OPTIMIZACIÓN MULTIOBJETIVO DE ESTRUCTURAS, UTILIZANDO ALGORITMOS DE ESTIMACIÓN DE DISTRIBUCIONES E INFORMACIÓN DE VECINDADES

Sergio Ivvan Valdez Peña, Salvador Botello Rionda y Arturo Hernández Aguirre

Comunicación Técnica No I-05-07/09-06-2005
(CC/CIMAT)

Optimización Multiobjetivo de Estructuras, Utilizando Algoritmos de Estimación de Distribuciones e Información de Vecindades

Sergio Ivvan Valdez Peña, Salvador Botello Rionda y, Arturo Hernández Aguirre
Centro de Investigación en Matemáticas A.C.
Departamento de Ciencias de la Computación
A.P. 402, Guanajuato, Gto. 36000, México
e-mail: ivvan,botello,artha@cimat.mx

Mayo 28, 2005

Resumen

En este trabajo se presentan un conjunto de técnicas utilizadas para la optimización multiobjetivo de estructuras (diseño óptimo automatizado), basadas en algoritmos de estimación de distribuciones. Esta nueva propuesta de solución, utiliza el criterio de dominancia de Pareto para el manejo de las restricciones y los objetivos. Los objetivos a optimizar son: el peso mínimo de la estructura y el desplazamiento en nodos especificados. Las restricciones de diseño de las estructuras son: a) no exceder un esfuerzo Von Misses máximo, b) no presentar elementos desconectados por los lados y c) no presentar agujeros pequeños (del tamaño de un elemento).
El algoritmo presentado, utiliza información de las vecindades para mejorar la exploración hacia las soluciones más probables, y evitar mínimos locales. Se utiliza el método de elemento finito para evaluar las funciones objetivo y restricción (desplazamiento y esfuerzo).

Capítulo 1

Introducción

En el presente capítulo se describe de manera general el alcance y complejidad del problema de optimización de formas, se pone en antecendetes al lector de algunos de los trabajos realizados en este campo y de sus virtudes y deficiencias. Brevemente se expone la técnica utilizada para mejorar las propuestas anteriores, solucionando algunos de los problemas que se presentan comunmente en optimización de formas con algoritmos evolutivos.

1.1 Introducción

Por la complejidad computacional en el problema de optimización de formas se pueden distinguir tres clases, estas son: dimensiones, forma y topológica (ver Richards [13], y Sandgren[14]).

- *La optimización de dimensiones* se refiere a la determinación de dimensiones geométricas específicas para una clase de diseño preseleccionado, tal como el espesor de una placa, el tamaño de un armazón o el radio de un elemento con esfuerzo circular. Esta clase de problemas han estado bajo investigación durante decadas.
- *La optimización de formas* introduce variables de diseño adicionales las cuales permiten el movimiento de la frontera. Debido al incremento en la dificultad relativa con la optimización de tamaño, los cambios geometricos han sido limitados historicamente; sin embargo, esta ha ganado importancia en las industrias aeronautica y automovilistica, como en otras, produciendo mejoras en las turbinas, formas de las laminas del avion y piezas conectoras. La optimización de dimensiones es un subconjunto de la optimización de formas.
- *La optimización topológica* involucra modificaciones tanto topológicas como de forma y dimensiones. Las modificaciones topológicas tratan con ensambles de componentes. Los componentes en el ensamble pueden ser modificados y se pueden agregar componentes, quitar o mover componentes en el ensamble en el intento de generar un diseño mejorado. Relativamente poco trabajo se ha hecho en esta área, no obstante la importancia del concepto.

En el trabajo actual tratamos con el problema de optimización de formas en 2 dimensiones; aunque parece factible abordar el problema de 3 dimensiones utilizando las misma metodología que se ha utilizado en 2. Para tratar con el problema se ha utilizado una técnica relativamente nueva en el campo de los algoritmos evolutivos: los algoritmos de estimación de distribuciones (EDA's, ver Baluja [1], Muhlenbein[10], Pelikan [12] [11], Li [7]), los cuales intentan encontrar una distribución (o conjunto de distribuciones en este caso) cuyas muestras generadas sean las mejores soluciones al problema. En intentos anteriores (Kane [6],Deb [3], Chapman[2]) se han logrado (relativamente) buenos resultados utilizando algoritmos genéticos con representación binaria, para la

optimización de formas. Estos intentos tanto monoobjetivo como multiobjetivo presentan varios problemas que evitan que la mejor forma para un estado de cargas dado sea encontrada. Algunos de los problemas generales mas representativos en estos intentos son:

- Estancamiento en mínimos locales debido a una exploración deficiente.
- Las formas resultantes no tienen todas las caracteristicas deseables, debido a que el modelo de optimización no guía adecuadamente la búsqueda, es decir, muchas veces con pedir que se minimíze el peso no es suficiente. Para encontrar la mejor estructura, los objetivos o restricciones que tienen que ver con esfuerzos o desplazamientos ayudan a dirigir mejor la exploración; también las funciones objetivo y restricción que involucran la forma misma, como el número de objetos o agujeros ayudan al algoritmo a mejorar su poder de decisión por las "mejores estructuras", desde el punto de vista de ingeniería.
- Existe una dependencia de la malla que limita el tipo de formas que se pueden generar, este problema se hace más grave para mallas burdas.

El trabajo actual trata con 2 de los 3 problemas mencionados anteriormente, con una estrategia, que detecta la falta de exploración y la mejora. además, se utiliza un modelo que intenta capturar todas las características deseables de la estructura desde el punto de vista de ingeniería. El último problema relativo a la dependencia de la malla por el momento no se ha tratado, pero se espera en un futuro trabajar con estrategias que nos permitan minimizar esta dependencia.

Capítulo 2

Definición del problema

En este capítulo se explica cual es el problema de optimización a resolver. Se exponen las ecuaciones que modelan las funciones objetivo y restricciones, asi como las variables y lo que representan en el problema físico. La forma en que se defina la representación de la información para construir las soluciones, y en que se modelen las características físicas deseables de las soluciones, impactará de manera importante en la capacidad del algoritmo de encontrar soluciones adecuadas.

2.1 Definición del problema

El problema multiobjetivo es encontrar un conjunto de estructuras con el mínimo peso y el menor desplazamiento en ciertos puntos dados (los puntos de carga), que cumplan con las siguientes restricciones de diseño: no exceder el máximo esfuerzo Von Misses permitido, no permitir mas de un objeto, ni agujeros pequeños. En la sección siguiente se definirá de manera más amplia el significado de estas funciones objetivo y restricciones. Para el diseño óptimo automatizado de las estructuras o la optimización de formas, el usuario entrega el estado de cargas y el espacio de búsqueda inicial. La Figura 2.1, muestra en la izquierda el espacio de busqeda inicial proporcionado por el usuario, y el estado de cargas al que esta sujeto; el lado derecho muestra una de las soluciones sobre el frente de Pareto, que representa la estructura de menor peso, con el mínimo desplazamiento posible para este peso, en el punto de aplicación de la carga.

2.2 Discretización y representación

El espacio de búsqueda inicial se discretiza en elementos finitos (triangulares en este caso). Esto produce una malla que puede ser estructurada o no estructurada. En este trabajo hemos probado con ambas, pero en general no se supone ningún conocimiento de la forma, asi que una malla no estructurada parece mas acorde con la generalidad de los problemas. Cada elemento es representado por una variable binaria, cuando esta variable es igual a 0, representa un hueco o un espacio sin material, del tamaño de un elemento; de otro modo (igual a 1), representa un valor de espesor dado. De esta manera, cada configuración de una estructura puede ser representada por un vector binario de la misma dimensión que el número de elementos. La Figura 2.2 muestra la discretización del espacio y como se pasa de la representación binaria a una forma de alguna estructura.

2.3 Primera función objetivo: Mínimo peso

El primer objetivo es el peso mínimo y esta dado por la Ecuación 2.1, que es la suma de los pesos w_i de cada uno de los elementos que tienen material o están representados por 1 en el vector binario $\hat{x}$.

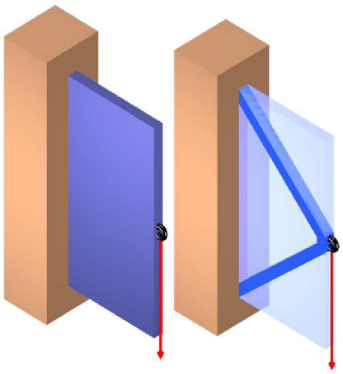

Figura 2.1: Izquierda: Espacio de búsqueda y estado de cargas, derecha: Estructura con mínimo peso y desplazamiento, una de las soluciones no dominadas.

