paralelismo de memoria compartida a nivel local en cada máquina, y por otro, paralelismo global de paso de mensajes aprovechando la memoria distribuida y la interconexión entre distintos nodos dentro de un mismo clúster. Además, se considera la combinación de los dos modelos de programación citados, dando lugar a algoritmos paralelos híbridos que explotan las ventajas del paralelismo de memoria compartida y de paso de mensajes.

Las metaheurísticas son algoritmos cuya estructura y modo de operar dificulta la paralelización de datos como paradigma de programación concurrente. No obstante, puesto que las metaheurísticas dividen su búsqueda del óptimo en diferentes elementos y diferentes regiones del espacio, aparece otros tipos de paralelismo que pueden ser explotados utilizando los modelos descritos anteriormente [4, 36, 46, 94, 96, 97, 104, 127]. Dependiendo de la fuente de paralelismo usada existen tres estrategias de paralelización de metaheurísticas:

- **Paralelismo de Tipo 1**. Este tipo de paralelismo (de bajo nivel) se da a nivel de iteración dentro de los procedimientos de búsqueda [136]. Puede haber varios niveles de paralelismo en cada rutina: se puede dividir en paralelo un bucle para varios elementos (primer nivel de paralelismo), y tratar también en paralelo la búsqueda en el vecindario de cada elemento (segundo nivel de paralelismo). Este paralelismo está normalmente diseñado para reducir el tiempo de ejecución pero no para mejorar la calidad de las soluciones o explorar nuevas regiones del espacio de búsqueda [42].

- **Paralelismo de Tipo 2**. También conocido como de descomposición del dominio, que consiste en dividir el espacio de búsqueda en varios conjuntos diferentes y explotar la búsqueda en cada uno de ellos como un procedimiento paralelo [90, 114]. Normalmente se utiliza un esquema maestro-esclavo donde cada esclavo analiza, de forma concurrente, su porción del espacio de búsqueda. Es el maestro el encargado de coordinar las comunicaciones entre esclavos en el caso de que las hubiera, así como de combinar todas las soluciones alcanzadas para obtener la óptima.

- **Paralelismo de Tipo 3**. Es un tipo de paralelismo de búsqueda múltiple en el que cada hilo de ejecución realiza un búsqueda concurrente en el espacio de soluciones con posibilidad de comunicaciones entre hilos durante la búsqueda o al final para intercambiar información sobre las mejores soluciones alcanzadas [40]. La implementación de diferentes estrategias en cada hilo y su combinación aumenta el rendimiento del algoritmo, que está destinado a mejorar la exploración del espacio de búsqueda, ofreciendo soluciones de calidad incluso en menos tiempo que de forma secuencial [44, 50, 78].

Paralelizar el esquema reduce el tiempo de ejecución, pero tener una rutina paralela no asegura que sea usada correctamente, y el tiempo de ejecución de la rutina paralela puede estar lejos del óptimo (o incluso ser mayor que el tiempo secuencial) si el número de hilos y/o procesos usado en la aplicación de la rutina no es el apropiado.

El problema de autooptimización de rutinas secuenciales y paralelas ha sido estudiado en diferentes campos, especialmente en rutinas básicas de álgebra lineal, que son el componente básico en la resolución de muchos problemas científicos [34, 35, 47, 48, 49, 70, 83, 138]. En esta tesis se considera, por primera vez, la aplicación de las metodologías de autooptimización a metaheurísticas parametrizadas en memoria compartida y distribuida, siendo válidas las técnicas de autooptimización para las diferentes metaheurísticas y combinaciones de metaheurísticas obtenidas a partir del esquema parametrizado. La misma técnica puede ser aplicada a hiperheurísticas que usen el mismo esquema metaheurístico para la selección satisfactoria de metaheurísticas o combinaciones/hibridaciones mediante la obtención de valores adecuados de los parámetros metaheurísticos en el esquema unificado. Para adaptar una metaheurística (o hiperheurística) a un problema concreto es necesario realizar un elevado número de experimentos, lo que implica un elevado tiempo de ejecución, y un esquema con autooptimización es muy útil para reducir los tiempos de experimentación con una buena selección del número de hilos y/o procesos a usar en las diferentes partes del esquema paralelo.

2.2. Problemas de optimización

Se resumen aquí los problemas de optimización que se usan como casos de prueba para mostrar la utilidad de las metaheurísticas y de las hiperheuristicas basadas en el esquemas parametrizados (paralelos) de metaheurísticas.

2.2.1. El problema de consumo de electricidad en la explotación de pozos de agua (PCEPA)

Comenzaremos describiendo el problema de minimización de costes en la explotación de pozos de agua [60, 61] utilizado como caso de prueba para el desarrollo de nuestra metodología. Se hará una primera introducción general sobre las características técnicas del problema y posteriormente se describirá la formulación matemática y algorítmica para la resolución del mismo.

Análisis técnico de la explotación de recursos hídricos

La gestión del suministro de agua a una gran ciudad es una labor compleja que depende fundamentalmente de la actuación de los operadores de la red. Aunque ciertas tareas se realizan de manera automatizada, la mayor parte de las decisiones se toman basándose en la intuición y la experiencia previa de los propios operadores.

Esto redunda en una falta de definición de la política global de gestión del sistema, que suele estar basada en las decisiones tomadas por distintos operadores para resolver los problemas que se puedan presentar en la red de suministro en cada situación concreta.

La adopción de una estrategia de eficiencia energética elaborada en un proceso de optimización global del régimen de explotación de aguas subterráneas y de la red de abastecimiento y distribución de agua a través de un control centralizado de determinación del esquema de operación de bombas más adecuado, permite minimizar el coste de explotación en la gestión de aguas de la empresa. Así, esta minimización pasará por la reducción de los costes de energía eléctrica utilizada en la explotación de las aguas subterráneas.

Ante la multitud de variables que pueden influir en la elección de los orígenes del agua producida, procedentes de los distintos sondeos disponibles, se automatiza dicho proceso de selección a partir de los requerimientos de demanda y de las condiciones propias de cada sondeo, como son: nivel piezométrico, calidad química, volumen máximo de extracción e interacción con otros sondeos en explotación. Adicionalmente, se tienen en cuenta los costes eléctricos de cada pozo y la tarifación horaria, factores fundamentales para la determinación de los costes de explotación.

Se partirá de una demanda cuya variación y distribución en el periodo a optimizar habrá sido ya previamente determinada por métodos apropiados y con la suficiente fiabilidad. Todos los datos no modelados pueden ser proporcionados en tiempo real por el sistema telemandado de la empresa de aguas, siendo fundamentales como parámetros de entrada diarios.

La demanda de agua de una población es variable en el tiempo. La cantidad de agua a suministrar para satisfacer dicha demanda deberá, por lo tanto, ser también variable en el tiempo. En general, el sistema de explotación consta de un conjunto de bombas de diferentes capacidades que extraen agua de diferentes sistemas acuíferos y aportaciones superficiales que contribuyen a la satisfacción de la demanda. Los caudales explotados llegan a una serie de depósitos desde donde son distribuidos por los distintos municipios.

En general, el sistema de bombeo cuenta con un conjunto de bombas de diferentes capacidades que transfieren agua a uno o más depósitos de regulación. Estas bombas trabajan combinadas para proporcionar la cantidad de agua necesaria, atendiendo a las restricciones del problema. Por lo tanto, dependiendo del momento, algunas bombas estarán encendidas y otras apagadas.

Programar el bombeo en una estación consiste en establecer la combinación de bombas a utilizar en cada intervalo de tiempo del horizonte de planificación. Por tanto, una programación de bombeo es el conjunto de todas las combinaciones de bombas a utilizar durante cada intervalo del horizonte de planificación. Una programación óptima de bombeo puede definirse, por ejemplo, como una programación que cumpla con las restricciones del problema (como atender la demanda), pero que además optimice los objetivos establecidos.

El coste de energía eléctrica consumida se define como el gasto que implica

el consumo de energía eléctrica por parte de las bombas empleadas para extraer el agua. Un factor importante en el coste de la energía eléctrica es la estructura tarifaria de la misma. En muchos sistemas de suministro de electricidad el coste de la energía eléctrica no es el mismo en todas las horas del día, dado que existen horas de mayor consumo y otras de menor consumo. Sin embargo, las instalaciones deben dimensionarse para el consumo máximo. Se intentará siempre concentrar la explotación en las horas de que consta el rango de tarifa más barata. De no ser posible, se intenta continuar la explotación en las correspondientes horas de tarifa intermedia, evitando en lo posible la explotación en horas con tarifas más caras.

En aquellas horas en las que no se centra la explotación, se ha tenido la precaución de continuar manteniendo un caudal mínimo que nos servirá para mantener siempre en carga las tuberías de abastecimiento, favoreciendo así la explotación desde un punto de vista técnico.

Formulación matemática y algorítmica del problema

Las metaheurísticas se aplican a problemas de consumo de electricidad de diferentes características [13, 14]. Como hemos visto, se considera un sistema hídrico que consiste en una serie de bombas (B) de potencia conocida, localizadas en pozos, que extraen un caudal de agua a lo largo de varios rangos de tiempo (R) en un día. El caudal total es la suma de los caudales aportados por cada pozo. Las bombas pueden estar en marcha o apagadas en un instante dado. Las bombas operan eléctricamente y la electricidad tiene un coste diario que debería ser mínimo. Se define la función coste de energía eléctrica C_e en base a la combinación de bombas adoptada en cada intervalo horario, en base a la energía empleada para elevar el caudal estimado a la altura pertinente y en base a la tarifa horaria aplicable a dicha energía. Con todo ello, establecemos la siguiente función objetivo, donde el valor de la potencia será conocido y función del caudal y la altura manométrica, que son datos conocidos extraídos de la curva de demanda y de las características de la explotación respectivamente. Además, debido a la existencia de programas específicos para su cálculo, el dato de potencia es de obtención directa:

$$\textbf{Minimizar} \quad C_e = \sum_{i=1}^{R} \sum_{j=1}^{B} T_i P_j N_i x_{ij} \tag{2.1}$$

sujeta a

$$H \cdot \sum_{i=1}^{R} \sum_{j=1}^{B} Q_j x_{ij} = V_{td} \tag{2.2}$$

$$\forall i, \quad \sum_{j=1}^{B} Q_j x_{ij} \geq Q_{min} \tag{2.3}$$

$$\forall j, \quad \frac{24}{R} Q_j \sum_{i=1}^{R} x_{ij} \leq V_{c,j} \tag{2.4}$$

$$\forall i, \quad \frac{\sum_{j=1}^{B} Q_j \sigma_j x_{ij}}{\sum_{j=1}^{B} Q_j x_{ij}} \leq \sigma_{lim} \tag{2.5}$$

$$\exists j \mid PD_j \leq PM_j \quad \Longrightarrow \quad \forall i, \quad x_{ij} = 0 \tag{2.6}$$

donde C_e representa el coste de electricidad consumida por la combinación de bombas seleccionada en un día; T_i es el coste de la electricidad en el rango horario i; P_j es la potencia eléctrica consumida por la bomba j; N_i es el número de horas de funcionamiento de las bombas en el rango horario i; y x_{ij} representa un elemento binario de una matriz con valores 1 o 0 para bomba encendida o apagada en la ecuación 2.1. En la ecuación 2.2, H representa el número de horas en cada rango horario (consideramos el mismo para todos los rangos); Q_j es el caudal extraído del pozo j, constante en todos los intervalos de tiempo en los que el pozo opera; y V_{td} es el volumen total diario demandado. En la ecuación 2.3, $Q_{min.}$ es el caudal total mínimo en la tubería para cada rango horario. $V_{c,j}$ es el volumen de explotación concedido diariamente al pozo j en la ecuación 2.4. En la ecuación 2.5, σ_j representa la conductividad de cada pozo y $\sigma_{lim.}$ es la conductividad límite de la mezcla de aguas. Y, finalmente, en la ecuación 2.6, PD_j es la profundidad (metros) del nivel dinámico del pozo j y PM_j es la profundidad máxima (metros) del nivel dinámico del pozo j.

Por tanto, en la codificación de una metaheurística, un elemento o individuo se representa por una matriz binaria, x, de tamaño $B \cdot R$, que codifica el conjunto de bombas distribuidas en diferentes intervalos de tiempo. El conjunto de individuos constituye el conjunto de referencia o la población. No todas las posibles combinaciones producen individuos factibles, y cada vez que se genera o modifica un individuo se evalúan las cinco restricciones correspondientes a las ecuaciones 2.2 - 2.6:

- **Satisfacción de la demanda** (ecuación 2.2). Esta restricción surje de la condición de que la suma de los volúmenes suministrados en el rango de horas establecido debe corresponder a la demanda programada al principio de cada día. Se intentará centrar la explotación en el rango horario más barato, y si no se llega a los requerimientos diarios establecidos el volumen restante se acumula al siguiente tramo horario de coste intermedio, evitando siempre la explotación con tarifa energética cara.

- **Mantenimiento del caudal mínimo** (ecuación 2.3). Por razones técnicas relativas al correcto mantenimiento de las tuberías de conducción de agua,

se pretende establecer un caudal mínimo en la tubería en todos los rangos horarios de operación aunque suponga un mayor coste por consumo eléctrico.

- **Conformidad con los volúmenes máximos de explotación de cada pozo** (ecuación 2.4). Cada pozo posee un volumen máximo de concesión anual, no debiendo éste ser superado al final de cada año de explotación. Como consecuencia, se debe imponer una restricción que limite diariamente el volumen explotado para no superar la parte proporcional de concesión diaria que le corresponda a cada pozo. Así, si un día no se extrae agua de un pozo, o se extrae menos del caudal de concesión acumulado que le correspondiera, este caudal no extraído se sumará al de concesión del día siguiente.

  En la práctica, el volumen acumulado concedido a cada pozo es actualizado diariamente e insertado como un parametro del problema.

- **Conformidad con la conductividad máxima establecida** (ecuación 2.5). La conductividad nos expresa una medida indirecta de la calidad química de las aguas, ya que proporciona una medida del contenido en sales disueltas en la misma. La conductividad eléctrica se define como la capacidad que tienen las sales inorgánicas en solución (electrolitos) para conducir la corriente eléctrica.

  En la mayoría de las soluciones acuosas, cuanto mayor sea la cantidad de sales disueltas, mayor será la conductividad. La conductividad no debería ser mayor que una cierta conductividad límite (2500 μS/cm a 20ºC) [24].

  Una vez establecido el valor límite de conductividad de la mezcla se debe determinar de una manera sencilla dicho valor en un caso real conociendo únicamente las conductividades de las aguas en el origen. Este punto se ha simplificado de manera que el valor de la conductividad de la mezcla se ha obtenido como una media ponderada en proporción a las conductividades y los caudales aportados. Si bien el valor de conductividad obtenido no va a ser el exacto para la mezcla de aguas, consideramos que sí proporcionará un orden de magnitud suficientemente cercano al propósito perseguido.

- **Conformidad con las profundidades máximas de los niveles dinámicos** (ecuación 2.6). La explotación de los pozos tiene como consecuencia un abatimiento en su nivel dinámico derivado de la extracción de un caudal continuo de los mismos. Dicho abatimiento no puede llegar a suponer un problema para el funcionamiento de las bombas, que pueden sufrir diversas patologías derivadas (cavitación) y, por supuesto, no puede suponer el agotamiento del pozo (a menos que el decisor lo crea oportuno). Es por ello que se establece como restricción la profundidad máxima a la que puede llegar el nivel dinámico de cada pozo. Ahora bien, en lugar de establecer la oportuna formulación para modelar el cono de descensos en cada sondeo, se opta por obtener diariamente un vector donde se recogen las profundidades a las que se encuentra

el nivel dinámico de cada pozo y compararlo con otro vector donde se recogen las profundidades máximas a las que éste puede llegar, quedando estas últimas afectadas por las holguras que se consideren oportunas. Los datos que constituyen los vectores mencionados son parámetros de obtención en tiempo real.

En el caso de que el nivel de un pozo alcanzara los márgenes determinados como de riesgo para el correcto funcionamiento hidráulico del mismo o para su bomba asociada, el pozo quedará fuera de servicio automáticamente, hasta que los niveles se vuelvan a estabilizar (presumiblemente en un nivel inferior al nivel estático del intervalo anterior) o en caso de que el decisor lo considere oportuno.

Esta formulación del problema significa en algunos casos que obtener un individuo es costoso en tiempo de ejecución debido a la dificultad de generar elementos factibles que cumplan con todas las restricciones. Además, para grandes explotaciones el número de pozos y rangos horarios puede ser grande, teniéndose en consecuencia tiempos de computación elevados.

2.2.2. El problema de determinación de las constantes cinéticas de una reacción química (PCOCI)

Se tiene aquí un problema clásico de optimización consistente en la estimación de los parámetros cinéticos que mejor se ajustan a los datos experimentales para un mecanismo de reacción dado. Este tipo de problemas se puede abordar con métodos metaheurísticos [63, 67]. El problema de minimización usado aquí como caso de prueba se describió en [54].

La determinación de las constantes cinéticas va asociada a la simulación del proceso de disolución de una pastilla antiácido en un medio de reacción de características similares al estómago humano. Así, se estudiará la cinética de la disolución del carbonato de calcio presente en la pastilla (fase sólida) como una función de la concentración de varias especies carbonatadas presentes en la disolución y, además, como función de la presión parcial de dióxido de carbono y del pH.

Cuando las reacciones químicas se producen en fase heterogénea, las variables que afectan a la velocidad de reacción no son sólo la temperatura, la presión o la composición. La velocidad de transferencia de materia adquiere importancia y debería ser incluida en los términos de la velocidad de reacción. Cuando se comparan o combinan las velocidades de reacción en sistemas complejos, se tiene que tener en cuenta que si el cambio en una propiedad se produce por medio de varios caminos paralelos mutuamente independientes, la velocidad global es simplemente la suma de todas las velocidades individuales.

Disolución del carbonato de calcio

Según [10], la cinética de la disolución del carbonato de calcio es una función de la concentración de las diversas especies carbonatadas en la disolución y, por tanto, una función de la presión parcial de dióxido de carbono y del pH. En [116] se desarrolló un modelo de la disolución del carbonato de calcio. De acuerdo con este modelo, dependiendo del pH, existen cuatro vías diferentes por las que puede producirse la disolución del carbonato de calcio. A valores de pH inferiores a 3.5 la velocidad de la reacción es proporcional a la concentración de H_3O^+ según la vía de reacción 1:

$$CaCO_3 + H_3O^+ \leftrightarrow Ca^{2+} + HCO_3^- + H_2O \tag{2.7}$$

donde las siguientes reacciones también ocurren debido al ambiente ácido y a estar el sistema abierto a la atmósfera:

$$CH_3COOH + H_2O \overset{K_a}{\longleftrightarrow} CH_3COO^- + H_3O^+ \tag{2.8}$$

$$HCO_3^- + H_3O^+ \overset{K_a'^{-1}}{\longleftrightarrow} H_2CO_3 + H_2O \tag{2.9}$$

$$H_2CO_3 \overset{K_H^{-1}}{\longleftrightarrow} CO_2 + H_2O \tag{2.10}$$

A estos valores de pH, se observa una dependencia de la velocidad de reacción con la velocidad de agitación. Esto indica que el transporte de iones es el factor que controla la velocidad del proceso.

A valores superiores de pH, la velocidad de reacción se hace menos dependiente del pH, pero más dependiente de la presión parcial de dióxido de carbono (vía de reacción 2). En este caso el proceso es controlado por el transporte de iones y por la propia reacción [123]:

$$CaCO_3 + H_2CO_3 \leftrightarrow Ca^{2+} + 2 \cdot HCO_3^- \tag{2.11}$$

El ion bicarbonato formado de esta manera no puede ser neutralizado como en el caso previo, ya que el pH no es lo suficientemente ácido.

Cuando el pH está próximo a la saturación, la velocidad de disolución del carbonato de calcio decrece drásticamente, y empieza a ser importante la reacción de hidrólisis (vía 3):

$$CaCO_3 + H_2O \leftrightarrow Ca^{2+} + HCO_3^- + OH^- \tag{2.12}$$

Finalmente, a valores de pH todavía mayores, se podría añadir la reacción inversa de precipitación del carbonato de calcio, pero dada la naturaleza, concentración y cantidad de ácido empleados, la vía 4 de precipitación de carbonato puede ser despreciada.

Determinación de las ecuaciones para obtener los parámetros cinéticos

El pH del medio de reacción es bastante similar al existente en el estómago humano, el cual varía entre 3.0 y 4.0. Los valores de pH de muchos alimentos, particularmente frutas, están en este rango. Así, el ácido acético diluido con un pH en torno a 3.0 es un simulador efectivo, barato y razonablemente representativo de los ácidos orgánicos débiles presentes en el cuerpo humano, y se usa como reactivo en los experimentos.

Como se ha visto, la pastilla se puede disolver de tres maneras: por reacción con ácido acético (ecuación 2.7), por reacción con ácido carbónico (ecuación 2.11), y mediante reacción de hidrólisis (ecuación 2.12). La velocidad global del proceso de disolución de la pastilla es la suma de las velocidades individuales de los tres caminos. Por lo tanto la variación de moles de calcio con el tiempo es:

$$\frac{1}{V}\frac{dN_{Ca^{2+}}}{dt} = -k_1 a^{n_1}\left[H_3O^+\right]^{n_2} - k_2 a^{n_3}\left[H_2CO_3\right]^{n_4} - k_3 \tag{2.13}$$

donde k_1, k_2 y k_3 son las constantes de velocidad de reacción combinadas y n_1, n_2, n_3 y n_4 son los órdenes de reacción; a es el área de la pastilla; V es el volumen de la disolución; y $[H_3O^+]$ y $[H_2CO_3]$ son las concentraciones de iones hidronio y ácido carbónico respectivamente. La integración de la ecuación para un cierto valor de tiempo t_j es:

$$\left(\frac{\Delta N_{Ca^{2+}}}{V}\right)_{t_j} = \left(-k_1 a^{n_1}\left[H_3O^+\right]^{n_2} - k_2 a^{n_3}\left[H_2CO_3\right]^{n_4} - k_3\right)_{t_j} \cdot \Delta t_j \tag{2.14}$$

que proporciona el incremento de calcio en la disolución como función de las concentraciones de protones y ácido carbónico, y donde el área de la pastilla es conocida para cada incremento de tiempo. Por tanto, la concentración de Ca^{2+} en la disolución en un instante t_j viene dada por:

$$\left[Ca^{2+}\right]_{t_j} = -\sum_{k=t_0}^{t_j}\frac{\left(\Delta N_{Ca^{2+}}\right)_k}{V} \tag{2.15}$$

Concentración de las diferentes especies presentes en la disolución

Con la idea de estimar la validez del modelo, se integra la ecuación general 2.13. Esto requiere del conocimiento de la concentración de todas las especies involucradas en el proceso en cada momento. Se pueden relacionar las concentraciones de las diferentes especies presentes en la disolución por medio de un balance de carga:

$$\left[H_3O^+\right] + 2\cdot\left[Ca^{2+}\right] = \left[CH_3COO^-\right] + \left[HCO_3^-\right] \tag{2.16}$$

donde se han despreciado las concentraciones de los iones hidroxilo y carbonato debido a que la reacción se produce siempre en medio ácido. La concentración de

Ca^{2+} (mol/L) presente en estos balances se puede calcular a partir del número de moles de calcio disuelto, calculados integrando la ecuación 2.13. La concentración de las otras especies se puede calcular como una función de la concentración de iones hidronio y de la concentración de calcio en la disolución. Para calcular las concentraciones de ácido acético e ion acetato, el balance de materia se puede escribir así:

$$[CH_3COOH]_0 = [CH_3COOH] + \left[CH_3COO^-\right] \tag{2.17}$$

donde $[CH_3COOH]_0$ es la concentración inicial conocida de acético. También tenemos la ecuación del equilibrio del ácido acético:

$$K_a = \frac{[CH_3COO^-]\,[H_3O^+]}{[CH_3COOH]} \tag{2.18}$$

y las concentraciones de acético y acetato se calculan como una función de $[H_3O^+]$ y $[Ca^{2+}]$.

Por otra parte, como se ha dicho previamente, la concentración de la totalidad de las especies carbonatadas se encuentra en todo momento en equilibrio con el dióxido de carbono atmosférico:

$$CO_2 + H_2O \overset{K_H}{\longleftrightarrow} H_2CO_3^* \qquad\qquad K_H = \frac{[H_2CO_3^*]}{p_{CO_2}} \tag{2.19}$$

Esto nos permite calcular la concentración de las especies H_2CO_3 y HCO_3^-, ya que $[H_2CO_3^*]$ representa la concentración de todas las especies carbonatadas en la disolución:

$$[H_2CO_3^*] = [H_2CO_3] + \left[HCO_3^-\right] \tag{2.20}$$

y estas especies están en equilibrio de la siguiente manera:

$$H_2CO_3 + H_2O \overset{K_a^{'}}{\longleftrightarrow} HCO_3^- + H_3O^+ \qquad\qquad K_a^{'} = \frac{\left[HCO_3^-\right]\,[H_3O^+]}{[H_2CO_3]} \tag{2.21}$$

Estas ecuaciones nos permiten expresar la concentración de las especies H_2CO_3 y HCO_3^- como una función de la concentración de los iones hidronio.

Determinación de los parámetros cinéticos y cálculo de la concentración de especies en disolución

Se presenta a continuación un esquema general para la determinación de los parámetros cinéticos y el cálculo de la concentración de las distintas especies:

1. Se estima el área inicial de la pastilla, a_0, a partir de los datos de la pastilla, de la disolución empleada y de las constantes de acidez. También se estima el número de moles iniciales de carbonato de calcio en la pastilla (N_{CaCO_3}), la concentración inicial de ácido acético ($[CH_3COOH]_0$) y la concentración de ácido carbónico en el medio ($[H_2CO_3^*]$). Obviamente, en $t = 0$ la concentración de calcio en la disolución ($[Ca^{2+}]$) es cero.

2. Se fija un incremento de tiempo y se asumen algunos valores iniciales para las constantes cinéticas k_i y n_i que aparecen en la ecuación 2.13.

3. Se calcula la concentración de ion hidronio ($[H_3O^+]$) resolviendo las ecuaciones 2.15-2.21.

4. A partir de $[H_3O^+]_{cal}$ y $[H_2CO_3^*]$, se calculan las concentraciones de las especies carbonatadas utilizando las ecuaciones 2.20 y 2.21.

5. Conocidos $[H_2CO_3]$ y $[H_3O^+]_{cal}$, el valor del área inicial de la pastilla (a_0) y con los valores supuestos de las constantes cinéticas, se integra analíticamente la ecuación 2.13 para el primer incremento de tiempo. Esto nos permite calcular el número de moles de calcio que están pasando a la disolución y, por lo tanto, $[Ca^{2+}]$ en ese instante (ecuaciones 2.14 y 2.15).

6. Con $[Ca^{2+}]$ y $[H_3O^+]_{cal}$, se calculan las concentraciones de las otras especies ($[CH_3COOH]$ y $[CH_3COO^-]$).

7. Se procede de la misma manera para todos los instantes de tiempo. Se usa la concentración de calcio calculada en el punto previo para calcular el área y la concentración de las diferentes especies en el siguiente intervalo de tiempo.

8. Para alcanzar la combinación de parámetros que mejor se ajusta al modelo cinético en un experimento, la función objetivo a minimizar es:

$$F.O. = \sum_{i=1}^{N} \left(pH_{exp,i} - pH_{cal,i}\right)^2 \tag{2.22}$$

   donde $pH_{exp,i}$ y $pH_{cal,i}$ (calculado a partir de $[H_3O^+]_{cal,i}$) representan los pH experimental y calculado en cada momento de la simulación. Se debe minimizar la diferencia para la suma de los N puntos del experimento.

Se han estudiado los efectos de diferentes parámetros físicos, como la velociadad de agitación, el pH inicial del medio, la cantidad total de ácido y base y el estado de agregación de la pastilla. Esto proporciona distintos conjuntos de experimentos, cada uno con su modelo teórico.

Resolución metaheurística del problema

Los valores de los parámetros cinéticos pueden ser estimados con metaheurísticas [54]. Un individuo se representa mediante un vector de números reales de tamaño siete, conteniendo dichos parámetros cinéticos. En este problema, el número de componentes de un individuo es bastante menor que en el problema previo, pero los posibles valores de cada parámetro son mucho mayores (números reales), y el espacio de búsqueda aumenta considerablemente. Cada vez que se tiene que evaluar

el fitness de un individuo, se debe resolver el sistema químico completo descrito en las subsecciones anteriores, por lo que la función de fitness coincide con la F.O. a minimizar de la ecuación 2.22, con un coste computacional elevado. El cálculo de la función de fitness se presenta en el algoritmo 1, con una complejidad lineal $O(N)$, donde se tiene un bucle con N intervalos de tiempo, que calcula el valor de cada variable del problema químico para un instante $i+1$ a partir del valor de dicha variable en el instante previo i. Por tanto, tenemos la siguiente función objetivo:

$$\textbf{Minimizar} \quad \sum_{i=1}^{N} \left(pH_{exp,i} - pH_{cal,i}\right)^2 \tag{2.23}$$

sujeta a

$$k_1 a^{n_1} \left[H_3O^+\right]^{n_2} \geq -k_2 a^{n_3} \left[H_2CO_3\right]^{n_4} \geq k_3, \quad pH \leq 3.5 \tag{2.24}$$

$$k_2 a^{n_3} \left[H_2CO_3\right]^{n_4} \geq k_1 a^{n_1} \left[H_3O^+\right]^{n_2} \geq k_3, \quad pH > 3.5 \tag{2.25}$$

$$\left[S\right]_i \geq 0, \quad \forall i \tag{2.26}$$

donde $\left[S\right]_i$ representa la concentración de cualquier especie en la disolución en el instante i.

Algoritmo 1 Cálculo de la función de fitness para un individuo en la optimización de PCOCI.

Calcular Fitness(k_1, k_2, k_3, n_1, n_2, n_3, n_4):
Para $i = 0 \rightarrow N$ **Hacer**
Calcular en el instante i: $\left[Ca^{2+}\right], a, \left[H_3O^+\right], \left[HCO^-\right], \left[H_2CO_3\right], pH_{cal}, \Delta\left[Ca^{2+}\right], \left[CH_3COOH\right], \left[CH_3COO^-\right]$
$Fitness = Fitness + \left(pH_{exp,i} - pH_{cal,i}\right)^2$
Fin Para

Cuando se llevan a cabo las simulaciones, existen una serie de restricciones que se deberán cumplir, tanto por parte de las constantes cinéticas como por las variables del problema. Primeramente, la vía de reacción principal debe ser respetada en cada rango de pH. Esto se consigue imponiendo la restricción por la cual, en el segundo miembro de la ecuación 2.13, el primer término sea mayor que el segundo a pH bajos, y el primer y segundo términos sean siempre mayores que el tercero. Así nos aseguramos de que las soluciones exploradas tengan sentido físico y respeten la vía principal de reacción en cada rango de pH (ecuaciones 2.24 y 2.25). Y en segundo lugar, todas las concentraciones de especies químicas deben ser positivas en

cada punto experimental (ecuación 2.26). Si se obtiene una concentración negativa para una combinación de parámetros cinéticos para cualquier especie química en un intervalo de tiempo, el individuo se vuelve no factible y se deberá descartar y generar nuevos elementos hasta encontrar uno factible.

2.2.3. El problema de selección de parámetros metaheurísticos óptimos (POPME)

La introducción de la hiperheurística como herramienta para la selección automática de las mejores metaheurísticas para un problema o conjunto de problemas, conlleva la aparición de un nuevo problema de optimización [56, 58, 59]: la selección de los parámetros metaheurísticos óptimos que definen dichas metaheurísticas óptimas para resolver ciertos problemas. Así pues, la hiperheurística buscará soluciones en el espacio de metaheurísticas, las cuales, a su vez, serán aplicadas a cada problema concreto.

En este caso, en la codificación de la hiperheurística, un individuo o elemento se representa por medio de un vector de enteros que contiene el conjunto de parámetros que caracterizan una metaheurística. El número y significado de los parámetros queda determinado por las metaheurísticas básicas integradas en el esquema metaheurístico, la implementación de las funciones básicas y los parámetros que se pueden variar en las funciones.

Como se verá en el capítulo 4 cuando describamos nuestra implementación hiperheurística [56, 58], en la versión considerada el número de componentes de un individuo asciende a veinte, que es el número de parámetros que definen cada metaheurística a aplicar. Tendremos aquí que el conjunto de individuos que constituye el conjunto de referencia estará formado por un conjunto de metaheurísticas o combinaciones/hibridaciones de metaheurísticas. El fitness de un elemento en la hiperheurística se basa en el valor que se obtiene cuando se aplica la metaheurística a una o varias instancias del problema para el cual se está obteniendo una metaheurística satisfactoria. En este caso las restricciones serán aquellas propias de los parámetros del esquema: que se sitúen dentro de los rangos de valores establecidos y ciertas incompatibilidades entre parámetros.

2.3. Conclusiones

Se han analizado las diferentes herramientas algorítmicas usadas en este trabajo para resolver los problemas que se plantean. Concretamente, se ha explicado el concepto de metaheurística, las razones para elegirlas como métodos de resolución idóneos para nuestros problemas y su estructura y funcionamiento. También se ha introducido el concepto de hiperheurística, como un método de búsqueda de más alto nivel orientado a la selección automática de metaheurísticas. También se ha presentado una introducción teórica sobre esquemas unificados parametrizados de

metaheurísticas destacando sus ventajas como herramienta de aplicación sencilla de metaheurísticas e hiperheurísticas a problemas de optimización. Por último, se han descrito los problemas de optimización que se usarán como casos de prueba para analizar la idoneidad de los métodos de resolución propuestos.

Capítulo 3

El esquema parametrizado de metaheurísticas: ampliación y aplicaciones

Vamos a describir en este capítulo el esquema metaheurístico parametrizado como herramienta de aplicación de metaheurísticas a problemas de optimización. Se detallarán las funciones que lo constituyen así como los parámetros metaheurísticos característicos de cada función. Además, se incluirá una clasificación de las combinaciones de metaheurísticas. Se presentará también, a modo de ejemplo, la manera de incluir una nueva metaheurística en un esquema metaheurístico dado, con la idea de ilustrar al lector en la forma de abordar el problema de la adaptación del algoritmo a nuevas configuraciones específicas para su aplicación a problemas de optimización.

Finalmente, se presentarán los resultados de la aplicación del esquema parametrizado de metaheurísticas a varios problemas de optimización, pudiéndose extraer conclusiones orientativas acerca de la mejor metaheurística obtenida en cada caso. Aunque veremos que las distintas metaheurísticas arrojan resultados diferentes dependiendo del problema a optimizar y de su configuración paramétrica, no se hará un estudio estadístico exhaustivo de estos resultados ya que lo que se pretende en este capítulo es constatar que el esquema es útil y eficaz a la hora de aplicar metaheurísticas y combinaciones de estas de forma sencilla. En el capítulo siguiente, se realizará un análisis más profundo de los resultados, comparando los valores de fitness obtenidos con la aplicación directa de metaheurísticas híbridas y los alcanzados al aplicar hiperheurísticas.

3.1. El esquema parametrizado de metaheurísticas

El concepto de esquema unificado parametrizado de metaheurísticas se desarrolla en [6, 7], y se ha aplicado satisfactoriamente en diferentes campos, para obtener

Modelos de Ecuaciones Simultáneas satisfactorios a partir de un conjunto de valores de las variables, al problema de asignación de tareas a procesadores con tareas independientes y restricciones de memoria, al problema p-hub, para la optimización de consumo de energía en la explotación de pozos de agua [61], y para la determinación de constantes cinéticas en una reacción química [54]. Se describen aquí las características del esquema.

La idea de representar metaheurísticas diferentes bajo un esquema común no es nueva. Vaessens et al. [134] y Raidl [118] usaron este enfoque y presentaron esquemas algorítmicos como el mostrado en el Algoritmo 2. El esquema considera un conjunto de funciones básicas (`Inicializar`, `CondiciónDeFin`, `Seleccionar`, `Combinar`, `Mejorar`, e `Incluir`) cuya instanciación determina la metaheurística particular que se está implementando. Los argumentos `S`, `SS`, `SS1`, y `SS2` se corresponden con el conjunto de soluciones que el método genera y manipula en sucesivas iteraciones. Las mismas implementaciones de las funciones básicas se pueden usar en métodos diferentes, y se pueden instanciar diferentes metaheurísticas con el mismo modelo. El esquema es también válido como un mecanismo genérico en la hibridación de metaheurísticas cuando, por ejemplo, las funciones básicas están compuestas por otras metaheurísticas o el mismo elemento de una metaheurística diferente.

Algoritmo 2 Esquema general de metaheurísticas.

```
Inicializar(S)
Mientras (no CondiciónDeFin(S)) Hacer
  SS=Seleccionar(S)
  SS1=Combinar(SS)
  SS2=Mejorar(SS1)
  S=Incluir(SS2)
Fin Mientras
```

También podemos observar que las funciones básicas del Algoritmo 2 podrían recibir parámetros adicionales, convirtiéndose así en un esquema unificado parametrizado de metaheurísticas (Algoritmo 3) que facilita el desarrollo de metaheurísticas y su aplicación [7]. Sin embargo, la selección de los valores adecuados de los *Parámetros Metaheurísticos* (*ParamX* en el algoritmo) para aplicar una metaheurística satisfactoria a un problema particular puede ser difícil y requiere de mucha computación. Tal como veremos en el siguiente capítulo, la selección de estos valores puede hacerse mediante un método hiperheurístico, el cual, además, puede ser desarrollado con el mismo esquema metaheurístco parametrizado.

Comentaremos a continuación cada una de las funciones del esquema parametrizado, sus variantes y los parámetros comunes para las metaheurísticas básicas consideradas (Búsqueda Voraz Adaptativa y Aleatoria (GRASP), Búsqueda Dispersa (SS), Algoritmos Genéticos (GA) y Búsqueda Tabú (TS)). Se ha considerado ya TS en el esquema general, aunque posteriormente se analizará su inclusión en detalle. También se considera la posibilidad de reutilizar las funciones básicas:

Algoritmo 3 Esquema parametrizado de metaheurísticas.

```
Inicializar(S,ParamIni)
Mientras (no CondiciónDeFin(S,ParamFin)) Hacer
  SS=Seleccionar(S,ParamSel)
  SS1=Combine(SS,ParamCom)
  SS2=Mejorar(SS1,ParamMej)
  S=Incluir(SS2,ParamInc)
Fin Mientras
```

- `Inicializar`: Se generan elementos aleatorios válidos para formar un conjunto inicial con $NEIIni$ elementos. Se selecciona un subconjunto más pequeño con $NEFIni$ elementos para las iteraciones en el Algoritmo 3. En algunas metaheurísticas (por ejemplo, SS y GRASP) algunos de los elementos iniciales se mejoran usando, por ejemplo, una búsqueda local o un método voraz. El parámetro $PEMIni$ indica el porcentaje de elementos a mejorar, y la mejora puede ser más o menor intensa, lo que puede ser representado con un parámetro de intensificación, $IMEIni$. El parámetro $MCPIni$ se usa para la extensión de la memoria a corto plazo Tabú de la mejora en la inicialización, y se explicará con más detalle en la sección de inclusión de la metaheurística Tabú en el esquema. Por tanto, se consideran cinco parámetros en la función de inicialización: $ParamIni = \{NEIIni, NEFIni, PEMIni, IMEIni, MCPIni\}$.

- `CondiciónDeFin`: Se puede considerar una condición de fin común a las diferentes metaheurísticas. Esta consiste en un número máximo de iteraciones ($NMIFin$) o en un número máximo de iteraciones sin mejorar la mejor solución ($NIRFin$), y $ParamFin = \{NMIFin, NIRFin\}$.

- `Seleccionar`: Se pueden agrupar los elementos del conjunto de referencia en dos conjuntos, los mejores y los peores de acuerdo con la función objetivo. El número de mejores elementos será $NEMSel$ y el de peores elementos $NEPSel$, y normalmente $NEMSel + NEPSel = NEFIni$. Por tanto, $ParamSel = \{NEMSel, NEPSel\}$.

- `Combinar`: El número total de elementos a obtener por combinación es $2(NMMCom + NMPCom + NPPCom)$, donde los tres parámetros representan el número de combinaciones de los mejores con los mejores elementos, el número de los mejores con los peores y el número de los peores con los peores elementos; estos constituyen el conjunto de parámetros de la función: $ParamCom = \{NMMCom, NMPCom, NPPCom\}$.

- `Mejorar`: Como en la mejora en la inicialización, $PEMMej$, $IMEMej$ y $MCMMej$ representan el porcentaje de elementos a mejorar, la intensificación de la mejora y la memoria a corto plazo en la mejora de los elementos de referencia y de los generados en la combinación, y $PEDMej$,

$IDEMej$ y $MCDMej$ representan los correspondientes valores en la diversificación, que es equivalente a la mutación en GA. Por tanto, $ParamMej = \{PEMMej, IMEMej, MCMMej, PEDMej, IDEMej, MCDMej\}$.

- `Incluir`: Los $NEMInc$ mejores elementos se mantienen en el conjunto de referencia, y los otros $NEFIni - NEMInc$ se seleccionan de los restantes elementos, con algún criterio de selección, por ejemplo, aleatoriamente o de acuerdo con alguna función de distancia. $MLPInc$ es un parámetro Tabú (memoria a largo plazo) usado para registrar los individuos explorados con más frecuencia. Así, $ParamInc = \{NEMInc, MLPInc\}$.

Los parámetros Tabú, cuya inclusión en el esquema metaheurístico es novedosa en el presente trabajo, se estudiarán con más detalle en la sección correspondiente. El conjunto de parámetros metaheurísticos es la unión de los seis conjuntos de parámetros de las funciones básicas del esquema. Así, tenemos un conjunto de veinte parámetros con los cuales es posible experimentar para hibridar, mezclar y adaptar las metaheurísticas al problema objetivo. Si se consideraran otras metaheurísticas básicas, el número de parámetros metaheurísticos y su significado cambiarían. No estamos interesados en una enumeración exhaustiva de todos los posibles parámetros que podrían ser incorporados al esquema, sino en mostrar como usarlo para facilitar la aplicación de metaheurísticas.

3.1.1. Clasificación de las combinaciones de metaheurísticas

Las metaheurísticas obtenidas utilizando el esquema unificado parametrizado quedan determinadas por los valores de sus parámetros, y son generalmente combinaciones o hibridaciones de metaheurísticas básicas. Así, dada la estructura funcional del esquema parametrizado y las metaheurísticas puras que son combinadas (GRASP, GA, SS y TS), la tabla 3.1 muestra la clasificación de diferentes combinaciones de las metaheurísticas básicas consideradas, usando la taxonomía y nomenclatura establecida en [81]. El significado de las estructuras que aparecen en la clasificación es:

- $HRH(A_1 + A_2)$, implica una hibridación a alto nivel entre las metaheurísticas A_1 y A_2, que se ejecutan secuencialmente sin alterar su estructura interna.
- $LRH(A_1(A_2))$ significa la metaheurística A_2 embebida dentro de A_1 con una hibridación a bajo nivel entre ellas.
- Se considera adicionalmente un tipo diferente de hibridación, $LRH(A_1, A_2)$, que representa una combinación a bajo nivel de las metaheurísticas A_1 y A_2.

La diferencia entre los dos tipos de hibridación a bajo nivel es que en el segundo tipo, las metaheurísticas mezclan sus estructuras al mismo nivel, mientras que en el

Metaheuristic	Design
GR+TS	LRH(GR(TS))
GR+GA	HRH(GR+GA)
GR+SS	HRH(GR+SS)
GA+SS	LRH(GA,SS)
GA+TS	HRH(LRH(GA(TS))+TS)
SS+TS	HRH(LRH(SS(TS))+TS)
GR+GA+SS	HRH(GR+LRH(GA,SS))
GR+GA+TS	HRH(LRH(GR(TS))+LRH(GA(TS))+TS)
GA+SS+TS	HRH(LRH(LRH(GA,SS)(TS))+TS)
GR+SS+TS	HRH(LRH(GR(TS))+LRH(SS(TS))+TS)
GR+GA+SS+TS	HRH(LRH(GR(TS))+LRH(LRH(GA,SS)(TS))+TS)

Tabla 3.1: Clasificación de las diferentes combinaciones/hibridaciones de las metaheurísticas básicas usadas en los experimentos.

primero la primera metaheurística se complementa con la estructura de la segunda cuando se ejecutan.

Para clarificar como se aplica la nomenclatura a las metaheurísticas básicas y al esquema unificado, se discutirá sobre el significado de la última fila en la tabla 3.1, $HRH(LRH(GR(TS)) + LRH(LRH(GA, SS)(TS)) + TS)$. Se aplica inicialmente GRASP con TS con memoria a corto plazo. A continuación se aplica una combiandción de GA y SS, con una mejora de cada elemento obtenido por diversificación y con el uso de la memoria a corto plazo, tal como en TS. Finalmente, se considera una restricción de memoria a largo plazo para elementos frecuentes en TS. Todas las combinaciones de metaheurísticas consideradas incluyen varias metaheurísticas puras (heterogeneidad), se busca en todo el espacio de soluciones (son globales) y todas resuelven el mismo problema de optimización (son generales). En el ejemplo previo, la representación completa de la metaheurística podría ser $HRH(LRH(GR(TS)) + LRH(LRH(GA, SS)(TS)) + TS)(het, glo, gen)$, pero por simplicidad, estos términos se omiten en todas las combinaciones.

En nuestro enfoque, la combinación de metaheurísticas no está tan claramente delimitada. Las metaheurísticas A_1 y A_2 quedan determinadas por sus valores de los parámetros metaheurísticos p_1 y p_2 ($A_1 = MH\,(p_1)$ y $A_2 = MH\,(p_2)$), y se puede determinar una combinación de ellas con una combinación lineal de sus parámetros, $C = MH\,(\alpha_1 p_1 + \alpha_2 p_2)$. Y, en general, una metaheurística que siga el esquema del Algoritmo 3 queda determinada por los valores de los parámetros ($MH\,(p)$), los cuales se pueden corresponder con valores típicos de algunas metaheurísticas básicas, con valores de una hibridación o combinación, o con valores aleatorios que no se corresponden con ninguna metaheurística conocida. La tabla 3.1 muestra la nomenclatura para la combinación de dos, tres o cuatro de las cuatro metaheurísticas básicas consideradas, pero puede haber muchas más combinaciones posibles, una

por cada posible conjunto de valores (dentro de rangos válidos) de los parámetros metaheurísticos.

3.2. Inclusión de la Búsqueda Tabú en el esquema parametrizado

En la sección anterior, describíamos cada una de las funciones básicas del esquema, considerando unas implementaciones concretas para nuestros problemas, variantes de las funciones comunes y los parámetros de las metaheurísticas, y la posibilidad de reutilizar las funciones básicas. Ahora vamos explicar en detalle aspectos relacionados con la inclusión de la Búsqueda Tabú en las funciones del esquema parametrizado. Hay que decir que el presente trabajo partía de un esquema metaheurístico donde sólo se consideraban tres metaheurísticas básicas (GRASP, SS y GA), pero se decidió incluir una cuarta metaheurística para enriquecer el algoritmo y, a la vez, hacer una descripción minuciosa de cómo incluir una metaheurística nueva en el esquema.

Debido a la naturaleza de la Búsqueda Tabú (ver descripción en la sección 2.1.1), su inclusión en el esquema parametrizado de metaheurísticas se reduce a algunas funciones. Específicamente, la inclusión de Tabú sólo tiene sentido dentro de la función de mejora, donde las modificaciones repetitivas de uno o varios elementos del individuo deben penalizarse para evitar caer en óptimos locales o la aparición de bucles. Para hacer esto, se introduce una lista de movimientos recientes (lista tabú o memoria a corto plazo) que será actualizada en cada iteración del bucle de mejora: si se ha modificado un elemento, se registra el cambio modificando el valor correspondiente en la posición de memoria, cambiándolo al valor máximo dado por los nuevos parámetros de la metaheurística $MCPX$. Donde X puede ser Ini, Mej o Div, dependiendo de si la mejora se ha hecho en la inicialización, en la mejora o en la diversificación. Los otros valores de la memoria se actualizan restándoles uno en cada paso del bucle. Así, con nuestra configuración, sólo se modificarán los elementos que tengan un cero en las posiciones correspondientes en la memoria a corto plazo. Esto asegura que el cambio de valores no se ha hecho recientemente. Como criterio de aspiración, un individuo se puede modificar si mejora el mejor valor aunque los movimientos estén prohibidos en la lista tabú. Hay que recordar que la diversificación no evalúa el vecindario del individuo, sino que pretende explorar áreas alejadas de él. Respecto a la memoria a largo plazo, hay varias funciones candidatas a incluirla en el esquema, pero por simplicidad sólo se considerará en la función `Incluir`.

A continuación se describen en detalle aquellas funciones del esquema que se han modificado con la inclusión de Tabú. Se muestra una manera concreta de incluir Tabú, aunque existen otras posibilidades:

- `Inicializar`: Como ya se vio anteriomente, la función de inicialización con-

tiene un total de cinco parámetros que permiten, mediante su variación, definir distintas formas de inicializar metaheurísticas. Respecto a trabajos anteriores, se ha introducido aquí un nuevo parámetro, $MCPIni$, para concretar la extensión de la memoria a corto plazo tabú de la mejora en la inicialización, que permite controlar que no se produzcan demasiados movimientos de búsqueda en un mismo sentido.

- `Mejorar:` Ésta es otra de las funciones en las que aparece Tabú. En este caso, como novedad, se introducen los parámetros $MCMMej$ fijando la extensión de la memoria Tabú en la mejora y $MCDMej$ que hace lo mismo en la mejora tras la diversificación. Después de seleccionar un individuo para mejorarlo, se selecciona un conjunto de vecinos para mejorarlos independientemente, escogiendo el mejor individuo del vecindario finalmente. Después de cada mejora (modificación de un elemento por ejemplo), se evalúa si esa modificación es tabú o no.

- `Incluir:` Existen varias posibilidades para la inclusión, pero se ha considerado que un algoritmo genético incluye los mejores elementos obtenidos en los conjuntos de referencia y combinaciones, y una búsqueda dispersa incluye un porcentaje de los mejores y el resto de los más dispersos de acuerdo con una función de distancia. Tabú tiene un efecto en todos los individuos incluidos. El nuevo parámetro $MLPInc$ (memoria a largo plazo) permite el registro de los individuos más frecuentemente explorados por la metaheurística en el conjunto de referencia y el de combinaciones. Para este propósito, se calcula un individuo promedio usando estos conjuntos y también recordando la memoria a largo plazo de iteraciones previas. Se calcula la distancia de cada individuo del conjunto de referencia al individuo promedio. Aquellos individuos que están próximos al individuo promedio serán reemplazados por otro más lejano elegido de los conjuntos auxiliares. De este modo, se asegura que se exploren áreas nuevas más lejanas de las frecuentemente visitadas.

