DISEÑO AUTOMÁTICO MEDIANTE TÉCNICAS EVOLUCIONISTAS EN PROBLEMAS DE OPTIMIZACIÓN AERO-HIDRODINÁMICA DE INGENIERÍA NAVAL

V. Díaz Casás, Fernando López Peña, Marcos Miguez Gonzalez, Richard J. Duro

1) Grupo Integrado de Ingeniería, Universidade da Coruña
vdiaz@udc.es , flop@udc.es , mmiguez@udc.es, richard@udc.es

1. Introducción

En este trabajo se presenta un entorno de optimización automático de aplicable a problemas de optimización hidrodinámica de ingeniería naval. Este entorno automático utiliza estrategias de resolución de problemas de optimización en ingeniería en la que la participación del diseñador se restringe a la definición del problema. El objetivo de aplicar este método es evitar introducir restricciones ficticias al problema por parte del diseñador.

La complejidad de este entorno hace necesario que sea abordado desde un punto de vista multidisciplinar que combine diferentes áreas de conocimiento:

1. Construcción naval. Para el problema de optimización es necesario establecer las características del problema y las restricciones aplicables a los modelos buscados. Además tiene que contemplar la respuesta estructural y el comportamiento dinámico del buque y su efecto sobre el comportamiento de los distintos elementos.
2. Mecánica de fluidos. En un proceso de optimización es necesario establecer una medida de calidad de cada una de las alternativas.
3. Computación evolutiva, creación de la estrategia de optimización.
4. Análisis de señales, procesado de los datos de los ensayos experimentales.

El entorno de diseño empleado en este trabajo ha sido creado con el objetivo de alcanzar resultados válidos en procesos de diseño genérico. Lo cual implica que esta implementación contiene un conjunto de bloques independientes que pueden ser sustituidos y reestructurados dependiendo del proceso concreto de diseño que se trate y de la clase de simulaciones y evaluaciones requeridas.

La estructura básica del proceso de diseño está compuesta de tres bloques:

- Bloque de búsqueda de solución, que integra los algoritmos evolucionistas. En él, dada una población y su calidad se abordan los cálculos para generar los individuos de la siguiente generación.
- Bloque de decisión. En este bloque se realiza la evaluación de los distintos individuos de forma que para cada alternativa. De esta forma puede establecerse una medida de calidad que permita seleccionar las mejores alternativas.
- Bloque de distribución computacional. El elevado coste computacional de estas estrategias exige que se realice de forma distribuida, es decir, repartir el carga total de trabajo entre distintos nodos de procesado. Esto exige la coordinación de la información requerida en cada nodo y la transmisión de información entre los mismos, así como la gestión de los diferentes módulos que intervienen en este proceso.

Si bien el bloque de distribución computacional es el controla el proceso de diseño a más bajo nivel es el que tiene que ser más flexible ya que tiene que dar soporte a los módulos de búsqueda y de decisión, los cuales pueden requerir muy diversas configuraciones. Por un lado el algoritmo de búsqueda puede optar por realizarse en un solo nodo o descomponer su búsqueda por razas y, dentro de estas, análisis por subgrupos cada una en un nodo; y, por otro lado, la evaluación de cada individuo puede realizarse en un solo nodo o en varios nodos, necesario en problemas con grandes exigencias de cálculo como el estudiado en este caso.

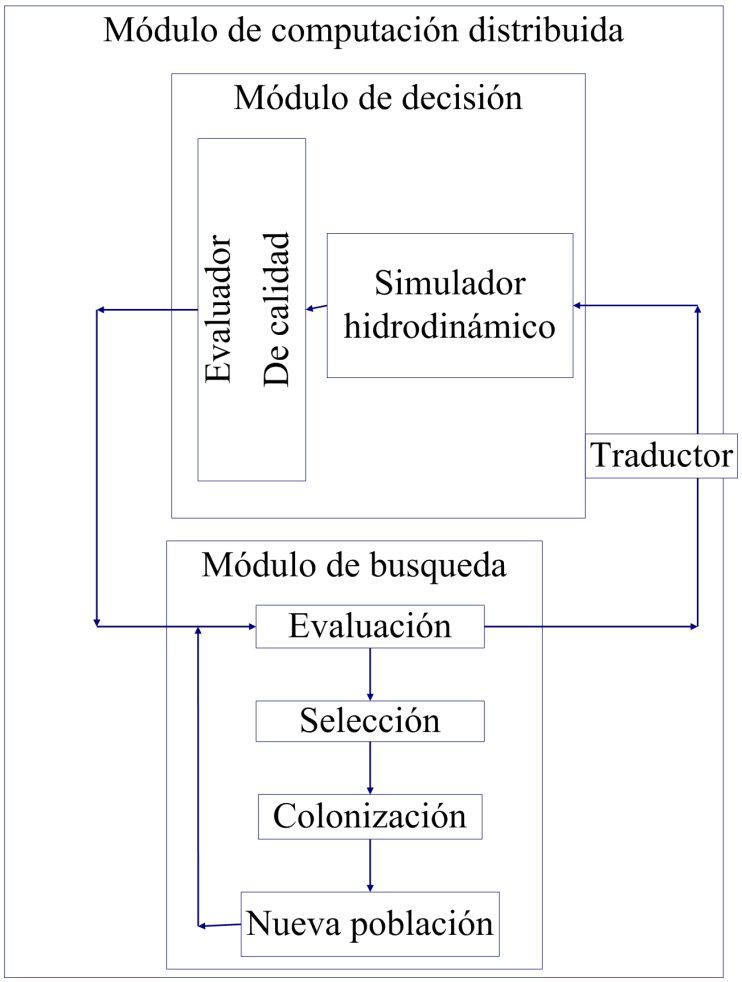


Fig. 1. Esquema del entorno de diseño

2. Algoritmo de busqueda

Los algoritmos evolutivos presentan una serie de ventajas respecto a otros métodos de búsqueda:

- Se pueden aplicar fácilmente a cualquier problema, basta idear una representación adecuada.
- Realizan la búsqueda de la solución desde múltiples puntos simultáneamente, con lo que son menos susceptibles de caer en máximos locales que métodos que realizan esta búsqueda desde un único punto.
- Son muy fácilmente paralelizables.

Estos algoritmos, cuyo concepto básico aplicado a la obtención de sistemas de control robóticos ya planteó

Norbert Wiener[i] en 1948, hacen uso de una analogía entre el proceso evolutivo en la naturaleza y la búsqueda de una solución en un espacio determinado. Su funcionamiento general se puede apreciar en la figura .

Sean:

I *espacio de búsqueda*
$F : I \to \mathbb{R}$ *la función de calidad*
μ *el tamaño de la población de los padres*
λ *el tamaño de la población de los descendientes*
$P(t) = (a_1(t), \ldots, a_\mu(t)) \in I^\mu$ *la población en la generación t*
$r : I^\mu \to I^\kappa$ *operador de recombinación*
Θ_r *parámetros que determinan el funcionamiento de r*
$m : I^\kappa \to I^\lambda$ *operador de mutación*
Θ_m *parámetros que determinan el funcionamiento de m*
$s : I^\lambda \to I^\mu$ *operador de selección*
Θ_s *parámetros que determinan el funcionamiento de s*
ι *criterio de parada*
Θ_ι *parámetros del criterio de parada*

Entonces:

$t \leftarrow 0$
$P(t) \leftarrow inicializar(\mu)$
$F(t) \leftarrow evaluar(P(t), \mu)$
mientras $(\iota(P(t), \Theta_\iota) \neq cierto)$ *hacer*
  $P'(t) \leftarrow recombinar(P(t), \Theta_r)$
  $P''(t) \leftarrow mutar(P'(t), \Theta_m)$
  $F(t) \leftarrow evaluar(P''(t), \lambda)$
  $P(t+1) \leftarrow seleccionar(P''(t), F(t), \mu, \Theta_s)$
  $t \leftarrow t+1$

Figura 2: Funcionamiento general de un algoritmo evolutivo.