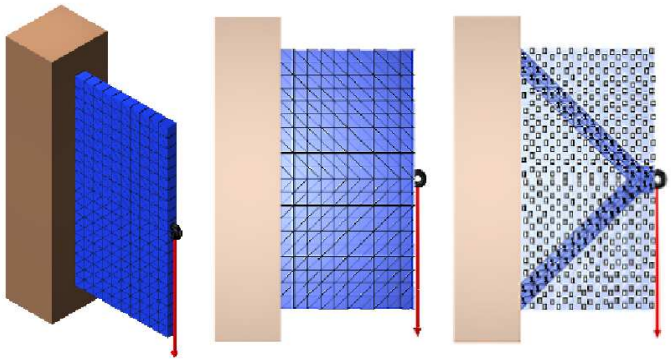

Figura 2.2: Izquierda: Discretización del espacio de búsqueda, derecha: representación de una estructura cualquiera.

Minimizar :

$$W(x) = \sum_{i=0}^{n} w_i x_i \tag{2.1}$$

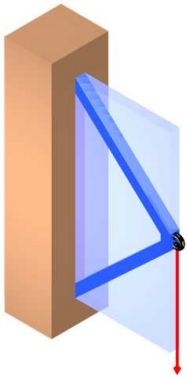

Figura 2.3: Estructura con el mínimo peso en el frente.

2.4 Segunda función objetivo: Mínimo desplazamiento

El segundo objetivo es el mínimo desplazamiento en algunos puntos (nodos), en los ejemplos mostrados se han utilizado los nodos con carga como los puntos donde se quiere minimizar el desplazamiento. La Ecuación 2.2 representa la segunda función objetivo, que es la sumatoria de los desplazamientos en los puntos definidos por el usuario. En todos los ejemplos estos puntos fueron los de aplicación de las cargas.
Minimizar :

$$G(\delta) = \sum_{j=0}^{m} |\delta_j| \tag{2.2}$$

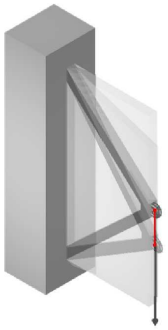

Figura 2.4: Simulación del desplazamiento de una estructura en el punto de carga (linea roja)

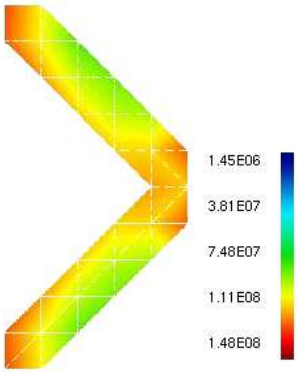

Figura 2.5: Simulación del estado de esfuerzos en una estructura, encontrados por el método del elemento finito

2.5 Primera restricción: Máximo esfuerzo Von Misses Permisible

La primera función restricción esta dada por el esfuerzo máximo Von Misses permisible (ver Malvern [8]), el cual representa la resistencia del material al estado de cargas, y es un criterio estándar de diseño. El esfuerzo Von Misses (o de máxima energía de distorsión) es calculado en cada elemento por el método del elemento finito (ver Zienkiewicz [15]), si este esfuerzo sobrepasa el máximo permisible para el material, aumenta el valor de la restricción, si ningún elemento tiene un valor mayor que este, entonces la restricción vale cero. La restricción se expresa por la Ecuación 2.3.

$$H(x, \rho(\sigma), \gamma(\sigma)) = \left(\sum_{i=0}^{n} \rho_i(\sigma)\right)\left(\sum_{i=0}^{n} x_i \gamma_i(\sigma)\right) \tag{2.3}$$

Donde :

$$\rho_i(\sigma) = \begin{cases} 0 & Si\ (\sigma_M - \sigma_i) \geq 0 \\ 1 & de\ otra\ forma \end{cases} \tag{2.4}$$

y :

$$\gamma_i(\sigma) = \begin{cases} 0 & Si\ (\sigma_M - \sigma_i) \geq 0 \\ (\sigma_i - \sigma_M) & de\ otra\ forma \end{cases} \tag{2.5}$$

En la Ecuación 2.3, la suma de $\rho_i(\sigma)$ cuenta el número de elementos que están esforzados más alla del esfuerzo máximo permisible, x_i es el valor del bit en la i^{esimo} posición del arreglo binario; y $\gamma_i(\sigma)$ es el exceso de esfuerzo en el i^{esimo} elemento, cuando el esfuerzo en el elemento es mayor que el máximo permisible. Note que las Ecuaciónes 2.4 y 2.5 dependen implicitamente de la configuración representada por el arreglo binario $\hat{x}$, debido a que los valores de estas funciones se derivan de la solución por el método del elemento finito, el cual es completamente dependiente de la configuración de la estructura $\hat{x}$.

La Figura 2.5 muestra una simulación del estado de esfuerzos en una estructura, utilizando el método del elemento finito.

2.6 Segunda Restricción: Número de Objetos

La segunda restricción penaliza el número de piezas de las que esta formada la estructura, el modelo supone que las mejores estructuras son las que están echas de una sola pieza. La Figura 2.6 muestra una configuración de una estructura formada por 4 objetos, podemos observar que estos objetos son un conjunto de elementos conectados

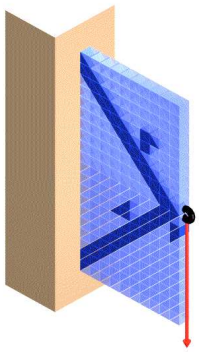

Figura 2.6: Se muestra una configuración con 4 objetos

entre si por los lados. El método del elemento finito puede evaluar las estructuras que estan compuestas por varios objetos mientras estos objetos esten conectados al menos por un vertice, desde el punto de vista de ingeniería los objetos conectados solamente por un vertice no son factibles.

2.7 Tercera Restricción: Agujeros Pequeños

La tercera restricción cuenta el número de agujeros pequeños. Un agujero pequeño se define como un agujero del tamaño de un elemento, y que está rodeado (en sus tres lados para elementos triangulares) por elementos que si tienen material. La Figura 2.7 muestra una configuración con 3 agujeros pequeños.

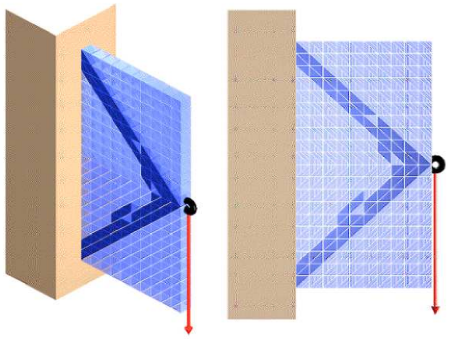

Figura 2.7: configuración que presenta 3 agujeros pequeños

Capítulo 3

Algoritmo Propuesto

En este capítulo se describe una propuesta de algoritmo para la optimización de formas, el algoritmo se basa en la estimación de probabilidades marginales de las variables a optimizar, ignorando las dependencias entre ellas para la estimación de las probabilidades; sin embargo, estas dependencias (o las dependecias más probables) son tomadas en cuenta mediante una regularización de la distribución de probabilidad, que se lleva a cabo cuando se estanca la búsqueda, y usando información de las vecindades.

3.1 Descripción del algoritmo

La Figura 3.1 muestra los principales procedimientos del algoritmo. Primero, se inicializan los vectores de probabilidad $\hat{p}$ utilizando la rutina presentada en la subsección 3.1.1. Después, cada vector de probabilidad p_i genera un subconjunto de la población. Finalmente, todos los individuos factibles no dominados y algunos de los no factibles (*Conjunto actual de Pareto*), se toman para actualizar las distribuciones de probabilidad. Cuando las distribuciones de probabilidad pierden su capacidad de exploración, se realiza un procedimiento de regularizado de las distribuciones, tomando información de las vecindades de cada elemento (ver 3.1.5). Cuando se ha realizado la regularización de la probabilidad, se vacía el *conjunto actual de Pareto*. En cada generación se actualiza el *conjunto conocido de Pareto* para guardar las mejores soluciones encontradas durante todas las generaciones; mientras el *conjunto actual de Pareto* guarda individuos tanto factibles como no factibles, el *conjunto conocido de Pareto* guarda solamente los individuos no dominados factibles, otra diferencia entre el *conjunto actual de Pareto* y el *conjunto conocido de Pareto*, es que el primero guarda no dominados únicamente con respecto a las generaciones desde la última vez que se vacío este conjunto, mientras que el segundo guarda los no dominados con respecto a todas las generaciones realizadas. El pseudocódigo del algoritmo principal se muestra en la Figura 3.2.

