MÁSTER EN MATEMÁTICAS AVANZADAS

Algoritmo de Búsqueda Armónica: Fundamentos y resolución de problemas de optimización con restricciones

TRABAJO FIN DE MÁSTER

MADRID
Septiembre de 2019

ESPECIALIDAD: ESTADÍSTICA E INVESTIGACIÓN OPERATIVA

Alumno: Juan Alberto Molina García

Director: Prof. Dr. Eduardo Ramos Méndez

ÍNDICE

AGRADECIMIENTOS

A mis familiares y amigos, de quienes siempre he obtenido comprensión y ayuda para alcanzar las metas que me propuse.

A mi tutor, Dr. D. Eduardo Ramos Méndez, por el trato humano, el asesoramiento académico que obtuve de él mientras me esforzaba en completar esta tarea y su apasionado entusiasmo por la investigación científica.

INTRODUCCIÓN

¿Existe alguna conexión entre la improvisación musical y la búsqueda de soluciones óptimas a problemas teórico-prácticos, científico-técnicos, socio-económicos y artísticos? La respuesta es afirmativa. Los recientes desarrollos del algoritmo de búsqueda armónica (en adelante, **HSA**), junto a sus posteriores versiones mejoradas, muestran los progresos de las investigaciones al respecto. Además, las aplicaciones prácticas de **HSA** son múltiples, pues, desde que viera la luz hacia el año 2000, el algoritmo se ha empleado para resolver una gran cantidad de problemas de optimización en varias áreas del saber; y también se ha comparado experimentalmente su eficiencia con la de otros algoritmos metaheurísticos. En consecuencia, ya existe una considerable bibliografía sobre los asuntos anteriormente mencionados, además de una página web (https://scholar.google.es/) con numerosas entradas para el término "Harmony Search Algorithm".

Muy escasos, sin embargo, son los trabajos orientados hacia el análisis matemático de los procesos de búsqueda de **HSA**, aun cuando podrían proporcionar a los estudiosos esclarecedoras pistas sobre la selección de los parámetros y las fortalezas y debilidades de este algoritmo.

HSA se basa en la hipótesis de que el objetivo principal de la improvisación musical es encontrar un excelente estado de armonía. Es decir, la improvisación musical aspira a lograr una rica variedad de combinaciones de sonidos que, según ciertas convenciones estéticas del período histórico en curso, aportan calidad y perfección a la pieza que se está interpretando. La improvisación musical, facultad que no todo el mundo posee por naturaleza y que es de uso común entre los músicos de Jazz, se puede ejercitar y mejorar gracias al conocimiento acumulado y a la experiencia de los intérpretes.

Las anteriores consideraciones, en definitiva, sugieren que existe cierto grado de equivalencia o similitud entre los procesos de optimización de funciones y la improvisación musical.

Bajo estos supuestos, los objetivos de este trabajo han sido:

1) Revisar el estado de conocimiento actual y el marco teórico de **HSA**.
2) Poner de relieve la equivalencia arriba aludida.
3) Explicar las principales hibridaciones y las nuevas configuraciones de los parámetros del algoritmo.

4) Aplicar **HSA** a la resolución de problemas de optimización no lineal con restricciones.
5) Enunciar las aplicaciones más recurrentes de **HSA** y sus hibridaciones en algunas ramas de la ciencia y la tecnología, poniendo de relieve su alto grado de eficacia y de versatilidad en comparación con otros algoritmos metaheurísticos.

Para ello, se ha efectuado una consulta pormenorizada y comparada de la bibliografía actualmente existente, así como de algunas obras ya clásicas sobre problemas de optimización no lineal con restricciones.

1. ESTADO DE CONOCIMIENTO Y MARCO TEÓRICO

1.1. Descripción y metáfora del algoritmo

Los creadores de **HSA** recurrieron al símil de la improvisación en música de Jazz para explicar el funcionamiento del algoritmo. Según ellos, caben tres acciones al respecto: (a) tocar de memoria alguna pieza aprendida anteriormente; (b) tocar algo parecido a esa pieza conocida y ajustarla poco a poco al tono deseado mientras se mantienen fijos los intervalos musicales (es decir, transportar una pieza de una tonalidad a otra); (c) componer algo nuevo basándose en los conocimientos previos y/o seleccionar notas nuevas de forma aleatoria. Estas tres opciones corresponden, literalmente, a las tres componentes principales de **HSA**: uso de la memoria armónica, ajuste tonal y aleatorización (Geem, Kim y Loganathan, 2001; Yang, 2010).

De modo que, al improvisar, el músico produce sonidos dentro de cierto rango y forma melodías con ellos; o acordes, que no son sino superposiciones simultáneas de dos o más notas que guardan entre sí ciertas relaciones armónicas obedeciendo a patrones estéticos establecidos. Si estas melodías y armonías forman buenas combinaciones, el intérprete las guarda en su memoria y así ve incrementada la posibilidad de hacer otra buena combinación la vez siguiente. De igual manera, al optimizar una función, cada variable de decisión toma valores aleatorios dentro de cierto rango, formando un vector solución. Éste se almacena en la memoria armónica de cada variable, con lo que aumenta la posibilidad de encontrar soluciones mejores en la subsecuente iteración.

Improvisación musical	Optimización de una función matemática
Búsqueda de melodías y armonías: armonía ideal	Búsqueda de óptimos locales y globales: óptimo global
Patrones estéticos aceptados	Función objetivo que se quiere optimizar
El propio músico que improvisa una pieza	Variable de decisión del problema de optimización
Melodías y acordes improvisados	Valores aleatorios que toma la variable de decisión
Rango tonal del instrumento musical	Rango o dominio de los valores de la variable de decisión
Combinación de melodías y/o acordes en la pieza improvisada	Vector de decisión
Experiencia del músico	Memoria armónica
Combinación musical elegida por el intérprete	Valor seleccionado de la variable de decisión
Práctica musical	Iteraciones en busca de las mejores soluciones

Tabla 1.2: Equivalencias entre improvisación musical y optimización de funciones matemáticas.

En definitiva, se pueden establecer equivalencias (tabla 1.2) entre diversos aspectos de la improvisación musical y la optimización de funciones matemáticas. Así, cabe concluir que la más bella combinación melódica y/o armónica corresponde al vector que contiene la solución óptima. Con las siguientes ventajas, por añadidura: **HSA** no precisa de cálculos matemáticos complejos; los resultados no dependen de valores iniciales de las variables de decisión; el algoritmo tiende a obviar los óptimos locales, puede manejar tanto variables discretas como continuas y se procesa y converge muy rápidamente.

La figura 1.1 ilustra con más detalle la analogía entre la improvisación musical y la optimización de funciones matemáticas. Los instrumentos de la parte superior (saxofón, contrabajo y guitarra) están relacionados con las variables de decisión de la parte inferior (X_1, X_2 *y* X_3, respectivamente). Además, el rango de cada instrumento (saxofón = {Do, Re, Mi}; contrabajo = {Mi, Fa, Sol}; guitarra = {Sol, La, Si}) está vinculado al rango de cada variable de decisión ($X_1 = \{1, 2, 3\}$; $X_2 = \{3, 4, 5\}$; $X_3 = \{5, 6, 7\}$). Las variables van acompañadas de una unidad de medida (m), seleccionada originalmente para la aplicación a un caso práctico y sin que por ello se vean afectadas ni la generalidad del planteamiento ni a la explicación del problema.

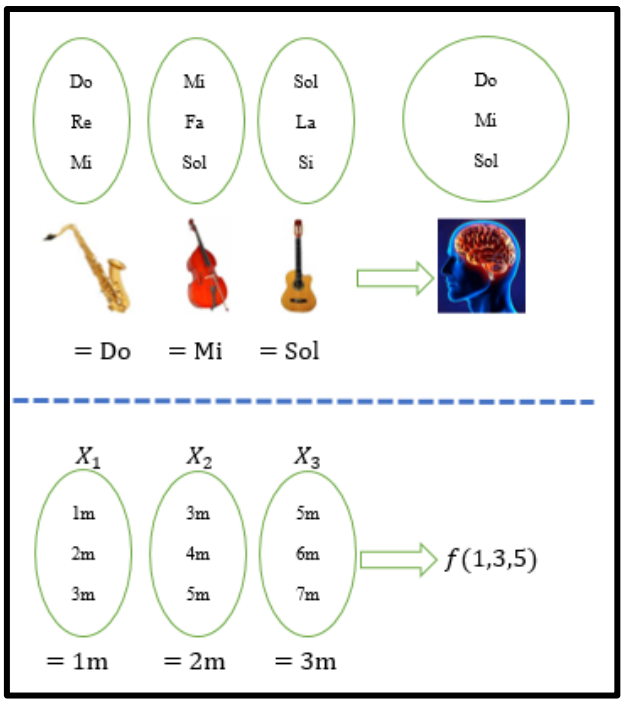

Figura 1.1: Analogía entre improvisación musical y optimización.

Si el saxofonista toca la nota "Do", el contrabajista la nota "Mi" y el guitarrista la nota "Sol", entre los tres forman el nuevo acorde "Do-Mi-Sol"; éste, si se considera bueno desde el punto de vista estético, se guarda en la memoria musical de los intérpretes. De forma análoga, el nuevo vector solución $(1m, 3m, 5m)$ generado en el proceso de optimización se añade a la memoria computacional si se considera bueno en términos del valor obtenido por la función objetivo. Y del mismo modo que la calidad de la armonía mejora con la práctica de los músicos, la calidad de la solución del problema de optimización aumenta con cada iteración.

Expresado en lenguaje matemático formal, **HSA** se ideó para resolver problemas del tipo:

$$\text{Optimizar } f(x) \qquad [1.3]$$

Sujeto a:

$$h_i(x) = 0; \qquad i = 1, 2, \dots, p \qquad [1.4]$$

$$g_i(x) \geq 0; \qquad i = 1, 2, \dots, q \qquad [1.5]$$

$$x_i \in \boldsymbol{X}_i = \{x_i(1), x_i(2), \dots, x_i(k), \dots, x_i(K_i)\} \qquad [1.6]$$

$$x_i^L \leq x_i \leq x_i^U \qquad [1.7]$$

HSA busca en el espacio de soluciones para encontrar el vector solución $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$ que optimiza la función objetivo $f(x)$ ([1.3]). Si las restricciones del problema de optimización son de igualdad, se considerará la ecuación [1.4]; si son de desigualdad, se recurrirá a [1.5]; si las variables de decisión tienen valores discretos, estos serán $x_i \in \boldsymbol{X}_i = \{x_i(1), x_i(2), \dots, x_i(k), \dots, x_i(K_i)\}$ ([1.6]); si son continuos, entonces cumplirán $x_i^L \leq x_i \leq x_i^U$ ([2.7]). En el caso de que el vector solución generado viole alguna de las restricciones [1.4] y [1.5], entonces es rechazado por el algoritmo; de otro modo, pasa a formar parte de la memoria armónica.

Con objeto de resolver el problema planteado en las expresiones [1.3] a [1.7], **HSA** mantiene una solución ($\boldsymbol{x}^k$) o una población $\boldsymbol{P}^k(\boldsymbol{x_1^k}, \boldsymbol{x_2^k}, \dots, \boldsymbol{x_n^k})$ de n soluciones candidatas que se modifican durante un número máximo ($\boldsymbol{NI}$) de iteraciones (improvisaciones) desde un estado inicial hasta otro final. En el estado inicial, **HSA** empieza con un vector aleatorio dentro de los límites del espacio $\boldsymbol{X}$ de soluciones. Para construir una nueva población $\boldsymbol{P}^{k+1}$, a cada improvisación posterior se aplica una serie de operadores a la población $\boldsymbol{P}^k$. Después, se evalúa la solución $\boldsymbol{x_i^k}$ $(i = 1, 2, \dots, n)$ a través de la función objetivo. Al final del proceso, la mejor solución de todas las evaluadas es la que se toma como óptimo global.

El éxito y la aceptación generalizada de **HSA** entre la comunidad científica se debe, entre otras, a estas razones:

- Para generar nuevas soluciones se consideran todas las ya existentes, y no sólo dos como ocurre con los algoritmos genéticos.
- Cada variable de decisión del vector de soluciones es considerada de forma independiente.
- Los valores de las variables de decisión son continuos, lo que refuerza su precisión.
- No hace falta hacer conversiones a otros sistemas de numeración.
- No es necesario inicializar previamente los puntos primigenios de las soluciones candidatas.
- No se utilizan procedimientos basados en el gradiente y la derivada.
- Su estructura es sencilla.

1.2. Parámetros

El uso de la memoria armónica se parece a la selección de los individuos mejor adaptados en los algoritmos genéticos, en el sentido de que la mejor armonía pasará a formar parte de la nueva memoria armónica. Su eficacia depende de cierta tasa de aceptación de la memoria armónica, $\boldsymbol{r} \in [0, 1]$ (también denotada por **HMCR**), que se escoge al azar de la memoria armónica. Si esta tasa es demasiado baja, sólo se seleccionarán unas cuantas armonías y el algoritmo convergerá con demasiada lentitud. En cambio, si la tasa es alta (≈ 1), casi todas las armonías pasarán a formar parte de la memoria armónica, dando lugar a soluciones potencialmente erróneas. Lo más usual es tomar $\boldsymbol{r} = 0{,}7 \sim 0{,}95$.

El segundo miembro de **HSA** es el ajuste tonal, determinado por un ancho de banda **BW**, la tasa $\boldsymbol{PAR}$ de ajuste del tono y el número $\boldsymbol{NI}$ de improvisaciones. El ancho de banda se refiere a la separación entre los puntos que dividen arbitrariamente el rango total de los valores de decisión; tiene su paralelismo en los trastes que dividen la longitud del mástil de la guitarra, correspondiendo cada uno de ellos a un intervalo de medio tono; se puede interpretar (Mukhopadhyay et al., 2008) como la desviación típica de los valores $x_i \in \boldsymbol{X}_i$ cuando **HMCR** ≈ 1:

$$\boldsymbol{BW}(I) = \sigma(x_i) = \sqrt{var(x_i)} \qquad [1.8]$$

Aunque en música el ajuste tonal conlleve un cambio de frecuencias, en **HSA** tan sólo supone una solución ligeramente diferente. Teóricamente, el tono se puede ajustar de forma lineal o no lineal, pero en la práctica se utiliza el ajuste lineal (Mahdavi, Fesanghary y Damangir, 2007):

$$\boldsymbol{X}_{nuevo} = \boldsymbol{X}_{viejo} \pm (2 \times \boldsymbol{r} - 1) \times \boldsymbol{BW} \qquad [1.9]$$

En la ecuación [1.9], $\boldsymbol{X}_{viejo}$ es el tono existente en la memoria armónica; el tono nuevo después del ajuste es $\boldsymbol{X}_{nuevo}$; y $\boldsymbol{r}$ ($= \boldsymbol{HMCR}$), como ya se vio anteriormente, es un generador de números aleatorios en el intervalo [0, 1]. Una tasa $\boldsymbol{PAR}$ baja y una anchura de banda $\boldsymbol{BW}$ estrecha pueden retrasar la convergencia de **HSA**, pues la exploración se limita a una pequeña región del espacio de búsqueda. Una tasa alta y un

amplio $\boldsymbol{BW}$ hacen que la solución se disperse alrededor de los óptimos potenciales. Generalmente, se utiliza $\boldsymbol{PAR} = 0{,}1 \sim 0{,}5$.

La tercera componente del algoritmo es la aleatorización, que proporciona un aumento de la diversidad de las soluciones. El ajuste del tono tiene un papel similar, pero está limitado a un ámbito local. En cambio, el uso de la aleatoriedad puede conducir al sistema a explorar varias soluciones y encontrar, finalmente, el óptimo global.

La probabilidad de asignación al azar y la de ajuste tonal son, respectivamente:

$$\boldsymbol{P}_{aleatorización} = 1 - \boldsymbol{r} = 1 - \boldsymbol{HMCR} \qquad [1.10]$$

$$\boldsymbol{P}_{tono} = \boldsymbol{r} \times \boldsymbol{PAR} \qquad [1.11]$$

En general, **HSA** es poco sensible a los parámetros (Geem, 2009-a; Yang, 2009; Yang, 2010), de modo que para obtener una buena solución no es necesario afinar éstos exhaustivamente. No obstante, como se ha visto, la tasa de consideración de la memoria armónica y el parámetro $\boldsymbol{PAR}$ contribuyen en gran medida al hallazgo de soluciones globales y locales, respectivamente.

Por otra parte, **HSA** genera nuevos vectores de soluciones tras considerar todos los que ya existen en la memoria armónica, mientras que los algoritmos genéticos sólo consideran los dos vectores "progenitores". Esta particularidad hace de **HSA** un algoritmo sumamente flexible, mediante el cual se encuentran mejores soluciones (de mejor calidad y obtenidas en menor tiempo) que con otros algoritmos de su misma especie.

Los valores asignados a los parámetros vistos anteriormente confieren fortaleza a **HSA**, dependen de la aplicación concreta de éste y son cruciales para conseguir un buen rendimiento del algoritmo y un adecuado equilibrio entre intensificación y diversificación (Yang, 2010). Particularmente, la consideración aleatoria del algoritmo permite que se generen soluciones nuevas y, por tanto, que se amplíe la posibilidad de explorar el espacio de búsqueda.

1.3. Funcionamiento

Los pasos que sigue el algoritmo son los siguientes (Geem, Kim y Loganathan, 2001; Geem, 2009-a; Yang, 2009; Yang, 2010):

A. Iniciar el problema y definir los parámetros.

B. Iniciar la memoria armónica.
C. Improvisar una nueva memoria armónica.
D. Actualizar la memoria armónica.
E. Comprobar el criterio de parada.

El problema de optimización (paso A y ecuaciones [1.3] a [1.7]) consiste en minimizar $f(\boldsymbol{x})$, sujeto a $x_i \in X_i$ $(i = 1, 2, \ldots, n)$, donde $f(\boldsymbol{x})$ es la función objetivo, $\boldsymbol{x}$ es el conjunto de variables de decisión x_i, n es el número de variables de decision y $\boldsymbol{X}_i$ es el conjunto de valores para cada variable de decision (esto es, $x_i^L \leq x_i \leq x_i^U$, siendo x_i^L y x_i^U, respectivamente, los límites inferior y superior de la variable de decisión x_i). En el paso A también se especifican los parámetros.

La matriz de memoria armónica (paso B) se rellena con tantos vectores solución de carácter aleatorio como sea el tamaño de la memoria armónica (**HMS**):

$$\mathbf{HM} = \begin{pmatrix} x_1^1 & \cdots & x_1^n \\ \vdots & \ddots & \vdots \\ x_n^{HMS} & \cdots & x_n^{HMS} \end{pmatrix} \qquad [1.12]$$

La improvisación de una nueva memoria armónica (paso C) ha de ajustarse a las tres normas básicas (consideración de la memoria armónica, ajuste tonal y selección aleatoria).