De esta manera se han introducido los cuatro parámetros nuevos correspondientes a la Búsqueda Tabú en el esquema parametrizado de metaheurísticas, pasando de los dieciséis parámetros originales a veinte.

De la misma manera que hemos introducido la Búsqueda Tabú en el esquema general, se podrían incluir otras metaheurísticas. Para ello, bastaría con considerar nuevos conjuntos de parámetros para cada nueva metaheurística, y su correcta introducción en las nuevas funciones que aparecerían en el esquema.

3.3. Aplicación del esquema metaheurístico al problema PCEPA

Para comprobar la eficacia del esquema parametrizado de metaheurísticas, se va a aplicar esta técnica al problema de minimización de costes por el consumo de electricidad en la explotación de pozos de agua [60].

Los experimentos se han llevado a cabo adaptando el esquema metaheurístico a nuestro problema. Las combinaciones de bombas en cada rango horario se generan aleatoriamente con la misma probabilidad para una bomba de estar inicialmente encendida o apagada en todos los tramos de tarificación. El tamaño de cada individuo quedará definido por el número de bombas considerado y el número de franjas horarias fijado. La complejidad del problema quedará también determinada por la severidad de las restricciones impuestas, haciendo un problema más o menos fácil de inicializar. Los valores de los parámetros usados para las distintas combinaciones de metaheurísticas consideradas se muestran en la tabla 3.2, donde las condiciones de finalización han sido omitidas, teniendo como valores típicos $NMIFin$=1000, $NIRFin = 10$.

Las columnas etiquetadas con GR, GA, SS y TS recogen los parámetros que conducen a las metaheurísticas GRASP, Algoritmos Genéticos, Búsqueda Dispersa y Búsqueda Tabú. Como hemos dicho, tenemos la novedad de la inclusión de la Búsqueda Tabú, lo que implica que el número de combinaciones de metaheurísticas asciende a quince, mientras que sin la inclusión de los parámetros propios de la Búsqueda Tabú hay siete posibles combinaciones. Por ejemplo, GR+GA+TS y GR+SS+TS son métodos GRASP seguidos de algoritmos genéticos o búsqueda dispersa con la inclusión de memoria Tabú, por lo que son técnicas híbridas. GR+GA+SS+TS representa un GRASP seguido de una combinación genético-búsqueda dispersa con una estructura de memoria Tabú intercalada. Puesto que se ha introducido, como novedad, la Búsqueda Tabú en el esquema, se concretan los valores de los parámetros Tabú considerados: $MCPX = 10$, porque es un buen valor promedio para los tamaños de problema elegidos, y $MLPInc = 30$, lo que implica una memoria a largo plazo cuya extensión es del orden del número medio de iteraciones efectuadas en los experimentos. Del resto de parámetros cabe destacar que se han considerado poblaciones iniciales relativamente grandes con valores altos de $NEIIni$ (entre 100 y 200), como también son altos los parámetros de combinación con valores de $NMMCom$, $NMPCom$ y $NPPCom$ de 90, 100 y 90 respectivamente, por ejemplo, en la Búsqueda Dispersa o sus combinaciones. En general, los porcentajes de mejora de elementos $PEMIni$ y $PEMMej$ se sitúan en 100 con intensidades de mejora en la inicialización $IMEIni$ de 50 en la mayoría de los casos. Los parámetros de la diversificación $PEDMej$ e $IDEMej$ suelen tener siempre valores bajos, y el resto de parámetros tomarán valores representativos de la metaheurística que determinan.

Los experimentos se llevaron a cabo con la configuración operativa del problema mostrada en la tabla 3.3, donde algunas variables del problema se han fijado a

		GR	TS	SS	GA	GR+TS	GR+SS	GR+GA	TS+SS	TS+GA
Ini	NEIIni	200	200	100	100	200	200	200	100	100
	NEFIni	1	1	20	100	1	20	100	20	100
	PEMIni	100	100	100	0	100	100	100	100	0
	IMEIni	50	10	50	0	50	50	50	50	0
	MCPIni	0	10	0	0	10	0	0	10	0
Sel	NEMSel	0	1	10	100	0	10	100	10	100
	NEPSel	0	0	10	0	0	25	0	10	0
Com	NMMCom	0	0	90	50	0	90	50	90	50
	NMPCom	0	0	100	0	0	100	0	100	0
	NPPCom	0	0	90	0	0	90	0	90	0
Mej	PEMMej	0	100	100	0	0	100	0	100	0
	IMEMej	0	5	5	0	0	5	0	5	0
	MCMMej	0	10	0	0	0	0	0	10	0
	PEDMej	0	0	0	10	0	0	10	0	10
	IDEMej	0	0	0	5	0	0	5	0	5
	MCDMej	0	0	0	0	0	0	0	0	10
Inc	NEMInc	0	1	10	100	0	10	100	10	100
	MLPInc	0	30	0	0	0	0	0	30	30

	SS+GA	GR+TS+SS	GR+TS+GA	GR+SS+GA	TS+SS+GA	GR+TS+SS+GA
NEIIni	100	200	200	200	100	200
NEFIni	50	20	100	50	50	50
PEMIni	100	100	100	100	100	100
IMEIni	50	50	50	50	50	50
MCPIni	0	10	10	0	10	10
NEMSel	25	10	100	25	25	25
NEPSel	25	25	0	25	25	25
NMMCom	90	90	50	90	90	90
NMPCom	100	100	0	100	100	100
NPPCom	90	90	0	90	90	90
PEMMej	100	100	0	100	100	100
IMEMej	5	5	0	5	5	5
MCMMej	0	10	0	0	10	10
PEDMej	10	0	10	10	10	10
IDEMej	5	0	5	5	5	5
MCDMej	0	0	10	0	10	10
NEMInc	25	10	100	25	25	25
MLPInc	0	30	30	0	30	30

Tabla 3.2: Valores de los parámetros para diferentes combinaciones de metaheurísticas aplicadas al problema PCEPA.

valores específicos razonables para el tamaño del sistema considerado y otros (entre corchetes) se han tomado aleatoriamente en un intervalo, representando valores reales de operación. Las variables utilizadas son aquellas descritas en la subsección 2.2.1. Las tarifas eléctricas (€/(Kw·h)) en los tres tramos horarios de facturación considerados son 0.168, 0.112 y 0.056, donde la tarifa básica de 0.056 se obtuvo de [25] y las otras dos se calcularon a partir de ésta, multiplicadas por dos y tres respectivamente, con el objetivo de diferenciar claramente entre costes por tramo horario de facturación eléctrica.

B-R	20-6	50-3	50-6	150-3	200-3	200-6
$V_{td} \cdot 10^{-4}$	4	10	10	25	35	35
$Q_{min} \cdot 10^{-2}$	10	25	25	80	90	90
Q_j	[40,300]	[40,300]	[40,300]	[40,300]	[40,300]	[40,300]
P_j	[30,400]	[30,400]	[30,400]	[30,400]	[30,400]	[30,400]
$\sigma_{lim} \cdot 10^{-2}$	25	25	25	25	25	25
$\sigma_j \cdot 10^{-2}$	[1,30]	[1,30]	[1,30]	[1,30]	[1,30]	[1,30]
$V_{c,j} \cdot 10^{-3}$	[1,10]	[3,10]	[3,10]	[5,15]	[10,20]	[10,20]
PD_j	[150,300]	[150,300]	[150,300]	[150,300]	[150,300]	[150,300]
PM_j	[250,350]	[230,350]	[230,350]	[250,350]	[230,350]	[230,350]

Tabla 3.3: Valores de las variables del problema PCEPA considerados en los experimentos.

En la figura 3.1 y la tabla 3.4 podemos ver los valores de fitness alcanzados al aplicar las distintas metaheurísticas y combinaciones a varios tamaños del problema PCEPA. En la mayoría de los casos el valor de la función objetivo (ecuación 2.1) es muy similar. Cuanto más bajo es el valor mejor es el resultado. La inclusión de Tabú no perseguía mejorar los resultados obtenidos con otras metaheurísticas, pero refleja el efecto de su consideranción en el esquema. En general, las combinaciones de metaheurísticas producen buenos valores de fitness, destacando que:

- Los Algoritmos Genéticos (GA) y las combinaciones de metaheurísticas que incluyen SS+GA son las más adecuadas para este problema de optimización.
- Con el método GRASP los resultados no son buenos en general, pero su hibridación con otras técnicas produce resultados satisfactorios.
- La metaheurística TS produce resultados intermedios normalmente aunque su hibridación con SS, con GR+SS y con SS+GA arroja los peores resultados en la mayoría de los casos.
- No es necesaria una hibridación de tipo cuaternario para alcanzar valores de bondad aceptables, aunque este tipo de combinación es efectiva la mayoría de las veces.

	GR	GR+TS+SS	TS+SS	TS+SS+GA	TS
20-6	2860.25	3245.76	2865.33	2840.92	2564.43
50-3	9123.97	10232.47	8891.75	8880.41	8385.82
50-6	10016.76	10594.90	10035.13	9789.85	8358.78
150-3	25933.37	26341.93	26186.97	24642.83	17271.43
200-3	35819.70	37678.13	34980.00	34598.30	24078.07
200-6	41065.60	24008.93	21602.40	23287.93	41054.70

	SS	GA	GR+SS	GR+TS	GR+GA
20-6	2520.97	2463.40	2524.55	2551.51	2547.55
50-3	7794.75	**7463.38**	7911.08	7796.24	7821.04
50-6	7785.64	**7577.02**	8042.12	7942.51	7819.84
150-3	15809.93	14545.97	15662.40	15772.27	16254.20
200-3	22090.57	21672.43	23015.10	22890.40	23109.33
200-6	37551.53	39789.57	23542.80	22513.33	21590.30

	TS+GA	GR+TS+GA	SS+GA	GR+TS+SS+GA	GR+SS+GA
20-6	2484.76	2528.21	**2460.79**	2522.54	2504.10
50-3	7647.06	7729.79	7607.04	7655.57	7687.98
50-6	7664.61	7689.25	7593.75	7782.81	7631.01
150-3	15528.13	15287.67	**14319.60**	14468.00	14588.53
200-3	21856.73	21848.97	21435.47	14666.47	**14314.80**
200-6	23413.67	**21504.13**	21615.77	22093.07	22024.43

Tabla 3.4: Valor de la función de fitness, C_e (€), obtenida para diferentes combinaciones de metaheurísticas para diferentes tamaños del problema PCEPA.

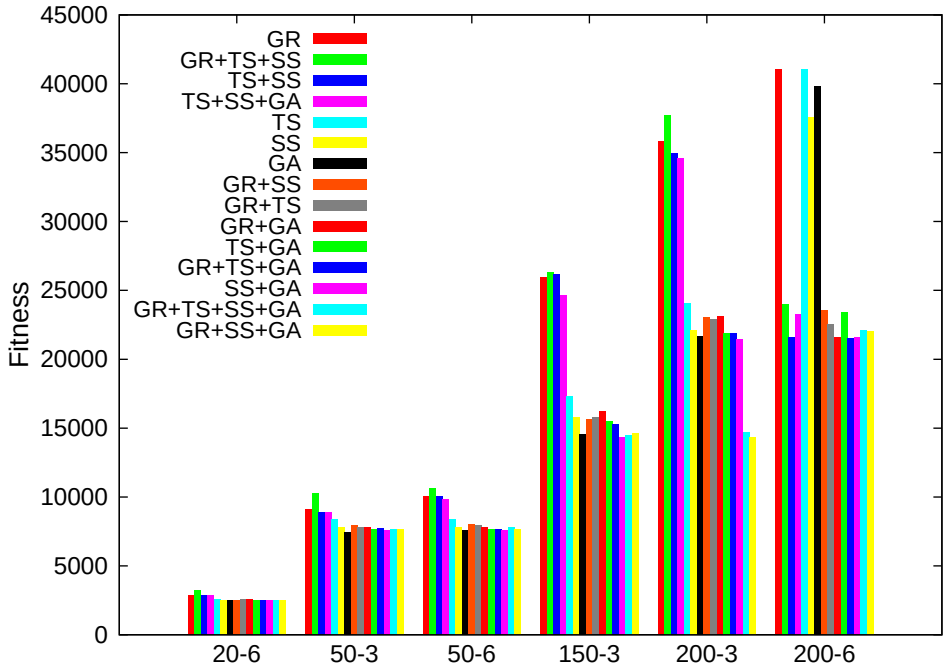


Figura 3.1: Valor de la función de fitness, C_e (€), obtenida para diferentes combinaciones de metaheurísticas para diferentes tamaños del problema PCEPA.

Como hemos visto, existen metaheurísticas que ofrecen mejores resultados de fitness que otras. Normalmente son las hibridaciones de metaheurísticas las que mejor se comportan aportando lo mejor de cada método individual a la estrategia global de resolución del problema. No obstante, es importante considerar también los tiempos de ejecución de cada metaheurística como criterio a tener en cuenta: no sólo importa la bondad de la solución alcanzada sino que también es importante el tiempo invertido en obtenerla. Así, en la figura 3.2 y la tabla 3.5 podemos ver los tiempos de ejecución empleados por las diferentes combinaciones de metaheurísticas presentadas en la figura 3.1 para los distintos tamaños del problema PCEPA. Debido a la diferencia en orden de magnitud de los resultados de tiempo de las distintas instancias, se ha hecho una representación semilogarítmica. Se puede apreciar que, como era de esperar en la mayoría de los casos, las metaheurísticas que alcanzan valores de fitness más bajos son las que consumen un mayor tiempo en su ejecución. Esto se aprecia resumidamente comparando las medias de las figuras 3.1 y 3.2, donde existe una inversión casi completa de la tendencia de los resultados. Esta tendencia, que en principio podría parecer obvia, a veces no se produce debido a que pueden existir configuraciones concretas de restricciones del problema que hagan que sea más difícil conseguir individuos factibles en cada iteración, con lo que el tiempo total puede incrementarse considerablemente aun considerando configuraciones metaheurísticas con valores reducidos de los parámetros o metaheurísticas puras que dieran lugar a resultados de menor calidad.

Para tener en cuenta tanto la abondad del resultado de fitness como el tiempo empleado en alcanzarlo por las metaheurísticas, vamos a introducir un indicador común que permita englobar ambos aspectos. Para encontrar una buena solución, sobre todo en instancias de problema grandes que requieran un consumo de tiempo elevado, mediremos la bondad de la solución con un indicador común definido como $IC = 10^6/(f \cdot t)$, donde f es el fitness y t es el tiempo de ejecución en segundos. Por tanto, son deseables valores elevados para este indicador. Existen otras posibilidades para definir el IC como, por ejemplo, dando más importancia al fitness (con un factor de ponderación mayor) que al tiempo de ejecución o considerando una función de bondad biobjetivo que integre ambas magnitudes. No obstante, el indicador elegido puede utilizarse como una primera solución al problema planteado.

En la figura 3.3 (semilogarítmica) y tabla 3.6 podemos ver los valores del IC para las diferentes combinaciones de metaheurísticas que resuelven las seis instancias del problema PCEPA. Debido a las notables diferencias en los valores de fitness y tiempos de ejecución de los diferentes tamaños de problema, solo son comparables los resultados entre cada metaheurística para un tamaño dado (comparación entre columnas para una misma fila en la tabla 3.6). Se aprecia que según este criterio, en general, los mejores resultados (valores mayores) corresponden a metaheurísticas que arrojan fitness moderados o altos y tiempos de ejecución relativamente bajos, como en el caso de TS, GR+TS y GR+TS+SS. Vemos que, bajo el criterio del indicador común, ya no son las combinaciones que incluyen SS+GA los mejores algoritmos puesto que, aunque ofrecen resultados buenos de fitness, el tiempo empleado es elevado. Así, podemos elegir otras metaheurísticas más rápidas a costa de perder un poco en calidad de la solución.

Podemos ver de forma gráfica como evoluciona el fitness y el tiempo de ejecución hasta alcanzar el óptimo al aplicar cada metaheurística. Así, la figura 3.4 muestra el valor de la función objetivo en iteraciones sucesivas para cada método, para el problema PCEPA con tamaño 50-6. Podemos ver que:

- Existen combinaciones de metaheurísticas que ofrecen resultados de fitness peores que el resto, como son, TS+GA, GR+TS y GR+TS+GA.

- Por otro lado, tenemos dos metaheurísticas puras, TS y GA, y las combinaciones GR+GA y TS+SS+GA, que ofrecen resultados aceptables pero a costa de un número elevado de iteraciones.

- En GRASP sólo se tiene una iteración (con valor relativamente alto de fitness), señalada como un punto en el gráfico y cuyo valor se puede consultar en la entrada correspondiente de la tabla 3.4.

- Finalmente, el resto de metaheurísticas o combinaciones ofrecen buenos resultados (excepto GRASP), destacando SS que alcanza muy buenos resultados en muy pocas iteraciones.

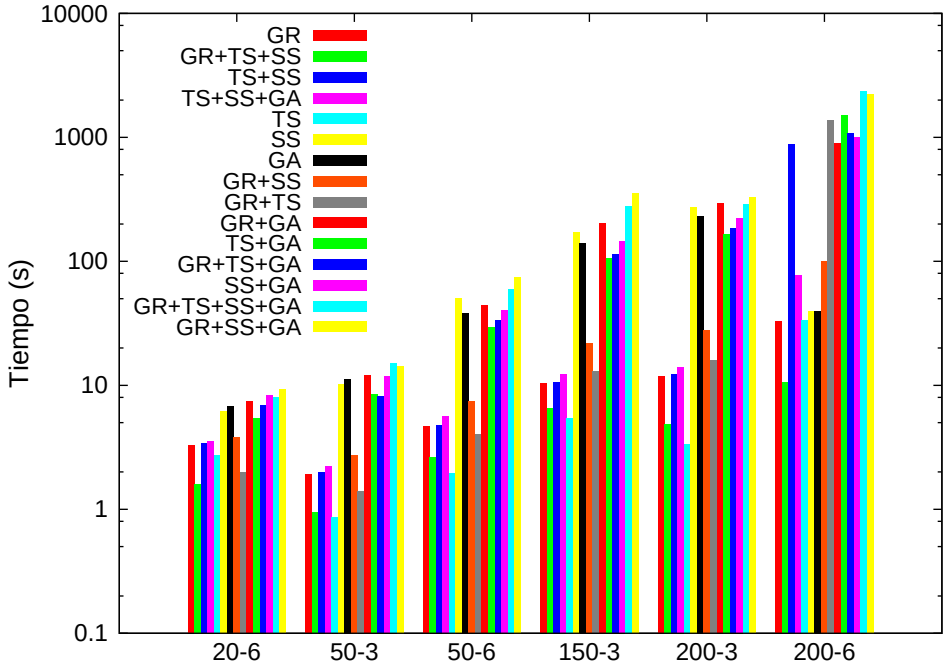


Figura 3.2: Tiempos de ejecución obtenidos para las diferentes combinaciones de metaheurísticas presentadas en la figura 3.1 para diferentes series del problema PCEPA.

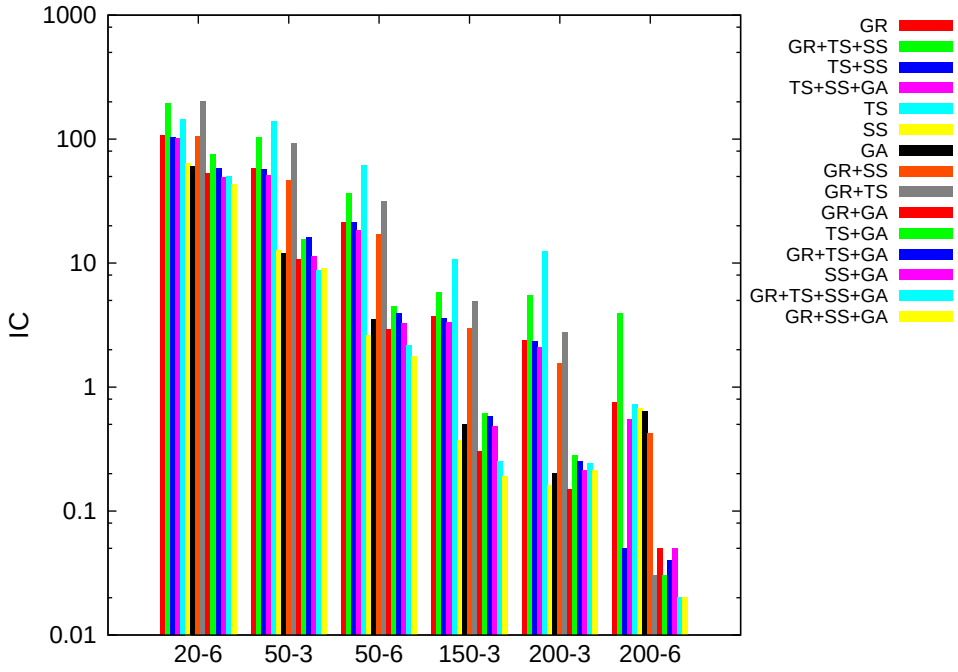


Figura 3.3: Indicador común ($IC = 10^6/(f \cdot t)$) para las diferentes combinaciones de metaheurísticas presentadas en las figuras 3.1 y 3.2 para diferentes series del problema PCEPA.

	GR	GR+TS+SS	TS+SS	TS+SS+GA	TS
20-6	3.256	1.603	3.423	3.520	2.705
50-3	1.907	0.946	1.977	2.234	0.857
50-6	4.707	2.615	4.740	5.626	1.951
150-3	10.357	6.540	10.602	12.276	5.417
200-3	11.838	4.837	12.265	13.949	3.365
200-6	32.571	10.549	877.538	77.555	33.749

	SS	GA	GR+SS	GR+TS	GR+GA
20-6	6.200	6.715	3.807	1.970	7.419
50-3	10.191	11.245	2.726	1.390	12.024
50-6	49.855	37.949	7.361	3.999	44.245
150-3	172.088	138.831	21.658	12.925	204.129
200-3	274.902	229.908	27.965	16.004	294.933
200-6	39.632	39.353	100.950	1360.390	894.242

	TS+GA	GR+TS+GA	SS+GA	GR+TS+SS+GA	GR+SS+GA
20-6	5.404	6.873	8.343	7.969	9.273
50-3	8.393	8.092	11.728	14.945	14.285
50-6	29.232	33.204	40.288	59.199	74.592
150-3	104.833	112.803	144.611	279.704	352.976
200-3	166.241	183.390	221.068	288.205	325.518
200-6	1515.577	1085.075	1008.692	2342.437	2243.617

Tabla 3.5: Tiempos de ejecución (en segundos) obtenidos para las diferentes combinaciones de metaheurísticas presentadas en la figura 3.1 para diferentes series del problema PCEPA.

	GR	GR+TS+SS	TS+SS	TS+SS+GA	TS
20-6	107.37	192.24	101.96	100.01	144.16
50-3	57.47	103.35	56.89	50.42	139.09
50-6	21.21	36.10	21.02	18.16	61.32
150-3	3.72	5.80	3.60	3.31	10.69
200-3	2.36	5.49	2.33	2.07	12.34
200-6	0.75	3.95	0.05	0.55	0.72

	SS	GA	GR+SS	GR+TS	GR+GA
20-6	63.97	60.46	104.05	198.99	52.91
50-3	12.59	11.92	46.37	92.29	10.63
50-6	2.58	3.48	16.89	31.48	2.89
150-3	0.37	0.50	2.95	4.91	0.30
200-3	0.16	0.20	1.55	2.73	0.15
200-6	0.67	0.64	0.42	0.03	0.05

	TS+GA	GR+TS+GA	SS+GA	GR+TS+SS+GA	GR+SS+GA
20-6	74.47	57.55	48.71	49.75	43.06
50-3	15.58	15.99	11.21	8.74	9.11
50-6	4.46	3.92	3.27	2.17	1.76
150-3	0.61	0.58	0.48	0.25	0.19
200-3	0.28	0.25	0.21	0.24	0.21
200-6	0.03	0.04	0.05	0.02	0.02

Tabla 3.6: Indicador común ($IC = 10^6/(f \cdot t)$) para las diferentes combinaciones de metaheurísticas presentadas en las figuras 3.1 y 3.2 para diferentes series del problema PCEPA.

En general se observa que todas las combinaciones en las que aparece SS+GA producen buenos resultados confirmándose la idoneidad de esta combinación metaheurística para esta instancia del problema.

La evolución del tiempo de ejecución en las sucesivas iteraciones al aplicar cada metaheurística al problema PCEPA de tamaño 50-6 se recoge en la figura 3.5. Se pueden comparar estos resultados con los de fitness de la figura 3.4. Cabe destacar que las metaheurísticas con resultados de fitness moderados y que consumen muchas iteraciones hasta el óptimo son las que arrojan tiempos de ejecución más bajos (TS, GA y GR+GA). Esto las hace especialmente atractivas para ser aplicadas a esta instancia de problema debido a su rapidez.

Se debe destacar que los resultados presentados en la figuras 3.4 y 3.5 son similares a los obtenidos para otros tamaños de problema, pero solo se ha elegido un tamaño por ser representativo del resto.

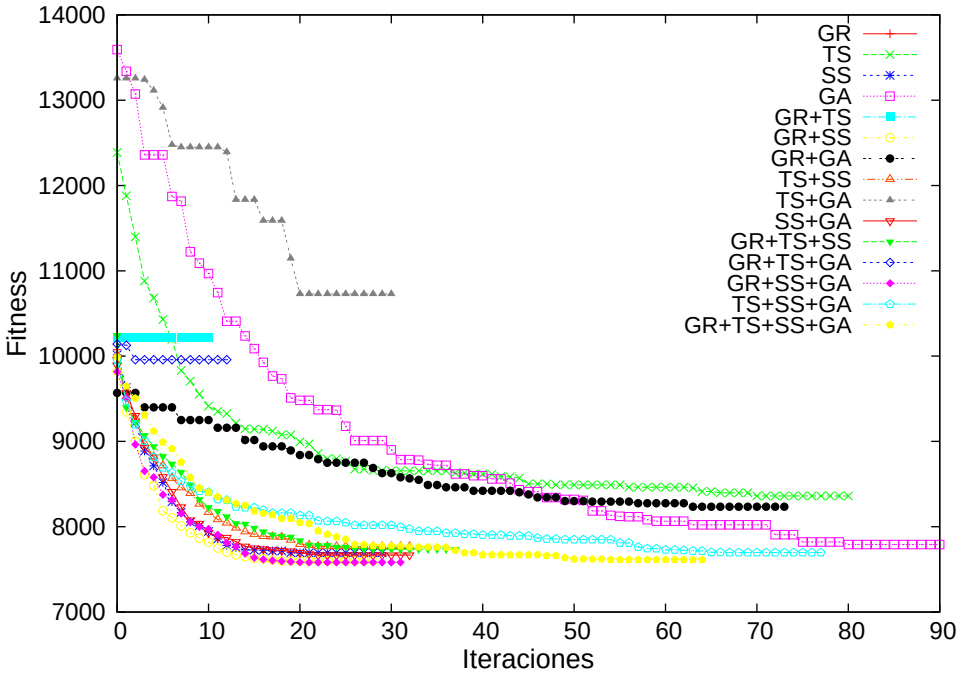


Figura 3.4: Valores de fitness obtenidos por diferentes metaheurísticas en iteraciones sucesivas para el tamaño de problema PCEPA 50-6.

Finalmente, como ilustración complementaria, en la figura 3.6 se muestra un ejemplo de resultado obtenido al aplicar una metaheurística de tamaño reducido a una instancia de problema PCEPA de tamaño pequeño (5 bombas y 3 rangos horarios). Se pueden ver primeramente las variables utilizadas, con valores para cada pozo en explotación y con las tolerancias de algunos valores entre paréntesis. A continuación, se muestran los 20 valores de los parámetros de la metaheurística aplicada, que puede considerarse un algoritmo genético de tamaño reducido. Finalmente, aparece la solución alcanzada (combinación de pozos óptima y fitness asociado), donde

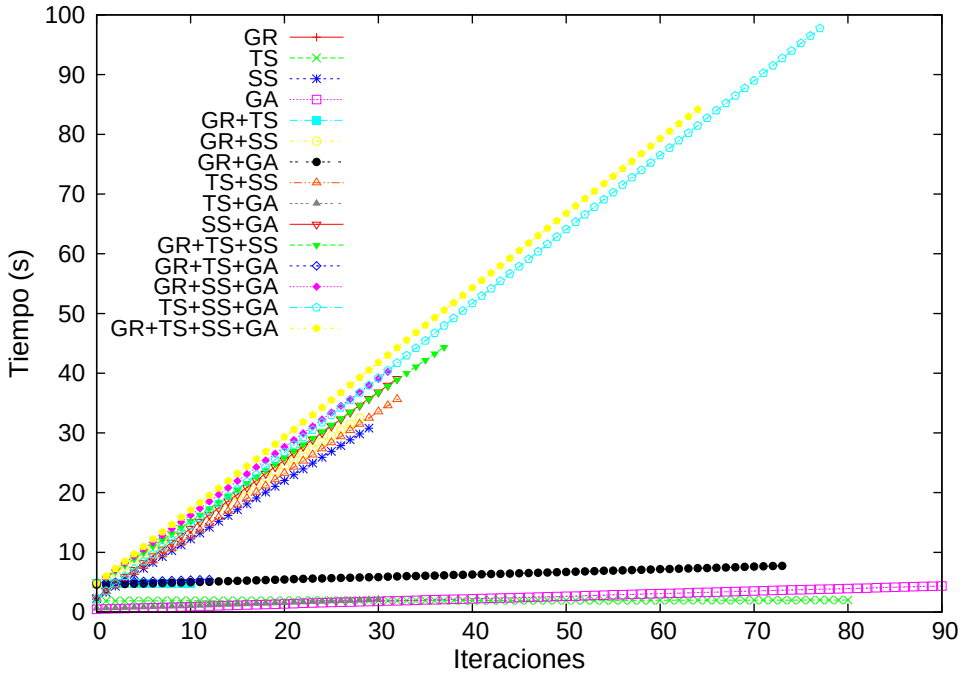


Figura 3.5: Tiempos de ejecución obtenidos por las diferentes metaheurísticas de la figura 3.4 en iteraciones sucesivas para la serie S5 del problema PCEPA.

se aprecia de manera clara la codificación empleada: los unos representan las bombas en marcha en cada tramo horario (3 tramos en este caso presentados con una ligera separación) y los ceros, las apagadas. Además se presentan los valores de las cinco restricciones impuestas, que se cumplen en todo momento: el volumen total se encuentra entre 10000 y 20000 m^3, el caudal total en cada franja horaria es siempre superior a 200 m^3/h, los volúmenes explotados en cada pozo no sobrepasan los máximos establecidos, las conductividades totales en cada tramo considerado son siempre inferiores a 2500 μS/cm y el estatus de los cinco pozos indica que están todos operativos (unos) porque sus niveles dinámicos no son en ningún caso inferiores al mínimo nivel establecido.

3.4. Aplicación del esquema metaheurístico al problema PCOCI

Se puede también contrastar la eficacia del esquema parametrizado de metaheurísticas aplicando esta técnica al problema de optimización de las constantes cinéticas de una reacción química (PCOCI).

Paralelamente a la obtención, con métodos metaheurísticos, de la configuración óptima de los parámetros cinéticos de una reacción química heterogénea, se ha completado la simulación del proceso de disolución de una pastilla antiácido y se ha

INSTANCIA PCEPA

Bombas: 5
Rangos Horarios: 3
Volumen total acumulado diario (m^3): 10000 (+10000)
Caudal total mínimo (m^3/h): 200
Caudales (m^3/h): 258 142 243 247 277
Tarifas (€/(Kw h)): 0.168 0.112 0.056
Potencias (Kw): 103 154 314 132 234
Conductividad máxima permitida (µS/cm): 2500
Conductividades (µS/cm): 1484 1923 1157 1588 2861
Volúmenes máximos concedidos diarios (m^3): 9245 6721 7455 2274 6462
Profundidades nivel dinámico (m): 152 186 170 270 173
Profundidades nivel dinámico máximas (m): 290 262 260 349 271 (+3)

METAHEURÍSTICA APLICADA (parámetros)
---
6 6 0 0 0 1000 10 6 0 3 0 0 0 0 0 10 5 0 3 0

SOLUCIÓN

Combinación de pozos: 0 1 1 0 1 0 1 1 0 1 1 1 1 1 0
Ce (€): 1887.42

RESTRICCIONES

Volumen total (m^3): 17712
Caudal franjas (m^3/h): 662 662 890
Volúmenes explotados bombas (m^3): 2064 3408 5832 1976 4432
Conductividad franjas (µS/cm): 2034.31 2034.31 1493.62
Estatus pozos: 1 1 1 1 1

Figura 3.6: Ejemplo de solución metaheurística para una instancia reducida de prueba del problema PCEPA.

determinado la evolución de las concentraciones de las especies químicas a lo largo del tiempo. Las ejecuciones se han llevado a cabo adaptando el esquema parametrizado de metaheurísticas a nuestro problema. Las combinaciones de las constantes cinéticas usadas en el estudio se han generado aleatoriamente con valores de k_i y n_i en los rangos $[10^{-10}, 10]$ y $[0, 5]$ respectivamente (ver [54]). Los valores utilizados de cada parámetro para una combinación dada de metaheurísticas se muestran en la tabla 3.7, donde las condiciones de finalización tienen los valores típicos $NMIFin$=1000, $NIRFin = 10$ y han sido omitidas. Se han elegido valores razonables de los parámetros después de una serie de pruebas con varios valores. Las combinaciones mostradas son aquellas para las cuales el fitness obtenido es aceptable sin asumir un coste computacional excesivo. Así, valores altos de los parámetros implicarían tiempos de ejecución elevados, lo que haría mucho más lento el proceso de experimentación, con resultados similares a los obtenidos con parámetros de valores moderados. Por ejemplo, las poblaciones iniciales en el problema PCEPA para las cuatro metaheurísticas puras (GR, TS, SS y GA) eran respectivamente de 200, 200, 100 y 100. En el caso de PCOCI los valores son 50, 50, 20 y 20. Igualmente disminuyen proporcionalmente los valores del conjunto de referencia en todas las metaheurísticas, el número de elementos mejores a incluir y las memorias tabú, como también son proporcionalmente inferiores los parámetros de la combinación $NMMCom$, $NMPCom$ y $NPPCom$ con valores de 15, 20 y 15 para el caso, por ejemplo, de la Búsqueda Dispersa aplicada al problema PCOCI cuando para PCEPA eran respectivamente 90, 100 y 90.

Las metaheurísticas se aplicaron a seis series de datos experimentales con la configuración de variables químicas presentada en la tabla 3.8, donde N es el número de puntos experimentales de tiempo y pH a integrar, MT (g) es la masa total de la pastilla que se está disolviendo en ácido, $[HAc]_0$ (mol/L) es la concentración inicial de ácido acético en cada serie, y V (mL) es el volumen de la disolución. Otras variables comunes a todas las series son: MA (g) = 0.68 que es la masa activa de carbonato de calcio en la pastilla, a_0 (cm^2) = 7.11 es el área inicial de la pastilla, $[H_2CO_3^*]$ (mol/L) = 10^{-5} es la concentración total de las especies carbonatadas en la disolución. Y finalmente, las constantes de equilibrio relacionadas [92] tienen los valores: $K_a = 10^{-4.76}$, $K_a^{'} = 10^{-6.35}$, $K_H = 10^{-1.5}$.

En muchos casos el valor final de la función objetivo (ecuación 2.23) es muy similar (figura 3.7 y tabla equivalente 3.9). Cuanto más bajo el valor, mejor es el resultado. Podemos ver que:

- En general, las combinaciones de metaheurísticas producen buenos valores de fitness, destacando la metaheurística GA y sus combinaciones como las más adecuadas para este problema de optimización.

- TS, SS y las hibridaciones de SS producen resultados que son intermedios en muchos casos. En el caso de SS y sus combinaciones, puede ser debido al amplio rango de búsqueda gestionado para las constantes cinéticas (especialmente k_i),

		GR	TS	SS	GA	GR+TS	GR+SS	GR+GA	TS+SS	TS+GA
Ini	NEIIni	50	50	20	20	50	50	50	20	20
	NEFIni	1	1	10	20	1	10	20	10	20
	PEMIni	100	100	100	0	100	100	100	100	0
	IMEIni	15	15	15	0	15	15	15	15	0
	MCPIni	0	2	0	0	2	0	0	2	0
Sel	NEMSel	0	1	5	20	0	5	20	5	20
	NEPSel	0	0	5	0	0	5	0	5	0
Com	NMMCom	0	0	15	10	0	15	10	15	10
	NMPCom	0	0	20	0	0	20	0	20	0
	NPPCom	0	0	15	0	0	15	0	15	0
Mej	PEMMej	0	100	100	0	0	100	0	100	0
	IMEMej	0	5	5	0	0	5	0	5	0
	MCMMej	0	2	0	0	0	0	0	2	0
	PEDMej	0	0	0	20	0	0	20	0	20
	IDEMej	0	0	0	5	0	0	5	0	2
	MCDMej	0	0	0	0	0	0	0	0	10
Inc	NEMInc	0	1	5	20	0	5	20	5	20
	MLPInc	0	3	0	0	0	0	0	3	3

	SS+GA	GR+TS+SS	GR+TS+GA	GR+SS+GA	TS+SS+GA	GR+TS+SS+GA
NEIIni	20	50	50	50	20	50
NEFIni	15	10	20	15	15	15
PEMIni	100	100	100	100	100	100
IMEIni	15	15	15	15	15	15
MCPIni	0	2	2	0	2	2
NEMSel	8	5	20	8	8	8
NEPSel	7	5	0	7	7	7
NMMCom	15	15	10	15	15	15
NMPCom	20	20	0	20	20	20
NPPCom	15	15	0	15	15	15
PEMMej	100	100	0	100	100	100
IMEMej	5	5	0	5	5	5
MCMMej	0	2	0	0	2	2
PEDMej	20	0	20	20	20	20
IDEMej	5	0	5	5	5	5
MCDMej	0	0	2	0	2	2
NEMInc	8	5	20	8	8	8
MLPInc	0	3	3	0	3	3

Tabla 3.7: Valores de los parámetros para diferentes combinaciones de metaheurísticas aplicadas al problema PCOCI.

Serie	S1	S2	S3	S4	S5	S6
N	70	50	34	77	81	69
MT	1.33	1.33	1.31	1.32	1.32	1.30
$[HAc]_0 \cdot 10^2$	3.98	3.31	2.39	2.18	3.01	1.99
V	250	250	250	100	150	200

Tabla 3.8: Valores de las variables del problema PCOCI consideradas en los experimentos.

que hace insuficiente el rango de búsqueda local para cubrir satisfactoriamente el espacio de soluciones.

- Con el métodos GRASP los resultados son peores en muchos casos, pero su hibridación con otras técnicas produce resultados satisfactorios. Las hibridaciones de TS también producen resultados de baja calidad.

En el último conjunto de datos de la figura 3.7 (última fila de la tabla 3.9) podemos ver una media de los resultados de cada metaheurística. Se ha considerado hacer la media en el problema PCOCI porque, a diferencia de PCEPA, los valores obtenidos para las distintas instancias de problema son del mismo orden de magnitud, lo que hace que las medias reflejen de forma adecuada las contribuciones de cada instancia. Es importante destacar la desviación más pronunciada respecto a la media de las series S4 y S5, lo que indica que el ajuste del modelo a los datos experimentales es menos preciso en estos casos, posiblemente debido a que las condiciones experimentales no se ajustaban totalmente a los requerimientos del modelo teórico.

Vamos a considerar aquí también los tiempos de ejecución de cada metaheurística como criterio a tener en cuenta. Así, en la figura 3.8 y la tabla 3.10 podemos ver los tiempos de ejecución empleados por las diferentes combinaciones de metaheurísticas presentadas en la figura 3.7 para las seis series del problema PCOCI. Se puede apreciar, como en el caso de PCEPA, que las metaheurísticas que alcanzan valores de fitness más bajos son las que consumen un mayor tiempo en su ejecución (aunque, como dijimos, esto puede no suceder a veces si los individuos no son fácilmente factibilizables).

Para tener en cuenta tanto la bondad del resultado de fitness como el tiempo empleado en alcanzarlo por las metaheurísticas, vamos a introducir un indicador común similar al introducido para PCEPA que definiremos $IC = 10/(f \cdot t)$.

En la figura 3.9 y tabla 3.11 podemos ver los valores del IC para las diferentes combinaciones de metaheurísticas que resuelven las seis instancias del problema PCOCI. Se aprecia que, según este criterio, los mejores resultados (valores mayores) corresponden a GR, TS+SS, GR+TS, GR+TS+GA y GR+TS+SS. Se puede ver que, bajo el criterio del indicador común, ya no son el genético y sus combinaciones los mejores algoritmos puesto que, aunque ofrecen resultados muy buenos de fitness,

	GR	TS+SS	GR+TS	GR+TS+GA	TS+GA
S1	2.828	2.713	2.418	3.032	1.860
S2	2.290	2.504	2.316	2.034	2.004
S3	2.244	2.075	1.899	1.317	2.001
S4	6.911	6.467	5.916	3.573	4.035
S5	3.815	3.682	3.377	2.308	2.680
S6	2.700	2.322	2.504	2.373	1.989
Media	3.465	3.294	3.072	2.439	2.428

	GR+SS	TS+SS+GA	SS+GA	GR+TS+SS	SS
S1	2.346	1.681	1.786	1.703	2.162
S2	1.773	1.947	1.653	2.025	1.513
S3	1.789	1.558	1.402	1.759	1.273
S4	3.619	3.304	2.817	1.420	1.331
S5	2.394	2.270	1.957	1.735	1.372
S6	1.880	1.714	1.676	2.226	2.121
Media	2.300	2.079	1.882	1.811	1.629

	TS	GR+GA	GA	GR+TS+SS+GA	GR+SS+GA
S1	1.603	1.533	1.295	1.193	1.238
S2	1.491	1.493	1.493	1.417	1.486
S3	1.312	1.480	1.092	1.163	1.095
S4	1.948	0.943	0.941	0.928	0.919
S5	1.584	1.305	1.175	1.169	1.167
S6	1.656	1.305	1.308	1.323	1.323
Media	1.599	1.343	1.217	1.199	1.205

Tabla 3.9: Valor de la función de fitness obtenida para diferentes combinaciones de metaheurísticas para diferentes series del problema PCOCI.

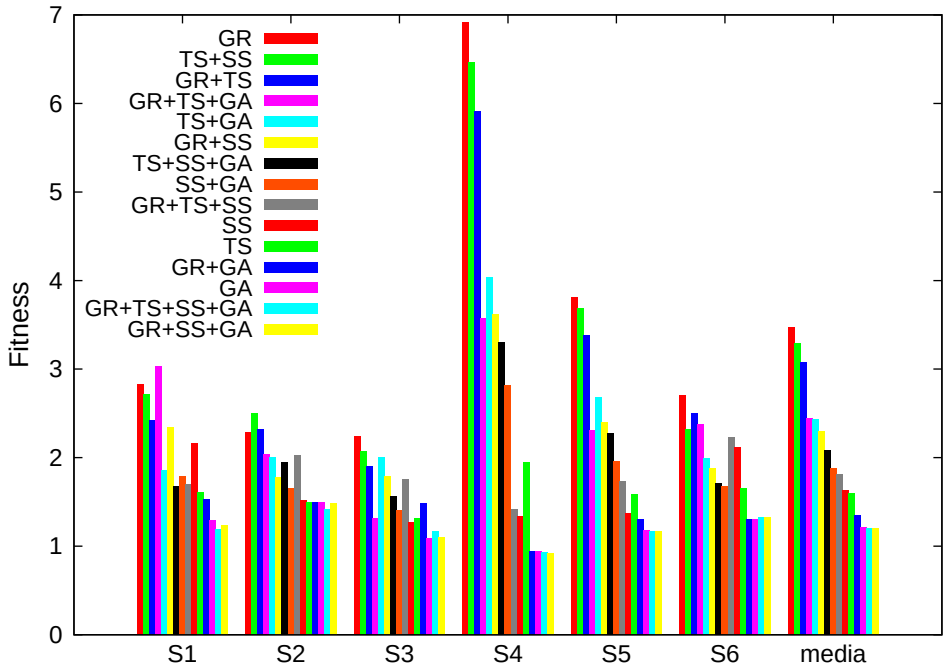


Figura 3.7: Valor de la función de fitness obtenida para diferentes combinaciones de metaheurísticas para diferentes series del problema PCOCI.

el tiempo empleado es elevado. Por tanto, vemos que existen metaheurísticas más rápidas con una bondad de la solución aceptable.

La figura 3.10 muestra el valor de la función objetivo en sucesivas iteraciones para cada método, para el problema descrito en la serie S5. Se puede ver que las metaheurísticas que inlcuyen GA arrojan valores de fitness más bajos y normalmente más iteraciones hasta el óptimo. Por otro lado, tenemos GRASP, TS, SS y sus combinaciones, con valores de fitness más altos y, finalmente, GR+TS+GA con valores de fitness intermedios. En GRASP solo tenemos una iteración, representada con un punto en el gráfico.

Podemos ver la evolución del tiempo de ejecución en las sucesivas iteraciones al aplicar cada metaheurística al problema S5 (figura 3.11), y se pueden comparar estos resultados con los de fitness de la figura 3.10. Se ve aquí claramente como las metaheurísticas con mejor función de bondad, SS+GA y GR+TS+SS+GA, son las que emplean mayores tiempos en alcanzar la solución. Hay que destacar la metaheurística TS+GA, pues es una metaheurística que, aunque requiere de muchas iteraciones, alcanza buenos resultados de fitness con tiempos de ejecución bajos. Esto la hace especialmente atractiva para ser aplicada a esta instancia del problema pues posee un IC alto.

Como en el problema PCEPA, se debe destacar que los resultados presentados en las figuras 3.10 y 3.11 son similares a otros obtenidos para otras series, habiéndose elegido sólo una por ser representativa del resto.

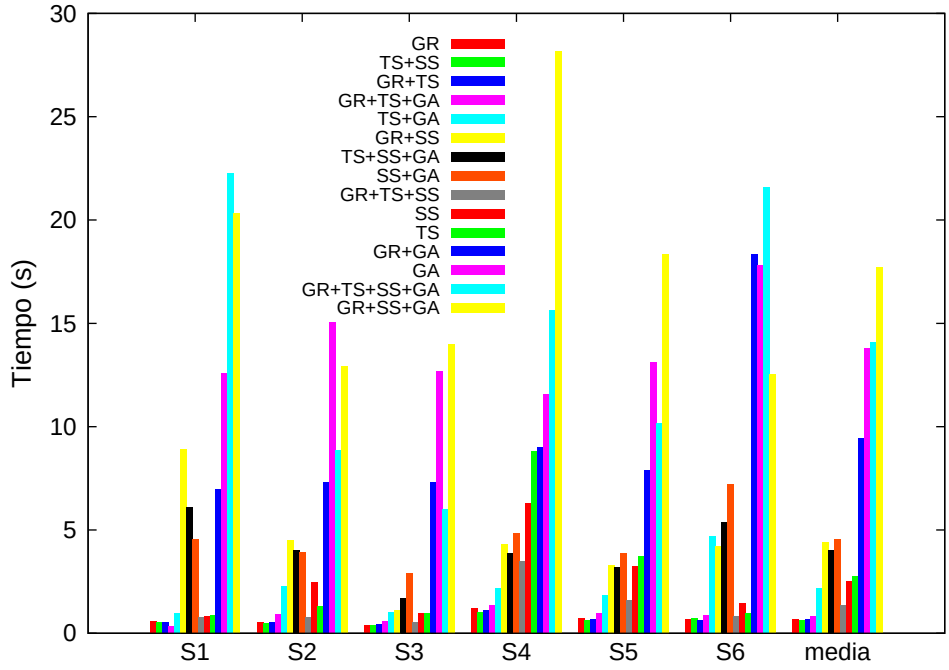


Figura 3.8: Tiempos de ejecución obtenidos para las diferentes combinaciones de metaheurísticas presentadas en la figura 3.7 para diferentes series del problema PCOCI.

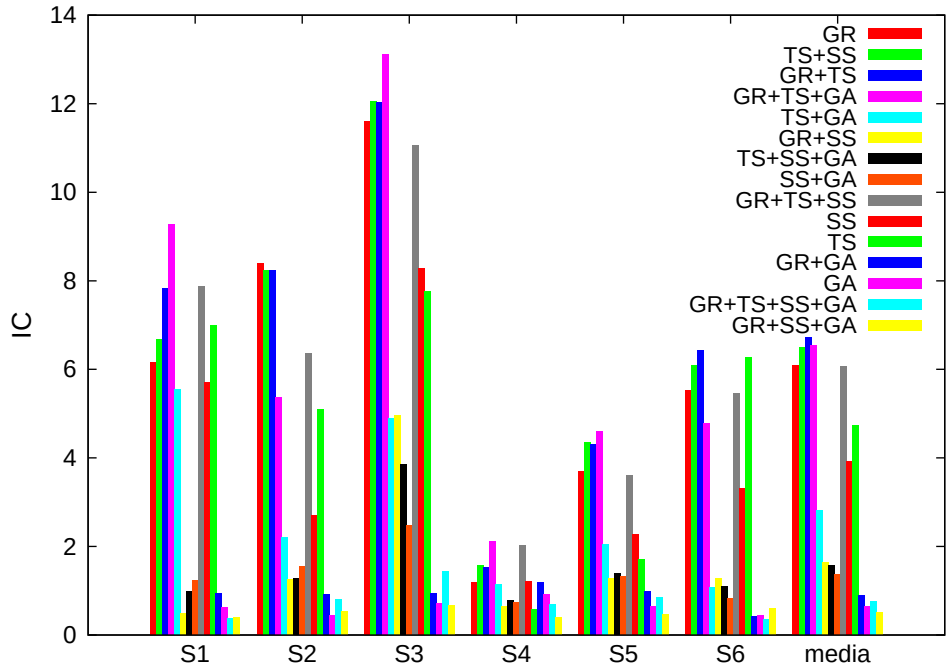


Figura 3.9: Indicador común ($IC = 10/(f \cdot t)$) para las diferentes combinaciones de metaheurísticas presentadas en las figuras 3.7 y 3.8 para diferentes series del problema PCOCI.