3.1.1 Inicialización de la Probabilidad

De acuerdo con el algoritmo mostrado en la Figura 3.3, las distribuciones inciales de probabilidad $\hat{p}$ muy posiblemente generarán individuos factibles con diferentes configuraciones. Por otro lado, los vectores de probabilidad pueden ser inicializados con el valor de 0.5 en cada posición (la máxima varianza en la población inicial) o incluso valores aleatorios; sin embargo, la rutina propuesta ayuda a la convergencia y al costo de evaluación, dando tan buenas soluciones como la incialización de máxima varianza, los valores aleatorios al contrario pueden sesgar la búsqueda y retardar la convergencia del algoritmo sin una ventaja evidente.

Básicamente, la rutina para la inicialización de la probabilidad busca por una configuración factible de bajo peso dentro del espacio de búsqueda. Cuando este individuo es encontrado, la rutina comienza inicializando una distribución de probabilidad que muy posiblemente generará esta estructura, y parte de ahi para inicializar

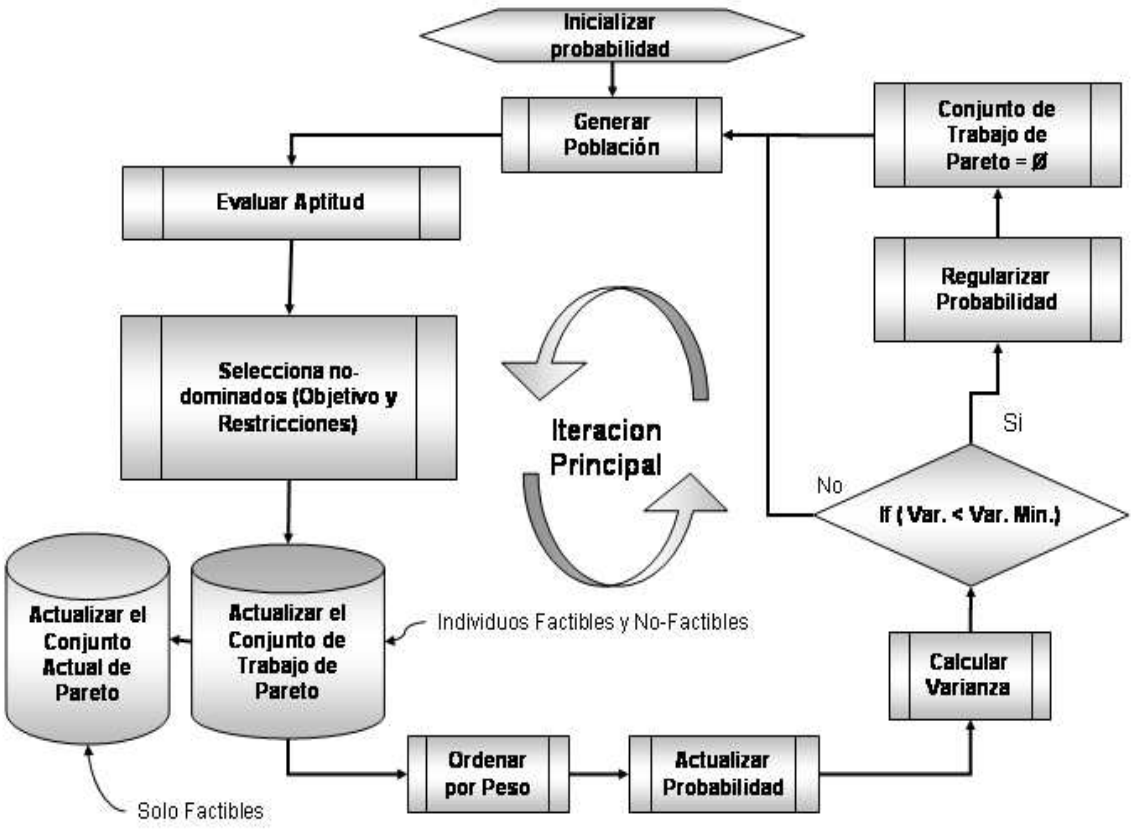

Figura 3.1: Descripción del algoritmo principal (MASO)

```
Insertar los siguientes parámetros:
μ:          Parámetro de suavizado, peso de la diferencia con las vecindades
            (ver [9] y la subsección 3.1.5 en la Ecuación 3.2)
MinVar:     Parámetro de varianza mínima, criterio para realizar el regularizado
LSP,LIP:    Límites superior e inferior de probabilidad
NV:          Número de vectores (distribuciones) de probabilidad.
NIM:         Número de individuos generados por cada vector de probabilidad.
λ:          Parámetro de aprendizaje para la actualización de la probabilidad
            (ver subsección 3.1.4 ).
NG:          Número máximo de generaciones que se procesarán.
NIS:         Número máximo de regularizaciones de la probabilidad.

 Inicialización_de_Probabilidad ();
 k = 0
 i = 0
 Mientras (i < NG & k<NIS)
  {
   Generar_Población ();
   Evaluar_Objetivos_y_Restricciones ();
   Seleccionar_No-Dominados_por_Objetivos_y_Restricciones();
   Seleccionar_No-Dominados_Factibles ();
   Seleccionar_No-Dominados_No-Factibles ();
   Ordenar_por_Peso();
   Actualizar_Distribuciones_de_Probabilidad ();
   Calcular_Varianza ();
   Si (VarianzaCalculada < MinVar)
    {
     Regularizar_Probabilidad ();
     k++;
    } #Fin Si
  i++;
 } #Fin Mientras
```

Figura 3.2: Pseudo-código del Algoritmo Multiobjetivo para Optimización de Formas

```
 T y Z son dos configuraciones de estructuras.
 maxVonMisses y minVonMisses son el máximo y el mínimo esfuerzo
  en T (todos los elementos presentes).

 Para i=1..NoElementos
  T_i=1 #Fin Para NoElementos
 Resolver_VonMisses(T);
 Intervalo = (maxVonMisses-minVonMisses)/(NoElementos)
 umbral = maxVonMisses - Intervalo
Hacer
{
 Para i=1..NoElementos
  {
   Si (EsfuerzoVonMises_i < Umbral)
    Z_i = 0
   Sino
    Z_i = 1
  } #Fin Para NoElementos
  umbral = umbral - Intervalo
} mientras ( EsNofactible(Z) )# Verdadero cuando Z es no factible
 Intervalo = (umbral-VonMisses )/( NoVectors-1 );
 Para i=1..NoVectores
 {
  Para k=2..NoElementos
   {
    Si ( EsfuerzoVonMises_i < umbral)
     VectorProbabilidad_{k,i} = LimInfProb;
    Sino
     VectorProbabilidad_{k,i} = LimInfProb;
   } #Fin Para NoElementos
   umbral = umbral - Intervalo;
 } #Fin Para NoVectores
```

Figura 3.3: Pseudo-código para la inicialización de la probabilidad

```
Para i=1..NoVectoresProbabilidad
 {
  Para j=1..NoIndividuosPorVector
   {
    Para k=1..NoDeElementos
     {
      Si (Aleatorio()< P_{i,k} )
       x_{(i-1)(NoIndividuosPorVector)+j,k} = 1
      Sino
       x_{(i-1)(NoIndividuosPorVector)+j,k} = 0
     } # Fin Para k
   } # Fin Para i
 } # Fin Para j
```

Figura 3.4: Pseudo-código para la generación de la población

las demás con distribuciones que muy posiblemente generarán estructuras más pesadas que la primera, pero permaneciendo dentro de la zona factible.

3.1.2 Generación de la población

El algoritmo utiliza k vectores de probabilidad $\hat{p}$, cada vector genera un subconjunto de la población, entonces, cada bit en $\hat{x}$ es generado por un experimento Bernoulli con probailidad de exito p_i (ver [1]). Todos los vectores generados $\hat{x}$ son la población actual, de donde saldrán los "mejores" individuos que actualizarán las distribuciones de probabilidad. Los individuos son generados de acuerdo al algoritmo en la Figura 3.4.

