Rev. Int. Mét. Num. Cálc. Dis. Ing.
Vol. **20, 2,** 139-167 (2004)

Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería

Un nuevo algoritmo evolutivo para la optimización de una o varias funciones objetivo sujetas a restricciones

Salvador Botello, Arturo Hernández y Giovanni Lizárraga

Centro de Investigación en Matemáticas
36000 Apdo. Postal 402
Guanajuato, Gto., México
Tel.: 52-473-732-71-55; Fax: 52-473-732-57-49
e-mail: botello@cimat.mx, artha@cimat.mx, giovanni@cimat.mx

Carlos Coello

CINVESTAV-IPN, D.I.E. Secc. Computación
Av. IPN 2580
07300 San Pedro Zacatenco, México D.F., México
Tel.: 52-555-061 38-00; Fax: 52-555-061 37 57
e-mail: ccoello@cs.cinvestav.mx

Resumen

Se presenta un nuevo algoritmo para considerar restricciones en funciones que tienen uno o varios objetivos a optimizar utilizando Algoritmos Evolutivos. El algoritmo aquí propuesto utiliza como base el algoritmo de PAES para controlar la diversidad de individuos en el frente de Pareto empleando una malla, la cual cambia dinámicamente el tamaño de sus celdas y dimensiones del espacio guiándose por las restricciones. Se propone un nuevo modelo que permite que la población evolucione hasta alcanzar las regiones factibles y se acerque al punto óptimo en caso de funciones con sólo un objetivo, o bien que tenga una muy buena dispersión en la región factible en el caso de funciones multiobjetivo. Varios ejemplos han sido utilizados para mostrar el potencial de ISPAES, y se presentan aplicaciones en la optimización de estructuras sujetas a cargas axiales.

ISPAES: A NEW EVOLUTIONARY ALGORITHM FOR THE OPTIMIZATION OF ONE OR MANY OBJECTIVE FUNCTIONS WITH CONTRAINS

Summary

We introduce a new evolutionary algorithm with constraint handling for single and multiple objective optimization. The proposed algorithm uses a grid to keep population diversity in a similar way PAES does it, but here the grid is adaptable, this is, its size changes accordingly to the constraints and adapts to the search space once the population reaches the feasible region. This adaptable mechanism has proven very powerful for approaching the true Pareto front, either in problems with one objective or many objectives with constraints. Several truss optimization problems subject to axial loads are solved to show the potential and robustness of the algorithm.

 ISSN: 0213–1315 Recibido: Agosto 2003 Aceptado: Nov. 2003

INTRODUCCIÓN

Existen problemas en los que se desea optimizar, ya se una o varias funciones objetivo en donde las variables a encontrar, o las propias funciones a optimizar deben satisfacer ciertas restricciones. Los algoritmos evolutivos (AE) proporcionan un mecanismo eficiente para encontrar la solución óptima en problemas en los que se tiene delimitada la región factible por medio de restricciones. Dichas restricciones pueden ser igualdades o desigualdades y, a su vez estas ser lineales o no. Se ha desarrollado bastante trabajo de investigación en los últimos años para perfeccionar los mecanismos que permiten incorporar las restricciones dentro de las funciones de mérito que se utilizan para optimizar en los AE[3,6]. Una aproximación, que comúnmente se ha utilizado para incorporar las restricciones utilizando optimización evolutiva es mediante la aplicación de funciones de penalización. Consiste en castigar la función de mérito cuando alguna de las restricciones ha sido violada, de tal suerte que cuando se tiene una solución factible, ésta se vea favorecida cuando se compara con alguna que viole una o varias restricciones. Un grave inconveniente para la utilización de funciones de penalización es poder determinar los parámetros de castigo, los cuales en general van a depender del tipo de problema a resolver[5,4]. Recientemente algunos investigadores han propuesto utilizar conceptos de optimización multiobjetivo para incluir las restricciones como objetivos en los AE[4]. En este trabajo se presenta una nueva aproximación basada en estrategias evolutivas y que originalmente fue propuesta para la optimización de multiobjetivos: Pareto Archived Evolution Strategy (PAES)[7]. En nuestra propuesta pueden considerarse las restricciones en la optimización de funciones que tienen un solo objetivo o también determinar la mejor solución compromiso en funciones multiobjetivo. En la sección a continuación se presenta la definición matemática del tipo de problemas que tratamos de resolver; en la sección siguiente se hace una breve descripción sobre los antecedentes a este trabajo; posteriormente se describe con todo detalle el algoritmo ISPAES propuesto en este artículo; luego se presentan los resultados que se obtuvieron al utilizar este algoritmo en la optimización de funciones de uno y varios objetivos sujetas a restricciones, y, en la sección final presentamos las conclusiones.

DEFINICIÓN DEL PROBLEMA

Estamos interesados en optimizar el problema general no lineal en el que se desea

$$\text{Encontrar} \quad \vec{x} \quad \text{que optimiza} \quad \vec{f}(\vec{x}) \tag{1}$$

sujeto a

$$g_i(\vec{x}) \leq 0, \quad i = 1, \ldots, n \tag{2}$$

$$h_j(\vec{x}) = 0, \quad j = 1, \ldots, p \tag{3}$$

donde $\vec{f}$ es el vector de k funciones objetivo $\vec{f} = [f_1(\vec{x}), \ldots, f_k(\vec{x})]$, $\vec{x}$ el vector solución $\vec{x} = [x_1, x_2, \ldots, x_r]^T$ que tiene r variables; n el número de restricciones de desigualdad; y p el número de restricciones de igualdad (en ambos casos pueden ser condiciones lineales o no lineales).

Si se denomina con $\mathcal{F}$ la región factible y con $\mathcal{S}$ el espacio donde deberá realizarse la búsqueda de la solución, debe ser claro que $\mathcal{F} \subseteq \mathcal{S}$.

ANTECEDENTES

Diversos modelos se han propuesto en los últimos años para tratar de resolver el problema de optimización con restricciones. Algunos proponen el uso de técnicas basadas en poblaciones[2], otros se sirven del uso de la dominancia de Pareto como mecanismo de selección[1] y otros hacen uso de la jerarquización de Pareto[8]. Dichas técnicas han sido utilizadas con éxito para determinar la región factible, pero no son muy efectivas para encontrar el óptimo global de un problema. En este trabajo se propone un algoritmo general que nos permite determinar la región factible y, dependiendo del número de funciones objetivo, poder estimar con facilidad el entorno del punto óptimo cuando se trata de un solo objetivo o bien el frente de Pareto óptimo cuando tenemos dos o más objetivos.

Dado que nuestro algoritmo está basado en utilizar conceptos de optimización multiobjetivo, se presenta a continuación una breve discusión sobre los más relevantes trabajos realizados en esta área. La idea principal que empleamos para incluir las restricciones en la optimización multiobjetivo, es redefinir el problema de optimización global $\vec{f}(\vec{x})$ como un problema de optimización multiobjetivo que tiene $k+m$ objetivos, donde m es el número total de restricciones ($m = n + p$) y k es el número de funciones objetivo del problema original. Entonces podemos aplicar cualquier técnica de optimización multiobjetivo al nuevo vector $\bar{v} = (f(\vec{x}), f_1(\vec{x}), \ldots, f_{k+m}(\vec{x}))$, donde $f_1(\vec{x}), \ldots, f_k(\vec{x})$ son las funciones objetivo del problema original y $f_{k+1}(\vec{x}), \ldots, f_{k+m}(\vec{x})$ son nuevas funciones objetivo que representan las restricciones del problema. Una solución $\vec{x}$ ideal de este problema sería que $f_i(\vec{x})$=0 para $k+1 \leq i \leq k+m$ (satisfacer las restricciones) y que $f(\vec{x}) \leq f(\vec{y})$ para todo $\vec{y}$ factible (asumiendo minimización).

En general, puede considerarse que son tres los mecanismos utilizados en las técnicas de optimización multiobjetivo que incorporan restricciones:

1. uso del criterio de dominancia de Pareto para hacer la selección;
2. uso de la jerarquización de Pareto[9] para asignar mejor aptitud entre individuos dominados o bien asignar una mayor aptitud a los individuos que son no dominados;
3. partir la población en subpoblaciones que son evaluadas con respecto a la función objetivo o con respecto a cada una de las restricciones del problema. Éste es el mecanismo de selección adoptado en el Vector Evaluated Genetic Algorithm (VEGA)[10].

A continuación se presenta una breve descripción de las diferentes aproximaciones que se han propuesto en la literatura y que adoptan alguna o una combinación de las tres principales ideas anteriormente indicadas.

VEGA

Parmee y Purchase[20] propusieron el uso de VEGA[10] para guiar la búsqueda de algoritmos evolutivos a la región factible en el diseño de una turbina de gas en un muy complicado espacio de restricciones. Esta aproximación realmente no utiliza dominancia de Pareto para explorar el espacio de búsqueda, requiere ser iniciado en la región factible y su máxima fortaleza es utilizar operadores especiales que no permiten perder la zona factible, por lo que esta propuesta es muy específica para cierto tipo de problemas.

COMOGA

Surry y Radcliffe[20] usaron una combinación del Vector Evaluated Genetic Algorithm (VEGA)[10] y jerarquización de Pareto para incluir las restricciones en el algoritmo denominado COMOGA (Constrained Optimization by Multi-Objective Genetic Algorithms).

En esta técnica, los individuos son jerarquizados dependiendo de la suma de restricciones violadas (o sea, el número de individuos dominados por la función objetivo). El proceso de selección es realizado no sólo por la jerarquización, sino que también considera la aptitud de cada solución. COMOGA usa un Algoritmo Genético no generacional y un conjunto de parámetros extra definidos por el usuario. Una desventaja de este algoritmo es que es muy sensible a los valores de los parámetros, lo que lo hace poco atractivo.

MOGA

Coello[1] propuso, el uso de la dominancia de Pareto en la selección para considerar las resticciones en AE. Esta es una aplicación del proceso de jerarquización de Pareto propuesta por Fonseca y Fleming[21] (llamada Multi-Objective Genetic Algorithm o MOGA) para incluir restricciones. En esta propuesta los mejores individuos son jerarquizados sobre los peores. Basándose en dicha jerarquización, el valor de aptitud es asignado para cada individuo. Esta técnica incluye un mecanismo de autoadaptación que es controlado por un parámetro que usualmente debe ser determinado en forma empírica.

NPGA

Coello y Mezura[4] implementaron una versión del Niched-Pareto Genetic Algorithm (NPGA)[16] para incluir las restricciones en problemas de optimización de un objetivo sujetos a restricciones. El NPGA es una técnica de optimización multiobjetivo en la que los individuos son seleccionados por torneo, basándose en la dominancia de Pareto. Este método tiene la ventaja de que no realiza comparaciones de cada individuo con los otros (como es tradicional en los métodos de jerarquización de Pareto); usa solamente una parte de la población para estimar la dominancia de Pareto, lo que reduce sustancialmente el tiempo de cómputo para obtener un buen resultado.