	GR	TS+SS	GR+TS	GR+TS+GA	TS+GA
S1	0.575	0.552	0.528	0.356	0.970
S2	0.520	0.486	0.525	0.919	2.270
S3	0.384	0.400	0.438	0.579	1.021
S4	1.226	0.992	1.106	1.327	2.190
S5	0.710	0.626	0.690	0.942	1.827
S6	0.672	0.708	0.621	0.882	4.711
Media	0.681	0.627	0.651	0.834	2.165

	GR+SS	TS+SS+GA	SS+GA	GR+TS+SS	SS
S1	8.894	6.098	4.532	0.745	0.812
S2	4.482	4.033	3.892	0.777	2.444
S3	1.130	1.671	2.897	0.515	0.948
S4	4.290	3.852	4.840	3.497	6.292
S5	3.301	3.186	3.876	1.596	3.228
S6	4.207	5.346	7.185	0.823	1.423
Media	4.384	4.031	4.537	1.326	2.525

	TS	GR+GA	GA	GR+TS+SS+GA	GR+SS+GA
S1	0.891	6.973	12.589	22.264	20.315
S2	1.316	7.305	15.048	8.877	12.911
S3	0.982	7.284	12.696	5.980	13.981
S4	8.799	8.991	11.559	15.616	28.146
S5	3.699	7.860	13.101	10.158	18.346
S6	0.964	18.321	17.795	21.561	12.529
Media	2.775	9.456	13.798	14.076	17.704

Tabla 3.10: Tiempos de ejecución obtenidos para las diferentes combinaciones de metaheurísticas presentadas en la figura 3.7 para diferentes series del problema PCOCI.

	GR	TS+SS	GR+TS	GR+TS+GA	TS+GA
S1	6.148	6.673	7.827	9.268	5.542
S2	8.398	8.220	8.225	5.352	2.198
S3	11.597	12.049	12.017	13.112	4.893
S4	1.181	1.559	1.529	2.110	1.132
S5	3.692	4.339	4.294	4.602	2.042
S6	5.513	6.086	6.429	4.778	1.067
Media	6.088	6.488	6.720	6.537	2.813

	GR+SS	TS+SS+GA	SS+GA	GR+TS+SS	SS
S1	0.479	0.976	1.236	7.877	5.694
S2	1.258	1.273	1.554	6.356	2.705
S3	4.947	3.841	2.462	11.046	8.284
S4	0.644	0.786	0.734	2.014	1.195
S5	1.266	1.383	1.318	3.612	2.258
S6	1.264	1.091	0.831	5.457	3.313
Media	1.643	1.558	1.356	6.060	3.908

	TS	GR+GA	GA	GR+TS+SS+GA	GR+SS+GA
S1	6.997	0.936	0.613	0.377	0.398
S2	5.095	0.917	0.445	0.795	0.521
S3	7.759	0.928	0.721	1.437	0.653
S4	0.583	1.180	0.920	0.690	0.386
S5	1.707	0.975	0.649	0.842	0.467
S6	6.267	0.418	0.430	0.351	0.603
Media	4.735	0.892	0.630	0.749	0.505

Tabla 3.11: Indicador común ($IC = 10/(f \cdot t)$) para las diferentes combinaciones de metaheurísticas presentadas en las figuras 3.7 y 3.8 para diferentes series del problema PCOCI.

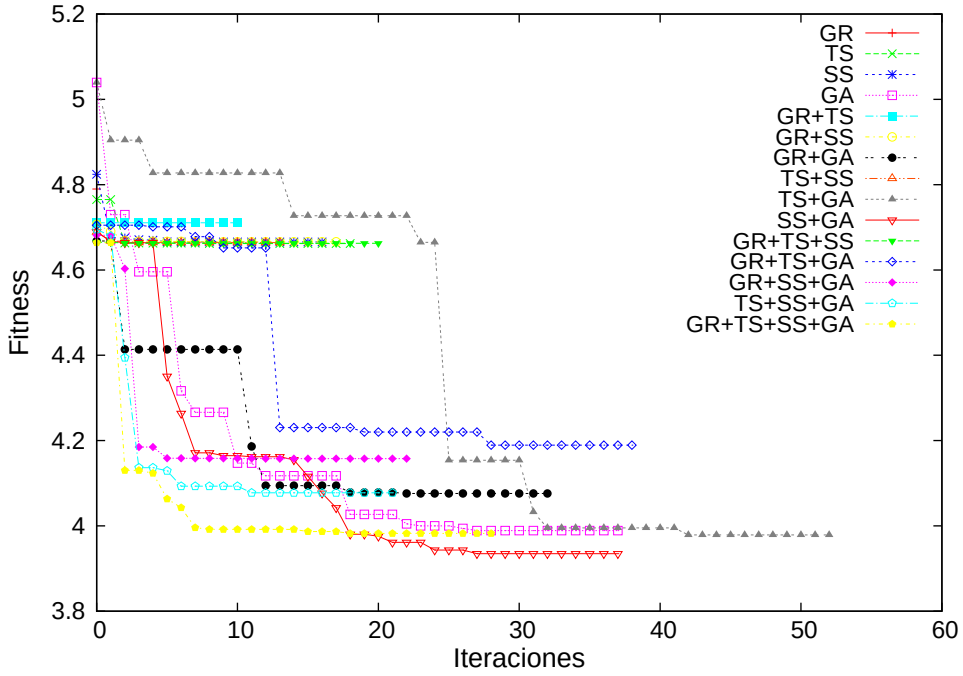


Figura 3.10: Valores de fitness obtenidos por diferentes metaheurísticas en iteraciones sucesivas para la serie S5 del problema PCOCI.

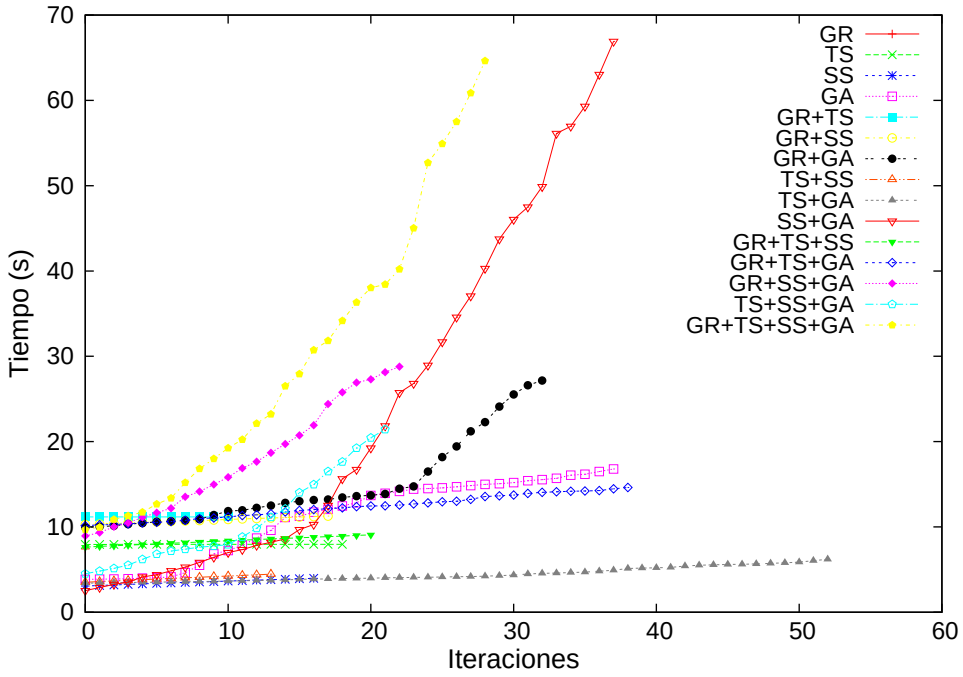
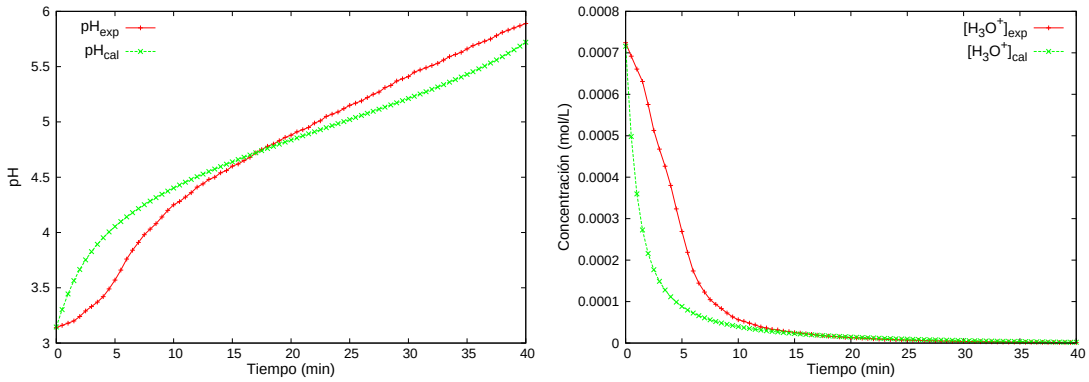


Figura 3.11: Tiempos de ejecución obtenidos por las diferentes metaheurísticas de la figura 3.10 en iteraciones sucesivas para la serie S5 del problema PCOCI.



(a) (b)

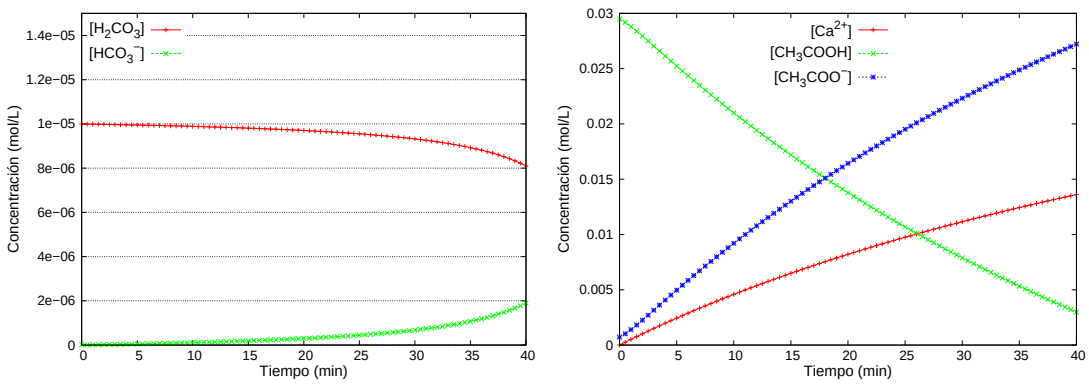
Figura 3.12: Evolución con el tiempo de (a) el pH y (b) la concentración de protones. Valores experimentales y calculados.

Finalmente, podemos contrastar la bondad de los resultados alcanzados con la simulación de la evolución temporal del sistema químico. Así, en las figuras 3.12 y 3.13 se puede ver la evolución del pH experimental y calculado y de la concentración de las diferentes especies químicas presentes en el modelo resuelto con las constantes cinéticas óptimas para la serie S5. Los valores finales son: $k_1 = 6.83 \cdot 10^{-7}$, $k_2 = 5.09 \cdot 10^{-4}$, $k_3 = 1.99 \cdot 10^{-6}$, $n_1 = 3.11$, $n_2 = 6.19 \cdot 10^{-3}$, $n_3 = 1.42$ y $n_4 = 0.339$. El pH calculado en la simulación no está muy lejos del experimental sobre todo a partir de los diez minutos, y la predicción dada por el modelo cinético es aceptable. La concentración de HCO_3^- se mantiene en todo momento por debajo de la de H_2CO_3, lo que es algo esperado en el rango de pH considerado para la serie S5 (siempre por debajo de 6). Se puede apreciar como la concentración de calcio aumenta conforme la pastilla se disuelve. A su vez, la concentración de acético disminuye a la misma velocidad con la que aumenta la concentración de ion acetato, lo que es de esperar si el modelo es correcto.

Por supuesto, el método o combinación preferido depende de la implementación particular de las funciones básicas que estemos usando. Sin embargo, no estamos interesados en obtener la mejor metaheurística, si no en la aplicación de la metodología para desarrollar metaheurísticas de manera sencilla y experimentar con familias de metaheurísticas.

Si comparamos los resultados arrojados al aplicar el esquema metaheurístico a los dos problemas de optimización considerados, vemos que el método selecciona la mejor metaheurística para cada problema y que normalmente suele ser diferente. Si la hibridación Búsqueda Dispersa + Algoritmo Genético y sus combinaciones con otras metaheurísticas eran las que mejor resolvían el problema PCEPA desde el punto de vista del fitness, en el caso de PCOCI, es el Algoritmo Genético y sus combinaciones los que proporcionan una mejor solución.

Aunque se ha visto que las metaheurísticas obtenidas para resolver cada instancia



(a) (b)

Figura 3.13: Evolución de (a) la concentración de especies carbonatadas con el tiempo y (b) la concentración de especies acéticas y calcio. Valores calculados en la simulación de la reacción.

concreta del problema ofrecen buenos resultados, es posible que no sean tan eficaces si las aplicamos a otras instancias diferentes de un mismo problema. Para evitar estas dependencias del problema, en el capítulo siguiente, se introducirá un nuevo algoritmo a un nivel superior de abstracción, que maneje a la vez varias instancias del problema, encontrando metaheurísticas eficaces para un conjunto de instancias sin perder demasiada calidad en los resultados.

3.5. Conclusiones

Se ha aplicado un esquema unificado parametrizado de metaheurísticas a dos problemas de optimización propuestos de forma secuencial. Se ha detallado la estructura interna del esquema, la funciones básicas que lo forman, así como los parámetros metaheurísticos con los que se invocan. Se ha presentado también un ejemplo de inclusión de una nueva metaheurística, Búsqueda Tabú, en el esquema y la facilidad con la que se aplica con la introducción de cuatro nuevos parámetros.

Se presentan resultados de fitness alcanzados por las diferentes metaheurísticas y combinaciones analizadas para cada problema, así como resultados de los tiempos de ejecución empleados. Desde el punto de vista de la función de bondad, para la resolución del problema PCEPA, se recomienda el uso de la hibridación metaheurística SS+GA y sus combinaciones con otras metaheurísticas pues son las que ofrecen valores más bajos de fitness. Sin embargo, se ha visto que la metaheurística GA y sus combinaciones son las más adecuadas para la resolución del problema PCOCI. En general, TS eleva el número de iteraciones para alcanzar el óptimo pero su combinación con otras metaheurísticas ha resultado ser satisfactoria. Desde el punto de vista del tiempo de ejecución, se ha comprobado que las metaheurísticas que ofrecen mejores resultados de función de bondad son las que invierten un mayor tiempo en

alcanzar el óptimo (aunque esto podría no ser así como hemos indicado previamente). Así, se ha introducido un indicador común para tener en cuenta ambos factores, lo que hace variar las conclusiones obtenidas. Teniendo en cuenta esto, los mejores resultados (*IC* más alto), para PCEPA, son: TS, GR+TS y GR+TS+SS. Para el problema PCOCI los mejores resultados corresponden a las metaheurísticas GR, TS+SS, GR+TS, GR+TS+GA y GR+TS+SS.

Hemos visto que el esquema metaheurístico selecciona la mejor metaheurística para cada problema concreto y que, normalmente, los algoritmos elegidos suelen ser diferentes. Además, se ha comprobado como el indicador de bondad elegido (fitness o *IC*) influye en los resultados finales alcanzados.

Aunque las metaheurísticas aplicadas a los problemas ofrecen buenos resultados, se muestran normalmente eficaces solo para las instancias concretas en las que se usaron. En el siguiente capítulo se presentará un nuevo algoritmo hiperheurístico más abstracto que permitirá encontrar buenas metaheurísticas para conjuntos de varias instancias, evitando dependencias del problema.

Capítulo 4

Hiperheurísticas basadas en esquemas metaheurísticos parametrizados

Se va a desarrollar en este capítulo la metodología hiperheurística basada en esquemas parametrizados de metaheurísticas. Al igual que en el caso de las metaheurísticas, la elección de los parámetros hiperheurísticos determinará qué hiperheurística se aplicará en cada caso. Puesto que el coste computacional de una hiperheurística es mucho mayor, se elegirán configuraciones de parámetros con valores moderados, buscando en todo momento un compromiso entre calidad de la solución alcanzada y tiempo de ejecución para obtenerla. Se utilizarán varias configuraciones hiperheurísticas para encontrar la mejor metaheurística o combinación para cada problema de optimización considerado. Así pues, podemos considerar que tenemos aquí un tercer problema de optimización a un nivel de abstracción superior en el sentido de que se van a optimizar las configuraciones paramétricas de las metaheurísticas aplicables a los problemas de optimización dados: PCEPA y PCOCI.

Se pretende hacer un estudio exhaustivo de la utilidad y aplicabilidad de las hiperheurísticas implementadas mediante esquemas metaheurísticos parametrizados, por lo que se compararán los resultados obtenidos con las diferentes configuraciones hiperheurísticas y los resultados alcanzados con diferentes metodologías para el cálculo del fitness. Se utilizará un criterio basado en el fitness para seleccionar automáticamente metaheurísticas buenas a partir de ciertas instancias de entrenamiento. Además, para un mayor contraste del método, se compararán los resultados de las hiperheurísticas con los obtenidos al aplicar directamente metaheurísticas híbridas satisfactorias a los problemas de optimización. Finalmente, para cada problema, se realizará un estudio estadístico para comparar si existen diferencias significativas en los resultados alcanzados al aplicar los diferentes algoritmos meta e hiperheurísticos agrupados por categorías según la manera en que han sido obtenidos.

4.1. Hiperheurísticas basadas en esquemas metaheurísticos parametrizados

El esquema parametrizado de metaheurísticas que facilita el desarrollo de metaheurísticas y su aplicación se usa aquí para implementar hiperheurísticas, las cuales deberán buscar en el espacio de metaheurísticas determinadas por los valores de los parámetros metaheurísticos. Al desarrollar las hiperheurísticas, se introducirán parámetros hiperheurísticos en el esquema metaheurístico.

En la hiperheurística, usando la notación de algoritmos evolutivos, un individuo o elemento se representa mediante un vector de enteros $ParamMetaheur$ que codifica el conjunto de parámetros que caracteriza una metaheurística utilizando el esquema del Algoritmo 3. El número y significado de los parámetros queda determinado por las metaheurísticas básicas integradas en el esquema, la implementación de las funciones básicas y los parámetros que pueden ser variados en las funciones. Teniendo en cuenta las funciones consideradas en el capítulo anterior se tiene un total de 20 parámetros metaheurísticos, por lo que un elemento en la hiperheurística tendrá 20 componentes, con valores en rangos que dependen de los propios valores de los parámetros y, además, de algunas restricciones impuestas a la hiperheurística para reducir el espacio de búsqueda. Por ejemplo, no existe límite en el tamaño de la población inicial en la metaheurística ($NEIIni$), pero se debería imponer un límite superior, y este límite puede ser pequeño con el fin de reducir el tiempo de ejecución de la hiperheurística. Los elementos y otros aspectos de la hiperheurística se explicarán en las secciones siguientes con ayuda de los casos de prueba establecidos.

El conjunto de individuos constituye el conjunto de referencia, lo que significa un conjunto de metaheurísticas, con cada metaheurística siendo la combinación/hibridación de metaheurísticas básicas dadas por los valores recogidos en $ParamMetaheur$.

El valor de fitness en la hiperheurística para un elemento $ParamMetaheur$ se basa en el obtenido cuando la metaheurística con los parámetros en $ParamMetaheur$ se aplica al problema para el cual se está obteniendo dicha metaheurística. El objetivo es minimizar (o maximizar) la función de fitness y por tanto obtener la combinación de parámetros metaheurísticos (la hibridación preferida de metaheurísticas básicas) que proporciona la mejor función de fitness para diferentes instancias del problema en cuestión. Aunque también podría optimizarse el tiempo simultáneamente, se considera solo a posteriori en forma de indicador común junto al fitness. Hay diferentes posibilidades para la función de fitness:

- Fit1P1E (Fitness con aplicación a una instancia de problema y una ejecución): Se puede obtener aplicando cada metaheurística a la misma instancia de problema. El inconveniente de esta aproximación es que la solución puede ser dependiente de la instancia utilizada.

- FitVPVE1 (Fitness con aplicación a varias instancias de problema, y fitness obtenido como la media de los fitness individuales alcanzados): Para evitar

dependencias con el problema, cada metaheurística se podría aplicar a un número de instancias del problema, y la función de fitness obtenerse con alguna combinación del fitness para las diferentes instancias. Esto significa un coste computacional más alto, lo que no es deseable dado el elevado coste de la hiperheurística, la cual se basa en aplicaciones sucesivas de metaheurísticas.

- FitVPVE2 (Fitness con aplicación a varias instancias de problema, una ejecución de la hiperheurística para cada instancia del problema y los parámetros metaheurísticos obtenidos como la media de los parámetros para ejecuciones diferentes): Como alternativa, la hiperheurística se puede aplicar varias veces con el fitness obtenido cada vez para una instancia diferente, y los valores de los parámetros metaheurísticos se pueden obtener como una media de aquellos en las distintas aplicaciones de la hiperheurística. Se puede reducir la dependencia del problema, pero el tiempo de ejecución de la hiperheurística se incrementa proporcionalmente al número de veces que ésta se aplica. Además, una media directa de los valores de los parámetros puede producir valores no válidos, que se podrían corregir para eliminar incompatibilidades entre parámetros, y se pueden perder posibles correlaciones entre los parámetros en la combinación.

- FitVP1E (Fitness con aplicación a varias instancias de problema en una ejecución): Otro modo de reducir la dependencia de la instancia de problema y reducir también el tiempo de ejecución es aplicar las metaheurísticas a instancias diferentes en cada iteración del algoritmo, con lo que en una misma ejecución de la hiperheurística se aplican varias instancias diferentes. El cambio de instancia se puede hacer en cada iteración o grupo de iteraciones y se pueden usar distintas instancias del mismo tamaño o de tamaños de problema diferentes.

Algunas de estas posibilidades y algunas variantes se analizan en la sección de experimentos.

La hiperheurística usa el esquema metaheurístico unificado del Algoritmo 3, por lo que a continuación se comentan las características de las funciones básicas del algoritmo para las hiperheurísticas:

- `Inicializar:` Debido al elevado coste computacional de las hiperheurísticas, el número de elementos en el conjunto inicial y en el conjunto de referencia es normalmente más pequeño que para las metaheurísticas. Así, es preferible la aplicación de métodos de búsqueda locales o métodos basados en poblaciones pero con poblaciones pequeñas. Se generan elementos aleatorios válidos, con diferentes rangos para diferentes parámetros, y con rangos más reducidos que para las metaheurísticas (poblaciones más pequeñas, porcentajes de mejora más bajos, intensificaciones no muy agresivas, etc).

- `CondiciónDeFin:` La condición de fin es similar a la usada para las metaheurísticas, pero los límites de los números de iteraciones serán más bajos.

- `Seleccionar:` La selección de elementos se hace como en las metaheurísticas, con conjuntos de mejores y peores elementos. La única diferencia es la función de fitness usada para clasificar los elementos. Como se ha comentado, hay varias posibilidades para la función de fitness, y en la sección de experimentos se compararán algunas de ellas.

- `Combinar:` Como en el caso de las metaheurísticas, se combinan pares de elementos mejores, peores y mejores-perores, pero la combinación es normalmente dependiente del problema, así que se deben obtener combinaciones satisfactorias para el problema de selección de metaheurísticas. Hay varias posibilidades:

  - ComUno: Cruce típico con selección aleatoria del punto para cruzar los dos ascendientes.
  - ComAlea: Como no existe un orden particular en los parámetros metaheurísticos, los parámetos a pasar de los ascendientes 1 y 2 a los descendientes 1 y 2 pueden seleccionarse aleatoriamente. El problema aquí es que la selección aleatoria puede producir configuraciones no válidas, por ejemplo, valores más altos de $NEFIni$ que de $NEIIni$. Por lo tanto, los valores obtenidos para los descendientes necesitan ser modificados para tener configuraciones válidas.
  - ComGru: Los parámetros se pueden organizar y seleccionar por grupos. Por ejemplo, $GrupoIni = \{NEIIni, NEFIni\}$, $GrupoMejIni = \{PEMIni, IMEIni\}$, etc, y se seleccionan grupos de parámetros de entre los ascendientes para generar los descendientes.
  - ComLin: La combinación se puede hacer con la combinación lineal de los parámetros de las metaheurísicas. Por ejemplo, si las metaheurísticas de los ascendientes quedan determinadas por los vectores de parámetros p_1 y p_2, $Asc1 = MH(p_1)$ y $Asc2 = MH(p_2)$, un descendiente se determina con un valor $0 \leq \alpha \leq 1$ de modo que $Des = MH(\alpha p_1 + (1-\alpha)p_2)$, que representa una metaheurística en el camino entre los dos ascendientes.

- `Mejorar:` La mejora funciona de la misma forma que con la metaheurísticas, pero con valores inferiores para los parámetros para reducir el tiempo de ejecución.

- `Incluir:` Las únicas diferencias con las metaheurísticas son la función de fitness para la selección de los mejores elementos, y un valor más bajo del parámetro para la memoria a largo plazo tabú, puesto que normalmente las hiperheurísticas alcanzan el óptimo en menos iteraciones y por tanto no es necesaria una memoria de iteraciones anteriores demasiado extensa.

Como en el caso de las metaheurísticas, el conjunto de los parámetros hiperheurísticos es la unión de los seis conjuntos de parámetros de las funciones básicas

en el esquema. Por tanto, tenemos un conjunto de 20 parámetros con los que se puede experimentar y aplicar las hiperheurísticas para seleccionar buenas metaheurísticas para los problemas objetivo.

4.2. Aplicación de hiperheurísticas a la optimización de metaheurísticas

La validez de la metodología se analiza aplicándola a los problemas de optimización descritos en el capítulo 2, que nos sirven como casos de prueba para extraer algunas conclusiones. En este apartado se va a contrastar la aplicabilidad de las hiperheurísticas y su idoneidad como herramienta de optimización general, independiente del problema, en términos de fitness para varios problemas de test. Tendremos en cuenta inicialmente algunas consideraciones generales válidas para todos los problemas, y se analizarán en detalle los resultados particulares obtenidos para cada problema concreto.

Como ya comentamos en capítulos anteriores, la introducción de la hiperheurística como herramienta para la selección automática de las mejores metaheurísticas para un problema o conjunto de problemas, conlleva la aparición de un nuevo problema de optimización [56, 58, 59]: la selección de los parámetros metaheurísticos óptimos que definen dichas metaheurísticas óptimas para resolver ciertos problemas (POPME). Teniendo en cuenta esto, cada vez que se utilice una hiperheurística para solucionar algún problema de optimización, en realidad se estará resolviendo el problema POPME.

Inicialmente es necesario establecer la hiperheurística o hiperheurísticas que se usarán. Puesto que las hiperheurísticas se desarrollan usando el esquema parametrizado de metaheurísticas, basta con fijar los parámetros metaheurísticos de la hiperheurística (los llamamos parámetros hiperheurísticos) a valores específicos. En la tabla 4.1 se pueden ver los parámetros usados para los tres tipos de hiperheurísticas que nos servirán como referencia para seleccionar metaheurísticas satisfactorias para cada problema:

- Se ha elegido una hiperheurística híbrida de tamaño medio (Hhi), con algunas decisiones tomadas para reducir el tiempo de ejecución: poblaciones pequeñas (valores de $NEIIni$ y $NEFIni$ moderados), con mejoras de la mitad de los elementos ($PEMIni$ y $PEMMej$ con un valor del cincuenta por ciento) y no muy intensas (intensidades $IMEIni$ e $IMEMej$ de cinco), y con un número de elementos a combinar moderado, como se puede ver en los valores de los parámetros de la combinación.

- La hiperheurística con un enfoque de algoritmo genético (Hge) también trabaja con una población pequeña y la mejora se hace solo en la diversificación (parámetros $PEDMej$ y $IDEMej$), lo que significa que los elementos obtenidos por mutación se mejoran (con los mismos parámetros $PEMMej$ e

	NEIIni	NEFIni	PEMIni	IMEIni	MCPIni	NEMSel
Hhi	20	20	50	5	5	10
Hre	5	5	50	3	2	3
Hge	20	20	0	0	0	20

	NEPSel	NMMCom	NMPCom	NPPCom	PEMMej	IMEMej
Hhi	10	15	20	15	50	5
Hre	2	2	3	2	50	3
Hge	0	10	0	0	0	0

	MCMMej	PEDMej	IDEMej	MCDMej	NEMInc	MLPInc
Hhi	5	10	5	5	10	5
Hre	2	10	5	2	3	5
Hge	0	10	5	0	20	0

Tabla 4.1: Valores de los parámetros hiperheurísticos utilizados para la selección de metaheurísticas. Hiperheurística híbrida (Hhi), Hiperheurística híbrida reducida (Hre) e Hiperheurística basada en un Algoritmo Genético (Hge).

$IMEMej$) para reducir la posibilidad de una desaparición rápida. Los valores de los parámetros Tabú se establecen en cero.

- Debido a que el tiempo de ejecución del enfoque genético es mucho más pequeño que el del enfoque híbrido, se considera también una hiperheurística híbrida reducida (Hre). Ésta tiene los mismos parámetros que la hiperheurística híbrida previa pero con valores más bajos para muchos de ellos con el fin de reducir el tiempo de ejecución. Por ejemplo, los valores de $NEIIni$ y $NEFIni$ se establecen en cinco, y el resto de parámetros, en la misma fila de la tabla 4.1, se reducen también de forma significativa, excepto los porcentajes de mejora, los parámetros de mutación y el de memoria a largo plazo tabú.

Para verificar la efectividad de las hiperheurísticas, parece razonable comparar los fitness obtenidos con la metaheurísticas seleccionadas por las hiperheurísticas con aquellos obtenidos con algunas metaheurísticas puras aplicadas directamente al problema de optimización. Se han elegido instancias de los parámetros correspondientes a las cuatro metaheurísticas básicas consideradas (GRASP, GA, SS y TS). En el capítulo anterior experimentábamos con metaheurísticas híbridas para obtener las más eficientes para nuestros problemas de optimización. Por lo tanto, parece también apropiado comparar los mejores resultados obtenidos con las mejores metaheurísticas híbridas con los obtenidos con las hiperheurísticas.

La tabla 4.2 muestra los valores seleccionados para los parámetros para las cuatro metaheurísticas básicas y algunas de las mejores metaheurísticas híbridas obtenidas para los problemas PCEPA (Mhi1) y PCOCI (Mhi2). Los valores de los parámetros metaheurísticos son en general mayores que los correspondientes valores para las hiperheurísticas, porque la aplicación de una metaheurística requiere tiempos de ejecución mucho menores que los de una hiperheurística, donde se aplican muchas metaheurísticas para la selección de una metaheurística satisfactoria.

	NEIIni	NEFIni	PEMIni	IMEIni	MCPIni	NEMSel
GR	200	1	100	50	0	0
GA	100	100	0	0	0	100
SS	100	20	100	50	0	10
TS	200	1	100	10	5	1
Mhi1	200	50	100	50	10	25
Mhi2	50	15	100	15	2	8

	NEPSel	NMMCom	NMPCom	NPPCom	PEMMej	IMEMej
GR	0	0	0	0	0	0
GA	0	50	0	0	0	0
SS	10	90	100	90	100	5
TS	0	0	0	0	100	5
Mhi1	25	90	100	90	100	5
Mhi2	7	15	20	15	100	5

	MCMMej	PEDMej	IDEMej	MCDMej	NEMInc	MLPInc
GR	0	0	0	0	0	0
GA	0	10	5	0	100	0
SS	0	0	0	0	10	0
TS	5	0	0	0	1	20
Mhi1	10	10	5	10	25	30
Mhi2	2	20	5	2	8	3

Tabla 4.2: Valores de los parámetros para las cuatro metaheurísticas puras y para las metaheurísticas híbridas (Mhi1 y Mhi2) consideradas.

En principio, cualquier valor de los parámetros puede determinar una metaheurística, pero es preferible establecer rangos que nos permitan comparar los resultados obtenidos con la aplicación de la hiperheurística con aquellos obtenidos con metaheurísticas puras. Por lo tanto, la hiperheurística seleccionará parámetros metaheurísticos para cada función del esquema en los rangos de valores presentados en la tabla 4.3, donde se considera que valores bajos mayores o iguales que cero

pueden ser preferibles para algunos parámetros en muchos casos. Los dos parámetros de la `CondiciónDeFin` se han fijado a 1000 y 10 para el problema PCEPA y a valores más bajos (10 para ambos) en el problema PCOCI debido a su mayor coste computacional.

	NEIIni	NEFIni	PEMIni	IMEIni	MCPIni	NEMSel
Inferior	5	5	0	1	0	2
Superior	200	100	100	20	15	100

	NEPSel	NMMCom	NMPCom	NPPCom	PEMMej	IMEMej
Inferior	2	5	5	5	0	1
Superior	100	100	100	100	100	20

	MCMMej	PEDMej	IDEMej	MCDMej	NEMInc	MLPInc
Inferior	0	0	1	0	2	0
Superior	15	100	10	15	100	15

Tabla 4.3: Límites inferiores y superiores de los parámetros metaheurísticos para la selección de individuos por medio de la hiperheurística.

Al ejecutar la hiperheurística, se aplican muchas metaheurísticas a diferentes instancias de problema, lo que conlleva elevados tiempos de ejecución, y es necesario paralelismo. Se pueden emplear metaheurísticas paralelas para reducir el tiempo de ejecución, pero es también posible, y preferible, usar paralelismo a nivel superior, en el esquema metaheurístico usado para la hiperheurística [56, 58]. También es posible combinar paralelismo en la hiperheurística y las metaheurísticas. Aunque los esquemas paralelos de las metaheurísticas (y consecuentemente de las hiperheurísticas) se estudian en los capítulos siguientes, para reducir los tiempos de ejecución los experimentos se han llevado a cabo con paralelismo de memoria compartida en los dos niveles, y en cada uno de ellos con paralelismo anidado, lo que proporciona cuatro niveles de paralelismo [6], y el número de hilos en cada nivel se ha seleccionado con técnicas de autooptimización [61]. Se analizarán en detalle los resultados de tiempo de ejecución en los siguientes capítulos de este trabajo.

En las subsecciones siguientes se presentan los resultados de la aplicación de las hiperheurísticas a los problemas de optimización utilizados como casos de prueba.

4.2.1. Aplicación de hiperheurísticas a PCEPA

En el capítulo 2 presentábamos el problema de consumo de electricidad en la explotación de pozos de agua. Comenzaremos estudiando la aplicación de hiperheurísticas a este problema, con la idea de extraer conclusiones generales sobre la

aplicación de hiperheurísticas basadas en esquemas metaheurísticos parametrizados, y el método se validará en la siguiente subsección con el otro problema considerado (PCOCI).

Los diferentes tamaños de PCEPA empleados en los experimentos tienen la configuración mostrada en la tabla 4.4, donde algunas de las variables del problema se han fijado a valores específicos razonables para el tamaño del sistema considerado y otros (entre corchetes) se han tomado aleatoriamente dentro de un intervalo. Las variables utilizadas son aquellas descritas en la subsección 2.2.1. Los valores de las tarifas eléctricas son los mismos que en el caso de la aplicación directa del esquema metaheurístico a PCEPA presentado en el capítulo 3. Los tamaños de problema son aquí notablemente inferiores (cuarenta bombas y seis rangos horarios como mucho) a los usados en el capítulo 3 debido a que el coste computacional es ahora más elevado con la aplicación de las hiperheurísticas.

B-R	10-3	10-6	20-3	20-6	40-3	40-6
$V_{td}{\cdot}10^{-4}$	2	2	4	4	8	8
$Q_{min}{\cdot}10^{-2}$	7	7	10	10	20	20
Q_j	[40-300]	[40,300]	[40,300]	[40,300]	[40,300]	[40,300]
P_j	[30,400]	[30,400]	[30,400]	[30,400]	[30,400]	[30,400]
$\sigma_{lim}{\cdot}10^{-2}$	25	25	25	25	25	25
$\sigma_j{\cdot}10^{-2}$	[1,30]	[1,30]	[1,30]	[1,30]	[1,30]	[1,30]
$V_{c,j}{\cdot}10^{-3}$	[1,10]	[1,10]	[1,10]	[1,10]	[3,10]	[3,10]
PD_j	[150,300]	[150,300]	[150,300]	[150,300]	[150,300]	[150,300]
PM_j	[270,350]	[270,350]	[250,350]	[250,350]	[230,350]	[230,350]

Tabla 4.4: Valores de las variables para los distintos tamaños de PCEPA considerados en los experimentos.

Como se ha dicho antes, se han seleccionado tres combinaciones de parámetros que definen tres configuraciones de hiperheurísticas a aplicar al problema. Hay que recordar que la hiperheurística, debido a su modo de operación, conlleva un elevado coste computacional. Considerando esto, se han hecho las primeras comparaciones del fitness y del tiempo de ejecución al aplicar las tres hiperheurísticas a una instancia reducida del problema (implementadas en memoria compartida). El experimento se llevó a cabo en *Saturno*.

Estamos interesados en minimizar la función de fitness pero, si es posible, sin incrementar demasiado el tiempo de ejecución, por lo que se usará como indicador común el inverso del producto del fitness y el tiempo de ejecución. Por lo tanto, son deseables valores elevados para este indicador.

La tabla 4.5 muestra una comparativa de las tres hiperheurísticas consideradas con la función de comparación ComGru. Para aumentar la rapidez de la experimentación se utiliza un paralelismo de cuatro niveles teniendo en cuenta el número

máximo de procesadores disponibles en el sistema. La tabla muestra los valores de fitness obtenidos, el tiempo de ejecución y el indicador común (IC), para una instancia de PCEPA de tamaño 10-3 (Fit1P1E). Como los valores de fitness son idénticos en este caso, la única comparación se da en el tiempo de ejecución, y la hiperheurística basada en algoritmos genéticos es preferible, y para tamaños grandes o ejecuciones computacionalmente costosas es recomendable aplicar hiperheurísticas reducidas.

	Hhi	Hre	Hge
fitness (€)	2047.81	2047.81	2047.81
tiempo (s)	49705	5212	1517
IC	9.82	93.69	321.90

Tabla 4.5: Comparativa de los valores de fitness (en euros), tiempos de ejecución (en segundos) e indicador común ($IC = 10^9/(f \cdot t)$) para las tres configuraciones hiperheurísticas consideradas obtenidos al ejecutar una instancia del problema PCEPA de tamaño 10-3. Implementación paralela con ComGru como función de combinación y fitness calculado como Fit1P1E.

Diferentes instancias del problema tendrán valores óptimos diferentes, y por ello la influencia del fitness sobre el indicador común depende de la instancia utilizada. Por otro lado, una implementación paralela supondrá normalmente un tiempo de ejecución menor, y ejecuciones en sistemas computacionales diferentes también darán tiempos diferentes; por tanto la influencia del tiempo de ejecución en el indicador depende de factores relacionados con la implementación y la experimentación. Por consiguiente, el indicador se puede usar sólo para comparaciones relativas entre implementaciones similares de hiperheurísticas aplicadas a un problema particular. Además, la bondad de una hiperheurística se mide mejor a través de la bondad de la metaheurística seleccionada cuando se aplica a instancias nuevas del problema, y el indicador será usado para esta evaluación conjunta.

En la sección 4.1 se presentaron las funciones que conformaban el esquema metaheurístico usado en este caso para definir la hiperheurística. Existen varias posibilidades para la implementación de la función de combinación, y algunas más adecuadas que otras. Dados los problemas mencionados en algunas de las combinaciones y por simplificar los experimentos, sólo se han considerado los dos últimos tipos de combinación en los experimentos: por grupos (ComGru) y por combinación lineal de los parámetros (ComLin).

En la tabla 4.6 se compara el fitness obtenido al usar los dos métodos de combinación para diferentes tamaños de problema. Los experimentos se han llevado a cabo para las dos configuraciones más ligeras de hiperheurísticas (Hre y Hge), aunque las conclusiones para la hiperheurística híbrida son esencialmente las mismas: no hay diferencia apreciable en el fitness obtenido usando combinación por grupos

de parámetros o combinación lineal de ellos. Por lo tanto, de aquí en adelante, la única combinación que se considerará en los experimentos será ComGru.

	ComGru	ComLin	Media
Hre	3043.23	3043.64	3043.44
Hge	3041.59	3041.85	3041.72
Media	3041.41	3042.75	

Tabla 4.6: Comparativa de los valores medios de fitness para diferentes instancias de PCEPA, obtenidos con dos implementaciones diferentes de la función de combinación para dos configuraciones de hiperheurísticas.

El objetivo de una hiperheurística es encontrar una metaheurística que proporcione una solución satisfactoria para diferentes instancias del problema. Se han planteado varias posibilidades para la función de fitness, algunas más convenientes que otras respecto a la dependencia del problema y el coste computacional. La idea básica es aplicar la hiperheurística a un número de instancias del problema y verificar que las metaheurísticas obtenidas proporcionan buenos resultados cuando se aplican a otras instancias del problema. Además, podemos evaluar la bondad de una hiperheurística comparando los valores de fitness alcanzados al aplicar las metaheurísticas obtenidas con dicha hiperheurística para una o varias instancias de problema sobre otras instancias de test, con los valores de fitness alcanzados al aplicar directamente las distintas configuraciones hiperheurísticas (que podemos considerar que representan los mejores valores alcanzables con la aplicación de hiperheurísticas) o aplicando directamente las metaheurísticas puras sobre dichas instancias de test.

Primeramente se han considerado experimentos calculando el fitness con una instancia del problema y en una ejecución (Fit1P1E) y calculándolo con varias instancias del problema con ejecuciones independientes y medias de los parámetros (FitVPVE2). La tabla 4.7 muestra los parámetros metaheurísticos obtenidos cuando se aplican diferentes configuraciones de la hiperheurística al problema PCEPA para una instancia del problema de tamaño 10-3. Las filas 1, 3 y 5 muestran el resultado de aplicar la hiperheurística a un instancia individual del problema (Fit1P1E), mientras que las filas 2, 4 y 6 representan los parámetros metaheurísticos obtenidos como resultado de hacer la media de tres ejecuciones (FitVPVE2). En este caso particular no hubo problemas de incompatibilidades de parámetros al calcular la media.

Nuestro principal interés aquí no está en el análisis de la influencia de los parámetros en la bondad de las soluciones, sino en el desarrollo de hiperheurísticas mediante el esquema parametrizado de metaheurísticas. Se puede hacer un análisis más detallado de la influencia de los parámetros metaheurísticos sobre la bondad de la solución por medio de un estudio estadístico similar al hecho en [7]. No obstante, se pueden sacar algunas conclusiones sobre los valores obtenidos. Es importante

	NEIIni	NEFIni	PEMIni	IMEIni	MCPIni	NEMSel
$Hhi_{Fit1P1E}$	8	8	50	16	14	8
$Hhi_{FitVPVE2}$	39	37	54	12	9	33
$Hre_{Fit1P1E}$	111	60	6	16	8	2
$Hre_{FitVPVE2}$	106	39	26	11	10	19
$Hge_{Fit1P1E}$	122	43	44	13	7	43
$Hge_{FitVPVE2}$	82	49	57	6	9	26

	NEPSel	NMMCom	NMPCom	NPPCom	PEMMej	IMEMej
$Hhi_{Fit1P1E}$	0	37	0	0	16	7
$Hhi_{FitVPVE2}$	4	33	22	5	27	12
$Hre_{Fit1P1E}$	58	30	69	94	94	10
$Hre_{FitVPVE2}$	19	44	23	31	72	13
$Hge_{Fit1P1E}$	0	99	0	0	40	11
$Hge_{FitVPVE2}$	23	94	21	29	36	10

	MCMMej	PEDMej	IDEMej	MCDMej	NEMInc	MLPInc
$Hhi_{Fit1P1E}$	4	60	4	9	8	9
$Hhi_{FitVPVE2}$	5	50	6	9	19	12
$Hre_{Fit1P1E}$	13	49	2	8	15	2
$Hre_{FitVPVE2}$	7	71	4	7	22	2
$Hge_{Fit1P1E}$	1	97	9	8	43	5
$Hge_{FitVPVE2}$	8	89	7	5	49	6

Tabla 4.7: Valores de los parámetros metaheurísticos obtenidos al aplicar las hiperheurísticas híbrida (Hhi), híbrida-reducida (Hre) y genética (Hge) a una instancia de tamaño 10-3 de PCEPA (Fit1P1E) y a varias instancias (FitVPVE2).

destacar que:

- Excepto la primera fila, el resto tienen valores intermedios del parámetro $NEIIni$, considerando que el intervalo tiene un valor máximo de 200.
- Vemos también que el porcentaje de mejora en muchos casos está en torno a cincuenta, con una intensidad de la mejora con valores relativamente altos en el rango considerado.
- Con respecto a las combinaciones, se suele dar prioridad a la combinación de los mejores elementos.
- Es destacable que la diversificación es importante para este problema y que el número de mejores elementos a incluir en el conjunto de referencia es normalmente alto.

Se podrían presentar tablas similares para otros tamaños de problema pero, por claridad, solo se presentan los valores correspondientes al tamaño 10-3.

La figura 4.1 y su tabla asociada 4.8 comparan el fitness obtenido para diferentes tamaños de problema cuando se les aplican las metaheurísticas obtenidas por las hiperheurísticas utilizando instancias de tamaño 10-3. Se muestran los cocientes del fitness para cada metaheurística con respecto al mejor para cada tamaño de problema. Este fitness mejor se obtiene normalmente aplicando directamente a cada tamaño de problema alguna de las tres configuraciones hiperheurísticas de la tabla 4.1, y se eligió el cociente porque permite una mejor visualización de los resultados. Los fitness se comparan también con aquellos obtenidos con la aplicación de las metaheurísticas puras (tabla 4.2) y con los obtenidos aplicando directamente a cada problema las metaheurísticas con los parámetros de la tabla 4.1.

Las metaheurísticas obtenidas a partir de las hiperheurísticas con el fitness calculado como FitVPVE2 y como Fit1P1E arrojan resultados de fitness intermedios entre aquellos alcanzados por las metaheurísticas puras y los obtenidos con las metaheurísticas seleccionadas por las hiperheurísticas, que representan los mejores valores posibles que se pueden obtener con la aplicación del enfoque hiperheurístico considerado. Las últimas entradas en la figura (última fila en la tabla) representan la media del cociente del fitness respecto del mejor fitness para cada tamaño de problema. Los resultados medios son similares a los que acabamos de describir. La razón por la que los resultados obtenidos con la aplicación de las metaheurísticas seleccionadas por las hiperheurísticas no mejoran significativamente los obtenidos con las metaheurísticas puras puede deberse a que el tamaño de problema elegido (10-3) no está cercano a la media de todos los tamaños considerados, por lo que las metaheurísticas seleccionadas pueden ser menos satisfactorias para tamaños de problema alejados del PCEPA 10-3.

La figura 4.2 y la tabla 4.9 presentan los resutados equivalentes a los de la figura 4.1 y tabla 4.8, pero en esta ocasión usando las metaheurísticas obtenidas a partir

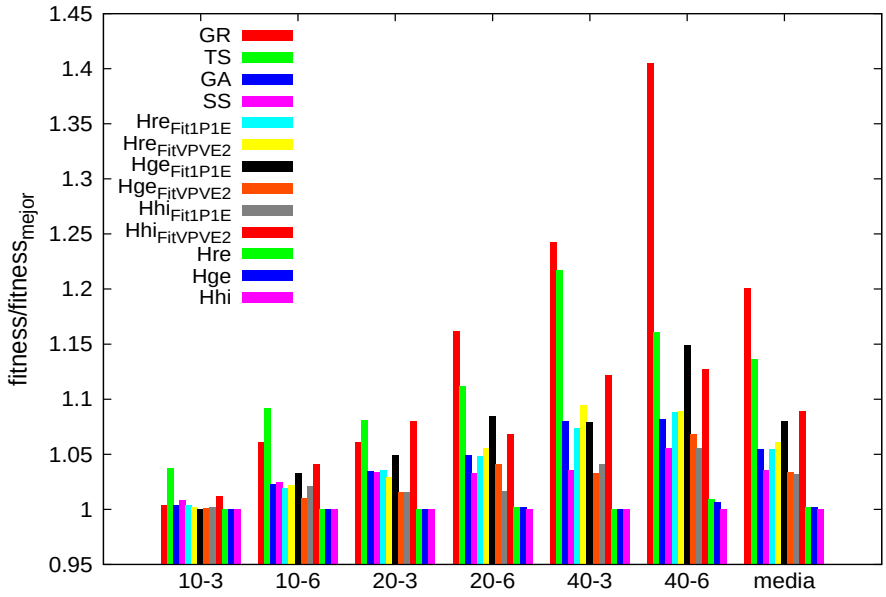


Figura 4.1: Cociente del fitness respecto al mejor fitness, para varios tamaños del PCEPA, para cuatro metaheurísticas puras (GRASP, Búsqueda Tabú, Algoritmo genético y Búsqueda Dispersa), para metaheurísticas híbridas seleccionadas por las hiperheurísticas para el tamaño de problema 10-3 con dos fitness (Fit1P1E y FitVP-VE2), y para hiperheurísticas directamente aplicadas a instancias de cada tamaño de problema.

	GR	TS	GA	SS	$\text{Hre}_{\text{Fit1P1E}}$	$\text{Hre}_{\text{FitVPVE2}}$	$\text{Hge}_{\text{Fit1P1E}}$
10-3	1.004	1.037	1.004	1.008	1.004	1.002	1.000
10-6	1.061	1.092	1.023	1.025	1.019	1.022	1.033
20-3	1.061	1.081	1.035	1.034	1.036	1.029	1.049
20-6	1.162	1.112	1.049	1.033	1.048	1.056	1.085
40-3	1.243	1.217	1.080	1.036	1.074	1.095	1.079
40-6	1.405	1.161	1.082	1.056	1.088	1.089	1.149
Media	1.201	1.136	1.055	1.036	1.055	1.061	1.080

	$\text{Hge}_{\text{FitVPVE2}}$	$\text{Hhi}_{\text{Fit1P1E}}$	$\text{Hhi}_{\text{FitVPVE2}}$	Hre	Hge	Hhi
10-3	1.001	1.002	1.012	1.000	1.000	1.000
10-6	1.010	1.021	1.041	1.000	1.000	1.000
20-3	1.016	1.016	1.080	1.000	1.000	1.000
20-6	1.041	1.017	1.068	1.002	1.002	1.000
40-3	1.033	1.041	1.122	1.000	1.000	1.000
40-6	1.068	1.056	1.127	1.009	1.007	1.000
Media	1.034	1.032	1.089	1.002	1.002	1.000

Tabla 4.8: Cociente del fitness respecto al mejor fitness, para varios tamaños del PCEPA, para cuatro metaheurísticas puras (GRASP, Búsqueda Tabú, Algoritmo genético y Búsqueda Dispersa), para metaheurísticas híbridas seleccionadas por las hiperheurísticas para el tamaño de problema 10-3 con dos fitness (Fit1P1E y FitVP-VE2), y para hiperheurísticas directamente aplicadas a instancias de cada tamaño de problema.

del problema de tamaño 20-3. La mejora alcanzada al aplicar las metaheurísticas seleccionadas a otras instancias del problema es más notable, y cercana al fitness resultado de la aplicación directa de las hiperheurísticas a cada problema. Esto se explica porque 20-3 es un tamaño de problema intermedio, por lo que los problemas de este tamaño son más representativos de aquellos con los que se ha experimentado. El uso de un problema mayor para la aplicación de la hiperheurística representa un coste computacional mayor, pero la hiperheurística se aplica una vez, y el incremento en el tiempo de ejecución puede ser asumible dada la mejora en las metaheurísticas obtenidas. En la tabla 4.10 podemos ver una comparativa de la media del tiempo de ejecución obtenido para las tres configuraciones hiperheurísticas consideradas al ejecutar tres instancias del problema de tamaños 10-3 y 20-3 en paralelo, confirmándose que al duplicar el tamaño del problema, el coste computacional, de media, casi se duplica.