3.1.3 Selección y manejo de restricciones

La selección de los mejores individuos se realiza utilizando el criterio de dominacia de Pareto, tratando las restricciones como tres objetivos más; una vez que han sido seleccionados de está manera, se separa a los factibles de los no factibles. Los primeros de manera directa para al *conjunto actual de Pareto*, mientras que los segundos seran filtrados nuevamente utilizando el criterio de Pareto, pero ahora solo sobre las restricciones. Este tipo de selección mejora la exploración y extrae información "importante" de los individuos no factibles; pero, al mismo tiempo dirige la población hacia la zona factible.

3.1.4 Actualización de la probabilidad

La actualización de la probabilidad se realiza utilizando los individuos seleccionas conforme se indica en la sección 3.1.3. Estos individuos deben de ser ordenados por el primer objetivo (peso); entonces, cada vector de probabilidad será modificado por un subconjunto de los mejores individuos, de acuerdo con la Ecuación 3.1. Para k vectores de probabilidad, cada nuevo vector de probabilidad se calcula con la siguiente ecuación:

Para j=1..número de Bits

$$P_{l,j}^{t+1} = \lambda P_{l,j}^{t} + (1-\lambda) \sum_{i=(l-1)*Redondea(C_t/k)}^{(l)*Redondea(C_t/k)} x_{i,j}/Redondea(C_t/k) \tag{3.1}$$

Donde C_t es número total de individuos seleccionados; λ es el *tasa de aprendizaje* (*learning rate*,ver [1]) el cual indica que tanto se preserva del conocimiento de la distribución anterior y; cuanto se "aprende" de la

nueva. La función *Redondea*() regresa el entero más cercano al valor real de C_t/k.

3.1.5 Regularización de la probabilidad

Debido al reforzamiento implícito del aprendizaje en las probabilidades de la Ecuación 3.1, las distribuciones de probabilidad tienden a perder su capacidad de exploración a través de las generaciones; entonces, las distribuciones de probabilidad son modificadas utilizando información de las vecindades, donde se supone una alta correlación positiva entre las variables x_i, esta correlación supuesta generalmente es cierta, debido a las implicaciones físicas que representa: es dificil que un elemento con material este junto a varios que no lo tienen, o viceversa. Esta correlación no será verdad en las fronteras, donde el que un elemento tenga material, no implica que su vecino también lo tenga. Por otro lado, al modificar la probabilidad no se toman decisiones deterministas; sino que únicamente se le da a la distribución la capacidad de explorar el espacio en direcciones con altas posibilidades de exito, que generalmente implican expandir o estrechar la estructura, mover las fronteras, o rellenar huecos. La Ecuación 3.2, muestra como las distribuciones de probabilidad se pueden modificar para mejorar la exploración. Básicamente, el funcional $U(f)$ se minimiza para encontrar una nueva distribución de probabilidad, la cual preserva información de los mejores individuos encontrados ,o la mejor distribución encontrada hasta ese momento; pero , trata de explorar en las vecindades, como se muestra en la Figura 3.5.

$$min \quad U(f) = \sum_{j}^{NoElem.} (f_j - g_j)^2 + \mu \sum_{j}^{NoElem.} \sum_{l \in vecinos} (f_j - f_l)^2 \tag{3.2}$$

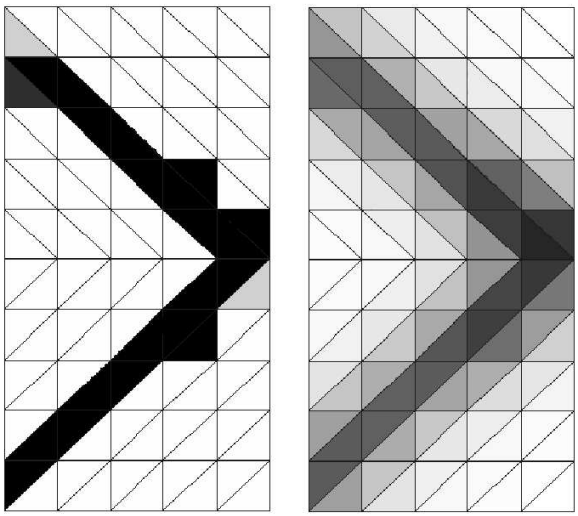

Figura 3.5: Relación entre la estructura y su distribución de probabilidad: (a) Antes de aplicar la regularización, (b) Después de aplicar la regularización.

En la Ecuación 3.2, g es la distribución conocida o la distribución actual (cualquier vector de probabilidad $\hat{p}$) y f es la nueva distribución (el nuevo vector de probabilidad con mejor capacidad de exploración, a través de las vecindades). En la segunda sumatoria f_j es la probabilidad en la j^{esima} posición del vector de probabilidad, y f_l son los elementos vecinos físicamente (normalmente tres para elementos triangulares, cuatro para rectangulares,

```
VarianzaCalculada=0
Para l=1.. NoVectores
 {
 Varianza=0
  Para i=1..NoElementos
   {
   Si (VectorProbabilidad_{l,i} ≤ LimInfProb
      && VectorProbabilidad_{l,i} ≥ LimSupProb)
    Varianza=Varianza+1;
  } #Fin Para NoElementos
   Si (Varianza > VarianzaCalculada)
    VarianzaCalculada =Varianza;
  }#Fin Para NoVectores
```

Figura 3.6: Pseudo-código del algoritmo de cálculo de la varianza

etc.). Minimizando $U(f)$ la función f será la mejor función que preserva la forma de g; pero, que minimiza al mismo tiempo la diferencia entre los valores de probabilidad de las vecindades. Esta diferencia se pondera por el parámetro μ (que para los experimentos realizados se le han dado valores entre 0.35 y 0.5). La Figura 3.5 muestra la probabilidad antes y después de el proceso de regularización utilizando la Ecuación 3.2. Más información acerca de el proceso se encuentra en [9](Marroquin). Como se describe en la subsección 3.1 el *conjunto actual de Pareto* debe de ser vacío después de la regularización de la probabilidad. Esto obedece a un sesgo muy fuerte que ocurre en al búsqueda, si se mantienen las soluciones encontradas durante todo el proceso.

3.1.6 Cálculo de la varianza

En la subsección 3.1.5, se abordo el problema de la exploración pobre, y se dio una técnica para mejorarla. Ahora, queda la cuestión de cuando debe de ser aplicada la regularización de la probabilidad. Se propone una medida para detectar la pérdida de la exploración. Básicamente, se cuenta el número de valores en cada vector de probabilidad que se encuentran dentro de los límites de probabilidad (no muy cercanos a cero ni a uno); se toma el valor mayor (el vector con mayor número de valores dentro de los límites de probabilidad), y este valor se considera una medida de la varianza de las muestras que pueden ser producidas. Cuando este valor es menor que un umbral dado por el usuario, se regularizan las distribuciones de probabilidad. Se sugiere que el valor de este parámetro sea aproximadamente el 12% del número de elementos del espacio de búsqueda. El algoritmo en la Figura 3.6 describe el cálculo de la medida de varianza.

Capítulo 4

Métricas

Dentro de este capítulo se presentan algunas metrícas de desempeño, que ayudan a mostrar el comportamiento del algoritmo y su capacidad para encontrar las soluciones. Las métricas miden convergencia y diversidad.

4.1 Métricas de desempeño

A fin de mostrar el comportamiento y capacidad del algoritmo, se utilizaron dos metricas de desempeño propuestas por Deb y Jain en [4]. Esta métricas miden convergencia y diversidad en las soluciones encontradas.

En [4], sugieren que las propiedades que deben de ser consideradas para las métricas de los MOEAS (Multi-Objective Evolutionary Algorithm), deben de ser las siguientes:

1. La métrica debe de tomar un valor entre cero y uno en un sentido absoluto. Para que esta pueda ser comparada entre generaciones, una escala absoluta de una métrica entre cero y uno permitirá evaluar el cambio en el valor de la métrica entre una generación y otra.

2. El valor objetivo (o deseado) de la métrica (calculado para un idealmente convergido o diversificado conjunto de puntos) debe ser conocido.

3. La métrica debe proveer un incremento o decremento monotono en su valor, cuando la población mejore o se deteriore en poco. Esto también ayudará a evaluar que tan superior es un conjunto de aproximación con respecto de otro.