Grupos de pareto y búsqueda en línea

Grupos de Pareto y Búsqueda en Línea (Pareto Set and Line Search), desarrollado por Camponogara y Talukdar[24], en donde se propone una transformación del problema de optimización global en un problema de dos objetivos: el primero que contiene las funciones objetivo originales y el segundo, las restricciones, teniendo como nuevo objetivo

$$\Phi(\mathbf{x}) = \sum_{i=1}^{n} \max(0, g_i(\mathbf{x})) \tag{4}$$

La ecuación (4) trata de minimizar la suma total de las restricciones violadas de una determinada solución. En cada generación del proceso son generados diversos grupos de Pareto. Un operador que sustituye al cruce utiliza dos grupos de Pareto S_i y S_j, donde $i < j$, y dos soluciones $x_i \in S_i$ y $x_j \in S_j$, donde x_i domina a x_j. Con esos dos puntos, una búsqueda direccional es definida usando

$$d = \frac{(x_i - x_j)}{|x_i - x_j|} \tag{5}$$

La búsqueda en línea comienza proyectando d sobre uno de los ejes de las variables de decisión para encontrar una nueva solución x que domine a las anteriores x_i y x_j.

En intervalos predefinidos, la peor mitad de la población es reemplazada con una nueva solución aleatoria para impedir una convergencia prematura. Esto indica que este modelo tiene algunos problemas para mantener la diversidad. Además, el uso de una búsqueda en línea con los algoritmos genéticos aumenta el coste computacional.

Min-Max

Una propuesta similar a la formulación Min-Max usada en la optimización multiobjetivo[27] combinada con selección de torneo, fue propuesta por Jiménez y Verdegay[28]. Ellos proponen cuatro funciones de prueba y sus resultados son muy cercanos al óptimo. Un problema de esta aproximación es que el proceso evolutivo está concentrado primero en satisfacer las restricciones, y en la región factible la búsqueda es esencialmente aleatoria[13].

Dominancia de Pareto y preselección

Jiménez *et al.*[29] proponen un algoritmo basado en la dominancia de Pareto, dentro de la técnica de preselección para resolver diversos problemas de optimización (multiobjetivo, satisfacción de restricciones y problemas de programación). Se redefine el problema como un problema multiobjetivo sin restricciones y se les da prioridad a los objetivos del problema original. Soluciones factibles con un buen valor de la función objetivo tendrán prioridad sobre los otros. Utilizan una codificación en números reales y un algoritmo genético no generacional con dos tipos de operadores de cruce (uniforme y aritmético) y dos operadores de mutación (uniforme y no uniforme).

Jerarquización de Pareto y conocimiento del dominio

Ray *et al.*[25] proponen el uso de jerarquización de Pareto y conocimiento del dominio (Pareto Ranking and Domain Knowledge) para operar en tres espacios: el espacio factible, el de las restricciones y una combinación de los dos espacios. En esta aproximación se mantiene una búsqueda sobre el espacio en el que son violadas las restricciones y de esta forma se permite hacer una mejor exploración. Para mantener la diversidad se utiliza un mecanismo basado en nichos, utilizando distancias euclidianas. Con esta propuesta pueden resolverse problemas de uno o varios objetivos sujetos a restricciones. La principal ventaja de esta aproximación es que requiere un bajo número de evaluaciones de la función objetivo (entre un 2 % y un 10 % del número de evaluaciones requeridas por el mapa homomorfo de Koziel y Michalewicz [22], que es una de las mejores técnicas conocidas hoy en día para optimizar considerando restricciones). Esta técnica tiene algunos problemas para determinar el óptimo global, pero produce muy buenas aproximaciones a un bajo coste computacional. La principal desventaja de esta aproximación es que requiere una implementación considerablemente más compleja que cualquiera de las otras técnicas anteriormente descritas.

Jerarquización de Pareto y optimización robusta

Ray[26] realiza una extensión a su trabajo previo para incluir restricciones[25] basándose en la jerarquización de Pareto y conceptos de optimización robusta (Pareto Ranking and Robust Optimization). Una solución óptima es robusta si no es sensible a variaciones de los parámetros. Esta aproximación considera restricciones y determina la región factible de forma robusta en problemas en los que puede haber variaciones de los parámetros en el tiempo.

ALGORITMO IS-PAES

El algoritmo IS-PAES es una extensión del Pareto Archived Evolution Strategy (PAES) propuesto por Knowles y Corne[17,7] para optimización multiobjetivo. La principal ventaja de PAES es que utiliza una malla adaptable en un espacio coordenado definido por las funciones objetivo. Dicha malla es la encargada de mantener la diversidad en el algoritmo. La malla es creada biseccionando cierto número de veces b, el espacio de funciones de dimensión $k + m$ (número de funciones objetivo más restricciones).

Algoritmo principal IS-PAES

maxsize: máximo tamaño del archivo
c: padre actual t ∈ X (Espacio de las variables de decisión)
h: hijo de c ∈ X, a_h: Individuo en el archivo que domina *h*
a_d: individuo en el archivo dominado por *h*
current: número actual de individuos en el archivo
cnew: número de individuos generados actualmente
actual = 1; cnew=0;
c = Nuevo Individuo();
agregar(c);
While cnew ≤MaxNew **do**
h = mutar(c); cnew+=1;
if (c⪯h) **then label A**
else if (h⪯c) **then**
{remover(c); agregar(h); c=h; }
else if(∃ a_h ∈file | $a_h \preceq$ h) **then label A**
else if (∃ a_d ∈file | h ⪯ a_d } **then**{
agregar(h); ∀ a_d[remover(a_d); actual-=1 }
else test(h,c,file)
label A
if (cnew % g==0) **then** {c = generar individuo en la región con menor densidad de población}
if (cnew % r==0) **then** CortarEspacio(file)
End While

Tabla I. Algoritmo principal IS-PAES

Una de las fuertes desventajas del PAES es que para problemas pequeños el número de celdas es muy grande ($2^{b(k+m)}$) y los requerimientos de memoria física pueden ser muy importantes. Por ejemplo, para un problema con 10 funciones objetivo (que puede incluir restricciones) y 5 biseccciones del espacio se requiere una malla con 2^{50} celdas. En esta propuesta, el primer cambio que se hace al algoritmo PAES original es "invertir" la descripción de la malla, es decir, almacenar solamente los datos de la celda donde está localizada cada solución. De esta forma se reduce sustancialmente el espacio de memoria que se requiere para resolver un problema con bastantes funciones objetivo. El algoritmo principal de ISPAES se muestra en la Tabla I, en donde $a \preceq b$ indica dominancia de a sobre b. En el algoritmo que estamos proponiendo reducimos en el tiempo el espacio de búsqueda, lo cual fue utilizado por otros investigadores[11], pero esto no ha sido anteriormente utilizado en computación evolutiva y mucho menos en optimización multiobjetivo con restricciones. La idea principal en este caso es reducir el espacio de búsqueda sobre la base de factibilidad y aptitud de las funciones objetivo. La reducción del espacio se hace de tal forma que, después de varias exploraciones en el espacio actual se seleccionan los mejores elementos

del archivo y con ellos se reduce el espacio de búsqueda. Es importante resaltar que debe tenerse cuidado al realizar los cortes, pues pueden dejarse fuera algunas zonas factibles del espacio. Eventualmente, al realizarse esta operación de forma repetitiva podemos encontrar el entorno de la solución para el caso de funciones de un objetivo con restricciones, o bien el frente de Pareto para funciones multiobjetivo con restricciones.

La función **test(h,c,file)** determina si un individuo puede ser agregado al archivo o es desechado. Nosotros estamos introduciendo la siguiente notación $x_1 \square x_2$, que indica que x_1 está localizado en una región menos poblada de la malla que x_2. El pseudocódigo de está función está descrito en la Tabla II.

Pseudocódigo test(h,c,file)

if (actual < maxsize) **then**
  agregar(h);
  if (h $\square$ c) **then** c=h
else if ($\exists$ a_p $\in$file | h $\square$ a_p) **then**
  remover(a_p); agregar(h)
  if (h $\square$ c) **then** c = h;

Tabla II. Pseudocódigo test

IS-PAES manipula la población *como parte de* una relación entre celdas de una malla, mientras que PAES manipula una celda de malla que *contiene* la relación entre la población. Es decir, PAES mantiene una lista de individuos en cualquier celda de la malla, pero el IS-PAES determina para cualquier individuo su posición en la malla, de tal forma que sólo se requiere una lista donde se determinen las áreas más pobladas y se necesitan solamente $maxsize$ elementos de memoria externa para poder generar una lista ordenada por densidad de población. En el algoritmo PAES este procedimiento necesita verificar en cada celda de la malla si existen individuos y después hacer una lista ordenada. La ventaja que se logra al hacer esta relación *invertida* (Inverted es la I en el nombre IS-PAES) es muy clara cuando optimizamos un número elevado de funciones objetivo o bien cuando se utilizan mallas finas (con muchas celdas).

Cortando el espacio objetivo

CortarEspacio(file) es una función muy importante de IS-PAES dado que es la que reduce el espacio de búsqueda. El pseudocódigo de **CortarEspacio(file)** se presenta en la Tabla III.

Pseudocódigo CortarEspacio(file)

$\underline{x}_{pob}$: vector que contiene el valor
  más pequeño de cualquier $x_i \in X$
$\overline{x}_{pob}$: vector que contine el valor
  más grande de cualquier $x_i \in X$
selecciona(file);
obtiene MinMax(file, $\underline{x}_{pob}$, $\overline{x}_{pob}$);
recorta($\underline{x}_{pob}$, $\overline{x}_{pob}$);
if($l == 1$)ajustaParámetros(file); * caso: una función objetivo *

Tabla III. Pseudocódigo CortarEspacio

La *función tamaño(file)* regresa una lista de los elementos que corresponden a los mejores individuos encontrados en el archivo *file*. El tamaño de la lista es del 15 % de $maxsize$. Los individuos pueden estar localizados en la región factible, en la no factible o en una mezcla de ambas; la lista de individuos generada descarta del archivo *file* los peores elementos para cada restricción i, según el orden descrito en el arreglo *restricciónválida*. Se advierte que la función *selección(file)* no escoge directamente los mejores individuos, sino que retira del archivo los individuos que más violan cada restricción.