El valor de la primera variable de decisión x_1^j (j = 1, 2, …, **HMS**) se escoge aleatoriamente entre los valores pertenecientes al rango $(x_1^1, x_1^{\boldsymbol{HMS}})$, de acuerdo con la ecuación:

$$x_1^j = x_1^L + (x_1^U - x_1^L) \times \boldsymbol{r} \qquad [1.13]$$

Los valores de las demás variables de decisión se eligen de forma análoga, de modo que:

$$x_i^j \leftarrow \begin{cases} x_i^j \in \left[x_i^1, x_i^2, \ldots, x_i^{HMS}\right] \ con\ probabilidad\ \boldsymbol{r} \\ x_i^j \in X_i\ con\ probabilidad\ (1 - \boldsymbol{r}) \end{cases} \qquad [1.14]$$

Por ejemplo, si $\boldsymbol{r} = 0.90$, el algoritmo escogerá, entre los valores almacenados en la memoria armónica, el valor de una variable de decisión determinada con una probabilidad del 90%. Cada componente obtenido en la memoria armónica ha de ser examinado para determinar si se debe o no ajustar su tono. Es aquí donde entra en juego el parámetro ***PAR***, de la siguiente manera:

$$\text{Ajuste tonal para la variable } x_i^j \leftarrow \begin{cases} Si, con\ probabilidad\ \boldsymbol{PAR} \\ No, con\ probabilidad\ (1 - \boldsymbol{PAR}) \end{cases} \quad [1.15]$$

El término $1 - \boldsymbol{PAR}$, contenido en la expresión [1.15], se refiere a la tasa con que **HSA** guarda el valor original obtenido de la memoria armónica.

Si se decide ajustar el tono para x_i^j, se efectúa el reemplazo:

$$\boldsymbol{X}_{nuevo} = \boldsymbol{X}_{viejo} \pm (2 \times \boldsymbol{r} - 1) \times \boldsymbol{BW} \quad [1.16]$$

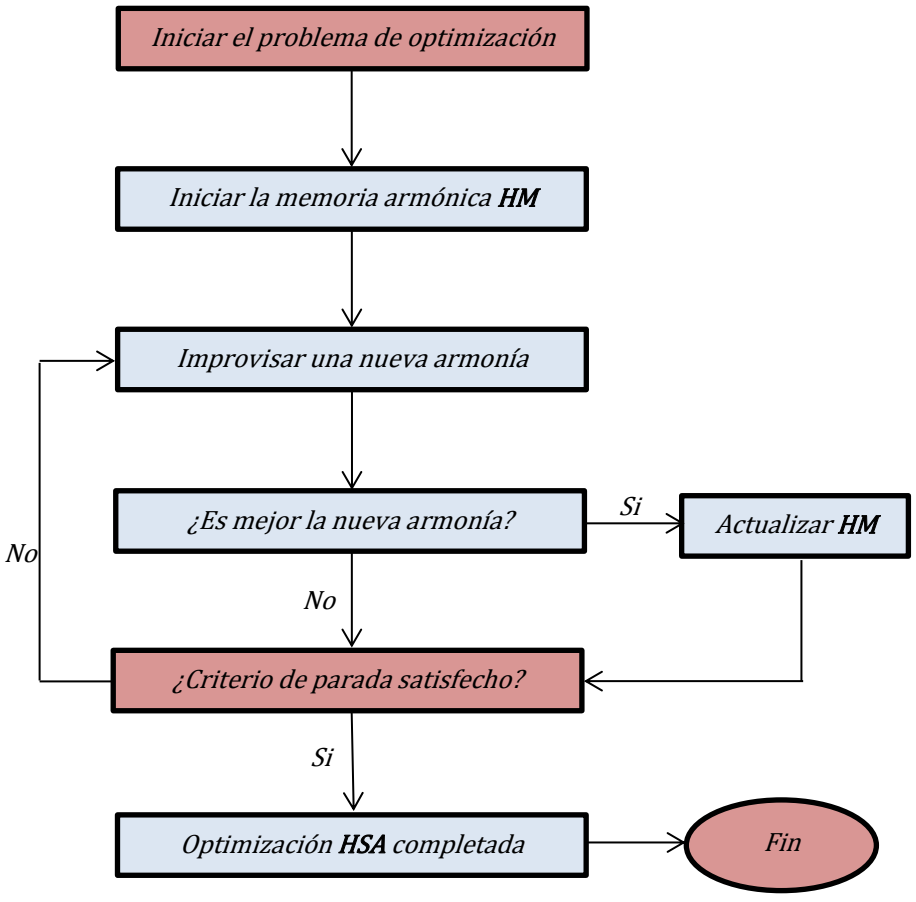

*Figura 1.2: Esquema de optimización de **HSA**.*

Según [1,16], siempre que $\boldsymbol{X}_{nuevo}$ sea mejor que $\boldsymbol{X}_{viejo}$, se añadirá dicho vector a la memoria armónica ($\boldsymbol{X}_{nuevo} \in \boldsymbol{HM}$) y se eliminará el tono antiguo ($\boldsymbol{X}_{viejo} \notin \boldsymbol{HM}$). El mismo proceso se aplica a cada variable de la nueva solución (paso D).

Si se cumple el criterio de parada (realización del máximo número de improvisaciones; hallazgo del vector solución que optimiza el valor de la función objetivo), el proceso ha llegado a su fin (paso E). En caso contrario, hay que volver a los pasos C y D. Véase en la figura 1.2 un esquema del proceso iterativo de **HSA**.

Obsérvese que, tras escoger $\boldsymbol{X}_{nuevo} \in \boldsymbol{HM}$, éste se puede ajustar aún más a otros valores próximos sin más que añadir cierto valor de ajuste tonal con probabilidad $\boldsymbol{PAR}$. Si las variables son discretas, el valor ajustado de $x_i^k = x_i^{nuevo}$ será x_i^{k+m}, con $m \in [-1, 1]$. Y para variables continuas, el nuevo tono ajustado valdrá $x_i^{nuevo} + \Delta_i$, donde $\Delta_i = U(0,1) \times \boldsymbol{BW}_i$.

En el sistema tonal de la música occidental, y en particular en el estilo jazzístico, las anteriores consideraciones tienen su correspondiente expresión en el lenguaje musical. En primer lugar, el músico escoge cada sonido del registro sonoro propio de su instrumento; por ejemplo, dos octavas y media (dos octavas más una quinta justa), como ocurre con el saxofón (figura 1.3).

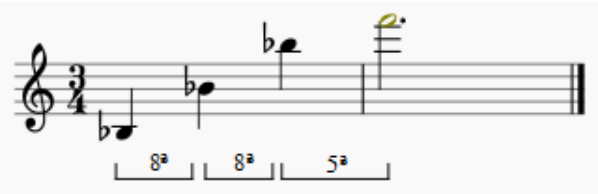

Figura 1.3: Registro sonoro del saxofón.

El ajuste tonal se realiza mejorando las notas anteriormente producidas, que forman determinado intervalo con la ideal; ésta pertenece al acorde (de tónica, dominante, subdominante, tríada, séptima, novena, etc., en estado fundamental o invertido) y tonalidad en curso; los intervalos más pequeños son los de semitono ascendente (#, sostenido) o descendente (*b*, bemol), como se puede ver en la figura 1.4.

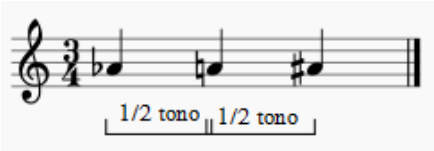

Figura 1.4: Acercamiento tonal por semitonos al La natural (becuadro) central.

La nota finalmente escogida es la que pertenece al acorde de la tonalidad considerada (figura 1.5).

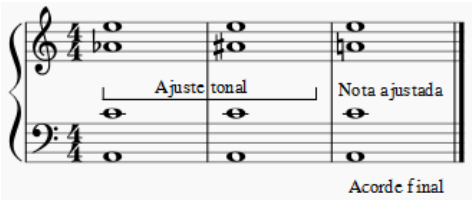

Figura 1.5: Nota la natural ajustada al acorde de tónica de la tonalidad La menor. La probabilidad de elección de cada una de las notas La bemol, La sostenido y La natural (becuadro) es $\frac{1}{3}$.

El incumplimiento de las restricciones iniciales por una solución del problema de optimización equivale a la violación de las reglas de formación o encadenamiento de acordes (figura 1.6).

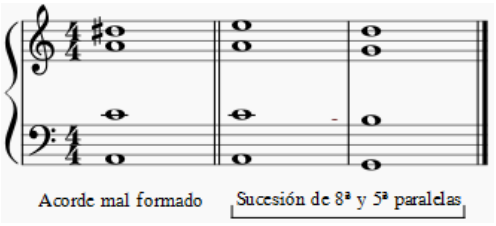

Figura 1.6: El Re # del primer compás no pertenece al acorde de La menor. Los acordes de los compases siguientes violan la prohibición de encadenarlos por 8^{as} *y* 5^{as} *consecutivas.*

Por último, las tres operaciones básicas de selección aleatoria, consideración de la memoria armónica y ajuste tonal se pueden expresar formalmente como sigue:

$$x_i^{nuevo} \leftarrow \begin{cases} (Probabilidad = 1 - \boldsymbol{HMCR}): \begin{cases} x_i \in \{x_i^1, \dots, x_i^k, \dots, x_i^{K_i}\} \\ x_i \in [x_i^L, x_i^U] \end{cases} \\ (Probabilidad = \boldsymbol{HMCR} \times (1 - \boldsymbol{PAR})): x_i^j \in HM = [x_i^1, x_i^2, \dots, x_i^{HMS}] \\ (Probabilidad = \boldsymbol{HMCR} \times \boldsymbol{PAR}): \begin{cases} x_i^{k+m}, si\ x_i^k \in \boldsymbol{HM} \\ x_i + \Delta_i, si\ x_i \in \boldsymbol{HM} \end{cases} \end{cases} \quad [1.17]$$

Para variables discretas, **HSA** tiene una expresión estocástica que no encuentra parangón en el método del gradiente (Geem y Sim, 2010-b):

$$\frac{\partial f}{\partial x_i} = \frac{1}{K_i} \times (1 - \boldsymbol{HMCR}) + \frac{n \times x_i^k}{\boldsymbol{HMS}} \times \boldsymbol{HMCR} \times (1 - \boldsymbol{PAR}) + \frac{n \times x_i^{k-m}}{\boldsymbol{HMS}} \times \boldsymbol{HMCR} \times \boldsymbol{PAR} \quad [1.18]$$

La ecuación anterior representa la probabilidad de elegir cierto valor x_i^k mediante la suma de otras tres probabilidades:

1. $P_{aleatorización} = \frac{1}{K_i} \times (1 - \boldsymbol{HMCR})$ es la probabilidad de elegir x_i^k por selección aleatoria.
2. $P_{\boldsymbol{HMCR}} = \frac{n \times x_i^k}{\boldsymbol{HMS}} \times \boldsymbol{HMCR} \times (1 - \boldsymbol{PAR})$ es la probabilidad de consideración de la memoria armónica.
3. $P_{tono} = \frac{n \times x_i^{k-m}}{\boldsymbol{HMS}} \times \boldsymbol{HMCR} \times \boldsymbol{PAR}$ es la probabilidad del ajuste tonal.

Aquí, la función de probabilidad acumulada para el conjunto completo de valores es igual a uno:

$$\sum_{k=1}^{K_i} \frac{\partial f}{\partial x_i}\Big|_{x_i = x_i^k} = 1 \quad [1.19]$$

Considérese a continuación un ejemplo aclaratorio sobre el funcionamiento de **HSA** y su paralelismo con la improvisación musical. Sea el problema de optimización siguiente:

$$f(x_1, x_2, x_3) = 2(x_1 - 1)^6 + 5(x_2 - 3)^4 + 4(x_3 - 2)^2 + 1 \quad [1.20]$$

	Variable x_1	Variable x_2	Variable x_3	Valor de f
Rango 1	1	2	2	6
Rango 2	2	3	4	19
Rango 3	2	1	1	87

Tabla 1.3: Valores iniciales de la memoria armónica.

Se puede ver, de manera inmediata, que el mínimo de la función recae en el vector $(1, 3, 2)$. Sin embargo, **HSA** encuentra esa misma solución de un modo distinto.

Como se muestra en la tabla 1.3, la memoria armónica (***HS***) se estructura inicialmente según una generación aleatoria de vectores cuyas componentes se ordenan según el valor de la función objetivo. Supóngase que x_1 escoge aleatoriamente el valor 1 del conjunto $\{1, 2, 2\}$, x_2 el valor 2 de $\{2, 3, 1\}$ y x_3 el valor 4 de $\{2, 4, 1\}$. Puesto que $f(1, 2, 4) = 22$, la nueva solución $(1, 2, 4)$ se incluye en la memoria armónica y la peor solución anterior $(2, 1, 1)$ se excluye de ella (tabla 1.3).

	Variable x_1	Variable x_2	Variable x_3	Valor de f
Rango 1	1	2	2	6
Rango 2	2	3	4	19
Rango 3	1	2	4	22

Tabla 1.4: Nueva memoria armónica.

Procediendo de forma similar, **HSA** improvisa finalmente la solución $(1, 3, 2)$, que es el óptimo global, cuyo valor es $f(1, 3, 2) = 1$.

Análogamente, las reglas armónicas clásicas sostienen que la cadencia perfecta (Dominante-Tónica) de cualquier tonalidad (por ejemplo, Do mayor) es la que mejor finaliza una pieza de música:

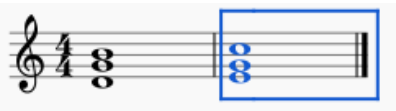

Figura 1.7: Cadencia perfecta (Dominante-Tónica).

¿Cómo llega el músico de Jazz a esta cadencia? Supóngase que en su memoria armónica existen los siguientes acordes:

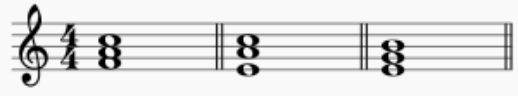

Figura 1.8: Posibles acordes de la memoria armónica.

Las correspondientes cadencias son:

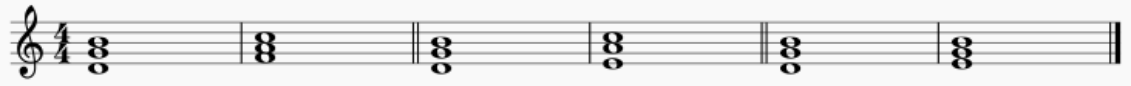

Figura 1.9: Cadencias.

Si el músico escoge la nota Do del primer acorde, Mi del segundo y Sol del tercero, forma el nuevo acorde Do-Mi Sol, que en su primera inversión adquiere la apariencia de la figura 1.7 (recuadro de color azul), y se obtiene así la cadencia conclusiva que mejora la memoria armónica. Sale de ésta el acorde Mi-Sol-Si, que está en el tercer compás de la figura 1.8

El siguiente ejemplo ilustra el comportamiento de la ecuación [1.18]. Sea de nuevo la función [1.20], cuyo mínimo recae en $(1, 3, 2)$, y la tabla 1.3. Considérese:

- $\boldsymbol{HMS} = 9$
- $\boldsymbol{HMCR} = 0.9$
- $\boldsymbol{PAR} = 0.3$
- $m = 2$
- $n = 3$
- $x_1, x_2, x_3 \in \{1, 2, \dots, 9\}$

La matriz de memoria armónica tiene la forma:

$$\begin{matrix} 1 & 2 & 2 & 6 \\ 2 & 3 & 4 & 19 \\ 2 & 1 & 1 & 87 \end{matrix}$$

Y de la ecuación [2.18], se tiene:

$$\frac{\partial f}{\partial x_i}|_{x_1=1} = \frac{1}{9} \times (1 - 0.9) + \frac{3 \times 1}{9} \times 0.9 \times (1 - 0.3) + \frac{3 \times 0}{9} \times 0.9 \times 0.3 = 0.2210$$

$$\frac{\partial f}{\partial x_2}|_{x_2=3} = \frac{1}{9} \times (1 - 0.9) + \frac{3 \times 3}{9} \times 0.9 \times (1 - 0.3) + \frac{3 \times 0}{9} \times 0.9 \times 0.3 = 0.6411$$

$$\frac{\partial f}{\partial x_3}|_{x_3=2} = \frac{1}{9} \times (1 - 0.9) + \frac{3 \times 2}{9} \times 0.9 \times (1 - 0.3) + \frac{3 \times 1}{9} \times 0.9 \times 0.3$$
$$= 0.5209$$

Si ahora se parte de la tabla 1.4, que contiene una solución mejorada, la matriz de memoria armónica se transforma en:

$$\begin{matrix} 1 & 2 & 2 & 6 \\ 2 & 3 & 4 & 19 \\ 1 & 2 & 4 & 22 \end{matrix}$$

Entonces, la ecuación [1.18] se reescribirá de la siguiente manera:

$$\frac{\partial f}{\partial x_i}|_{x_1=1} = \frac{1}{9} \times (1 - 0.9) + \frac{3 \times 1}{9} \times 0.9 \times (1 - 0.3) + \frac{3 \times 0}{9} \times 0.9 \times 0.3 = 0.2210$$

$$\frac{\partial f}{\partial x_2}|_{x_2=3} = \frac{1}{9} \times (1 - 0.9) + \frac{3 \times 3}{9} \times 0.9 \times (1 - 0.3) + \frac{3 \times 0}{9} \times 0.9 \times 0.3 = 0.6411$$

$$\frac{\partial f}{\partial x_3}|_{x_3=2} = \frac{1}{9} \times (1 - 0.9) + \frac{3 \times 2}{9} \times 0.9 \times (1 - 0.3) + \frac{3 \times 2}{9} \times 0.9 \times 0.3$$
$$= 0.6109$$

Permanecen inmutables los valores de $\frac{\partial f}{\partial x_i}|_{x_1=1}$ y $\frac{\partial f}{\partial x_2}|_{x_2=3}$, pero aumenta el de $\frac{\partial f}{\partial x_3}|_{x_3=2}$ ($0.6109 > 0.5209$) porque la probabilidad de ajuste tonal (P_{tono}) también ha aumentado ($0.1799 > 0.0899$) en la siguiente iteración.

1.4. Convergencia y criterio de parada

En el epígrafe anterior se ha presupuesto que todas las componentes de la solución global del problema de optimización están inicialmente presentes en $\boldsymbol{HM}$. Si no fuese así, **HSA** empezaría por establecer el parámetro $\boldsymbol{HMCR} \in [0, 1]$. Si el valor generado con distribución uniforme en $[0, 1]$ es superior al valor actual de $\boldsymbol{HMCR}$, el algoritmo encuentra valores aleatorios dentro del rango permitido, sin tener en cuenta $\boldsymbol{HM}$. Si

$\boldsymbol{HMCR} = 0.95$, el algoritmo elegirá de $\boldsymbol{HM}$, en el siguiente paso, un valor de la variable con una probabilidad del 95% (Geem, Kim, Loganathan, 2001).

Para mejorar las soluciones encontradas y evitar los óptimos locales, se debe ajustar el tono intercambiando valores vecinos pertenecientes a un conjunto de valores posibles. Por ejemplo, si dicho conjunto es $\{2, 4, 5, 6, 8, 9\}$, el valor $\{5\}$ se podrá mover a sus vecinos $\{4\}$ o $\{6\}$ en el proceso de ajuste tonal. Si $\boldsymbol{PAR} = 0.25$, entonces **HSA** escogerá $\{4\}$ o $\{6\}$ con idéntica probabilidad igual a $0.125 = \frac{0.25}{2}$.

Análogamente, si el conjunto de las notas que puede dar un instrumento musical en un instante determinado es {Do, Re *b*, Mi, Fa #, Sol #, La, Si *b*} (siendo los valores de los parámetros $\boldsymbol{HMCR} = 0.95$ y $\boldsymbol{PAR} = 0.25$) y el instrumento tiene {Do, Mi, La} en su memoria armónica, el algoritmo escogerá una nota del conjunto {Do, Mi, La} con probabilidad 0.95 o una nota del conjunto {Do, Re *b*, Mi, Fa #, Sol #, La, Si *b*} con probabilidad 0.05. En este caso, la nota {Mi}, por ejemplo, se puede mover a la posición que ocupa {Re *b*} o {Fa #} con idéntica probabilidad igual a $0.125 = \frac{0.25}{2}$.