4. La métrica debe ser escalable a cualquier número de objetivos. Sin embargo, esta no es una propiedad absolutamente necesaria, pero si se tiene, seguramente será conveniente para evaluar problemas referentes a la escalabilidad de los MOEAs en términos del número de objetivos.

5. La métrica podría ser computacionalmente barata, sin embargo esta no es una condición estricta que deba tener.

4.1.1 Métrica de convergencia

Un conjunto de referencia P^* se determina de la unión de varias corridas independientes (del *conjunto conocido de Pareto* optenido por cada corrida). Esta métrica básicamente mide la distancia normalizada entre el frente que se está midiendo $F^{(t)}$, y el conjunto de referencia P^*. Toma un valor en el intervalo $[0, 1]$; cercano a 0 significa que el valor de convergencia $C(P^{(t)})$ es mejor, el valor en $C(P^{(N)})$ (al final de la corrida), indica que tan lejos se encuentra el frente medido del conjunto de referencia, que es grosso modo la efectividad del algoritmo

para encontrar las mejores soluciones (suponiendo que el conjunto de referencia tiene las mejores soluciones), el medir $C(P^{(t)})$ durante las generaciones nos indica como el conjunto evoluciona y se acerca cada vez más a el conjunto de referencia. La métrica de convergencia se mide como se muestra a continuación:

1. Identificar el conjunto no dominado $F^{(t)}$ de $P^{(t)}$ (una población).

2. Para cada punto i en $F^{(t)}$, calcule la menor distancia euclideana normalizada a P^* como en la Ecuación 4.1, f_k^{max} y f_k^{min} son el máximo y el mínimo valor de la función en la k^{esimo} función objetivo en P^*.

$$d_i \overset{|P^*|}{min}_{j=1} \sqrt{sum_{k=1}^{M} \left(\frac{f_k(i) - f_k(j)}{f_k^{max} - f_k^{min}} \right)} \tag{4.1}$$

3. Calcular la métrica de convergencia promediando la distancia normalizada para todos los puntos en $F^{(t)}$:

$$C(P^{(t)}) = \frac{sum_{i=1}^{|F^{(t)}|} d_i}{|F^{(t)}|} \tag{4.2}$$

Deb y Jain en [4] proponen normalizar la métrica de convergencia entre el valor máximo (usualmente $C(P^{(0)})$): $\hat{C}(P^{(t)}) = C(P^{(t)})/C(P^{(0)})$. Esta normalización no fue calculada para este trabajo. Se utilizó en cambio otra normalización para medir la convergencia. Para que el conjunto medido y el conjunto de referencia se puedan comparar se necesita que el conjunto a ser medido tenga por lo menos el mismo número de puntos que el conjunto de referencia; de otra manera, podemos tener mediciones muy sesgadas, debido a que en un principio podemos conocer puntos iguales a los del conjunto de referencia (distancia cero; por lo tanto $C(P^{(t)}) = 0$) pero que sean muy pocos, lo cual no indica realmente la convergencia del algoritmo, esto se solucionó igualando la distacia normalizada a 1 para los puntos que no se han encontrado aún (la diferencia de los puntos en el frente de referencia y el frente a medir).

4.1.2 Métrica de diversidad

Usando el mismo conjunto de referencia que en la subsección 4.1.1, se calcula la métrica de diversidad propuesta por Deb y Jain en [4].A continuación se describe el cálculo de la métrica de diversidad, tal como la proponen Deb y Jain en [4].
Basado en un estudio de Farhang-Mehr y Azarm (ver [5]) sugirió una técnica basada en entropía.
La idea esencial es que los puntos no dominados de cada generación se proyecten en un hiperplano conveniente, perdiendo dimensionalidad de los puntos. El plano se divide en un número de pequeñas celdas (o $(M-1)-dimensional$ cajas). Dependiendo de si cada celda contiene un punto no dominado o no, la métrica de diversidad es definida. Si todas las celdas tienen por lo menos un punto, la mejor medida de diversidad posible es alcanzada (con respecto a el número de celdas escogido). Si algunas celdas no están representadas por un punto no dominado, entonces la diversidad es pobre. Los parámetros requeridos del usuario son la dirección del coseno del plano de referencia, el número de celdas (G_i) en cada una de las $(M-1)$ dimensiones, y el conjunto de puntos objetivo (o de referencia) P^*. El procedimiento es el siguiente:

1. De $P^{(t)}$, encuentre el conjunto $F^{(t)}$ que son los no dominados por P^*.

2. Para cada celda indexada por $(i, j, ...)$, calcular los siguientes dos arreglos:

$$H(i,j,...) = \begin{cases} 1, & Si\ la\ celda\ tiene\ un\ punto\ representativo\ en\ P*. \\ 0. & De\ otra\ forma. \end{cases} \tag{4.3}$$

$$h(i,j,...) = \begin{cases} 1, & Si\ H(i,j,...)\ y\ la\ celda\ tienen\ un\ punto\ representativo\ en\ F^{(t)}. \\ 0. & De\ otra\ forma. \end{cases} \tag{4.4}$$

CD-R.desktop

3. Asignar un valor $m(h(i,j,...))$ a cada celda dependiendo de su función $h()$ y la de sus vecinos. De la misma manera, calcular $m(H(i,j,...))$ usando $H()$ para los puntos de referencia.

4. Calcular la métrica de diversidad promediando los valores de $m()$ para $h()$ con respecto a aquellos para $H()$:

$$D(P^{(t)}) = \frac{\sum\limits_{i,j...\ para\ H(i,j,...)\neq 0} m(h(i,j,...))}{\sum\limits_{i,j...\ para\ H(i,j,...)\neq 0} m(H(i,j,...))} \tag{4.5}$$

En el caso simple, el valor de la función $m()$ para una celda puede ser calculado utilizando su $h()$ y la de dos vecinos por dimensión. Con un conjunto de tres valores binarios consecutivos de $h()$, existen un total de 8 posibilidades. Cualquier función debe ser asignada manteniendo en mente lo siguiente:

- Un 111 es el mejor distribuido y un 000 es el peor.
- Un 010 o un 101 significan un patrón periódico con una buena dispersión y puede ser mejor valuado que un 110 o un 011. Por ejemplo, el valor de arriba puede estar cubriendo un 50% de las celdas pero con una amplia dispersión (tal como 1010101010) mejor que otro conjunto que tenga la misma covertura pero con una menos dispersión (tal como 1111100000).
- Un 110 o un 011 deben de valer más que un 001 o un 100, por que cubren más celdas.

Basado en las observaciones de arriba, los valores de $m()$ para $h()$ mostrados en la Figura 4.1 son sugeridos:

$h(...j-1...)$	$h(...j...)$	$h(...j+1...)$	$m(h(...j...))$
0	0	0	0.00
0	0	1	0.50
1	0	0	0.50
0	1	1	0.67
1	1	0	0.67
0	1	0	0.75
1	0	1	0.75
1	1	1	1.00

Figura 4.1: Valores sugeridos para m()

Mayores detalles con respecto de esta métrica pueden ser encontrados en [4].

Capítulo 5

Experimentos

Los experimentos mostrados en este capítulo utilizan las métricas del capitulo anterior y se muestran de manera gráfica las soluciones encontradas, todos los experimentos presentan buenas soluciones, se compara también con algunas soluciones encontradas por búsqueda exhaustiva, para un problema de pocas variables.

Se muestran 3 casos de estudio, el primero es un problema con un espacio de búsqueda pequeño del cual se pueden conocer algunas soluciones en el *conjunto de Pareto real*, lo cual sirve para comparar la efectividad del algoritmo. El segundo experimento tiene las mismas condiciones que el primero, pero con un espacio de búsqueda mayor; además, se calculan las métricas antes mencionadas para mostrar el comportamiento y capacidad del algoritmo. El último experimento aumenta la complejidad del problema, al hacer más complejo el estado de cargas, utilizar mallas no estructuradas, y aumentar la dimensionalidad (número de elementos) del problema.

5.1 Experimento 1

El primer experimento está definido por una placa simplemente soportada en las dos esquinas inferiores, con una carga puntual en la parte superior-media de la misma. Se encontraron a través de búsqueda exahustiva algunas soluciones en el *conjunto de Pareto real* (las soluciones simétricas), lo cual es posible para un problema de pocas variables binarias (pocos elementos), para mallas más refinadas esto se vuelve imposible desde el punto de vista de costo computacional, dado que la cantidad de soluciones posibles es 2^N siendo N el número de elementos, en el espacio físico de búsqueda. La Figura 5.1 muestra el estado de cargas y las dimensiones de la placa.