Básicamente, tenemos una población inicial de individuos. Cada individuo es una posible solución al problema (un punto en el espacio de soluciones) y viene caracterizado por uno o varios cromosomas. Esta población evoluciona con el tiempo: los individuos se combinan y sufren mutaciones en los genes de sus cromosomas, de tal forma que las posibilidades que tienen de pasar, ellos o sus descendientes, a la siguiente generación son proporcionales a lo cerca que estén de la solución del problema. La forma en la que los individuos se combinan, cómo tienen lugar las mutaciones, la frecuencia de ambas formas de reproducción y sobre qué individuos se producen dan lugar a los diversos tipos de algoritmos evolutivos.

En el caso que nos ocupa, los individuos representan los parámetros que determinan la las características y forma del modelo. Para determinar lo cerca que está un individuo de la solución óptima, se corresponde con los genes de dicho individuo, y se le asigna una calidad en función de los resultados del simulador

aerodinámico. Esta calidad determina las posibilidades de reproducirse del individuo, y así se van obteniendo nuevas poblaciones de individuos que se corresponden cada vez más con la vela deseada.

Bajo esta denominación de algoritmos evolutivos se engloban todas las técnicas que tratan de simular el proceso de la evolución natural[ii]]. Los tipos principales son los algoritmos genéticos, las estrategias de evolución y la programación evolutiva, que describiremos a continuación, y que se diferencian fundamentalmente en los mecanismos de reproducción utilizados. También describiremos un nueva aproximación, basada en un ecosistema con especies, conocida como algoritmos macroevolutivos, y que ha sido la utilizada para resolver nuestro problema por sus peculiares características.

2.1. Algoritmos genéticos.

Las bases de los algoritmos genéticos fueron propuestas por John Holland a principios de la década de 1960 y la idea de los algoritmos genéticos como tales fue propuesta por él mismo en 1975[iii]. El funcionamiento de este tipo de algoritmos se puede resumir en lo siguiente.

Los valores de los genes de los individuos de la población inicial se inicializan aleatoriamente. Se calcula la calidad de los individuos de la población inicial. A partir de ese momento, el proceso evolutivo es un bucle en el cual se seleccionan los individuos a reproducirse siguiendo algún criterio relacionado con la calidad, se obtiene una población de hijos mediante el cruce y mutación de individuos, se evalúan los individuos de la nueva población, se seleccionan los individuos de la población de los padres a ser reemplazados y se funden ambas poblaciones.

Originalmente, los cromosomas estaban codificados en binario, es decir, cada gen era un 0 o un 1. Posteriormente se han utilizado otras codificaciones como, por ejemplo, codificar cada gen como un número real.

El método original de selección de los individuos a reproducirse es la selección proporcional, también conocido como selección por "ruleta". Cada individuo tiene una probabilidad de ser seleccionado para reproducirse igual a su calidad dividida por la suma de las calidades de todos los individuos. Este método, en algunas ocasiones, tiene el problema de aplicar muy poca presión selectiva. Por ejemplo, si se desea hallar el valor de x que maximiza $y=ax^2+b$ y b es muy grande, todos los individuos tendrán una calidad parecida y, por tanto, una probabilidad de reproducirse muy similar. Una forma de arreglar este problema es escalando las calidades. Sin embargo, puede existir el problema contrario. Si durante el proceso evolutivo aparece un individuo con mucha mayor calidad que el resto de los individuos, la población rápidamente convergerá hacia ese individuo, ya que su probabilidad de reproducirse es mucho mayor que la de los demás. Si ese individuo se correspondía con un máximo local, la evolución probablemente nunca llegue a la solución real.

Existen otros dos tipos de selección que pretenden arreglar este problema. Una es la selección por posición[iv] en la que los individuos son ordenados por su calidad y su probabilidad de selección es una función de su posición en esa lista ordenada. Alterando dicha función se altera la presión selectiva. El otro tipo de selección es la selección por torneo[v] , donde se escoge aleatoriamente un grupo de individuos y dentro de este grupo se selecciona para reproducirse aquel que tiene mayor calidad. Este proceso se repite hasta que se tienen todos los individuos que se necesitan para reproducirse. Cambiando el tamaño de ese grupo se altera la presión de selección: a mayor tamaño, mayor presión.

El cruce es el operador principal en los algoritmos genéticos. Consiste en escoger dos individuos, un punto de cruce, y crear dos individuos nuevos de tal forma que cada uno tenga los primeros n genes de un padre y los siguientes m-n genes del otro (siendo m la longitud total del cromosoma en genes). Existen otras variantes de cruce, usando, por ejemplo, dos o tres puntos de cruce en lugar de uno. Otra variante más consiste en que los descendientes se crean alternando los genes de los padres (cruce uniforme). La idea del operador cruce se basa en que, dados dos individuos con buena calidad pero por razones distintas (la buena calidad de los individuos viene dada por el valor de dos grupos de genes distintos en cada individuo), al cruzarse, el individuo resultante se quede con los genes que provocaban una buena calidad en ambos padres, dando

lugar a un individuo todavía mejor. Esto evidentemente no se produce siempre, sino con una cierta probabilidad, pero cuanta mayor es la calidad de un individuo más veces se reproduce y, por tanto, mayor es la probabilidad de que la parte de su genoma causante de su buena calidad se mezcle con la parte correspondiente de otro individuo, dando lugar a individuos cada vez mejores, que tendrán una mayor probabilidad de reproducirse en posteriores generaciones, guiando así el proceso evolutivo.

Si la codificación es binaria, es decir, cada gen es un 0 o un 1, la mutación consiste en cambiar el gen con una determinada probabilidad. Si la codificación es distinta se amplia la definición de forma que cada gen se cambia, también con una determinada probabilidad, por otro gen válido siguiendo una distribución uniforme. En este caso existen, sin embargo, otras posibilidades. Si los genes son números reales, es frecuente que la mutación consista en sumar o restar al valor del gen una cantidad aleatoria con una distribución distinta a la uniforme (una normal, por ejemplo). La mutación en los algoritmos genéticos es un medio de introducir variedad en la población y, por eso, la probabilidad de mutación suele ser baja. Habitualmente se escoge de tal manera que de media se produzca una mutación en cada cromosoma.

Para seleccionar los individuos a ser reemplazados también se tienen varias alternativas. Podemos reemplazar los peores individuos o reemplazar aleatoriamente. En el primer caso aceleramos el proceso evolutivo y perdemos variedad. En el segundo podemos perder, quizás para siempre, buenos individuos y frenar la convergencia o incluso impedirla.

2.2. Estrategias evolutivas.

Las estrategias evolutivas, que fueron ideadas por Rechenberg[vi] y Schwefel[vii], se diferencian principalmente de los algoritmos genéticos en que el operador que guía el proceso evolutivo es la mutación y no el cruce. De hecho, en la propuesta inicial de estrategias evolutivas no existía el operador cruce, y la generación de individuos nuevos consistía en sumar a cada individuo un vector aleatorio, cuyos valores, entre 0 y 1, seguían una distribución normal, multiplicado por un escalar mayor que 0 y comprobando si la calidad del nuevo individuo era mejor que la del individuo original. En caso afirmativo lo reemplazaba en la población y si no el individuo original pasaba a la siguiente generación.