HSA termina cuando se logra el máximo número de iteraciones y la función objetivo alcanza un valor óptimo (Srinivasa Rao et al., 2011).

Para demostrar la capacidad de convergencia de HSA, considérense los siguientes parámetros de $\boldsymbol{HM}$:

- $\boldsymbol{M}$: Tamaño de $\boldsymbol{HM}$
- $\boldsymbol{N}$: Número de variables
- $\boldsymbol{L}$: Número de posibles valores de las variables
- $\boldsymbol{H_i}$: Número de valores óptimos de la variable i en $\boldsymbol{HM}$
- $\boldsymbol{H_r}$: Tasa de consideración de la memoria armónica ($\boldsymbol{HMCR}$)
- $(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z})$: Vector de la solución óptima

Teniendo en cuenta la ecuación [1.18], que las i variables son independientes y que en el momento de alcanzar la solución óptima se cumple $\boldsymbol{PAR} = 0$ (puesto que ya se ha logrado un ajuste definitivo del tono y, además, es un operador opcional), la probabilidad de encontrar dicha solución óptima $\boldsymbol{H}$ es:

$$\Pr(\boldsymbol{H}) = \prod_{i=1}^{\boldsymbol{N}} [H_r \frac{\boldsymbol{H_i}}{\boldsymbol{M}} + (1 - \boldsymbol{H_r}) \frac{1}{\boldsymbol{L}}] \qquad [1.21]$$

Inicialmente, $\boldsymbol{HM}$ se completa con valores aleatorios de las variables; si ninguno de ellos es el óptimo, entonces $\boldsymbol{H_1} = \boldsymbol{H_2} = \cdots = \boldsymbol{H_N} = 0$, con lo que:

$$\Pr(\boldsymbol{H}) = \prod_{i=1}^{\boldsymbol{N}} \left[(1 - \boldsymbol{H_r})\frac{1}{\boldsymbol{L}}\right] = [(1 - \boldsymbol{H_r})\frac{1}{\boldsymbol{L}}]^{\boldsymbol{N}} \qquad [1.22]$$

Esta probabilidad es muy pequeña, ya que $\boldsymbol{H_r} = 0.95$ y, por consiguiente, $1 - \boldsymbol{H_r} = 0.05$. De modo que, por pequeños que sean los valores de $\boldsymbol{L}$ y $\boldsymbol{N}$ (siempre ≥ 2, aunque pueden llegar a ser muy grandes), se tendrá que $\Pr(\boldsymbol{H}) \leq 6.25^{-4}$. Ahora bien, a medida que las soluciones candidatas $(*, \boldsymbol{Y}, \boldsymbol{Z})$, $(\boldsymbol{X}, *, \boldsymbol{Z})$ y $(\boldsymbol{X}, \boldsymbol{Y}, *)$ vayan aproximándose a la solución óptima $(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z})$ y, por tanto, la función objetivo se acerque a su valor óptimo, la cantidad $\boldsymbol{H_i}$ de valores óptimos de la variable i en $\boldsymbol{HM}$ crecerá con cada iteración, lo que implica que el valor de $\Pr(\boldsymbol{H})$ aumentará de forma gradual.

1.5. Implementación

Las tres componentes de **HSA**, descritas en la sección anterior, se pueden implementar mediante el lenguaje de programación Matlab/Octave. Considérese, por ejemplo, la función "banana" de Rosenbrock:

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2 \qquad [1.23]$$

En el dominio $(x, y) \in [-10, 10] \times [-10, 10]$, la función tiene un mínimo global $f_{mínimo} = 0$ en el punto $(1, 1)$. Mediante el archivo "hs_simple.m" (Yang, 2010), después de 25000 iteraciones en menos de un minuto, se encuentra que una buena estimación de la solución óptima es (1.005, 1.0605).

Sea ahora la función de dos variables de Michalewicz (figura 1.11):

$$f(x, y) = -\sin(x)\, sin^{20}\left(\frac{x^2}{\pi}\right) - \sin(y) sin^{20}(\frac{2y^2}{\pi}) \qquad [1.24]$$

En el dominio $0 \leq x \leq \pi$ y $0 \leq y \leq \pi$, dicha función tiene un mínimo global $f_{min} \approx -1{,}801$ en (2,20319, 1,57049), que se alcanza, tras efectuar las oportunas modificaciones en el archivo "hs_simple.m", después de 23.000 evaluaciones.

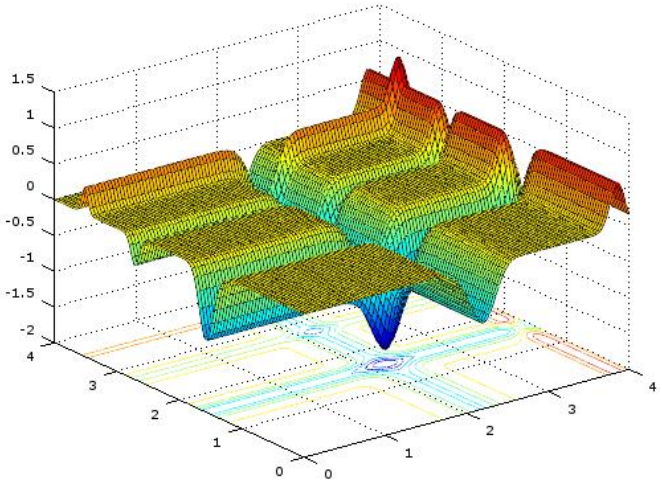

Figura 1.10: Representación gráfica de la función de Michalewicz. Las bandas de puntos situados en el plano horizontal corresponden a óptimos locales; en el cruce de ambas se encuentra el óptimo global.

Funciones como las anteriores se conocen con el nombre de funciones de prueba o benchmark functions y son importantes para validar y comparar el rendimiento de los algoritmos en los problemas de optimización. Con ese fin, es necesario identificar el tipo de problema para el cual es adecuado un algoritmo concreto, lo que sólo es posible si el conjunto de pruebas es suficientemente grande (Jamil y Yang, 2013). En cualquier caso, son muchos los ejemplos (algunos de ellos, mencionados en este trabajo) en los que estas funciones se han probado con éxito mediante **HSA** o sus variantes.

2. CONTEXTO Y PRINCIPALES VARIANTES

2.1. Diversificación e intensificación

Los algoritmos heurísticos se usan para encontrar buenas soluciones a problemas de optimización en un tiempo razonable. Recurren para ello al procedimiento de "ensayo y error"; éste, dicho sea de paso, no garantiza el éxito completo, aunque puede llegar a estar muy cerca de conseguirlo. Y esto es así porque los parámetros de búsqueda se centran en las variaciones locales de las funciones dentro de determinadas regiones, en tanto que las soluciones óptimas pueden encontrarse fuera de aquéllas. A pesar de todo, y siempre que el tiempo a invertir se prevea que pueda ser excesivamente largo, se suelen aceptar las soluciones factibles locales como soluciones de muchos problemas de optimización.

Recientemente se han desarrollado algoritmos metaheurísticos más exitosos y simples que los heurísticos, pues no requieren de conocimientos matemáticos complejos por parte de los programadores y se basan en la simulación de procesos naturales y/o sociales perfectamente conocidos (Yang, 2010). Combinan, pues, tácticas de bajo nivel con técnicas de exploración y explotación de los espacios de búsqueda; entre ellos, se encuentra **HSA** (Geem, 2009-a; Yang, 2009).

Los algoritmos metaheurísticos modernos constan de dos importantes componentes: la diversificación y la intensificación, terminología derivada del algoritmo de búsqueda tabú (Glover y Laguna, 1997). Cualquier algoritmo metaheurístico exitoso ha de guardar un buen equilibrio entre estas dos características (Kirkpatrick, Gelatt y Vecchi, 1983; Kennedy y Eberhart, 1995; Glover y Laguna, 1997; Bonabeau, Dorigo y Theraulaz, 1999; Blum y Roli, 2003; Dorigo y Stutzle, 2004; Dorigo y Blum, 2005; Karaboga y Basturk, 2008; Yang, 2010). Esto quiere decir que deberá ser capaz de generar un amplio rango de soluciones, entre las cuáles se encuentren las potencialmente óptimas; y también que habrá de ser apto para explorar eficazmente el espacio de soluciones, a la vez que pueda intensificar la búsqueda de éstas en la vecindad de una solución óptima o casi óptima. Por dicho motivo, cada parte del espacio de soluciones debe ser accesible a la búsqueda, aunque no necesariamente sea explorada por el algoritmo en el transcurso de ésta.

La diversificación es el proceso de explorar eficientemente muchos lugares del espacio de soluciones, desechando, siempre que sea posible, los óptimos locales. Si la diversificación es demasiado fuerte, se ralentizará la convergencia del algoritmo y las soluciones saltarán de un lugar a otro alrededor de las potencialmente óptimas; si es demasiado débil, existe el riesgo de limitar el espacio explorado, lo que da lugar a que las soluciones encontradas sean sesgadas y estén atrapadas en óptimos locales.

La intensificación pretende explotar la historia y la experiencia del proceso de búsqueda. Esta característica asegura la velocidad de convergencia, reduce la aleatoriedad y limita la diversificación. Si es muy fuerte, da lugar a una convergencia prematura, lo que lleva al hallazgo de óptimos locales sesgados o a soluciones sin sentido, como a menudo ocurre en las búsquedas basadas en el gradiente o en el método de Newton-Raphson; y si es muy débil, la convergencia se vuelve lenta.

En suma, la diversificación se presenta en forma de aleatorización para explorar el espacio de búsqueda de manera efectiva, mientras que la intensificación consiste en la explotación de las soluciones encontradas con el fin de seleccionar las mejores a través

del elitismo, el uso de la memoria armónica o ambas cosas a la vez. Por lo general, el algoritmo empieza a encontrar soluciones a partir de algún generador aleatorio; gradualmente, reduce la diversificación mientras aumenta la intensificación, aunque está todavía por dilucidar cómo se puede hacer esto de la manera más rápida posible.

En **HSA**, la tasa de ajuste tonal (***PAR***) y la aleatoriedad ($\boldsymbol{P}_{aleatorización} = 1 - \boldsymbol{r}$) controlan la diversificación, de modo que las nuevas soluciones locales se guardan en la memoria armónica y se explora eficientemente el espacio de búsqueda global. La generación de nuevas soluciones a través de la asignación al azar está, al menos, en el mismo nivel de eficacia que en otros algoritmos metaheurísticos; sin embargo, el parámetro ***PAR*** supone un refinamiento de las soluciones locales encontradas. Es decir, mientras que la aleatorización facilita la exploración del espacio de búsqueda de forma eficiente, el ajuste tonal asegura que las soluciones recién generadas sean suficientemente buenas o que no difieran demasiado de las ya existentes.

Por su parte, la intensificación está representada por la tasa $\boldsymbol{r}$, la cual se ve reforzada por ***PAR***. Cuanto más elevada sea dicha tasa, más valores almacenados en la memoria armónica podrán reutilizarse; y cuanto más baja sea, más lentamente convergerá el algoritmo.

Una buena afinación de los parámetros es imprescindible para mejorar la idoneidad de un algoritmo. Sin embargo, los métodos utilizados para resolver un problema particular dependen en gran medida del tipo y de las características del propio problema de optimización. Y se ha podido demostrar que ni hay un método universal igualmente válido para todos los problemas de optimización ni, por lo general, hay garantía alguna de encontrar la solución óptima en los problemas de optimización global. De hecho, los denominados "No Free Lunch Theorems" establecen que, si un algoritmo A supera a otro algoritmo B en la búsqueda de los extremos de una función objetivo, entonces el algoritmo B superará al algoritmo A en la búsqueda de los extremos de otras funciones objetivo (Wolpert y Macready, 1997).

2.2 Hibridaciones y nuevas configuraciones de los parámetros

Aunque se ha probado la potencia y la eficiencia de **HSA**, todavía no está completamente claro por qué y bajo qué condiciones funciona bien este algoritmo. Es más, la algoritmia metaheurística carece de un marco teórico general que proporcione alguna orientación analítica para responder a cuestiones del tipo siguiente: ¿Cómo

mejorar la eficiencia de un algoritmo para encontrar la solución óptima de un problema concreto? ¿Qué condiciones son necesarias para garantizar una buena tasa de convergencia? ¿Cómo probar que se han alcanzado los óptimos globales mediante cierto algoritmo metaheurístico? Las preguntas anteriores siguen estando abiertas y es preciso efectuar rigurosos estudios para despejar las dudas. Lo alentador es que muchos investigadores están interesados en hacer frente a estos retos y ya han logrado avances importantes.

A falta del marco teórico referido, se puede recurrir a la reconfiguración de los parámetros y al diseño de algoritmos híbridos para mejorar el rendimiento de **HSA**. Es decir, con el fin de disminuir la sensibilidad al ruido e incrementar la precisión y el grado de convergencia de las soluciones, se utilizan diversos algoritmos ya existentes para desarrollar nuevas variantes en las que se reajustan los parámetros (Geem, 2009-a; Yang, 2010; Omran y Mahdavi, 2008).

2.2.1. Búsqueda Armónica mejorada

La búsqueda armónica mejorada (**IHS**) es un algoritmo que genera vectores solución a través del ajuste dinámico de los parámetros $\boldsymbol{PAR}$ y $\boldsymbol{BW}$; se logra así una mejora de la precisión, a la vez que aumenta la velocidad de convergencia (Mahdavi, Fesanghary y Damangir, 2007). Ha sido probado con éxito en problemas de optimización en ingeniería y se ha evaluado con éxito, mediante determinadas funciones de prueba, frente a otros algoritmos. Los parámetros se definen así:

$$\boldsymbol{PAR} = \boldsymbol{PAR}_{min} + \frac{\boldsymbol{PAR}_{max} - \boldsymbol{PAR}_{min}}{\boldsymbol{NI} - 1} \times (\boldsymbol{iter} - 1) \qquad [2.1]$$

$$\boldsymbol{BW} = \boldsymbol{BW}_{max} \times e^{\boldsymbol{C} \times (\boldsymbol{Iter} - 1)} \qquad [2.2]$$

$$\boldsymbol{C} = \frac{Ln(\frac{\boldsymbol{BW}_{min}}{\boldsymbol{BW}_{max}})}{\boldsymbol{NI} - 1} \qquad [2.3]$$

Donde:

- $\boldsymbol{PAR}$ = Tasa de ajuste del tono para cada iteración
- $\boldsymbol{PAR_{min}}$ = Tasa mínima de ajuste de tono
- $\boldsymbol{PAR_{max}}$ = Tasa máxima de ajuste de tono

- NI = Número máximo de improvisaciones
- $iter$ = Número de iteración actual
- BW = Ancho de banda para cada iteración
- BW_{min} = Ancho mínimo de banda
- BW_{max} = Ancho máximo de banda

El parámetro PAR crece linealmente con el número de iteraciones, mientras que el ancho de banda BW decrece exponencialmente (ecuaciones [2.1], [2.2] y [2.3]). Con estos cambios, **IHS** encuentra mejores soluciones globales y locales que **HSA** (Das et al., 2010; Geem y Sim, 2010-b). Su principal inconveniente, no obstante, es que hace falta especificar los valores de BW_{min} y BW_{max}, cosa que no es nada fácil de hacer y que depende del problema concreto a solucionar (Omran y Mahdavi, 2008).

2.2.2. Búsqueda armónica global

La búsqueda armónica global (**GHS**) es un algoritmo híbrido que combina **HSA** con el de partículas de enjambre. Cada partícula representa un candidato a la solución óptima, y su posición depende de su propia experiencia (la mejor posición ocupada por sí misma) y de la experiencia del enjambre (mejores posiciones ocupadas por las partículas).

para cada $i \in [1, N]$ *hacer*
  si $U\ (0,1)\ \leq HMCR$ *entonces* $/*$ consideración de la memoria $*/$
    empezar
    $x_i^{'} = x_i^j$, donde $j \sim (1, \ldots, HMS)$
    si $U(0,1) \leq PAR(t)$ *entonces* $/*$ adjuste $*/$
    empezar
      $x_i^{'} = x_k^{mejor}$, donde mejor es el índice de la mejor memoria en HM y $k \sim U(1, N)$
    fin_si
  en otro caso $/*$ selección aleatoria $*/$
    $x_i^{'} = LB_i\ + r\ \times (UB_i - LB_i)$
  fin_si
hecho

Figura 2.1: Pseudocódigo para improvisación de una nueva armonía en ***GHS****.*

La manera de ajustar el tono en **GHS** consiste en imitar el mejor armónico de la memoria armónica, lo cual le permite trabajar eficazmente en problemas continuos y discretos. En general, **GHS** es mejor que **IHS** y **HSA** cuando se aplica a problemas de grandes dimensiones y hay presencia de ruido (Omran y Mahdavi, 2008). En problemas de simulación de redes de agua, sin embargo, los resultados son dispares (Geem, 2009-b).

GHS consta de los mismos pasos que **IHS**, con la excepción de que la improvisación de la nueva armonía se efectúa como se indica en el pseudocódigo mostrado en la figura 2.1 (Omran y Mahdavi, 2008).

HSA	$\boldsymbol{HMS} = 5$ $\boldsymbol{HMCR} = 0.9$ $\boldsymbol{PAR} = 0.3$ $\boldsymbol{BW} = 0.1$
IHS	$\boldsymbol{HMS} = 5$ $\boldsymbol{HMCR} = 0.9$ $\boldsymbol{PARmin} = 0.01$ $\boldsymbol{PARmax} = 0.99$ $\boldsymbol{BWmin} = 0.0001$ $\boldsymbol{BWmax} = \frac{1}{20 \times (x_i^U - x_i^L)}$
GHS	$\boldsymbol{HMS} = 5$ $\boldsymbol{HMCR} = 0.9$ $\boldsymbol{PARmin} = 0.01$ $\boldsymbol{PARmax} = 0.99$

*Tabla 2.1: Parámetros empleados en los algoritmos **HSA**, **IHS** y **GHS** para optimizar algunas funciones.*

En otro orden de cosas, se ha probado la eficacia de **GHS** ante **HSA** e **IHS** mediante la optimización de funciones como: Sphere, Schwefel, Step, Rosenbrock, Rotated Hyper-Ellipsoid, Generalized Schwefel, Rastrigin, Ackley, Griewank, Six Hump Camel-Back. El objetivo era encontrar el mínimo global, formalmente definido así: dada la función $f: R^{N_d} \rightarrow R$, encontrar $x^* \in R^{N_d} \mid \forall\, x \in R^{N_d}, f(x^*) \leq f(x)$. Excepto la función bidimensional Camel-Back, todas se implementaron en 30 dimensiones. Los resultados del problema de optimización se pueden presentar mediante las medias y las desviaciones típicas de 30 simulaciones, y en cada una de éstas se efectúan 50.000 evaluaciones de la función objetivo. Los parámetros empleados aparecen en la tabla 2.1.