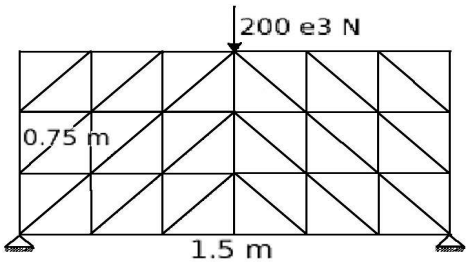

Figura 5.1: Dimensiones y estado de carga para el experimento 1

$Módulo\ de\ Young$	$2.1 \times 10^{11}\ Pa$
$MódulodePoisson$	0.2
$Espesor$	$0.10m$
$Esfuerzo\ máximo\ permissible$	$250 \times 10^{6}\ Pa$

Figura 5.2: Propiedades del material del experimento 1

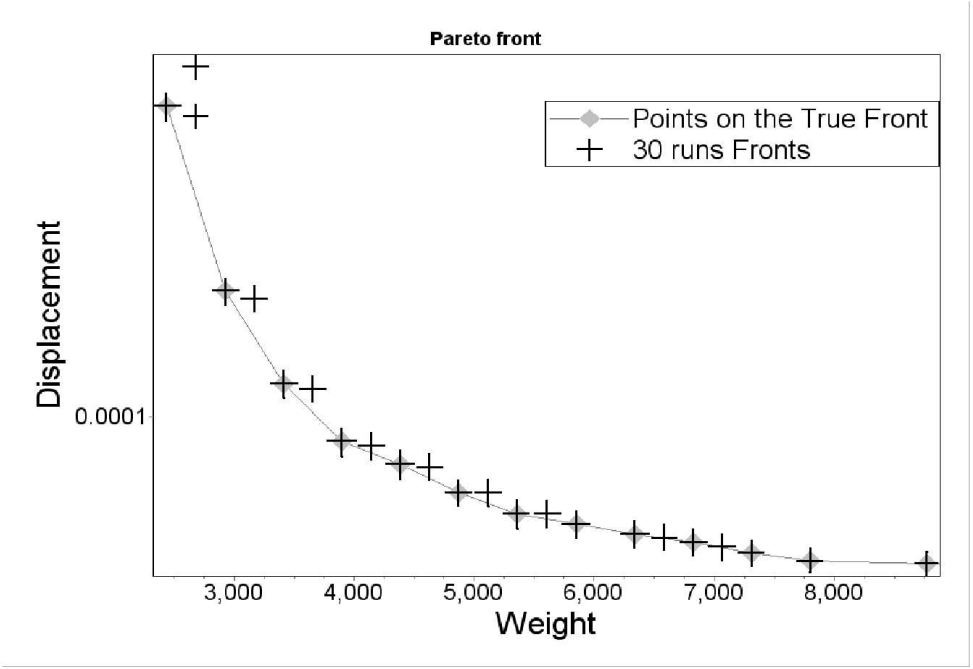

Figura 5.3: Frente Verdadero con diamantes, y frentes de las 30 corridas con cruzes.

5.1.1 Comparación con el Conjunto Verdadero de Pareto

Para el experimento 1 se cálcularon las soluciones simétricas que pertenecen al conjunto verdadero de Pareto. Se realizaron 30 corridas y se compararon contra las soluciones conocidas. Se tienen 13 soluciones conocidas en el conjunto verdadero. El promedio de estas mismas soluciones encontradas por el algoritmo en las corridas realizadas es de 12.6667 (de las 13 conocidas), con una desviación estándar de 0.479463 . El promedio de las soluciones totales encontradas es de 20.667 con una desviación estándar de 1.3218. El promedio de las soluciones encontradas que son dominadas por alguna solución de las conocidas en el conjunto verdadero es de 0.066667 con una desviación estándar de 0.253708 en la Figura 5.3, se puede observar el $frente\ verdadero\ de\ Pareto$ y los puntos no dominados encontrados en las 30 corridas.

5.1.2 Experimento 1. Métrica de convergencia

Se calculó la métrica de convergencia $C(P^{(N)})$ para el experimento 1, utilizando un frente de referencia que contiene las 13 estructuras conocidas del conjunto verdadero de Pareto y las no dominadas de las 30 corridas independientes. La media de la métrica calculada para todas las 30 corridas en la última generación es de 0.06783 (mejor mientras está más cerca de 0) con una desviación estándar de 0.05586 . En la Figura 5.5 se

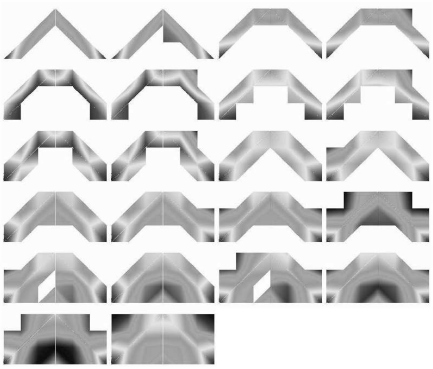

Figura 5.4: Estructuras resultantes de una corrida típica.

muestra una gráfica con los valores de la métrica de convergencia para las 30 corridas y en la Figura 5.6 se muestra el comportamiento en la métrica de convergencia a través de las generaciones para una corrida típica del algoritmo.

5.2 Experimento 2

El experimento 2 tiene las mismas condiciones de carga que el experimento 1, como se muestra en la Figura 5.7; pero, en este caso el espacio de búsqueda se ha discretizado en 144 elementos. Dado que el frente de Pareto no es conocido, las comparasiones se han llevado a cabo contra un frente de referencia, resultado de 30 corridas independientes.

5.2.1 Experimento 2. Frente de referencia

La Figura 5.9 muestra el frente de referencia y todos los individuos no dominados encontrados en 30 corridas. Aunque, algunos de ellos son dominados por el frente de referencia (en rombos), no fueron dominados en su corrida independiente.

5.2.2 Experimento 2. Métrica de convergencia

La Figura 5.11 muestra el valor $C(P^{(t)})$ calculado para la última generación. Como se puede observar, el valor esta muy cercano a 0,con media de 0.008312 y desviación estándar de 0.007347 lo cual implica una buena convergencia. La Figura 5.12 muestra el comportamiento de la métrica de convergencia en una corrida típica.

5.3 Experimento 3

El experimento 3 consiste en el diseño hipotético de una bicicleta. Las métrica de convergencia y diversidad se calcularon utilizando un frente de referencia de 12 corridas independientes. Se encontraron más de 1000 soluciones en cada corrida (en promedio 1580.83). Se uso una malla no estructurada con 307 elementos triangulares para evaluar la aptitud con el método del elemento finito. La Figura 5.13 muestra el espacio inicial y las condiciones de carga, la Figura 5.14 muestra las propiedades del material y las coordenadas de los puntos que definen el espacio inicial de búsqueda. Los resultados preliminares de 12 corridas en la Figura 5.15 (con asteriscos), el frente de referencia en rombos, y algunas estructuras para una corrida se muestran en

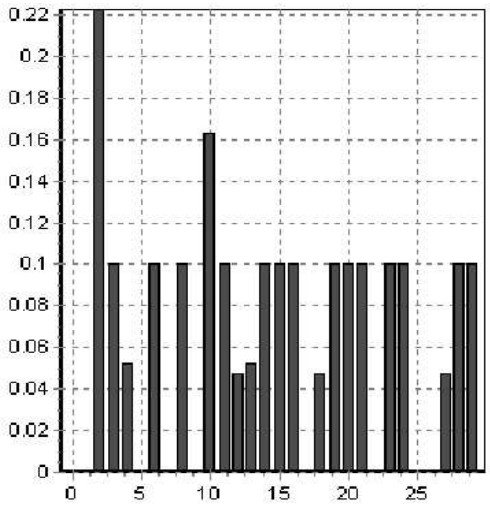

Figura 5.5: Valores de la métrica de convergencia $C(P^{(N)})$ para 30 corridas en la última generación.