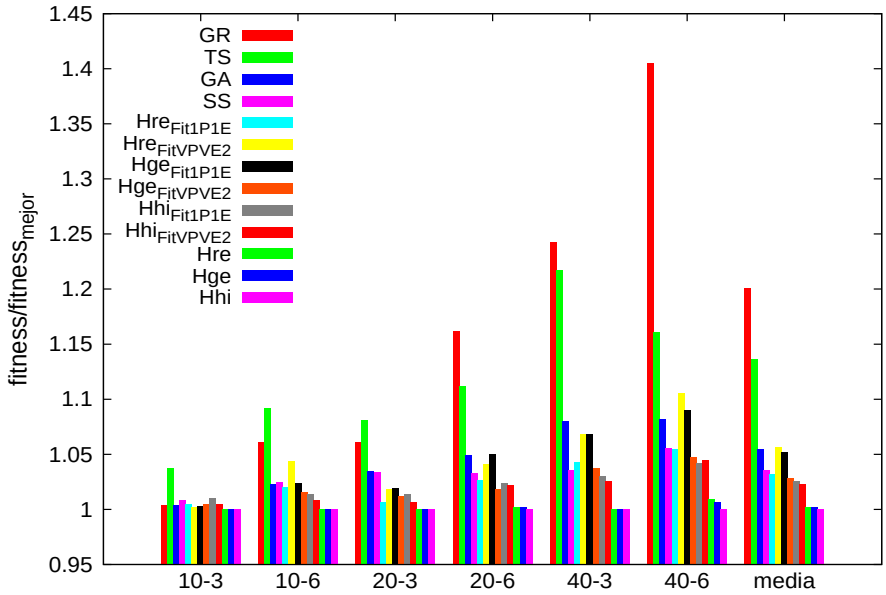


Figura 4.2: Cociente del fitness respecto al mejor fitness, para varios tamaños del PCEPA, para cuatro metaheurísticas puras (GRASP, Búsqueda Tabú, Algoritmo genético y Búsqueda Dispersa), para metaheurísticas híbridas seleccionadas por las hiperheurísticas para el tamaño de problema 20-3 con dos fitness (Fit1P1E y FitVPVE2), y para hiperheurísticas directamente aplicadas a instancias de cada tamaño de problema.

Hasta este punto, los resultados de fitness y tiempo de ejecución se han obtenido a partir de la aplicación directa de la hiperheurística a cada instancia del problema. Se han comparado los resultados de la aplicación de metaheurísticas puras y de metaheurísticas seleccionadas por la hiperheurística. En este caso, se han obtenido buenas metaheurísticas. A partir de aquí, el estudio se centra en la aplicación a

	GR	TS	GA	SS	$\text{Hre}_{\text{Fit1P1E}}$	$\text{Hre}_{\text{FitVPVE2}}$	$\text{Hge}_{\text{Fit1P1E}}$
10-3	1.004	1.037	1.004	1.008	1.005	1.002	1.003
10-6	1.061	1.092	1.023	1.025	1.020	1.044	1.024
20-3	1.061	1.081	1.035	1.034	1.007	1.018	1.019
20-6	1.162	1.112	1.049	1.033	1.027	1.041	1.050
40-3	1.243	1.217	1.080	1.036	1.043	1.068	1.068
40-6	1.405	1.161	1.082	1.056	1.055	1.106	1.090
Media	1.201	1.136	1.055	1.036	1.032	1.057	1.052

	$\text{Hge}_{\text{FitVPVE2}}$	$\text{Hhi}_{\text{Fit1P1E}}$	$\text{Hhi}_{\text{FitVPVE2}}$	Hre	Hge	Hhi
10-3	1.005	1.010	1.005	1.000	1.000	1.000
10-6	1.016	1.014	1.008	1.000	1.000	1.000
20-3	1.012	1.014	1.007	1.000	1.000	1.000
20-6	1.018	1.024	1.022	1.002	1.002	1.000
40-3	1.037	1.030	1.026	1.000	1.000	1.000
40-6	1.047	1.042	1.045	1.009	1.007	1.000
Media	1.028	1.026	1.023	1.002	1.002	1.000

Tabla 4.9: Cociente del fitness respecto al mejor fitness, para varios tamaños del PCEPA, para cuatro metaheurísticas puras (GRASP, Búsqueda Tabú, Algoritmo genético y Búsqueda Dispersa), para metaheurísticas híbridas seleccionadas por las hiperheurísticas para el tamaño de problema 20-3 con dos fitness (Fit1P1E y FitVP-VE2), y para hiperheurísticas directamente aplicadas a instancias de cada tamaño de problema.

	Hhi	Hre	Hge	Media
10-3	49705	5212	1517	18811
20-3	77552	8191	2863	29535
Media	63629	6702	2190	

Tabla 4.10: Comparativa de la media de los tiempos de ejecución (en segundos) para las tres configuraciones hiperheurísticas consideradas al ejecutar tres instancias del problema PCEPA de tamaños 10-3 y 20-3 en paralelo.

nuevos problemas de metaheurísticas obtenidas con hiperheurísticas que aprenden de problemas pequeños.

La tabla 4.11 resume el fitness medio (equivalente a la media en las figuras 4.1 y 4.2) alcanzado al aplicar las metaheurísticas obtenidas con las hiperheurísticas cuando se usan los problemas de tamaños 10-3 y 20-3 para la computación del fitness a los otras instancias del problema. Existen diferencias significativas entre los valores obtenidos usando las diferentes configuraciones hiperheurísticas. Esto confirma la dependencia del problema postulada previamente.

Hay también diferencias en cuanto a tiempo de ejecución en la aplicación de las diferentes metaheurísticas obtenidas por diferentes métodos hiperheurísticos. Al decidir sobre si una metaheurística es buena o no puede ser aconsejable no solo analizar sus valores de fitness sino también si fueron obtenidos en un tiempo de ejecución moderado. Por tanto se presenta una tercera fila con los valores del indicador común (*IC*). Así, se concluye que los mejores valores se logran cuando se aplican las metaheurísticas obtenidas a partir de la hiperheurística híbrida usando el tamaño de problema 10-3 de PCEPA. Si hemos obtenido metaheurísticas usando instancias del PCEPA 20-3, es mejor aplicar aquellas obtenidas con la hiperheurísticaa genética a partir de un sola instancia de problema (Fit1P1E).

Como en la tabla 4.6, donde se compararon dos métodos de combinación, la tabla 4.12 compara el fitness, el tiempo de ejecución y el indicador común (*IC*) obtenidos con el cálculo del fitness mediante Fit1P1E y FitVPVE2. La tabla muestra la media del fitness, tiempo e *IC* de la tabla 4.11 para cada hiperheurística y cálculo del fitness, y la media para cada hiperheurística con los dos métodos de cálculo del fitness (últimas tres filas) y para los dos métodos de cálculo del fitness cuando se usan en las tres hiperheurísticas (última columna). El mejor método de cálculo del fitness es Fit1P1E (en términos de fitness e *IC*), lo que puede ser debido a posibles correlaciones entre los parámetros, mencionadas como un inconveniente para FitVPVE2. La hiperheurística híbrida arroja los mejores resultados seguida de la genética, en fitness e *IC*. En conclusión, la mejor opción es aplicar metaheurísticas obtenidas a partir de hiperheurísticas híbridas, aunque esto pueda suponer aumentar considerablemente los tiempos de experimentación para obtener dichas metaheurísticas. También podemos usar las metaheurísticas obtenidas a partir de la hiperheurística genética para obtener resultados competitivos, siendo inferiores los tiempos de selección de las metaheurísticas mediante la aplicación de dicha hiperheurística.

Para resolver la dependencia de los resultados del problema de forma eficiente, el fitness puede calcularse en las hiperheurísticas con el método FitVP1E (varias instancias de problema en una misma ejecución). En este caso, en lugar de hacer múltiples ejecuciones de varias instancias del problema, solo se lleva a cabo una ejecución, y la instancia o tamaño de problema se varía en cada iteración. Esto proporciona una gran reducción en el tiempo de ejecución, además de reducir la dependencia del problema. La tabla 4.13 muestra los parámetros metaheurísticos obtenidos cuando se aplican diferentes configuraciones de las hiperheurísticas a tres instancias de PCEPA usando FitVP1E como criterio de cálculo del fitness. En este

		$\text{Hhi}_{\text{Fit1P1E}}$	$\text{Hhi}_{\text{FitVPVE2}}$	$\text{Hre}_{\text{Fit1P1E}}$
10-3	(*f*)	3135.28	3310.67	3206.20
	(*t*)	1.58	5.88	14.47
	(*IC*)	20.22	5.14	2.16
20-3	(*f*)	3118.50	3109.68	3136.11
	(*t*)	7.58	9.68	8.41
	(*IC*)	4.23	3.32	3.79

		$\text{Hre}_{\text{FitVPVE2}}$	$\text{Hge}_{\text{Fit1P1E}}$	$\text{Hge}_{\text{FitVPVE2}}$
10-3	(*f*)	3223.37	3280.93	3143.72
	(*t*)	13.07	13.59	17.72
	(*IC*)	2.37	2.24	1.79
20-3	(*f*)	3212.36	3198.34	3123.03
	(*t*)	13.42	4.51	10.54
	(*IC*)	2.32	6.94	3.04

Tabla 4.11: Medias de fitness (*f*), tiempo (*t*) e indicador común ($IC = 10^5/(f \cdot t)$) para los seis tamaños de problema PCEPA obtenidas al aplicar la metaheurística seleccionada por las hiperheurísticas con el fitness calculado como Fit1P1E y FitVPVE2 para los tamaños de problema 10-3 y 20-3.

		Hhi	Hre	Hge	Media
Fit1P1E	(*f*)	3126.89	3171.16	3239.63	3179.23
	(*t*)	4.58	11.44	9.05	8.36
	(*IC*)	12.23	2.97	4.59	6.60
FitVPVE2	(*f*)	3210.17	3217.87	3133.37	3187.14
	(*t*)	7.78	13.25	14.13	11.72
	(*IC*)	4.23	2.35	2.42	3.00
Media	(*f*)	3168.53	3194.51	3186.50	
	(*t*)	6.18	12.34	11.59	
	(*IC*)	8.23	2.66	3.50	

Tabla 4.12: Medias de fitness (*f*), tiempo (*t*) e indicador común ($IC = 10^5/(f \cdot t)$) en la tabla 4.11 para cada hiperheurística y cálculo del fitness, media de cada hiperheurística con los dos métodos de cálculo del fitness (últimas tres filas) y para los dos métodos de cálculo del fitness cuando se usan en las tres hiperheurísticas (última columna).

caso no hubo problemas de incompatibilidades de parámetros al calcular las medias.

	NEIIni	NEFIni	PEMIni	IMEIni	MCPIni	NEMSel
$Hhi_{FitVP1E}$	127	42	62	3	7	18
$Hre_{FitVP1E}$	143	60	14	11	5	36
$Hge_{FitVP1E}$	139	43	64	7	2	11

	NEPSel	NMMCom	NMPCom	NPPCom	PEMMej	IMEMej
$Hhi_{FitVP1E}$	21	74	34	77	16	15
$Hre_{FitVP1E}$	9	81	63	21	49	8
$Hge_{FitVP1E}$	31	71	25	66	43	9

	MCMMej	PEDMej	IDEMej	MCDMej	NEMInc	MLPInc
$Hhi_{FitVP1E}$	9	50	4	5	36	5
$Hre_{FitVP1E}$	4	86	8	9	58	11
$Hge_{FitVP1E}$	9	81	3	13	43	0

Tabla 4.13: Valores de los parámetros metaheurísticos obtenidos al aplicar las hiperheurísticas híbrida (Hhi), híbrida-reducida (Hre) y genética (Hge) a tres instancias de PCEPA variando la instancia en cada iteración en una misma ejecución (FitVP1E).

Hay que destacar que todas las configuraciones de hiperheurísticas tienen valores intermedios-altos del parámetro $NEIIni$. También podemos ver que el porcentaje de mejora en muchos casos está sobre el cincuenta por ciento, y la intensidad de la mejora en promedio es baja en el intervalo considerado. Como en las configuraciones de cálculo de fitness previas, se da prioridad normalmente a la combinación de los mejores elementos. Es destacable que la diversificación es también importante para estas configuraciones en este problema y que el número de mejores elementos a incluir en el conjunto de referencia es normalemte alto.

La figura 4.3 y la tabla 4.14 muestran los resultados de la aplicación de FitVP1E a los tres problemas más grandes usados como test. Las hiperheurísticas calculan el fitness para diferentes instancias de los tamaños de problema 10-3, 10-6 y 20-3, una instancia diferente en cada iteración de la hiperheurística.

En promedio, las metaheurísticas obtenidas con FitVP1E mejoran considerablemente los resultados conseguidos al aplicar las metaheurísticas puras, y proporcionan valores de fitness que son próximos a los mejores obtenidos aplicando la hiperheurística a los problemas de validación directamente. Significativamente, la metaheurística obtenida mediante la hiperheurística genética devuelve un fitness muy bueno y con tiempos de ejecución reducidos.

A continuación, se comparan los resultados obtenidos con las tres hiperheurísticas mediante las tres formas de calcular el fitness en la tabla 4.15. La tabla muestra

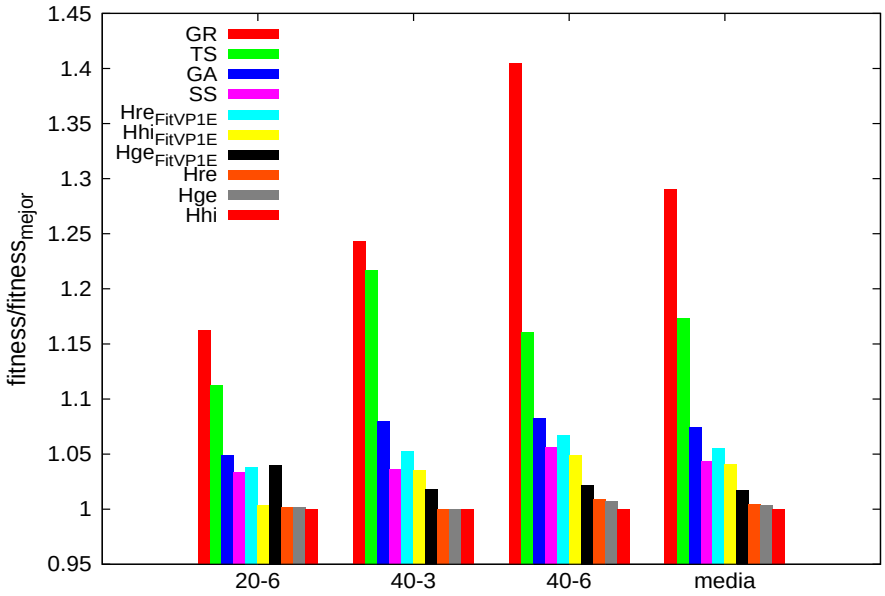


Figura 4.3: Cociente del fitness respecto al mejor fitness, para varios tamaños de PCEPA, para las cuatro metaheurísticas puras, para las metaheurísticas híbridas seleccionadas por las hiperheurísticas con problemas variables en una ejecución (FitVP1E), y para hiperheurísticas directamente aplicadas a instancias de cada tamaño de problema.

	GR	TS	GA	SS	Hre$_{FitVP1E}$
20-6	1.162	1.112	1.049	1.033	1.038
40-3	1.243	1.217	1.080	1.036	1.052
40-6	1.405	1.161	1.082	1.056	1.067
Media	1.290	1.173	1.074	1.043	1.055

	Hhi$_{FitVP1E}$	Hge$_{FitVP1E}$	Hre	Hge	Hhi
20-6	1.003	1.040	1.002	1.002	1.000
40-3	1.035	1.018	1.000	1.000	1.000
40-6	1.049	1.022	1.009	1.007	1.000
Media	1.041	1.017	1.004	1.003	1.000

Tabla 4.14: Cociente del fitness respecto al mejor fitness, para varios tamaños de PCEPA, para las cuatro metaheurísticas puras, para las metaheurísticas híbridas seleccionadas por las hiperheurísticas con problemas variables en una ejecución (FitVP1E), y para hiperheurísticas directamente aplicadas a instancias de cada tamaño de problema.

la media de fitness, tiempo e indicador común obtenido al aplicar las diferentes técnicas a tres problemas de validación de tamaños 20-6, 40-3 y 40-6. Se muestran también las medias para cada hiperheurística (columna) y método para el fitness (fila). Los valores de fitness obtenidos con FitVP1E mejoran los obtenidos con Fit1P1E y FitVPVE2. Hay poca diferencia entre los dos últimos, pero FitVPVE2 es ligeramente mejor en términos de fitness, posiblemente debido a su menor dependencia del problema. El mejor indicador común viene de Fit1P1E, posiblemente porque se obtiene una metaheurística menos efectiva a expensas de una significativa reducción del tiempo de ejecución. Esto puede también ocurrir en la aplicación de FitVPVE2.

		Hhi	Hre	Hge	Media
Fit1P1E	(f)	4065.60	4190.89	4322.45	4192.98
	(t)	2.80	25.76	22.91	17.16
	(IC)	8.78	0.93	1.01	3.57
FitVPVE2	(f)	4117.35	4174.70	4161.14	4151.06
	(t)	11.26	22.23	20.42	17.97
	(IC)	2.16	1.08	1.18	1.47
FitVP1E	(f)	4062.84	4118.14	3966.74	4049.24
	(t)	23.25	41.53	18.06	27.61
	(IC)	1.06	0.58	1.40	1.01
Media	(f)	4081.93	4161.24	4150.11	
	(t)	12.44	29.84	20.46	
	(IC)	4.00	0.86	1.19	

Tabla 4.15: Comparativa de las medias de fitness (f), tiempo (t) e indicador común ($IC = 10^5/(f \cdot t)$) obtenidas al aplicar las mejores metaheurísticas alcanzadas con Fit1P1E, FitVPVE2 and FitVP1E para varias configuraciones hiperheurísticas a instancias de tamaños 20-6, 40-3 y 40-6 del problema PCEPA.

Con respecto a la mejor hiperheurística, podemos decir que la híbrida arroja el mejor resultado global, en términos de fitness y tiempo, con un indicador relativamente alto (un valor de cuatro). Esto puede ser porque, aunque el tiempo experimental requerido puede ser alto, al final, se tiene una metaheurística altamente efectiva para la aplicación a otras instancias de problema. Como en la tabla 4.12, la segunda mejor hiperheurística es la genética, con un valor del indicador mayor que uno. Esto significa que se debería usar una hiperheurística híbrida si queremos asegurar resultados de calidad a expensas de un esfuerzo computacional considerable. Sin embargo, en muchos casos podemos usar una hiperheurística genética para obtener resultados competitivos con mucho menos tiempo de análisis experimental.

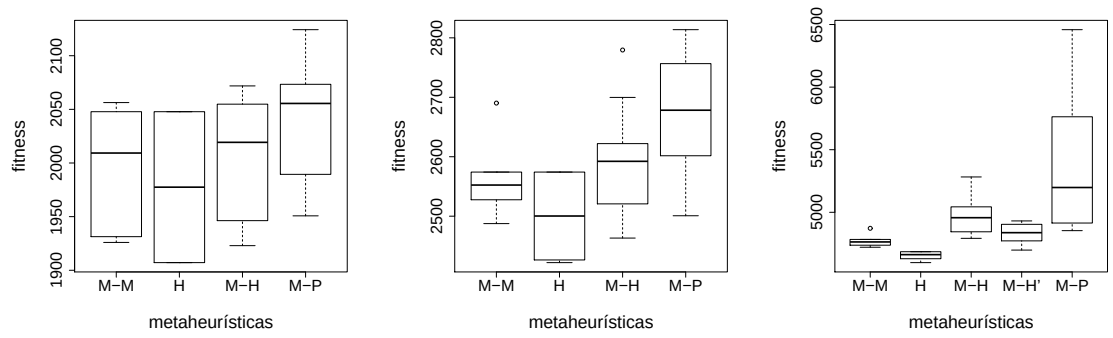
Otro modo de verificar la necesidad de una hiperheurística es compararla con resultados previos y ver si mejora lo que se obtuvo en términos de fitness. En la última columna de la tabla 4.16 se presentan los resultados obtenidos con la aplica-

ción de una de las mejores metaheurísticas (Mhi1 en la tabla 4.2) que se presentó en el capítulo 3 a seis problemas de tamaños diferentes de PCEPA. Podemos ver que en todos los casos las tres configuraciones hiperheurísticas mejoran esos resultados. Esto nos proporciona un nuevo argumento para la hiperheurística como un método automático de búsqueda bueno para encontrar la mejor metaheurística para un conjunto de instancias de un problema dado.

	Hhi	Hge	Hre	Mhi1
10-3	2047.81	2047.81	2047.81	2050.65
10-6	1907.14	1907.14	1907.14	1942.68
20-3	2574.21	2574.21	2574.21	2612.89
20-6	2421.89	2426.37	2426.37	2515.22
40-3	4685.18	4685.18	4685.18	4760.00
40-6	4598.05	4629.18	4638.37	4786.65
Media	3039.05	3044.98	3046.51	3111.35

Tabla 4.16: Comparativa del fitness obtenido al ejecutar las tres configuraciones hiperheurísticas directamente a las seis instancias diferentes de PCEPA, con el fitness obtenido al aplicar una de las mejores metaheurísticas híbridas (Mhi1) a esos problemas.

Finalmente, en la figura 4.4 se presenta un resumen estadístico de las medias obtenidas al aplicar los diferentes algoritmos metaheurísticos a varios tamaños del problema PCEPA. Los resultados se agrupan en conjuntos con un número variable de puntos en cada uno (entre seis y veinticuatro). Los conjuntos se obtuvieron a partir de resultados de aplicación de metaheurísticas de distinta naturaleza: la aplicación directa de las tres configuraciones hiperheurísticas (H), metaheurísticas obtenidas a partir de hiperheurísticas (M-H), la aplicación de instancias de la mejor metaheurística (M-M) no seleccionada automáticamente (combinación Mhi1 en la tabla 4.2), y el conjunto de metaheurísticas puras (M-P). Se consideró un número de puntos relativamente pequeño en algunos casos debido al elevado tiempo de ejecución de algunos algoritmos con varianzas pequeñas. Los tamaños de problema se agrupan en categorías según el número de bombas, así, por ejemplo, 20-x incluye instancias de tamaño 20-3 y 20-6. Las figuras 4.4 (a) y (b) muestran los resultados de aplicar M-H con el fitness calculado como Fit1P1E y FitVPVE2, y la figura 4.4 (c) presenta los mismos resultados pero con un conjunto más (M-H') que computa el fitness como FitVP1E. Se compara M-H y M-H' en (c) porque hay una gran diferencia entre las medias de los grupos estudiados para el tamaño 40-x. El test de Kruskal-Wallis reveló diferencias estadísticas en las medias para las instancias de PCEPA de tamaños 20-x y 40-x, pero no hubo diferencias significativas para las instancias de tamaño 10-x. Se observa que la varianza de los resultados disminuye al aumentar el tamaño de problema lo que contribuye a diferenciar unos métodos de



(a) PCEPA 10-x (b) PCEPA 20-x (c) PCEPA 40-x

Figura 4.4: Resumen estadístico de las medias de fitness obtenidas al aplicar diferentes algoritmos metaheurísticos a varios tamaños del problema PCEPA (10-x, 20-x y 40-x). Se han considerado cuatro conjuntos de algoritmos en las figuras (a) y (b): la aplicación directa de las tres configuraciones hiperheurísticas (H), el conjunto de metaheurísticas obtenidas a partir de las hiperheurísticas (M-H) con computación del fitness agrupado en Fit1P1E y FitVPVE2, la aplicación de las mejores metaheurísticas no seleccionadas automáticamente (M-M), y el conjunto de las cuatro metaheurísticas puras (M-P). La figura (c) es igual pero con un conjunto más (M-H') que computa el fitness como FitVP1E.

otros. Debido a que los tamaños 10-x requieren de menos cómputo que los otros tamaños, es posible que los métodos de optimización tengan menos margen de mejora para diferenciarse unos de otros significativamente.

Se hizo un análisis más profundo entre los grupos dentro de cada conjunto de tamaños de problema. El test de Wilcoxon para dos muestras con corrección de continuidad se aplicó con un nivel de significación $\alpha = 0.05$. El algoritmo que presentó un mejor comportamiento para un conjunto de datos particular se indica con el símbolo $+$. Los algoritmos frente a los cuales es estadísticamente superior se indican con un $-$, y $\sim$ representa que no hubo diferencia en las medias. La tabla 4.17 muestra que el mejor algoritmo en todos los casos fue la hiperheurística (H) aplicada directamente a cada instancia, con los otros algoritmos siendo peores en todos los casos excepto las instancias M-M aplicadas a 20-x.

Como nuestro interés es comprobar la aplicabilidad de las metaheurísticas obtenidas automáticamente a partir de hiperheurísticas (M-H) frente a metaheurísticas obtenidas no automáticamente (M-M y M-P), se aplicó el test de Wilcoxon tomando M-H (o M-H' para FitVP1E) como grupo de control. La tabla 4.18 muestra que para todos los tamaños de problema el grupo M-H es mejor o igual que M-P, y comparado con M-M, iguala los resultados significativamente en, al menos, dos de tres casos. Es de destacar que el método de cálculo de fitness FitVP1E mejora los resultados obtenidos con la aplicación de Fit1P1E o FitVPVE2.

Tabla 4.17: Resultados del test de Wilcoxon para dos muestras con un nivel de significación $\alpha = 0.05$, para los grupos de tamaños de problema que mostraron diferencias significativas en sus respectivos subgrupos algorítmicos.

instancia	Fit (M-H)	M-M	H	M-H	M-P
20-x	1P1E & VPVE2	2564.06∼	2499.54+	2580.62−	2673.44−
40-x	1P1E & VPVE2	4773.33−	4653.52+	4965.53−	5382.86−
40-x	VP1E	4773.33−	4653.52+	4829.93−	5382.86−

Tabla 4.18: Resultados del test de Wilcoxon para dos muestras con un nivel de significación $\alpha = 0.05$, para comparar los grupos M-M y M-P con el grupo de referencia M-H.

instancia	Fit (M-H)	M-M	M-P
20-x	1P1E & VPVE2	∼	−
40-x	1P1E & VPVE2	+	∼
40-x	VP1E (M-H')	∼	−

4.2.2. Aplicación de hiperheurísticas a PCOCI

En el capítulo 2 se resumió el problema de la determinación de las constantes cinéticas de una reacción química, y en el capítulo 3 se aplicó el esquema metaheurístico directamente al problema. Continuamos aquí con la validación del método con la aplicación de hiperheurísticas. Las diferentes instancias de PCOCI usadas en los experimentos tienen la configuración de variables mostrada en la sección 3.4 (ver tabla 3.8).

Como se ha dicho, no estamos interesados aquí en el análisis de la influencia de los parámetros en la bondad de las soluciones, sino en el desarrollo de hiperheurísticas mediante el esquema parametrizado de metaheurísticas. En la subsección previa se analizaban en detalle varias maneras de aplicar hiperheurísticas a instancias del problema de optimización PCEPA. El propósito era contrastar que el algoritmo usado era efectivo no solo para conseguir buenos resultados de fitness y tiempo de ejecución en instancias específicas del problema, sino que también demostró ser útil para proporcionar buenas metaheurísticas (obtenidas a través de la aplicación de las diferentes configuraciones de la hiperheurística) para ser aplicadas a conjuntos de instancias del problema más generales. Algunas configuraciones han producido mejores resultados que otras en téminos de fitness, tiempo de ejecución y facilidad de aplicación del algoritmo (por ejemplo, es más costoso de implementar FitVPVE2 que FitVP1E). Considerando esto, la experimentación de la hiperheurística sobre

PCOCI se hace aplicando el método FitVP1E porque proporciona el mejor fitness, tiene menor dependencia del problema y más fácil aplicación. Además, se usan dos configuraciones de hiperheurísticas, Hge y Hre, debido a que los tiempos de experimentación son más pequeños y se ha visto que proporcionan resultados similares a los obtenidos al aplicar la hiperheurística híbrida (Hhi).

Se comentó que para solucionar eficientemente la dependencia de los resultados del problema, el fitness puede calcularse en las hiperheurísticas con el método FitVP1E (aplicación a varias instancias de problema en una sola ejecución). Es ese caso, en lugar de hacer varias ejecuciones de varias instancias del problema, solo se lleva a cabo una ejecución, y la instancia del problema se varía en cada iteración. La tabla 4.19 muestra los parámetros metaheurísticos obtenidos cuando se aplican diferentes configuraciones hiperheurísticas a tres instancias de PCOCI usando FitVP1E como criterio de cálculo del fitness. Se observa que:

- Hay una diferencia significativa en el número de elementos que constituyen el conjunto inicial ($NEIIni$), aunque el número de elementos seleccionados a partir del conjunto de referencia ($NEFIni$) es similar y pequeño.

- El porcentaje de elementos a mejorar en la inicialización ($PEMIni$) es cercano a cincuenta, con intensidades de mejora ($IMEIni$) pequeñas y valores intermedios de memoria a corto plazo tabú ($MCPIni$).

- Es destacable que la hiperheurística genética tiene algunos valores de los parámetros de combinación ($NMPCom$ y $NPPCom$) que son significativamente mayores que los correspondientes en la hiperheurística reducida.

- La mejora ($PEMMej$, $IMEMej$) y la diversificación ($PEDMej$, $IDEMej$) son importantes; valores intermedios de la memoria tabú ($MCMMej$, $MCDMej$, $MLPInc$) producen resultados satisfactorios; y existe prioridad en la selección de los mejores elementos del conjunto de referencia ($NEMInc$).

La figura 4.5 y la tabla 4.20 muestran los resultados de la aplicación de FitVP1E a tres instancias de problema usadas como test (series S4, S5 y S6). Las mejores metaheurísticas se obtuvieron a partir del conjunto de validación (S1, S2 y S3) en una ejecución para cada hiperheurística. En promedio, las metaheurísticas obtenidas con FitVP1E mejoran apreciablemente los resultados alcanzados con la aplicación de las metaheurísticas puras, y proporcionan valores de fitness próximos a los mejores obtenidos al aplicar la hiperheurística directamente a los problemas de test.

Los resultados obtenidos con las dos configuraciones hiperheurísticas con la metodología FitVP1E se comparan en la tabla 4.21. La tabla muestra la media del fitness, del tiempo secuencial y del indicador común obtenidos al aplicar la técnica a los tres problemas de validación S4, S5 y S6. El mejor indicador (IC) se encuentra en la Hre debido a que tiene el menor tiempo de ejecución, con valores de fitness similares.

	NEIIni	NEFIni	PEMIni	IMEIni	MCPIni	NEMSel
$Hre_{FitVP1E}$	24	17	45	2	12	6
$Hge_{FitVP1E}$	145	32	66	11	8	5

	NEPSel	NMMCom	NMPCom	NPPCom	PEMMej	IMEMej
$Hre_{FitVP1E}$	9	99	1	8	86	10
$Hge_{FitVP1E}$	21	71	77	62	72	14

	MCMMej	PEDMej	IDEMej	MCDMej	NEMInc	MLPInc
$Hre_{FitVP1E}$	11	86	4	9	16	9
$Hge_{FitVP1E}$	8	74	8	12	27	3

Tabla 4.19: Valores de los parámetros metaheurísticos obtenidos al aplicar las hiperheurísticas híbrida-reducida (Hre) y genética (Hge) a tres instancias de PCOCI, variando la instancia en cada iteración en la misma ejecución (FitVP1E).

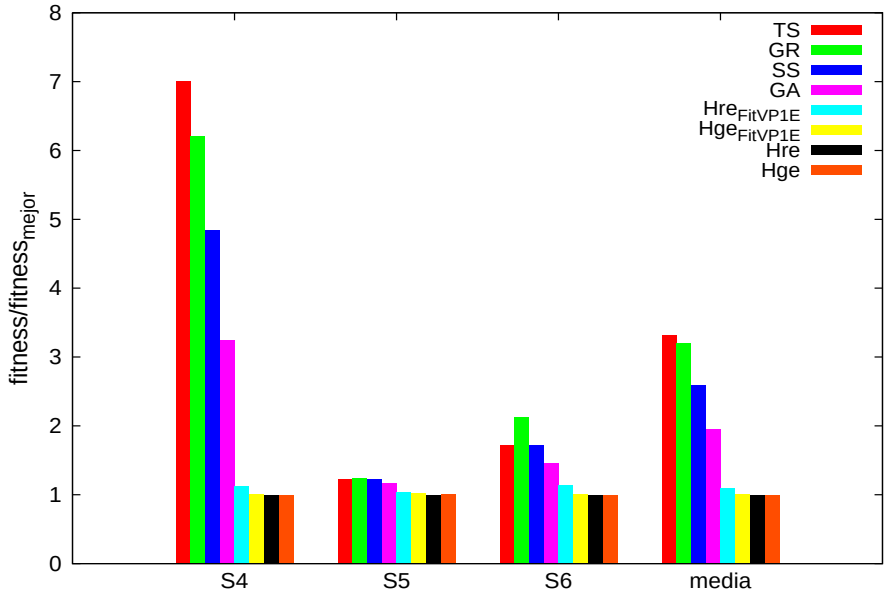


Figura 4.5: Cociente del fitness respecto al mejor fitness, para varios tamaños de PCOCI, para las cuatro metaheurísticas puras, para las metaheurísticas híbridas seleccionadas por las hiperheurísticas con problemas variables en una ejecución (FitVP1E), y para hiperheurísticas directamente aplicadas a instancias de cada problema.

	GR	TS	SS	GA
S4	7.003	6.212	4.838	3.249
S5	1.228	1.240	1.225	1.167
S6	1.724	2.130	1.716	1.456
Media	3.318	3.194	2.593	1.957

	$\text{Hre}_{\text{FitVP1E}}$	$\text{Hge}_{\text{FitVP1E}}$	Hre	Hge
S4	1.118	1.006	1.000	1.000
S5	1.034	1.018	1.000	1.003
S6	1.139	1.010	1.000	1.000
Media	1.097	1.011	1.000	1.001

Tabla 4.20: Cociente del fitness respecto al mejor fitness, para varios tamaños de PCOCI, para las cuatro metaheurísticas puras, para las metaheurísticas híbridas seleccionadas por las hiperheurísticas con problemas variables en una ejecución (FitVP1E), y para hiperheurísticas directamente aplicadas a instancias de cada problema.

		Hre	Hge
FitVP1E	(f)	2.1444	2.0348
	(t)	59.93	783.17
	(IC)	77.81	6.28

Tabla 4.21: Comparativa de las medias de fitness (f), tiempo (t) e indicador común ($IC = 10^5/(f \cdot t)$) obtenidas al aplicar las mejores metaheurísticas alcanzadas con FitVP1E para dos configuraciones hiperheurísticas a las instancias S4, S5 y S6 del problema PCOCI.

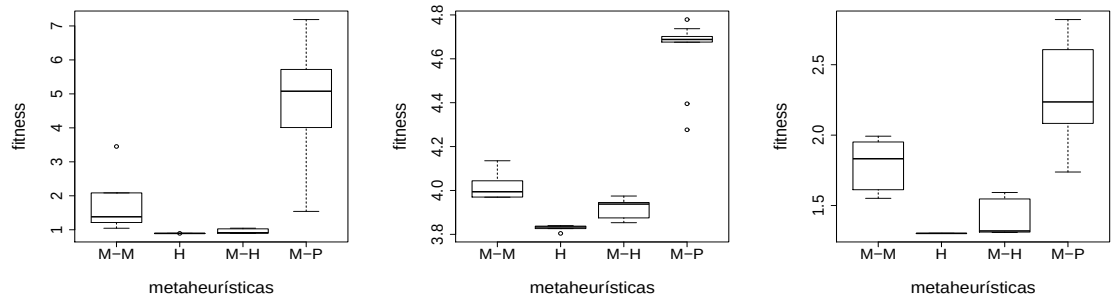
Como en el caso de PCEPA, se van a comparar las hiperheurísticas con resultados previos para ver si se mejora el fitness. En la última columna de la tabla 4.22 se presentan los resultados obtenidos al aplicar una de las mejores metaheurísticas híbridas (Mhi2 en la tabla 4.2) que se describieron en el capítulo 3 a las tres instancias de test de PCOCI. Se puede ver que en todos los casos las dos configuraciones hiperheurísticas mejoran esos resultados. Como en el problema previo, esto corrobora la necesidad de una hiperheurística como un buen método de búsqueda automática para ambos problemas individuales y para evitar dependencias del problema.

	Hge	Hre	Mhi2
S4	0.8937	0.8931	1.7600
S5	3.8209	3.8311	4.0178
S6	1.3023	1.3022	1.7953
Media	2.0055	2.0088	2.5243

Tabla 4.22: Comparativa del fitness obtenido al ejecutar dos configuraciones hiperheurísticas directamente a las tres instancias de test de PCOCI, con el fitness obtenido al aplicar una de las mejores metaheurísticas híbridas (Mhi2) a esos problemas.

Finalmente, como para PCEPA, en la figura 4.6, se presenta un resumen estadístico de las medias obtenidas al aplicar diferentes algoritmos metaheurísticos a varias instancias del problema PCOCI. Los resultados se agrupan en los mismos conjuntos que en PCEPA (H, M-H, M-M y M-P) para cada tamaño de problema con un número de puntos variable en cada conjunto que va de cuatro a doce. En este caso, M-H representa las metaheurísticas obtenidas a partir de las hiperheurísticas con computación de fitness FitVP1E, y M-M agrupa los resultados de la aplicación de la mejor metaheurística con parámetros Mhi2 en la tabla 4.2. Se consideró un número de puntos relativamente pequeño en algunos casos debido al elevado tiempo de ejecución de algunos algoritmos con varianzas pequeñas. El test de Kruskal-Wallis reveló diferencias estadísticas en las medias para los conjuntos de instancias en los tres tamaños de problema considerados.

El test de Wilcoxon para dos muestras con corrección de continuidad se aplicó con un nivel de significación $\alpha = 0.05$. La tabla 4.23 muestra que el mejor algoritmo en todos los tamaños de problema fue la hiperheurística (H) aplicada directamente a cada instancia, siendo los otros algoritmos peores en todos los casos. Además, existen diferencias significativas entre todas las parejas de algoritmos aplicados en cada tamaño de problema. Esto confirma estadísticamente los resultados presentados en la figura 4.5 e indica que las metaheurísticas obtenidas con las hiperheurísticas (M-H) superan en calidad los mejores resultados obtenidos con la mejor metaheurística (M-M) presentados en el capítulo 3.



(a) PCOCI S4 (b) PCOCI S5 (c) PCOCI S6

Figura 4.6: Resumen estadístico de las medias de fitness obtenidas al aplicar diferentes algoritmos metaheurísticos a varios tamaños del problema PCOCI (S4, S5 y S6). Se han considerado cuatro conjuntos de algoritmos: la aplicación directa de las configuraciones hiperheurísticas Hge y Hre (H), el conjunto de metaheurísticas obtenidas a partir de las hiperheurísticas (M-H) con computación del fitness FitVP1E, la aplicación de las mejores metaheurísticas no seleccionadas automáticamente (M-M), y el conjunto de las cuatro metaheurísticas puras (M-P).

Tabla 4.23: Resultados del test de Wilcoxon para dos muestras con un nivel de significación $\alpha = 0.05$, para los grupos de tamaños de problema considerados.

instancia	M-M	H	M-H	M-P
S4	1.7600−	0.8932+	0.9485−	4.7576−
S5	4.0178−	3.8291+	3.9206−	4.6414−
S6	1.7953−	1.3023+	1.3997−	2.2873−

4.3. Conclusiones

Se ha contrastado la utilidad y aplicabilidad de la metodología hiperheurística basada en esquemas parametrizados de metaheurísticas. Se ha visto la eficacia y la idoneidad de nuestra implementación hiperheurística para resolver el problema POPME a un nivel superior de abstracción cada vez que se aplicaba a cualquier problema de optimización.

Primeramente se han definido tres configuraciones de hiperheurísticas que se han usado para experimentar con los problemas de optimización planteados. Además, se han estudiado varias implementaciones de la función de combinación, decidiéndose que la combinación por grupos es la más adecuada a la vista de los resultados obtenidos. Se ha medido la eficacia de una metaheurística aplicada a un problema en términos de un indicador común calculado como la inversa del producto del fitness y el tiempo de ejecución.

Respecto al problema PCEPA, podemos decir que los mejores resultados en términos del indicador común se han obtenido con la aplicación de una hiperheurística híbrida y el método de cálculo del fitness FitVP1E. Los valores de fitness obtenidos al aplicar las metaheurísticas obtenidas a partir de las instancias de aprendizaje sobre otras instancias de test fueron mejores que los alcanzados por las metaheurísticas puras, y próximos a los teóricamente mejores obtenidos al aplicar directamente las hiperheurísticas a cada problema. No obstante, se ha visto también que se pueden conseguir resultados aceptables aplicando la hiperheurística genética reduciéndose considerablemente el tiempo de ejecución. Además, se ha comprobado que la hiperheurística arroja mejores resultados que cualquiera de las metaheurísticas aplicadas directamente a los problemas. En este sentido, no sólo se consigue un algoritmo bueno para conjuntos de problemas (evitando dependencias con éstos) sino que también es eficaz para instancias individuales.

Por otro lado, tenemos algunos resultados interesantes al aplicar hiperheurísticas al problema PCOCI. En este caso, debido al elevado coste computacional que requería, solo se experimentó con las configuraciones hiperheurísticas genética y reducida. Es esta última la que arroja mejores resultados medidos en términos del indicador común debido principalmente a los tiempos de ejecución bajos que se tienen. Puesto que se vio para el problema anterior que el método de cálculo del fitness más efectivo es FitVP1E, solo se ha experimentado con este método pues ahorra tiempo de ejecución y evita a la vez dependencias del problema. Cabe destacar que las metaheurísticas obtenidas a partir de hiperheurísticas arrojan resultados significativamente mejores que el resto de configuraciones metaheurísticas consideradas y están muy próximos a los teóricamente mejores derivados de la aplicación directa de las hiperheurísticas al problema. Finalmente, decir que en este caso la hiperheurística ha demostrado ser también una buena herramienta para la obtención de la mejor metaheurística, tanto para problemas individuales como para conjuntos de problemas.

Capítulo 5

Modelado y autooptimización de metaheurísticas e hiperheurísticas paralelas en memoria compartida

Se analiza aquí la introducción de paralelismo en memoria compartida para acelerar la aplicación de las metaheurísticas e hiperheurísticas. Se va a ampliar la estructura del algoritmo a un esquema parametrizado paralelo. Esto significa que se van a considerar otro tipo de parámetros (paralelos) en el esquema, para controlar los niveles y la intensidad del paralelismo establecido. Se persigue reducir el tiempo de ejecución de los algoritmos. Se pretende establecer un número óptimo de hilos a ejecutar con la inclusión de la metodología de autooptimización orientada a minimizar el tiempo de ejecución. Así, se procederá a desarrollar modelos del tiempo de ejecución para cada una de las rutinas que forman el esquema unificado parametrizado paralelo, para posteriormente determinar las constantes características de dichos modelos en los sistemas donde van a ser ejecutados (proceso de instalación). Finalmente, se procederá a la ejecución de las metaheurísticas con unos parámetros de paralelismo optimizados para el sistema concreto y dependientes de los parámetros metaheurísticos y de las constantes del modelo. La metodología de autooptimización será probada en varios sistemas, sobre el problema de minimización de costes eléctricos en la explotación de pozos de agua (PCEPA) y sobre el problema de optimización de metaheurísticas (POPME), utilizando varios niveles de paralelismo para poder sacar conclusiones lo más generales posibles. Se aplicará autooptimización a dos niveles: por un lado a las metaheurísticas aplicadas directamente al problema y, por otro, se autooptimizará el esquema hiperheurístico que controla la ejecución de metaheurísticas a nivel superior.

5.1. Esquemas parametrizados en memoria compartida de metaheurísticas e hiperheurísticas

Las ideas sobre un esquema unificado parametrizado de metaheurísticas se presentaron en el capítulo 3 de esta memoria. En esta sección se ampliará el concepto introduciendo nuevos parámetros (de paralelismo) en el esquema, transformándose en un esquema metaheurístico en memoria compartida.

El interés del uso de un esquema metaheurístico general es doble. El nivel de abstracción involucrado permite analizar y visualizar las metaheurísticas desde una perspectiva genérica debido a que el mismo esquema representaba varias metaheurísticas, y el esquema general puede verse como un mecanismo para reusar elementos entre las diferentes técnicas.

La selección de valores apropiados de los parámetros metaheurísticos para aplicar una metaheurística satisfactoria a un problema particular podía ser difícil y es costosa computacionalmente. La selección de esos valores se podía hacer por medio de un método hiperheurístico también desarrollado con el esquema parametrizado de metaheurísticas. De esta forma, en el capítulo 4, la hiperheurística se definía como un método inteligente de selección de la metaheurística o algoritmo adecuado para un problema dado. Por tanto la fortaleza del método se encuentra en su capacidad de tomar buenas decisiones en el sentido de encontrar una buena metaheurística.

Por simplificar la notación, puesto que este capítulo no se centra en los efectos de las diferentes configuraciones internas del esquema metaheurístico e hiperheurístico sobre la bondad de los resultados, sino en paralelizar los algoritmos, de aquí en adelante se hará referencia al esquema metaheurístico directamente aplicado al problema de optimización como EM, y HEM se referirá a una hiperheurística basada en un esquema metaheurístico para la selección adecuada de los valores de los parámetros metaheurísticos.

Al ejecutar una hiperheurística, se aplican muchas metaheurísticas a diferentes instancias de problema, dando como resultado tiempos de ejecución elevados, y es necesario usar paralelismo. Se pueden usar metaheurísticas paralelas para reducir el tiempo de ejecución, pero también es posible, y preferible, usar paralelismo en un nivel superior, en el cual se usa el esquema metaheurístico parametrizado en memoria compartida para la hiperheurística, y son válidas las mismas técnicas de autooptimización para las metaheurísticas y la hiperheurística.

5.1.1. Esquemas parametrizados en memoria compartida

En nuestra técnica, el esquema parametrizado del Algoritmo 3 se convierte en un esquema parametrizado en memoria compartida paralelizando independientemente cada función básica del esquema (Algoritmo 4) con nuevos *parámetros de paralelismo* ($HilosX$ en el Algoritmo 4) indicando el número de hilos a usar en cada parte del algoritmo.

Se usa aquí un esquema parametrizado paralelo para aplicar una técnica común de autooptimización para seleccionar el número óptimo de hilos obteniéndose tiempos de ejecución bajos. Al desarrollar hiperheurísticas con el mismo esquema usado para las metaheurísticas, son aplicables las mismas técnicas de paralelización [2]. Se pueden identificar dos esquemas básicos:

Algoritmo 4 Esquema parametrizado en memoria compartida de metaheurísticas.

```
Inicializar(S,ParamIni,HilosIni)
Mientras (no CondiciónDeFin(S,ParamFin)) Hacer
  SS=Seleccionar(S,ParamSel)
  SS1=Combinar(SS,ParamCom,HilosCom)
  SS2=Mejorar(SS1,ParamMej,HilosMej)
  S=Incluir(SS2,ParamInc,HilosInc)
Fin Mientras
```

- En el primer esquema los elementos de un conjunto se tratan independientemente, y se selecciona el número de hilos a trabajar en un bucle. Este esquema aparece, por ejemplo, al combinar elementos en un algoritmo genético o al generar aleatoriamente un conjunto inicial de elementos. Así, $HilosIni$ y $HilosCom$ contienen un parámetro de paralelismo indicando el número de hilos a usar en la generación del conjunto inicial y en la combinación de los elementos seleccionados, y estos valores pueden ser diferentes, por lo que se pueden obtener valores diferentes de los parámetros de paralelismo en cada función.

- El segundo esquema tiene dos niveles de paralelismo y se puede usar para obtener paralelismo de grano fino. Se establece el número de hilos en cada nivel de paralelismo. Este tipo de paralelismo aparece en las funciones de mejora y diversificación, donde algunos elementos se seleccionan (primer nivel) y cada elemento se mejora analizando su vecindario (segundo nivel).

El número de hilos (un valor o varios) se establece para cada función en el esquema parametrizado en memoria compartida. El número de parámetros de paralelismo para cada función depende de la implementación particular de las funciones del esquema unificado, pero la metodología es común a varias metaheurísticas e implementaciones paralelas. Por ejemplo, algunas metaheurísticas incluyen una parte de mejora en la inicialización, y el número de hilos en los dos niveles de esta mejora se añaden al número de hilos de la inicialización del conjunto de referencia. En la función de mejora hay dos subgrupos de parámetros paralelos con la misma estructura, uno para la intensificación y otro para la diversificación de elementos. Los parámetros de paralelismo en cada función son:

- `Inicializar`: Se utiliza un bucle *for* para generar el conjunto inicial de elementos, por lo que se usa un esquema de un nivel, con un número de hilos

$HGEIni$. A continuación aparece un esquema de dos niveles de paralelismo con la mejora de los elementos generados, con dos parámetros ($HM1Ini$ y $HM2Ini$) para el número de hilos. Por lo que $HilosIni = \{HGEIni, HM1Ini, HM2Ini\}$.

- `Combinar`: Se combinan parejas de elementos en un bucle con un nivel de paralelismo, por lo que tenemos $HCPCom$ hilos para trabajar con el bucle de combinaciones, e $HilosCom = \{HCPCom\}$.

- `Mejorar`: Como en la mejora de la inicialización, se usan funciones de dos niveles de paralelismo para la mejora y diversificación de elementos. Por lo que tenemos cuatro parámetros de paralelismo: $HilosMej = \{HM1Mej, HM2Mej, HD1Mej, HD2Mej\}$.

- `Incluir`: Se ha considerado una paralelización de un nivel con $HilosInc = \{HIEInc\}$ hilos para la inclusión de elementos.