Actualmente, se puede considerar que existen, básicamente, dos tipos de estrategias evolutivas. Las estrategias evolutivas $(\mu+\lambda)$ donde μ individuos generan λ hijos y se seleccionan para la siguiente generación los μ mejores individuos del total $\mu+\lambda$, y las estrategias evolutivas (μ,λ) donde μ individuos generan λ hijos y se seleccionan para la siguiente generación los μ mejores individuos de los λ hijos. En ambos casos, los λ hijos se generan mutando todos los genes de los cromosomas. Esa mutación consiste en sumar un valor a cada gen. El total de los valores sumados a los genes suele seguir una distribución de media cero. El valor máximo a sumar a los genes suele venir fijado por una regla o puede incluirse dentro del cromosoma como un gen más. Igualmente, dicho valor puede ser el mismo para todos los genes del cromosoma o puede haber varios valores para distintos grupos de genes.

2.3. Programación evolutiva.

La programación evolutiva, originalmente definida por Fogel[viii]], fue planteada desde un principio, a diferencia de las otras técnicas evolutivas, como una alternativa a la inteligencia artificial clásica, una forma de obtener comportamientos inteligentes sin basarse en heurísticas.

Es similar a las estrategias evolutivas, pero hace hincapié en que lo realmente importante es el fenotipo (el controlador final y, por ende, el comportamiento) y no el genotipo (la codificación del individuo en el cromosoma). Así, los cromosomas son programas en un lenguaje de alto nivel, que describen un autómata de estados finitos (como era originalmente) o una RNA. La mutación es más compleja que en los casos anteriores, y consiste en cambiar una sentencia o grupo de sentencias de ese lenguaje de alto nivel por otra sentencia o grupo de sentencias (que no tiene por qué ser de la misma longitud), comprobando que no se

generan individuos imposibles, incorrectos sintáctica o semánticamente. Por tanto, estamos hablando aquí de que, inevitablemente, los cromosomas van a ser de longitud variable.

Existe una variante, a caballo entre la programación evolutiva y los algoritmos genéticos, denominada programación genética, en la que los cromosomas son programas en un lenguaje de más bajo nivel y donde también existe el operador cruce, en el que dos programas se cruzan por un punto determinado, teniendo en cuenta que el intercambio de dos trozos de código de dos cromosomas debe resultar sintáctica y semánticamente correcto.

2.3.1. Algoritmos macroevolutivos.

Un algoritmo macroevolutivo es un tipo muy reciente de algoritmo evolutivo que fue presentado por Marín y Solé[ix] en 1999. La diferencia fundamental de esta aproximación estriba en que mientras en los otros algoritmos evolutivos una población de individuos evoluciona mediante el principio de que cuanto mejor es un individuo más probabilidades tiene de reproducirse, en los algoritmos macroevolutivos se considera una nueva escala evolutiva, la de especies, y así un ecosistema formado por un conjunto de especies evoluciona mediante la extinción de las peores y la apropiación de su nicho ecológico por parte de especies nuevas. Aunque tanto en las implementaciones originales de los autores como en la realizada en este trabajo una especie está formada por un único individuo, es importante tener esta diferencia de concepto en mente porque es la base de los algoritmos macroevolutivos y representa una puerta abierta a que cada especie esté formada por varios individuos y evolucione independientemente.

El funcionamiento de los algoritmos macroevolutivos se presenta en la figura 2. Básicamente, se calcula una matriz de conexión, en la que cada elemento mide la diferencia de calidades entres dos especies ponderándola por la distancia en el espacio de soluciones que existe entre ellas, de forma que dicha diferencia de calidad es más importante cuanto más cerca están las especies en dicho espacio. Si la suma de

Sean:

d *la dimensionalidad del espacio de búsqueda*

$f : \mathbb{R}^d \to \mathbb{R}$ *la función de calidad*

$p_i = (p_i^1, \ldots, p_i^d)$ *la especie* i

$W_{i,j}$ *la conexión entre la especie* i *y la especie* j

h_i *el coeficiente de supervivencia para la especie* i

p_b *una especie con coeficiente de supervivencia* ≥ 0

p_n *una especie generada aleatoriamente*

$\xi \in [0,1]$ *número aleatorio con distribución uniforme*

$\lambda \in [-1:1]$ *número aleatorio con distribución uniforme*

ρ *y* τ *parámetros del algoritmo*

Entonces:

$t \leftarrow 0$

Inicializar cada p_i

Evaluar cada p_i

mientras no se cumpla el criterio de parada hacer

$$\text{Calcular la matriz de conexiones } W \text{ donde } W_{i,j} = \begin{cases} \dfrac{f(p_i) - f(p_j)}{|p_i - p_j|}, & si\ |p_i - p_j| \neq 0 \\ 0, & \text{en caso contrario} \end{cases}$$

$$\text{Calcular el coeficiente de supervivencia } h_i(t) = \sum_{j=1}^{p} W_{i,j}(t)$$

$$\text{Calcular el estado } S_i(t+1) = \begin{cases} 1, & si\ h_i(t) \geq 0 \\ 0, & \text{en caso contrario} \end{cases}$$

$$\text{Colonización: } p_i(t+1) = \begin{cases} p_i(t), & si\ S_i(t+1) = 1 \\ p_b(t) + \rho\lambda(p_b(t) - p_i(t)), & si\ S_i(t+1) = 0\ y\ \xi > \tau \\ p_n, & si\ S_i(t+1) = 0\ y\ \xi \leq \tau \end{cases}$$

$t \leftarrow t+1$

Figura 3: Funcionamiento de un algoritmo macroevolutivo.

todos los elementos de la matriz de conexión en los que interviene una especie es positiva (coeficiente de supervivencia positivo, según la definición de los autores), dicha especie sobrevive. En caso contrario, se extingue. Si sobrevive se mantiene tal cual en la siguiente generación. Si se extingue es sustituida por una nueva especie colonizadora que puede ser generada aleatoriamente o puede derivarse de la especie extinguida y de otra cualquiera que sobrevive.

Hay dos parámetros que determinan el funcionamiento del algoritmo macroevolutivo: ρ, que marca la distancia máxima que una nueva especie generada a partir de una extinta y una superviviente puede tener respecto a ésta última, y τ, que controla el porcentaje de especies aleatorias que se generan en cada generación.

En definitiva, ρ determina cómo se lleva a cabo la explotación, es decir, cómo se realiza la búsqueda en las proximidades de individuos ya existentes. Cuánto más pequeño sea más cerca estarán las nuevas especies de las supervivientes y, por tanto, mayor será la presión evolutiva, puesto que más rápido se abandonarán las zonas del espacio de búsqueda con aparentemente peores especies.

Por su parte, τ controla el balance entre explotación y exploración (búsqueda de nuevo material genético, nuevos individuos no necesariamente cerca, o incluso preferentemente alejados, de los ya existentes). Si τ es muy grande se generarán muchas especies aleatoriamente y estaremos ante una evolución guiada por la exploración, lo cual es útil para buscar nuevo material genético. Si es muy pequeño se generarán pocas especies aleatoriamente y estaremos ante una evolución guiada por la explotación, lo que quiere decir que se realizará una búsqueda exhaustiva en las proximidades de los individuos ya existentes, a costa de no buscar nuevo material genético en zonas no exploradas. Lo habitual es que este parámetro actúe como una "temperatura" y decrezca con el tiempo, de forma que se empiecen generando muchos individuos aleatoriamente y, a medida que avanza el proceso evolutivo, cada vez se generen menos individuos aleatorios.