Pseudocódigo selecciona(file)

```
l: número de funciones objetivo
m: número de restricciones
mr: número de restricciones violadas
i: índice de restricción
maxsize: tamaño máximo de archivo
listsize: if(l == 1) then 15 % de maxsize
          else 50 % de maxsize
valorrestricción(x,i): valor de la
       restricción i
mrfile(file,mr): calcula mr
Peor(file,i): peor individuo del archivo para restricción i
restricciónválida={1,2,3,...,m};
i=Primerode(restricciónválida);
if(l > 1){
   mrfile(file,mr);
   if(mr <listsize)listsize=mr;
}
While (tamaño(file) > listsize and
          tamaño(restricciónválida) > 0) {
    x=peor(file,i)
    if (x viola restricción i)
       file=borra(file,x)
    else restricciónválida=
       borraíndice(restricciónválida,i)
if (tamaño(restricciónválida) > 0)
       i=siguienteen(restricciónválida)
}
if(l == 1){
   if (tamaño(file)== listsize) list=file
   else{
       file=sort(file)
       list=copea(file,listsize) *Toma los mejores elementos de listsize *
    }
}
```

Tabla IV. Pseudocódigo selecciona

Los individuos que forman el archivo son, por orden jerárquico: 1) solamente los mejores individuos factibles o 2) una combinación de los individuos factibles y los mejores no factibles o 3) los mejores individuos no factibles. Es de esperar que al finalizar el algoritmo la lista esté formada por individuos factibles. El algoritmo de selección se muestra en la Tabla IV.

La función *restricciónválida* genera una lista ordenada de los índices de las restricciones. Esta lista puede definirse en orden determinado y fijo para todo el análisis, o bien recalcularse en forma aleatoria y reorganizarse después de cierto número de generaciones. Según las pruebas que realizamos no es importante el orden en que se hace este listado, por lo que pueden definirse en forma estática desde el principio del algoritmo. La función obtieneMinMax(file) encuentra los valores extremos de las variables de decisión de todos los individuos que se encuentran en el archivo *file*. De esta forma se determinan los vectores $\underline{x}_{pob}$ y $\overline{x}_{pob}$.

Pseudocódigo recorta

n: tamaño del vector decisión;
$\overline{x}_i$: actual frontera superior en la i_{esima} variable de decisión
$\underline{x}_i$: actual frontera inferior en la i_{esima} variable de decisión
$\overline{x}_{pob,i}$: frontera superior de i_{esima} variable de decisión
en la población $\forall i$: i ∈ { 1, . . . , n }
$\underline{x}_{pob,i}$: frontera inferior de i_{esima} variable de decisión
en la población $\forall i$: i ∈ { 1, . . . , n }
$parte_i = 0.05 \times (\overline{x}_{pob,i} - \underline{x}_{pob,i})$
$ancho\_pob_i = \overline{x}_{pob,i} - \underline{x}_{pob,i}$; $ancho_i^t = \overline{x}_i^t - \underline{x}_i^t$
$deltaMin_i = \frac{\beta * ancho_i^t - ancho\_pob_i}{2}$
$\text{delta}_i = \max(\text{parte}_i, \text{deltaMin}_i)$;
$\overline{x}_i^{t+1} = \overline{x}_{pob,i} + delta_i$; $\underline{x}_i^{t+1} = \underline{x}_{pob,i} - delta_i$;
if $(\overline{x}_i^{t+1} > \overline{x}_{original,i})$**then**
$\underline{x}_i^{t+1} -= \overline{x}_i^{t+1} - \overline{x}_{original,i}$;
$\overline{x}_i^{t+1} = \overline{x}_{original,i}$;
if $(\underline{x}_i^{t+1} < x_{original,i})$**then** {
$\overline{x}_i^{t+1} += \underline{x}_{original,i} - \underline{x}_i^{t+1}$;
$\underline{x}_i^{t+1} = \underline{x}_{original,i}$;}
if $(\overline{x}^{t+1} > \overline{x}_{original,i})$**then** $\overline{x}_i^{t+1} = \overline{x}_{original,i}$;

Tabla V. Pseudocódigo recorta

La función recorta ($\underline{x}_{pob}$, $\overline{x}_{pob}$) determina el potencial espacio factible alrededor de las mejores soluciones encontradas en el hipervolumen definido por los vectores $\underline{x}_{pob}$ y $\overline{x}_{pob}$ (Tabla V). La ventaja que se logra al hacer el *corte* ("Shrink" es la "S" en el nombre IS-PAES) es reducir el espacio de las variables de decisión, lo que permite determinar el entorno de la solución en el caso de funciones de un objetivo o las fronteras de regiones válidas en funciones con varios objetivos, ambas sujetas a restricciones.

El valor de β es el porcentaje por el cual los valores de la frontera para cada $x_i \in X$ será reducida, tal que el resultado del hipervolumen H sea una fracción α de su valor previo. En IS-PAES todas las variables objetivo son reducidas en el mismo porcentaje β. El valor de β puede ser deducido en función de α como se describe a continuación

$$H_{t+1} \geq \alpha H_t \tag{6}$$

$$\prod_{i=1}^{n} (\overline{x}_i^{t+1} - \underline{x}_i^{t+1}) = \alpha \prod_{i=1}^{n} (\overline{x}_i^t - \underline{x}_i^t)$$

Cada x_i es reducida en el mismo porcentaje β; tendremos por tanto

$$\prod_{i=1}^{n} \beta(\overline{x}_i^t - \underline{x}_i^t) = \alpha \prod_{i=1}^{n} (\overline{x}_i^t - \underline{x}_i^t)$$
$$\beta^n \prod_{i=1}^{n} (\overline{x}_i^t - \underline{x}_i^t) = \alpha \prod_{i}^{n} (\overline{x}_i^t - \underline{x}_{i=1}^t)$$
$$\beta^n = \alpha$$
$$\beta = \alpha^{\frac{1}{n}}$$

En cada corte, el intervalo de búsqueda de cada variable de decisión x_i es reducido como se muestra a continuación (el algoritmo completo se presenta en la Tabla V)

$$ancho_{new} \geq \beta \times ancho_{old}$$

En nuestras pruebas, $\alpha = 0,90$ ha trabajado bien en todos los casos. El valor de α controla la velocidad de corte, por lo que la convergencia adecuada de este algoritmo depende de este parámetro. En todas las pruebas que realizamos probamos los valores en el rango [85 %, 95 %] y no se percibió un cambio en el funcionamiento del algoritmo[39]. Por supuesto que un valor de α cercano al 100 % reduce la velocidad de convergencia del algoritmo.

El último paso de la función CortarEspacio() es la llamada a la función que reinicia las variables de mutación *ajustaparámetros(file)*. La idea de este paso es reiniciar la variable de control σ utilizando

$$\sigma_i = (\overline{x}_i - \underline{x}_i)/\sqrt{n} \quad i \in (1, \ldots, n) \tag{7}$$

Esta expresión es también utilizada durante la generación de la población inicial. En este caso, las fronteras inferior y superior toman los valores iniciales del espacio de búsqueda indicados en el problema. La variación de la probabilidad de mutación sigue el comportamiento exponencial descrito por Bäck[12].

Elitismo

Una forma particular de elitismo es implementada en el IS-PAES para impedir la pérdida del mejor individuo en cada generación. El mejor individuo podría perderse cuando se utiliza el algoritmo normal, donde se puede eliminar el individuo de la zona más densamente poblada. Para evitarlo proponemos que el mejor individuo no puede ser eliminado, y si es necesario, se debe eliminar algún vecino cercano cuando coincida que el mejor se encuentra en la celda más poblada y sólo será reemplazado si la aptitud del nuevo individuo es mejor.

Optimización de variables discretas

El algoritmo IS-PAES ha sido implementado para resolver problemas con variables que pueden tener representación tanto real como entera. Ahora discutiremos los ajustes que hay que hacer en el algoritmo para incluir espacios de búsqueda enteros. El operador de mutación depende de la variable de control σ_i y afecta a las variables objetivo x_i. El valor inicial de la variable σ_i es calculado antes (ver ec. (7)), pero deberá tomarse el valor entero

más cercano y nunca deberá ser menor que 1. En cualquier generación σ_i es actualizado como sigue

$$If(random() < 0,45)\ then\ |\sigma_i| = |\sigma_i| + \theta;\ else|\sigma_i| = |\sigma_i| - \theta$$

$$If(random() < 0,5)\ then\ \sigma_i = |\sigma_i|;\ else\ \sigma_i = -|\sigma_i|$$

El valor de θ es un parámetro definido por el usuario y representa el valor de cambio aplicado a σ_i. La función $random()$ genera un número real entre $[0, 1]$. El valor de $0,45$ favorece el decremento de la variable σ_i al hacerse varias llamadas en el proceso iterativo.

El valor inicial de las variables es un número entero aleatorio (de una distribución uniforme) en el rango válido del espacio de búsqueda. La mutación se realiza de la forma siguiente:

$$x_i^{t+1} = x_i^t + sign(\sigma_i) * rand(|\sigma_i|)$$

La función $rand(|\sigma_i|)$ genera un número entero en $[0, \sigma_i]$. La función $sign()$ transforma los números negativos y positivos a $\{-1, 1\}$.

La última modificación en el algoritmo se realiza en la función $CortaEspacio()$ donde la reducción del espacio deberá ser un número entero de valor η. Es decir, el espacio de búsqueda se reduce como $\overline{x}_i = \overline{x}_i - \eta$, y $\underline{x}_i = \underline{x}_i + \eta$, pero garantizando que el nuevo intervalo debe ser menor que los valores actualmente utilizados para encerrar la población, tal que

$$If\ (\overline{x}_i - \eta) < \overline{x}_{pob,i}\ then\ \overline{x}_i = \overline{x}_{pob,i};\ else\ \overline{x}_i = \overline{x}_i - \eta$$

y

$$If\ (\underline{x}_i + \eta) > \underline{x}_{pob,i}\ then\ \underline{x}_i = \underline{x}_{pob,i};\ else\ \ \underline{x}_i = \underline{x}_i + \eta$$

son las expresiones para recortar el espacio de búsqueda en η.

EXPERIMENTOS

En todos los ejemplos en esta sección utilizamos los siguientes parámetros: el tamaño del archivo es 100; todos los miembros del archivo pueden participar en la siguiente generación; el corte se realiza cada dos generaciones si es necesario y solamente se reduce el 10 % del hipervolúmen en cada corte, reduciendo al 50 % el tamaño de elementos en el archivo; se utilizaron 500 generaciones en cada problema. El parámetro de mutación en todos los casos fue $\phi = 1$ y la σ no fue reinicia en cada corte.