El esquema metaheurístico se usa en dos niveles: para la hiperheurística (HEM) y para la aplicación de las metaheurísticas determinadas por medio de los parámetros metaheurísticos (*ParamMetaheur*) en cada elemento del conjunto de referencia de la hiperheurística (EM) al problema de optimización dado. Por tanto, se puede aplicar paralelismo en la hiperheurística y en las metaheurísticas, con un total de cuatro niveles de paralelismo, pero es preferible paralelizar en el nivel más alto, y normalmente se aplica paralelismo solo a la hiperheurística.

5.2. La metodología de modelado y autooptimización

Para reducir el tiempo de ejecución es necesario seleccionar los valores de los parámetros de paralelismo ($HilosIni$, $HilosCom$, $HilosMej$ e $HilosInc$) adecuadamente. Para hacerlo, se obtiene un modelo teórico del tiempo de ejecución para cada función y se establece el número de hilos en algunos bucles o el número de hilos del primer y segundo nivel de paralelismo.

El proceso de autooptimización usado en [48] en rutinas de algebra lineal se adapta aquí al esquema metaheurístico. Se utilizan el problema de minimización de consumo eléctrico en la explotación de pozos de agua (PCEPA) y el problema de optimización de metaheurísticas (POPME) para mostrar la metodología de modelado y autooptimización. Para contrastar de manera clara el efecto del paralelismo sobre el esquema metaheurístico (EM), se ha elegido un tamaño del problema PCEPA relativamente grande (200-24), con valores de las variables iguales a las del tamaño 200-6 en la tabla 3.3, con el mismo número de bombas (B) pero con un rango horario (R) más amplio. En contraste, para aplicar el HEM, se ha elegido un tamaño de problema más bien reducido (20-3) debido al elevado tiempo necesario

para las ejecuciones de la hiperheurística. Los valores de las variables de este tamaño son similares a las del PCEPA 20-6. Las metaheurísticas utilizadas para mostrar el funcionamiento de la metodología son las mismas que en capítulos anteriores: Algoritmos Genéticos, Búsqueda Dispersa, GRASP y Búsqueda Tabú. No obstante, se podría aplicar la misma metodología con otros conjuntos de metaheurísticas básicas, implementaciones de las funciones básicas en el esquema metaheurístico y problema de optimización a resolver. El proceso se divide en tres fases:

- **Diseño:** La rutina se desarrolla junto con su tiempo de ejecución teórico. Se obtiene un tiempo de ejecución teórico para cada rutina básica del Algoritmo 4 como una función de los parámetros metaheurísticos y de paralelismo (número de hilos a usar en cada rutina y subrutina). Debido a que se identifican dos tipos de paralelismo, se pueden usar dos tipos de modelos, uno para rutinas de un nivel y otro para paralelismo anidado. Para rutinas de un nivel, donde se tratan NE elementos (generados, combinados o evaluados) el modelo es:

$$t_{un-nivel} = \frac{k_g \cdot NE}{p} + k_p \cdot p \tag{5.1}$$

  donde k_g representa el coste de la generación de un individuo; k_p el coste de generación de un hilo; y p es el número de hilos.

  Para las rutinas de dos niveles en las funciones de mejora, con un número de elementos NE, una probabilidad de mejora PM y una intensificación IM, el modelo es:

$$t_{dos-niveles} = \frac{k_m \cdot NE \cdot PM \cdot IM}{100 \cdot p_1 \cdot p_2} + k_{p,1} \cdot p_1 + k_{p,2} \cdot p_2 \tag{5.2}$$

  donde k_m representa el coste de mejorar un elemento; $k_{p,1}$ y $k_{p,2}$ el coste de generar hilos en el primer y segundo nivel; y p_1 y p_2 el número de hilos de cada nivel.

  Para cada función del esquema, el modelo se obtiene sustituyendo los valores de NE, PM e IM por los correspondientes parámetros metaheurísticos de la función. Los modelos así obtenidos son muy simples y no consideran algunos aspectos arquitectónicos, como la asignación de memoria o de hilos, pero su simplicidad facilita su uso, permitiendo obtener resultados satisfactorios. Además, en algunos ambientes de ejecución estos aspectos de la arquitectura del sistema no pueden ser considerados cuando se ejecuta el código; por ejemplo, cuando se envía el trabajo a una cola, el sistema decide los cores donde se mapean los hilos y la asignación de datos.

- **Instalación:** Cuando se ha instalado el esquema parametrizado en memoria compartida en un sistema particular, se estima el valor de los parámetros que dependen del sistema. Se estiman los valores de los parámetros k_g, k_m, k_p, $k_{p,1}$

y $k_{p,2}$ presentados en el punto anterior para cada una de las rutinas básicas del esquema. La estimación se puede hacer por medio de la experimentación con cada función básica de la metaheurística, para algunos valores de los parámetros metaheurísticos y de paralelismo ($NEIIni$ y p en la ecuación 5.1 para la generación inicial de elementos; y $NEIIni$, $PEMIni$ y $IMEIni$, y p_1 y p_2 en la ecuación 5.2 para la mejora de los elementos iniciales) y con un ajuste por mínimos cuadrados.

Los experimentos se han llevado a cabo en dos sistemas computacionales diferentes: *Ben* y *Saturno*. Las principales diferencias entre ambos son el tamaño y la estructura, lo que produce diferencias en las latencias de acceso a la memoria compartida debido a que tienen arquitecturas diferentes.

Primeramente, se resumen los resultados de la instalación del esquema en *Ben*. El número óptimo de hilos varía con el número de elementos, y estamos interesados en seleccionar un número de hilos cercano al óptimo a partir de un reducido número de elementos (para tiempos de instalación bajos). Como ejemplo, para rutinas de un nivel se usa el modelo de la ecuación 5.1 en la inicialización de una hiperheurística, y se obtienen los parámetros k_g y k_p del modelo por medio de mínimos cuadrados con $NE = NEIIni = 5$. Los valores obtenidos son $k_g = 0.577$ y $k_p = 0.0491$. Para una rutina de dos niveles, como la rutina para mejorar elementos después de la generación inicial o tras la combinación o la diversificación, los valores de los parámetros de paralelismo se obtienen por mínimos cuadrados con experimentos con parámetros para la hiperheurística $NE = NEIIni = 10$, $PM = PEMIni = 100$ y $IM = IMEIni = 1$. Los resultados son $k_m = 1.21$, $k_{p,1} = 0.104$ y $k_{p,2} = 0.0989$. Al sustituir estos valores en el modelo teórico del tiempo de ejecución (ecuación 5.2), el comportamiento de la rutina en el sistema se predice bien, como se puede ver en la figura 5.1, donde se representan los speed-ups experimental y teórico para la mejora de la población inicial para la combinación de parámetros hiperheurísticos $NE = NEIIni = 50$, $PM = PEMIni = 50$ y $IM = IMEIni = 1$.

- **Ejecución:** En tiempo de ejecución el número de hilos de cada función básica se selecciona a partir del tiempo de ejecución teórico (ecuaciones 5.1 y 5.2) con los valores de los parámetros metaheurísticos de las metaheurísticas (o hiperheurísticas) con las que estamos experimentando y con los valores de los parámetros del sistema estimados en la fase de instalación. El número de hilos que proporciona el tiempo de ejecución mínimo teórico se obtiene minimizando la correspondiente ecuación después de sustituir en ella los valores de los parámetros meta o hiperheurísticos y del sistema.

  Después de sustituir en el modelo teórico los valores de las constantes estimadas experimentalmente y por mínimos cuadrados, se obtienen los valores de los parámetros de paralelismo que proporcionan el tiempo teórico más bajo como una función de los parámetros metaheurísticos, por lo que las ecuaciones son

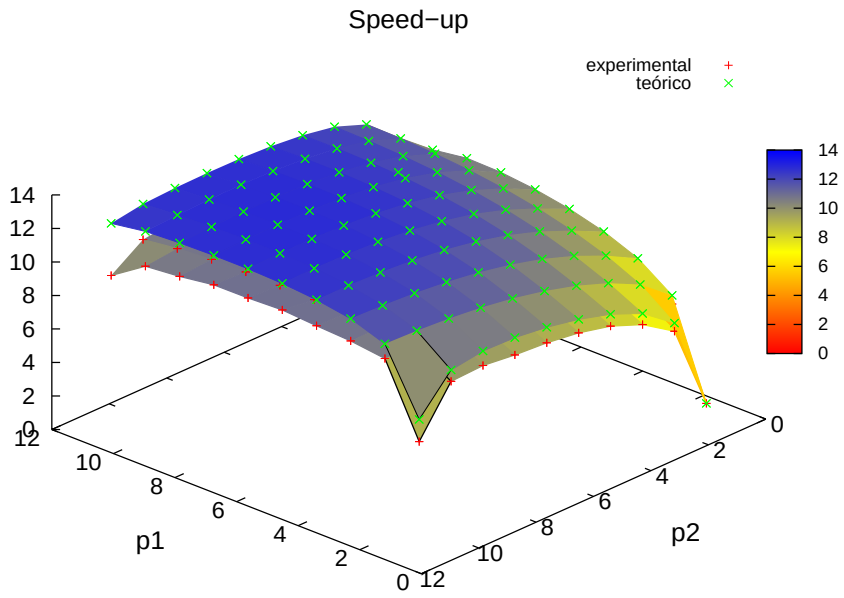


Figura 5.1: Speed-ups teórico y experimental variando el número de hilos del primer y segundo nivel de paralelismo en una rutina paralela de dos niveles al aplicar la hiperheurística basada en el esquema metaheurístico (HEM)
a PCEPA en *Ben*.

válidas para las diferentes metaheurísticas o hiperheurísticas en el sistema computacional particular donde se instale el esquema.

Para la generación inicial del conjunto de referencia, para los experimentos realizados en nuestro sistema, el número de hilos que proporciona el tiempo de ejecución teórico más bajo es:

$$p_{opt} = \sqrt{\frac{k_g}{k_p} \cdot NEIIni} = 3.43 \cdot \sqrt{NEIIni} \tag{5.3}$$

y para la mejora de los elementos generados el número óptimo de hilos en la función de dos niveles es:

$$p_{1,opt} = 0.479 \cdot \sqrt[3]{NEIIni \cdot PEMIni \cdot IMEIni} \tag{5.4}$$

$$p_{2,opt} = 0.505 \cdot \sqrt[3]{NEIIni \cdot PEMIni \cdot IMEIni} \tag{5.5}$$

5.3. Aplicación de la metodología a PCEPA y POPME

Para validar la metodología de autooptimización, se calculan el número óptimo de hilos y el speed-up máximo alcanzado a partir de los modelos para diferentes parámetros metaheurísticos utilizando los parámetros del sistema obtenidos en la instalación. Estos parámetros del sistema se calcularon usando valores pequeños de los parámetros metaheurísticos para reducir el tiempo de instalación. En la aplicación del EM, los valores usados en la instalación son $NEIIni = 20$ para la inicialización y $NEIIni = 20$, $PEMIni = 50$, $IMEIni = 20$ para la mejora en la inicialización. En el HEM esos valores son $NEIIni = 5$, $NEIIni = 10$, $PEMIni = 100$ y $IMEIni = 1$. Las diferencias entre los valores de los parámetros de EM y HEM se explican por el elevado tiempo de ejecución de la hiperheurística, lo que hace necesario tener valores más bajos para unos tiempos de instalación moderados.

Puesto que se han utilizado metaheurísticas e hiperheurísticas para la optimización del problema PCEPA, los experimentos tienen en cuenta implementaciones de las funciones básicas con parámetros metaheurísticos con costes computacionales diferentes.

Las tablas 5.1 y 5.2 comparan los resultados obtenidos en la generación inicial del conjunto de referencia y en la mejora de elementos para dos combinaciones de parámetros aplicando el EM en *Ben* usando 128 cores. Si el número de hilos seleccionado por el modelo es superior a 128, se ponen en marcha el número máximo de hilos disponible en *Ben*, es decir, 128. El número de hilos seleccionado con la

metodología de autooptimización no está lejos de los mejores valores obtenidos experimentalmente y, como consecuencia, el speed-up alcanzado con autooptimización no está lejos del máximo experimental y la metodología es útil para la reducción del tiempo de ejecución de las metaheurísticas.

	hilos		speed-up		
$NEIIni$	exp	mod	exp	mod	exp-auto
100	48	55	27	27	25
500	121	122	77	61	75

Tabla 5.1: Comparativa del speed-up experimental más alto (exp), el speed-up modelado (mod) y el obtenido en los experimentos al usar la metodología de autooptimización (exp-auto), para $NEIIni = 100$ y 500, en la rutina paralela de un nivel de `Inicializar`, al aplicar el esquema metaheurístico (EM)
a PCEPA en *Ben*.

			hilos		speed-up		
$NEIIni$	$PEMIni$	$IMEIni$	exp	mod	exp	mod	exp-auto
100	50	10	89	67	35	17	21
500	100	5	128	150	78	51	78

Tabla 5.2: Comparativa del speed-up experimental más alto (exp), el speed-up modelado (mod) y el obtenido en los experimentos al usar la metodología de autooptimización (exp-auto), para otras combinaciones de parámetros en la rutina paralela de dos niveles de `Inicializar`, al aplicar el EM a PCEPA en *Ben*.

Se pueden comparar los resultados obtenidos al aplicar directamente las metaheurísticas individuales al problema de optimización considerado con aquellos obtenidos con la hiperheurística usando la metodología de autooptimización. Como el esquema metaheurístico es el mismo, son de esperar resultados similares en ambos casos, aunque puede haber diferencias debido a implementaciones diferentes. Por ejemplo, en la función de mejora del EM, el segundo nivel se usó para poner en marcha más hilos para trabajar en la mejora de la función de fitness (se analizan más vecinos) pero no para reducir el tiempo de ejecución, con lo cual el número de hilos del segundo nivel se puede tomar como constante. Así, en este caso el modelo es ligeramente diferente. En la función de generación inicial de elementos no hay diferencias en la implementación. Se ha predicho bien el comportamiento de la rutina de un nivel al aplicar el EM, como puede verse en la figura 5.2, donde se representan los speed-ups teórico y experimental.

Las tablas 5.3 y 5.4 comparan los resultados obtenidos en la generación inicial del conjunto de referencia y en la mejora de elementos para dos combinaciones de

parámetros usando el HEM en *Ben*. Como en el caso del EM, el número de hilos y el speed-up seleccionados con la técnica de autooptimización no están muy distantes de los mejores valores obtenidos experimentalmente, por lo que la metodología de autooptimización se muestra útil para la reducción del tiempo de ejecución de las hiperheurísticas, las cuales tienen un elevado coste debido a que aplican un gran número de metaheurísticas. Se puede ver que la técnica aplicada en EM es también válida para HEM.

	hilos		speed-up		
$NEIIni$	exp	mod	exp	mod	exp-auto
20	22	15	11	8	8
100	24	34	12	17	12

Tabla 5.3: Comparativa del speed-up experimental más alto (exp), el speed-up modelado (mod) y el obtenido en los experimentos al usar la metodología de autooptimización (exp-auto), para $NEIIni = 20$ y 100, en la rutina paralela de un nivel de `Inicializar`, al aplicar el HEM a PCEPA en *Ben*.

	hilos		speed-up		
Comb.	exp	mod	exp	mod	exp-auto
c1	9×8	6×7	14	15	11
c2	9×4	8×9	15	24	14

Tabla 5.4: Comparativa del speed-up experimental más alto (exp), el speed-up modelado (mod) y el obtenido en los experimentos al usar la metodología de autooptimización (exp-auto), para dos combinaciones de $NEIIni$, $PEMIni$ y $IMEIni$ (c1: 50,50,1; c2: 100,50,1), en la rutina paralela de dos niveles de `Inicializar`, al aplicar el HEM a PCEPA en *Ben*.

Las ventajas de usar autooptimización se pueden ver más claramente al comparar el speed-up obtenido al lanzar el número máximo de hilos disponible en el sistema y la mitad de este máximo (como primera aproximación a la selección de hilos), con los resultados alcanzados al seleccionar el número de hilos de cada nivel con nuestra técnica de autooptimización. Los resultados se presentan en las tablas 5.5 a 5.8. Tanto en el caso de la aplicación directa del EM al problema de optimización como aplicando el HEM, el speed-up alcanzado con el modelo es casi siempre mejor que el obtenido al ejecutar de manera no óptima el máximo número de hilos disponible o su mitad, y se acerca al valor óptimo experimental en muchos casos.

Hasta aquí se ha comprobado la validez de la metodología de autooptimización en un sistema grande como *Ben*, donde el acceso a la memoria compartida puede

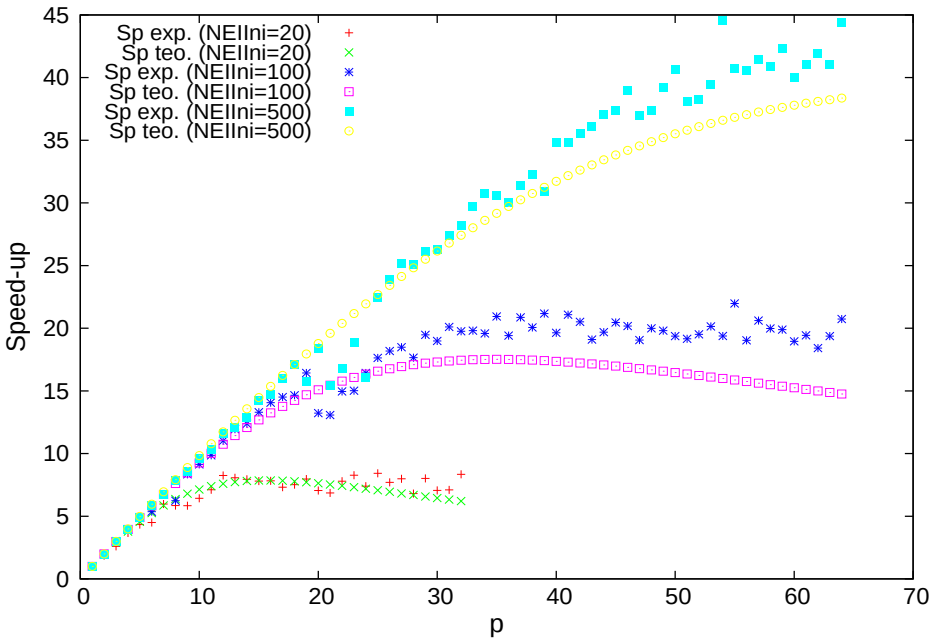


Figura 5.2: Speed-ups teórico y experimental variando el número de hilos para tres valores del parámetro $NEIIni$ en la rutina paralela de un nivel de `Inicializar` al aplicar el EM a PCEPA en *Ben*.

	hilos				speed-up			
$NEIIni$	max	$\frac{max}{2}$	exp	mod	max	$\frac{max}{2}$	exp	exp-auto
100	128	64	48	55	20	23	27	25
500	128	64	121	122	73	49	77	75

Tabla 5.5: Comparativa del speed-up obtenido con el número máximo de hilos disponible (max), con la mitad del número máximo de hilos disponible ($\frac{max}{2}$), el speed-up experimental más alto (exp) y el experimental obtenido con nuestra metodología de autooptimización (exp-auto) y usando el número de hilos óptimo obtenido a partir del modelo (mod), para $NEIIni = 100$ y 500, en la rutina paralela de un nivel de `Inicializar`, al aplicar el EM a PCEPA en *Ben*.

Comb.	hilos max	hilos $\frac{max}{2}$	hilos exp	hilos mod	speed-up max	speed-up $\frac{max}{2}$	speed-up exp	speed-up exp-auto
c1	128	64	89	67	27	15	35	21
c2	128	64	128	150	78	52	78	78

Tabla 5.6: Comparativa del speed-up obtenido con el número máximo de hilos disponible (max), con la mitad del número máximo de hilos disponible ($\frac{max}{2}$), el speed-up experimental más alto (exp) y el experimental obtenido con nuestra metodología de autooptimización (exp-auto) y usando el número de hilos óptimo obtenido a partir del modelo (mod), para dos combinaciones de $NEIIni$, $PEMIni$ y $IMEIni$ (c1: 100,50,10; c2: 500,100,5), en la rutina paralela de dos niveles de `Inicializar`, al aplicar el EM a PCEPA en *Ben*.

$NEIIni$	hilos max	hilos $\frac{max}{2}$	hilos exp	hilos mod	speed-up max	speed-up $\frac{max}{2}$	speed-up exp	speed-up exp-auto
20	128	64	22	15	11	11	11	8
100	128	64	24	34	11	11	12	12

Tabla 5.7: Comparativa del speed-up obtenido con el número máximo de hilos disponible (max), con la mitad del número máximo de hilos disponible ($\frac{max}{2}$), el speed-up experimental más alto (exp) y el experimental obtenido con nuestra metodología de autooptimización (exp-auto) y usando el número de hilos óptimo obtenido a partir del modelo (mod), para $NEIIni = 20$ y 100, en la rutina paralela de un nivel de `Inicializar`, al aplicar el HEM a PCEPA en *Ben*.

Comb.	hilos un-nivel $\sqrt{max}$	hilos un-nivel $\sqrt{\frac{max}{2}}$	hilos un-nivel exp	hilos un-nivel mod	speed-up $\sqrt{max}$	speed-up $\sqrt{\frac{max}{2}}$	speed-up exp	speed-up exp-auto
c1	11×11	8×8	9×8	6×7	9	11	14	11
c2	11×11	8×8	9×4	8×9	11	12	15	14

Tabla 5.8: Comparativa del speed-up obtenido con el número máximo de hilos disponible (max), con la mitad del número máximo de hilos disponible ($\frac{max}{2}$), el speed-up experimental más alto (exp) y el experimental obtenido con nuestra metodología de autooptimización (exp-auto) y usando el número de hilos óptimo obtenido a partir del modelo (mod), para dos combinaciones de $NEIIni$, $PEMIni$ y $IMEIni$ (c1: 50,50,1; c2: 100,50,1), en la rutina paralela de dos niveles de `Inicializar`, al aplicar el HEM a PCEPA en *Ben*.

suponer un retardo adicional en el tiempo de ejecución. Para poder obtener conclusiones más generales, se ha analizado el proceso de autooptimización completo en *Saturno*. Las tablas 5.9 y 5.10 detallan los valores de las constantes del modelo obtenidas en la fase de instalación en *Saturno* para el EM y el HEM.

	Rutinas paralelas de un nivel			Rutinas paralelas de dos niveles		
	Ini	Com	Inc	Mej-Ini	Mej	Div
$k_s \cdot 10^4$	4.56	5.72	5.60	6.05	6.01	56.8
$k_{p,1} \cdot 10^4$	0.482	1.76	12.2	3.16	2.31	12.2
$k_{p,2} \cdot 10^4$	-	-	-	-0.05	37.1	-38.9

Tabla 5.9: Valores de las constantes del modelo para todas las funciones al aplicar el EM a PCEPA en *Saturno*.

	Rutinas paralelas de un nivel			Rutinas paralelas de dos niveles		
	Ini	Com	Inc	Mej-Ini	Mej	Div
$k_s \cdot 10^2$	1.47	2.95	0.302	26.2	26.2	26.6
$k_{p,1} \cdot 10^2$	0.096	0.593	0.465	3.27	2.43	2.07
$k_{p,2} \cdot 10^2$	-	-	-	2.74	1.68	3.56

Tabla 5.10: Valores de las constantes del modelo para todas las funciones al aplicar el HEM a PCEPA en *Saturno*.

Teniendo en cuenta los valores de estas constantes, se puede verificar la validez de la metodología de autooptimización ejecutando todas las funciones del esquema metaheurístico de forma secuencial. La tabla 5.11 presenta los valores de los parámetros metaheurísticos típicos usados en el EM y en el HEM. Se debe tener en cuenta que hay diferencias en el tiempo de ejecución entre metaheurísticas e hiperheurísticas. Mientras que la metaheurística ejecuta instancias de problema directamente, la hiperheurística es más costosa en cuanto a tiempo, debido a que ejecuta diferentes metaheurísticas al mismo tiempo para optimizar la resolución del problema. Por tanto, se ha elegido estos valores de los parámetros porque producen metaheurísticas e hiperheurísticas de tamaño intermedio, permitiendo un estudio exhaustivo en un tiempo relativamente reducido.

La tabla 5.12 muestra el número de hilos óptimo del primer y segundo nivel de paralelismo obtenidos con autooptimización para todas las funciones del esquema y para cuatro combinaciones de parámetros metaheurísticos en *Saturno*. Se presentan los resultados de la aplicación del EM y del HEM. El valor del parámetro del segundo nivel se ha fijado a 1 en el caso del EM porque el segundo nivel se usa para lanzar más hilos para trabajar en la mejora de la función de fitness pero no para reducir

	NEIIni	NEFIni	PEMIni	IMEIni	MCPIni	NEMSel
m1	75	50	75	20	-	10
m2	150	100	25	15	-	25
h1	10	10	100	1	7	5
h2	20	20	50	3	7	10

	NEPSel	NMMCom	NMPCom	NPPCom	PEMMej	IMEMej
m1	10	50	45	50	100	20
m2	25	100	90	100	40	10
h1	5	20	5	10	100	1
h2	10	50	10	5	20	3

	MCMMej	PEDMej	IDEMej	MCDMej	NEMInc	MLPInc
m1	-	15	15	-	10	-
m2	-	15	10	-	50	-
h1	7	100	1	7	10	7
h2	7	20	3	7	10	7

Tabla 5.11: Valores de los parámetros metaheurísticos usados para los experimentos de autooptimización al aplicar el EM (metaheurísticas m1 y m2) y el HEM (hiperheurísticas h1 y h2) a PCEPA en *Saturno*.

el tiempo de ejecución. Como se aprecia en la tabla, en muchos casos el número de hilos sugerido por el modelo sobrepasa el número de cores disponible, por lo que, en estos casos, se ha optado por lanzar el número máximo de 24 hilos.

Finalmente, en la tabla 5.13 se puede ver el speed-up alcanzado con el número óptimo de hilos dado por el modelo, y éste se puede comparar con el valor obtenido al seleccionar los parámetros de paralelismo sin un método sistemático. Se consideran el número máximo de hilos disponible y la mitad del número máximo de hilos disponible. Se podría elegir otro número de hilos, pero esta es una buena primera aproximación para hacer una comparación con nuestra metodología de autooptimización.

	Rutinas paralelas de un nivel			Rutinas paralelas de dos niveles		
	HGEIni	HCPCom	HIEInc	HM1Ini	HM1Mej	HD1Mej
m1	27	22	12	46	51	23
m2	38	31	17	33	32	26

	nivel	Rutinas paralelas de un nivel			Rutinas paralelas de dos niveles		
		HGEIni	HCPCom	HIEInc	HM_Ini	HM_Mej	HD_Mej
h1	p_1	12	13	7	5	4	6
	p_2	-	-	-	4	6	4
h2	p_1	17	18	10	6	4	6
	p_2	-	-	-	7	6	4

Tabla 5.12: Valores de los parámetros de paralelismo para cuatro combinaciones de parámetros metaheurísticos en *Saturno*. Número óptimo de hilos del primer nivel de paralelismo para todas las funciones al aplicar el EM a PCEPA (metaheurísticas m1 y m2); el valor del parámetro del segundo nivel se fijó a 1. Número óptimo de hilos (niveles uno y dos de paralelismo) para todas las funciones al aplicar el HEM a PCEPA (hiperheurísticas h1 y h2).

El speed-up obtenido al aplicar el EM con autooptimización mejora en ambos casos los valores obtenidos con un número de hilos seleccionado de una manera no óptima. Sin embargo, para el caso de HEM no existen grandes diferencias en los resultados de speed-up obtenidos, lo que puede deberse al reducido tamaño de las hiperheurísticas estudiadas ejecutadas dentro de un sistema como *Saturno* con un número de cores del orden de los tamaños poblacionales considerados.

5.4. Conclusiones

En este capítulo se ha introducido paralelismo en memoria compartida en el esquema parametrizado de metaheurísticas e hiperheurísticas. Se ha visto su utilidad

Comb.	speed-up max	speed-up $\frac{max}{2}$	speed-up exp-auto
m1	10	8	10
m2	14	10	16
h1	6	5	6
h2	6	6	7

Tabla 5.13: Speed-ups para varias combinaciones de parámetros metaheurísticos al aplicar todas las funciones del EM (metaheurísticas m1 y m2 en las tablas 5.11 y 5.12) y todas las funciones del HEM (hiperheurísticas h1 y h2 en las tablas 5.11 y 5.12) a PCEPA. Valores experimentales obtenidos con el número máximo de hilos disponible (max), con la mitad del número máximo de hilos disponible ($\frac{max}{2}$) y valores obtenidos con autooptimización (exp-auto), en *Saturno*.

para reducir el tiempo de ejecución de los algoritmos, sobre todo en el caso de las hiperheurísticas, puesto que son los métodos más costosos computacionalmente. Se ha visto que muchas veces no basta con paralelizar sin utilizar algún método con fundamento teórico, sino que lo realmente efectivo es utilizar una metodología sistemática y rigurosa que asegure una reducción óptima de los tiempos de ejecución. Así, se ha introducido satisfactoriamente la técnica de autooptimización aplicada a metaheurísticas e hiperheurísticas paralelas, lo que es novedoso en este ámbito. Se ha aplicado la metodología sobre el problema de optimización PCEPA. Se han considerado, para los experimentos, parámetros metaheurísticos de tamaños intermedios, lo que facilita el análisis con la idea de mostrar la utilidad del método.

Después de desarrollar las tres fases de la metodología y de obtener valores de las constantes y parámetros de paralelismo, se puede decir que la técnica proporciona valores satisfactorios del número de hilos a usar en sistemas NUMA. Se han comparado los resultados de speed-up obtenidos con otro número de hilos seleccionados de forma no sistemática y se ha visto que en todos los casos la autooptimización mejora o por lo menos iguala esos valores.

Como conclusión final, cabe resaltar que la autooptimización es una herramienta complementaria muy útil cuando se utiliza junto con hiperheurísticas puesto que permite implementar algoritmos efectivos en bondad y en rapidez en la obtención de resultados.

Capítulo 6

Modelado y autooptimización de metaheurísticas paralelas en memoria distribuida

Este capítulo aporta la introducción de paralelismo de paso de mensajes para acelerar la aplicación de las metaheurísticas. Se va a ampliar la estructura del algoritmo a un esquema parametrizado paralelo en memoria distribuida. Para ello, se introducen nuevos parámetros metaheurísticos y de paralelismo en el esquema, que permitirán controlar la intensidad y la frecuencia del intercambio de información entre procesos, así como el volumen de datos transferidos.

La idea básica de un algoritmo paralelo en memoria distribuida es dividir una tarea en varias partes independientes y resolverlas simultáneamente utilizando múltiples procesadores. La paralelización se usa normalmente como un medio para reducir el tiempo de ejecución. El objetivo es obtener un speed-up cercano al número de procesadores paralelos usados. Después de establecer el entorno paralelo, que puede ser un sistema computacional multiprocesador o varios sistemas interconectados, se debe elegir el esquema de trabajo entre ellos [2, 69]. Se considerará el modelo maestro-esclavo [1, 17, 62, 66, 76].

Una vez fijado el modelo de comunicaciones que seguirán los procesos, se va a considerar un modelo algorítmico para aplicar el esquema metaheurístico paralelo en memoria distribuida cuya implementación se basará en el paradigma maestro-esclavo. En este trabajo se utilizará el modelo metaheurístico de islas [95, 100, 101, 102, 115], que consiste en un conjunto de procesos que resuelven el problema de forma cooperativa mediante el intercambio de información a intervalos establecidos. Las islas pueden ejecutar el mismo algoritmo, algoritmos diferentes o el mismo algoritmo con parámetros diferentes. Este enfoque tiene como objetivo mejorar la efectividad del algoritmo secuencial y, gracias a su implementación mediante el modelo maestro-esclavo, se consigue, además, reducir los tiempos de ejecución.

Se pretende mejorar la eficiencia de los algoritmos, para lo que se utilizarán varios sistemas computacionales dentro de un clúster heterogéneo. Así, se realizará un

estudio del speed-up para distintos tamaños del problema de minimización de costes eléctricos en la explotación de pozos de agua (PCEPA) variando las metaheurísticas aplicadas y el número y tipo de procesadores usados.

Adicionalmente, se pretende establecer un número óptimo de procesos a ejecutar con la inclusión de la metodología de autooptimización orientada a minimizar el tiempo de ejecución. Así, se procederá a desarrollar modelos de tiempo, para posteriormente determinar las constantes características de dichos modelos en los sistemas donde van a ser ejecutados. Finalmente, se procederá a la ejecución de las metaheurísticas con unos parámetros de paralelismo optimizados para el sistema concreto y dependientes de los parámetros metaheurísticos y de las constantes del modelo. La metodología de autooptimización será probada sobre el problema de minimización de costes eléctricos en la explotación de pozos de agua (PCEPA).

6.1. Esquemas parametrizados de metaheurísticas en memoria distribuida

En el capítulo 5 se paralelizaba el esquema parametrizado de metaheurísticas en memoria compartida, introduciendo nuevos parámetros de paralelismo (número de hilos de ejecución) en el esquema. Se vio que el esquema metaheurístico tiene un doble uso: por un lado permite analizar y visualizar las metaheurísticas desde una perspectiva genérica debido a que el mismo esquema representa varias metaheurísticas y, por otro, el esquema general permite reusar las diferentes funciones y variables que lo constituyen.

En este capítulo se va a realizar la paralelización del esquema en memoria distribuida utilzando una implementación basada en la librería de paso de mensajes MPI. Puesto que nuestro objetivo es mostrar la utilidad y eficacia del esquema parametrizado de metaheurísticas en memoria distribuida, se va a considerar sólo paralelismo de paso de mensajes, lo que, además, nos permitirá comparar la eficiencia de este paradigma de paralelismo con los resultados obtenidos en memoria compartida del capítulo 5.

Se aplicará, por tanto, una paralelización de alto nivel con descomposición del dominio en la que se divide el espacio de búsqueda en varios conjuntos diferentes y se explota la búsqueda en cada uno de ellos como un procedimiento paralelo en el que todos los procesadores pueden estar aplicando el esquema metaheurístico a la vez.

Como hemos dicho, se va a emplear un modelo de islas implementado mediante un esquema maestro-esclavo donde, inicialmente, el maestro se encarga de enviar al resto de procesos (esclavos) los datos del problema a resolver así como los parámetros metaheurísticos y de paralelismo usados en el esquema unificado de metaheurísticas. Una vez establecidas las condiciones iniciales en todos los procesos, se procede a resolver el problema aplicando el esquema general en paralelo. Así, se considera un sistema con p procesos con identificadores desde 0 hasta $p-1$, donde el proceso P_0

será considerado el maestro y el resto (desde P_1 hasta P_{p-1}) los esclavos.

Aunque el objetivo fundamental del modelo utilizado es reducir el tiempo de ejecución del algoritmo y, por tanto, mejorar su eficiencia, los resultados de fitness permiten constatar que también se mejora su efectividad.

6.1.1. Esquemas parametrizados en memoria distribuida

La idea básica del modelo algorítmico utilizado es particionar la población en distintas subpoblaciones que se asignarán a los distintos procesos. Esta partición de datos podrá ser homogénea si asignamos la misma cantidad de elementos a cada proceso (adecuado para sistemas computacionales homogéneos), o heterogénea si la división de la población de elementos se hace con algún tipo de ponderación que considere las diferentes prestaciones de distintos procesadores en sistemas heterogéneos.

Bajo este enfoque, el esquema parametrizado del Algoritmo 3 se amplía para tener en cuenta las características del modelo de islas dando lugar a un esquema parametrizado en memoria distribuida con la introducción de una nueva función (de migración) en el esquema y con nuevos parámetros metaheurísticos y de paralelismo ($ParamPar$) indicando la intensidad y la frecuencia del intercambio de información entre procesos, así como el volumen de datos transferidos.

Así, se dispone de un esquema principal (Algoritmo 5) que representa el modelo de islas, donde cada isla o subpoblación, $\mathtt{S_i}$, que es tratada en paralelo por el proceso P_i, se obtiene de manera homogénea dividiento la población total por el número de procesos puestos en marcha, $|\mathtt{S_i}| = \frac{|\mathtt{S}|}{p}$, siendo la unión de todas las subpoblaciones igual a la población total $\mathtt{S} = \mathtt{S_0} \cup ... \cup \mathtt{S_{p-1}}$.

Como se ha considerado una partición homogénea de la población, se inicializará cada proceso con un número de individuos iniciales $\frac{NEIIni}{p}$ que será función del número total de procesos considerado, p. A continuación se aplicará el esquema metaheurístico del Algoritmo 6 secuencialmente a cada subpoblación durante un número de ciclos o iteraciones que vendrá dado por el nuevo parámetro metaheurístico-paralelo $NGMPar$ que establece el número de generaciones que se desarrollan en cada isla de manera aislada hasta la siguiente migración. Como en el caso de los individuos iniciales, el resto de conjuntos de cada subpoblación queda determinado como una función del nuevo parámetro p: $\frac{NEFIni}{p}$, $\frac{NEMSel}{p}$, $\frac{NEPSel}{p}$, $\frac{NMMCom}{p}$, $\frac{NMPCom}{p}$, $\frac{NPPCom}{p}$ y $\frac{NEMInc}{p}$.

La condición de finalización del Algoritmo 5 se producirá cuando se hayan dado una serie de ciclos de evolución-migración, $C = \frac{NMIFin}{NGMPar}$, donde $NMIFin$ fija el número máximo de iteraciones del algoritmo. Una vez alcanzado el número de ciclos máximo establecido, es el proceso maestro el encargado de informar a los esclavos de que deben finalizar sus ejecuciones. La solución quedará determinada por el mejor individuo $\mathtt{s_k}$ (k=0,...,$\frac{NEFIni}{p}$-1) de la subpoblación manejada por el proceso maestro $\mathtt{S_0}$.

Una vez establecido el esquema de trabajo y el tamaño de las distintas subpoblaciones, se debe determinar la magnitud y la frecuencia de las comunicaciones,

Algoritmo 5 Esquema parametrizado de metaheurísticas en memoria distribuida. Modelo de Islas(`S,ParamPar`).

1: EN PARALELO en cada proceso P_i $(i = 0, ..., p-1)$ HACER
2: `Inicializar(`$\mathtt{S_i}$`,ParamIni)`
3: **Mientras** `(no CondiciónDeFin(ParamFin,NGMPar))` **Hacer**
4: Esquema_Metaheurístico_Secuencial($\mathtt{S_i}$`,NGMPar`)
5: `Inmigrar(`$\mathtt{S_i}$`,`$\mathtt{S_0}$`,NEMPar)`
6: **Si** $i = 0$ **Entonces**
7: Integrar Subpoblaciones($\mathtt{S_0}$)
8: **Fin si**
9: `Emigrar(`$\mathtt{S_0}$`,`$\mathtt{S_i}$`,NEMPar)`
10: **Fin Mientras**
11: FIN PARALELO
12: Solución: mejor $\mathtt{s_k} \in \mathtt{S_0}$

Algoritmo 6 Esquema Metaheurístico Secuencial($\mathtt{S_i}$`,NGMPar`).

Mientras `(no CondiciónDeMigración(NGMPar))` **Hacer**
  $\mathtt{SS_i}$`=Seleccionar(`$\mathtt{S_i}$`,ParamSel)`
  $\mathtt{SS1_i}$`=Combinar(`$\mathtt{SS_i}$`,ParamCom)`
  $\mathtt{SS2_i}$`=Mejorar(`$\mathtt{SS1_i}$`,ParamMej)`
  $\mathtt{S_i}$`=Incluir(`$\mathtt{SS2_i}$`,ParamInc)`
Fin Mientras

así como su topología. Para ello se va a describir en detalle la nueva función de migración, así como sus parámetros característicos.

Aunque existen muchas posibilidades, puesto que nuestra idea es analizar las ventajas del uso de un esquema general de metaheurísticas paralelizado en memoria distribuida, se han fijado unos criterios de migración sencillos pero ilustrativos para la metodología empleada. Así, se consideran inmigraciones de los esclavos al maestro (línea 5 del Algoritmo 5) y emigraciones del proceso maestro a los esclavos (línea 9 del Algoritmo 5) del mismo número de individuos (homogeneidad de datos). No se considera intercambio de elementos entre esclavos, permitiéndose solo la combinación de los mejores elementos provenientes de cada subpoblación (y posteriores mejoras y diversificaciones) en el proceso maestro (línea 7 del Algoritmo 5).

Otro aspecto a determinar en nuestro modelo es la tasa de migración que controla el número de individuos a migrar en cada intervalo considerado. El parámetro $NEMPar$ fija dicha tasa que, por simplicidad, será igual tanto para emigraciones como para inmigraciones (aunque podría haber sido diferente). El porcentaje de elementos a migrar de cada subpoblación no debería ser demasiado alto, para potenciar la migración solo de los mejores individuos de cada subpoblación y, así, reducir el tiempo de ejecución manteniendo siempre una cierta cantidad de individuos nativos en cada una de estas subpoblaciones. Por ejemplo, si tenemos una subpoblación con $NEFIni = 50$ inidividuos, un valor de $NEMPar = 10$ establecería una tasa de migración del veinte por ciento.

Por último, se puede analizar la frecuencia en iteraciones del algoritmo con la que ocurren las migraciones, que viene determinada por el parámetro $NGMPar$. Por ejemplo, si se considera un valor de 10 para este parámetro, y se fija el número de iteraciones $NMIFin$ a un máximo de 200, se puedan dar un total de $C = \frac{NMIFin}{NGMPar} = 20$ ciclos de migraciones de elementos. En principio, no parece conveniente considerar frecuencias de migración demasiado altas porque podrían suponer una merma en la capacidad de intercambio de información entre procesos, lo que repercutiría en una pérdida de efectividad. No obstante, en posteriores secciones, se hará un estudio experimental para contrastar esta hipótesis.

Por tanto, se consideran tres nuevos parámetros metaheurístico-paralelos en el esquema de islas, $ParamPar = \{NGMPar, NEMPar, p\}$, que, junto a los anteriores considerados en el capítulo 3, forman un conjunto de veintitrés parámetros metaheurísticos.

6.2. Aplicación del esquema metaheurístico parametrizado en memoria distribuida a PCEPA

Se presentan a continuación los resultados obtenidos al aplicar el esquema metaheurístico en memoria distribuida al problema PCEPA. Se han estructurado los experimentos en dos grupos diferentes dependiendo de si se ejecutaron sobre sistemas homogéneos o heterogéneos. En este último caso se optó por hacer una asignación ho-

mogénea de datos con un mapeo a procesos basado principalmente en las velociades relativas de los componentes computacionales del clúster heterogéneo considerado. Los experimentos se han llevado a cabo ejecutando las distintas combinaciones de metaheurísticas de la tabla 6.1 obtenidas a partir del esquema metaheurístico (EM) paralelo y variando el número de procesos p, el número de elementos a migrar y la frecuencia de migración de los individuos de las diferentes subpoblaciones. Los parámetros metaheurísticos de la tabla 6.1 han sido elegidos porque conforman una serie de metaheurísticas con una amplia variedad de tamaños poblacionales (poblaciones entre 20 y 500 individuos) y con parámetros razonables de paralelismo en memoria distribuida (valores de $NGMPar$ entre 5 y 10, y valores de $NEMPar$ entre 5 y 20). Además, se ha tenido en cuenta un número fijo de 100 iteraciones como condición de finalización en la aplicación de las metaheurísticas, porque permite comparar más fácilmente tiempos de ejecución entre subpoblaciones de diferente tamaño y se entiende que es un valor suficientemete alto como para obtener buenos resultados de fitness.

6.2.1. Aplicación del esquema metaheurístico paralelo en sistemas computacionales homogéneos a PCEPA

Se han considerado dos sistemas homogéneos: por un lado *Saturno* para poder comparar los resultados en memoria distribuida con aquellos obtenidos en memoria compartida, y por otro *Marte* + *Mercurio*, que forman un pequeño clúster con un coste de comunicaciones mayor que el de *Saturno*.

En la tabla 6.2 se muestra una comparativa de los mejores resultados experimentales de speed-up (y su fitness asociado) obtenidos al aplicar dos implementaciones diferentes del esquema metaheurístico en memoria distribuida (MD1 y MD2) con aquellos que se obtuvieron al aplicar el esquema paralelo en memoria compartida (MC) al problema PCEPA de tamaño 50-6 en *Saturno*. Los valores presentados son la media de cinco observaciones. Se han utilizado los parámetros metaheurísticos m1 y m2 de la tabla 6.1. En las dos primeras columnas de la tabla 6.2 aparecen el número de procesos con los que se obtienen los speed-ups mejores para las dos implementaciones de MD.

La figura 6.1 representa la variación del speed-up de las versiones en MD frente a p, y muestra que es mejor usar un número reducido de procesos que usar el número máximo de cores disponible siempre que el speed-up no mejore el alcanzado con este máximo, dejando procesadores libres para otras tareas. Para m1, p toma valores como máximo de 20 porque la población total es $NEFIni = 20$. Las diferencias entre MD1 y MD2 son mínimas, porque ambas usan el modelo de islas con diferencias en la implementación de algunas funciones básicas. Así, MD1 no tiene implementada la función Tabú que usa memoria a corto y largo plazo para evitar movimientos frecuentes y recientes. Su mejora local implica un vecindario de tamaño reducido con una intensidad de búsqueda proporcional al número de hilos de segundo nivel puestos en marcha. La implementación de la diversificación es tal que si diversificamos un

	NEIIni	NEFIni	PEMIni	IMEIni	MCPIni	NGMPar	NEMPar
m1	20	20	50	20	-	10	5
m2	100	50	100	10	-	10	5
m3	50	50	100	15	4	10	5
m4	100	100	100	15	8	10	5
m5	200	200	100	15	12	10	5
m6	20	20	100	20	12	5	5
m7	50	50	75	15	8	5	10
m8	100	100	50	10	4	5	15
m9	500	500	25	5	2	5	20

	NEMSel	NEPSel	NMMCom	NMPCom	NPPCom	PEMMej	IMEMej
m1	10	10	20	5	10	100	20
m2	25	25	100	20	5	50	10
m3	25	25	45	50	45	100	5
m4	50	50	90	100	90	100	5
m5	100	100	10	200	180	100	5
m6	10	10	15	20	15	100	5
m7	25	25	45	50	45	75	5
m8	50	50	90	100	90	50	5
m9	250	250	450	500	450	25	5

	MCMMej	PEDMej	IDEMej	MCDMej	NEMInc	MLPInc
m1	-	50	20	-	10	-
m2	-	100	20	-	25	-
m3	4	10	5	4	25	4
m4	8	10	5	8	50	8
m5	12	10	5	12	100	12
m6	4	10	5	2	10	12
m7	4	10	5	2	25	15
m8	4	10	5	2	50	12
m9	4	10	5	2	250	15

Tabla 6.1: Valores de los parámetros metaheurísticos usados en los experimentos de memoria distribuida.

	Procesos MPI		Speed-up			Fitness		
	MD1	MD2	MC	MD1	MD2	MC	MD1	MD2
m1	12	12	7(−)	17(−)	21(+)	7704.39(+)	7735.39(∼)	7842.02(−)
m2	22	22	9(−)	39(+)	34(−)	8139.18(−)	8050.83(−)	7903.52(+)

Tabla 6.2: Comparativa de los speed-ups mejores (y número de procesos con los cuales se han obtenido) y de los fitness obtenidos con los esquemas de paso de mensajes (MD, implementaciones 1 y 2) y de memoria compartida para dos combinaciones metaheurísticas (m1 y m2 en la tabla 6.1) aplicadas a PCEPA 50-6 en *Saturno*.

individuo y éste resulta no factible, se rechaza y se toma el original. Sin embargo, la implementación MD2 está más optimizada: incluye funciones Tabú, el tamaño del vecindario de elementos en la mejora local es proporcional a la raíz cuadrada del tamaño del individuo a mejorar, y si un individuo se diversifica y resulta no ser factible, se sigue diversificando hasta que lo sea.

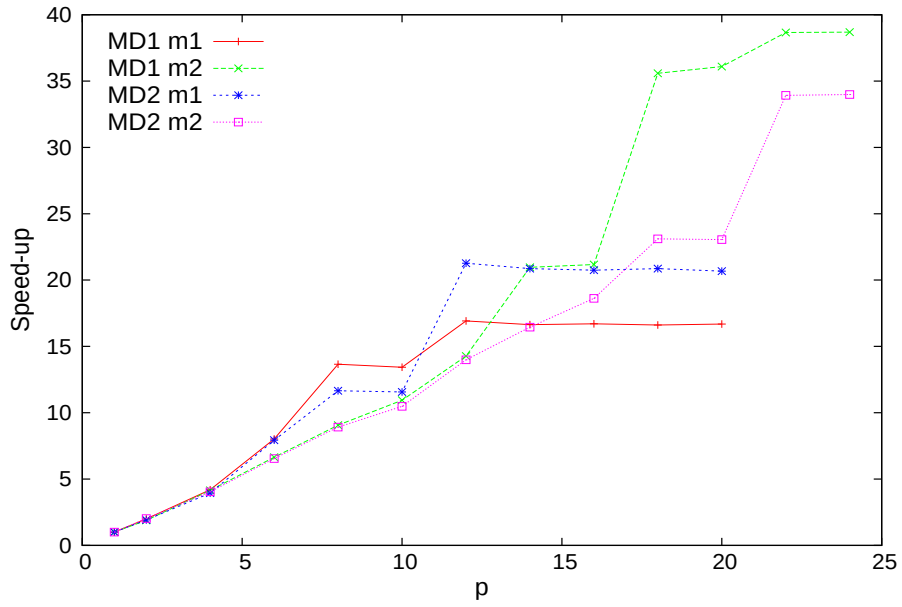


Figura 6.1: Speed-ups obtenidos con los esquemas de paso de mensajes al variar el número de procesos, para dos combinaciones metaheurísticas (m1 y m2 en la tabla 6.1) aplicadas a PCEPA 50-6 en *Saturno*.