Una ventaja de los algoritmos macroevolutivos respecto a otros es que solamente hay dos parámetros que determinan su funcionamiento, con lo que ajustarlo a un problema determinado es mucho más fácil. Pensemos, por ejemplo, en los algoritmos genéticos, donde es necesario seleccionar el tipo de selección, el tipo de cruce, el tipo de mutación, la probabilidad con la que se realizan ambos operadores genéticos, el tipo de reemplazo, etc., y donde además muchos de estos factores tienen a su vez otros parámetros que ajustar.

La otra ventaja de los algoritmos macroevolutivos es su particular forma de realizar la búsqueda por el espacio de soluciones, dando la oportunidad a que se reproduzcan todas las especies antes de ser extinguidas, lo cual es muy útil en problemas propensos a caer en máximos locales. Esto se traduce en que el algoritmo, en líneas generales, tiene una menor presión evolutiva que los algoritmos genéticos o las estrategias evolutivas, ya que toda especie, antes de desaparecer, puede dejar descendencia en la población con mucha mayor probabilidad que en los otros algoritmos evolutivos. Además, al tener todas las especies supervivientes la misma probabilidad de ocupar el nicho dejado por las que se extinguen, no existe el problema del superindividuo. No sólo la presión es menor, sino que la dinámica de la evolución tiende a generar agrupamientos alrededor de las mejores especies, con lo que la explotación es también muy eficiente.

2.3.2. Comparación del comportamiento de los algoritmos macroevolutivos frente a los evolutivos

Aunque los algoritmos evolutivos son utilizados con éxito en un muy amplio rango de problemas, eso no quiere decir que obtener la solución óptima a cualquier problema sea trivial. Existen una serie de circunstancias que pueden conducir a que eso no suceda. Por ejemplo, un reducido ratio número de individuos/dimensión del espacio de búsqueda motivado por el elevado coste computacional del cálculo de calidad de los individuos, una evaluación no determinista de dicha calidad, alta epístasis, etc.

En general, a mayor presión en un algoritmo evolutivo, más rápida es la convergencia a una solución, pero mayores son también las probabilidades de que dicha solución sea subóptima. Cuán grandes son esas probabilidades dependerá de muchos factores, como el número de individuos frente a la dimensionalidad del espacio de búsqueda, la forma de la función para la cual se está tratando de encontrar el óptimo, etc. Por eso, en cada campo de aplicación de los algoritmos evolutivos podemos encontrarnos con una combinación de parámetros ideal distinta, que nos lleven a diferentes formas de equilibrio entre la exploración (buscar zonas no exploradas en el espacio de búsqueda) y la explotación (buscar mejores individuos en las cercanías de los individuos buenos ya existentes). Cuanto más costosa en tiempo es la evaluación de un individuo, menos población podremos usar y, por tanto, más crítico se vuelve el encontrar ese punto de equilibrio.

Una posibilidad para tratar de solucionar este problema es concentrar la búsqueda alrededor de varios individuos buenos, y no de uno sólo. Una forma de conseguir esto es utilizar operadores genéticos que produzcan un efecto de agrupamiento alrededor de varias soluciones candidatas. En biología este fenómeno es conocido como "nitching". Aplicado a la evolución artificial, este fenómeno significa el agrupamiento de los individuos alrededor de zonas determinadas del espacio de búsqueda, que serán aquellas que albergan posibles soluciones al problema. Como hemos visto cuando se describió el funcionamiento de los algoritmos

macroevolutivos, éstos producen precisamente este efecto, ya que las especies que se van generando a partir de las que se extinguen se van agrupando alrededor de las que sobreviven por la propia naturaleza del operador de colonización.

El siguiente ejemplo permitirá apreciar perfectamente el comportamiento de los algoritmos macroevolutivos en este tipo de problemas comparado con uno de los algoritmos evolutivos tradicionales: un algoritmo genético. Este ejemplo presenta un espacio de búsqueda muy complejo, con un muy alto grado de epístasis. Para apreciar esta afirmación, hemos seleccionado el mejor individuo de todas las evoluciones y hemos visto cómo cambiaba su calidad alterando únicamente 4 del total de 76 genes que conforman su cromosoma. En la figura 3 representamos 4 gráficas que se corresponden con 4 parejas distintas de valores para dos genes modificados. Los valores de los otros dos genes modificados son mostrados en todas las gráficas en los ejes x e y. En estas gráficas podemos apreciar varios hechos que demuestran la dificultad presente en este problema:

1. Existen grandes áreas del espacio de soluciones en las cuales todos sus puntos tienen la misma calidad, con lo que el algoritmo evolutivo no tiene información útil en esas zonas que guíe la evolución.
2. Existen puntos con elevada y similar calidad que están separados por zonas con mucha menor calidad, lo que es una doble dificultad para el algoritmo evolutivo: debe superar esas zonas con menor calidad para llegar a otras con alta calidad pero muy similar a aquella en la que ya se está, lo cual puede confundir al algoritmo evolutivo respecto a por dónde seguir.
3. Si comparamos las dos gráficas superiores, vemos que variando únicamente dos genes podemos pasar de un máximo a una solución pésima, peor que las que la rodean que se corresponden con puntos que bajo ninguna circunstancia son máximos ni tan siquiera locales.

Lo peor no es que existan problemas con espacios de búsqueda complicados, si no que, por lo general, no podemos preveer esa situación.

Estamos, por tanto, ante un problema complejo donde la epístasis es enorme: un valor para un gen puede ser óptimo o pésimo dependiendo de los valores de otros genes. Nos interesa, por tanto, usar un algoritmo en el que haya una fase de exploración importante, al menos durante una parte de la evolución, para escapar de las zonas con calidad constante, pero, al mismo tiempo, con una fase de explotación que permita buscar meticulosamente las zonas alrededor de las mejores soluciones.

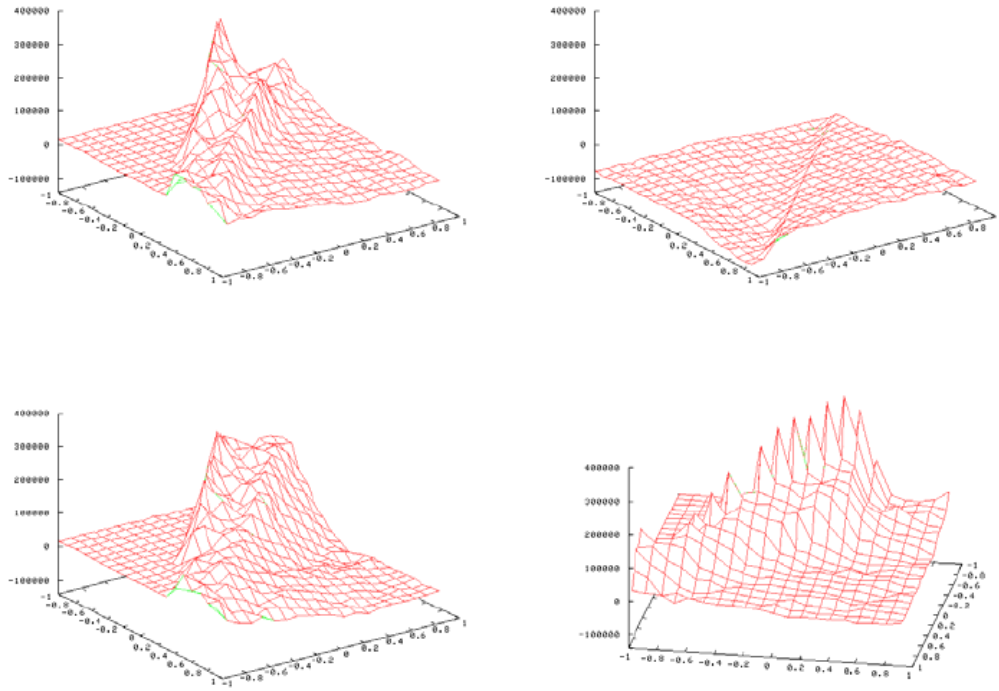


Figura 4: Muestra de un fragmento del espacio de búsqueda para un problema con 76 dimensiones. 72 están fijadas a los valores obtenidos para el mejor individuo. Dos dimensiones son recorridas para todo el rango de posibles valores en los ejes x e y de las gráficas y las otras dos dimensiones son recorridas en el tiempo, utilizando en ambos casos muestras cada 0.1. Las cuatro gráficas se corresponden a 4 parejas distintas de valores para esas dos dimensiones que son recorridas en el tiempo.