Ejemplo 1

Ejemplo de optimización con dos objetivos. La armadura de la Figura 1 tiene que soportar una carga de 100 kN. El objetivo es minimizar el volumen de material (diseñar para el mínimo coste de fabricación) y reducir al máximo los valores de los esfuerzos en

cada barra. Este problema fue propuesto por Deb[14] como un problema de optimización con dos objetivos y con tres variables libres x_1, x_2 y

$$\begin{aligned} \text{Minimizar } f_1(x) &= x_1\sqrt{(16+y^2)} + x_2\sqrt{(1+y^2)} \\ \text{Minimizar } f_2(x) &= \max(\sigma_{AC}, \sigma_{BC}) \end{aligned}$$

$$\text{sujeto a } \max(\sigma_{AC}, \sigma_{BC}) \leq 10^5;\ 1 \leq y \leq 3$$

Los esfuerzos son calculados por la forma cerrada siguiente

$$\sigma_{AC} = \frac{20\sqrt{(16+y^2)}}{yx_1}; \qquad \sigma_{BC} = \frac{80\sqrt{(1+y^2)}}{yx_2}$$

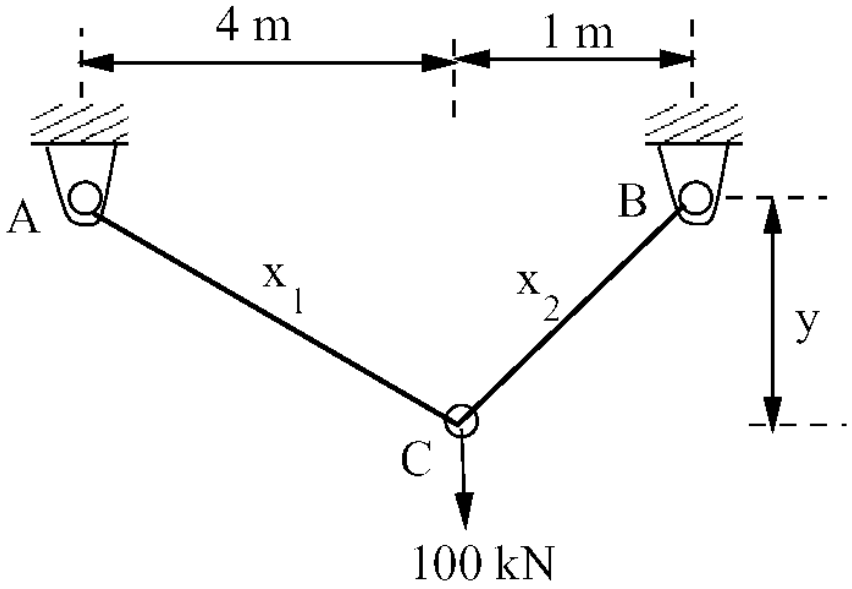

Figura 1. Armadura de dos barras para optimización con dos objetivos y restricciones en esfuerzo máximo permisible

En la Figura 2 puede verse el real frente de Pareto (calculado por enumeración de las funciones objetivo, verificando las restricciones) y el resultado del algoritmo presentado en este trabajo. La solución tiene una dispersión que varía entre (0,004 m^3, 100 000 kPa) y (0,051387 m^3, 8432,740427 kPa). El IS-PAES tiene una forma suave en todo el rango de solución y está sobre el frente de Pareto "verdadero". Presentamos también los resultados de NSGAII[15], NSGA[18] y Palli *et al.*[19].

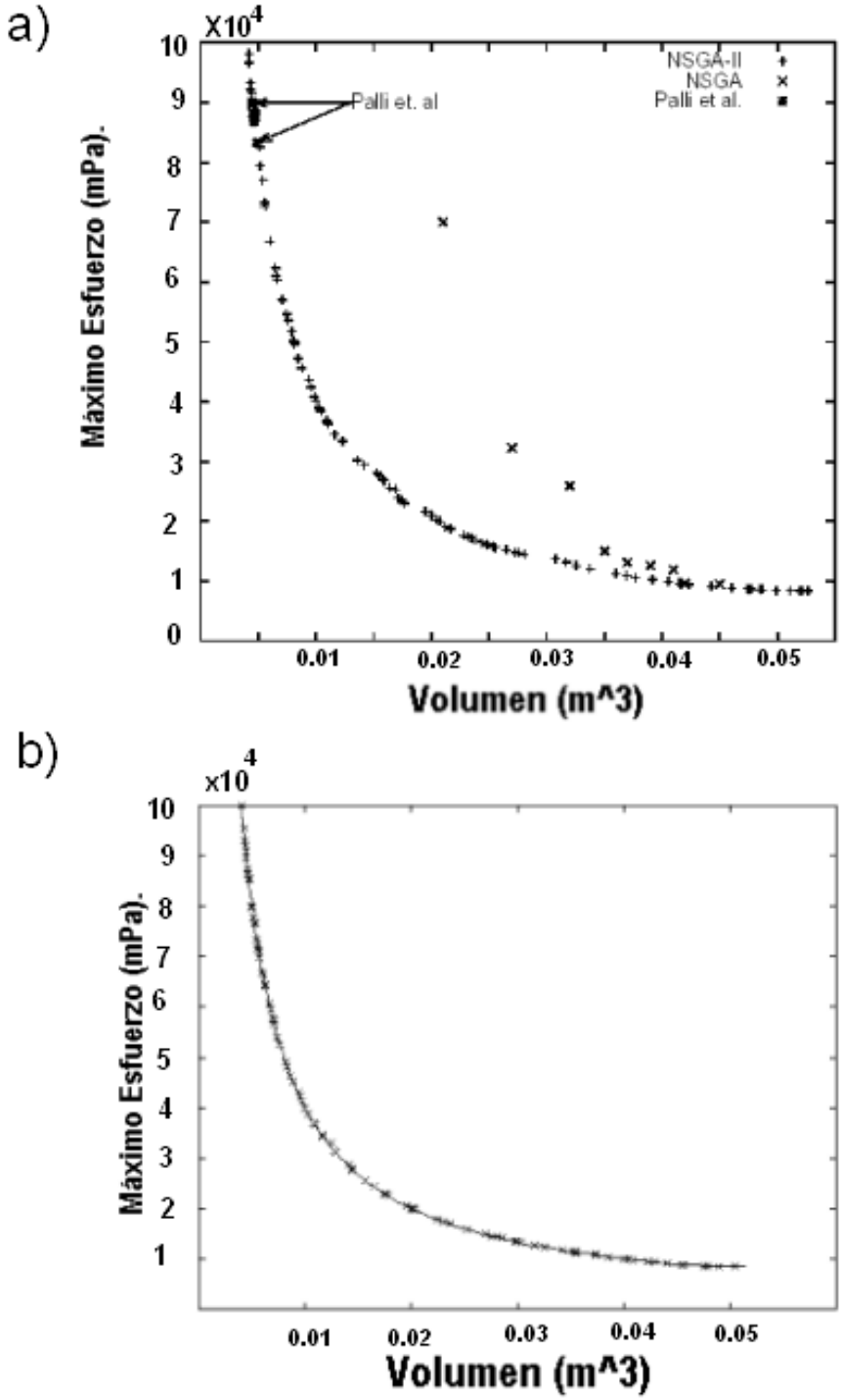

Figura 2. Resultados del Ejemplo 1: a) NSGAII, NSGA, Palli *et al.*; b) resultados del ISPAES y el frente de Pareto real en línea sólida

Ejemplo 2

Se desea optimizar la estructura de 10 barras mostrada en la Figura 3. El objetivo es encontrar la sección transversal de cada una de las barras que forman la estructura, sujeta a restricciones de esfuerzo permisible y desplazamiento, minimizando el peso total de la misma. El peso de la armadura está dado por

$$F(\mathbf{x}) = \sum_{j=1}^{10} \gamma A_j L_j \tag{8}$$

donde $\mathbf{x}$ es una solución candidato; A_j el área de la sección transversal de la barra j; L_j la longitud de la barra j; y γ el peso volumétrico del material. El desplazamiento máximo permitido para cada nodo (horizontal y vertical) es de 5,08 cm. Se tienen 10 restricciones de esfuerzo y 8 de desplazamiento en total. Los valores mínimo y máximo para determinar el área de cada barra son 0,5062 cm^2 y 999,0 cm^2, respectivamente. Los datos restantes para determinar el material de cada barra son: módulo de Young $E = 7{,}3\times10^5$ kg/cm^3; esfuerzo máximo permisible $-1742,11 \leq \sigma_i \leq 1742,11$ kg/cm^2; y peso específico $\gamma = 7,4239 \times 10^{-3}$ kg/cm^3. Las cargas verticales aplicadas en los nodos 2 y 4 de la estructura tienen un valor de $-45454,0$ kg.

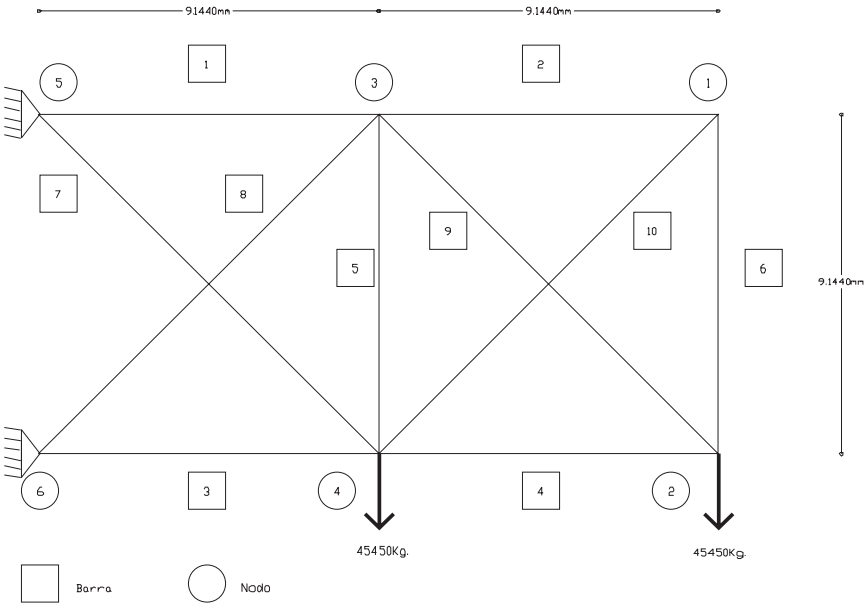

Figura 3. Estructura de 10 barras

Solución con un solo objetivo. En la Tabla VI se muestran los valores para mínimo peso de este problema para diferentes algoritmos y los valores de las áreas de las secciones transversales de las 10 barras aplicando cada algoritmo. En la Tabla VII se presentan los resultados de 30 ejecuciones del IS-PAES, iniciadas aleatoriamente y un análisis estadístico de las mismas. Como puede verse, la desviación estándar es bastante pequeña, lo que indica que el algoritmo llega casi siempre al mismo sitio sin importar su iniciación.

Elemento	I-PAES	GSSA[30]	VGA[31]	ISA[32]
1	190,53	205,17	206,46	269,48
2	0,6466	0,6452	0,6452	79,810
3	146,33	134,20	151,62	178,45
4	95,07	90,973	103,23	152,90
5	0,6452	0,6452	0,6452	70,390
6	3,0166	0,6452	0,6452	10,260
7	47,677	55,487	54,84	147,87
8	129,826	127,75	129,04	14,710
9	133,282	133,56	132,27	156,06
10	0,6452	0,6452	0,6452	87,740
V (cm^3)	801624,5	805777	833258	1313131
Peso (kg)	5951	6186	6186	9750

Tabla VI. Resultados de mínimo peso para el Ejemplo 2

Ejemplo 2 (peso en kg)	
Mejor	5951
Peor	6036,83057
Media	6032,30511
Desviación estándar	15,3775575
Mediana	6034,79443
Soluciones factibles	30

Tabla VII. Resultados estadísticos para el Ejemplo 2

En la Figura 4 puede verse el comportamiento del algoritmo al minimizar el peso de la estructura para una ejecución aleatoria. Puede verse que antes de las 30 000 evaluaciones de la función objetivo (en este caso minimizar el peso), se mantiene constante y en el entorno del valor óptimo. Este número de evaluaciones es bastante competitivo con los de otras estrategias evolutivas y muestra el comportamiento del algoritmo para optimizar este tipo de estructuras.