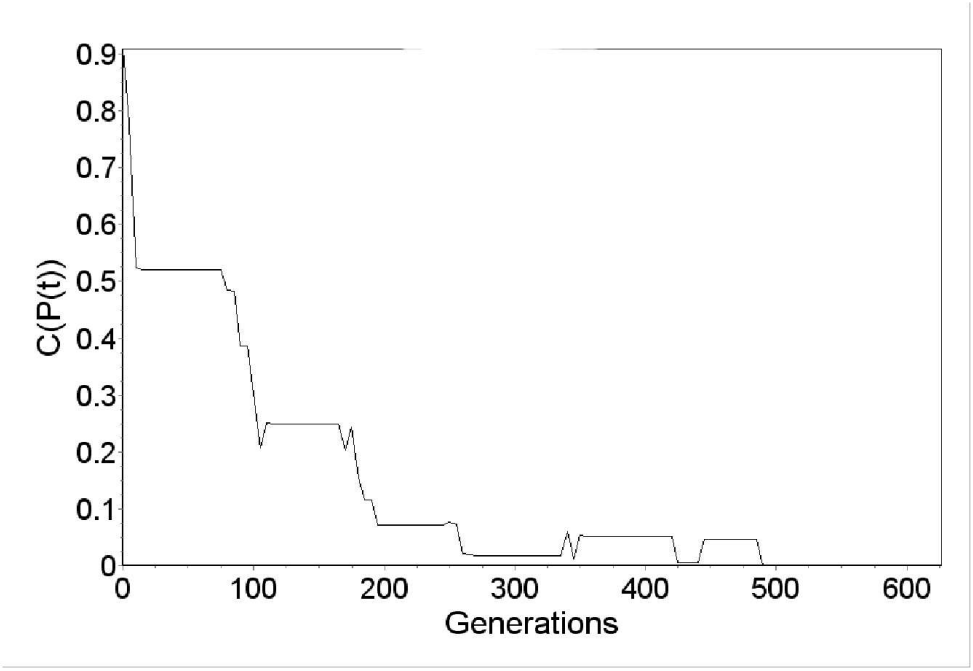

Figura 5.6: Comportamiento de la convergencia para una corrida típica del algoritmo.

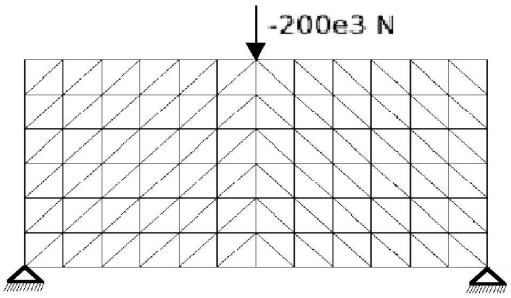

Figura 5.7: Viga simplemente apoyada, discretizada en 144 elementos.

Módulo de Young	$2.1 \times 10^{11}\ Pa$
MódulodePoisson	0.2
Espesor	$0.10m$
Esfuerzo máximo permissible	$250 \times 10^{6}\ Pa$

Figura 5.8: Propiedades del material para el experimento 2.

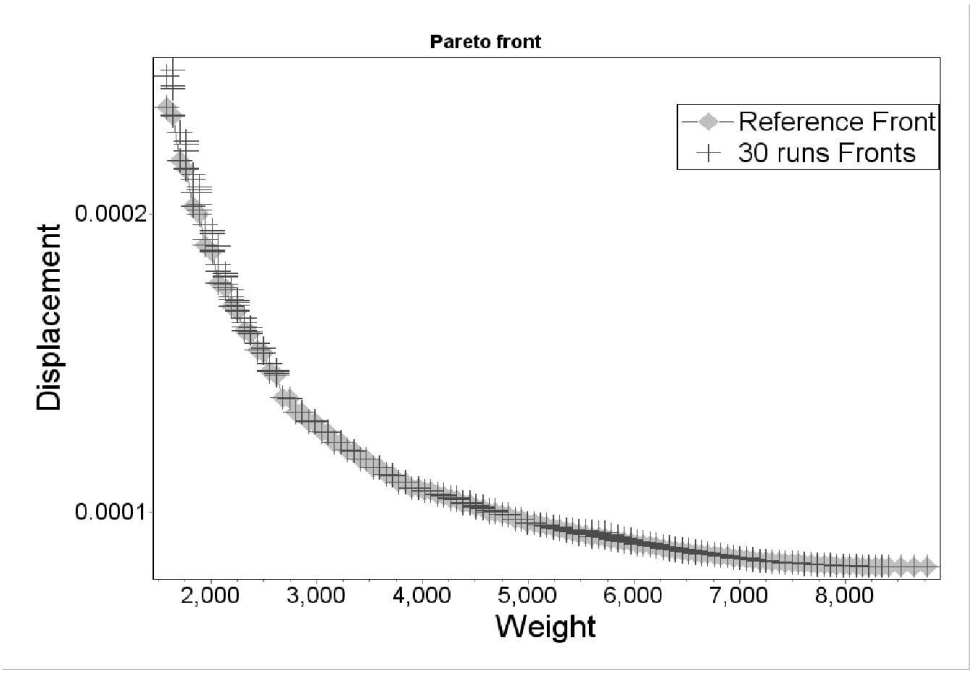

Figura 5.9: Los rombos representan al frente de referencia y las cruzes a los no dominados en cada corrida independiente.

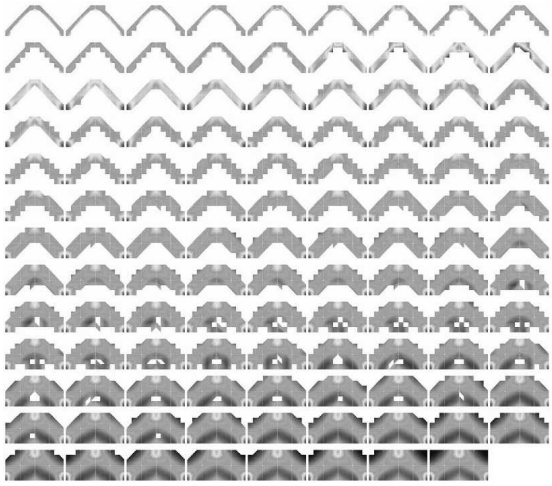

Figura 5.10: Estructuras encontradas en una corrida típica del algoritmo.

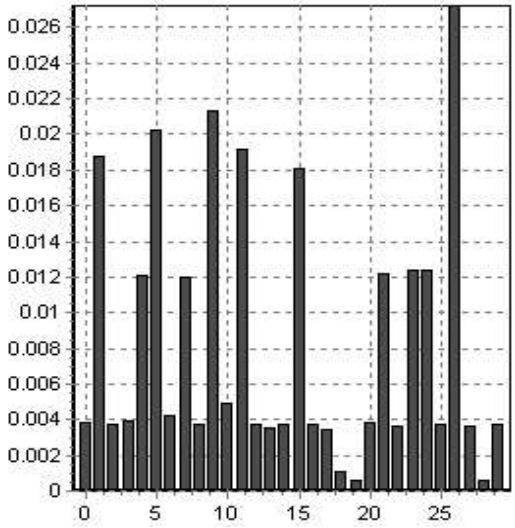

Figura 5.11: Valores de convergencia para la última generación en 30 corridas independientes.

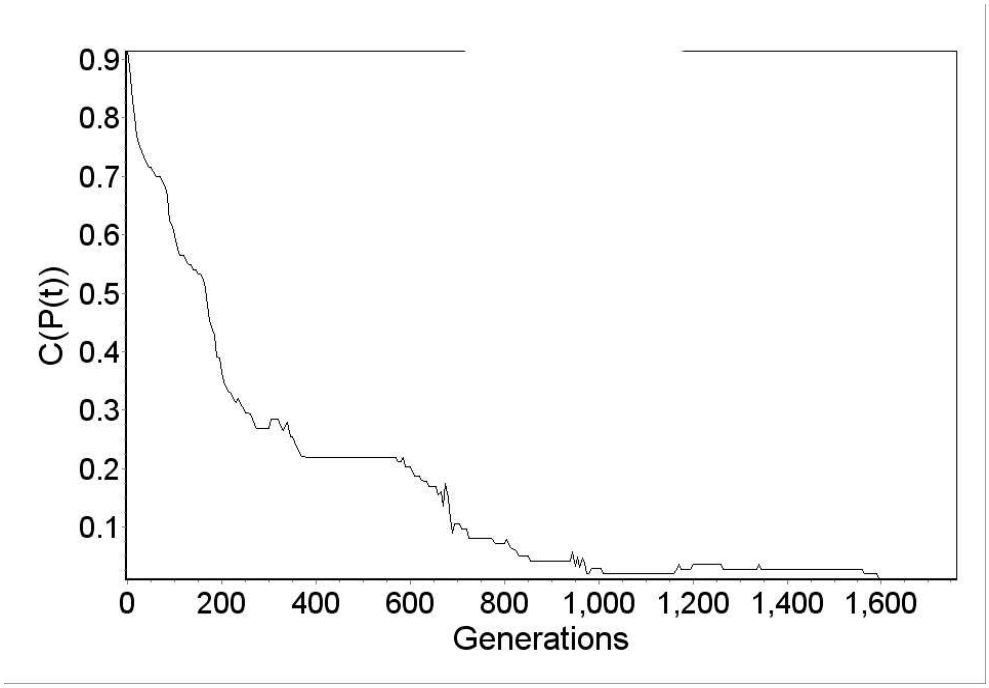

Figura 5.12: Comportamiento del valor de convergencia en una corrida típica.