La comparación entre el algoritmo genético y el algoritmo macroevolutivo se ha hecho con evoluciones que se prolongaron un total de 2000 generaciones, con tres poblaciones distintas (480, 800 y 1600 individuos), migración local cada 40 generaciones, migración global cada 80 generaciones, 8 razas, probabilidad de cruce de 0.8 y de mutación de 0.1 para el algoritmo genético, y $\rho = 0.5$ y variación lineal de 1 a 0 para τ entre la generación 0 y la 1200, siendo permanentemente 0 desde esa generación para el algoritmo macroevolutivo.

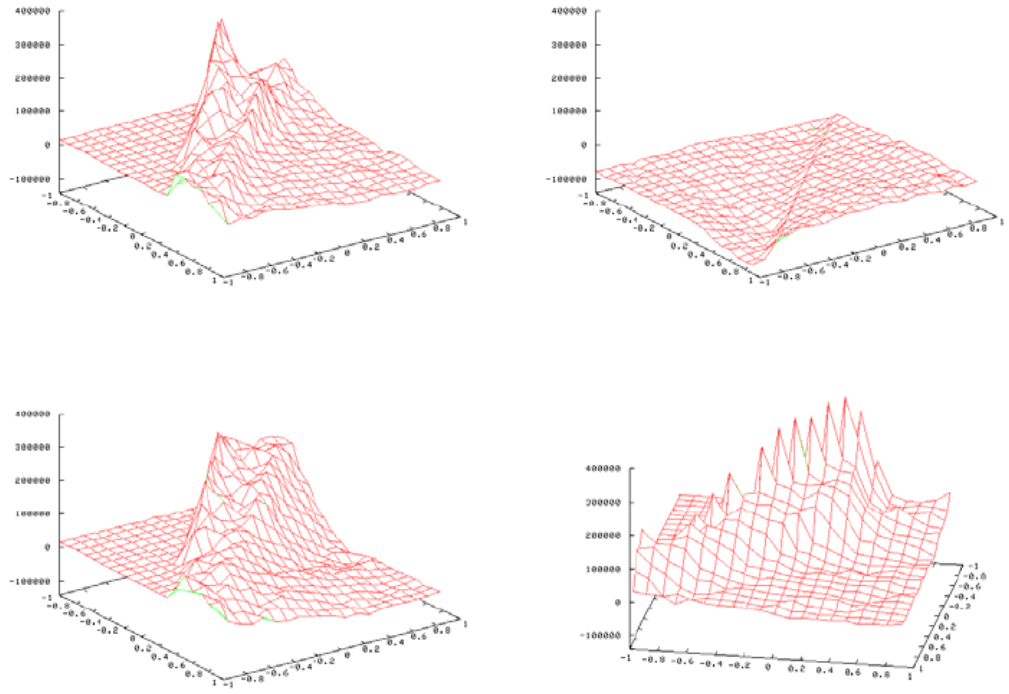


Figura 5: Muestra de un fragmento del espacio de búsqueda para un problema con 76 dimensiones. 72 están fijadas a los valores obtenidos para el mejor individuo. Dos dimensiones son recorridas para todo el rango de posibles valores en los ejes x e y de las gráficas y las otras dos dimensiones son recorridas en el tiempo, utilizando en ambos casos muestras cada 0.1. Las cuatro gráficas se corresponden a 4 parejas distintas de valores para esas dos

En la gráfica izquierda de la figura 4 vemos las calidades del mejor individuo para las tres evoluciones utilizando el algoritmo genético y el macroevolutivo. Lo primero que observamos en esta gráfica es que, a igualdad de población el macroevolutivo se comporta mejor que el genético, excepto en el caso de 1600 individuos, población suficiente para que el genético también alcance la mejor solución. O dicho de otra manera, el macroevolutivo con la mitad de población iguala los resultados del genético. El distinto comportamiento de ambos algoritmos también se aprecia claramente en la gráfica. El algoritmo genético evoluciona más rápidamente al principio debido a que los mejores individuos se reproducen mucho más que los restantes. Sin embargo, pronto se queda atascado al perder variedad genética y sólo poder adquirir ésta mediante las mutaciones. Si la evolución nos ha llevado a la solución correcta, perfecto, pero el riesgo de caer en máximos locales es elevado y mayor cuanto menor es la población. El macroevolutivo, por contra, realiza una evolución mucho más suave, dominada en un principio por una fase de exploración que poco a poco, a medida que τ va decreciendo, deja paso a la explotación. Por supuesto, sigue existiendo la posibilidad de caer en un máximo local, pero ésta es ahora menor por moverse el algoritmo evolutivo desde las zonas de menor calidad a las de mayor, no casi exclusivamente por las zonas de mayor calidad.

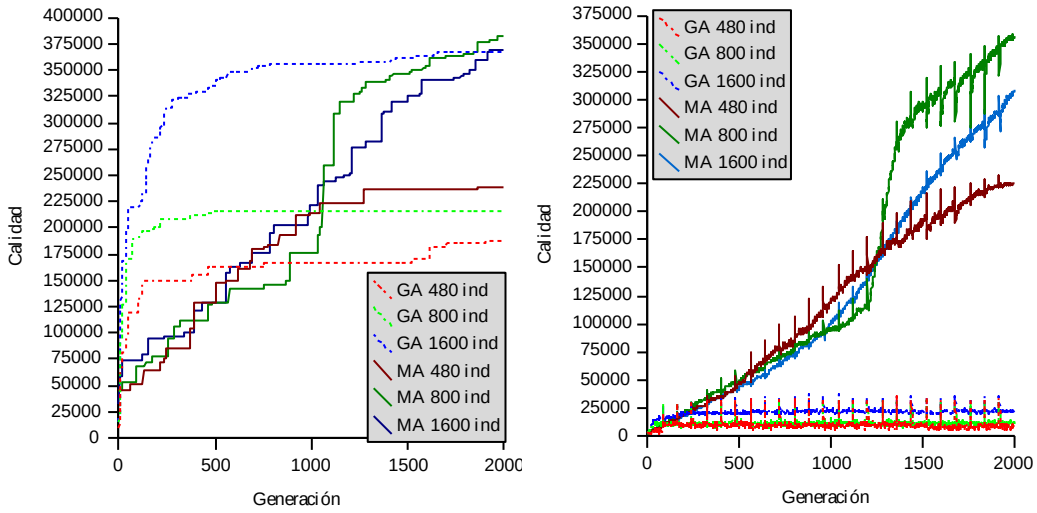


Figura 6: Calidad del mejor individuo (a la izquierda) y calidad media (a la derecha) para 6 evoluciones del comportamiento de seguir paredes para el Pioneer 2-DX.