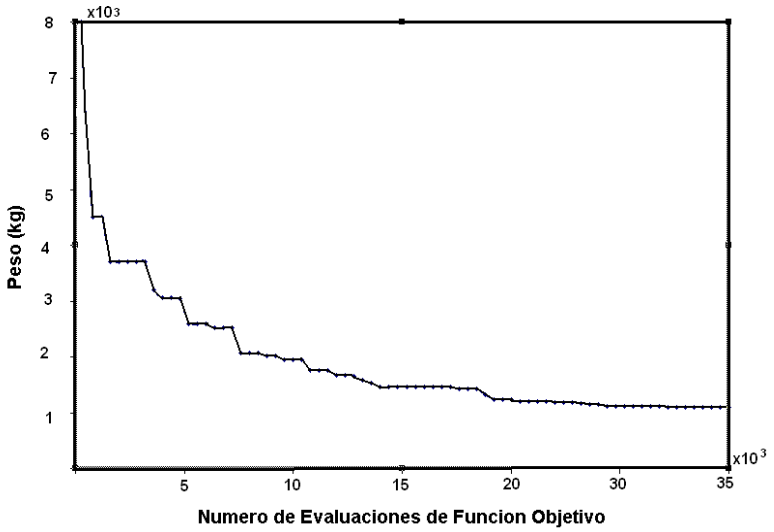

Figura 4. Gráfica para mostrar la convergencia del ISPAES en la optimización de un objetivo (estructura de 10 barras)

Solución para la optimización con dos objetivos. Para la misma estructura del Ejemplo 2 obtuvimos la mejor solución compromiso para dos funciones objetivo: minimizar el desplazamiento total del nodo 2 (el extremo derecho inferior de la estructura) como primer objetivo y minimizar el peso de la estructura como segundo. El frente de Pareto que resulta de esta optimización puede verse en la Figura 5. En el eje horizontal está el peso (kg) de la estructura y en el vertical el desplazamiento (cm) del nodo 2 de la misma. El 81 % del total de individuos están en la región factible y sobre el frente de Pareto.

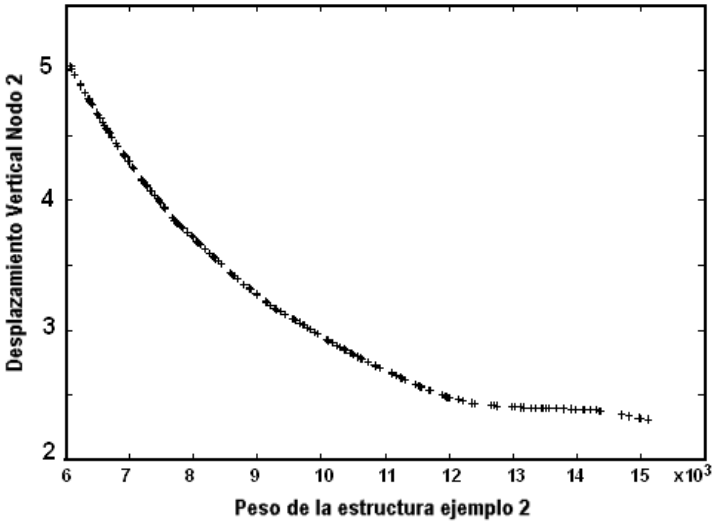

Figura 5. Frente de Pareto para la estructura de 10 barras. En el eje horizontal está el peso (kg) y en el eje vertical el desplazamiento del nodo 2 (cm)

Ejemplo 3

El siguiente ejemplo es la optimización de la estructura de 25 barras en tres dimensiones que se presenta en la Figura 6. Este problema fue originalmente propuesto por Rajeev y Krishamoorthy[23] y consiste en encontrar la sección transversal de cada miembro de la armadura de tal forma que se minimice el peso de la estructura, sujeta a las restricciones de desplazamiento de los nodos y máximos esfuerzos permisibles en cada barra.

Las condiciones de carga se muestran en la Tabla VIII, las coordenadas de los nodos están disponibles en la Tabla IX y el número de grupos de elementos se muestra en la Tabla X. Los datos utilizados son módulo de elasticidad $E = 7,3 \times 10^5$ kg/cm^3, peso volumétrico $\gamma = 7,4239 \times 10^{-3}$ kg/cm^3, máximo esfuerzo permisible $-2787,38 \leq \sigma_i \leq 2787,38$ kg/cm^2; y máximo desplazamiento permisible $-0,889 \leq u \leq 0,889$ cm. Este problema tiene 8 variables de diseño, 25 restriccciones en esfuerzo y 18 restricciones en desplazamiento. El peso de la estructura está dado por

$$F(\vec{x}) = \sum_{j=1}^{25} \gamma A_j L_j \tag{9}$$

donde x es la probable solución; A_j la sección transversal del miembro j-ésismo; y L_j la longitud del j-ésimo elemento.

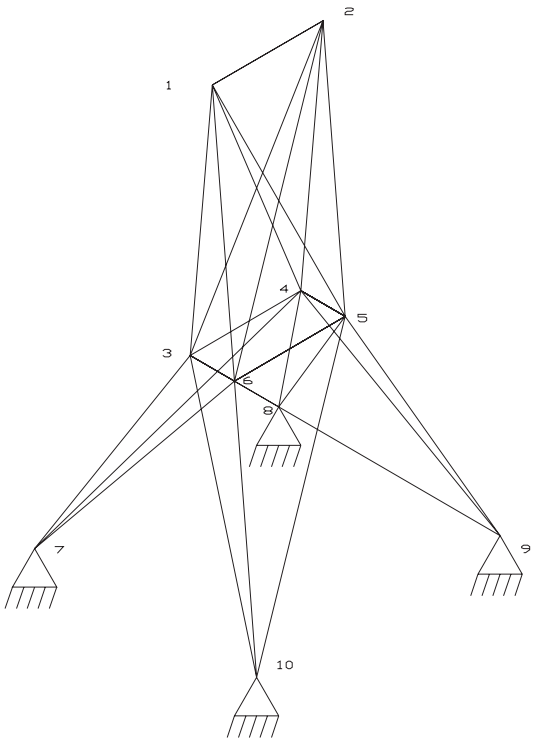

Figura 6. Estructura tridimensional de 25 barras del Ejemplo 3

Nodo	F_x (kg)	F_y (kg)	F_z (kg)
1	4540	-4540	-4540
2	0	-4540	-4540
3	227	0	0
4	272,4	0	0

Tabla VIII. Fuerzas aplicadas en la estructura del Ejemplo 3

Nodo	X (cm)	Y (cm)	Z (cm)
1	-95,25	0	508,0
2	95,25	0	508,0
3	-95,25	95,25	254,0
4	95,25	95,25	254,0
5	95,25	-95,25	254,0
6	-95,25	-95,25	254,0
7	-254,0	254,0	0
8	254,0	254,0	0
9	254,0	-254,0	0
10	-254,0	-254,0	0

Tabla IX. Coordendas de los nodos de la estructura del Ejemplo 3

Número de grupo	Barra
1	1-2
2	1-4, 2-3, 1-5, 2-6
3	2-5, 2-4, 1-3, 1-6
4	3-6, 4-5
5	3-4, 5-6
6	3-10, 6-7, 4-9, 5-8
7	3-8, 4-7, 6-9, 5-10
8	3-7, 4-8, 5-9, 6-10

Tabla X. Conectividades de los elementos por grupos para el Ejemplo 3

Solución para un sólo objetivo. En la Tabla XI se muestran los valores para mínimo peso de este problema para diferentes algoritmos y los valores de las áreas de las secciones transversales, resultado de la aplicación de cada algoritmo. En la Tabla XII se presentan los resultados de 30 ejecuciones del IS-PAES que fueron iniciadas aleatoriamente y un análisis estadístico de las mismas. Como puede verse, la desviación estándar es pequeña, lo que indica que el algoritmo llega casi siempre al mismo sitio sin importar su iniciación. En la Figura 7 se muestra el comportamiento del algoritmo en una ejecución al azar. Puede verse que en las primeras iteraciones el algoritmo hace una búsqueda exhaustiva y los pesos que obtiene son muy altos. Alrededor de las 10 000 evaluaciones de la función objetivo empieza a converger monótonamente y aproximadamente 30 000 evaluaciones de la función objetivo está en el orden de magnitud óptimo.

Resultados comparativos del Ejemplo 3					
Variables	IS-PAES	Coello[40]	Chao[41]	CONMIN[42]	NEWSUMT[42]
x_1 (plg^2)	0,1030	0,1303	0,0100	0,1660	0,0100
x_2 (plg^2)	0,1013	0,1201	2,0415	2,0170	1,9850
x_3 (plg^2)	3,5594	3,4834	3,0011	3,0260	2,9960
$_x4$ (plg^2)	0,1045	0,1102	0,0100	0,0870	0,0100
x_5 (plg^2)	1,9140	1,6583	0,0100	0,0970	0,0100
x_6 (plg^2)	0,7775	0,8373	0,6836	0,6750	0,6840
x_7 (plg^2)	0,1379	0,1172	1,6248	1,6360	1,6670
x_8 (plg^2)	3,9864	4,0900	2,6716	2,6690	2,6620
V (plg^3)	4675,4	4700,9	5450,3	5484,7	5451,7

Tabla XI. Resultados de mínimo peso para el Ejemplo 3

Ejemplo 3 (peso en kg)	
Mejor	568,8
Peor	583,573181
Media	583,080742
Desviación estándar	2,69720152
Mediana	583,573181
Soluciones factibles	30

Tabla XII. Resultados estadísticos para Ejemplo 2

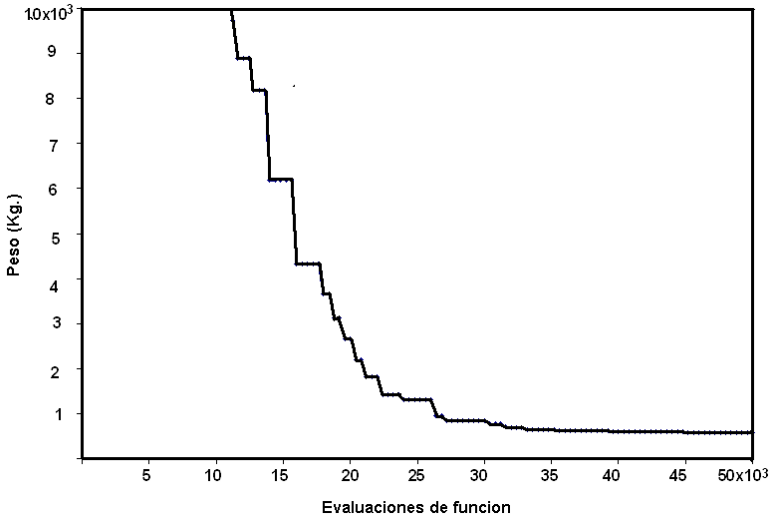

Figura 7. Gráfico para mostrar la convergencia del ISPAES en la optimización de la estructura 3D con 25 barras y un objetivo

Solución para dos objetivos. Se obtuvo la mejor solución compromiso para dos funciones objetivo: minimizar el desplazamiento total del nodo 1 (uno de los extremos superiores de la estructura) como primer objetivo y el peso de la estructura como segundo. El frente de Pareto que resulta de esta optimización puede verse en la Figura 8, donde en el eje horizontal se grafica el peso (kg) y en el eje vertical el desplazamiento del nodo 1 (cm). El 100 % del total de individuos están en la región factible y sobre el frente de Pareto.