2.2.3. Nueva búsqueda armónica global

La Nueva búsqueda armónica global (**NGHS**) combina **HSA** con el algoritmo de partículas de enjambre y con el genético (Zou et al., 2010).

para *cada* $i \in [1, N]$ ***hacer***
  $x_R = 2 \times x_i^{mejor} - x_i^{peor}$
  si $x_R > x_{iU}$ ***entonces***
    empezar
      $x_R = x_{iU}$
    fin
  en otro caso
  si $x_R < x_{iL}$ ***entonces***
      $x_R < x_{iL}$
    fin_si
  fin_si
  $x_i^{'} = x_i^{peor} + r \times (x_R - x_i^{peor})$ //% *posición actualizada*
  si $r \leq p_m$ ***entonces***
    $x_i^{'} = x_{iL} + r \times (x_{iU} - x_{iL})$ //% *mutacióngenética*
  fin_si

*Figura 2.2: Pseudocódigo para improvisar una nueva armonía mediante **NGHS**.*

El algoritmo se emplea para resolver problemas de fiabilidad, los cuales consisten en determinar la probabilidad de que un sistema sea seguro y eficiente sin violar ninguna restricción. Sirvan los siguientes ejemplos: (a) sistemas de programación no lineal entera y mixta y (b) protección de exceso de velocidad en turbinas de gas. También se ha empleado con éxito para resolver el problema de la mochila. El algoritmo **NGHS** muestra una convergencia más rápida y una mayor capacidad de exploración del espacio de soluciones que **IHS** y **HSA**. En la figura 2.2 se ha transcrito el pseudocódigo de **NGHS**.

2.2.4. Búsqueda armónica diferencial

En algunos experimentos con **HSA** y sus hibridaciones se han detectado problemas de convergencia, a veces prematura y a veces lenta, especialmente en el caso multimodal y en el de sobredimensión del espacio de búsqueda. Para evitar estos inconvenientes, el algoritmo de búsqueda armónica diferencial (**DHS**) reemplaza la

operación de ajuste de tono en **HSA** por una estrategia de mutación prestada de los algoritmos diferenciales evolutivos (**DEA**). Los experimentos muestran que la hibridación entre **DEA** y **HSA** produce muy buenos resultados. Dicha hibridación consiste en mutar la función objetivo con la diferencia de dos miembros aleatorios de la población (Chakraborty et al., 2009):

$$x_i^{'} = x_i + F(x_{r_1} - x_{r_2}) \qquad [2.4]$$

HSA	$\boldsymbol{HMCR} = 0.90$ $\boldsymbol{PAR} = 0.30$ $\boldsymbol{BW} = 0.01$
IHS	$\boldsymbol{HMCR} = 0.95$ $\boldsymbol{PAR}_{mínimo} = 0.35$ $\boldsymbol{PAR}_{máximo} = 0.99$ $\boldsymbol{BW}_{mínimo} = 1.00e - 06$ $\boldsymbol{BW}_{máximo} = \frac{1}{20(x_{máximo} - x_{mínimo})}$
GHS	$\boldsymbol{PAR}_{mínimo} = 0.01$ $\boldsymbol{PAR}_{máximo} = 0.99$ $\boldsymbol{HMCR} = 0.90$
DHS	$\boldsymbol{HMCR} = 0.90$ $F = 0.80$

*Tabla 2.2: Elección de parámetros de **HSA**, **IHS**, **GHS** y **DHS**.*

En la ecuación [2.4], F es una variable aleatoria uniforme en $[0, 1]$. Bajo ciertas condiciones experimentales aplicadas a seis funciones de prueba (Sphere, Rosenbrock, Rastriging, Griewank, Ackley y Shekel´s Fosholes) y según el teorema 2.7.1 (Chakraborty et al.), **DHS** tiene una varianza poblacional creciente, lo que garantiza mejor poder de exploración que **HSA**, **IHS** y **GHS**. Las condiciones aludidas se refieren a los parámetros, que se especifican en la tabla 2.2.

La justificación teórica de los resultados experimentales se encuentra en el siguiente apartado.

2.2.4.1. Eficiencia y capacidad exploratoria de DHS

Las interacciones entre las diversas componentes de **DHS** y sus hibridaciones adquieren una especial relevancia para el éxito del algoritmo. Pero existen otras consideraciones que refuerzan su eficiencia, como se verá a continuación.

Ante todo, la aplicación de **DHS** es relativamente fácil debido a la insensibilidad del algoritmo a los parámetros elegidos; esto quiere decir que no es necesario afinarlos exhaustivamente para obtener soluciones de calidad.

Además, **DHS** es un algoritmo metaheurístico basado en la población. Esta peculiaridad facilita la utilización en paralelo de varios grupos de armónicos y, por lo tanto, proporciona una implementación eficiente.

En definitiva, la conjunción de todas estas características (delicado equilibrio entre diversificación e intensificación, insensibilidad hacia los parámetros elegidos y empleo en paralelo de varios grupos de armónicos) es la clave del éxito de **DHS**. De hecho, es la clave del éxito de cualquier técnica metaheurística. Estas ventajas hacen que **DHS** sea muy versátil al combinarlo con otros algoritmos de su clase -es decir, mediante hibridaciones metaheurísticas (Omran y Mahdavi, 2008)- y aplicarlo a problemas de diversa índole (Geem, Kim y Loganathan, 2001; Lee y Geem, 2005; Geem, 2006-a; Geem, 2007-a; Omran y Mahdavi, 2008).

La eficiencia de la mayor parte de los algoritmos metaheurísticos depende de las características particulares de exploración y explotación y del adecuado equilibrio entre ambas. En general, estos algoritmos, especialmente los evolutivos, exploran el espacio en busca de mejores soluciones y explotan la selección de éstas en la siguiente iteración. El valor esperado de la varianza poblacional de una iteración a otra es una buena estimación del poder exploratorio de todo algoritmo metaheurístico (Chakraborty et al., 2009). Se tratará de encontrar, a continuación, dicho valor para **DHS**.

Como en **HSA**, cada dimensión se perturba de forma independiente y, sin pérdida de generalidad, se puede llevar a cabo este análisis para las componentes de una población unidimensional. Sea, por lo tanto, una población $x = \{x_1, x_2, \dots, x_m\}$, donde $x_i \in R, \forall\, i = 1, 2, \dots, m$. La varianza de esta población es:

$$Var(x) = \frac{1}{m}\sum_{i=1}^{m}[x_i - \overline{\boldsymbol{x}}]^2 = \overline{\boldsymbol{x^2}} - (\overline{\boldsymbol{x}})^2 \qquad [2.5]$$

En [2.5], $\overline{\boldsymbol{x}}$ es la media poblacional y $\overline{\boldsymbol{x^2}}$ es la media cuadrática poblacional. Si los elementos de la población se ven afectados por algunos valores aleatorios, $Var(x)$ será una variable aleatoria y $E[Var(x)]$ proporciona una medida del poder exploratorio de aquélla y puede calcularse como sigue.

Teorema 2.2.4.1 (Chakraborty et al.). Sea $x = \{x_1, x_2, \dots, x_m\}$ la población actual, sea y un vector aleatorio con cierta $\boldsymbol{HMCR}$ y sea z el vector obtenido tras ajustar el tono de y. Sea $\omega = \{\omega_1, \omega_2, \dots, \omega_m\}$ la población final, resultante de la selección aleatoria. Si $P_{\boldsymbol{HMCR}}$ es la probabilidad del parámetro $\boldsymbol{HMCR}$, el rango de los nuevos valores de x es $\{x_{mínimo}, x_{máximo}\} = \{-a, a\}$ y los números aleatorios requeridos se distribuyen uniformemente en $[0, 1]$, entonces:

$$E[Var(x)] = \frac{2}{3}Var(x) + \frac{a^2}{3}\left(\frac{m-1}{m}\right)(1 - P_{\boldsymbol{HMCR}}) + \frac{\overline{x^2}}{m}\left(P_{HMCR} + \frac{m-1}{m}\right) - \frac{(\bar{x})^2 P_{\boldsymbol{HMCR}}}{m^2}[P_{\boldsymbol{HMCR}} + 2(m-1)] \quad [2.6]$$

Demostración.

De la población actual se obtiene:

$$\overline{\boldsymbol{x}} = \frac{1}{m}\sum_{i=1}^{m} x_i \quad [2.7]$$

$$\overline{\boldsymbol{x^2}} = \frac{1}{m}\sum_{i=1}^{m} x_i^2 \quad [2.8]$$

Dadas las características del vector y, se podrá escribir:

$$y = \begin{cases} x_l, con\ probabilidad\ P_{\boldsymbol{HMCR}} \\ x_r, con\ probabilidad\ 1 - P_{\boldsymbol{HMCR}} \end{cases} \quad [2.9]$$

Se tiene:

$$\begin{cases} l, k \in \{1, 2, \dots, m\} \\ x_r \in \{-a = x_{mínimo}, x_{máximo} = a\} \\ k\ es\ v.a.\ con\ valores\ en\ \{1, 2, \dots, m\}\ con\ probabilidad\ P_k = P(i = k) = \dfrac{1}{m} \end{cases}$$

Por lo tanto,

$$\begin{cases} E(x_l) = \sum_l x_l P_l = \frac{1}{m}\sum_l x_l = \overline{\boldsymbol{x}} \\ E(x_l^2) = \sum_l x_l^2 P_l = \frac{1}{m}\sum_l x_l^2 = \overline{\boldsymbol{x^2}} \end{cases} \quad [2.10]$$

Similarmente, sea x_r un elemento de una población unidimensional aleatoria del espacio de búsqueda. Se puede modelar así:

$$x_r = -a + 2Ra \quad [2.11]$$

Usando estimaciones estadísticas de R, se puede escribir:

$$\begin{cases} E[R] = \int_0^1 Rf(R)dR = \int_0^1 RdR = [\frac{R^2}{2}]_0^1 = \frac{1}{2} \\ E[R^2] = \int_0^1 R^2 f(R)dR = \int_0^1 R^2 dR = [\frac{R^3}{3}]_0^1 = \frac{1}{3} \end{cases} \quad [2.12]$$

Y también se puede estimar x_r:

$$\begin{cases} E[x_r] = -a\ 2aE[R] = -a + 2a\frac{1}{2} = 0 \\ E[x_r^2] = a^2\{4E[R^2] - 4E[R] + 1\} = a^2\left\{\frac{4}{3} - \frac{4}{2} + 1\right\} = \frac{a^2}{3} \end{cases} \quad [2.13]$$

En virtud de las ecuaciones [2.12] y [2.13], se tiene:

$$E[y] = E[x_l]P_{\boldsymbol{HMCR}} + E[x_r](1 - P_{\boldsymbol{HMCR}}) = P_{\boldsymbol{HMCR}}\overline{\boldsymbol{x}} \quad [2.14]$$

$$E[y^2]E[x_l^2]P_{\boldsymbol{HMCR}} + E[x_r^2](1 - P_{\boldsymbol{HMCR}}) = \overline{\boldsymbol{x^2}}P_{\boldsymbol{HMCR}} + \frac{a^2}{3}(1 - P_{\boldsymbol{HMCR}}) \quad [2.15]$$

Ahora bien, puesto que el vector z resulta de la mutación del vector y, su estructura es $z = y + F(x_{\beta_1} - x_{\beta_2})$, donde F se distribuye uniformemente en $[0, 1]$. De las ecuaciones [2.12] se desprende:

$$\begin{cases} \overline{F} = E[F] = \frac{1}{2} \\ \overline{F^2} = E[F^2] = \frac{1}{3} \end{cases} \quad [2.16]$$

Además, x_{β_1}, x_{β_2} se escogen aleatoriamente de la población x, de tal forma que $\beta_1 \neq \beta_2$. Así, de las ecuaciones [2.10] se sigue:

$$\begin{cases} E\left[x_{\beta_1}\right] = E\left[x_{\beta_2}\right] = \overline{\boldsymbol{x}} \\ E\left[x_{\beta_1}^2\right] = E\left[x_{\beta_2}^2\right] = \overline{\boldsymbol{x}^2} \end{cases} \quad [2.17]$$

Por otro lado,

$$E\left[x_{\beta_1}x_{\beta_2}\right] = \sum_{\beta_1 \neq \beta_2}\sum_{\beta_2} x_{\beta_1}x_{\beta_2}P\left(x_{\beta_1}, x_{\beta_2}\right) = \frac{1}{m(m-1)}\sum_{\beta_1 \neq \beta_2}\sum_{\beta_2} x_{\beta_1}x_{\beta_2} =$$
$$\frac{1}{m(m-1)}\left\{\left(\sum x_{\beta_1}\right)^2 - \left(\sum x_{\beta_2}\right)^2\right\} = \frac{1}{m(m-1)}\{(m\overline{\boldsymbol{x}})^2 - m\boldsymbol{x}^{2\neg}\} \quad [2.18]$$

Ahora, de [2.17] y [2.18] se deduce:

$$E[z] = E[y] + E[F]E\left[x_{\beta_1} - x_{\beta_2}\right] = E[y] + \overline{F}\left(E\left[x_{\beta_1}\right] - E\left[x_{\beta_2}\right]\right) = \overline{\boldsymbol{x}}P_{\boldsymbol{HMCR}} \quad [2.19]$$

Y también;

$$E[z^2] = E\left[y + F\left(x_{\beta_1} - x_{\beta_2}\right)^2\right] = E\left[y^2 + F^2\left(x_{\beta_1}^2 + x_{\beta_2}^2 - 2x_{\beta_1}x_{\beta_2}\right) + 2Fy\left(x_{\beta_1} -\right.\right.$$
$$\left.\left. x_{\beta_2}\right)\right] = E[y^2] + \overline{F^2}\left(E\left[x_{\beta_1}^2\right] + E\left[x_{\beta_2}^2\right] - 2E\left[x_{\beta_1}x_{\beta_2}\right]\right) + 2\overline{F}E[y]\left(E\left[x_{\beta_1}\right] - E[x_{\beta_2}]\right) =$$
$$E[y^2] + 2\overline{F^2}\frac{m}{m-1}\left(\overline{\boldsymbol{x}^2} - (\overline{\boldsymbol{x}})^2\right) \quad [2.20]$$

Como $\omega = \{\omega_1, \omega_2, \ldots, \omega_m\}$ es la población final, obtenida al reemplazar y por un elemento aleatorio de la población original, cada ω_k se podrá escribir de la forma:

$$\omega_k = \begin{cases} z, con\ probabilidad\ p = \frac{1}{m} \\ x_k, con\ probabilidad\ q = 1 - \frac{1}{m} \end{cases} \quad [2.21]$$

Por lo tanto,

$$\begin{cases} E[\omega_k] = E[z]p + E[x_k](1-p) = \frac{1}{m}\overline{\boldsymbol{x}}P_{HMCR} + \frac{m-1}{m}\overline{\boldsymbol{x}} \\ E[\omega_k^2] = E[z^2]p + E[x_k^2](1-p) \end{cases} \quad [2.22]$$

Como $\overline{\omega} = \frac{1}{m}\sum \omega_k$, se tendrá:

$$E[\omega^2] = E[\frac{1}{m}\sum \omega_k]^2 \; = \frac{1}{m^2}E\left[\sum \omega_k^2 \; + \; \sum\sum_{l \neq j} \omega_l \omega_j\right] = \frac{1}{m^2}\left\{\sum E[\omega_k^2] + \sum\sum_{l \neq j} E\left[\omega_l \omega_j\right]\right\} = \frac{1}{m^2}\{mE[\omega_k^2] + m(m \; - 1)E^2[\omega_k]\} \quad [2.23]$$

Y, como $\overline{\omega^2} \; = \frac{1}{m}\sum \omega_k^2$, entonces:

$$E\left[\overline{\omega^2}\right] = \frac{1}{m}E[\sum \omega_k^2] = E[\omega_k^2] \quad [2.24]$$

La varianza de la población final es $Var(\omega) = \; \overline{\omega^2} \; - \; (\overline{\omega})^2$, luego:

$$E[Var(\omega)] = E\left[\overline{\omega^2} \; - \; (\overline{\omega})^2\right] = E[\omega_k^2] - \frac{1}{m}E[\omega_k^2] + \frac{m-1}{m}E^2[\omega_k] = \frac{m-1}{m}\{E\left[\overline{\omega_k^2}\right] \; - \; E^2[\omega_k]\} \quad [2.25]$$

Sustituyendo en [2.25] los valores hallados en [2.22] y simplificando, se llega a la ecuación [2.6] del enunciado, como se quería demostrar.

Lema 2.2.4.1 (Chakraborty et al.). Considérese :

- $\boldsymbol{BW} \; = 0.25$
- $P_{\boldsymbol{PAR}} \; = 0.63$
- $P_{\boldsymbol{HMCR}} \; = 1$
- $m = 8$

Entonces,

$$E[Var(z)]_{\boldsymbol{DHS}} \; > \; E[Var(z)]_{\boldsymbol{HSA}} \quad [2.26]$$

Demostración.

Por [2.17] y [2.18], para **HSA** se cumple:

$$E[Var(z)] = \frac{m-1}{m}[\boldsymbol{HMCR} \times Var(x) + \boldsymbol{HMCR} \times (1 - \boldsymbol{HMCR}) \times (\bar{\boldsymbol{x}})^2 + \boldsymbol{HMCR} \times (1 - \boldsymbol{HMCR}) \times \boldsymbol{PAR} \times \boldsymbol{BW} \times \bar{\boldsymbol{x}} + \boldsymbol{HMCR} \times \boldsymbol{PAR} \times \boldsymbol{BW}^2 \times (\frac{1}{3} - \frac{\boldsymbol{HMCR} \times \boldsymbol{PAR}}{4}) + \frac{a^2}{3} \times (1 - \boldsymbol{HMCR})] \quad [2.27]$$

Sustituyendo $\boldsymbol{BW} = 0.25$, $P_{\boldsymbol{PAR}} = 0.63$ en [2.6], y asumiendo los valores $P_{\boldsymbol{HMCR}} = 1$ y $m = 8$, se obtiene:

$$E[Var(z)]_{HSA} = 0.85 \times Var(x) + 0.0069 \quad [2.28]$$

Considerando ahora $P_{\boldsymbol{HMCR}} = 1$ y $m = 8$. Entonces:

$$Var(z)_{DHS} = 0.901 \times Var(x) \quad [2.29]$$

Es decir,

$$E[Var(z)]_{\boldsymbol{DHS}} - E[Var(z)]_{\boldsymbol{HSA}} > 0$$

Con lo que se ha demostrado la validez de la ecuación [2.26].

3. RESOLUCIÓN DE PROBLEMAS DE OPTIMIZACIÓN CON RESTRICCIONES MEDIANTE HSA

3.1. Método de penalización exterior

Aunque muchas aplicaciones prácticas de los modelos de optimización incluyen restricciones, es importante estudiar los modelos sin restricciones porque los primeros resuelven los problemas de optimización a través de los segundos, utilizando para ello multiplicadores de Lagrange, funciones de penalización y funciones de barrera. Algunos algoritmos de optimización con restricciones parten de un punto dado y después buscan una dirección a través de la cual optimizan la función objetivo; este procedimiento, al fin y al cabo, es un problema sin restricciones, o bien con meras restricciones de acotación. También puede ocurrir que ciertas técnicas de optimización sin restricciones se extiendan

de manera natural para incluir restricciones, por lo que son útiles para idear soluciones de problemas con restricciones (Ramos Méndez, 2017).

En el método de las funciones de penalización, los problemas con restricciones se transforman en un único problema sin restricciones, lo que se consigue introduciendo las restricciones en la función objetivo a través de un término que penaliza la violación de alguna de éstas. Es decir, el término en cuestión hace que aumente el valor de la función a minimizar cuando los puntos se alejan de la región factible (Ramos Méndez, 2017). Así, el problema penalizado puede tratarse igual que un problema o una sucesión de problemas sin restricciones. Por lo tanto, se puede tratar de localizar el óptimo global de la función objetivo mediante **HSA**. Esto es, precisamente, lo que se hará a continuación, tomando algún ejemplo del método de penalización exterior con función cuadrática (**MPEFC**).