En la gráfica derecha de la figura 4 también podemos ver la calidad media de la población para las 6 evoluciones anteriores. Se aprecia claramente la dificultad, que ya conocíamos, del problema, ya que los operadores de cruce y de mutación en el algoritmo genético generan en la mayor parte de las ocasiones individuos con baja calidad, llevando a calidades medias muy bajas. Se observa también cómo eso mismo no sucede en el caso del macroevolutivo, lo que indica, dado que la función que estamos optimizando es la misma, que los individuos se agrupan a medida que pasa el tiempo alrededor de los mejores individuos, o de lo contrario la calidad media no podría alcanzar esos valores.

Por tanto, el funcionamiento de los algoritmos macroevolutivos en este tipo de problemas con amplias superficies con valores de calidad similares, con picos esporádicos con mayor calidad y una elevada epístasis, se ha mostrado muy adecuado. La evolución es más lenta en un principio, pero es más constante, menos propensa a caer en máximos locales y con un buen equilibrio entre exploración y explotación.

Ahora, usando este mismo ejemplo, veremos cómo afectan los parámetros ρ y τ al funcionamiento del macroevolutivo. Respecto a ρ, que define el entorno de vecindad alrededor de las especies que sobreviven cuando se generan especies nuevas a partir de las que se extinguen, las pruebas llevadas a cabo para diferentes valores de este parámetro no mostraron una clara tendencia para valores razonables entre 0.3 y 0.8. Por eso, al igual que Marín y Solé utilizamos 0.5 en las restantes pruebas. Los resultados de las pruebas con τ requieren un análisis detenido. En la figura 5 vemos la calidad del mejor individuo para 8 evoluciones con distintos valores de este parámetro.

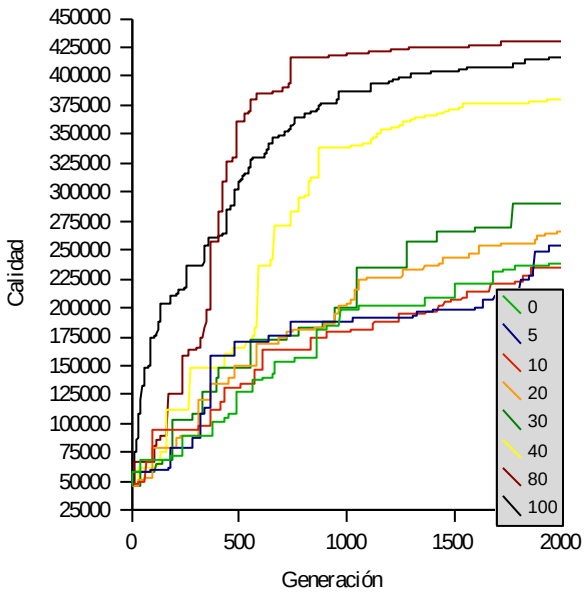


Figura 7: Calidad del mejor individuo para 8 evoluciones cambiando el parámetro τ a través de NR (0, 5, 10, 20, 30, 40, 80 y 100).

Existen numerosas formas de variar el parámetro τ. En este trabajo se ha escogido hacerlo de forma lineal de 1 a 0 hasta una determinada generación, a partir de la cual valdrá siempre 0 (fase de explotación pura). A ese número de generaciones con τ=0 le hemos denominado NR (*non randomness*) y es el indicado en la leyenda de la figura 5. En dicha gráfica podemos observar cómo cuanto mayor es la fase de pura explotación, más rápida es la evolución y antes se llega a la solución. Sin embargo, es preciso hacer notar que también aumentan las posibilidades de caer en máximos locales pues estaremos haciendo menos exploración y podemos dejar fuera de la búsqueda zonas del espacio de soluciones no cubiertas por la población inicial. Por tanto, es útil dejar reservado un número de generaciones para explotación pura en problemas como éste donde los máximos están en zonas muy delimitadas y que deben ser exploradas con minuciosidad, pero no hacer ese valor exagerado pues aumentaremos el riesgo de caer en máximos locales si la población es pequeña. De ahí, que se haya escogido un valor de NR = 40 para el resto de pruebas.

3. Simulador aerodinámico

Para realizar el trabajo de programación se ha optado por el uso de una biblioteca de módulos programados en C++ desarrollados siguiendo licencias GPL y con el código abierto OpenFOAM (Open Field Operation and Manipulation)[1] ya que los requerimientos computacionales de los sistemas utilizados hace que no sea económicamente rentable la utilización de software comercial. Esta biblioteca permite discretizar y resolver sistemas de ecuaciones en derivadas parciales que modelan problemas de ingeniería: ecuaciones de Navier-Stokes, de Maxwell... Además incorpora utilidades de pre y postprocesado para la transformación de información, obtención de magnitudes derivadas y su visualización. Aunque OpenFoam tiene módulos de mallado, para las configuraciones más complejas se ha empleado el programa GMSH[2], que permite generar mallas tridimensionales con diversas geometrías.

[1] http://www.opencfd.co.uk/openfoam

[2] http://geuz.org/gmsh

Como los orígenes de OpenFOAM están vinculados a la mecánica de fluidos, discretiza las ecuaciones usando el método de volúmenes finitos. Está previsto para trabajar sobre mallas no estructuradas tridimensionales con elementos tetraédricos, hexaédricos y pentaédricos. Además, puede transformar la información y resolver sencillos problemas mediante elementos finitos. Esto resulta especialmente útil en algunos casos de sistemas de ecuaciones acopladas. Otra ventaja a la hora de resolver problemas de gran tamaño es que incorpora un procedimiento de paralelización automática de los algoritmos, que no necesita la intervención directa del usuario.
Al tratarse de un código completamente abierto permite la incorporación de nuevos módulos, lo que proporciona un entorno de trabajo flexible y cómodo que puede ser utilizado tanto por el usuario no iniciado en los métodos numéricos (que se limitaría a utilizar los programas de la biblioteca) como por el experto (que podría incorporar sus propios códigos de discretización, resolución y postprocesado).

3.1. Método de volúmenes finitos

El método de volúmenes finitos es una técnica de discretización de ecuaciones en derivadas parciales que permite resolverlas numéricamente incluso en regiones con geometrías muy complejas. Para ello se subdivide el dominio del modelo en volúmenes de control (en nuestro caso el volumen de control será construido a partir de mallas de tetraedros), se asigna un valor de la magnitud incógnita a cada uno de ellos y se escriben ecuaciones algebraicas que traducen aproximadamente los ecuaciones continuas del modelo.
Una de las ventajas de los métodos de volúmenes finitos es que, al usar una discretización directa de la forma integral de las leyes de conservación, se pueden garantizar las propiedades de conservación de masa, momento y energía en el problema discreto[x].
Comenzaremos dando unas ideas básicas del principio del método para una ecuación modelo. Siguiendo a Godlewski [xi], consideremos la siguiente ecuación de conservación:

(1)

donde Ω es un abierto de $\Re^n$ y las funciones

(2)

son suficientemente regulares. Integrando la ecuación anterior en el volumen de control Ω_i ,

(3)

y aplicando el teorema de Gauss resulta

(4)

Siendo

$$\partial\Omega_i = \bigcup_{j=1}^{m_i} \Gamma_j^i \tag{5}$$

Con m_i el número de caras que componen la frontera de Ω_i y Γ_j^i es la frontera común a las celdas Ω_i y Ω_j . Se obtiene