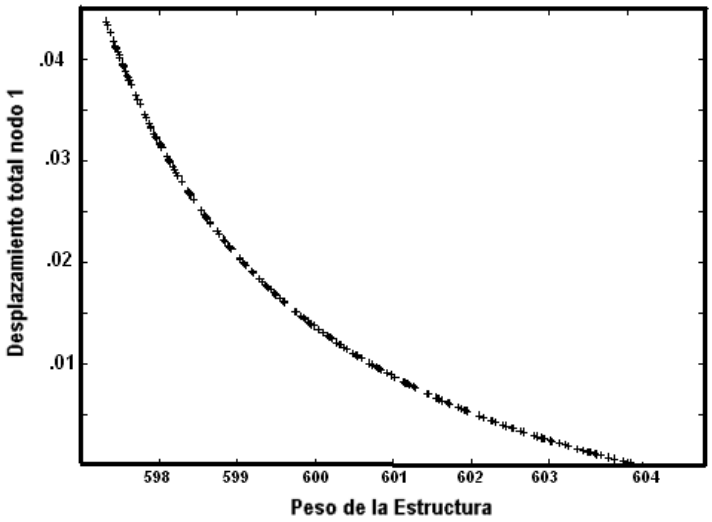

Figura 8. Frente de Pareto para la estructura de 25 barras. En el eje horizontal está el peso (kg) y en el eje vertical el desplazamiento total del nodo 1 (cm)

Ejemplo 4

El siguiente problema de optimización es minimizar el peso de una armadura plana de 49 barras, que por facilidad se considera simétrica (25 diferentes tipos de elementos), propuesta por Galante[43], la cual se muestra en la Figura 9. La idea es encontrar el área de la sección transversal de cada miembro de la armadura, de tal forma que se minimice el peso total de la misma, sujeto a condiciones de restricción en los esfuerzos y en los desplazamientos máximos en la estructura. El peso de la armadura está dado por $F(\mathbf{x}) = \sum_{j=1}^{49} \gamma A_j L_j$, donde A_j es el área de la sección transversal del j-ésimo miembro, L_j la longitud correspondiente de la barra y γ el peso volumétrico del material.

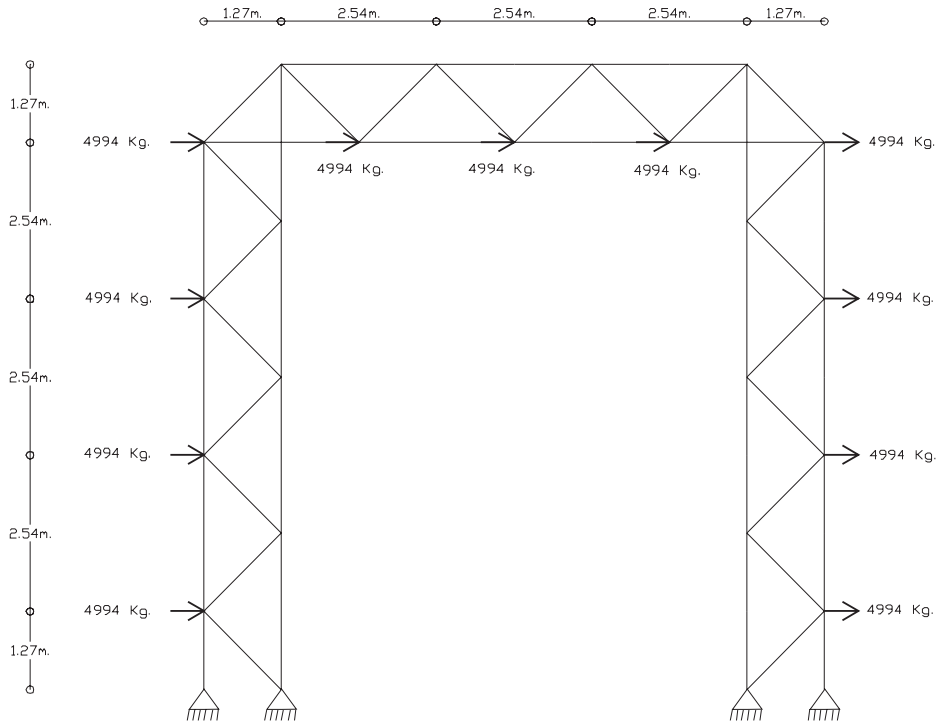

Figura 9. Problema de optimización de una armadura plana de 49 barras

Usamos el catálogo de Altos Hornos de México, S.A., AHMSA[44], con 65 posibles secciones transversales que pueden utilizarse en el diseño. Otros datos relevantes son módulo de Young $= 2,1 \times 10^6$ kg/cm^3; máximo esfuerzo permisible $3500 \leq \sigma_i \leq 3500$ kg/cm^2; $\gamma = 7,4250 \times 10^{-3}$; y una carga horizontal de 4994,00 kg aplicada en los nodos 3, 5, 7, 9, 12, 14, 16, 19, 21, 23, 25 y 27, tal como se muestra en la Figura 9. Resolvemos el problema para tres casos con un solo objetivo

1. **Caso 1. Solamente restricciones en esfuerzo:** esfuerzo máximo permisible de -3500 $\leq \sigma_i \leq$ 3500 kg/cm^2. En este caso se tiene un total de 49 restricciones y una función objetivo.
2. **Caso 2. Restricciones en esfuerzo y desplazamiento:** esfuerzo máximo permisible -3500 $\leq \sigma_i \leq$ 3500 kg/cm^2 y un desplazamiento máximo por nodo de 10 cm. En este caso tenemos 72 restricciones y una función objetivo.

Algoritmo	Peso promedio (kg)
IS-PAES	610
SA[30]	627
GA50[30]	649
GSSA50[30]	619
GSSA5[30]	625

Tabla XIII. Resultado promedio de optimización de armadura de 49 barras, Caso 1

Algoritmo	Peso promedio (kg)
IS-PAES	725
SA[30]	737
GA50[30]	817
GSSA50[30]	748
GSSA5[30]	769

Tabla XIV. Resultado promedio de optimización de armadura de 49 barras, Caso 2

3. **Caso 3. Problema del mundo real:** el diseño considera fuerzas de tracción y de compresión en el diseño de cada barra, así como el peso propio de la estructura. El esfuerzo máximo permisible es de $-3500 \leq \sigma_i \leq 3500$ kg/cm^2, y el desplazamiento máximo por nodo de 10 cm. Un total de 72 restricciones y una función objetivo.

Algoritmo	Peso promedio (kg)
IS-PAES	2603
SA[30]	2724
GA50[30]	2784
GSSA50[30]	2570
GSSA5[30]	2716

Tabla XV. Resultado promedio de optimización de armadura de 49 barras, Caso 3

El promedio de 30 ejecuciones del IS-PAES es mostrado en las Tablas XIII, XIV y XV. Comparamos IS-PAES con resultados obtenidos por Botello *et al.*[30] usando otras técnicas heurísticas con función de penalización SA: Simulated Annealing, GA50: Genetic Algorithm con una poblacion de 50, y GSSA: General Stochastic Search Algorithm con poblaciones de 50 y 5. Puede verse claramente que en todos los casos el IS-PAES produce un menor peso en promedio.

Solución para dos objetivos. Para probar el algoritmo en el caso de multiobjetivo con restricciones presentamos en la Figura 10 el frente de Pareto que se obtuvo al optimizar la estructura anterior considerando en el diseño fuerzas de tracción y de compresión en cada barra, así como el peso propio de la estructura (Caso 3). El esfuerzo máximo permisible es de $-3500 \leq \sigma_i \leq 3500$ kg/cm^2 y el desplazamiento máximo de cualquier nodo de $2,2$ cm. En este caso, las dos funciones objetivo a minimizar son el peso de la estructura y el máximo desplazamiento horizontal en el nodo superior.

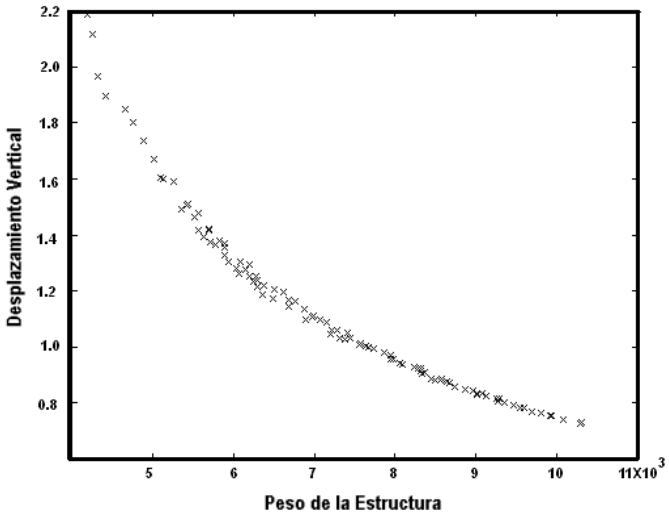

Figura 10. Frente de Pareto para la estructura plana de 49 barras. En el eje horizontal está la función de peso de la estructura y en el vertical está el desplazamiento de la esquina superior derecha de la armadura