3.1.1. Funciones de penalización exterior

Sea el problema de programación no lineal con una restricción de igualdad:

$$\begin{cases} Minimizar\ f(x) \\ \quad Sujeto\ a: \\ \quad h(x) = 0 \end{cases}$$

Sea ahora $\mu > 0$ arbitrariamente grande y considérese el siguiente problema sin restricciones:

$$\begin{cases} Minimizar\ f(x) + \mu h(x)^2 \\ \quad Sujeto\ a: \\ \quad x \in R^n \end{cases}$$

Es obvio que cualquier solución óptima del problema anterior ha de ser tal que $h(x) \approx 0$, pues, de lo contrario, $f(x) + \mu h(x)^2$ tendría un valor muy alto y no podría ser el mínimo.

Considérese ahora el siguiente problema con restricción de desigualdad:

$$\begin{cases} Minimizar\ f(x) \\ \quad Sujeto\ a: \\ \quad g(x) \leq 0 \end{cases}$$

La función $f(x) + \mu h(x)^2$ es un término de penalidad inadecuado para este problema, pues incurre en penalidad para $x \mid g(x) > 0$ y para $x \mid g(x) < 0$, siendo los últimos factibles. Por eso, habría que penalizar el problema de forma más adecuada, como, por ejemplo:

$$\begin{cases} Minimizar\ f(x) + Máx\{0, g(x)\} \\ Sujeto\ a: \\ x \in R^n \end{cases}$$

Si $g(x) \leq 0$, entonces $Máx\{0, g(x)\} = 0$ y no se incurre en penalidad. Por el contrario, si $g(x) > 0$, sí se incurre en penalidad.

Desde un punto de vista práctico, es deseable trabajar con funciones diferenciables; pero $Máx\{0, g(x)\}$ no es diferenciable en $g(x) = 0$, incluso si $g(x)$ lo es. De modo que una función de penalidad más apropiada para el caso de restricción de desigualdad podría ser $Máx\{0, g(x)\}^2$ (Bazaraa et al., 2006; Ramos Méndez, 2017).

De forma más general, sea el *problema primal*:

$$\begin{cases} Minimizar\ f(x) \\ Sujeto\ a: \\ g(x) \leq 0 \\ h(x) = 0 \\ x \in X \end{cases}$$

Donde $f: R^n \to R^1;\ g: R^n \to R^m;\ h: R^n \to R^l$ son funciones continuas en R^n y $\emptyset \neq X \subset R^n$. Una función de penalización adecuada para el problema anterior puede ser:

$$\alpha(x) = \sum_{i=1}^{m} \varphi[g_i(x)] + \sum_{i=1}^{l} \theta[h_i(x)]$$

Aquí, $\varphi\ y\ \theta$ son funciones reales de variable real, continuas y con las propiedades:

$$\begin{cases} Si\ y \leq 0 \Longrightarrow \varphi(y) = 0 \\ Si\ y > 0 \implies \varphi(y) > 0 \end{cases} \qquad \begin{cases} Si\ y = 0 \implies \theta(y) = 0 \\ Si\ y \neq o \implies \theta(y) > 0 \end{cases}$$

Por ejemplo,

$$\varphi(y) = [Máximo\{0, y\}]^p; \theta(y) = |y|^p$$

En la ecuación anterior, p es un número entero positivo. La función de penalidad es, generalmente, de la forma:

$$\alpha(x) = \sum_{i=1}^{m} [Máximo\{0, g_i(x)\}]^2 + \sum_{i=1}^{l} |h_i(x)|^p$$

Se llamará *función auxiliar* a la función:

$$f(x) + \mu\alpha(x)$$

Entonces, si α es una función de penalización, el método de penalización básico consiste en resolver el *problema penalizado*:

$$\begin{cases} Minimizar\ \delta(\mu) \\ Sujeto\ a: \\ \mu \geq 0 \\ Siendo: \\ \delta(\mu) = Inf\{f(x) + \mu\alpha(x) | x \in X\} \end{cases}$$

Los resultados siguientes (cuyas demostraciones se encuentran en Bazaraa et al., 2006 y Ramos Méndez, 2017) prueban que mediante la resolución del *problema penalizado* se puede resolver el *problema primal*:

Lema 3.2.1 Sean $f, g_1, \dots, g_m, h_1, \dots, h_l$ son funciones continuas en R^n, $\emptyset \neq X \subset R^n$ y α una función de penalidad continua. Si $\forall \mu, \exists x_\mu \mid \delta(\mu) = f(x_\mu) + \mu\alpha(x_\mu)$, entonces se cumple:

$$\begin{cases} Inf\{f(x) | x \in X; g(x) \leq 0; h(x) = 0\} \geq Sup_{\mu \geq 0}\delta(\mu) \\ f(x_\mu)\ es\ una\ función\ no\ decreciente\ de\ \mu \geq 0 \\ \delta(\mu)\ es\ una\ función\ no\ decreciente\ de\ \mu \\ \alpha(x_\mu)\ es\ una\ función\ no\ creciente\ de\ \mu \end{cases}$$

Teorema 3.2.1. Sean $f, g_1, \dots, g_m, h_1, \dots, h_l$ *son funciones continuas en* R^n, $\emptyset \neq X \subset R^n$ *y* α *una función de penalidad continua. Supóngase que el problema primal tiene al menos una solución factible y que* $\forall \mu$ *existe una solución* x_μ *del problema:*

$$\begin{cases} Minimizar\ f(x) + \mu\alpha(x) \\ Sujeto\ a: \\ x \in X \end{cases}$$

Supóngase, además, que la sucesión $\{x_\mu\}$ *está contenida en un subconjunto compacto de* X*. Entonces, se cumple:*

$$\begin{cases} Inf\{f(x) | x \in X; g(x) \leq 0; h(x) = 0\} = Sup_{\mu \geq 0}\delta(\mu) = \lim_{\mu \to \infty} \delta(\mu) \\ Donde: \\ \delta(\mu) = Inf\{f(x) + \mu\alpha(x) | x \in X\} = f(x_\mu) + \mu\alpha(x_\mu) \end{cases}$$

Por otra parte, el límite de cualquier subsucesión convergente de $\{x_\mu\}$ *es una solución óptima del problema original y* $\lim_{\mu \to \infty} \mu\alpha(x_\mu) = 0$.

Corolario 3.2. Si $\alpha(x_\mu) = 0\ para\ algún\ \mu$*, entonces* x_μ *es una solución óptima del problema primal.*

La solución del *problema penalizado* puede hacerse arbitrariamente próxima a la solución óptima del *problema primal* haciendo que μ sea arbitrariamente grande. Pero entonces se pueden encontrar dificultades de tipo numérico, debido a que el *problema penalizado* puede estar mal acondicionado. Y es que, cuando μ es muy grande, las técnicas de optimización sin restricciones conducen con rapidez a un punto factible que puede encontrarse alejado del óptimo, por lo que el algoritmo converge prematuramente.

3.1.2. Ejemplo de optimización por MPEFC

Ejemplo 3.1.2. Considérese el siguiente problema de optimización: *Resolver, utilizando el* ***MPEFC****, el siguiente problema de programación no lineal:*

$$\begin{cases} Minimizar\ \theta(x_1, x_2) = x_1^2 + x_2^2 \\ Sujeto\ a: \\ x_1 \geq 0 \\ x_2 \geq 0 \\ x_1 x_2 - 1 = 0 \end{cases}$$

Solución. Función de penalidad para restricciones del tipo $g_i(x_j) \geq 0, (j = 1,2)$:

$$p_k(x_j) = t_k \sum_{i=1}^{m} [\min(0, g_i(x_j))]^2 \qquad [3.1]$$

Función de penalidad para restricciones del tipo $g_i(x_j) = 0, (j = 1,2)$:

$$p_k(x_j) = t_k \sum_{i=1}^{m} [\min(g_i(x_j))]^2 \qquad [3.2]$$

Función auxiliar:

$$\varphi_k(x_1, x_2) = x_1^2 + x_2^2 + t_k[\min(0, x_1)]^2 + t_k[\min(0, x_2)]^2 + t_k[\min(x_1 x_2 - 1)]^2 \qquad [3.3]$$

Pero

$$[\min(0, g_i(x_j))]^2 = [\frac{g_i(x) - |g_i(x)|}{2}]^2 \qquad [3.4]$$

Además, $\varphi_k(x_1, x_2)$ es simétrica en x_1 y x_2. Luego el problema auxiliar es:

$$Minimizar\ \varphi_k(x_1, x_2) = 2x_1^2 + 2t_k(\frac{x_1 - |x_1|}{2})^2 + t_k(x_1^2 - 1)^2 \qquad [3.5]$$

A su vez, el problema se puede simplificar así (recuérdese que $\frac{x_1 - |x_1|}{2} = \frac{x_1 - x_1}{2} = 0, \forall x_1 \geq 0$):

$$Minimizar\ \varphi_k(x_1, x_2) = 2x_1^2 + t_k(x_1^2 - 1)^2 \qquad [3.6]$$

Condición de mínimo:

$$\frac{\partial \varphi}{\partial x_1} = 4x_1 + 4t_k(x_1^2 - 1)x_1 = 0 \qquad [3.7]$$

De modo que:

$$\begin{cases} x_1(t_k) = x_2(t_k) = 0 \\ x_1(t_k) = x_2(t_k) = +\sqrt{\frac{t_k - 1}{t_k}} \end{cases} \qquad [3.8]$$

Como en la región factible se cumple que $x_1, x_2 \geq 0$, entonces sólo se tomarán las raíces positivas. Por lo tanto, la solución directa es:

$$x_1(t_k) = x_2(t_k) = \lim_{t_k \to \infty} \sqrt{\frac{t_k - 1}{t_k}} = \lim_{t_k \to \infty} \sqrt{\frac{t_k - \frac{1}{t_k}}{\frac{t_k}{t_k}}} \qquad [3.9]$$

O sea:

$$(x_1(t_k), x_2(t_k)) = (1,1) \qquad [3.10]$$

Por otra parte, la tabla de iteraciones es:

$\boldsymbol{k}$	$\boldsymbol{t_k}$	$\boldsymbol{x_1(t_k)}$	$\boldsymbol{x_2(t_k)}$	$\boldsymbol{\varphi_k(x_1, x_2)}$	$\boldsymbol{\theta_k(x_1, x_2)}$
1	1	0.000	0.000	1.000	0.000
2	10	0.948	0.948	1.900	1.800
3	50	0.990	0.990	1.980	1.960
4	100	0.995	0.995	1.990	1.980
5	1000	0.999	0.999	1.999	1.998
	∞	1.000	1.000	2.000	2.000

*Tabla 3.1: Cálculos para la optimización por **MPEFC**.*

Se pone de manifiesto, por consiguiente, la convergencia de $x_1(t_k) = x_2(t_k)$ a 1 y de $\varphi_k(x_1, x_2)$, y $\theta_k(x_1, x_2)$ a 2.

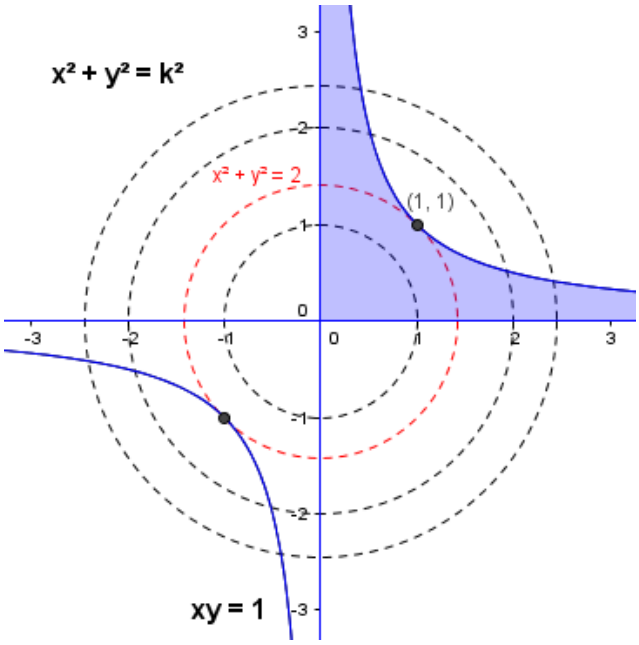

Figura 3.1: Solución al problema de minimización.

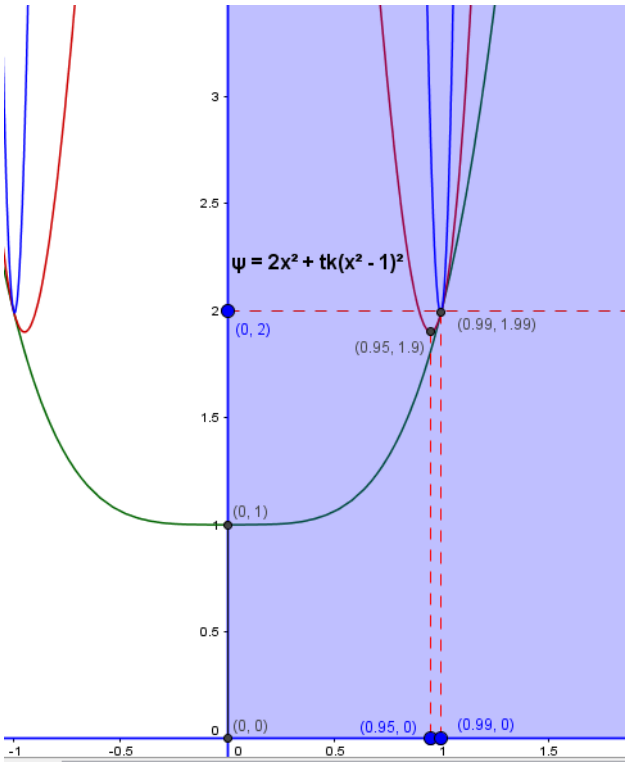

Figura 3.2: Funciones auxiliares para distintos valores del parámetro t_k.

La figura 3.2 representa la sucesión de funciones auxiliares para distintos valores del parámetro t_k. Como la función auxiliar es simétrica, sólo se tienen en cuenta los valores del primer cuadrante, sombreado en azul. La tendencia, efectivamente, es alcanzar el mínimo $\varphi(1) = 2$.

3.1.3. Implementación de MPEFC mediante HSA

Las ecuaciones [3.6] y [3.8] se pueden escribir, respectivamente, de la siguiente forma:

$$\begin{cases} funstr = \text{'}2 * (x1\text{\^{}}2) + tk * ((x1\text{\^{}}2 - 1)\text{\^{}}2)\text{'} \\ \quad tk = 1; 10; 50; 100; 1000; 10000; \dots; \infty \end{cases} \quad [3.11]$$

$$\begin{cases} x1(1) = x2(1) = 0 \\ x1(tk) = x2(tk) = +sqrt[(tk-1)/tk] \\ tk = 1;\ 10; 50; 100; 1000; 10000; \dots;\ \infty \end{cases} \quad [3.12]$$

Para resolver el problema mediante **HSA**, habrá que cambiar convenientemente el código fuente (Yang, 2010) por el de la figura 3.3:

```
%
=============================================================
%
% Files of the Matlab programs included in the book:      %
% Xin-She Yang, Nature-Inspired Metaheuristic Algorithms %
% Second Edition, Luniver Press, (2010).   www.luniver.com %
%
=============================================================
%

% Harmony Search (Simple Demo) Matlab Program
        % Written by X S Yang (Cambridge University)
% Usage: hs_simple1
% or    hs_simple1 (`x^2+(y-5)^2',25000);
function [solution,fbest]=hs_simple1(funstr,MaxAttempt)
help hs_simple1.m
disp('It may take a few minutes ...');
% MaxAttempt=25000;  % Max number of Attempt
if nargin<2, MaxAttempt=25000; end
if nargin<1,
% Quadraticpenalty1 function with the
% global fmin=2 at (1,1).
funstr = '2*(x^2)+((x^2)-1)^2';
end
% Converting to an inline function
```

```
f=vectorize(inline(funstr));
ndim=1;  %Number of independent variables
% The range of the objective function
range(1,:)=[0 3];
% Pitch range for pitch adjusting
pa_range=[200 200];
% Initial parameter setting
HS_size=20;       %Length of solution vector
HMacceptRate=0.95; %HM Accepting Rate
PArate=0.7;       %Pitch Adjusting rate
% Generating Initial Solution Vector
for i=1:HS_size,
  for j=1:ndim,
  x(j)=range(j,1)+(range(j,2)-range(j,1))*rand;
  end
  HM(i, :) = x;
  HMbest(i) = f(x(1));
end %% for i
% Starting the Harmony Search
for count = 1:MaxAttempt,
 for j = 1:ndim,
   if (rand >= HMacceptRate)
     % New Search via Randomization
     x(j)=range(j,1)+(range(j,2)-range(j,1))*rand;
   else
     % Harmony Memory Accepting Rate
     x(j) = HM(fix(HS_size*rand)+1,j);
     if (rand <= PArate)
     % Pitch Adjusting in a given range
     pa=(range(j,2)-range(j,1))/pa_range(j);
     x(j)= x(j)+pa*(rand-0.5);
     end
   end
 end %% for j
 % Evaluate the new solution
 fbest = f(x(1));
 % Find the best in the HS solution vector
 HSmaxNum = 1; HSminNum=1;
 HSmax = HMbest(1); HSmin=HMbest(1);
 for i = 2:HS_size,
    if HMbest(i) > HSmax,
      HSmaxNum = i;
      HSmax = HMbest(i);
    end
    if HMbest(i)<HSmin,
       HSminNum=i;
       HSmin=HMbest(i);
    end
 end
 % Updating the current solution if better
 if fbest < HSmax,
    HM(HSmaxNum, :) = x;
    HMbest(HSmaxNum) = fbest;
 end
 solution=x;  % Record the solution
end %% (end of harmony search).
```

*Figura 3.3: Código para la resolución por **MPEFC** mediante **HSA**.*

$\mathbf{x_1(1)}$	$\mathbf{x_2(1)}$	$\boldsymbol{\varphi_1(x_1(1), x_2(1))}$	$\boldsymbol{\theta_1(x_1(1), x_2(1))}$
5.2304×10^{-5}	5.2304×10^{-5}	$749.41004 \times 10^{-10}$	54.71416×10^{-10}
1.7616×10^{-5}	1.7616×10^{-5}	10.63005×10^{-10}	6.20646×10^{-5}
6.6433×10^{-5}	6.6433×10^{-5}	$1949.94093 \times 10^{-10}$	88.26686×10^{-5}
3.2890×10^{-5}	3.2890×10^{-5}	$118.01875 \times 10^{-10}$	21.63504×10^{-5}
4.8449×10^{-5}	4.8449×10^{-5}	$551.98435 \times 10^{-10}$	46.94611×10^{-5}
5.1305×10^{-5}	5.1305×10^{-5}	$693.84927 \times 10^{-10}$	52.64406×10^{-5}
3.2754×10^{-5}	3.2754×10^{-5}	$116.09524 \times 10^{-10}$	21.45649×10^{-5}
4.6122×10^{-5}	4.6122×10^{-5}	$453.51451 \times 10^{-10}$	42.54477×10^{-5}
$\boldsymbol{\overline{x_1(1)} = 4.34812\times 10^{-5} \approx 0}$	$\boldsymbol{\overline{x_2(1)} = 4.34812\times 10^{-5} \approx 0}$	$\boldsymbol{\overline{\varphi_1(x_1(1), x_2(1))} = 580.43039 \times 10^{-10} \approx 0}$	$\boldsymbol{\overline{\theta_1(x_1(1), x_2(1))} = 41.8017 \times 10^{-5} \approx 0}$