(6)

donde n_{ij} denota la normal a la cara Γ_j^i dirigida hacia el exterior de Ω_i . Definiendo u_i como

(7)

en donde $|\Omega_i|$ es la medida de la celda Ω_i y puesto que ésta no cambia con el tiempo, se puede escribir de la forma

(8)

con lo que, introduciendo u_i se obtiene

(9)

quedando por discretizar el término del flujo en la frontera. Numerosos esquemas de aproximación de ese término están descritos en la bibliografía. De manera general buscaremos un flujo numérico que en cada cara de la frontera, Γ_j^i , sea una aprocimación del flujo físico, es decir,

(10)

construida con los valores u_i, u_j y n_{ij} y verificando las siguientes propiedades:

1. Consistencia

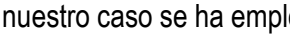
(11)

2. conservación (en ausencia de términos fuente)

(12)

Finalmente tendremos una expresión en la que intervendrán solamente los valores promedio en la celda Ω_i y otros elementos relacionados con las características geométricas de las mallas

(13)

Como ya se ha dicho, en la bibliografía se proponen diferentes expresiones para  . En nuestro caso se ha empleado una media ponderada de:

3. una aprocimación centrada (CD),

(14)

con $\alpha_{ij} = \frac{\|x_i - x_{ij}\|}{\|d_j\|}$ siendo x_{ij} el baricentro de la cara Γ_j^i y $d_j=x_j-x_i$,,

4. y una aprocimación descentrada, *upwind*, (UD), en la que el valor Φ_{ij} se calcula atendiendo a la dirección del flujo,

(15)

Dado un parámetro $\eta \in [0,1]$ se tomará como flujo numérico la media ponderada:

(16)

3.2. Formulación conservativa de las ecuaciones de Navier-Stokes

Vamos a tratar de construir una formulación de las ecuaciones de Navier-Stokes incompresibles y estacionarias sin fuerzas exteriores, tomando la presión como dato, que encaje con la ecuación modelo. Consideremos el sistema:

(17)

agrupando términos se obtiene que

$$\nabla\cdot\left(u\otimes\rho u-\mu\nabla u\right)=-\nabla\bar{p}.\tag{18}$$

Definiendo

$$F(u)=\left(u\otimes\rho u-\mu\nabla u\right).\tag{19}$$

el sistema se puede escribir

$$\nabla\cdot F(u)=-\nabla\bar{p}.\tag{20}$$

3.2.1. Cálculo de la velocidad

Aun tomando la presión como dato, las ecuaciones de Navier-Stokes presentan dos dificultades: la primera es el carácter no lineal del término

$$\nabla\cdot\left(u\otimes\rho u\right).\tag{21}$$

y la segunda es que al discretizar espacialmente con técnicas de volúmenes finitos, las derivadas de las soluciones del problema discreto no pertenecen a los espacios funcionales usuales.

Por lo que se refiere al segundo problema, el cálculo del gradiente de una magnitud vectorial θ definida constante por elemento se realiza mediante una aproximación de mínimos cuadrados[xii]. De este modo se obtiene

$$(\nabla\Theta)^i=\sum_{j=1}^{m}\frac{1}{\|d_j\|^2}\left(G^i\right)^{-1}\left(\Theta_j-\Theta_i\right)\otimes d_j\tag{22}$$

donde G^i es el tensor definido para cada celda

$$G^i=\sum_{j=1}^{m}\frac{1}{\|d_j\|^2}d_j\otimes d_j,\tag{23}$$

de forma que G^i es una constante que sólo depende de la malla y que $(\nabla\Theta)^i$ es una relación lineal para los valores de θ.

La primera dificultad se obviará mediante el uso de una técnica de punto fijo, más concretamente se emplea el siguiente algoritmo:

5. Dado un campo de velocidades inicial u^0 y considerando el campo de presiones p constante.
6. para n=1,2, ...

Se resuelve el problema lineal (no simétrico) con las condiciones de contorno adecuadas

$$\nabla\cdot F(u)=-\nabla\bar{p},\tag{24}$$

7. Test de parada.

Este procedimiento reduce la no linealidad cuadrática a sucesivas resoluciones de problemas lineales. El problema anterior admite la formulación conservativa

$$\int_{\Omega_i}\nabla\cdot\left(u\otimes\rho u-\mu\nabla u\right)d\Omega=-\int_{\Omega_i}\nabla\bar{p}\,d\Omega\tag{25}$$

.

Aplicando la condición de incompresibilidad y el terorema de Gauss resulta:

$$\int_{\partial\Omega_i}\left(\left(u_j n_j\right)\rho u-\mu\nabla u\cdot n\right)d\Gamma=-\int_{\partial\Omega_i}\bar{p}\,n_i\,d\Gamma\tag{26}$$

.

Definiendo

$$F(u)=\left(u_j n_j\right)\rho u-\mu\nabla u\cdot n.\tag{27}$$

Aplicando formulaciones centrada y upwind

. (28)

se tomará como flujo numérico en la cara común a las celdas Ω_k y Ω_l

. (29)

3.2.2. Cálculo de la presión

Para el ajuste de la presión se ha optado por un algoritmo PISO (Pressure Implicit Splitting of Operators) propuesto por Issa[xiii] que también permite su empleo en procesos dependientes del tiempo. Tomando la divergencia de la ecuación del momento se obtiene

 . (30)

utilizando que los operadores divergencia y laplaciano conmutan y la ecuación de la continuidad, finalmente resulta

 . (31)

De este modo, conocida la velocidad, se puede calcular la presión resolviendo una ecuación de Poisson. Esto sugiere la utilización de un método de punto fijo. El algoritmo sería el siguiente:

- Dada p^0, presión inicial,
- para m=1,2 ...

  Cálculo de u^{m+1} como solución del algoritmo anterior.

  Cálculo de

   (32)

  .

  Cálculo de la nueva presión

  . (33)

  Test de parada

permitiese adoptar los valores de estos parámetros que se ajustasen mejor al problema concreto tratado.

4. Ensayos numéricos

Dos casos han sido probados usando este entorno de diseño, palas de aerogenerador y formas de popa de carena. El uso de técnicas evolutivas en un proceso de diseño automático, es determinar las condiciones reales del problema y los parámetros necesarios para definir a cada individuo.

En el primer problema test, el aerogenerador objetivo tiene un diámetro máximo de 5.5 m trabajando siguiendo una distribución de viento Weibull con una velocidad media de 7 m/s y un parámetro de forma de valor 2, en este caso se han utilizado perfiles NACA para la definición de los perfiles de las palas.

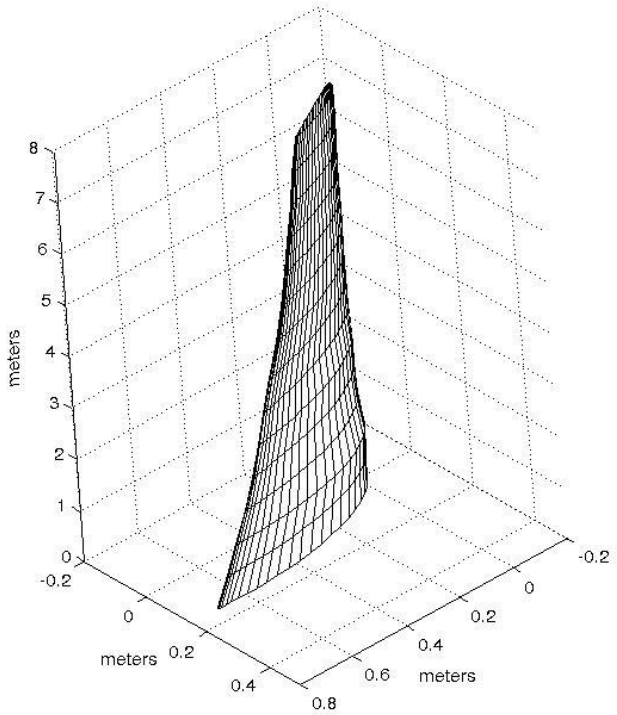
Cada pala es codificada en el proceso evolutivo mediante un cromosoma de 181 genes, seis para cada sección de las 30 que definen el perfil (cuatro para el perfil NACA, uno para el ángulo y uno para la cuerda) y uno adicional para la velocidad de roración.