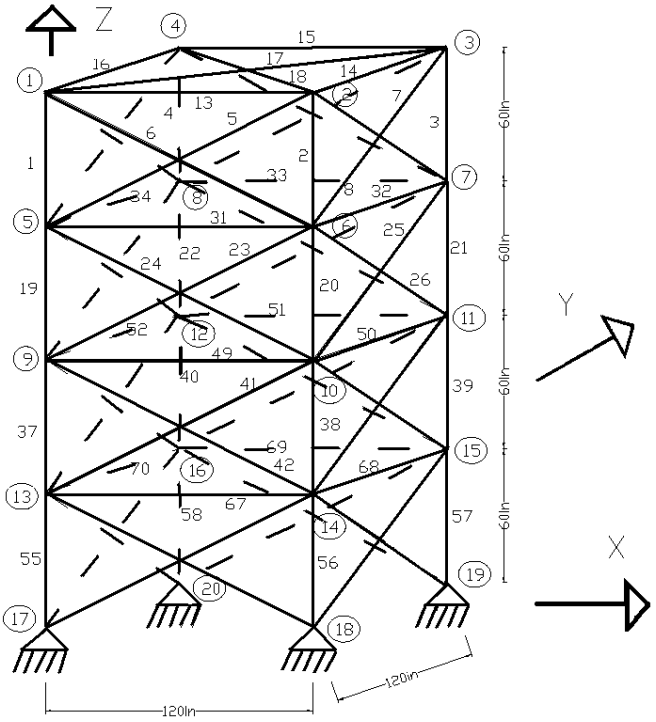

Figura 11. Estructura tridimensional de 72 barras

Ejemplo 5

Diseño de una estructura tridimensional de 72 barras sujeta a dos diferentes casos de carga y dieciseis variables de diseño independientes (Figura 11). Todos los nodos de la estructura tienen una restricción a desplazamiento de $\Delta \leq 0,635$ cm en direcciones x e y. Todas las barras tienen un esfuerzo máximo permisible de $-1759,25 \leq (\sigma_a)_i \leq 1759,25$ kg/cm^2, $i = 1, 2 \ldots 72$. El valor mínimo de la sección transversal es de $0,254$ cm$^2 \leq A_i$, $i = 1, 2 \ldots 72$. Las propiedades de los materiales son módulo de elasticidad $7,031 \times 10^6$kg/cm^2, y peso volumétrico de $2,77 \times 10^{-3}$ kg/cm^3. El primer caso de carga consiste en colocar una carga puntual en el nodo 1, con 2270 kg en dirección del eje x, 2270 kg en dirección del eje y y -2270 kg en dirección del eje z. El segundo caso de carga consiste en aplicar cuatro cargas puntuales en los nodos 1, 2, 3 y 4 con un valor de -2270 kg en dirección del eje z.

Número de grupo	Barras
1	A_1-A_4
2	A_5-A_{12}
3	A_{13}-A_{16}
4	A_{17}-A_{18}
5	A_{19}-A_{22}
6	A_{23}-A_{30}
7	A_{31}-A_{34}
8	A_{35}-A_{36}
9	A_{37}-A_{40}
10	A_{41}-A_{48}
11	A_{49}-A_{52}
12	A_{53}-A_{54}
13	A_{55}-A_{58}
14	A_{59}-A_{66}
15	A_{67}-A_{70}
16	A_{71}-A_{72}

Tabla XVI. Conectividades de los elementos por grupos para el Ejemplo 5

El diseño en este caso debe contemplar las condiciones más desfavorables en esfuerzo y desplazamiento para ambos casos de carga. Este problema es de un solo objetivo.

Los resultados de este problema pueden verse en la Tabla XVII, donde se comparan con resultados de otros autores y en la Tabla XVIII, donde se hace un análisis estadístico de 30 ejecuciones del IS-PAES.

Algoritmo	Pesos mínimos (kg)
IS-PAES	172,02
Venkayya[33]	173,06
Gellatly[34]	179,77
Renwei[35]	172,36
Schmit[36]	176,44
Xichengy [37]	172,90
GAOS[38]	173,94

Tabla XVII. Resultados de mínimo peso para el Ejemplo 5

Ejemplo 5 (peso en kg)	
Mejor	172,02
Peor	172,09
Media	172,05
Desviacion estándar	0,015
Mediana	172,04
Soluciones factibles	30

Tabla XVIII. Resultados estadísticos para el Ejemplo 5

También se realizó la optimización utilizando el catálogo de secciones de acero *Altos Hornos de México, S.A.*, AHMSA[44], con 65 diferentes secciones transversales, las cuales pueden ser indistintamente seleccionadas para realizar la optimización con las mismas propiedades materiales descritas anteriormente. En el Caso 1 se consideró solamente como restricción el esfuerzo permisible, por lo que se redujo el peso de la estructura al no considerar la restricción en desplazamiento. En el Caso 2 se consideró, además, la restricción en desplazamiento; en este caso el peso es mayor que en el problema original, pues ahora estamos trabajando con un catálogo.

Ejemplo 5 (peso en kg) Caso 1	
Mejor	92,3295
Peor	92,3295
Media	92,3295
Desviación estándar	0,0
Mediana	92,3295
Soluciones factibles	30

Tabla XIX. Resultados promedio de optimización de armadura 3D de 25 barras, Caso 1

Ejemplo 5 (peso en kg) Caso 2	
Mejor	192,7194
Peor	193,4353
Media	192,9098
Desviación estándar	0,3060
Mediana	192,7194
Soluciones factibles	30

Tabla XX. Resultados promedio de optimización de armadura 3D de 25 barras, Caso 2

Ejemplo 5 (peso en kg) Caso 3	
Mejor	630,400
Peor	640,3640
Media	633,2354
Desviación estándar	2,7371
Mediana	632,9665
Soluciones factibles	30

Tabla XXI. Resultados promedio de optimización de armadura 3D de 25 barras, Caso 3

Para el Caso 3 se consideró el diseño, adicionalmente a las restricciones en esfuerzo y desplazamiento permisible, considerando efectos de pandeo local en elementos sujetos a fuerza de compresión según criterios del AHMSA, por lo que los pesos son mucho mayores que en el problema original.

Ejemplo 6

Diseño del domo de acero de la Figura 12 utilizando el catálogo mexicano de AHMSA[44]. La armadura tiene siete variables independientes, como puede verse. Todos los nodos tienen una restricción a desplazamiento de $\Delta \leq 2$ cm en la direccion del eje z. Las propiedades de los materiales son módulo de elasticidad $2,1 \times 10^6$ kg/cm^2, y esfuerzo permisible de $-2750 \leq \sigma_i \leq 2750$ kg/cm^2.

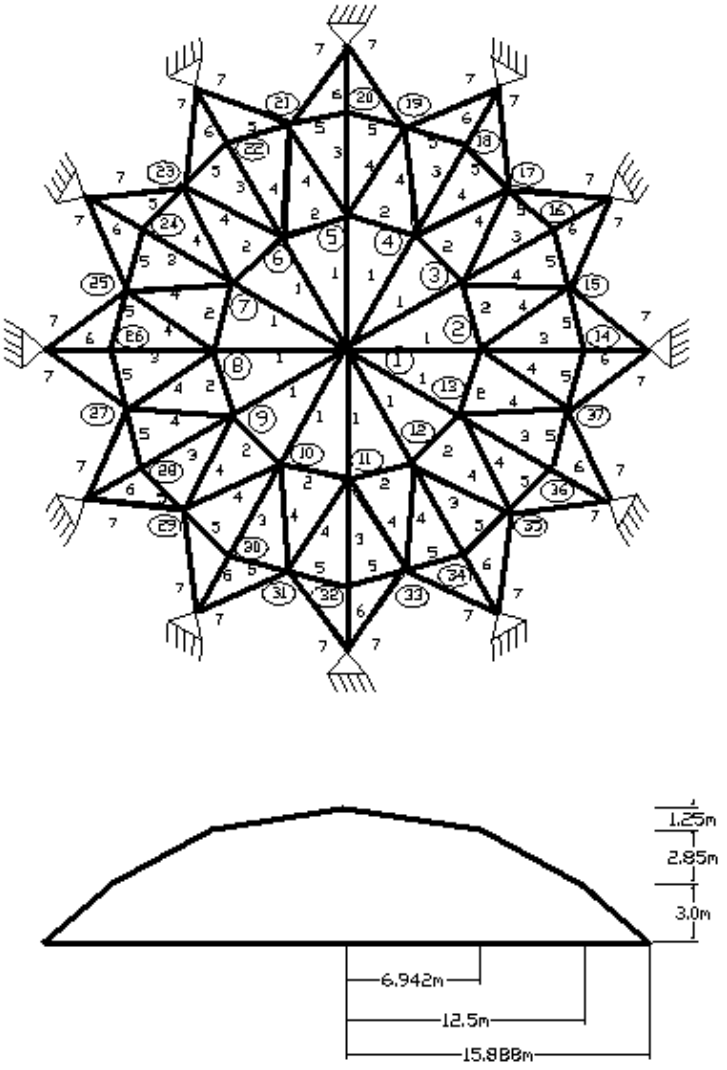

Figura 12. Optimización del domo de acero

Las cargas son puntuales aplicadas en dirección del eje z con diferentes magnitudes: en el nodo 1, con -500 kg, en los nodos 17, 23, 29, 35 con -40 kg, en los nodos 16, 18, 22, 24, 28, 30, 34, 36 con -120 kg. Todos los otros nodos -200 kg. En el primer caso, solamente se han considerado restricciones en esfuerzo y desplazamiento. En el segundo, además de las restricciones en esfuerzo y desplazamiento, se tomaron en cuenta para el diseño de los elementos estructurales el pandeo local provocado por las fuerzas de compresión y se incluyó el peso propio de la estructura (este caso corresponde a un diseño real). En las Tablas XXII y XXIII presentamos los resultados promedios de 30 ejecuciones del IS-PAES para optimizar con un solo objetivo, o sea, minimizar el peso de la estructura.

Ejemplo 6 (peso en kg) Caso 1	
Mejor	703,57
Peor	703,57
Media	703,57
Desviación estándar	0,0
Mediana	703,57
Soluciones factibles	30

Tabla XXII. Resultados promedio de optimización del domo de acero, Caso 1

Ejemplo 6 (peso en kg) Caso 2	
Mejor	13642,33
Peor	13651,93
Media	13644,56
Desviación estándar	4,1304
Mediana	13642,33
Soluciones factibles	30

Tabla XXIII. Resultados promedio de optimización del domo de acero, Caso 2

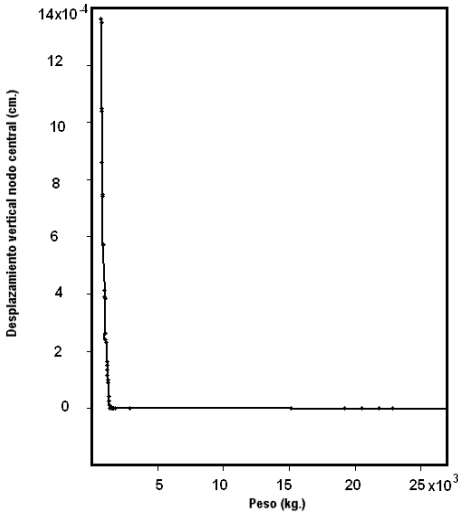

Figura 13. Frente de pareto para el domo de acero. Peso en kg y desplazamiento en cm

Solución para dos objetivos. En el caso de multiobjetivo con restricciones, presentamos en la Figura 13 el frente de Pareto que se obtuvo al optimizar el Caso 2. En este caso, las dos funciones objetivo a minimizar son el máximo desplazamiento vertical del nodo central del domo y el peso de la estructura.