*Tabla 3.2: Optimización por **MPEFC** y **HSA** ($t_k = 1$).*

$\mathbf{x_1(10)}$	$\mathbf{x_2(10)}$	$\boldsymbol{\varphi_1(x_1(10), x_2(10))}$	$\boldsymbol{\theta_1(x_1(10), x_2(10))}$
0.95070	0.9507	1.90006	1.80526
0.94946	0.94946	1.90001	1.80294
0.94598	0.94598	1.90025	1.78975
0.94966	0.94966	1.90002	1.80370
0.94868	0.94868	1.89999	1.79998
0.94185	0.94185	1.90166	1.77416
0.95533	0.95533	1.90160	1.82531
0.95540	0.95540	1.90162	1.82557
$\boldsymbol{\overline{x_1(10)} = 0.94963}$	$\boldsymbol{\overline{x_2(10)} = 0.94963}$	$\boldsymbol{\overline{\varphi_1(x_1(10), x_2(10))} = 1.90065}$	$\boldsymbol{\overline{\theta_1(x_1(10), x_2(10))} = 1.80333}$

*Tabla 3.3: Optimización por **MPEFC** y **HSA** ($t_k = 10$).*

$\mathbf{x_1(50)}$	$\mathbf{x_2(50)}$	$\boldsymbol{\varphi_1(x_1(50), x_2(50))}$	$\boldsymbol{\theta_1(x_1(50), x_2(50))}$
0.99395	0.99395	1.983146	1.97587
0.99579	0.99579	1.98671	1.98319
0.99504	0.99504	1.98509	1.98020
0.99678	0.99678	1.98920	1.98714
0.98292	0.98292	1.98961	1.93226
0.99081	0.99081	1.98013	1.96340
0.99060	0.99060	1.98007	1.96257
0.98995	0.98995	1.97999	1.96000
$\mathbf{\overline{x_1(50)} = 0.99196}$	$\mathbf{\overline{x_2(50)} = 0.99196}$	$\mathbf{\overline{\varphi_1(x_1(50), x_2(50))} = 1.98428}$	$\mathbf{\overline{\theta_1(x_1(50), x_2(50))} = 1.96807}$

*Tabla 3.4: Optimización por **MPEFC** y **HSA** ($t_k = 50$).*

$\mathbf{x_1(10^2)}$	$\mathbf{x_2(10^2)}$	$\boldsymbol{\varphi_1(x_1(10^2), x_2(10^2))}$	$\boldsymbol{\theta_1(x_1(10^2), x_2(10^2))}$
0.99408	0.99408	1.99032	1.97639
0.99167	0.99167	1.99433	1.96681
0.99888	0.99888	1.99602	1.99552
0.99499	0.99499	1.99000	1.98001
0.98879	0.98879	2.00511	1.95541
0.99690	0.99690	1.99144	1.98761
0-98824	0-98824	2.00790	1.95323
0.99619	0.99619	1.99056	1.98478
$\mathbf{\overline{x_1(10^2)} = 0{,}99371}$	$\mathbf{\overline{x_2(10^2)} = 0.99371}$	$\mathbf{\overline{\varphi_1(x_1(10^2), x_2(10^2))} = 1.99571}$	$\mathbf{\overline{\theta_1(x_1(10^2), x_2(10^2))} = 1.97497}$

*Tabla 3.5: Optimización por **MPEFC** y **HSA** ($t_k = 100$).*

$\mathbf{x_1(10^3)}$	$\mathbf{x_2(10^3)}$	$\boldsymbol{\varphi_1(x_1(10^3), x_2(10^3))}$	$\boldsymbol{\theta_1(x_1(10^3), x_2(10^3))}$
0.99950	0.99950	1.99899	1.99800
0.99900	0.99900	1.99999	1.99600
0.99976	0.99976	1.99927	1.99904
0.99983	0.99983	1.99943	1.99932
0.99919	0.99919	1.99938	1.99676
0.99927	0.99927	1.99708	1.99708
0-99934	0-99934	1.99910	1.99736
0.99958	0.99958	1.99902	1.99832
$\boldsymbol{\overline{x_1(10^3)} = 0.99943}$	$\boldsymbol{\overline{x_2(10^3)} = 0.99943}$	$\boldsymbol{\overline{\varphi_1(x_1(10^3), x_2(10^3))} = 1.99903}$	$\boldsymbol{\overline{\theta_1(x_1(10^3), x_2(10^3))} = 1.99773}$

*Tabla 3.6: Optimización por **MPEFC** y **HSA** ($t_k = 1000$).*

En el límite, resulta:

$\boldsymbol{\overline{x_1(\infty)}}$	$\boldsymbol{\overline{x_2(\infty)}}$	$\boldsymbol{\overline{\varphi_1(x_1(\infty), x_2(\infty))}}$	$\boldsymbol{\overline{\theta_1(x_1(\infty), x_2(\infty))}}$
1.00000	1.00000	2.00000	2.00000

*Tabla 3.7: Optimización por **MPEFC** y **HSA** ($t_k = \infty$).*

Es decir, el problema, que inicialmente era bidimensional, se ha transformado en una sucesión de problemas sin restricciones de una variable (x) y con valores discretos del parámetro t_k. Arriba se han tabulado algunos resultados en función de éste. Como se ve, dichos resultados son muy parecidos a los obtenidos en la tabla 3.1, lo que muestra la eficacia de **HSA** para resolver el problema de optimización por **MPEFC**.

3.2. Método SUMT

A consecuencia de las dificultades asociadas a los valores elevados del parámetro de penalización, los algoritmos que emplean funciones de penalidad recurren a una sucesión creciente de parámetros de penalización. Para cada nuevo valor del parámetro se usa una técnica de minimización sin restricciones, siendo el punto de partida el obtenido como solución óptima del problema correspondiente al anterior valor del parámetro. Este procedimiento se denomina SUMT (Sequential Unconstrained Minimization Technique) y su esquema de procedimiento es el siguiente (Bazaraa et al., 2006; Ramos Méndez, 2017):

Paso de inicialización. Sea $\epsilon > 0$ un escalar para finalizar. Seleccionar un punto inicial x_1, un valor inicial del parámetro de penalización $\mu_1 > 0$ y un escalar $\beta > 0$. Hacer $k = 1$ e ir al paso principal.

Paso principal. (1) Partiendo de x_k, resolver el problema:

$$\begin{cases} Minimizar\ f(x) + \mu_k \alpha(x) \\ Sujeto\ a: \\ x \in X \end{cases}$$

Sea x_{k+1} una solución óptima del problema anterior. Ir a (2).

(2) Si $\mu_k \alpha(x_{k+1}) < \epsilon$, entonces FIN. En otro caso, hacer $\mu_{k+1} = \beta \mu_k$; reemplazar k por $k + 1$ y repetir el paso principal.

3.2.1. Ejemplo de optimización por el método SUMT

Considérese el problema:

$$\begin{cases} Minimizar\ (x_1 - 2)^4 + (x_1 - 2x_2)^2 \\ Sujeto\ a: \\ x_1^2 - x_2 = 0 \\ x \in X = R^2 \end{cases}$$

Al emplear la función de penalidad pérdida cuadrática, el problema penalizado para μ_k en la iteración k es:

$$Minimizar\ (x_1 - 2)^4 + (x_1 - 2x_2)^2 + \mu_k(x_1^2 - x_2)^2 \qquad [3.13]$$

Se toma $x_1 = (2.0, 1.0)$ como punto de partida, en el cual la función objetivo vale cero. También se toma $\mu_1 = 0.1$ y $\beta = 10$. La tabla siguiente muestra los resultados:

K	μ_k	$x_{k+1} = x_{\mu_k}$	$f(x_{k+1})$	$\alpha(x_{\mu_k}) = h^2(x_{\mu_k})$	$\theta(\mu_k)$	$\mu_k\alpha(x_{\mu_k})$
1	0.1	(1.4539, 0.7608)	0.0935	1.8307	0.2766	0.1831
2	1.0	(1.1687, 0.7407)	0.5753	0.3908	0.9661	0.3908
3	10.0	(0.9906, 0.8425)	1'5203	0.01926	1.7129	0.1926
4	100.0	(0.9507, 0.8875)	1,8917	0.000267	1.9184	0.0267
5	1000.0	(0.9461094, 0.8934414)	1.9405	0.0000028	1.9433	0.0028

Tabla 3.8: Cálculos para la optimización por el método SUMT.

3.2.2. Implementación del método SUMT por HSA

Basta con introducir los cambios adecuados en [3.13] y en el código fuente:

$$\begin{cases} help\ hs\_simple2.m \\ funstr = \text{'}(x1 - 2)\text{^}4 + (x1 - 2 * x2)\text{^}2 + tk * (x1\text{^}2 - x2)\text{^}2\text{'} \\ tk = 0.1; 1.0; 10.0; 100.0; 1000.0 \\ range(1,:) = [0.9\ \ 1.5] \\ range(2,:) = [0.7\ 0.9] \end{cases} \qquad [3.14]$$

En la primera línea aparece el nuevo nombre del archivo. En ésta y en la segunda, se ha sustituido el carácter μ por t, ya que el programa no reconoce el primero de ellos. Finalmente, en la tercera y cuarta líneas aparecen los nuevos rangos de las dos variables.

Se procede ahora de forma análoga a como se hizo en el apartado 3.2.2 y se agrupan los resultados en la tabla 3.9; puede comprobarse inmediatamente la similitud con los de la tabla 3.8.

$x_1(0.1)$	*1.45317*	*1.45248*	*1.45388*	*1.45532*	*1.45515*	*1.45322*	*1.45386*	*1.45387*	$\overline{x_1(0.1)}=$ ***1.45386***
$x_2(0.1)$	*0.76076*	*0.76076*	*0.76070*	*0.76076*	*0.76076*	*0.76115*	*0.76083*	*0.76100*	$\overline{x_2(0.1)}=$ ***0.76084***
$x_1(1)$	*1.16872*	*1.16872*	*1.16918*	*1.17000*	*1.16850*	*1.16753*	*1.16746*	*1.16998*	$\overline{x_1(1)}=$ ***1.16876***
$x_2(1)$	*0.74032*	*0.74034*	*0.74067*	*0.74067*	*0.74027*	*0.74045*	*0.74028*	*0.74067*	$\overline{x_2(1)}=$ ***0.74045***
$x_1(10)$	*0.99210*	*0.99061*	*0.99062*	*0.99095*	*0.98936*	*0.98970*	*0.99056*	*0.99084*	$\overline{x_1(10)}=$ ***0.99059***
$x_2(10)$	*0.84246*	*0.84274*	*0.84248*	*0.84246*	*0.84246*	*084232*	*0.84221*	*0.84261*	$\overline{x_2(10)}=$ ***0.84246***
$x_1(100)$	*0.95075*	*0.94998*	*0.94946*	*0.95076*	*0.94987*	*0.95175*	*0.95077*	*0.95072*	$\overline{x_1(100)}=$ ***0.95050***
$x_2(100)$	*0.88704*	*0.88730*	*0.88747*	*0.88793*	*0.88746*	*0.88774*	*0.88787*	*0.88710*	$\overline{x_2(100)}=$ ***0.88748***
$x_1(1000)$	*0.94746*	*0.94647*	*0.94610*	*0.94580*	*0.94607*	*0.94612*	*0.94589*	*0.94667*	$\overline{x_1(1000)}=$ ***0.94632***
$x_2(1000)$	*0.89293*	*0.89385*	*0.89360*	*0.89393*	*0.89382*	*0.89347*	*0.89323*	*0.89257*	$\overline{x_2(1000)}=$ ***0.89342***

*Tabla 3.9: Resultados de la optimización por el método SUMT mediante **HSA**.*

3.3. Otros ejemplos de optimización por MPEFC

El método de penalización exterior con función cuadrática se puede aplicar a problemas más complicados, pero es posible simplificarlos si se penalizan convenientemente y se efectúa un oportuno cambio de ejes coordenados; en

consecuencia, se obtiene un problema unidimensional al que es posible aplicar de forma sencilla e inmediata **HSA**. Véanse a continuación un par de ejemplos.

Ejemplo 3.3.1. Sea el problema:

$$\begin{cases} Minimizar\ f(x_1, x_2) = 2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2 \\ Sujeto\ a: \\ g_1(x_1, x_2) = x_1 + x_2 - 2 \leq 0 \\ g_2(x_1, x_2) = x_1 + 5x_2 - 5 \leq 0 \\ g_3(x_1, x_2) = -x_1 \leq 0 \\ g_4(x_1, x_2) = -x_2 \leq 0 \end{cases}$$

Este problema se puede resolver por el método de Zoutendijk con restricciones lineales de desigualdad (Bazaraa et al., 2006; Ramos Méndez, 2017), dando como resultado $(x_1, x_2)_{min} = \left(\frac{35}{31}, \frac{24}{31}\right), f((x_1, x_2)_{min}) = -7.16.$

Se resolverá ahora mediante **MPEFC**. Las expresiones anteriores equivalen a:

$$\begin{cases} Minimizar\ f(x_1, x_2) = 2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2 \\ Sujeto\ a: \\ h_1(x_1, x_2) = -x_1 - x_2 + 2 \geq 0 \\ h_2(x_1, x_2) = -x_1 - 5x_2 + 5 \geq 0 \\ h_3(x_1, x_2) = x_1 \geq 0 \\ h_4(x_1, x_2) = x_2 \geq 0 \end{cases}$$

Para restricciones del tipo $\geq$, se tiene:

$$\min\left(0, h_i(x_1, x_2)\right)^2 = \left(\frac{h_i(x_1, x_2) - |h_i(x_1, x_2)|}{2}\right)^2$$

$$= \left(\frac{h_i(x_1, x_2) - h_i(x_1, x_2)}{2}\right)^2 = 0, (i = 1,2,3,4)$$

El problema penalizado será:

$$\varphi_k(x_1, x_2) = 2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2$$
$$+ \sum_{i=1}^{4} t_k min(0, h_i(x_1, x_2))^2 = 2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2$$

Véase en la figura 3.4 un esbozo de la función a minimizar, la región factible (sombreada en verde) y la solución del problema (punto F).

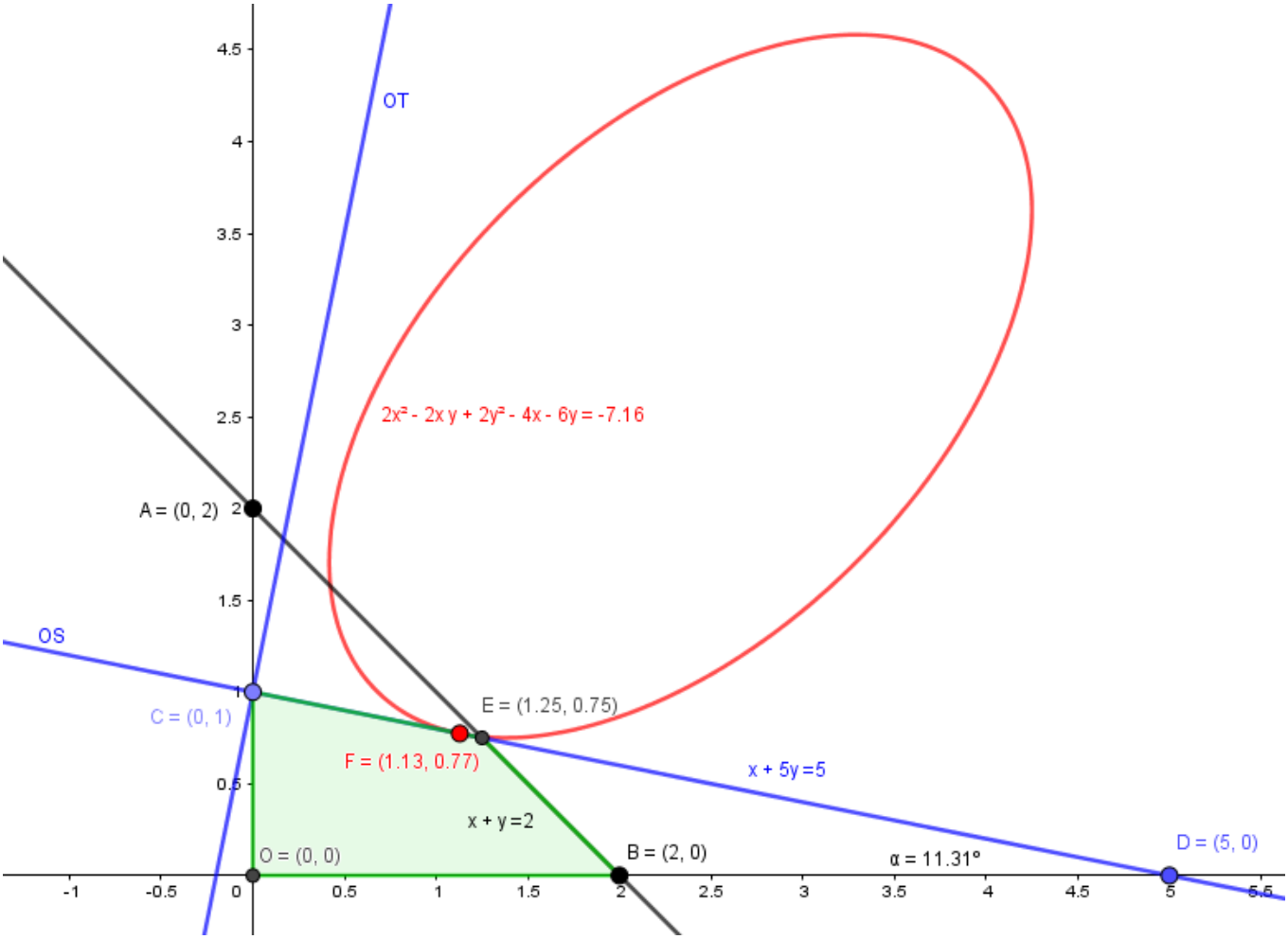

Figura 3.4: Resolución del ejemplo 3.3.1.

El punto $F(1.13, 0.77) \approx \left(\frac{35}{31}, \frac{24}{31}\right) = (x_1, x_2)_{min}$. Corresponde a la intersección de la elipse $2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2 = -7.16$ con la frontera más cercana de la región factible, que no es sino la recta $x_1 + 5x_2 = 5$ (en los demás puntos de la región

factible, la función objetivo arroja un valor superior a -7.16, de ahí que sea precisamente $F(1.13, 0.77)$ el punto mínimo buscado).

Para poder aplicar **MPEFC** y **HSA**, habrá que hacer un cambio previo de ejes coordenados: aquel cuyas componentes horizontales recaigan sobre la recta $x_1 + 5x_2 = 5$ y las componentes verticales sobre la perpendicular a ésta por el punto $C(0, 1)$, que en el esquema aparecen en color azul.