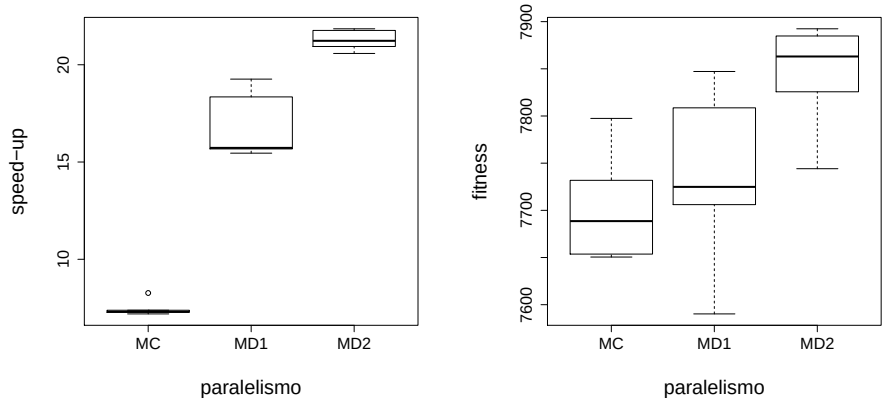
Se han alcanzado valores de speed-up mayores con las implementaciones de MD, posiblemente debido a una paralelización de más alto nivel frente a los resultados obtenidos con MC donde se usó paralelismo dentro de cada función básica. Además, los valores de fitness obtenidos en MC y en MD son similares, lo que implica que las

versiones en memoria distribuida son preferibles a aquellas en memoria compartida en cuanto a tiempo de ejecución, con una calidad de los resultados de fitness similares. Para las metaheurísticas con valores de parámetros más elevados el speed-up es superlineal lo que puede deberse a un mejor uso de la memoria y a la naturaleza cuadrática de algunas funciones dentro del esquema, lo que hace que al dividir los conjuntos en subconjuntos más pequeños el tiempo de ejecución en esas funciones disminuya cuadráticamente con el número de procesos.

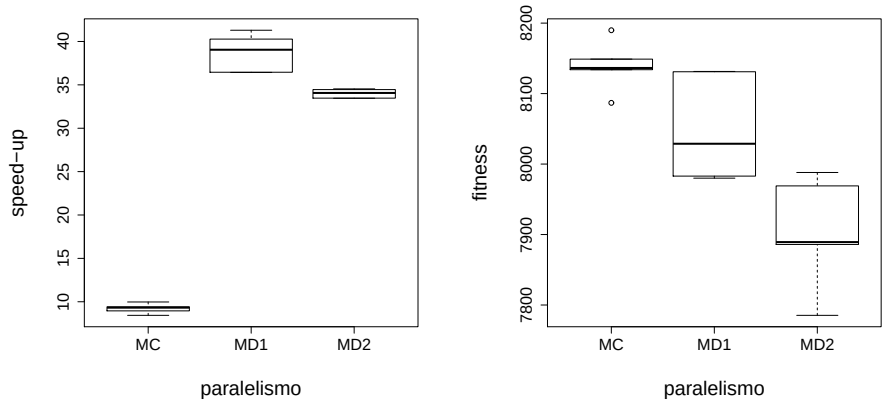
El test de Kruskal-Wallis reveló diferencias estadísticas en las medias de speed-up de la tabla 6.2 para las metaheurísticas m1 y m2, y hubo también diferencias significativas en las medias de fitness para ambas metaheurísticas. Podemos ver estos resultados de manera gráfica en la figura 6.2. Se realizó una comparación más profunda entre los grupos de aplicación de cada metaheurística, con la aplicación del test de Wilcoxon con un nivel de significación $\alpha = 0.05$. El algoritmo con un mejor comportamiento (valores más altos de speed-up y más bajos de fitness) para un conjunto de datos particular se indica con el símbolo $+$. Los algoritmos frente a los cuales es estadísticamente superior se indican con $-$, y $\sim$ refleja que no hubo diferencia en la medias. A la vista de los resultados, a partir de aquí se ha decidido trabajar solo con la implementación MD2 debido a que tiene una implementación más optimizada de las funciones básicas e incluye la Búsqueda Tabú, presentando valores de fitness significativamente mejores en, por lo menos, una aplicación metaheurística (fila 2 de la tabla 6.2).

Como se dijo en la sección 6.1, el modelo de islas en memoria distribuida aporta tres nuevos parámetros metaheurístico-paralelos al esquema: el número de elementos a migrar ($NEMPar$), el número de generaciones hasta la migración ($NGMPar$) y el número de subpoblaciones consideradas (p), que coincide con el número de procesos puestos en marcha en el sistema computacional. Por tanto, se puede evaluar la manera en que estos tres parámetros influyen en el fitness obtenido al aplicar el esquema metaheurístico al problema PCEPA. Primeramente, se va a analizar la influencia de los parámetros $NEMPar$ y $NGMPar$ sobre el fitness para, una vez fijados unos valores razonables para estos parámetros, continuar estudiando la variación del fitness con el número de subpoblaciones (p). Todos los puntos representados a continuación son la media de diez observaciones.

En la figura 6.3 se puede ver la evolución del fitness al aplicar MD2 sobre el problema PCEPA de tamaño 50-6 para dos combinaciones metaheurísticas diferentes (m8 y m9 en la tabla 6.1) en *Saturno*. Se ha elegido el tamaño de problema 50-6 por ser intermedio y necesitar menos tiempo de ejecución en los experimentos, y porque permite comparar los resultados de fitness obtenidos en MD con los que se obtuvieron secuencialmente (figura 3.4). Se ha decidido usar las metaheurísticas m8 y m9 por tener poblaciones grandes (valores de $NEFIni$ de 100 y 500 respectivamente), lo que permite variar los parámetros paralelos en un rango más amplio para obtener conclusiones más generales. Se ha fijado un número de procesos (p) proporcional al tamaño de cada población, con valores de 5 y 10 para m8 y m9 respectivamente. Los experimentos se realizaron manteniendo fijos todos los parámetros de m8 y m9



(a) m1 (b) m1



(c) m2 (d) m2

Figura 6.2: Resumen estadístico del speed-up y del fitness obtenidos al aplicar dos combinaciones metaheurísticas a PCEPA 50-6 en paralelo, en *Saturno*. Se han considerado tres grupos: paso de mensajes con dos implementaciones de las funciones básicas del esquema (MD1 y MD2), y memoria compartida (MC).

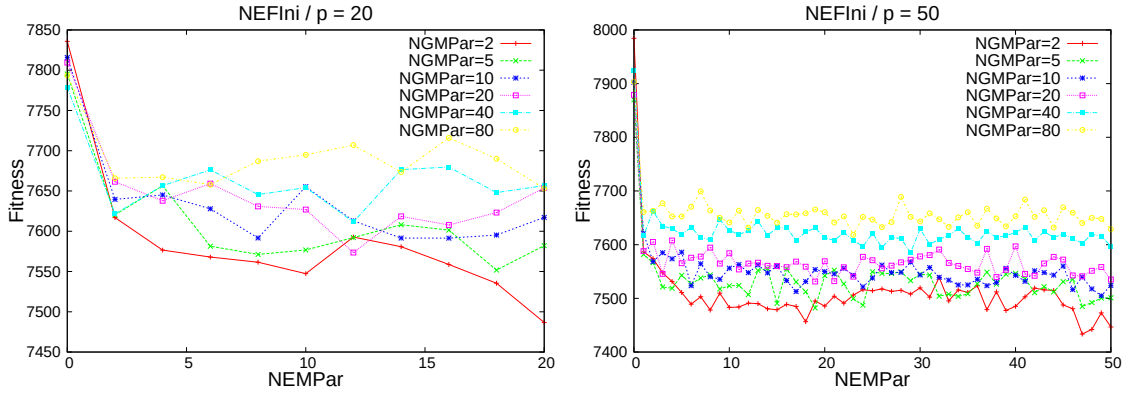
excepto $NEMPar$ y $NGMPar$, que fueron variados dando lugar a las distintas series de resultados. El tamaño de las subpoblaciones para m8 y m9 es $\frac{NEFIni}{p} = 20$ y $\frac{NEFIni}{p} = 50$ respectivamente. El eje de abcisas representa los valores del parámetro $NEMPar$, cuya escala ha sido fijada de manera que se corresponde con un intervalo de porcentaje de migración de elementos (respecto al tamaño de subpoblación considerada) que va de un 0 % ($NEMPar$=0) a un 100 % ($NEMPar$=20 para m8 y $NEMPar$=50 para m9).

Se aprecia como, para las dos metaheurísticas consideradas y para todas las frecuencias de migración, existe una tendencia descendente del fitness hasta un valor de $NEMPar$ a partir del cual el fitness permanece prácticamente constante (con las fluctuaciones debidas a las diferentes configuraciones de parámetros algorítmicos en cada punto). Este valor está en torno al 20 - 25 % de tasa de migración, que se corresponde con valores de $NEMPar$ de aproximadamente 5 y 10 para m8 y m9, respectivamente. En cuanto a la influencia del número de generaciones hasta la migración, se aprecia claramente como las series con valores de $NGMPar$ más bajos son las que obtienen mejores resultados de fitness en todos los casos. Los resultados anteriores podrían explicarse si se considera que es necesario que migre, con cierta frecuencia (valores bajos de $NGMPar$), una mínima cantidad de los mejores individuos de cada subpoblación para que el intercambio de información entre islas sea efectivo. Un aumento de este valor mínimo de $NEMPar$ no tiene influencia significativa sobre el fitness obtenido, manteniéndose este prácticamente constante.

Podemos ver más claramente los resultados anteriores en la tabla 6.3 (y en la figura 6.4), donde se recoge un resumen de los valores medios de fitness obtenidos al variar los dos parámetros metaheurísticos de migración para las dos metaheurísticas estudiadas. Así, en la primera columna de valores de fitness, se aprecia como este disminuye de forma constante al hacerlo el número de generaciones hasta migrar $NGMPar$. Sin embargo, el fitness medio no varía significativamente con el número de elementos que migran (con $NEMPar > 0$), como se ve en la segunda columna de valores de fitness, siendo el primer valor ($NEMPar = 0$) el peor de todos, lo que corrobora la necesidad de migrar elementos para mejorar la bondad de los resultados.

La figura 6.5 muestra la evolución del tiempo de ejecución al aplicar MD2 sobre el problema PCEPA de tamaño 50-6 para las combinaciones metaheurísticas m8 y m9 en *Saturno*. Las series presentadas se corresponden con los tiempos de ejecución necesarios para obtener los resultados de fitness de la figura 6.3.

Se aprecian en detalle los resultados anteriores en la tabla 6.4 (y en la figura 6.6), donde se recoge un resumen de los valores medios de tiempo obtenidos al variar $NEMPar$ y $NGMPar$ para m8 y m9. Así, en la primera columna de valores de tiempo, se aprecia como este disminuye de forma constante al hacerlo la frecuencia de migración. Sin embargo, el tiempo medio no varía de manera significativa con el número de elementos que migran (con $NEMPar > 0$), como se ve en la segunda columna de valores de tiempo. Esto es debido al bajo coste de las comunicaciones de datos en comparación con el coste de inicio de las comunicaciones y el coste

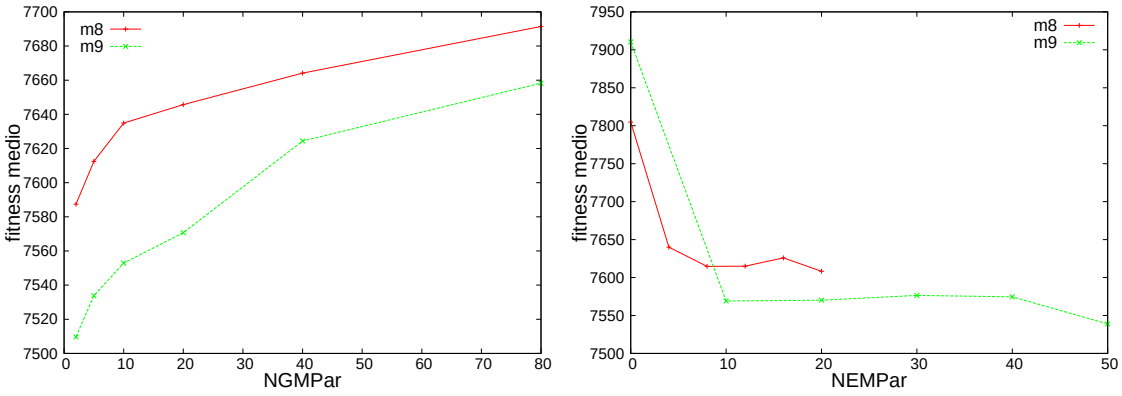


(a) m8 (b) m9

Figura 6.3: Evolución del fitness con el número de elementos a migrar ($NEMPar$) para varias frecuencias de migración ($NGMPar$) en *Saturno* para las combinaciones metaheurísticas m8 y m9 de la tabla 6.1 para el tamaño de problema 50-6 de PCEPA.

	Series de $NEMPar$		Series de $NGMPar$	
	$NGMPar$	fitness medio	$NEMPar$	fitness medio
m8	2	7587.34	0	7804.79
	5	7612.45	4	7639.96
	10	7634.88	8	7614.63
	20	7645.69	12	7614.98
	40	7664.14	16	7625.84
	80	7691.46	20	7608.29
m9	2	7509.68	0	7910.42
	5	7533.81	10	7569.06
	10	7552.89	20	7570.12
	20	7570.67	30	7576.44
	40	7624.41	40	7574.55
	80	7658.28	50	7538.85

Tabla 6.3: Resultados numéricos medios relacionados con la figura 6.3. Fitness medio obtenido al variar $NEMPar$ para varios valores de $NGMPar$ y variando $NGMPar$ para varios valores de $NEMPar$ en *Saturno* para las combinaciones metaheurísticas m8 y m9 de la tabla 6.1 para el tamaño de problema 50-6 de PCEPA.



(a) Series de $NEMPar$ (b) Series de $NGMPar$

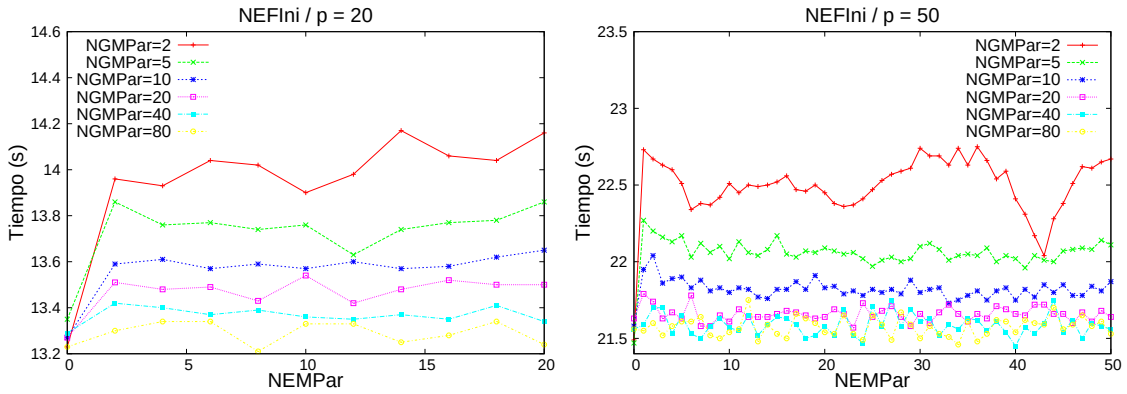
Figura 6.4: Representación gráfica de los valores medios de fitness de la tabla 6.3. (a) Fitness medio obtenido al variar $NEMPar$ para varios valores de $NGMPar$ y (b) variando $NGMPar$ para varios valores de $NEMPar$. En *Saturno* para las combinaciones metaheurísticas m8 y m9 de la tabla 6.1 para el tamaño de problema 50-6 de PCEPA.

computacional.

Según lo visto en las gráficas anteriores, se prodría pensar en fijar el parámetro $NEMPar$ a valores no muy elevados, correspondientes a un 20 - 25 % de migración, y el valor de $NGMPar$ lo más reducido posible (2 según la figura 6.3). No obstante, como se aprecia en la figura 6.5, frecuencias elevadas de migración ($NGMPar$=2) suponen un incremento del tiempo de ejecución que puede llegar a ser de hasta un 4 % superior respecto a frecuencias más bajas ($NGMPar$=80). También se aprecia que existen frecuencias intermedias, en torno a valores de $NGMPar$ de 5 o 10, que permiten potenciar la migración de elementos y que no suponen un excesivo sobrecoste en cuanto a tiempo invertido en estas migraciones.

Por tanto, se pueden considerar valores del parámetro $NEMPar$ relativamente bajos (20 - 25 % de migración) y valores de $NGMPar$ en torno a 5 - 10 como una primera aproximación.

Una vez fijados unos valores razonables para los parámetros $NEMPar$ y $NGMPar$, que podrán variar ligeramente con el tamaño de la población considerada, se va a estudiar la influencia del número de islas o subpoblaciones consideradas. En la figura 6.7 se muestra la evolución del speed-up al variar el número de procesos ejecutados en el clúster homogéneo *Marte* + *Mercurio*, para cuatro tamaños de PCEPA diferentes y cuatro combinaciones metaheurísticas (m6 a m9 en la tabla 6.1). Se han utilizado *Marte* + *Mercurio* porque constituyen un pequeño clúster donde el coste de las comunicaciones es mayor que en *Saturno*. En todos los casos el speed-up se incrementa linealmente con p hasta un número de procesos igual a 12, correspondientes a los cores del clúster *Marte* + *Mercurio*. A partir de $p = 12$ el speed-up fluctúa debido a la carga no balanceada, y un número de procesos mayor

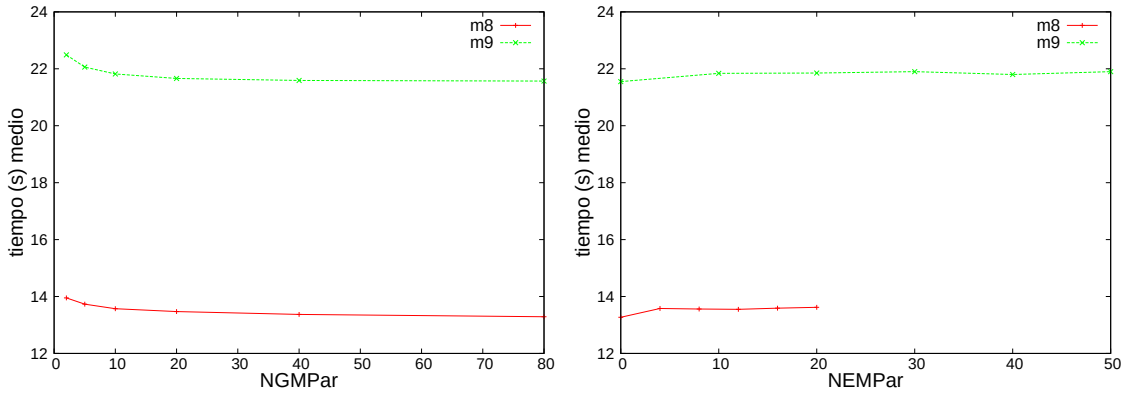


(a) m8 (b) m9

Figura 6.5: Evolución del tiempo de ejecución (en segundos) con el número de elementos a migrar ($NEMPar$) para varias frecuencias de migración ($NGMPar$) en *Saturno* para las combinaciones metaheurísticas m8 y m9 de la tabla 6.1 para el tamaño de problema 50-6 de PCEPA.

	Series de $NEMPar$		Series de $NGMPar$	
	$NGMPar$	tiempo (s) medio	$NEMPar$	tiempo (s) medio
m8	2	13.95	0	13.27
	5	13.73	4	13.58
	10	13.57	8	13.56
	20	13.47	12	13.55
	40	13.37	16	13.59
	80	13.29	20	13.62
m9	2	22.49	0	21.55
	5	22.06	10	21.84
	10	21.82	20	21.85
	20	21.66	30	21.90
	40	21.59	40	21.80
	80	21.57	50	21.90

Tabla 6.4: Resultados numéricos medios relacionados con la figura 6.5. Tiempo (en segundos) medio obtenido al variar $NEMPar$ para varios valores de $NGMPar$ y variando $NGMPar$ para varios valores de $NEMPar$, en *Saturno*, para las combinaciones metaheurísticas m8 y m9 de la tabla 6.1 para el tamaño de problema 50-6 de PCEPA.



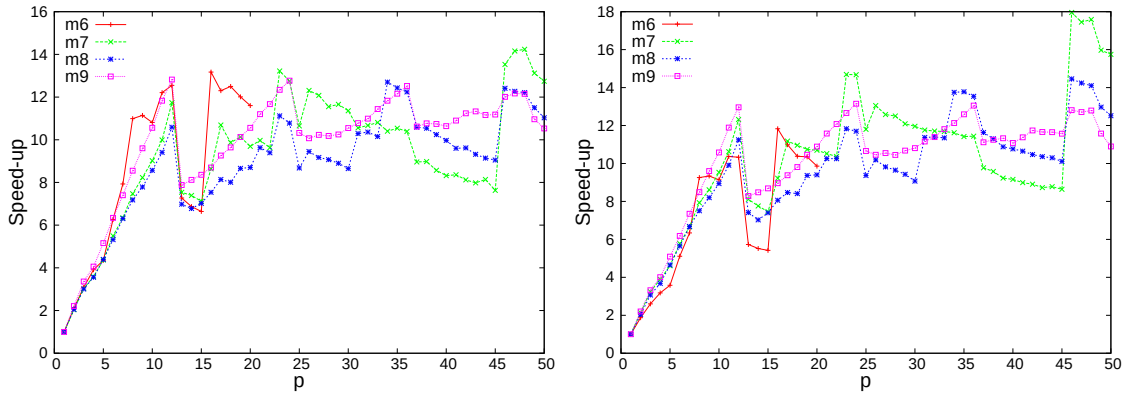
(a) Series de $NEMPar$ (b) Series de $NGMPar$

Figura 6.6: Representación gráfica de los valores medios de tiempo de la tabla 6.4. (a) Tiempo medio en segundos obtenido al variar $NEMPar$ para varios valores de $NGMPar$ y (b) variando $NGMPar$ para varios valores de $NEMPar$. En *Saturno* para las combinaciones metaheurísticas m8 y m9 de la tabla 6.1 para el tamaño de problema 50-6 de PCEPA.

que el número de cores del sistema arroja un speed-up ligeramente superior (cuando la carga está balanceada), lo que puede ser debido a la asignación de procesos a cores desocupados.

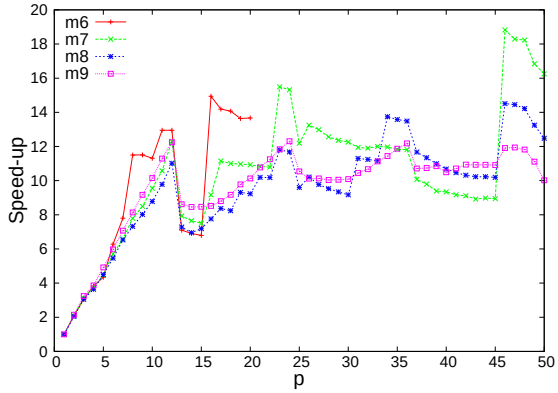
La figura 6.8 muestra la evolución de la función de fitness. Podemos ver que para todos los tamaños de problema se observa un mínimo en la evolución del fitness que, en algunos casos, se sitúa en un número de procesadores cercano a la mitad de los individuos considerados en el conjunto de referencia (m7). En el caso de la combinación metaheurística con parámetros poblacionales más bajos (m6), el mínimo se sitúa en valores de p entre 5 y 10. Para los otros tamaños poblacionales más altos (m8 y m9) la tendencia a alcanzar un mínimo es menos marcada, produciéndose, en estos casos, un estancamiento de los valores de fitness (m9) o incluso un ligero repunte a valores de p mayores de 25 o 30 (caso de m8). Se puede explicar este comportamiento teniendo en cuenta la manera en la que se distribuyen las subpoblaciones entre los distintos procesos. Así, cuando el número de procesos aumenta, el tamaño de las subpoblaciones disminuye proporcionalmente, con lo que se produce una pérdida de diversidad que es todavía más marcada para islas con pocos individuos, como en el caso de m6 y m7, donde el tamaño de las subpoblaciones se hace igual a uno, con lo que se anula la posibilidad de combinación entre individuos.

En resumen, el uso de una cantidad elevada de procesos puede provocar una pequeña pérdida de efectividad de las metaheurísticas (comparar los rangos de fitness de PCEPA 50-6 con los de la figura 3.4), por tanto, desde el punto de vista del fitness y para la implementación considerada, parece adecuado trabajar con valores de p moderados cercanos a la mitad del valor del parámetro poblacional $NEFIni$. Sin embargo, a efectos prácticos, es preferible trabajar con un número de procesos

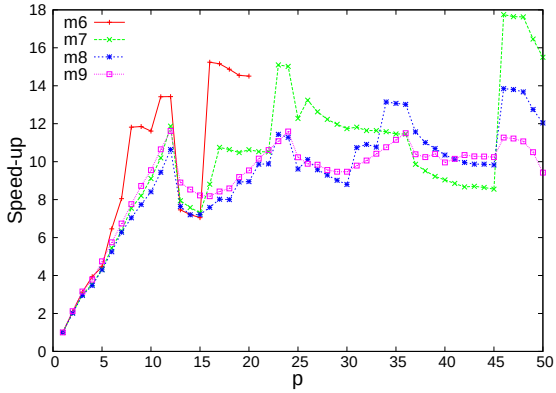


(a) PCEPA 20-6

(b) PCEPA 50-3

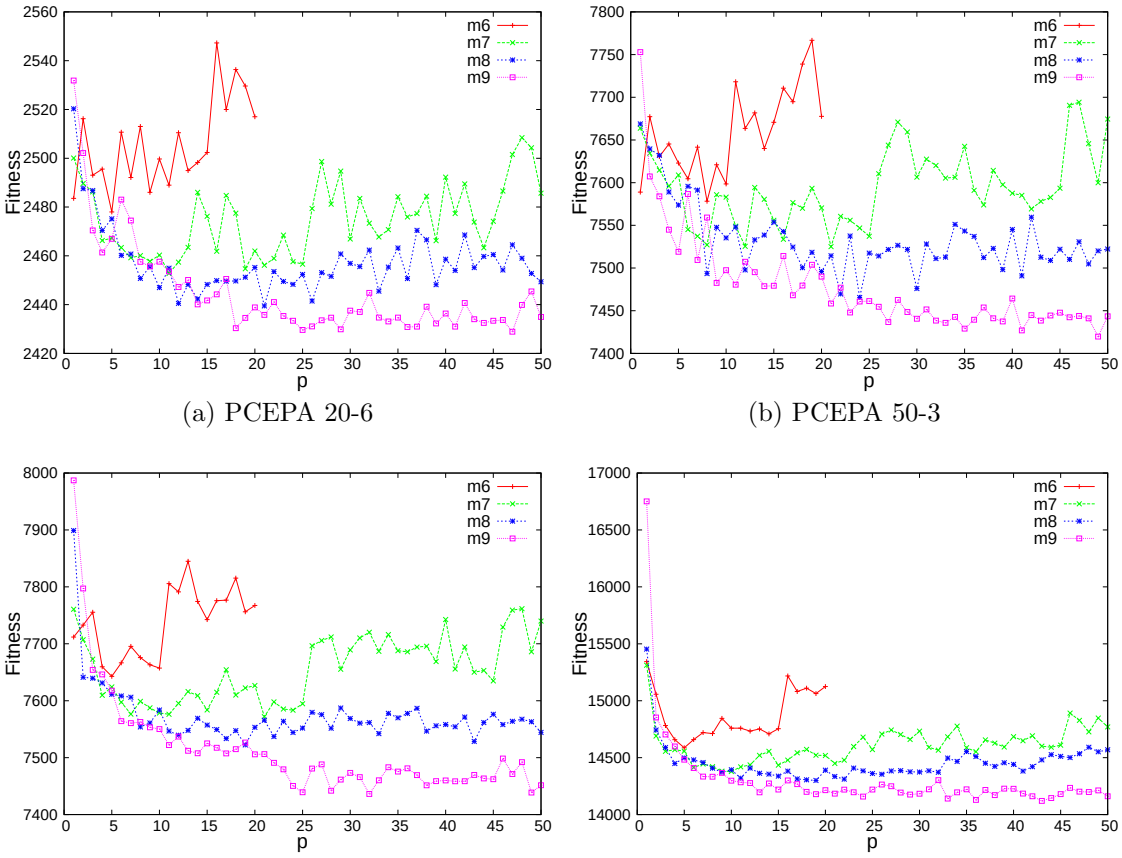


(c) PCEPA 50-6



(d) PCEPA 150-3

Figura 6.7: Evolución del speed-up con el número de procesos (p) en *Marte + Mercurio* para las combinaciones metaheurísticas m6, m7, m8 y m9 de la tabla 6.1 para varios tamaños de PCEPA.



(a) PCEPA 20-6

(b) PCEPA 50-3

(c) PCEPA 50-6

(d) PCEPA 150-3

Figura 6.8: Evolución del fitness con el número de procesos (p) en *Marte* + *Mercurio* para las combinaciones metaheurísticas m6, m7, m8 y m9 de la tabla 6.1 para varios tamaños de PCEPA.

elevado (pero sin sobrecargas excesivas de los procesadores), puesto que el speed-up se incrementa de forma notable asumiéndose solo pérdidas de efectividad residuales en las metaheurísticas.

6.2.2. Aplicación del esquema metaheurístico paralelo en un sistema computacional heterogéneo a PCEPA

Hasta ahora se han presentado resultados de speed-up en sistemas homogéneos. Se plantea ahora la posibilidad de estudiar el comportamiento del esquema metaheurístico en cuanto a tiempo de ejecución en nuestro sistema computacional heterogéneo formado por los nodos *Saturno* + *Marte* + *Mercurio* + *Luna*. En este caso, podemos considerar dos posibilidades, por un lado, realizar una partición heterogénea de los datos que tenga en cuenta las distintas prestaciones de cada procesador para cargarlo con la cantidad de datos proporcional a su velocidad de cómputo

y, por otro lado, asignar una cantidad de procesos a cada nodo proporcionalmente, por ejemplo, a su velocidad relativa pero manejando siempre datos homogéneos en cada proceso. Los experimentos se han llevado a cabo considerando la segunda opción. Se han aplicado las combinaciones metaheurísticas m3, m4 y m5 de la tabla 6.1, variándose el número de procesos y mapeos a los nodos computacionales. En la subsección anterior veíamos que el speed-up aumentaba al aumentar el número de procesos puestos en marcha, con un mínimo en el fitness que suponía una pequeña pérdida de calidad de la solución si se utilizaba un valor de p elevado y próximo a $NEFIni$. Para medir la calidad de los resultados en cuanto a tiempo de ejecución y fitness se introduce aquí un indicador común definido como el inverso del producto del fitness y el tiempo de ejecución. Son deseables valores elevados para este indicador.

Existen muchas posibilidades de asignación homogénea de datos a procesadores heterogéneos. Se han seguido dos criterios básicos de mapeo: uno basado en el número de cores en cada sistema y otro basado en la velocidad relativa de los nodos. La asignación basada en la velocidad relativa es más natural, pero resultados preliminares aconsejan seguir también un criterio basado en el número de cores de cada nodo, el cual es más simple y en algunos casos produce resultados satisfactorios cuando no hay una gran diferencia en las velocidades relativas de los cores en los diferentes nodos. La tabla 6.5 muestra el número de procesos asignado a cada nodo del sistema, para diferentes criterios de asignación:

- *Cores Sin Sobrecarga (CSS)*: El número de procesos asignado a un nodo coincide con el número de cores en el nodo.
- *Sobrecarga Sin Balancear (SSB)*: Un número de procesos (p) proporcional al número de cores se asigna a cada nodo, con el número total de procesos igual a $NEFIni$. Por ejemplo, si consideramos el nodo *Saturno*,

$$p_{sat} = \left\lfloor \frac{NEFIni}{numCores_{total}} \right\rfloor \cdot numCores_{sat} + \left\lceil DIF \cdot \frac{numCores_{sat}}{numCores_{total}} \right\rceil \qquad (6.1)$$

  con $numCores_{total} = 40$ para nuestro clúster heterogéneo, $numCores_{sat} = 24$ para *Saturno*, $DIF = NEFIni - numCores_{total} \cdot \left\lfloor \frac{NEFIni}{numCores_{total}} \right\rfloor$ es el resto del cociente $\frac{NEFIni}{numCores_{total}}$, y $\left\lceil DIF \cdot \frac{numCores_{sat}}{numCores_{total}} \right\rceil$ representa el redondeo al entero más próximo, que es $\left\lfloor DIF \cdot \frac{numCores_{sat}}{numCores_{total}} \right\rfloor$ y $\left\lceil DIF \cdot \frac{numCores_{sat}}{numCores_{total}} \right\rceil$ para los nodos más lentos y más rápidos, respectivamente. Así, para $NEFIni = 100$ (m4 en la tabla 6.1), $p_{sat} = 60$. Del mismo modo, podemos calcular p_{mar} (con $numCores_{mar} = 6$ para *Marte*), p_{mer} (con $numCores_{mer} = 6$ para *Mercurio*) y p_{lun} (con $numCores_{lun} = 4$ para *Luna*), dando lugar a $p_{mar} = p_{mer} = 15$ y $p_{lun} = 10$.

		CSS	SSB	STB	0.1·STB	0.2·STB	0.3·STB
m3	sat	24	30	31	3	6	9
	mar	6	8	7	1	1	2
	mer	6	7	7	1	1	2
	lun	4	5	5	1	1	2
	total	40	50	50	5	10	15
m4	sat	24	60	62	6	12	19
	mar	6	15	14	1	3	4
	mer	6	15	14	1	3	4
	lun	4	10	10	1	2	3
	total	40	100	100	10	20	30
m5	sat	24	120	122	12	24	37
	mar	6	30	28	3	6	8
	mer	6	30	28	3	6	8
	lun	4	20	22	2	4	6
	total	40	200	200	20	40	60

		0.4·STB	0.5·STB	0.6·STB	0.7·STB	0.8·STB	0.9·STB
m3	sat	12	16	19	22	25	28
	mar	3	4	4	5	6	7
	mer	3	3	4	5	6	6
	lun	2	3	3	4	4	5
	total	20	25	30	36	41	46
m4	sat	25	31	37	43	50	56
	mar	6	7	9	10	11	13
	mer	6	7	8	10	11	13
	lun	4	5	6	7	8	9
	total	40	50	60	70	81	91
m5	sat	49	61	73	85	98	110
	mar	11	14	17	20	22	25
	mer	11	14	17	20	22	25
	lun	9	11	13	15	17	19
	total	80	100	120	140	160	180

Tabla 6.5: Número de procesos lanzados para tres combinaciones metaheurísticas (m3, m4 y m5 en la tabla 6.1) aplicadas a PCEPA 50-6 en el sistema heterogéneo *Saturno*(sat) + *Marte*(mar) + *Mercurio*(mer) + *Luna*(lun), con las técnicas de mapeo: *Cores Sin Sobrecarga* (CSS), *Sobrecarga Sin Balancear* (SSB), *Sobrecarga Totalmente Balanceada* (STB), y *ySTB*.

- *Sobrecarga Totalmente Balanceada (STB)*: Se ejecutan un número total de procesos igual a $NEFIni$, pero en este caso la carga computacional se distribuye proporcionalmente a las velocidades relativas de los nodos. Si tenemos $NEFIni$ individuos, el número de procesos asignado a cada nodo, p_x (con $x = sat, mar, mer, lun$), se obtiene con las ecuaciones:

$$\begin{aligned} p_{mar} &= p_{sat} \cdot \frac{v_{mar}}{v_{sat}} \\ p_{mer} &= p_{sat} \cdot \frac{v_{mer}}{v_{sat}} \\ p_{lun} &= p_{sat} \cdot \frac{v_{lun}}{v_{sat}} \\ p_{sat} + p_{mar} + p_{mer} + p_{lun} &= NEFIni \end{aligned} \tag{6.2}$$

  donde $v_x = \frac{numCores_x}{t_{secuen,x}}$ representa la velocidad relativa del algoritmo metaheurístico secuencial en cada nodo, y $numCores_x$ y $t_{secuen,x}$ son el número de cores y el tiempo de ejecución secuencial del algoritmo en el nodo x. Por ejemplo, considerando las velocidades relativas $v_{sat} = 1.000$, $v_{mar} = 0.236$, $v_{mer} = 0.224$ and $v_{lun} = 0.177$, y el valor de $NEFIni = 50$ (m3 en la tabla 6.1), los valores del número de procesos son $p_{sat} = 31$, $p_{mar} = p_{mer} = 7$ y $p_{lun} = 5$, donde cada variable obtenida es redondeada al entero más cercano.

- *ySTB*: Para reducir la sobrecarga que produce un elevado número de procesos, el número de procesos asignado a cada nodo debería ser proporcional a la velocidad de los nodos pero con el número de procesos escalado con un valor y (los valores considerados en los experimentos fueron $y = 0.1, 0.2, \ldots, 0.9$). Así, $p_x(ySTB) = y \cdot p_x(STB)$ (donde $x = sat, mar, mer, lun$). Por ejemplo, para un número de procesos en *Saturno* siguiendo la configuración STB, $p_{sat}(STB) = 31$, y para $y = 0.5$ se obtiene un valor de $p_{sat}(0.5FBO) = 16$ (redondeando a entero).

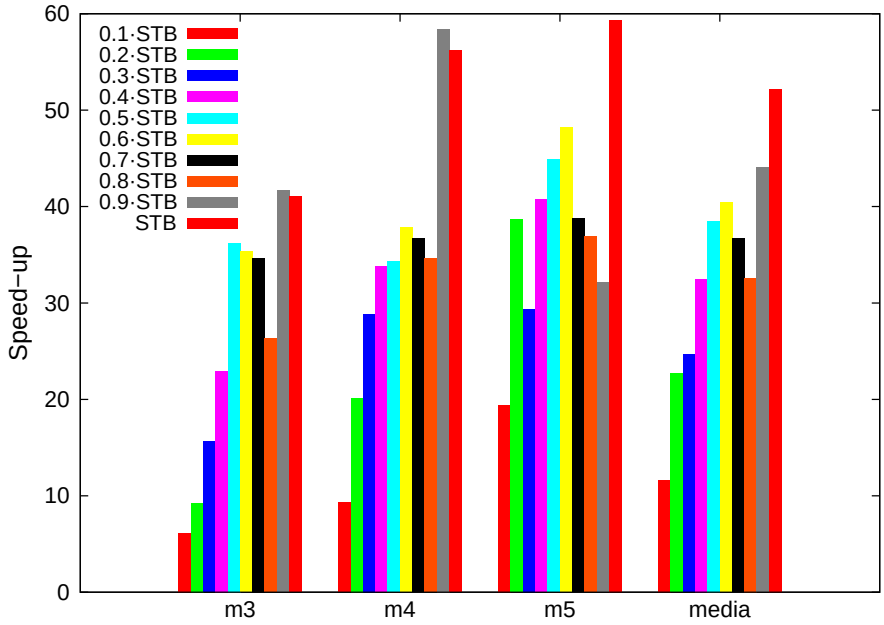
En las dos primeras configuraciones, el número de procesos en cada nodo es proporcional al número de cores. En la tercera configuración la distribución es proporcional a las velocidades relativas de los nodos, que se calculan a partir de los tiempos secuenciales obtenidos a partir de ejecuciones del algoritmo en cada nodo.

Las figuras y tablas siguientes representan los valores medios de diez observaciones de cada variable medida. La figura 6.9 (a) muestra los speed-ups alcanzados al aplicar las metaheurísticas m3, m4 y m5 de la tabla 6.1 al problema PCEPA, para las tres configuraciones básicas de procesos heterogéneos de la tabla 6.5 (columnas 1 a 3), y la figura 6.9 (b) y su tabla asociada 6.6 compara los speed-ups obtenidos al variar el número de procesos de STB (configuraciones $ySTB$) multiplicándolos por varios factores de reducción (de 0.1 hasta 0.9). Las configuraciones SSB y STB presentan, de media, el mismo speed-up (52), que es el más alto obtenido. Podemos ver como, de media, el speed-up aumenta al aumentar el número de cores puestos en marcha y distribuidos según STB con algunas fluctuaciones en $0.7 \cdot STB$ y $0.8 \cdot STB$.

Los valores de fitness se han considerado también. La figura 6.10 y su tabla asociada 6.7 muestran los valores de fitness correspondientes a los speed-ups de la figura 6.9.

	CSS	*SSB*	*STB*
m3	34	41	41
m4	44	54	56
m5	38	61	59
Media	39	52	52

(a)



(b)

Figura 6.9: Speed-up alcanzado al aplicar el esquema metaheurístico a PCEPA 50-6 en el sistema heterogéneo *Saturno* + *Marte* + *Mercurio* + *Luna*: (a) con las tres configuraciones básicas de procesos heterogéneos de la tabla 6.5 y las tres metaheurísticas consideradas en los experimentos heterogéneos y (b) cuando se varía el número de procesos de *STB* al multiplicarlo por varios factores de reducción (0.1 a 0.9).

Se ha construido un Indicador Común usando los resultados de tiempo de ejecución (t) y fitness (f), $IC = \frac{10^6}{f \cdot t}$. Los valores para las configuraciones de procesos y metaheurísticas consideradas se muestran en la figura 6.11 y su tabla de valores numéricos 6.8. El test de Kruskal-Wallis reveló diferencias estadísticas en las medias del IC para las tres configuraciones metaheurísticas aplicadas. A continuación se aplicó el test de Wilcoxon para dos muestras con un nivel de significación $\alpha = 0.05$. El algoritmo con mejores valores de IC (más altos) para un conjunto particular de datos se indica con un simbolo +. Los algoritmos frente a los cuales es

	STB·0.1	STB·0.2	STB·0.3	STB·0.4	STB·0.5
m3	6	9	16	23	36
m4	9	20	29	34	34
m5	19	39	29	41	45
Media	12	23	25	32	38

	STB·0.6	STB·0.7	STB·0.8	STB·0.9	STB
m3	35	35	26	42	41
m4	38	37	35	58	56
m5	48	39	37	32	59
Media	40	37	33	44	52

Tabla 6.6: Speed-up alcanzado cuando se varía el número de procesos de STB al multiplicarlo por varios factores de reducción (0.1 a 0.9), en el sistema heterogéneo *Saturno* + *Marte* + *Mercurio* + *Luna* al aplicar el esquema metaheurístico a PCEPA 50-6.

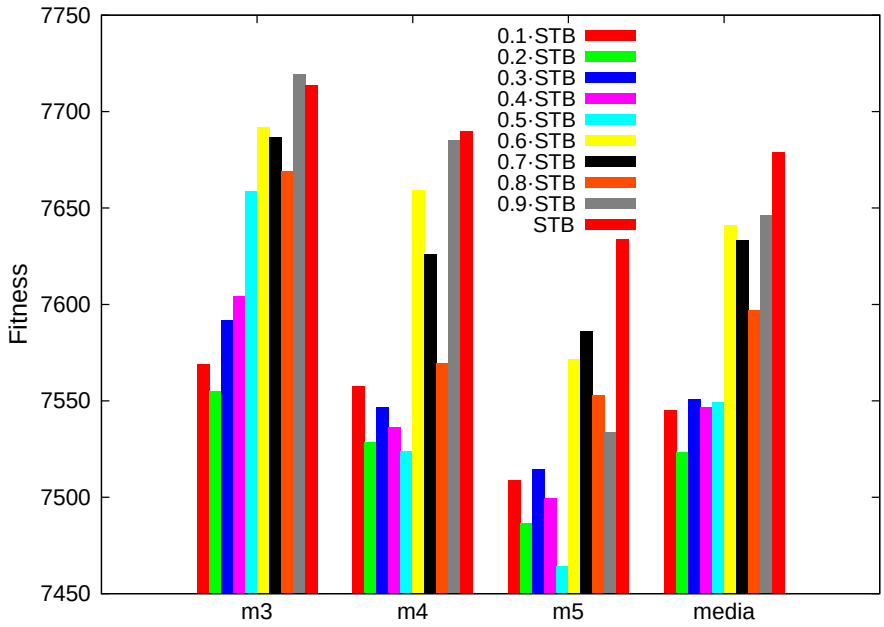
	STB·0.1	STB·0.2	STB·0.3	STB·0.4	STB·0.5
m3	7568.83	7554.83	7591.61	7604.44	7658.83
m4	7557.65	7528.55	7546.69	7536.14	7523.92
m5	7508.95	7486.46	7514.69	7499.48	7464.20
Media	7545.14	7523.28	7551.00	7546.69	7548.98

	STB·0.6	STB·0.7	STB·0.8	STB·0.9	STB
m3	7692.07	7686.74	7669.22	7719.20	7713.48
m4	7658.96	7626.10	7569.27	7685.08	7690.01
m5	7571.51	7586.30	7552.77	7533.79	7633.67
Media	7640.85	7633.05	7597.09	7646.02	7679.06

Tabla 6.7: Fitness alcanzado cuando se varía el número de procesos de STB al multiplicarlo por varios factores de reducción (0.1 a 0.9) en el sistema heterogéneo *Saturno* + *Marte* + *Mercurio* + *Luna* al aplicar el esquema metaheurístico a PCEPA 50-6.

	CSS	*SSB*	*STB*
m3	7674.64	7719.67	7713.48
m4	7529.90	7646.13	7690.01
m5	7483.48	7611.97	7633.67
Media	7562.67	7659.26	7679.06

(a)



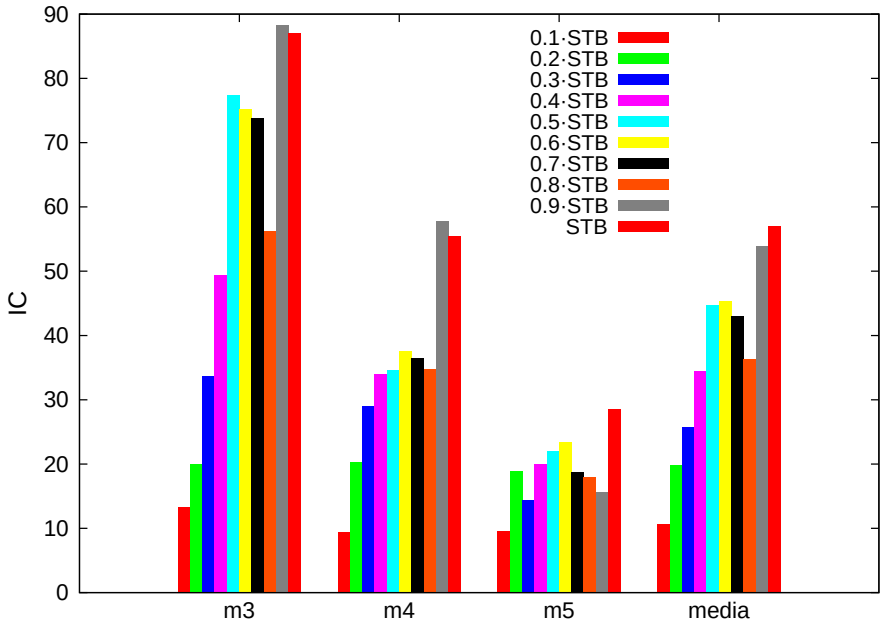
(b)

Figura 6.10: Fitness alcanzado al aplicar el esquema metaheurístico a PCEPA 50-6 en el sistema heterogéneo *Saturno* + *Marte* + *Mercurio* + *Luna*: (a) con las tres configuraciones básicas de procesos heterogéneos de la tabla 6.5 y las tres metaheurísticas consideradas en los experimentos heterogéneos y (b) cuando se varía el número de procesos de *STB* al multiplicarlo por varios factores de reducción (0.1 a 0.9).

estadísticamente superior se indican con $-$, y $\sim$ refleja que no hubo diferencia en la medias. Para m3 y m4 la mejor configuración fue STB, y SSB fue la mejor para m5. Además, para m4, STB fue significativamente mejor que los otros métodos (figura 6.12). Estos resultados nos llevan a elegir STB para hacer los experimentos de la sección (b) de las figuras 6.9, 6.10 y 6.11. En ellas se muestra la evolución del speed-up, fitness e IC al reducir progresivamente el número de procesos lanzados, con el criterio STB y variando el número total de procesos ($ySTB$). Un elevado número de procesos podría producir un gran sobrecarga (overhead), por lo que podría ser interesante una reducción en el número total de procesos cuando la población es muy grande.

	CSS	SSB	STB
m3	73($-$)	86($\sim$)	87($+$)
m4	44($-$)	53($-$)	55($+$)
m5	18($-$)	29($+$)	29($\sim$)
Media	45	56	57

(a)



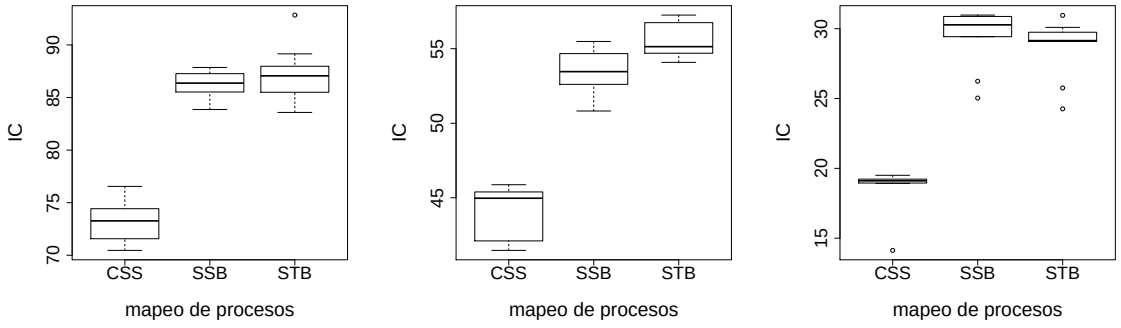
(b)

Figura 6.11: $IC = \frac{10^6}{f \cdot t}$ alcanzado al aplicar el esquema metaheurístico a PCEPA 50-6 en el sistema heterogéneo *Saturno* + *Marte* + *Mercurio* + *Luna*: (a) con las tres configuraciones básicas de procesos heterogéneos de la tabla 6.5 y las tres metaheurísticas consideradas en los experimentos heterogéneos y (b) cuando se varía el número de procesos de STB al multiplicarlo por varios factores de reducción (0.1 a 0.9).

	STB·0.1	STB·0.2	STB·0.3	STB·0.4	STB·0.5
m3	13	20	34	49	77
m4	9	20	29	34	35
m5	9	19	14	20	22
Media	11	20	26	34	45

	STB·0.6	STB·0.7	STB·0.8	STB·0.9	STB
m3	75	74	56	88	87
m4	37	36	35	58	55
m5	23	19	18	16	29
Media	45	43	36	54	57

Tabla 6.8: $IC = \frac{10^6}{f \cdot t}$ alcanzado cuando se varía el número de procesos de STB al multiplicarlo por varios factores de reducción (0.1 a 0.9) en el sistema heterogéneo *Saturno* + *Marte* + *Mercurio* + *Luna* al aplicar el esquema metaheurístico a PCEPA 50-6.



(a) m3 (b) m4 (c) m5

Figura 6.12: Resumen estadístico de $IC = \frac{10^6}{f \cdot t}$, para las tres configuraciones básicas de procesos heterogéneos de la tabla 6.5 y las tres metaheurísticas consideradas (m3, m4 y m5 en la tabla 6.1) en la aplicación del esquema metaheurístico a PCEPA 50-6.