Puesto que los resultados de un procedimiento evolutivo no es totalmente determinístico, para poder extrapolar resultados de análisis para poder analizar los resultados. En la siguiente tabla se comparan los resultados con diferentes parámetros del algoritmo evolutivo.

Generacones	Individuos	Mejor individuo	Calidad media
1500	1200	49452,57	48186,10
800	800	49744,40	48728,92

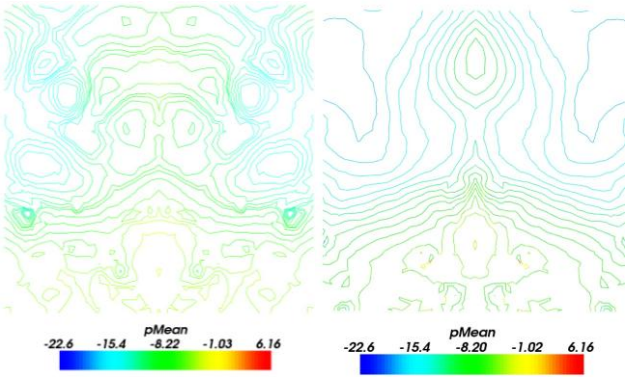
Estos problemas implican hipersuperficies de búsqueda muy complejas con gran número de óptimos locales dispersos rodeados por regiones de calidad baja. Así, si la región explorada es pequeña el procedimiento de búsqueda converge rápidamente al máximo local más cercano, así descuidando la búsqueda en el resto de las regiones y no explorando nuevas soluciones óptimas. Por esto es necesario buscar un equilibrio entre los elementos generados al azar y los generados de la combinación de individuos anteriores.

La figura siguiente figura muestra la forma de la mejor pala obtenida para las condiciones indicadas encima. Esta forma fue alcanzada en la 1370 generación y su eficacia es del 79.4 %. El poder generado por esta lámina es muy cerca del grado óptimo para el caso en la consideración.



El proceso de optimización aplicado a la optimización de las formas del casco por esta técnica evolutiva exige las cantidades sumamente grandes de recursos computacionales. De hecho, el coste computacional del simulador hace necesario de reducir el número de evaluaciones en varios órdenes de magnitud para permitir alcanzar soluciones en tiempos de cálculo aceptables.
Para el caso de la optimización de las forma de popa el proceso consiste de optimización consiste en la introducción en pequeñas deformaciones a un diseño de casco anterior obtenido por métodos de diseño

convencionales. Estas pequeñas deformaciones siguen una distribución de polinomio que conserva el volumen de casco. El objetivo de este proceso de optimización es mejorar la distribución de velocidad de flujo antes del propulsor. En este caso el valor de calidad es obtenido por una medida de la uniformidad de flujo en la región de propulsor tomada como la desviación estándar de la velocidad.
El proceso evolutivo fue realizado con una población de 15 individuos para 15 generaciones. El cromosoma codifica los coeficientes de la distribución de polinomio de las deformaciones del casco. La figura siguiente muestra el campo de presión inicial y final en el disco de propulsor. La mejora de la uniformidad puede ser observada en esta figura, y, a pesar del número bajo de evaluaciones la distribución de presión alcanza aumentos la eficacia de propulsor del 3.5 %.



5. Conclusiones

El desarrollo del sistema de diseño automático para velas ha implicado e implica la resolución de una serie de problemas técnicos y científicos que permite abrir la puerta a la adaptación de otras componentes y sistemas a dicho entorno mejorando sus prestaciones y posibles aplicaciones, incluso en un futuro alcanzando el diseño global de ciertos tipos de barcos. Para que esto pueda ser así se ha considerado de gran importancia avanzar en el estudio de la aplicación de técnicas como las planteadas en el proyecto de forma que los diseños, además de ser adecuados, permitan la interacción con los clientes y/o fabricantes adaptándose fácilmente a los gustos, costumbres y medios disponibles.
En cuanto a los paquetes de trabajo que componen el proyecto global hay que destacar las contribuciones que significan el estudio y desarrollo de metodologías de simulación que, con el aporte necesario de técnicas experimentales, permiten tener en cuenta de forma eficiente las interacciones tridimensionales del fluido con las velas. Además, la aplicación de técnicas evolucionistas y de modelado basado en redes neuronales artificiales en un caso complejo y real como el que se aborda en el presente proyecto, implican una novedad con sus modificaciones y avances en el conocimiento y aplicación práctica de estas técnicas, ya que la mayor parte de las publicaciones que aparecen en la literatura especializada se refieren muy mayoritariamente a análisis realizados sobre problemas simples.
Otro aspecto relevante del proyecto en el marco del diseño computacional y la simulación es la utilización en paralelo de técnicas experimentales de aplicación directa al problema para el ajuste y validación del sistema global. Este enfoque resulta de importancia crucial desde un punto de vista de la calidad de los resultados.

---

[i] *Wiener, N.* **Cibernetics, or Control and Communication in the Animal and the Machine.** Wiley 1948

[ii] *Darwin, C. R.* **On The Origin of Species by Means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life**. Murray. 1859

[iii] *Holland, J.* **Ann Arbor**. Adaptation in Natural and Artificial Systems, 1975

[iv] *Whitley, D.* **The GENITOR Algorithm and Selection Pressure: Why Rank-based Allocation of Reproductive Trials is Best**. Proc. 3rd Int. Conf. on Genetic Algorithms. 116-121. 1989

[v] *Goldberg, D. E. and Deb, K.* **A Comparative Analysis of Selection Schemes Used in Genetic Algorithms**. Foundations of Genetic Algorithms. Pp 69-93, 1991

[vi] *Rechenberg, I.* **Cybernetic Solution Path of an Experimental Problem.** Royal Aircraft Establishment Library Translation Pp 1122, 1965

[vii] *Schewefel, H. P.* **Evolutionsstrategie und Numerische Optimierung.** Technische Universität Berlin, 1975

[viii] *Fogel, L. J.* **On the Organization of Intellect**. University of California, 1964

[ix] *Marín, J. and Solé, R. V.* **Macroevolutionary Algorithms: A New Optimization Method on Fitness Landscapes.** IEEE Transactions on Evolutionary Computation. Vol. 4 pp 272-286, 1999

[x] *J.H. Ferziger.* **Computational Methods for Fluid Dynamics.** Springer, 2002.

[xi] *E. Godlewski and R. Raviart.* **Numerial Approximation of Hyperbolic Systems of Conservation Laws**. Springer, 1991.

[xii] *D. Whitlow and J.J. Chattot.* **A finite volume least-squares method.J. Comp. Fluid Dynamics**, 11(4):392_410, 2003.

[xiii] *Issa, R.I.* Soluition of the implicitly discretised fluid flow equations by operator-splitting. J. of Computatinal Physics, 1986.