Después de efectuar sencillos cálculos trigonométricos, se obtienen las coordenadas de cualquier punto $P(x_1, x_2)$ del primer cuadrante en el nuevo sistema de referncia (CS, CT). En efecto, puesto que $\alpha = arctag\frac{1}{5} = 11.31^o$, se desprende que la nueva componente vertical será:

$$t = (x_2 - \tfrac{5-x_1}{5})cos\alpha \qquad [3.15]$$

Análogamente, la nueva componente horizontal será:

$$s = \sqrt{26} - \left(\tfrac{5-x_1}{cos\alpha} + x_2 - \tfrac{5-x_1}{5}\right) sen\alpha \qquad [3.16]$$

Operando en [3.15] y [3.16], se obtiene:

$$\begin{cases} s \approx 0.2 + 0.98x_1 - 0.2x_2 \\ t \approx 0.2x_1 + 0.98x_2 - 0.98 \end{cases} \qquad [3.17]$$

O, equivalentemente:

$$\begin{cases} x_1 \approx 0.98s - 0.2t \\ x_2 \approx -0.2s + 0.98t + 1 \end{cases} \qquad [3.18]$$

Ahora bien, puesto que el punto mínimo buscado se encuentra en el eje CS, la componente vertical será $t = 0$. De modo que las ecuaciones [3.18] se simplifican como sigue:

$$\begin{cases} x_1 \approx 0.98s \\ x_2 \approx -0.2s + 1 \end{cases} \qquad [3.19]$$

Sustituyendo [3.19] en $\varphi_k(x_1, x_2) = 2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2$, se obtiene: $\varphi_k(x_1, x_2) = 2.3128s^2 - 5.4s - 4$. Ahora se puede aplicar fácilmente **HSA** sin más que introducir en el código fuente de la figura 3.3 los cambios:

$$\begin{cases} help\ hs\_simple3.m \\ funstr = '2.3128 * (s\text{^}2) - 5.4 * s - 4' \\ range(1,:) = [0\ \ 5] \end{cases} \quad [3.20]$$

Se procede de modo similar a como se hizo en los ejemplos anteriores, tomando ocho muestras y hallando las medias. Se obtiene $\bar{s} = 1.1587$. Al sustituir este valor en las ecuaciones [3.19], se llega a:

$$(x_1, x_2)_{min} = (1.1355, 0.76826) \approx F(1.13, 0.77) \approx (\frac{35}{31}, \frac{24}{31})$$

Lo que prueba la precisión de los cálculos efectuados por **HSA**.

Ejemplo 3.3.2. Sea el problema:

$$\begin{cases} Minimizar\ f(x_1, x_2) = 2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2 \\ Sujeto\ a: \\ g_1(x_1, x_2) = x_1 + 5x_2 - 5 \leq 0 \\ g_2(x_1, x_2) = 2x_1^2 - x_2 \leq 0 \\ g_3(x_1, x_2) = -x_1 \leq 0 \\ g_4(x_1, x_2) = -x_2 \leq 0 \end{cases}$$

El problema se puede resolver por el método de Zoutendijk con restricciones no lineales de desigualdad o por el método de Topkis-Veinot (Bazaraa et al., 2006; Ramos Méndez, 2017). En ambos casos, se obtiene como resultado $(x_1, x_2)_{min} = (0.658872, 0.868226), f((x_1, x_2)_{min}) = -6.5590$.

Se resolverá ahora el problema mediante **MPEFC**. Las expresiones anteriores equivalen a:

$$\begin{cases} Minimizar\ f(x_1, x_2) = 2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2 \\ Sujeto\ a: \\ h_1(x_1, x_2) = -x_1 - 5x_2 + 5 \geq 0 \\ h_2(x_1, x_2) = -2x_1^2 + x_2 \geq 0 \\ h_3(x_1, x_2) = x_1 \geq 0 \\ h_4(x_1, x_2) = x_2 \geq 0 \end{cases}$$

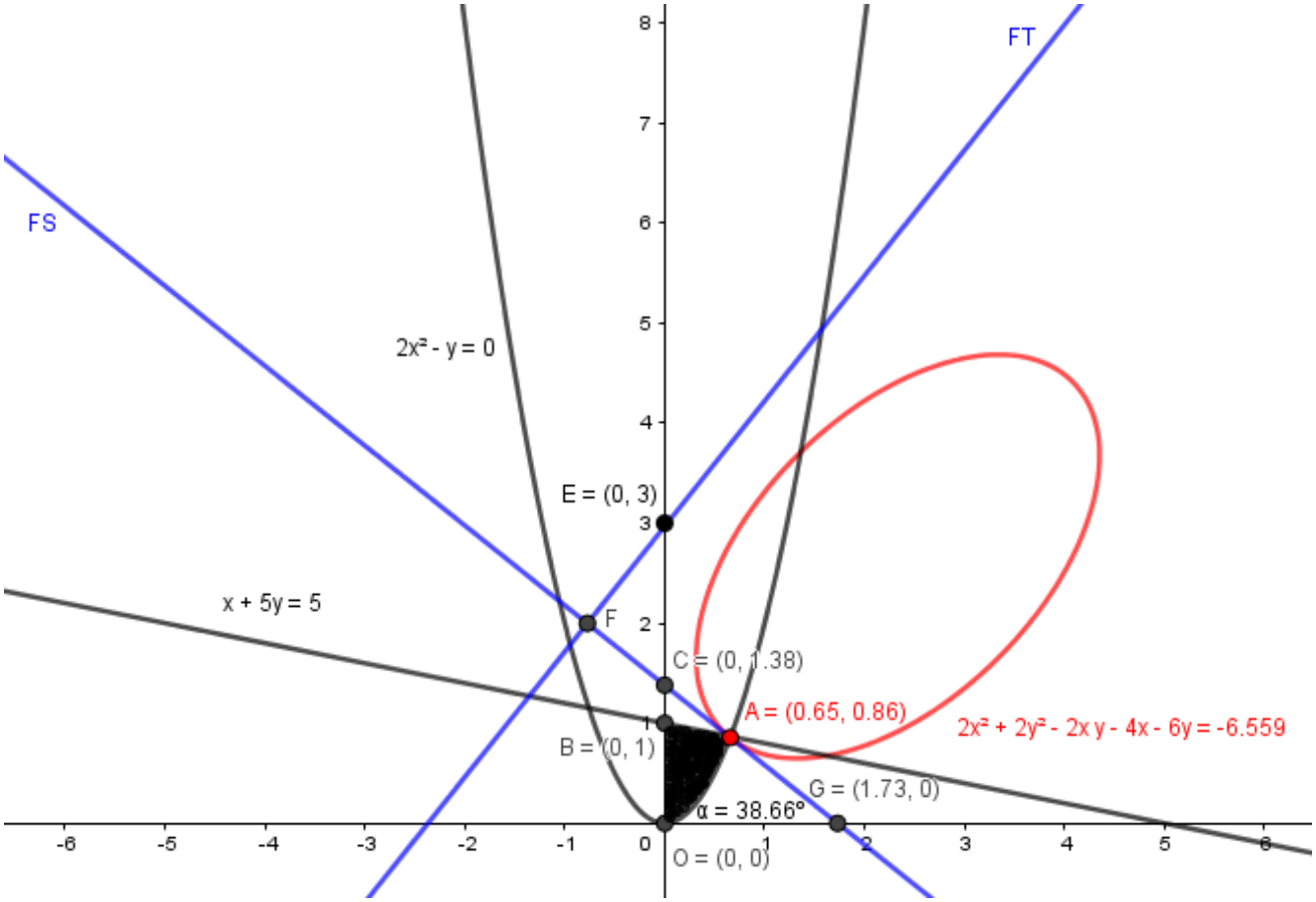

Figura 3.4: Resolución del ejemplo 3.3.1.

Igual que en el ejemplo 3.3.1, y puesto que todas las restricciones son del tipo ≥ 0, el problema penalizado es:

$$\varphi_k(x_1, x_2) = 2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2$$

La figura 3.4 muestra un esquema de la función a minimizar, la región factible (sombreada en negro) y la solución del problema (punto A).

El punto $A(0.65, 0.86) = (x_1, x_2)_{min}$ es la intersección de la elipse $2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2 = -6.5590$ con la frontera más cercana de la región factible (en los demás puntos de la región factible, la función objetivo arroja un valor superior a -6.5590, de ahí que sea precisamente A el punto mínimo buscado).

Para poder aplicar **HSA**, se hace un cambio de ejes coordenados (en la figura, en color azul), correspondiente a la tangente a la elipse $2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2 = -6.5590$ en el punto A (eje FS) y su perpendicular por el punto E (eje FT).

Tras efectuar algunos cálculos trigonométricos, se obtienen las coordenadas de cualquier punto $P(x_1, x_2)$ del primer cuadrante en el nuevo sistema de referncia (FS, FT). En efecto, puesto que $\alpha = arctag\frac{1.38}{1.73} = 38.66^o$, la nueva componente vertical será:

$$t = [x_2 + (x_1 - 1.73)tag\alpha]cos\alpha \qquad [3.21]$$

Y la horizontal:

$$s = \frac{x_1}{cos\alpha} - [t - (3 - 1.38)]tag\alpha = \frac{x_1}{cos\alpha} - \{[x_2 + (x_1 - 1.73)tag\alpha]cos\alpha - 1.62\}tag\alpha \qquad [3.22]$$

Operando en [3.21] y [3.22], se obtiene:

$$\begin{cases} s \approx 1.1872 + 0.8x_1 - 0.62x_2 \\ t \approx 0.62x_1 + 0.78x_2 - 1.08 \end{cases} \qquad [3.23]$$

O, equivalentemente:

$$\begin{cases} x_1 \approx 0.78s + 0.62t - 0.79 \\ x_2 \approx -0.62s + 0.8t + 2.02 \end{cases} \qquad [3.24]$$

Como el punto mínimo ha de encontrarse en el eje FS, la componente vertical será $t = 0$. De modo que las ecuaciones [3.24] se reducen a:

$$\begin{cases} x_1 \approx 0.78s - 0.79 \\ x_2 \approx -0.62s + 2.02 \end{cases} \qquad [3.25]$$

Al sustituir [3.25] en $\varphi_k(x_1, x_2) = 2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2$, se obtiene: $\varphi_k(x_1, x_2) = 2.95s^2 - 11s - 4.36$, con lo que se puede aplicar fácilmente **HSA** sin más que introducir en el código fuente de la figura 3.3 los cambios:

$$\begin{cases} help\ hs\_simple4.m \\ funstr = '2.95 * (s\text{^}2) - 11 * s - 4.36' \\ range(1,:) = [0\ \ 5] \end{cases}$$

Procediendo del modo habitual, se toman ocho muestras, siendo la media $\bar{s} = 1.8666$, que al sustituirla en [3.25] conduce a $(x_1, x_2)_{min} = (0.665948, 0.862708) \approx A(0.65, 0.86)$, que a su vez es muy próximo a $(x_1, x_2)_{min} = (0.658872, 0.868226)$, obtenido, como se dijo más arriba, tanto por el método de Zoutendijk con restricciones no lineales de desigualdad como por el de Topkis-Veinot. Una vez más, los cálculos efectuados mediante **HSA** han funciondo con gran precisión.

3.3.1. Resoluciones directas de los ejemplos anteriores

Los ejemplos 3.3.1 y 3.3.2 se pueden resolver directamente, sin necesidad de preparación ni análisis previos, cuando se tienen en cuenta algunas restricciones de igualdad; y no es preciso reducir el número de variables, como se muestra a continuación.

Ejemplo 3.3.1. Sea el problema:

$$\begin{cases} Minimizar\ f(x_1, x_2) = 2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2 \\ Sujeto\ a: \\ g_1(x_1, x_2) = x_1 + x_2 - 2 \leq 0 \\ g_2(x_1, x_2) = x_1 + 5x_2 - 5 = 0 \\ g_3(x_1, x_2) = -x_1 \leq 0 \\ g_4(x_1, x_2) = -x_2 \leq 0 \end{cases}$$

Equivalentemente:

$$\begin{cases} Minimizar\ f(x_1, x_2) = 2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2 \\ Sujeto\ a: \\ h_1(x_1, x_2) = -x_1 - x_2 + 2 \geq 0 \\ h_2(x_1, x_2) = x_1 + 5x_2 - 5 = 0 \\ h_3(x_1, x_2) = x_1 \geq 0 \\ h_4(x_1, x_2) = x_2 \geq 0 \end{cases}$$

Problema penalizado:

$$\varphi_k(x_1, x_2) = 2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2 + \sum_{i=1}^{4} t_k min\big(0, h_i(x_1, x_2)\big)^2 = 2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2 + t_k(x_1 + 5x_2 - 5)^2$$

Resolución mediante **HSA**:

$$\begin{cases} help\ hs\_simple100.m \\ funstr = '2 * (x1\hat{}2) + 2 * (x2\hat{}2) - 2 * x1 * x2 - 4 * x1 - 6 * x2 + tk * ((x1 + 5 * x2 - 5)\hat{}2)' \\ range(1,:) = [0\ \ 5] \\ range(2,:) = [0\ 5] \end{cases}$$

Tomando 8 muestras para cada valor de t_k y calculando las medias, se llega a:

t_k	$(\overline{x_1}, \overline{x_2})_{t_k}$
1	(1.18460, 0.86155)
10	(1.13292, 0.78334)
50	(1.13072, 0.78777)
100	(1.13039, 0.77060)
1000	(1.13002, 0.77041)

Tabla 3.10: Resolución directa del ejemplo 3.3.1

En el límite,

$$(\overline{\boldsymbol{x_1}}, \overline{\boldsymbol{x_2}})_\infty = (1.13, 0.77)$$

Ejemplo 3.3.2. Sea el problema:

$$\begin{cases} Minimizar\ f(x_1, x_2) = 2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2 \\ Sujeto\ a: \\ g_1(x_1, x_2) = x_1 + 5x_2 - 5 = 0 \\ g_2(x_1, x_2) = 2x_1^2 - x_2 = 0 \\ g_3(x_1, x_2) = -x_1 \leq 0 \\ g_4(x_1, x_2) = -x_2 \leq 0 \end{cases}$$

O bien:

$$\begin{cases} Minimizar\ f(x_1, x_2) = 2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2 \\ Sujeto\ a: \\ h_1(x_1, x_2) = x_1 + 5x_2 - 5 \geq 0 \\ h_2(x_1, x_2) = 2x_1^2 - x_2 \geq 0 \\ h_3(x_1, x_2) = x_1 \geq 0 \\ h_4(x_1, x_2) = x_2 \geq 0 \end{cases}$$

Problema penalizado:

$$\begin{aligned} \varphi_k(x_1, x_2) = {} & 2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2 \\ & + \sum_{i=1}^{4} t_k min\big(0, h_i(x_1, x_2)\big)^2 = 2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2 \\ & + t_k(x_1 + 5x_2 - 5)^2 + t_k(2x_1^2 - x_2)^2 \end{aligned}$$

Resolución mediante **HSA**:

$$\begin{cases} help\ hs\_simple200.m \\ funstr = '2 * (x1\text{^}2) + 2 * (x2\text{^}2) - 2 * x1 * x2 - 4 * x1 - 6 * x2 + tk * (x1 + 5x2 - 5)\text{^}2 + tk * ((2 * (x1\text{^}2) - x2))\text{^}2' \\ range(1,:) = [0\ \ 5] \\ range(2,:) = [0\ 5] \end{cases}$$

Igual que antes, se toman 8 muestras para cada valor de t_k y se calculando las medias, con lo que se llega a los resultados de la tabla 3.11:

t_k	$(\overline{x_1}, \overline{x_2})_{t_k}$
1	(0.78312, 0.92106)
10	(0.67576, 0.87297)
50	(0.66095, 0.86208)
100	(0.66064, 0.86880)
1000	(0.65894, 0.868291)

Tabla 3.11: Resolución directa del ejemplo 3.3.1

En el límite,

$$(\overline{x_1}, \overline{x_2})_\infty = (0.658872, 0.868226)$$

4. PRINCIPALES APLICACIONES DE HSA Y SUS HIBRIDACIONES

Los pioneros de **HSA** especificaron las aplicaciones más usuales del algoritmo (Geem, 2009-c; Manjarres, Landa Torres, Gil López, Del Ser, Bilbao, Salcedo Sanz y Geem, 2013). Desde entonces, no han dejado de aparecer trabajos sobre dichas aplicaciones, como se verá a continuación.

Ciencias de la computación:

- Clustering o agrupaciones de páginas web (Forsati y Mahdavi, 2010; Mahdavi y Abolhassani, 2009; Mahdavi, 2008). Para ello, han entrado en escena dos hibridaciones: una de **HSA** con k-means, resultando el algoritmo Hierarchical Clustering Algorithm (Forsati et al. 2008); y la otra, de Global Harmony Search Algorithm también con k-means, dando lugar a Improved Global-Best Harmony Search K-means Algorithm (Cobos et al. 2010).

- Robótica. Se ha aplicado con éxito una hibridación de **HSA** con la programación cuadrática secuencial: Hybrid Harmony Search Algorithm (Tangpattanakul, Meesomboon y Artrit, 2010).

Ingeniería eléctrica:

- Como alternativa a los métodos tradicionales de solución a los problemas de alimentación eléctrica (Fesanghary, 2009), empleando el llamado Population Variance Harmony Search Algorithm. Los resultados han sido altamente eficientes y muestran una rápida convergencia (Panigrahi, 2010).
- Foto electrónica; el híbrido resultante se llama Adaptative Harmony Search (Dong et al., 2007).
- Optimización de la onda armónica de los inversores multinivel (Majidi et al., 2008) y las redes móviles (Rong y Hanzo, 2009).

Ingeniería civil:

- Construcción de estructuras metálicas (Saka, 2009).
- Construcción de redes hidráulicas (Geem, 2006 a; Geem, 2006 b).
- Estudio de los embalses de agua, retenciones de agua para regadío, abastecimiento urbano, navegación, usos industriales, suministro de energía eléctrica, recreo, control de inundaciones (Geem, Tseng y Williams, 2009).
- Localización de aguas subterráneas empleando modelos Broyden-Fletcher-Goldfarb-Shanno Algorithm (Geem, Tseng y Williams, 2009; Kim, Geem y Kim, 2001).
- Estabilidad de los terrenos inclinados y análisis de la distribución de fuerzas internas (Cheng, 2008).

Ingeniería mecánica:

- Diseño de intercambiadores de calor de carcasas y tubos (Fesanghary, Damangir y Soleimani, 2009).
- Diseño de satélites (Geem y Hwangbo, 2006).
- Amarres de plataformas marítimas (Ryu, Duggal, Heyl y Geem, 2007).
- Cúpulas geodésicas (Saka, 2007).
- Marcos de acero (Degertekin, 2008).

Medicina:

- Minimización de los costes de determinación computacional de la estructura del ácido ribonucleico con el híbrido Harmony Search Ribonucleic Acid Fold Algorithm (Mohsen, Khader y Ramachandram, 2010).
- Clasificación de los sonidos mediante audífonos (Alexandre, Cuadra y Gil-Pita, 2009) y al de transferir gran cantidad de radiación a las células cancerosas (Panchal, 2009).

Economía:

- Problemas relacionados con la contratación gubernamental, las finanzas, la economía, etc. La variante Proposed Harmony Search de **HSA** produce excelentes resultados (Jaberipour y Khorram, 2010; Mahdavi, Fesanghary y Damangir, 2007; Li et al., 2007; Yang, 2010).

Ecología:

- **HSA** se ha modificado para estudiar el problema de maximizar el número de especies conservadas mientras se ve limitado el número de reservas naturales (Geem, Tseng y Williams, 2009; Geem y Williams, 2007; Geem, 2015).

Transporte:

- Enrutamiento de vehículos (Koumousis y Georgiou, 1994).

Composición musical:

- Piezas inspiradas en el canto gregoriano (Geem y Choi, 2007; Geem et al., 2008).

Programación de actividades colectivas:

- Se han encontrado soluciones satisfactorias a este problema (Mahdavi, 2009) y al de la planificación de cursos universitarios (Al-Betar, Khader y Gani, 2008).

Problema del agente viajero:

- **HSA** es muy eficaz en la resolución de este problema (Bouzidi y Riffi, 2014).