De media, cuando el número de procesos lanzado es reducido (multiplicando STB por un factor de reducción y), el Indicador Común también decrece, por lo que para valores moderados del parámetro poblacional $NEFIni$ (entre 50 y 200) es recomendable poner en marcha un número de procesos cercano al valor de este parámetro, usando un mapeo basado en las velocidades relativas de los nodos (STB).

6.3. Autooptimización de metaheurísticas aplicadas a PCEPA en sistemas homogéneos

Al igual que veíamos en el estudio de memoria compartida, para reducir el tiempo de ejecución es necesario seleccionar los valores de los parámetros de paralelismo adecuadamente. Dichos parámetros son, para el paradigma de paso de mensajes, el número de procesos p, el número de elementos a migrar $NEMPar$ y la frecuencia de la migración $NGMPar$. Se utiliza el problema de minimización de consumo eléctrico en la explotación de pozos de agua (PCEPA) de tamaño 50-6 para mostrar la metodología de modelado y autooptimización en varios sistemas computacionales. Las metaheurísticas utilizadas para mostrar el funcionamiento de la metodología son las recogidas en la tabla 6.1, donde se ha elegido la metaheurística m7 para el cálculo de las constantes del modelo en el proceso de instalación por ser de tamaño medio.

Se llevará a cabo un primer estudio de la metodología en el sistema homogéneo *Saturno*, que es de memoria compartida y en el que usaremos el esquema de paso de mensajes, considerándolo como un sistema distribuido homogéneo. Como sabemos, el primer paso consiste en obtener un modelo teórico del tiempo de ejecución total, que se divide esencialmente en una parte de computación t_{cmp}, otra de comunicaciones t_{cmc} entre procesos y otra de ordenación de elementos en el maestro t_{ord} (ver desarrollo completo en la sección 2 del apéndice A):

$$t_{total} = t_{cmp} + t_{cmc} + t_{ord} \tag{6.3}$$

Un análisis previo nos permitirá despreciar algunos términos de esta ecuación (como t_{ord} y el componente de difusión de las comunicaciones) como queda reflejado en la evolución de todos los tiempos en la figura 6.13 en función del número de procesos p. En ella se ha representado la media de los tiempos para varias series de ejecuciones con valores de $NEMPar$ correspondientes a un 100 %, 75 %, 50 % y 25 % de migración.

El tiempo de computación puede modelarse según la siguiente ecuación, donde p es el número de procesos puestos en marcha, y $k_{i(j)}$ y $Param_{i(j)}$ son las constantes y los conjuntos de parámetros metaheurísticos de cada función del esquema (i para la generación inicial de elementos y su mejora, y j para el resto de funciones que se ejecutan $NGMPar$ veces hasta cada migración):

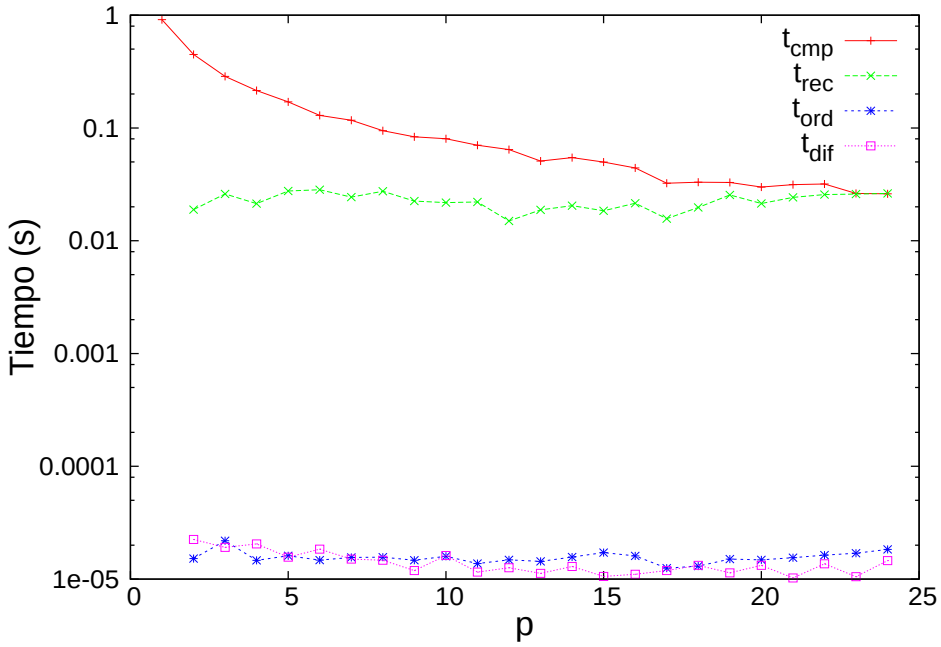


Figura 6.13: Tiempos experimentales de computación t_{cmp}, comunicaciones t_{rec} y t_{dif}, y de ordenación t_{ord} en función del número de procesos p al aplicar la metaheurística m7 a PCEPA en *Saturno*. Valores medios de cuatro series de ejecuciones con valores de $NEMPar$ correspondientes para cada valor de p a un 100 %, 75 %, 50 % y 25 % de migración.

$$t_{cmp} = \sum_{i=1}^{2} k_i \cdot \frac{Param_i}{p} + \left(\sum_{j=1}^{6} k_j \cdot \frac{Param_j}{p} \right) \cdot NGMPar \tag{6.4}$$

Los valores concretos de las constantes se obtienen en la instalación haciendo experimentos con valores concretos de los parámetros metaheurísticos. En la tabla 6.9 se muestran los valores obtenidos en *Saturno* cuando se utiliza en la instalación la metaheurística m7 (tabla 6.1) y con un valor de $NGMPar$ fijado a 1 para considerar el caso de mayor contribución de las comunicaciones (un proceso de comunicación por cada iteración), lo que permite minimizar el fitness según se muestra en la figura 6.3.

$func$	k_i	$Param_i$
Gen-Ini	$4.70 \cdot 10^{-3}$	50
Mej-Ini	$3.36 \cdot 10^{-4}$	$\frac{50 \cdot 75 \cdot 15}{100}$

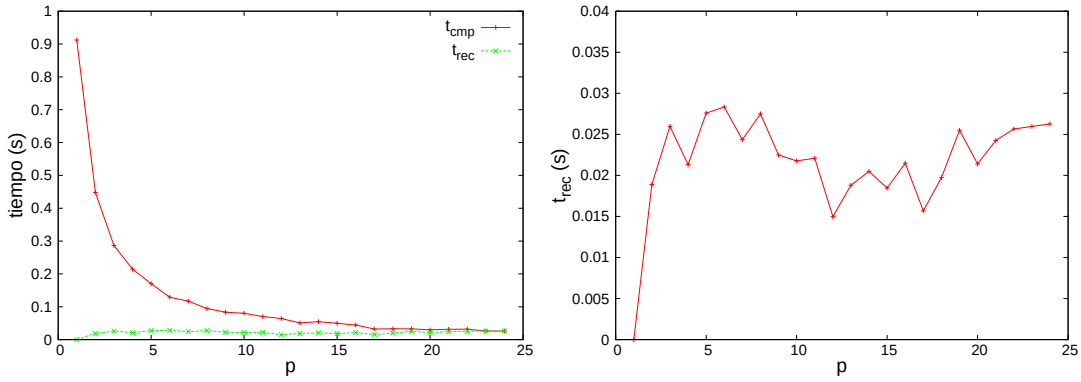
$func$	k_j	$Param_j$
Com	$3.98 \cdot 10^{-5}$	$2 \cdot (45 + 50 + 45)$
Mej-Ref	$3.36 \cdot 10^{-4}$	$\frac{50 \cdot 75 \cdot 5}{100}$
Mej-Com	$6.72 \cdot 10^{-4}$	$\frac{(45+50+45) \cdot 75 \cdot 5}{100}$
Div-Ref	$3.53 \cdot 10^{-4}$	$\frac{50 \cdot 10 \cdot 5}{100}$
Div-Com	$7.06 \cdot 10^{-4}$	$\frac{(45+50+45) \cdot 10 \cdot 5}{100}$
Inc	$1.50 \cdot 10^{-5}$	$50 + 2 \cdot (45 + 50 + 45) - 25$

Tabla 6.9: Valores específicos de las constantes secuenciales del sistema *Saturno* y de los parámetros metaheurísticos considerados para el modelo en la ecuación 6.4. Funciones: Gen-Ini, generación de elementos en la inicialización; Mej-Ini, Mej-Ref y Mej-Com, mejora de elementos en la inicialización, mejora del conjunto de referencia y del de combinaciones, respectivamente; Com, combinación; Div-Ref y Div-Com, diversificación de los conjuntos de referencia y combinaciones; Inc, función de inclusión de elementos.

Por otra parte, el tiempo de comunicaciones es debido fundamentalmente al proceso de recogida de elementos por parte del maestro t_{rec}, siendo despreciable la posterior difusión de los mismos a los esclavos, como puede apreciarse en la figura 6.13. Así, simplificamos el coste de las comunicaciones quedando:

$$t_{cmc} = t_{rec} \tag{6.5}$$

con



(a) $t_{cmp} + t_{rec}$ (b) t_{rec}

Figura 6.14: Evolución en *Saturno* frente a p de (a) los tiempos de computación t_{cmp} y de recogida de elementos en el maestro t_{rec}, y (b) t_{rec} por separado para apreciar mejor la forma de la curva en la zona comprendida entre p=6 y p=24.

$$t_{rec} = f(p, NEMPar) \tag{6.6}$$

y donde se ha determinado experimentalmente esta función f junto a sus constantes características en el mismo proceso de instalación utilizando la metaheurística m7 y para un rango de p comprendido entre 6 y 24 procesos. Se ha elegido este rango para obtener una mayor precisión de la función en la región donde las comunicaciones pueden afectar significativamente a los tiempos de computación, como puede apreciarse en las figuras 6.14 (a) y (b). Además, como se aprecia en la figura 6.5, se ha constatado experimentalmente que el tiempo de comunicación para una metaheurística dada, no varía significativamente con el número de elementos a migrar $NEMPar$, con lo que finalmente tenemos:

$$t_{rec} = -4 \cdot 10^{-6} \cdot p^3 + 0.0003 \cdot p^2 - 0.006 \cdot p + 0.0549 \tag{6.7}$$

donde se ha ajustado t_{rec} a una función polinómica de grado tres dependiente solo de p, considerando el ajuste experimental de datos variando p para varios tamaños de $NEMPar$. Se ha probado a hacer el ajuste con varias funciones, determinándose los correspondientes valores de sus coeficientes de regresión. Se ha decidido utilizar una función polinómica de grado tres por ser la que ofrece un ajuste razonablemente bueno y con una complejidad relativamente baja.

Como hemos dicho, en la figura 6.13 se ve que el tiempo de ordenación t_{ord} es despreciable en todo el rango de p considerado.

Finalmente, derivando el tiempo total de la ecuación 6.3 respecto al número de procesos, obtenemos el número de procesos óptimo p_{opt} en función de las constantes del sistema y de los parámetros metaheurísticos considerados, en forma de ecuación de cuarto grado cuya resolución se detalla en el apéndice A:

$$\frac{-K_{cmp}}{p_{opt}^2} + 3 \cdot \ -4 \cdot 10^{-6} \cdot p_{opt}^2 + 2 \cdot 0.0003 \cdot p_{opt} - 0.006 = 0 \tag{6.8}$$

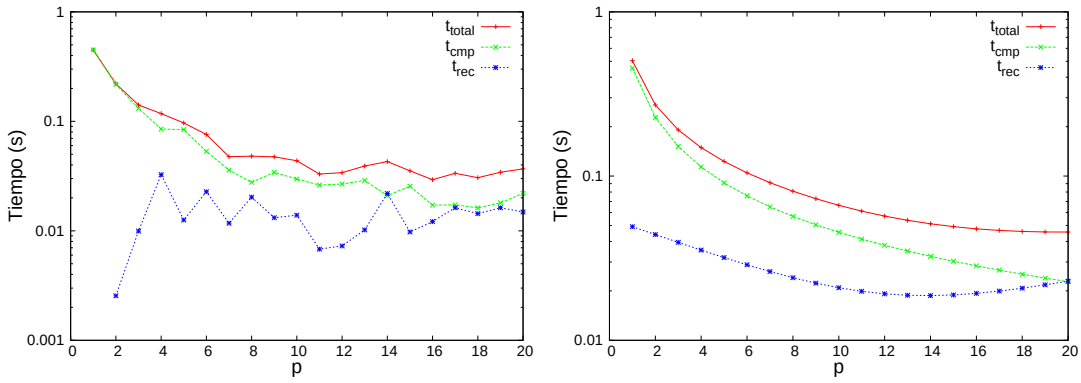
donde K_{cmp} engloba la suma de todas las constantes y parámetros metaheurísticos del término de computación especificados en el apéndice A.

En la tabla 6.10 se presentan los resultados de speed-up y número de procesos óptimo correspondiente al ejecutar las metaheurísticas m6 y m8 de la tabla 6.1. Se puede ver que el número de procesos predicho por el modelo se acerca bastante al experimental en ambos casos, siendo los speed-ups alcanzados próximos a los mejores obtenidos experimentalmente. En el caso de m8, $p_{opt.}$ coincide con el número total de cores de *Saturno* (24), con lo que los speed-ups experimental (exp) y el experimental seleccionado con $p_{opt.}$ (exp-mod) coinciden. Esto ocurre normalmente para metaheurísticas con poblaciones relativamente grandes ($NEFIni$=100 para m8), donde el tiempo de comunicaciones entre procesos no llega a tener suficiente peso comparado con el de computación hasta un número de procesos superior al de cores disponibles en el sistema. Sin embargo, para metaheurísticas más reducidas como m6 ($NEFIni$=20) el óptimo puede no coincidir con el número máximo de cores disponible, pues el tiempo de computación es más reducido y queda compensado por el incremento constante del tiempo de comunicación. Podemos ver este efecto en la figura 6.15, que recoge la evolución experimental y teórica de los tiempos de ejecución para las metaheurísticas m6 y m8. Se aprecia que los tiempos modelados siguen la tendencia de los experimentales sin las fluctuaciones propias de estos.

	Procesos MPI		Speed-up		
metaheurística	exp	mod	exp	mod	exp-auto
m6	16	20	15	11	12
m8	24	24	17	16	17

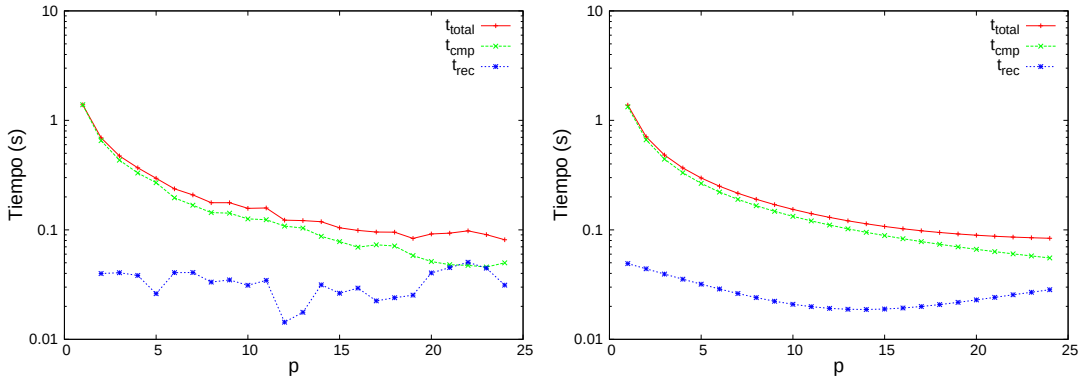
Tabla 6.10: Comparativa del speed-up experimental más alto (exp), el speed-up modelado (mod) y el obtenido en los experimentos al usar la metodología de auto-optimización (exp-auto), para las metaheurísticas m6 y m8 de la tabla 6.1, al aplicar el esquema metaheurístico a PCEPA 50-6 en *Saturno*.

Para comprobar que la metodología de autooptimización para paso de mensajes funciona, se ha aplicado al sistema homogéneo formado por *Marte* + *Mercurio*. Puesto que tenemos dos nodos diferentes con las mismas características y 6 cores cada uno, tendremos comunicaciones intranodo hasta un número de procesos $p = 6$, y comunicaciones intranodo + internodo a partir de este número de procesos. Los pasos seguidos son similares a los descritos para *Saturno*, por lo que nos centraremos más en los resultados obtenidos obviando algunos pasos intermedios. Primeramente se han obtenido las constantes secuenciales para el modelo del tiempo de computación, que pueden verse en la tabla 6.11. Comparando los valores de los parámetros en *Marte*



(a) m6 — tiempo experimental

(b) m6 — tiempo modelado



(c) m8 — tiempo experimental

(d) m8 — tiempo modelado

Figura 6.15: Evolución en *Saturno* de los tiempos experimentales y modelados de computación t_{cmp}, de recogida de elementos en el maestro t_{rec} y total $t_{total} = t_{cmp} + t_{rec}$, al aplicar la metodología de autooptimización a (a) y (b) la metaheurística m6 y (c) y (d) la metaheurística m8 de la tabla 6.1.

+ *Mercurio* (tabla 6.11) con los obtenidos en *Saturno* (tabla 6.9) observamos que en general las prestaciones en *Saturno* son ligeramente mejores que en *Marte+Mercurio*, pero hay ligeras variaciones, lo que podría producir decisiones distintas en cuanto al número de procesos a usar, y justifica la utilización de un método de optimización automático.

$func$	k_i	$Param_i$
Gen-Ini	$5.30 \cdot 10^{-3}$	50
Mej-Ini	$3.83 \cdot 10^{-4}$	$\frac{50 \cdot 75 \cdot 15}{100}$

$func$	k_j	$Param_j$
Com	$4.54 \cdot 10^{-5}$	$2 \cdot (45 + 50 + 45)$
Mej-Ref	$3.83 \cdot 10^{-4}$	$\frac{50 \cdot 75 \cdot 5}{100}$
Mej-Com	$7.66 \cdot 10^{-4}$	$\frac{(45+50+45) \cdot 75 \cdot 5}{100}$
Div-Ref	$3.92 \cdot 10^{-4}$	$\frac{50 \cdot 10 \cdot 5}{100}$
Div-Com	$7.84 \cdot 10^{-4}$	$\frac{(45+50+45) \cdot 10 \cdot 5}{100}$
Inc	$9.38 \cdot 10^{-6}$	$50 + 2 \cdot (45 + 50 + 45) - 25$

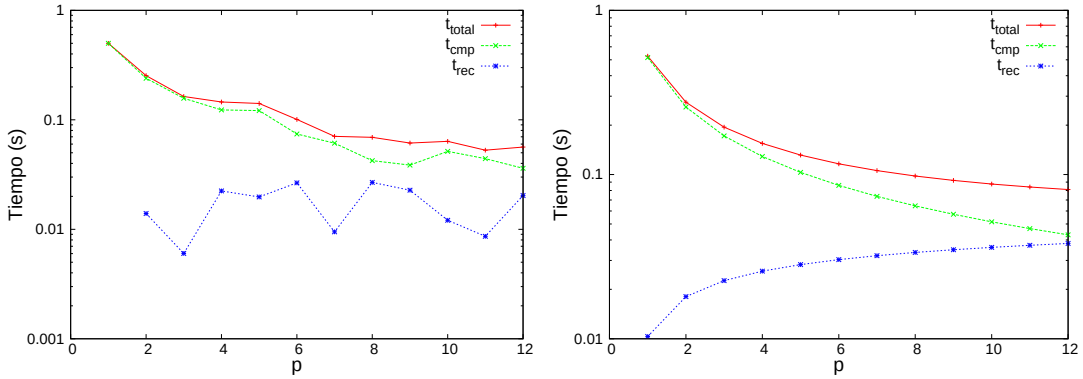
Tabla 6.11: Valores específicos de las constantes secuenciales del sistema *Marte* + *Mercurio* y de los parámetros metaheurísticos considerados para el modelo en la ecuación 6.4. Funciones: Gen-Ini, generación de elementos en la inicialización; Mej-Ini, Mej-Ref y Mej-Com, mejora de elementos en la inicialización, mejora del conjunto de referencia y del de combinaciones, respectivamente; Com, combinación; Div-Ref y Div-Com, diversificación de los conjuntos de referencia y combinaciones; Inc, función de inclusión de elementos.

Al igual que en *Saturno*, el número de procesos predicho por el modelo se acerca bastante al experimental, siendo los speed-ups alcanzados igualmente próximos a los mejores obtenidos experimentalmente. Podemos verlo en la tabla 6.12. Puesto que ahora tenemos un total de 12 cores de la misma naturaleza (6 en *Marte* y 6 en *Mercurio*), cabe esperar que, para los tamaños de metaheurística considerados ($NEFIni$=20 para m6 y $NEFIni$=100 para m8), los tiempos de computación sean todavía significativamente superiores a los de comunicaciones, con lo que el número de procesos óptimo está próximo al número máximo de cores disponible. Dicho de otro modo, todavía tendríamos margen para reducir el tiempo de computación si dispusiéramos de más procesadores en el sistema sin un aumento considerable del coste de las comunicaciones. Podemos ver la evolución de los tiempos experimentales y teóricos dados por el modelo en las figuras 6.16 (a) y (b) para m6 y (c) y (d) para m8.

Finalmente, dado que disponemos de un sistema heterogéneo (*Júpiter* + *Luna* +

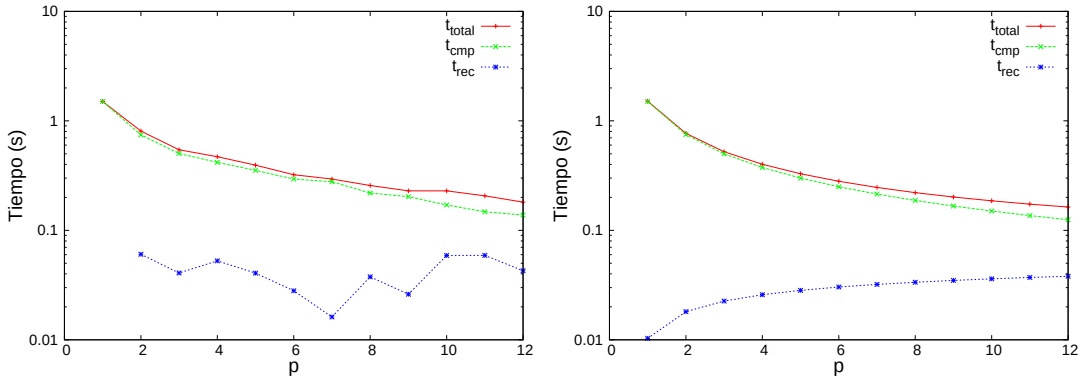
metaheurística	Procesos MPI exp	mod	Speed-up exp	mod	exp-auto
m6	11	12	10	7	9
m8	12	12	8	9	8

Tabla 6.12: Comparativa del speed-up experimental más alto (exp), el speed-up modelado (mod) y el obtenido en los experimentos al usar la metodología de autooptimización (exp-auto), para las metaheurísticas m6 y m8 de la tabla 6.1, al aplicar el esquema metaheurístico a PCEPA 50-6 en el clúster homogéneo *Marte* + *Mercurio*.



(a) m6 — tiempo experimental

(b) m6 — tiempo modelado



(c) m8 — tiempo experimental

(d) m8 — tiempo modelado

Figura 6.16: Evolución en el clúster homogéneo *Marte* + *Mercurio* de los tiempos experimentales y modelados de computación t_{cmp}, de recogida de elementos en el maestro t_{rec} y total $t_{total} = t_{cmp} + t_{rec}$, al aplicar la metodología de autooptimización a (a) y (b) la metaheurística m6 y (c) y (d) la metaheurística m8 de la tabla 6.1.

Saturno + *Marte* + *Mercurio*), podríamos considerar el desarrollo de alguna técnica de autooptimización adaptada a sistemas heterogéneos. Hasta ahora sólo hemos analizado la aplicación a un clúster heterogéno de la técnica para sistemas homogéneos, considerando que el tiempo de ejecución vendrá determinado por las islas asignadas a los cores más lentos. En el sistema donde se han hecho los experimentos se dispone de 52 cores, pero solo se han considerado 50 islas al obtener el modelo a partir de la metaheurística m7 con $NEFIni$=50. Las constantes secuenciales para modelar los tiempos de cómputo son las recogidas en la tabla 6.11, puesto que se ha comprobado que el sistema formado por *Marte* + *Mercurio* es el que posee los procesadores con menor velocidad, lo que supone el cuello de botella del clúster en cuanto a computación.

Lo primero que se ha hecho ha sido ver el orden de magnitud de todos los tiempos considerados en el modelo para hacer las primeras simplificaciones. Como ocurría en los sistemas anteriores, los tiempos de difusión y ordenación son despreciables en todo el rango de p considerado. Se aprecia esto en la figura 6.17. Además, el criterio de asignación de procesos a nodos ha consistido en asignar primero a los nodos más rápidos según el siguiente orden: *Júpiter* + *Luna* + *Saturno* + *Marte* + *Mercurio*.

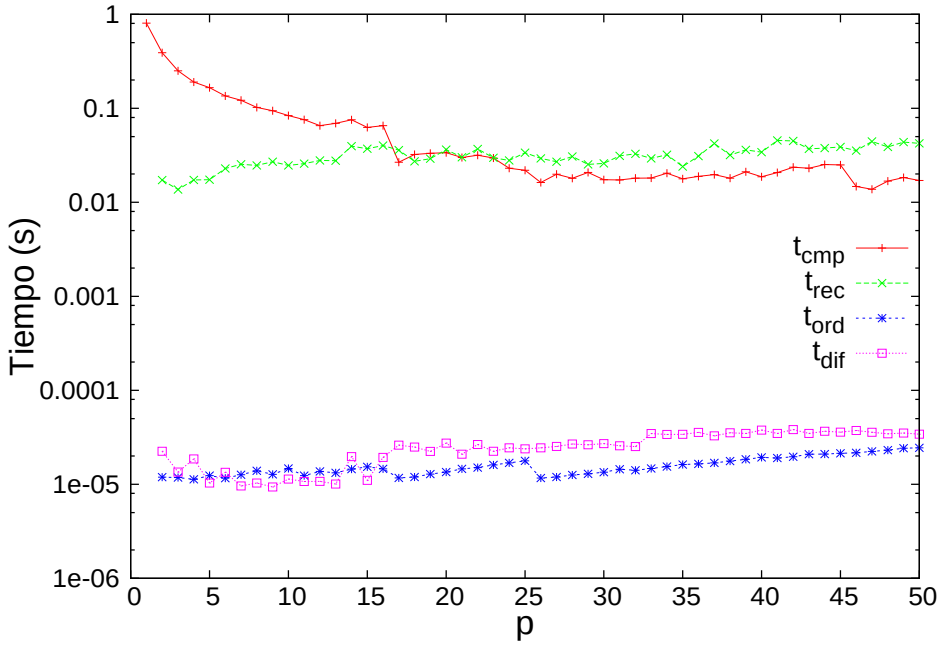
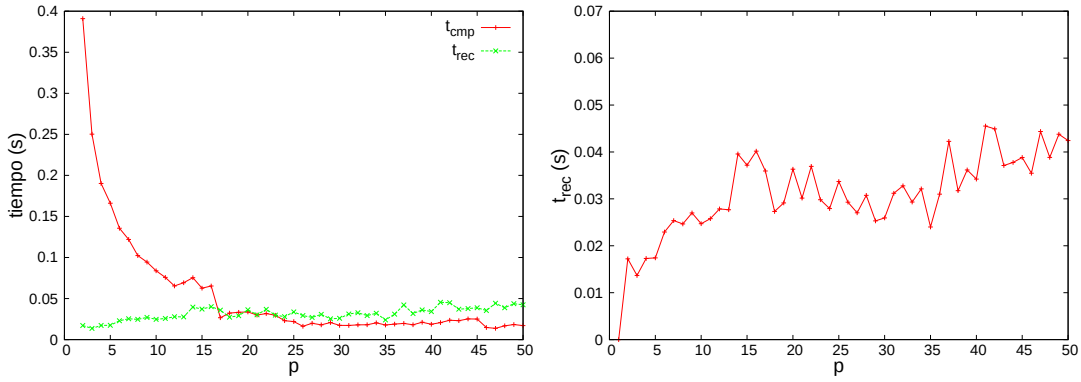


Figura 6.17: Tiempos experimentales de computación t_{cmp}, comunicaciones t_{rec} y t_{dif}, y de ordenación t_{ord} en función del número de procesos p al aplicar la metaheurística m7 a PCEPA en el clúster heterogéneo *Júpiter* + *Luna* + *Saturno* + *Marte* + *Mercurio*. Valores medios de cuatro series de ejecuciones con valores de $NEMPar$ correspondientes para cada valor de p a un 100 %, 75 %, 50 % y 25 % de migración.

Así, los únicos tiempos a considerar en el modelo son, como en los sistemas



(a) $t_{cmp} + t_{rec}$ (b) t_{rec}

Figura 6.18: Evolución en el clúster heterogéneo *Júpiter* + *Luna* + *Saturno* + *Marte* + *Mercurio* frente a p de (a) los tiempos de computación t_{cmp} y de recogida de elementos en el maestro t_{rec}, y (b) t_{rec} por separado para apreciar mejor la forma de la curva en la zona comprendida entre p=6 y p=24.

homogéneos, los de computación y recogida de datos en el maestro. Podemos ver en la figura 6.18 (a) que el tiempo de comunicaciones supera al de computación a partir de $p = 16$. Obtendremos el modelo del tiempo de comunicaciones (t_{rec}) ajustando la curva de la figura 6.18 (b) a una función, tal como hicimos en casos anteriores.

Podemos ver los resultados de speed-up y número óptimo de procesos para cada metaheurística en la tabla 6.13. Se aprecia que el número de procesos óptimo obtenido con el modelo está próximo al experimental. De igual manera los speed-ups teóricos y experimentales están bastante próximos, sobre todo en el caso de las metaheurísticas m6 y m9. El mapeo de procesos y el modelo considerados sugieren que es suficiente con el número de cores disponible en el clúster para minimizar el tiempo de ejecución de forma efectiva sin llegar a requerirse en ningún caso el número máximo de procesadores para ello.

	Procesos MPI		Speed-up		
metaheurística	exp	mod	exp	mod	exp-auto
m6	20	18	10	7	9
m8	38	26	18	13	11
m9	40	37	24	23	20

Tabla 6.13: Comparativa del speed-up experimental más alto (exp), el speed-up modelado (mod) y el obtenido en los experimentos al usar la metodología de autooptimización (exp-auto), para las metaheurísticas m6, m8 y m9 de la tabla 6.1, al aplicar el esquema metaheurístico a PCEPA 50-6 en el clúster heterogéneo *Júpiter* + *Luna* + *Saturno* + *Marte* + *Mercurio*.

Se aprecia las contribuciones experimentales y teóricas al total de los tiempos de cómputo y comunicaciones en la figura 6.19 para las tres metaheurísticas consideradas. Como en los sistemas anteriores, las curvas teóricas simulan de forma aceptable la evolución experimental de los tiempos en función de p. Para m6, las comunicaciones no llegan a invertir la tendencia descendente del tiempo total quedando la curva final prácticamente plana. Sin embargo, la contribución de las comunicaciones es más marcada para m8 y m9 llegando a aparecer un mínimo relativo en el tiempo total como consecuencia.

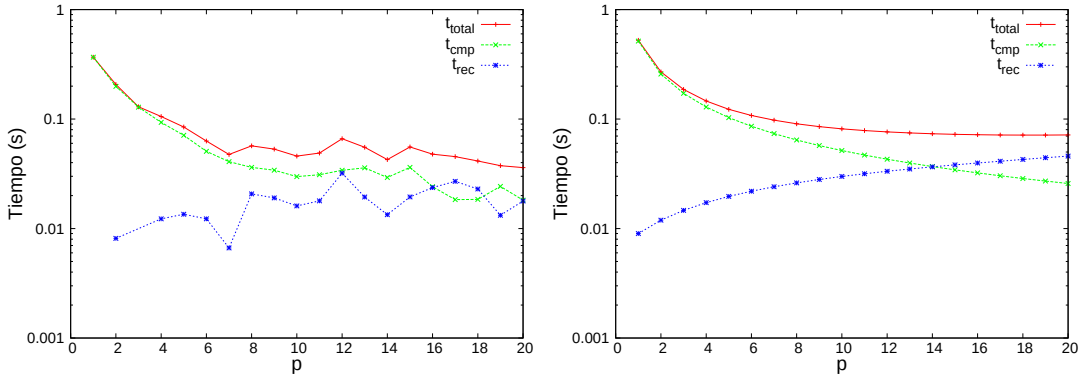
6.4. Conclusiones

En este capítulo se ha utilizado paralelismo de paso de mensajes para acelerar la aplicación de metaheurísticas ampliándose el esquema general a un esquema parametrizado en memoria distribuida. Se han introducido tres nuevos parámetros metaheurístico-paralelos ($NEMPar$, $NGMPar$ y p) que han permitido controlar la intensidad y la frecuencia del intercambio de información entre procesos, así como el volumen de datos transferidos.

Se ha empleado un modelo de islas implementado mediante un esquema maestro-esclavo que permite abordar la resolución del problema de optimización PCEPA aplicando el esquema general en paralelo. Aunque el objetivo fundamental del modelo utilizado es reducir el tiempo de ejecución del algoritmo y, por tanto, mejorar su eficiencia, los resultados de fitness permiten constatar que también se mejora su efectividad. Para ello, se han utilizado varios sistemas computacionales dentro de un clúster heterogéneo y se ha realizado un estudio del speed-up y fitness para distintos tamaños del problema variando las metaheurísticas aplicadas, el número y tipo de procesadores (p) usados, así como los otros dos nuevos parámetros de migración ($NEMPar$ y $NGMPar$).

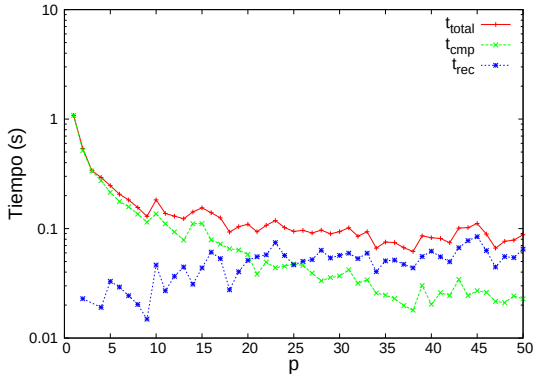
Se ha comprobado la utilidad del esquema metaheurístico en memoria distribuida como herramienta para reducir los tiempos de ejecución manteniendo la calidad de las soluciones obtenidas. Así, los speed-ups obtenidos en el sistema homogéneo *Saturno* son muy superiores a los alcanzados usando paralelismo de memoria compartida, siendo el fitness alcanzado con paso de mensajes similar al obtenido en memoria compartida en este sistema.

En cuanto a la influencia de los tres nuevos parámetros metaheurístico-paralelos, se ha comprobado que existe una dependencia del fitness obtenido con todos ellos. Así, son aconsejables valores de $NEMPar$ bajos que impliquen migraciones de en torno al 20 - 25 % (el valor concreto de $NEMPar$ dependerá del tamaño de las subpoblaciones), mientras que $NGMPar$ puede tomar valores de entre 5 y 10 con buenos resultados de fitness y tiempos de ejecución moderados. Respecto al número de islas, subpoblaciones o procesos a poner en marcha (p) se ha visto que existe, en todos los casos, un mínimo aconsejable de procesos que hacen que el fitness obtenido sea óptimo y que depende del tamaño de las metaheurísticas (poblaciones) elegido.

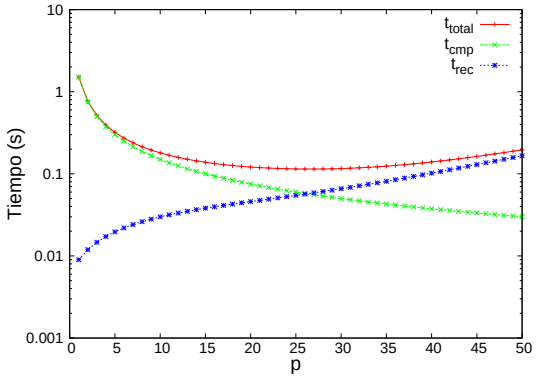


(a) m6 — tiempo experimental

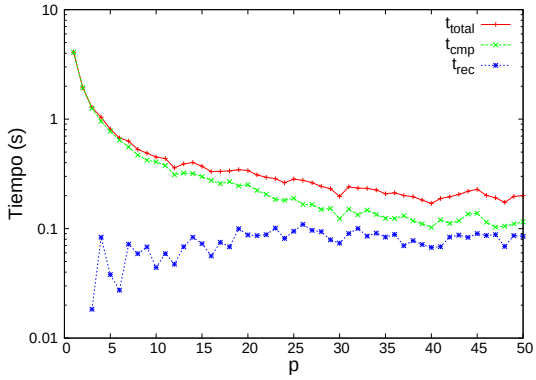
(b) m6 — tiempo modelado



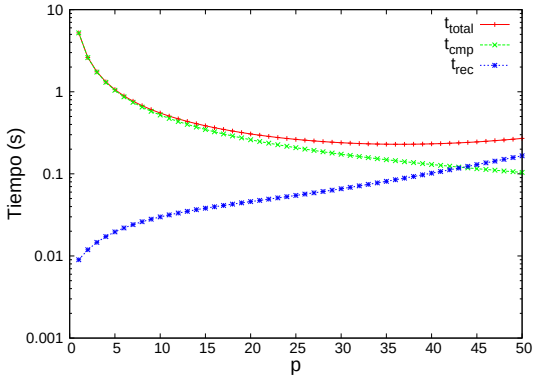
(c) m8 — tiempo experimental



(d) m8 — tiempo modelado



(e) m9 — tiempo experimental



(f) m9 — tiempo modelado

Figura 6.19: Evolución en el clúster heterogéneo *Júpiter* + *Luna* + *Saturno* + *Marte* + *Mercurio* de los tiempos experimentales y modelados de computación t_{cmp}, de recogida de elementos en el maestro t_{rec} y total $t_{total} = t_{cmp} + t_{rec}$, al aplicar la metodología de autooptimización a (a) y (b) la metaheurística m6, (c) y (d) la metaheurística m8 y (e) y (f) la metaheurística m9 de la tabla 6.1.

Se han llevado a cabo experimentos orientados a optimizar el speed-up en el sistema heterogéneo considerado teniendo en cuenta sobrecarga de procesadores y balanceo de carga. Los mejores resultados en términos de speed-up y calidad de la solución se han obtenido a través de un mapeo de procesos a procesadores basado en las velocidades relativas de los nodos del clúster. Además, para tamaños de población moderados, es mejor usar un número total de procesos cercano al tamaño de la población.

Adicionalmente se ha desarrollado satisfactoriamente un proceso de autooptimización orientado a reducir los tiempos de ejecución de las metaheurísticas paralelas. Se ha utilizado un modelo de tiempos que ha permitido predecir de manera adecuada, a partir de los resultados de la instalación en dos sistemas computacionales homogéneos y uno heterogéneo, la evolución del speed-up al aplicar distintas metaheurísticas al problema PCEPA. Así, el número de procesos óptimo determinado en todos los casos está próximo al mejor valor experimental y, por tanto, el speed-up seleccionado también está cerca del óptimo.

Capítulo 7

Conclusiones y trabajos futuros

Se presentan en este capítulo las conclusiones generales extraídas a partir de todo el trabajo de investigación desarrollado. Se pretende dar una visión general de las principales respuestas que se han dado a las hipótesis de partida planteadas en cada capítulo. Además, se muestran los documentos producidos durante la elaboración de la tesis así como los proyectos de investigación en los que se han desarrollado. Finalmente, se enumeran las líneas de investigación a seguir propuestas como trabajos futuros.

7.1. Conclusiones

Como se vio en la introducción, algunos problemas de optimización pueden ser abordados solo con métodos metaheurísticos, y para obtener una metaheurística satisfactoria es necesario normalmente desarrollar y experimentar con varios métodos y adaptar cada uno de ellos al problema particular con el que se está trabajando. Se pueden usar hiperheurísticas para seleccionar, combinar, generar o adaptar de manera automática varias metaheurísticas para resolver de manera eficiente problemas de optimización. En esta tesis se ha utilizado un esquema unificado parametrizado de metaheurísticas para facilitar el desarrollo de metaheurísticas y su aplicación reusando las funciones básicas que lo componen. Además, se ha usado el esquema general para desarrollar hiperheurísticas a un nivel de abstracción superior, por encima de las metaheurísticas parametrizadas, que han permitido seleccionar valores adecuados de los parámetros metaheurísticos y, como consecuencia, las propias metaheurísticas. La aplicabilidad del esquema metaheurístico parametrizado ha sido comprobada con la aplicación de métodos de búsqueda local y global (GRASP, Búsqueda Tabú, Algoritmos Genéticos y Búsqueda Dispersa) y sus combinaciones a dos problemas científicos: la optimización del consumo eléctrico en la explotación de pozos de agua (PCEPA) y la determinación de las constantes cinéticas de una reacción química (PCOCI). Además, como consecuencia de la aplicación de las hiperheurísticas, aparece un tercer problema de optimización consistente en la selección de la mejor metaheurística para resolver los dos anteriores (POPME).

Considerando la aplicación directa de metaheurísticas a los problemas de optimización planteados, se ha visto que cada problema individual se resuelve de manera óptima con un algoritmo diferente. En el caso de PCEPA y considerando solo el fitness como criterio de calidad de la solución, se recomienda el uso de la hibridación metaheurística SS+GA y sus combinaciones con otras metaheurísticas pues son las que ofrecen valores más bajos de fitness. Sin embargo, se ha visto que la metaheurística GA y sus combinaciones son las más adecuadas para la resolución del problema PCOCI. Se ha introducido un indicador común (IC), calculado como el inverso del producto del fitness por el tiempo de ejecución, para tener en cuenta también el tiempo de ejecución, lo que hace variar las conclusiones obtenidas. Así, se ha visto que, en algunos casos, se pueden obtener resultados de calidad aceptable en menos tiempo. Aunque las metaheurísticas aplicadas a los problemas ofrecen buenos resultados, se muestran normalmente eficaces solo para las instancias concretas en las que se usaron. Para solucionar esto y como una de las contribuciones de esta tesis, se ha experimentado con algoritmos hiperheurísticos más abstractos, permitiendo encontrar buenas metaheurísticas para conjuntos de varias instancias, evitando dependencias del problema.

Se ha visto la eficacia y la idoneidad de nuestra implementación hiperheurística para resolver el problema POPME a un nivel superior de abstracción cada vez que se aplicaba a cualquier problema de optimización. Primeramente se han definido tres configuraciones de hiperheurísticas que se han usado para experimentar con los problemas de optimización planteados, cada una de ellas con unos valores de parámetros diferentes, con valores más o menos altos para tener en cuenta con ello diferentes tiempos de ejecución totales. Además, se han tenido en cuenta varias maneras de computar el cálculo del fitness, persiguiendo en este caso un objetivo doble: evitar en lo posible la dependencia del problema reduciendo al máximo el tiempo de ejecución de la hiperheurística.

En concreto, los mejores resultados para el problema PCEPA, en términos de fitness y tiempo de ejecución, se obtienen aplicando una hiperheurística híbrida con valores moderados de los parámetros. Cuando el objetivo es obtener metaheurísticas satisfactorias para un conjunto de instancias del problema, es recomendable utilizar la metodología de aplicación de la hiperheurística a varias instancias de problema en una sola ejecución, lo que además reduce considerablemente los tiempos de ejecución de las hiperheurísticas. Respecto al problema PCOCI, se obtuvieron resultados similares aplicando las dos hiperheurísticas (reducida y genética) consideradas. Sin embargo, puede ser recomendable usar la hiperheurística reducida porque minimiza significativamente el tiempo de ejecución. En general, en los problemas estudiados las hiperheurísticas mejoran significativamente los resultados obtenidos al aplicar metaheurísticas puras directamente a los problemas y también mejoran el fitness resultante de la aplicación de la mejor metaheurística híbrida obtenida. Esto nos hace pensar que las hiperheurísticas basadas en esquemas metaheurísticos parametrizados son una buena herramienta de optimización que nos permite seleccionar metaheurísticas óptimas tanto para problemas específicos como para un amplio con-

junto de estos.

Aunque el esquema metaheurístico sea eficiente, su uso para resolver instancias de problema grandes causa incrementos significativos en el tiempo de ejecución. Basándonos en las posibilidades ofrecidas por las arquitecturas modernas de hardware, la aplicación de estrategias de computación de altas prestaciones a metaheurísticas es una interesante opción para reducir el tiempo de ejecución. En el presente trabajo, la paralelización de diferentes metaheurísticas se ha llevado a cabo a través del esquema unificado parametrizado de metaheurísticas, y por tanto las diferentes metaheurísticas que se obtienen a partir del esquema se paralelizan de un modo unificado. Se han utilizado dos paradigmas de paralelización diferentes pero complementarios entre sí: por un lado, paralelismo de memoria compartida a nivel local en cada máquina, y por otro, paralelismo global de paso de mensajes aprovechando la memoria distribuida y la interconexión entre distintos nodos dentro de un mismo clúster.

Los resultados obtenidos utilizando ambos tipos de paralelismo han sido satisfactorios al aplicarse sobre los problemas PCEPA y POPME. Se consideró primero el uso de paralelismo en memoria compartida, modificándose el esquema metaheurístico secuencial con la introducción de parámetros paralelos en sus funciones básicas. Se ha constatado su utilidad para reducir el tiempo de ejecución de los algoritmos, sobre todo en el caso de las hiperheurísticas, puesto que son los métodos más costosos computacionalmente. Se ha visto que lo realmente efectivo a la hora de paralelizar es utilizar una metodología sistemática y rigurosa que asegure una reducción óptima de los tiempos de ejecución. Así, se ha introducido satisfactoriamente la técnica de autooptimización basada en un modelo teórico del tiempo de ejecución aplicada a metaheurísticas e hiperheurísticas paralelas, lo que significa una contribución en este campo. Después de desarrollar la metodología y de obtener valores de las constantes y parámetros de paralelismo, se puede decir que la técnica proporciona valores satisfactorios del número de hilos a usar en sistemas NUMA, con valores de speed-up, en los mejores casos, de hasta 16 para las metaheurísticas y de hasta 7 para las hiperheurísticas.

Por otro lado, se ha utilizado paralelismo de paso de mensajes para acelerar la aplicación de metaheurísticas ampliándose el esquema general a un esquema parametrizado en memoria distribuida. Se introdujeron nuevos parámetros metaheurístico-paralelos permitiendo controlar la intensidad y la frecuencia del intercambio de información entre procesos, así como el volumen de datos transferidos. Se empleó un modelo de islas implementado mediante un esquema maestro-esclavo con el objetivo fundamental de reducir el tiempo de ejecución del algoritmo y, por tanto, mejorar su eficiencia, aunque los resultados de fitness permiten constatar que también se mejora su efectividad. Para ello, se han utilizado varios sistemas computacionales dentro de un clúster heterogéneo y se ha realizado un estudio del speed-up y fitness para distintos tamaños del problema variando las metaheurísticas aplicadas, el número y tipo de procesadores usados, así como los otros parámetros de migración.

El esquema metaheurístico en memoria distribuida ha resultado útil como herra-

mienta para reducir los tiempos de ejecución manteniendo la calidad de las soluciones obtenidas. Así, los speed-ups obtenidos en el sistema homogéneo *Saturno* son muy superiores a los alcanzados usando paralelismo de memoria compartida (hasta 10x en algunos casos), siendo el fitness alcanzado con paso de mensajes similar al obtenido en memoria compartida en este sistema. En cuanto a la influencia de los nuevos parámetros metaheurístico-paralelos introducidos con paso de mensajes, se ha comprobado, en sistemas homogéneos, que existe una dependencia del fitness obtenido con todos ellos. Así, es recomendable migrar cada 5 a 10 iteraciones, con una cantidad de individuos de entre el 20 - 25 % del total, y con la existencia, en general, de un número de islas óptimo que depende del tamaño de la población total elegida. Se han llevado a cabo experimentos orientados a optimizar el speed-up en el sistema heterogéneo considerado teniendo en cuenta sobrecarga de procesadores y balanceo de carga. Los mejores resultados en términos de speed-up y calidad de la solución se han obtenido a través de un mapeo de procesos a procesadores basado en las velocidades relativas de los nodos del clúster. Además, para tamaños de población moderados (hasta 200 individuos), es mejor usar un número total de procesos cercano al tamaño de la población. Finalmente, se ha aplicado satisfactoriamente un proceso de autooptimización orientado a reducir los tiempos de ejecución de las metaheurísticas paralelas en un sistema computacional homogéneo. Se ha utilizado un modelo de tiempos que ha permitido predecir de manera adecuada el número de procesos óptimo y la evolución del speed-up al aplicar distintas metaheurísticas al problema PCEPA. Se ha comprobado la utilidad de la metodología de autooptimización en otros sistemas (un clúster homogéneo y un clúster heterogéneo), obteniéndose resultados satisfactorios en ambos casos.

7.2. Resultados y entorno de trabajo

El trabajo de esta tesis se enmarca dentro de varios proyectos de investigación regionales y nacionales:

- Proyecto de la Comisión Interministerial de Ciencia y Tecnología, "Construcción y optimización de librerías científicas paralelas (TIN2008-06570-C04-02)", coordinado con grupos de investigación de la Universidad Jaume I, Universidad de La Laguna y Universidad de Alicante, desarrollado desde el 1 de enero de 2009 hasta el 31 de diciembre de 2011.

- Proyecto del Ministerio de Economía y Competitividad, "Mejora de arquitectura de servidores, servicios y aplicaciones (TIN2012-38341-C04-03)", coordinado con grupos de investigación de la Universidad de Castilla-La Mancha y Universidad Politécnica de Valencia, desarrollado desde el 1 de enero de 2013 hasta el 31 de diciembre de 2015.

- Proyecto de la Consejería de Cultura y Educación de la Región de Murcia, Fundación Séneca, "Adaptación y Optimización de Código Científico en Siste-

mas Computacionales Jerárquicos (08763/PI/08)", coordinado con Universidad Politécnica de Cartagena, desarrollado desde el 1 de enero de 2009 hasta el 31 de diciembre de 2013.

Adicionalmente cabe destacar la participación en las siguientes Redes de Investigación:

- Red Europea, COST actions, "Open European Network for High-Performance Computing in Complex Environments", coordinado con grupos de investigación de varias universidades españolas entre ellas la Universidad de Extremadura, Universidad Jaime I, Universidad Politécnica de Valencia y Universidad de La Laguna, además de otras entidades europeas, desarrollado desde marzo de 2009 hasta marzo de 2013.

- Dirección General de Investigación, Ministerio de Educación y Ciencia, "Red de computación de altas prestaciones sobre arquitecturas paralelas heterogéneas (CAPAP-H3) (TIN2010-12011-E)", desarrollado desde el 15 de junio de 2011 hasta el 14 de junio de 2012 (ampliado a 2013).