CONCLUSIONES

Proponemos un nuevo algoritmo de optimización multiobjetivo, donde el método de selección utiliza el concepto de dominancia de Pareto y se incluyen de forma natural las restricciones. IS-PAES determina automáticamente la región donde se localiza el óptimo, descartando de la búsqueda las regiones no factibles al ir evolucionando el algoritmo entre generaciones. El algoritmo aquí propuesto es más robusto que el original algoritmo PAES cuando se trabaja con un número elevado de funciones objetivo, como se muestra en la referencia 39. En este algoritmo se tiene un ahorro sustancial en el espacio de memoria que utiliza para almacenar la malla, y la eficiencia computacional al operar es mejor que en el original PAES. La forma de manipular las restricciones reduce drásticamente la complejidad del cálculo de la dominancia de Pareto. Los experimentos demuestran una buena dispersión del frente de Pareto.

AGRADECIMIENTOS

El primer y segundo autor agradecen el apoyo del proyecto CONACyT No. P40721-Y, el segundo autor agradece el apoyo del proyecto CONCyTEG No. 03-02-K118-037 y el último autor agradece el apoyo del proyecto CONACyT No. 34201-A.

REFERENCIAS

1 C.A. Coello Coello, "Constraint-handling using an evolutionary multiobjective optimization technique", *Civil Engineering and Environmental Systems*, Vol. **17**, pp. 319–346, (2000).

2 C.A. Coello Coello, "Treating constraints as objectives for single-objective evolutionary optimization", *Engineering Optimization*, Vol. **32**, Nº 3, pp. 275–308, (2000).

3 C.A. Coello Coello, D.A. van Veldhuizen y G.B. Lamont, "Evolutionary algorithms for solving multi-objective problems", Kluwer Academic Publishers, New York, (2002).

4 C.A. Coello Coello y E. Mezura-Montes, "Handing constrains in genetic algorithms using dominance-based tournaments", In *Proceedings of the Fifth International Conference on Adaptive Computing Design and Manufacture (ACDM 2002)*, Vol. **5**, I.C. Parnee (ed.), University of Exeter, Devon, UK, abril 2002, Springer-Verlag, pp. 273–284, (2002).

5 A.E. Smith y D.W. Coit, "Constraint handiling techniques-penalty funtions", In *Handbook of Evolutionary Computation*, Cap. C 5.2, T. Back, D.B. Fogel y Z. Michalewicz (ed.), Oxford University Press and Istitute of Physics Publishing, (1997).

6 Z. Michalewicz y M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems", *Evolutionary Computation*, Vol. **4**, Nº 1, pp. 1–32 (1996).

7 J.D. Knowles y D.W. Corne. "Approximating the nondominated front using the pareto archived evolution strategy", *Evolutionary Computation*, Vol. **8**, Nº 2, pp. 149–172, (2000).

8 P.D. Surry y N.J. Radcliffe, "The COMOGA method: constrained optimization by multiobjective genetic algorithms", *Control and Cybernetics*, Vol. **26**, Nº 3, pp. 391–412, (1997).

9 D. Goldberg, "Genetic algorithms in search, optimization and machine learning", Addison-Wesley Publishing Company, Reading, MA, (1989).

10 J.D. Shaffer, "Multiple objective optimization with vector evaluated genetic algorithm", In *Genetic algorithms and their applications: proceedings of the first international conference on genetic algorithms*, pp. 93–100, (1985).

11 F.Y. Cheng y X.S. Li, "Generalized center method for multiobjective engineering optimization", *Engineering Optimization*, Vol. **31**, pp. 641–661, (1999).

12 T. Bäck. "*Evolutionary algorithms in theory and practice*", Oxford University Press, New York, (1996).

13 P.D. Surry, N.J. Radcliffe, I.D. Boyd, P.D. Surry, N.J. Radcliffe e I.D. Boyd, "A Multi-objective approach to constrained optimization of gas supply networks: the COMOGA method", In *Evolutionary Computing*, AISB Workshop, Selected Papers, Lecture Notes in Computer Science, T.C. Fogarty (ed.), Sheffield, UK, Springer-Verlag, pp. 166–180, (1995).

14 E. Zitzler, K. Deb y L. Thiele, "Comparison of multiobjective evolutionary algorithms: empirical results", *Evolutionary Computation*, Vol. **8**, N° 2, pp. 173–195, (2000).

15 K. Deb, A. Pratap, S. Agarwal y T. Meyarivan, "A fast and elitist multi-objective genetic algorithm-NSGA-II", *KanGAL Report Number 2000001*, Indian Institute of Technology, Kanpur, India, (2000).

16 J. Horn, N. Nafpliotis y D.E. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization", In "*Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, Vol. **1**, Piscataway, New Jersey, IEEE Service Center, pp. 82–87, (1994).

17 J.D. Knowles y D.W. Corne, "The Pareto archived evolution strategy: a new baseline algorithm for multiobjective optimization", In *1999 Congress on Evolutionary Computation*, Washington D.C., IEEE Service Center, pp. 98–105, (1999).

18 N. Srinivas y K. Deb, "Multiobjective function optimization using nondominated sorting genetic algorithms", *Evolutionary Computation*, Vol. **2**, N° 3, pp. 221–248, (1995).

19 N. Palli, S. Azaram, P. McCluskey y R. Sundararajan, "An interactive multistage e-inequality constraint method for multiple objectives decision making", *ASME Journal of Mechanical Design*, Vol. **120**, N° 4, pp. 678–686, (1999).

20 I.C. Parmee y G. Purchase. "The development of a directed genetic search technique for heavily constrained desig spaces", In *Adaptative computing in egineering design and control-'94*, I.C. Parmee (ed.), Plymouth, UK, University of Plymouth, pp. 97–102, (1994).

21 C.M. Fonseca y P.J. Fleming, "Genetic algorithms for multiobjective optimization: formulation, discussion and generalization", In *Proceedings of the Fifth International Conference on Genetic Algorithms*, S. Forrest (ed.), San Mateo, California, University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers, pp. 416–423, (1993).

22 S. Koziel y Z. Michalewicz, "Evolutionary algorithms, homomorphous mappings and constrained parameter optimization", *Evolutionary Computation*, Vol. **7**, N° 1, pp. 19–44, (1999).

23 S. Rajeev y C.S. Krishamoorthy, "Genetic algorithms-based methodologies for design optimization of trusses", *Journal of Structural Engineering*, Vol. **123**, N° 3, pp. 350–358, (1997).

24 E. Camponogara y S.N. Talukdar, "A genetic algorithm for constrained and multiobjective optimization", In *3rd Nordic Workshop on Genetic Algorithms and Their Applications (3NWGA)*, J.T. Alander (ed.), Vaasa, Finland, University of Vaasa, pp. 49–62, (1997).

25 R. Tapabrata, T. Kang, y S.K. Chye, "An evolutionary algorithm for constrained optimization", In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2000)*, D. Whitley *et al.* (eds.), San Francisco, California, Morgan Kaufmann, pp. 771–777, (2000).

26 R. Tapabrata y K.M. Liew, "A swarm metaphor for multiobjective design optimization", *Engineering Optimization*, Vol. **34**, N° 2, pp. 141–153, (2002).

27 V. Chankong y Y.Y. Haimes, "Multiobjective decision making: theory and methodology", In *Systems Science and Engineering*, A.P. Sage (ed.), North Holland, (1983).

28 F. Jiménez, J.L. Verdegay y A.F. Gómez-Skarmeta, "Evolutionary techiniques for constrained multiobjective optimization problems", In*Proceedings of the 1999 GECCO Conference. Workshop programs*, A.S. Wu (ed.), pp. 115–116, (1999).

29 F. Jiménez, A.F. Gómez-Skarmeta y G. Sánchez, "How evolutionary multi-objective optimization can be used for goals and priorities based optimization", In *Primer Congreso Español de Algoritmos Evolutivos y Bioinspirados (AEB'02)*, E. Alba *et al.* (eds.), Mérida, España, Universidad de Extremadura, (2002).

30 S. Botello, J.L. Marroquín, E. Oñate y J. van Horebeek, "Solving structural optimization problems with genetic algorithms and simulated annealing", *International Journal for Numerical Methods in Engineering*, Vol. **45**, pp. 1069–1084, (1999).

31 S. Rajeev y C.S. Krishamoorthy, "Genetic algorithms–based methodologies for design optimization of trusses", *Journal of Structural Engineering*, Vol. **123**, N° 3, pp. 350–358, (1997).

32 D.H. Acckley, " An empirical study of bit vector function optimization", In *Genetic algorithms and simulated annealing*, D. Lawrence (ed.), Norgan Kaufmann Publishers, Los Altos Calif., pp. 170–271.

33 V.B. Venkayya, "Design of optimun structures", *Computers & Structures*, Vol. **1**, pp. 265–309, (1971).

34 R.A. Gellatly y L. Berke, "Optimal structural design", AFFDL-TR-70-165, (1971).

35 X. Renwei y L. Peng, "Structural optimization based on second orden approximations of functions and dual theory", *Computer Methods in Applied Mechanics and Engineering*, Vol. **65**, pp. 101–4, (1987).

36 L.A. Schmit y B. Farshi, "Some approximation concepts for structural synthesis", *AIAA J.*, Vol. **12**, pp. 231–3, (1974).

37 W. Xicheng y M. Guixu, "A parallel iterative algorithm for structural optimization", *Computer Methods in Applied Mechanics and Engineering*, Vol. **96**, pp. 25–32, (1992).

38 F. Erbatur, O. Hasancebi, I. Tutuncu y H. Kilic, "Optimal design of planar and space structures with genetic algorithms", *Computer & Structures*, Vol. **75**, pp. 209–224, (2000).

39 A. Hernández, S. Botello, G. Lizárraga y C. Coello, "IS-PAES: a single and multiple-objective optimization method", Technical Report I-02-19-CC, Centro de Investigaciones en Matemáticas, (2002).

40 C. Coello, "Constraint-handling using an evolutionary multiobjective optimization technique", *Civil Engineering Systems*, Gordon and Breach Science Publishers, Vol. **17**, pp. 319–346, (2000).

41 N.H. Chao, S.J.Fenves y A.W. Westerberg, "Application of reduced quadratic programing technique to structural optimal design", In *New Directions in Optimum Structural Design*, E. Atrek, R.H. Gallagher, K.M. Radsdell y O.C. Zienkiewicz (eds.), Jhon Wiley, New York, (1984).

42 L.A. Schmit y H. Mira, "A new structural analysis/synthesis capability-ACCESS 1", *AIAA Journal*, Vol. **15**, N° 5, pp. 661–671, (1976).

43 M. Galante, "Un algoritmo genético simple para la optimización de estructuras planas articuladas", *Rev. Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería*, Vol. **9**, N° 2, pp. 179–199, (1993).

44 Altos Hornos de México, S.A., "Base de datos para el manual de la industria siderúrgica para la construcción en acero", AHMSA, (1991).