Problema de la mochila:

- Las variantes Simplified Binary Harmony Search (Kong, Gao, Ouyang y Li, 2015), Novel Global Harmony Search Algorithm (Zou, Gao, Li y Wu, 2011), Quantum Inspired Harmony Search Algorithm (Layeb, 2013) y Adaptative Binary Harmony Search Algorithm (Wang, Yang, Xu, Niu, Pardalos y Fei, 2013) son muy exitosas para encontrar soluciones a este problema.

Sudoku:

- El juego del Sudoku se puede ver como un problema de optimización y se puede aplicar **HSA** con éxito para resolverlo (Geem, 2007-b; Mandal y Sadhu, 2011; Mandal y Sadhu, 2013).

El puente y la antorcha:

- Se ha probado la variante Adaptative Harmony Search para resolver el problema del puente y la antorcha. Las pruebas experimentales muestran que, sin una adecuada predeterminación de los parámetros, no sería posible alcanzar el óptimo global en un tiempo aceptable (Galip y Galip, 2015).

Tetris:

- Los resultados experimentales muestran que, dada una secuencia aleatoria de 1042354 tetrominós, **HSA** encuentra la mejor solución posible en el lapso de dos semanas (Romero, Tomes y Yusiong, 2011).

Kakuro:

- Mediante la versión Self-Adaptative Harmony Search Algorithm, se obtiene una solución de este juego en poco tiempo (Panov y Koceski, 2014).

Coloración de mapas:

- La versión Modified Harmony Search Algorithm modifica el proceso de improvisación de **HSA** y reduce el tiempo de convergencia de éste (Daham, Mohammed y Mohammed, 2014-a).

CONCLUSIONES

Los algoritmos metaheurísticos desempeñan un papel de primera magnitud en el campo de la optimización. Al utilizar nuevas técnicas, como la hibridación y la simulación de procesos naturales y sociales, los algoritmos metaheurísticos han permitido corregir las debilidades de los heurísticos e identificar las características a mejorar de estos, los factores que los hacen fuertes o débiles y sus aplicaciones prácticas.

GHS, **IHS**, **NGHS** y **DHS** son desarrollos científicos destacados que han solucionado algunas deficiencias originales de **HSA** y mejorado su rendimiento y propiedades de exploración y explotación. Es de esperar que en el futuro cercano surjan nuevos procesos de hibridación con otras técnicas metaheurísticas para resolver asuntos de diversas áreas del saber; también es posible que se use **HSA** en entornos metaheurísticos para solucionar problemas discretos y continuos; o que se encuentren

reglas precisas que definan los valores de los parámetros, de acuerdo a las necesidades específicas del espacio de solución y las restricciones del problema.

En suma, **HSA** ha demostrado ser una poderosa y eficiente herramienta de optimización que no precisa de cálculos matemáticos complejos para encontrar soluciones óptimas a problemas determinados. Además, permite adaptar su estructura y sus parámetros a cada campo de aplicación.

Por otro lado, los algoritmos metaheurísticos carecen de un marco teórico que proporcione orientaciones analíticas sobre asuntos importantes: ¿Cómo mejorar la eficiencia para un problema dado? ¿Qué rango y agrupamientos de los parámetros son los mejores? ¿Qué condiciones se necesitan para garantizar una buena velocidad de convergencia? ¿Cómo solucionar problemas de optimización con objetivos múltiples?

Cualesquiera que sean los desafíos, **HSA** y sus hibridaciones se podrían aplicar a numerosos a estudios sistemáticos, marcando un camino para futuras investigaciones y proporcionando alguna guía para nuevas formulaciones algorítmicas.

En este trabajo se ha hecho una modesta -aunque novedosa- contribución al estudio de la resolución de problemas de optimización con restricciones mediante **HSA**. La primera técnica empleada es sencilla, pues los problemas iniciales conducen a una sucesión de problemas unidimensionales sin restricciones susceptibles de ser tratados mediante el algoritmo con sólo modificar unos cuantos datos del código fuente; la segunda es aún más directa y simple, pues únicamente es preciso tener en cuenta algunas restricciones de igualdad e introducir los cambios pertinentes en el código fuente, sin necesidad de efectuar análisis previos ni de reducir el número de variables. Los resultados avalan, en ambos casos, la potencia y eficacia de **HSA**.

BIBLIOGRAFÍA

1. Al-Betar, M.; Khader, A.; Gani, T. (2008): "A Harmony Search Algorithm for University Course Timetabling", *7th International Conference on the Practice and Theory of Automated Timetabling*, Montreal, Canada, pp. 12 y ss.
2. Alexandre, E.; Cuadra, L.; Gil-Pita, R. (2009): "Sound Classification in Hearing Aids by the Harmony Search Algorithm", *Music-Inspired Harmony Search Algorithm*, Berlín, Springer, pp. 173-188.
3. Bazaraa, M. S.; Sherali, H. D.; Shetty, C. M. (2006, 3ª edición): Nonlinear Programming. Theory and Algorithms, New Jersey, Wiley.

4. Blum, C.; Roli, A. (2003): “Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison”, *ACM Comput. Surv.*, 35: 268-308.
5. Bonabeau, E.; Dorigo, M.; Theraulaz, G. (1999): *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press.
6. Bouzidi, M.; Riffi, M. E. (2014): “Adaptation of the Harmony Search Algorithm to Solve the Travelling Salesman Problem”, *Journal of Theoretical and Applied Information Technology*, pp. 154-160.
7. Chakraborty, P.; Roy, G. G.; Das, S.; Jain, D. (2009): “An Improved Harmony Search Algorithm with Differential Mutation Operator”, *Fundamenta Informaticae*, 95: 1-26.
8. Cheng, Y. M. (2008): “An Improved Harmony Search Minimization Algorithm using Different Slip Surface Generation Methods for Slope Stability Analysis”, *Engineering Optimization*, 40 (2): 95-115.
9. Cobos, C. et al. (2010): “Web Document Clustering based on Global-Best Harmony Search, K-Means, Frequent Term Sets and Bayesian Information Criterion”, Barcelona, IEEEE Congress on Evolutionary Computation.
10. Daham, B. F. A.; Mohammed, M. N.; Mohammed, K. Sh. (2014-a): “Parameter Controlled Harmony Search Algorithm for Solving the Four-Color Mapping Problem”, *International Journal of Computer and Information Technology*, 3(6): 1398-1402.
11. Daham, B. F. A.; Mohammed, M. N.; Mohammed, K. Sh. (2014-b): “Modified Harmony Search Algorithm for solving the Four-Color Mapping Problem”, International Journal of Computer Applications, 91(6): 34-38.
12. Das, S. et al. (2010): “Exploratory Power of the Harmony Search Algorithm: Analysis and Improvements for Global Numerical Optimization”, IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics, 41 (1): 89-106.
13. Degertekin, S. (2008): “Optimum Design of Steel Frames using Harmony Hearch Algorithm”, *Structural and Multidisciplinary Optimization*, 36 (4): 393 – 401. Dong, H. et al. (2007): “Improved Harmony Search for Detection with Photo Density Wave”, *International Symposium on Photo electronic Detection and Imagine 2007: Related Technologies and Applications*, Beijing, SPIE.
14. Dorigo, M.; Stutzle, T. (2004): *Ant Colony Optimization*. Cambridge, MIT Press.
15. Dorigo, M.; Blum, C. (2005): “Ant Colony Optimization Theory: A Survey”, *Theor. Comput. Sci.*, 344: 243-278.
16. Fesanghary, M. (2009): “Harmony Search Applications in Mechanical, Chemical and Electrical Engineering”, *Music-Inspired Harmony Search Algorithm*, Berlín, Springer, pp. 71-86.
17. Fesanghary, M.; Damangir, E.; Soleimani, I. (2009): “Design Optimization of Shell and Tube Heat Exchangers using Global Sensitivity Analysis an Harmony Search Algorithm”, *Applied Thermal Engineering*, 29 (5-6): 1026-1031.
18. Forsati, R. et al. (2008): “Hybridization of K-Means and Harmony Search Methods for Web Page Clustering”, *Web Intelligence and Intelligent Agent Technology*, IEEE/WIC/ACM International Conference.
19. Forsati, R.; Mahdavi, M. (2010): “Web Text Mining using Harmony Search”, *Recent Advances in Harmony Search Algorithm*. Berlín, Springer, pp. 51-64.

20. Galip, E.; Galip, F. (2015): “A Performance Study for Harmony Search Algorithm”, *International Journal of Scientific Research in Information Systems and Engineering*, 1 (2).
21. Geem, Z. W.; Kim, J. H.; Loganathan, G. V. (2001): “A New Heuristic Optimization Algorithm: Harmony Search”, *Simulation*, 76: 60-68.
22. Geem, Z. W.; Lee, K.; Park, Y. (2005): “Application of harmony search to vehicle routing”, *American Journal of Applied Sciences*, 2 (12): 1552–1557.
23. Geem, Z. W. (2006-a): “Optimal Cost Design of Water Distribution Networks using Harmony Search”, *Engineering Optimization*, 38: 259-280.
24. Geem, Z. W. (2006-b): “Improved Harmony Search from Ensemble of Music Players”, *Lecture Notes in Computer Science*, 4251: 86 - 93.
25. Geem, Z. W.; Hwangbo, H. (2006): “Application of harmony search to multi-objective optimization for satellite heat pipe design”, *Proceedings of US-Korea Conference on Science, Technology, and Entrepreneurship*, Teaneck, NJ, USA.
26. Geem, Z. W. (2007-a): “Optimal Scheduling of Multiple Dam System using Harmony Search Algorithm”, *Lecture Notes in Computer Science*, 4507: 316-323.
27. Geem, Z. W. (2007-b): “Harmony Search Algorithm for solving Sudoku”, *Proceedings of the 11th International Conference, KES 2007 and XVII Italian Workshop on Neural Networks Conference on Knowledge-Based Intelligence Information and Engineering Systems, Vietri sul Mare, Italy*, Berlín, Springer, pp. 371-378.
28. Geem, Z. W.; Williams, J. C. (2007): “Harmony Search and Ecological Optimization”, *International Journal of Energy and Environment*, 1: 150-154.
29. Geem, Z. W.; Choi, J. (2007): “Music composition using harmony search algorithm”, *Lecture Notes in Computer Science*, 4448: 593.
30. Geem, Z. W. (2009-a): *Music-Inspired Harmony Search Algorithm: Theory and Applications*, Berlín, Springer.
31. Geem, Z. W. (2009-b): “Particle-Swarm Harmony Search for Water Network Design”, *Engineering Optimization*, 41: 297-311.
32. Geem, Z. W.; Tseng, C. L.; Williams, J. (2009-c): “Harmony Search Algorithms for Water and Environmental Systems”, *Music-Inspired Harmony Search Algorithm: Theory and Applications*, Berlín, Springer, pp. 113-127.
33. Geem, Z. W. (2010-a): “State-of-the-Art in the Structure of Harmony Search Algorithm”, *Recent Advances in Harmony Search Algorithm*, Berlín, Springer, pp. 1-10.
34. Geem, Z. W.; Sim, K. B. (2010-b): “Parameter-setting-free harmony search algorithm”, *Applied Mathematics and Computation*, 207 (8): 3881-3889.
35. Geem, Z. W. (2015): “Can Music Supplant Math in Environmental Planning?” *Leonardo*, Mit Press Journal, 48 (2): 147-150.
36. Glover, F.; Laguna, M. (1997): *Tabu Search*, Kluwer Academic Publishers.
37. Ingram, G,; Zhamg, T. (2009): “Overview of Applications and Developments in the Harmony Search Algorithm”, *Music-Inspired Harmony Search Algorithm: Theory and Applications*, Berlin, Springer, pp. 15, 37.

38. Jaberipour, M.; Khorram, E. (2010): "Solving the Sum-of-Ratios Problems by Harmony Search Algorithm", *Journal of Computational and Applied Mathematics*, 234 (3): 733-742.
39. Jamil, M.; Yang, X. S. (2013): "A literature survey of benchmark functions for global optimization problems", *Int. Journal of Mathematical Modelling and Numerical Optimization*, 4, (2): 150-194.
40. Karaboga, D.; Basturk, B. (2008): "On the Performance of Artificial Bee Colony (ABC) Algorithm", *Applied Soft Computing*, 8: 687-697.
41. Kennedy, J.; Eberhart, R. C. (1995): "Particle Swarm Optimization", *Proceedings of IEEE Int. Conf. Neural Networks*, pp.1942-1948.
42. Kim, J. H. Geem, Z. W.; Kim, E. S. (2001): "Parameter Estimation of the Nonlinear Muskingum Model using Harmony Search", *JAWRA Journal of the American Water Resources Association*, 37 (5): 1131-1138.
43. Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P. (1983): "Optimization by Simulated Annealing", *Science*, 220: 671-680.
44. Kong, X.; Gao, L.; Ouyang, H.; Li, S. (2015): "A simplified binary harmony search algorithm for large scale 0–1 knapsack problems", *Expert Systems with Applications*.
45. Koumousis, V.K.; Georgiou, P.G. (1994): "Genetic algorithms in discrete optimization of steel truss roofs", ASCE J. Comp. in Civil Eng., 8, 309–325.
46. Layeb, A. (2013): "A Hybrid Quantum Inspired Harmony Search Algorithm for 0–1 Optimization Problems", *Journal of Computational and Applied Mathematics*, 253: 14–25.
47. Lee, K. S.; Geem, Z. W. (2005): "A New Metaheuristic Algorithm for Continuous Engineering Optimization: Harmony Search Theory and Practice", *Comput. Methods Appl. Mech. Engrg.*, 194: 3902-3933.
48. Li, L; Chi, S. C.; Lin, G. et al. (2007): "Slope stability analysis using extremum principle by Pan Jiazheng and harmony search method", *Yantu Lixue/Rock and Soil Mechanics*, 28: 157–162 (en chino).
49. Mahdavi, M.; Fesanghary, M.; Damangir, E. (2007): "An Improved Harmony Search Algorithm for Solving Optimization Problems", *Applied Mathematics and Computation*, 188 (2): 1567-1579.
50. Mahdavi, M. et al. (2008): "Novel Metaheuristic Algorithms for Clustering Web Documents", *Applied Mathematics and Computation*, 201 (1-2): 441-451.
51. Mahdavi, M.; Abolhassani, H. (2009): "Harmony K-Means Algorithm for Document Clustering", *Data Mining and Knowledge Discovery*, 18 (3): 370-391.
52. Mahdavi, M. (2009): "Solving NP-Complete Problems by Harmony Search", *Music-Inspired Harmony Search Algorithm*, Berlín, Springer, pp. 53-70.
53. Majidi, B. et al. (2008): "Harmonic Optimization in Multi-Level Inverters using Harmony Search Algorithm", *Power and Energy Conference, 2008. PECON 2008*, IEEE 2nd International.
54. Mandal, S. N.; Sadhu, S. (2011): "An Efficient Approach to Solve Sudoku Problem by Harmony Search Algorithm", *Research Cell: An International Journal of Engineering Sciences*, 4: 312-323.
55. Mandal, S. N.; Sadhu, S. (2013): "Solution and Level Identification of Sudoku Using Harmony Search", *Modern Education and Computer Science*, 3, 49-55.

56. Manjarres, D.; Landa Torres, I.; Gil López, S.; Del Ser, J.; Bilbao, M. N.; Salcedo Sanz, S.; Geem, Z. W. (2013): "A survey on applications of the harmony search algorithm", Engineering Applications of Artificial Intelligence, 26 (8): 1818-1831.
57. Matyas, J. (1965): "Random Optimization" (en ruso), *Automation and Remote Control*, 26, 246-253.
58. Mohsen, A.; Khader, A.; Ramachandram, D. (2010): "An Optimization Algorithm based on Harmony Search for RNA Secondary Structure Prediction", *Recent Advances in Harmony Search Algorithm*, Berlín, Springer, pp. 163-174.
59. Mukhopadhyay, A.; Roy, A.; Das, S.; Abraham, A. (2008): "Population-variance and explorative power of harmony search: an analysis", Proceedings of 3rd IEEE International Conference on Digital Information Management (ICDIM 2008), pp. 13-16.
60. Omran, M. G. H.; Mahdavi, M. (2008): "Global-Best Harmony Search", *Applied Mathematics and Computation*, 198 (2): 653-656.
61. Panchal, A. (2009): "Harmony Search in Therapeutic Medical Physics", *Music-Inspired Harmony Search Algorithm*, Berlín, Springer, pp. 189-203.
62. Panigrahi, B. (2010): "Population Variance Harmony Search Algorithm to solve Optimal Power Flow with Non-Smooth Cost Function", *Recent Advances in Harmony Search Algorithm*, Berlín, Springer, pp. 65-75.
63. Panov, S.; Koceski, S. (2014): "Solving Kakuro Puzzle using Self Adapting Harmony Search Metaheuristic Algorithm", International Journal of Engineering Practical Research, 3(2): 34-39.
64. Ramos Méndez, E. (2017): *Modelización*, Madrid, UNED.
65. Romero, V. M.; Tomes, L. L.; Yusiong, J. P. T. (2011): "Tetris Agent Optimization using Harmony Search Algorithm", *International Journal of Computer Science Issues*, 8(1): 22-31.
66. Rong, Z.; Hanzo, L. (2009): "Iterative Multiuser Detection and Channel Decoding for DS-CDMA using Harmony Search", *Signal Processing Letters, IEEE*, 16 (10): 917-920.
67. Ryu, S.; Duggal, A.; Heyl, C.; Geem, Z. W. (2007): "Mooring cost optimization via harmony search", Proceedings of the 26th ASME International Conference on Offshore Mechanics and Arctic Engineering, pp. 1-8.
68. Saka, M. (2007): "Optimum geometry design of geodesic domes using harmony search algorithm", *Advances in Structural Engineering*, 10 (6): 595 – 606.
69. Saka, M. (2009): "Optimum Design of Steel Sway Frames to BS5950 using Harmony Search Algorithm", *Journal of Constructional Steel Research*, 65 (1): 36–43.
70. Srinivasa Rao, R. et al. (2011): "Optimal Network Reconfiguration of Large-Scale Distribution System using Harmony Search Algorithm", IEEE Transactions on Power Systems, 26 (3): 1080-1088).
71. Tangpattanakul, P.; Meesomboon, A.; Artrit, P. (2010): "Optimal Trajectory of Robot Manipulator using Harmony Search Algorithm", *Recent Advances in Harmony Search Algorithm*, Berlín, Springer, pp. 23-36.
72. Wang, L.; Yang, R.; Xu, Y.; Niu, Q.; Pardalos, P. M.; Fei, M. (2013): "An improved adaptive binary Harmony Search algorithm", *Information Sciences*, 232: 58–87.
73. Wolpert, D. H. y Macready, W. G. (1997): "No Free Lunch Theorems for Optimization", IEEE Transactions on Evolutionary Computation, 1 (1): 67-82.

74. Yang, X. S. (2009): "Harmony Search as a Metaheuristic Algorithm", *Music-Inspired Harmony Search Algorithm: Theory and Applications*, Berlin, Springer, vol. 191, pp. 1-14.
75. Yang, X. S. (2010): *Nature-inspired Metaheuristic Algorithms*, University of Cambridge, Luniver Press.
76. Zou, D. et al. (2010): "A Novel Global Harmony Search Algorithm for Reliability Problems", *Computers & Industrial Engineering*, 58 (2): 307-316.
77. Zou, D.; Gao, L.; Li, S.; Wu, J. (2011): "Solving 0-1 Knapsack Problem by a Novel Harmony Search Algorithm", *Applied Soft Computing*, 11: 1556-1564.