- Red Murciana de Investigación, Formación e Innovación Tecnológica en e-Salud (Infinite-Salud).

Durante la realización de la tesis se han generado varias publicaciones y comunicaciones que se relacionan a continuación junto con una breve descripción de cada una:

- Resolución de un problema de optimización de consumo eléctrico en explotación de pozos por medio de metaheurísticas parametrizadas. José-Matías Cutillas-Lozano, Luis-Gabino Cutillas-Lozano y Domingo Giménez. VIII Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB 2012), pp 391–398, 2012. [53]

  En esta comunicación se aborda la resolución de un problema de optimización de costes de consumo eléctrico en la explotación de pozos mediante métodos metaheurísticos. Para obtener una metaheurística satisfactoria se utiliza un esquema unificado parametrizado en memoria compartida de metaheurísticas. Se facilita así el desarrollo de metaheurísticas reutilizando las funciones básicas y se reducen los tiempos de ejecución. Dado un sistema computacional particular y fijados los parámetros para la metaheurística secuencial, la selección apropiada de parámetros en el esquema unificado paralelo facilita el desarrollo de metaheurísticas paralelas eficientes. Se aplicaron las metaheurísticas GRASP, Algoritmos Genéticos, Búsqueda Dispersa y sus combinaciones. El contenido de este trabajo se recoje en el capítulo 3 de esta tesis.

- Modeling Shared-Memory Metaheuristic Schemes for Electricity Consumption. Luis-Gabino Cutillas-Lozano, José-Matías Cutillas-Lozano and Domingo Giménez. In proceedings of the 9th International Symposium on Distributed Computing and Artificial Intelligence (DCAI'12), pp 33–40, 2012. [61]

  Esta comunicación amplía el estudio hecho en la anterior con la introducción del modelado y autooptimización del esquema metaheurístico en memoria compartida. El modelo de tiempos permite decidir durante la ejecución el número de hilos óptimo para obtener un tiempo de ejecución reducido. Se analiza la forma en que el número de hilos depende del problema a resolver, de la implementación de las funciones del esquema, del sistema computacional, etc. Este trabajo se incluye en el capítulo 5 de esta tesis.

- Modelling Parameterized Shared-memory Hyperheuristics for Auto-Tuning. José-Matías Cutillas-Lozano, Domingo Giménez and Luis-Gabino Cutillas-Lozano. In proceedings of the 12th International Conference on Computational and Mathematical Methods in Science and Engineering (CMMSE'12), pp 389–400, 2012. [59]

  Se presenta aquí, por primera vez, el desarrollo de la metodología de modelado y autooptimización aplicada a las hiperheurísticas basadas en esquemas metaheurísticos parametrizados, ampliándose los contenidos del trabajo anterior. Se resalta la necesidad de utilizar la metodología debido al elevado coste computacional de las hiperheurísticas que se encargan de seleccionar la mejor metaheurística para un problema dado, en este caso el de optimización de consumo eléctrico en pozos. El contenido de este trabajo se recoje en el capítulo 5 de esta tesis.

- Use of parallel, parameterized metaheuristics for the determination of the kinetic constants of a chemical reaction in heterogeneous phase. José-Matías Cutillas-Lozano and Domingo Giménez. Simposio Doctoral de la red Infinite Salud, Murcia 2013. [55]

  En esta comunicación se aborda la resolución del problema de determinación de las constantes cinéticas de una reacción química en fase heterogénea mediante el uso de metaheurísticas. El uso de un esquema metaheurístico facilita la aplicación de varias metaheurísticas simplemente modificando algunos parámetros del esquema. Así, se aplicaron las metaheurísticas GRASP, Algoritmos Genéticos, Búsqueda Dispersa, Búsqueda Tabú y las combinaciones de estas. Este trabajo se incluye en el capítulo 3 de esta tesis.

- Determination of the kinetic constants of a chemical reaction in heterogeneous phase using parameterized metaheuristics. José-Matías Cutillas-Lozano and Domingo Giménez. In proceedings of the 2013 International Conference on Computational Science (ICCS'13), pp 787–796, 2013. [54]

Se trata de una comunicación a un *workshop* de química computacional que amplía el anterior trabajo. Se obtuvieron resultados satisfactorios de las constantes cinéticas que optimizaban el sistema químico estudiado, así como los valores de la evolución temporal de todas las especies químicas relacionadas. Se incluye en el capítulo 3 de esta tesis.

- Using hyperheuristics to improve the determination of the kinetic constants of a chemical reaction in heterogeneous phase. José-Matías Cutillas-Lozano and Domingo Giménez. In proceedings of the 2014 International Conference on Computational Science (ICCS'14), admitido como póster, 2014. [58]

  En esta comunicación (también a *workshop* de química computacional) se pretende mejorar los resultados de [54] y [55] con la aplicación de hiperheurísticas como algoritmos de selección de la mejor metaheurística para el problema de determinación de constantes cinéticas. El estudio estadístico realizado confirma que el fitness obtenido tanto con la aplicación directa de hiperheurísticas como con la aplicación de metaheurísticas obtenidas a partir de hiperheurísticas, mejora los resultados obtenidos con las mejores combinaciones metaheurísticas de los trabajos anteriores. Este trabajo se incluye en el capítulo 4 de esta tesis.

- Optimizing Shared-Memory Hyperheuristics on top of Parameterized Metaheuristics. José-Matías Cutillas-Lozano and Domingo Giménez. In proceedings of the 2014 International Conference on Computational Science (ICCS'14), admitido, 2014. [56]

  Se amplían los resultados presentados en [59] al aplicar la metodología de modelado y autooptimización de metaheurísticas e hiperheurísticas basadas en esquemas metaheurísticos parametrizados. Se ha realizado un modelado global de todas las funciones del esquema presentándose tablas con todos los valores de las constantes del sistema y número de hilos óptimo. Se ha constatado que la metodología permite seleccionar adecuadamente los parámetros de paralelismo que permiten obtener valores de speed-ups óptimos cercanos a los alcanzados experimentalmente. El contenido de este trabajo se recoje en el capítulo 5 de esta tesis.

- Hyperheuristics based on parameterized metaheuristic schemes. José-Matías Cutillas-Lozano and Francisco Almeida and Domingo Giménez. In proceedings of the 2014 Genetic and Evolutionary Computation Conference (GECCO'14), admitido como póster, 2014. [51]

  Esta comunicación tiene como objetivo mostrar la utilidad de las hiperheurísticas basadas en esquemas metaheurísticos parametrizados como algoritmos eficaces para la selección de las mejores metaheurísticas para un problema de optimización dado. Los resultados obtenidos y su análisis estadístico así lo confirman, mejorando los resultados de fitness obtenidos al aplicar directamente el esquema metaheurístico al problema de optimización del consumo

electrico en la explotación de pozos de agua. En este caso no se ha dado prioridad al tiempo de ejecución en los resultados, cuyo estudio detallado se ha realizó en [59]. Este trabajo se incluye en el capítulo 4 de esta tesis.

- Parameterized message-passing metaheuristic schemes. José-Matías Cutillas-Lozano and Domingo Giménez. In proceedings of the 2014 Genetic and Evolutionary Computation Conference (GECCO'14), admitido como póster, 2014. [57]

  Se presenta aquí un estudio sobre la paralelización en memoria distribuida del esquema metaheurístico parametrizado, haciéndose una comparación con los resultados obtenidos en memoria compartida. Se ha visto que el paradigma de paso de mensajes implementado con un modelo de islas usando el esquema maestro-esclavo, mejora considerablemente los resultados de speed-up obtenidos con memoria compartida, con speed-up superlineales en algunos casos. Se ha estudiado la influencia sobre el tiempo de ejecución y el fitness de los nuevos parámetros de paralelismo introducidos tanto en sistemas homogéneos como heterogéneos, y se han determinado las configuraciones paramétricas más adecuadas en ambos casos. Este trabajo se incluye en el capítulo 6 de esta tesis.

- Hyperheuristics based on parameterized metaheuristic schemes. José-Matías Cutillas-Lozano, Francisco Almeida, and Domingo Giménez. In the 5th International Conference on Metaheuristics and Nature Inspired Computing, META'2014, admitido, 2014. [52]

  Esta comunicación sigue la línea de [51] y pretende mostrar la utilidad de las hiperheurísticas basadas en esquemas metaheurísticos parametrizados como algoritmos eficaces para la selección de las mejores metaheurísticas para un problema de optimización dado. Se presentan conclusiones similares a las del trabajo previo citado. Este trabajo se incluye en el capítulo 4 de esta tesis.

Cabe destacar que los congresos ICCS y GECCO, donde se han presentado o admitido trabajos en los años 2013 y 2014, tienen categoría "A"dada por la *Computing Research and Education Association of Australasia (CORE).*

7.3. Trabajos futuros

En base a los resultados obtenidos en esta tesis se consideran una serie de líneas de investigación a corto y medio plazo que pueden complementar o ampliar el trabajo aquí presentado, y que se se pueden agrupar en distintas direcciones de trabajo futuro:

- Metaheurísticas. Las líneas de investigación a seguir pueden ser:

  - Ampliar el número de metaheurísticas puras aplicadas mediante el esquema unificado parametrizado, considerando, por ejemplo, algoritmos como Colonia de Hormigas, Recocido Simulado, etc. De este modo aparecerían nuevos parámetros cuya selección adecuada permite crear nuevas metaheurísticas híbridas.
  - Además, se pueden considerar también parámetros de tipo categórico con la idea de seleccionar bloques de heurísticas enteras para formar algoritmos específicos para cada problema particular, con el posterior ajuste fino de cada heurística mediante la selección apropiada de parámetros de tipo numérico.
  - Se puede adaptar el esquema metaheurístico parametrizado a problemas multiobjetivo.

- Hiperheurísticas. Algunas líneas de investigación importantes son:

  - Se pretende aplicar las hiperheurísticas a otros problemas de optimización, más ampliamente estudiados, para comparar nuestro enfoque con otras implementaciones hiperheurísticas. Lo dicho en el punto anterior es aplicable también a las hiperheurísticas puesto que se basan en el mismo esquema metaheurístico.
  - Además, se plantea la posibilidad de aplicar las hiperheurísticas con nuevas formas de computar el fitness para comparar los resultados con los obtenidos en este trabajo.
  - Se puede plantear un problema bi-objetivo para la búsqueda de metaheurísticas paralelas optimizando el fitness y el tiempo de ejecución. Se sustituiría de esta forma el uso de un indicador común para indicar la bondad de la metaheurística teniendo en cuenta esos dos factores.

- Esquemas metaheurísticos paralelos. Se pueden considerar en un futuro los siguientes aspectos:

  - El paralelismo de paso de mensajes ha resultado especialmente útil con metaheurísticas y se puede aplicar también a hiperheurísticas, donde la aplicación del modelo de islas puede disminuir de manera considerable los tiempos de ejecución para estos algoritmos de alto coste.
  - A medio plazo, la paralelización del esquema metaheurístico en GPU. Este tipo de paralelismo puede ser especialmente beneficioso para la aplicación de hiperheurísticas con grandes conjuntos de referencia o con elevados costes de la función de fitness.
  - Así mismo, se pretende a corto plazo establecer modelos de tiempo con la correspondiente autooptimización de esquemas metaheurísticos en sistemas heterogéneos, con GPU y con paralelismo híbrido, para conseguir un

control más preciso de los parámetros paralelos que permitan minimizar los tiempos de ejecución al utilizar de forma eficiente sistemas computacionales jerárquicos de alta complejidad.

- También se considera el diseño de esquemas de paso de mensajes con una distribución de datos heterogénea y un número de procesos igual al número de cores. Se busca así realizar una asignación más racional de datos a procesadores de características diferentes, pudiéndose comparar los resultados con los obtenidos con la asignación homogénea de datos y ver si la división proporcional de datos mejora los speed-ups alcanzados.
- Todo el paralelismo anterior podría ser combinado, a largo plazo, para conseguir esquemas de metaheurísticas e hiperheurísticas paralelas híbridas ejecutadas en clústers heterogéneos.

- Aplicación de la metodología a otros problemas de optimización. Algunos problemas en los que se ha empezado a trabajar en colaboración con otros grupos de investigación son:

  - Ampliar los casos de estudio a más problemas de determinación de parámetros químicos u optimización de procesos.
  - Aplicación de hiperheurísticas al problema de obtención y resolución de modelos de ecuaciones simultáneas.
  - Estudio de otros problemas de optimización más ampliamente utilizados como test en investigación operativa incluyendo problemas multiobjetivo.

Apéndice A

Modelos de tiempos

En este capítulo desarrollamos los cálculos necesarios para obtener las ecuaciones de los modelos de tiempos para los esquemas metaheurísticos en memoria compartida y distribuida.

A.1. Modelo de tiempos para memoria compartida

Se presenta a continuación el desarrollo matemático que lleva a la obtención de las ecuaciones para el número de hilos óptimo para las funciones de uno y dos niveles de paralelismo del esquema hiperheurístico paralelo (HMS). De la misma forma se obtienen la ecuaciones para el esquema metaheurístico, con algunas modificaciones en los resultados finales debidas a la diferente estructura de algunas funciones básicas.

Para las funciones de un nivel de paralelismo, el tiempo de ejecución en función de las constantes del sistema k_{s1} y k_p y de los parámetros metaheurístico NE y de paralelismo p viene dado por:

$$t_{un-nivel} = \frac{k_{s1} \cdot NE}{p} + k_p \cdot p \tag{A.1}$$

y para encontrar el óptimo respecto al número de hilos p:

$$\frac{dt}{dp} = \frac{-k_{s1} \cdot NE}{p^2} + k_p = 0 \tag{A.2}$$

$$p_{opt.} = \sqrt{\frac{k_{s1} \cdot NE}{k_p}} \tag{A.3}$$

Puesto que tenemos tres funciones con un nivel de paralelismo (GenerarConjuntoInicial dentro de `Inicializar`, `Combinar` e `Incluir`), las constantes y parámetros a utilizar dentro de la ecuación A.3 serán diferentes, como viene recogido en la tabla

A.1. Los valores de k_{s1} y k_p óptimos para una combinación dada de parámetros metaheurísticos NE se obtienen como función del tiempo de ejecución y del número de hilos p puestos en marcha en cada experimento, minimizando la suma de los cuadrados de las diferencias entre los tiempos experimentales y los teóricos obtenidos con el modelo dado por la ecuación A.1 al variar el parámetro de paralelismo.

$F_{un-nivel}$	k_{s1}	NE
Gen-Ini	k_g	$NEIIni$
Combinar	k_c	$2 \cdot (NMMCom + NMPCom + NPPCom)$
Incluir	k_i	$NEFIni + 2 \cdot (NMMCom + NMPCom + NPPCom) - NEMInc$

Tabla A.1: Valores específicos de las constantes del sistema y de los parámetros metaheurísticos para cada función considerada con un nivel de paralelismo. Con Gen-Ini representando la generación del conjunto inicial de elementos dentro de `Inicializar`.

De igual manera, para las funciones de dos niveles de paralelismo, podemos calcular el tiempo de ejecución en función de las constantes del sistema k_{s2}, $k_{p,1}$ y $k_{p,2}$ y de los parámetros metaheurísticos $Param$ y de paralelismo p_1 y p_2:

$$t_{dos-niveles} = \frac{k_{s2} \cdot Param}{p_1 \cdot p_2} + k_{p,1} \cdot p_1 + k_{p,2} \cdot p_2 \tag{A.4}$$

con $Param = \frac{NE \cdot PM \cdot IM}{100}$ y cada parámetro tomando un valor específico para cada función del esquema, como podemos ver en la tabla A.2. Si derivamos el tiempo respecto al número de hilos de primer y segundo nivel, obtendremos:

$$\frac{dt}{dp_1} = \frac{-k_{s2} \cdot Param}{p_1^2 \cdot p_2} + k_{p,1} = 0 \tag{A.5}$$

$$\frac{dt}{dp_2} = \frac{-k_{s2} \cdot Param}{p_1 \cdot p_2^2} + k_{p,2} = 0 \tag{A.6}$$

y resolviendo el sistema formado por las ecuaciones A.5 y A.6, obtenemos los valores óptimos del número de hilos de primer y segundo nivel en función de los parámetros de paralelismo y de las constantes del sistema:

$$p_{1,opt.} = \sqrt[3]{\frac{k_{s2} \cdot k_{p,2}}{k_{p,1}^2} \cdot Param} \tag{A.7}$$

$$p_{2,opt.} = \sqrt[3]{\frac{k_{s2} \cdot k_{p,1}}{k_{p,2}^2} \cdot Param} \tag{A.8}$$

Finalmente, las constantes del sistema y parámetros metaheurísticos específicos para las cuatro funcines básicas de dos niveles de paralelismo consideradas, vienen

recogidos en la tabla A.2. Al igual que para las funciones de un nivel, los valores de k_{s2}, $k_{p,1}$ y $k_{p,2}$ óptimos para una combinación dada de parámetros metaheurísticos $Param$ se obtienen como función del tiempo de ejecución y del número de hilos p_1 y p_2 puestos en marcha en cada experimento, minimizando la suma de los cuadrados de las diferencias entre los tiempos experimentales y los teóricos obtenidos con el modelo dado por la ecuación A.4 al variar los parámetros de paralelismo.

$F_{dos-niveles}$	k_{s2}	NE	PM	IM
Mej-Ini	k_{mi}	$NEIIni$	$PEMIni$	$IMEIni$
Mej-Ref	k_{mr}	$NEFIni$	$PEMMej$	$IMEMej$
Mej-Com	k_{mc}	$2 \cdot (NMMCom + NMPCom + NPPCom)$	$PEMMej$	$IMEMej$
Div	k_d	$NEFIni$	$PEDMej$	$IDEMej$

Tabla A.2: Valores específicos de las constantes del sistema y de los parámetros metaheurísticos para cada función considerada con dos niveles de paralelismo. Con mejora en la inicialización (Mej-Ini), mejora en el conjunto de referencia (Mej-Ref), mejora en el conjunto de combinaciones (Mej-Com) y diversificación (Div).

A.2. Modelo de tiempos para paso de mensajes

Presentamos aquí la deducción matemática de las ecuaciones que definen el modelo de tiempo para el paradigma de paso de mensajes, y que lleva a la obtención de los parámetros de paralelismo óptimos como función de las constantes del sistema y de los parámetros metaheurísticos dentro del proceso de autooptimización.

El tiempo de ejecución total t_{total} del esquema metaheurístico en memoria distribuida viene dado por:

$$t_{total} = t_{cmp} + t_{cmc} + t_{ord} \tag{A.9}$$

donde t_{cmp} es el tiempo de cómputo, t_{cmc} es el tiempo empleado en las comunicaciones y t_{ord} es el tiempo de ordenación de elementos en el proceso maestro.

Se ha comprobado experimentalmente que el término de ordenación es despreciable en el rango de combinaciones de parámetros considerado, por lo que el tiempo total vendría dado únicamente por el término de cómputo más el de comunicaciones:

$$t_{total} = t_{cmp} + t_{cmc} \tag{A.10}$$

Desarrollando cada término de A.10, tenemos primero la computación como la suma de los tiempos empleados en cada función del esquema parametrizado:

$$t_{cmp} = \sum_{i=1}^{2} k_i \cdot \frac{Param_i}{p} + \left(\sum_{j=1}^{6} k_j \cdot \frac{Param_j}{p} \right) \cdot NGMPar \tag{A.11}$$

con los valores específicos de las constantes y parámetros metaheurísticos para cada función del esquema en la tabla A.3. Se determinarán mediante un ajuste de mínimos cuadrados los valores de estas constantes secuenciales que mejor se ajustan a nuestros datos experimentales durande el proceso de instalación.

$func$	k_i	$Param_i$
Gen-Ini	k_g	$NEIIni$
Mej-Ini	k_{mi}	$\frac{NEIIni \cdot PEMIni \cdot IMEIni}{100}$

$func$	k_j	$Param_j$
Com	k_c	$2 \cdot (NMMCom + NMPCom + NPPCom)$
Mej-Ref	k_{mr}	$\frac{NEFIni \cdot PEMMej \cdot IMEMej}{100}$
Mej-Com	k_{mc}	$\frac{(NMMCom+NMPCom+NPPCom) \cdot PEMMej \cdot IMEMej}{100}$
Div-Ref	k_{dr}	$\frac{NEFIni \cdot PEDMej \cdot IDEMej}{100}$
Div-Com	k_{dc}	$\frac{(NMMCom+NMPCom+NPPCom) \cdot PEDMej \cdot IDEMej}{100}$
Inc	k_i	$NEFIni + 2 \cdot (NMMCom + NMPCom + NPPCom) - NEMInc$

Tabla A.3: Valores específicos de las constantes secuenciales del sistema y de los parámetros metaheurísticos considerados en la ecuación A.11. Funciones: Gen-Ini, generación de elementos en la inicialización; Mej-Ini, Mej-Ref y Mej-Com, mejora de elementos en la inicialización, mejora del conjunto de referencia y del de combinaciones respectivamente; Com, combinación; Div-Ref y Div-Com, diversificación de los conjuntos de referencia y combinaciones; Inc, función de inclusión de elementos.

Además t_{cmc} se subdivide en dos:

$$t_{cmc} = t_{rec} + t_{dif} \tag{A.12}$$

con

$$t_{rec} = f(p, NEMPar) \tag{A.13}$$

y

$$t_{dif} = g(p, NEMPar) \tag{A.14}$$

donde t_{rec} es el tiempo empleado en recoger por parte del maestro todos los elementos enviados de desde los $p-1$ procesos esclavos y t_{dif} es el tiempo de difusión de elementos desde el maestro a todos los esclavos.

Se ha determinado experimentalmente que el tiempo de difusión t_{dif} es despreciable en el rango de combinaciones de parámetros considerado, por lo que el tiempo total de comunicaciones coincide con t_{rec}:

$$t_{cmc} = t_{rec} \tag{A.15}$$

por tanto habrá que determinar la función $f(p, NEMPar)$ que mejor se ajuste a los datos experimentales para poder modelar el tiempo de recogida de elementos t_{rec} en el maestro. Se ha probado a hacer el ajuste con varias funciones, determinándose los correspondientes valores de sus coeficientes de regresión. Se ha decidido utilizar una función polinómica de grado tres (en el caso de *Saturno*) por ser la que ofrece un ajuste razonablemente bueno y con una complejidad relativamente baja. Así, tenemos que:

$$t_{rec} = A \cdot p^3 + B \cdot p^2 + C \cdot p + D \tag{A.16}$$

donde t_{rec} depende de únicamente de p, considerando el ajuste experimental de datos para varios tamaños de $NEMPar$ teniendo en cuenta que, según la figura 6.5, existe poca variación del tiempo total con $NEMPar$. El resto de sistemas computacionales se han modelado de igual forma ajustándose los datos a distintas funciones: logarítmica para el sistema homogéneo *Marte* + *Mercurio* y polinómica de grado tres para el clúster heterogéneo *Saturno* + *Júpiter* + *Marte* + *Mercurio* + *Luna*.

Una vez determinadas las constantes del modelo y la función que ajusta el tiempo de las comunicaciones, si derivamos el tiempo total respecto al número de procesos p, obtendremos el valor de p óptimo que permite alcanzar los mejores speed-ups teóricos. Así, para *Saturno*:

$$\frac{dt_{total}}{dp} = \frac{-K_{cmp}}{p^2} + 3 \cdot A \cdot p^2 + 2 \cdot B \cdot p + C = 0 \tag{A.17}$$

donde se ha fijado $NGMPar$ a valores razonables (1 para minimizar el fitness según la figura 6.3), y donde K_{cmp} engloba la suma de todas las constantes y parámetros metaheurísticos del término de computación recogidos en la tabla A.3:

$$K_{cmp} = \sum_{i=1}^{2} k_i \cdot Param_i + \left(\sum_{j=1}^{6} k_j \cdot Param_j \right) \cdot NGMPar \tag{A.18}$$

Resolviendo la ecuación A.17 mediante el método iterativo ofrecido por la función "Solver"de *OpenOffice*, tenemos el número de procesos óptimo $p_{opt.}$ para cada combinación de parámetros metaheurísticos para cada problema.

Bibliografía

[1] K. Aida, W. Natsume, and Y. Futakata. Distributed computing with hierarchical master-worker paradigm for parallel branch and bound algorithm. In *Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2003)*, pages 156–163, 2003.

[2] E. Alba. *Parallel Metaheuristics: A New Class of Algorithms*. Wiley-Interscience, 2005.

[3] E. Alba, F. Almeida, M. J. Blesa, C. Cotta, M. Díaz, I. Dorta, J. Gabarró, C. León, G. Luque, and J. Petit. Efficient parallel LAN/WAN algorithms for optimization. The mallba project. *Parallel Computing*, 32(5-6):415–440, 2006.

[4] E. Alba, F. Luna, and A. J. Nebro. Advances in parallel heterogeneous genetic algorithms for continuous optimization. *International Journal of Applied Mathematics and Computer Science*, 14(3):101–117, 2004.

[5] F. Almeida, J. Cuenca, D. Giménez, A. Llanes-Castro, and J. P. Martínez-Gallar. A framework for the application of metaheuristics to tasks-to-processors assignation problems. *Journal of Supercomputing*, 62(2):2012, 698-723.

[6] F. Almeida, D. Giménez, and J.-J. López-Espín. A parameterized shared-memory scheme for parameterized metaheuristics. *The Journal of Supercomputing*, 58(3):292–301, 2011.

[7] F. Almeida, D. Giménez, J.-J. López-Espín, and M. Pérez-Pérez. Parameterised schemes of metaheuristics: basic ideas and applications with Genetic algorithms, Scatter Search and GRASP. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 43(3):570–586, 2013.

[8] F. Almeida, D. Giménez, J. M. Mantas, and A. M. Vidal. *Introducción a la programación paralela*. Paraninfo Cengage Learning, 2008.

[9] C. Ansótegui, M. Sellmann, and K. Tierney. A gender-based genetic algorithm for the automatic configuration of algorithms. In *CP*, pages 142–157, 2009.

[10] C. A. J. Appelo and D. Postma. *Geochemistry, Groundwater and Pollution*. A. A. Balkema, Rotterdam, 1993.

[11] S. Asta, E. Özcan, and A. J. Parkes. Batched mode hyper-heuristics. In *7th Learning and Intelligent OptimizatioN Conference (LION13)*. published online, 2013.

[12] S. Asta, E. Özcan, A. J. Parkes, and A. S. Etaner-Uyar. Generalizing hyper-heuristics via apprenticeship learning. In *EvoCOP*, pages 169–178, 2013.

[13] A. Azadeh and Z. S. Faiz. A meta-heuristic framework for forecasting household electricity consumption. *Applied Soft Computing*, 11(1):614 – 620, 2011.

[14] A. Azadeh, S. F. Ghaderi, S. Tarverdian, and M. Saberi. Integration of artificial neural networks and genetic algorithm to predict electrical energy consumption. *Applied Mathematics and Computation*, 186(2):1731–1741, 2007.

[15] T. Bäck, D. Fogel, and Z. Michalewicz. *Handbook of Evolutionary Computation*. Institute of Physics Publishing and Oxford University Press, 1997.

[16] Barcelona Supercomputing Center. `www.bsc.es`.

[17] A. Bendjoudi, N. Melab, and E.-G. Talbi. An adaptive hierarchical master-worker (AHMW) framework for grids - application to B&B algorithms. *J. Parallel Distrib. Comput.*, 72(2):120–131, 2012.

[18] G. Bilardi and F. P. Preparata. Horizons of parallel computation. *J. Parallel Distrib. Comput*, 27(2):172–182, 1995.

[19] B. Bilgin, E. Özcan, and E. E. Korkmaz. An experimental study on hyper-heuristics and exam timetabling. In *PATAT*, pages 394–412, 2006.

[20] M. Birattari. *The Problem of Tuning Metaheuristics as seen from a machine lerning perspective*. PhD thesis, Universite Libre de Bruxelles, 2004.

[21] R. H. Bisseling. *Parallel Scientific Computation*. Oxford University Press, 2004.

[22] C. Blum, M. J. Blesa Aguilera, A. Roli, and M. Sampels. *Hybrid Metaheuristics – An Emerging Approach to Optimization*, volume 114. Springer, 2008. Studies in Computational Intelligence.

[23] C. Blum, J. Puchinger, G. R. Raidl, and A. Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Appl. Soft Comput.*, 11(6):4135–4151, 2011.

[24] Boletín Oficial del Estado. Real Decreto 140/2003, de 7 de febrero, por el que se establecen los criterios sanitarios de la calidad del agua de consumo humano, 2003.

[25] Boletín Oficial del Estado. ORDEN ITC/3860/2007, de 28 de diciembre, por la que se revisan las tarifas eléctricas a partir del 1 de enero de 2008, 2007.

[26] M. Boratto, P. Alonso, D. Giménez, M. Barreto, and K. Oliveira. Auto-tuning methodology to represent landform attributes on multicore and multi-GPU systems. In *PMAM*, pages 125–132, 2013.

[27] P. Borovska. Efficiency of parallel metaheuristics for solving combinatorial problems. In *CompSysTech*, page 15, 2007.

[28] W. Bozejko and M. Wodecki. Parallel scatter search algorithm for the flow shop sequencing problem. In *PPAM*, pages 180–188, 2007.

[29] E. Burke, E. Hart, G. Kendall, J. Newall, P. Ross, and S. Schulenburg. *Hyper-heuristics: An Emerging Direction in Modern Search Technology*, chapter 16, pages 457–474. Kluwer, 2003. In Fred Glover and Gary Kochenberger, editors, Handbook of Meta-heuristics.

[30] E. K. Burke, M. Gendreau, M. R. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu. Hyper-heuristics: a survey of the state of the art. *JORS*, 64(12):1695–1724, 2013.

[31] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. Woodward. Exploring hyper-heuristic methodologies with genetic programming. In *Computational Intelligence*, pages 177–201. Springer, 2009.

[32] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward. *A Classification of Hyper-heuristic Approaches*, volume 146, pages 449–468. Springer, 2010. In Michel Gendreau, Jean-Yves Potvin, editors, Handbook of Meta-heuristics.

[33] E. K. Burke, G. Kendall, M. Misir, and E. Özcan. Monte carlo hyper-heuristics for examination timetabling. *Annals OR*, 196(1):73–90, 2012.

[34] J. Cámara, J. Cuenca, L.-P. García, and D. Giménez. Auto-tuned nested parallelism: a way to reduce the execution time of scientific software in NUMA systems. *Admitido en Parallel Computing*, 2014.

[35] J. Cámara, J. Cuenca, D. Giménez, and A. M. Vidal. Empirical autotuning of two-level parallel linear algebra routines on large cc-numa systems. *Admitido en Journal of Parallel Programming*, 2014.

[36] E. Cantú-Paz. A survey of parallel genetic algorithms. *Calculateurs Parallèles, Réseaux et Systèmes répartis*, 10(2):141–170, 1998.

[37] K. Chakhlevitch and P. Cowling. Hyperheuristics: Recent developments. In Carlos Cotta, Marc Sevaux, and Kenneth Sörensen, editors, *Adaptive and Multilevel Metaheuristics*, volume 136 of *Studies in Computational Intelligence*, pages 3–29. Springer, 2008.

[38] R. Chandra, R. Menon, L. Dagum, D. Kohr, D. Maydan, and J. McDonald. *Parallel Programming in OpenMP*. Morgan Kauffman, 2001.

[39] R. Chandra, R. Menon, L. Dagum, D. Kohr, D. Maydan, and J. McDonald. *OpenMP C and C++ Application Program Interface*. OpenMP Architecture Review Board. `http://www.openmp.org/drupal/mp-documents/cspec20.pdf`, 2002.

[40] T. G. Crainic. *Parallel Computation, Co-operation, Tabu Search*, pages 283–302. Kluwer Academic Publishers, Norwell, MA, 2005. In C. Rego and B. Alidaee, editors, Metaheuristic Optimization Via Memory and Evolution: Tabu Search and Scatter Search.

[41] T. G. Crainic, M. Gendreau, and J. Y. Potvin. Parallel tabu search. *Parallel Metaheuristics, E. Alba (ed.)*, 2005.

[42] T. G. Crainic and M. Toulouse. *Parallel Metaheuristics*, pages 205–251. Kluwer Academic Publishers, Norwell, MA, 1998. In T. G. Crainic and G. Laporte, editors, Fleet Management and Logistics.

[43] T. G. Crainic and M. Toulouse. Introduction to the Special Issue on Parallel Meta-Heuristics. *Journal of Heuristics*, 8(3):247–249, 2002.

[44] T. G. Crainic and M. Toulouse. *Parallel Strategies for Meta-heuristics*, pages 475–513. Kluwer Academic Publishers, Norwell, MA, 2003. In F. Glover and G. Kochenberger, editors, Handbook of Meta-heuristics.

[45] T. G. Crainic and M. Toulouse. Parallel Meta-Heuristics. Technical Report 2009-22, Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), 2009.

[46] T. G. Crainic, M. Toulouse, and M. Gendreau. Towards a taxonomy of parallel tabu search algorithms. *INFORMS Journal on Computing*, 9(1):61–72, 1997.

[47] J. Cuenca, L.-P. García, and D. Giménez. Improving linear algebra computation on NUMA platforms through auto-tuned nested parallelism. In *Proceedings of the 2012 EUROMICRO Conference on Parallel, Distributed and Network Processing*, 2012.

[48] J. Cuenca, D. Giménez, and J. González. Architecture of an automatic tuned linear algebra library. *Parallel Computing*, 30(2):187–220, 2004.

[49] J. Cuenca, D. Giménez, J. González, J. Dongarra, and K. Roche. Automatic optimisation of parallel linear algebra routines in systems with variable load. In *Proceedings of the 11th EUROMICRO Workshop on Parallel, Distributed and Networked Processing (PDP 2003)*, pages 409–416, 2003.

[50] V.-D. Cung, S. L. Martins, C. C. Ribeiro, and C. Roucairol. *Strategies for the Parallel Implementations of Metaheuristics*, pages 263–308. Kluwer Academic Publishers, Norwell, MA, 2002. In C. C. Ribeiro and P. Hansen, editors, Essays and Surveys in Metaheuristics.

[51] J.-M. Cutillas-Lozano, F. Almeida, and D. Giménez. Hyperheuristics based on parameterized metaheuristic schemes. In *Admitido como póster en GECCO'14*, 2014.

[52] J.-M. Cutillas-Lozano, F. Almeida, and D. Giménez. Hyperheuristics based on parameterized metaheuristic schemes. In *Proceedings of the 5th International Conference on Metaheuristics and Nature Inspired Computing (META'2014)*, 2014.

[53] J.-M. Cutillas-Lozano, L.-G. Cutillas-Lozano, and D. Giménez. Resolución de un problema de optimización de consumo eléctrico en explotación de pozos por medio de metaheurísticas parametrizadas. In *MAEB*, pages 391–398, 2012.

[54] J.-M. Cutillas-Lozano and D. Giménez. Determination of the kinetic constants of a chemical reaction in heterogeneous phase using parameterized metaheuristics. In *Proceedings of the 2013 International Conference on Computational Science (ICCS'13)*, pages 787–796, 2013.

[55] J.-M. Cutillas-Lozano and D. Giménez. Use of parallel, parameterized metaheuristics for the determination of the kinetic constants of a chemical reaction in heterogeneous phase. In *Simposio Doctoral de la red Infinite Salud, Murcia 2013.*, 2013.

[56] J.-M. Cutillas-Lozano and D. Giménez. Optimizing shared-memory hyperheuristics on top of parameterized metaheuristics. In *Proceedings of the 2014 International Conference on Computational Science (ICCS'14)*, 2014.

[57] J.-M. Cutillas-Lozano and D. Giménez. Parameterized message-passing metaheuristic schemes. In *Admitido como póster en GECCO'14*, 2014.

[58] J.-M. Cutillas-Lozano and D. Giménez. Using hyperheuristics to improve the determination of the kinetic constants of a chemical reaction in heterogeneous phase. In *Proceedings of the 2014 International Conference on Computational Science (ICCS'14)*, 2014.

[59] J.-M. Cutillas-Lozano, D. Giménez, and L.-G. Cutillas-Lozano. Modelling parameterized shared-memory hyperheuristics for auto-tuning. In *Proceedings of the 12th International Conference Computational and mathematical methods in Science and Engineering (CMMSE'12)*, pages 389–400, 2012.

[60] L.-G. Cutillas-Lozano. *Metaheurística aplicada a la optimización de los criterios de producción de aguas subterráneas. Proyecto Sondea*, 2008. Proyecto Final de Carrera, Universidad de Alicante.

[61] L.-G. Cutillas-Lozano, J.-M. Cutillas-Lozano, and D. Giménez. Modeling shared-memory metaheuristic schemes for electricity consumption. In *Proceedings of the 9th International Symposium on Distributed Computing and Artificial Intelligence (DCAI'12)*, pages 33–40, 2012.

[62] L. M. de Assumpcao Drummond, L. S. Vianna, M. B. da Silva, and L. S. Ochi. Distributed parallel metaheuristics based on GRASP and VNS for solving the traveling purchaser problem. In *Proceedings of the Ninth International Conference on Parallel and Distributed Systems*, pages 257–263, 2002.

[63] S. S. Dhumal and R. K. Saha. Application of genetic algorithm for evaluation of kinetic parameters of coal pyrolysis. *Journal of Energy & Environment*, 5:112, 2006.

[64] J. Dréo, A. Pétrowski, P. Siarry, and E. Taillard. *Metaheuristics for Hard Optimization*. Springer, 2005.

[65] I. Dumitrescu and T. Stuetzle. *Combinations of local search and exact algorithms*, volume 2611, pages 211–223. Springer, 2003. Lecture Notes in Computer Science.

[66] J. J. Durillo, A. J. Nebro, F. Luna, and E. Alba. A study of master-slave approaches to parallelize NSGA-II. In *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing (IPDPS 2008)*, pages 1–8, 2008.

[67] L. Elliott, D. B. Ingham, A. G. Kyne, N. S. Mera, M. Pourkashanian, and C. W. Wilson. Genetic algorithms for optimisation of chemical kinetics reaction mechanisms. *Progress in Energy and Combustion Science*, 30(3):297–328, 2004.

[68] T. Feo and M. Resende. Greedy randomized adaptative search procedure. *Journal of Global Optimization*, 42:860–878, 1995.

[69] M. J. Flynn. Some computer organizations and their effectivness. *IEEE Transactions on Computers*, 21:948–960, 1972.

[70] M. Frigo. FFTW: An Adaptive Software Architecture for the FFT. In *Proceedings of the ICASSP Conference*, volume 3, page 1381, 1998.

[71] F. Glover and G. A. Kochenberger. *Handbook of Metaheuristics*. Kluwer, 2003.

[72] F. Glover and M. Laguna. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[73] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic, 1997.

[74] F. Glover and R. Martí. *Scatter Search and Path Relinking: Advances and applications*. Kluwer Academic, 2003. In Handbook of Metaheuristics.

[75] F. Glover and B. Melián-Batista. Búsqueda tabú. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 7(19):29–48, 2003.

[76] J.-P. Goux, S. Kulkarni, J. Linderoth, and M. Yoder. An enabling framework for master-worker applications on the computational grid. In *Proceedings of the Ninth International Symposium on High-Performance Distributed Computing*, pages 43–50, 2000.

[77] J. H. Holland. Genetic algorithm and the optimal allocation of trials. *SIAM Journal on Computing*, 2:88–105, 1973.

[78] K. Holmqvist, A. Migdalas, and P. M. Pardalos. *Parallelized Heuristics for Combinatorial Search*, pages 269–294. Kluwer Academic Publishers, Norwell, MA, 1997. In A. Migdalas, P. M. Pardalos, and S. Storoy, editors, Parallel Computing in Optimization.

[79] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *LION*, pages 507–523, 2011.

[80] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle. ParamILS: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36:267–306, October 2009.

[81] L. Jourdan, M. Basseur, and E.-G. Talbi. Hybridizing exact methods and metaheuristics: A taxonomy. *European Journal of Operational Research*, 199(3):620–629, 2009.

[82] S. Kadioglu, Y. Malitsky, M. Sellmann, and K. Tierney. ISAC - instance-specific algorithm configuration. In *ECAI*, pages 751–756, 2010.

[83] T. Katagiri, K. Kise, and H. Honda. Effect of auto-tuning with user's knowledge for numerical software. In J. L. Gaudiot S. Vassiliadis and V. Piuri, editors, *Proceedings of the First Conference on Computing Frontiers*, pages 12–25, 2004.

[84] T. Katagiri, K. Kise, H. Honda, and T. Yuba. ABCLib_DRSSED: A parallel eigensolver with an auto-tuning facility. *Parallel Computing*, 32(3):231–250, 2006.

[85] A. Kheiri and E. Özcan. A hyper-heuristic with a round robin neighbourhood selection. In *EvoCOP*, pages 1–12, 2013.

[86] B. Kiraz, S. Uyar, and E. Özcan. An investigation of selection hyper-heuristics in dynamic environments. In *EvoApplications (1)*, pages 314–323, 2011.

[87] M. Laguna and R. Martí. Scatter search: Diseño básico y estrategias avanzadas. *Revista Iberoamericana de Inteligencia Artificial*, 19:123–130, 2003.

[88] M. Laguna and R. Martí. *Scatter Search. Methodology and Implementations in C.* Kluwer Academic Publisher, 2003.

[89] A. L. Lastovetsky and V. Rychkov. Accurate and efficient estimation of parameters of heterogeneous communication performance models. *IJHPCA*, 23(2):123–139, 2009.

[90] P. S. Laursen. *Parallel Heuristic Search - Introductions and a New Approach*, pages 248–274. Springer-Verlag, Berlin, 1996. In A. Ferreira and P. M. Pardalos, editors, Solving Combinatorial Optimization Problems in Parallel, volume 1054 of Lecture Notes in Computer Science.

[91] Y. Li, Y. Zhang, Y.-Q. Liu, G. Long, and H. Jia. MPFFT: An Auto-Tuning FFT Library for OpenCL GPUs. *J. Comput. Sci. Technol.*, 28(1):90–105, 2013.

[92] D. R. Lide. *Handbook of Chemistry and Physics.* CRC Press, 85th edition, 2004.

[93] A. Liefooghe, L. Jourdan, T. Legrand, J. Humeau, and E.-G. Talbi. ParadisEO-MOEO: A Software Framework for Evolutionary Multi-Objective Optimization. In *Advances in Multi-Objective Nature Inspired Computing*, pages 87–117. 2010.

[94] S.-C. Lin, W. Punch, and E. Goodman. Coarse-grain parallel genetic algorithms: Categorization and new approach. In *Sixth IEEE Symposium on Parallel and Distributed Processing*, pages 28–37. IEEE Computer Society Press, 1994.

[95] T. Van Luong, N. Melab, and E.-G. Talbi. GPU-based island model for evolutionary algorithms. In *GECCO*, pages 1089–1096, 2010.

[96] G. Luque and E. Alba. Parallel genetic algorithms. *Parallel Metaheuristics, E. Alba (ed.)*, 2005.

[97] G. Luque and E. Alba. *Parallel Genetic Algorithms: Theory and Real World Applications*. Springer, 2011.

[98] F. Mascia, M. López-Ibáñez, J. Dubois-Lacoste, and T. Stützle. From grammars to parameters: Automatic iterated greedy design for the permutation flow-shop problem with weighted tardiness. In *LION*, pages 321–334, 2013.

[99] F. Mascia, M. López-Ibáñez, J. Dubois-Lacoste, and T. Stützle. Grammar-based generation of stochastic local search heuristics through automatic algorithm configuration tools. Technical Report 2013-015, IRIDIA, Université Libre de Bruxelles, 2013.

[100] N. Melab, M.-S. Mezmaz, and E.-G. Talbi. Parallel hybrid multi-objective island model in peer-to-peer environment. In *IPDPS*, 2005.

[101] M.-S. Mezmaz, Y. Kessaci, Y. C. Lee, N. Melab, E.-G. Talbi, A. Y. Zomaya, and D. Tuyttens. A parallel island-based hybrid genetic algorithm for precedence-constrained applications to minimize energy consumption and makespan. In *GRID*, pages 274–281, 2010.

[102] M.-S. Mezmaz, N. Melab, and E.-G. Talbi. Using the multi-start and island models for parallel multi-objective optimization on the computational grid. In *e-Science*, page 112, 2006.

[103] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer Verlag, 1999.

[104] H. Mühlenbein. *Parallel Genetic Algorithms in Combinatorial Optimization*, pages 441–456. Pergamon Press, New York, NY, 1992. In O. Balci, R. Sharda, and S. Zenios, editors, Computer Science and Operations Research: New Developments in their Interface.

[105] Murcia Supercomputing Center. `www.parquecientificomurcia.es`.

[106] G. L. Nicholson and M. J. Lancaster. Coupling matrix synthesis of cross-coupled microwave filters using a hybrid optimisation algorithm. *IET Microwaves, Antennas and Propagation*, 3(6):950–958, 2009.

[107] G. Ochoa and E. Özcan. Special issue on hyper-heuristics in search and optimization. *J. Heuristics*, 16(6):745–748, 2010.

[108] OpenMPI: Open Source High Performance Computing. `http://www.open-mpi.org`.

[109] J. C. Ortiz-Bayliss, H. Terashima-Marín, E. Özcan, A. J. Parkes, and S. E. Conant-Pablos. Exploring heuristic interactions in constraint satisfaction problems: A closer look at the hyper-heuristic space. In *IEEE Congress on Evolutionary Computation*, pages 3307–3314, 2013.

[110] D. Ouelhadj and S. Petrovic. A cooperative hyper-heuristic search framework. *J. Heuristics*, 16(6):835–857, 2010.

[111] E. Özcan, B. Bilgin, and E. Korkmaz. A comprehensive analysis of hyper-heuristics. *Intelligent Data Analysis*, 12(1):3–23, 2008.

[112] E. Özcan and E. K. Burke. Multilevel search for choosing hyper-heuristics. In *4th Multidisciplinary Int. Conf. on Scheduling: Theory and Applications*, pages 788–789, 2009.

[113] Parallel Computing Group of the University of Murcia. `http://luna.inf.um.es/grupo_investigacion`.

[114] P. M. Pardalos, L. Pitsoulis, T. Mavridou, and M. G. C. Resende. Parallel search for combinatorial optimization: Genetic algorithms, simulated annealing, tabu search and GRASP. In *Proceedings of Workshop on Parallel Algorithms for Irregularly Structured Problems, Lecture Notes in Computer Science, volume 980*, pages 317–331. Springer-Verlag, Berlin, 1995.

[115] C. B. Pettey, M. R. Leuze, and J. J. Grefenstette. A Parallel Genetic Algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms*, pages 155–161, 1987.

[116] L. N. Plummer, T. M. L. Wigley, and D. L. Parkhurst. The kinetics of calcite dissolution in CO_2–water systems at 5 to 60^{o}C and 0.0 to 1.0 atm CO_2. *American Journal of Science*, 278:179–216, 1978.

[117] M. J. Quinn. *Parallel Programming in C with MPI and OpenMP*. McGraw Hill, 2004.

[118] G. R. Raidl. A unified view on hybrid metaheuristics. In *Hybrid Metaheuristics, Third International Workshop, LNCS*, volume 4030, pages 1–12, October 2006.

[119] C. R. Reeves. *Modern Heuristic Techniques for Combinatorial Problems*. McGraw-Hill, UK, 1995.

[120] A. Renato, V. da Silva, and L. Satoru Ochi. A hybrid evolutionary algorithm for the dynamic resource task scheduling problem. In *Proc. of the 10th International Workshop on Nature Inspired Distributed Computing (NIDISC' 07) held in conjunction with The 21th IEEE/ACM Int. Parallel and Distributed Processing Symposium (IPDPS 2007)*, March 2007.

[121] M. G. C. Resende and J. L. González. GRASP: Procedimientos de búsqueda miopes aleatorizados y adaptativos. *Revista Iberoamericana de Inteligencia Artificial*, 19:61–762, 2003.

[122] M. G. C. Resende and C. C. Ribeiro. *Greedy Randomized Adpative Search Procedures*. Kluwer Academic, 2003. In Handbook of Metaheuristics.

[123] D. Rickard and E. L. Sjöberg. Mixed kinetic control of calcite dissolution rates. *American Journal of Science*, 283:815–830, 1983.

[124] Casiano Rodríguez León. *The Design, Analysis and Implementation of Algorithms for Parallel Shared Memory Machines*. http://nereida.deioc.ull.es/ pp1/openmp/openmpbook.ps, 2002.

[125] S. H. Roosta. *Parallel Processing and Parallel Algorithms*. Springer, 2000.

[126] B. A. Shirazi, A. R. Hurson, and K. M. Kavi. *Scheduling and Load Balancing in Parallel and Distributed Systems*. IEEE Computer Society Press, 1995.

[127] R. Shonkwiler. Parallel genetic algorithms. In *Fifth International Conference on Genetic Algorithms*, pages 199–205. Morgan Kaufmann, San Mateo, CA, 1993.

[128] E. A. Silver, R. V. Vidal, and D. De Werra. A tutorial on heuristic methods. *European Journal of Operational Research*, 5:153–162, 1980.

[129] M. Snir and W. Gropp. *MPI. The Complete Reference. 2nd edition*. The MIT Press, 1998.

[130] É. D. Taillard, L. M. Gambardella, M. Gendreau, and J. Potvin. Adaptive memory programming: A unified view of metaheuristics. *European Journal of Operational Research*, 135(1):1–16, 2001.

[131] E.-G. Talbi. *Metaheuristics - From Design to Implementation*. Wiley, 2009.

[132] E.-G. Talbi and G. Hasle. Metaheuristics on GPUs. *J. Parallel Distrib. Comput.*, 73(1):1–3, 2013.

[133] The OpenMP API Specification for Parallel Programming. `http://openmp.org`.

[134] R. Vaessens, E. Aarts, and J. Lenstra. A local search template. In R. Manner and B. Manderick, editors, *Parallel Problem Solving from Nature*, pages 67–76. Elsevier, 1992.

[135] L. G. Valiant. A bridging model for parallel computation. *Comunications of the ACM*, 33:103–111, 1990.

[136] M. G. A. Verhoeven and E. H. L. Aarts. Parallel local search. *Journal of Heuristics*, 1(1):43–65, 1995.

[137] S. Voss. *Tabu Search: Applications and Prospects*, pages 333–353. World Scientific Publishing Co., Singapore, 1993. D.-Z. Du and P. M. Pardalos, editors, Network Optimization Problems.

[138] R. C. Whaley, A. Petitet, and J. Dongarra. Automated empirical optimizations of software and the ATLAS project. *Parallel Computing*, 27(1-2):3–35, 2001.

[139] B. Wilkinson and M. Allen. *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers*. Prentice-Hall, second edition, 2005.