la Figura 5.16. Se utilizaron 200 vectores de probabilidad, cada uno de los cuales generó 10 individuos, para encontrar 1580.83 estructuras en promedio en el *conjunto de Pareto final*. La razón de aprendizaje fue de 0.85, el desplazamiento se minimizó en los 3 nodos con carga. La máxima varianza permitida fue de 35.0. En promedio, 3.928×10^6 evaluaciones de la función fueron calculadas. El peso propio no fue considerado como carga en el problema de optimización.

5.3.1 Experimento 3. Métrica de convergecia

Se calculó la métrica de convergencia para las 12 corridas. La Figura 5.17 muestra la métrica de convergencia $C(P^t)$ en la última generación para todas las corridas. La media fue de 0.174017 (0 es mejor) con una desviación estándar de 0.1000716. La Figura 5.18 muestra el comportamiento de la convergencia, la gráfica muestra el valor de convergencia $C(P^t)$ calculado cada 25 generaciones.

5.3.2 Experimento 3: Métrica de diversidad

La métrica de diversidad fue calculada para las 12 corridas La Figura 5.19 muestra una gráfica de los valores de diversidad para cada corrida. La media es 0.94562 (1 es mejor) con una desviación estándar de 0.036598, la linea de referencia tiene 100 celdas; fue calculada utilizando mínimos cuadrados. La Figura 5.20 Muestra la métrica de diversidad claculada cada 25 generaciones. Esta gráfica muestra el comportamiento y capacidad de el algoritmo para esparcir los individuos en el frente a través de las generaciones. Note que a diferencia de los algoritmos géneticos y de las estrategias evolutivas, la capacidad de exploración del algoritmo aumenta conforme pasan las generaciones, expandiendo su espacio de búsqueda en lugar de cerrarlo. Finalmente, la Figura 5.21 muestra el frente de referencia (con rombos(y el plano de referencia (línea).

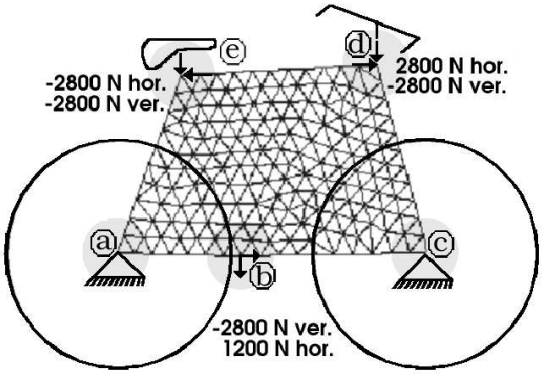

Figura 5.13: Definición del problema.

Módulo de Young	$70 \times 10^9\ Pa$
MódulodePoisson	0.2
Espesor	$0.005m$
Esfuerzo máximo permissible	$100 \times 10^6\ Pa$
Coordenadas de los puntos (m)	
a	$(0, 0)$
b	$(0.45966, -0.00935)$
c	$(1.14826, 0)$
d	$(0.96026, 0.68974)$
e	$(0.23978, 0.64925)$

Figura 5.14: Propiedades del material para el experimento 2.

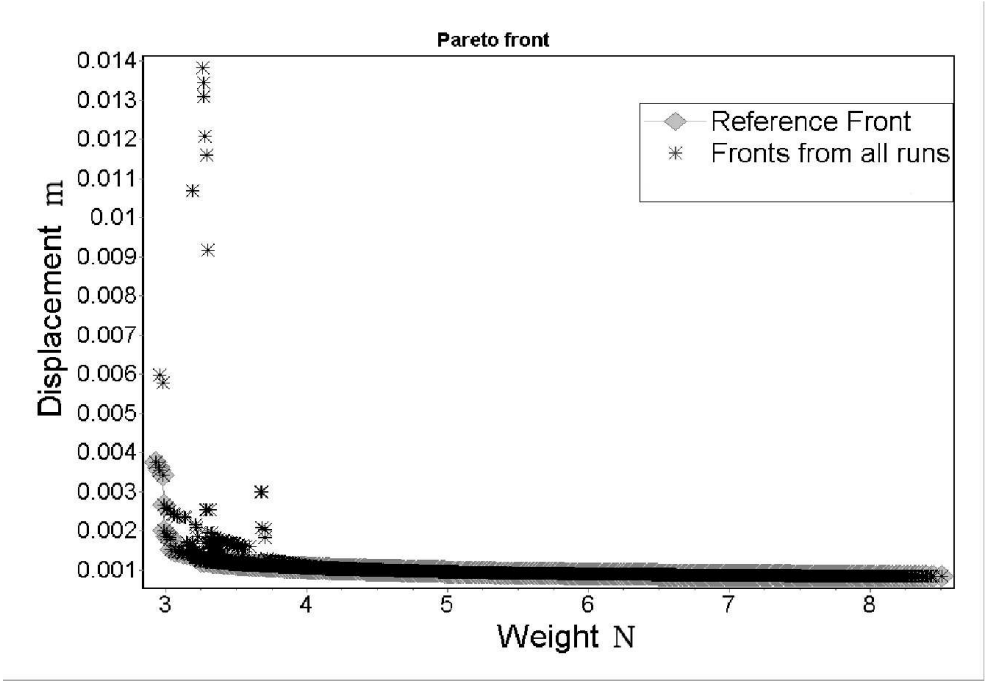

Figura 5.15: Frente de referencia con rombos y frentes de 12 corridas independientes con cruzes,

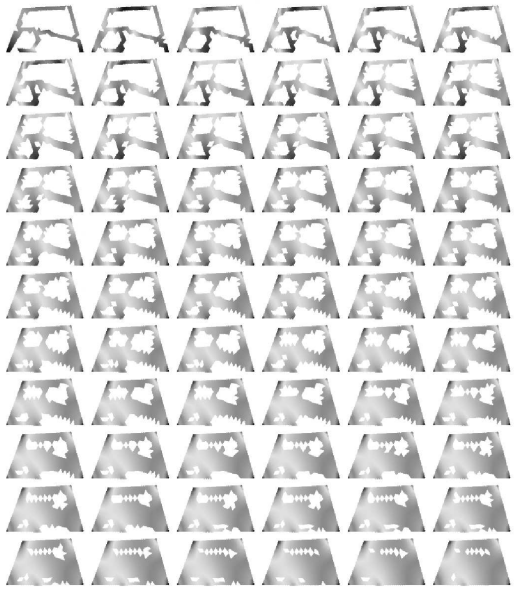

Figura 5.16: Estructuras de una corrida típica.

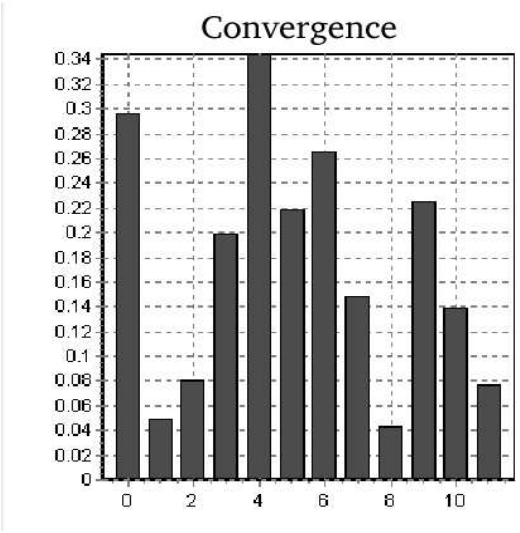

Figura 5.17: Métrica de convergencia de 12 corridas independientes.

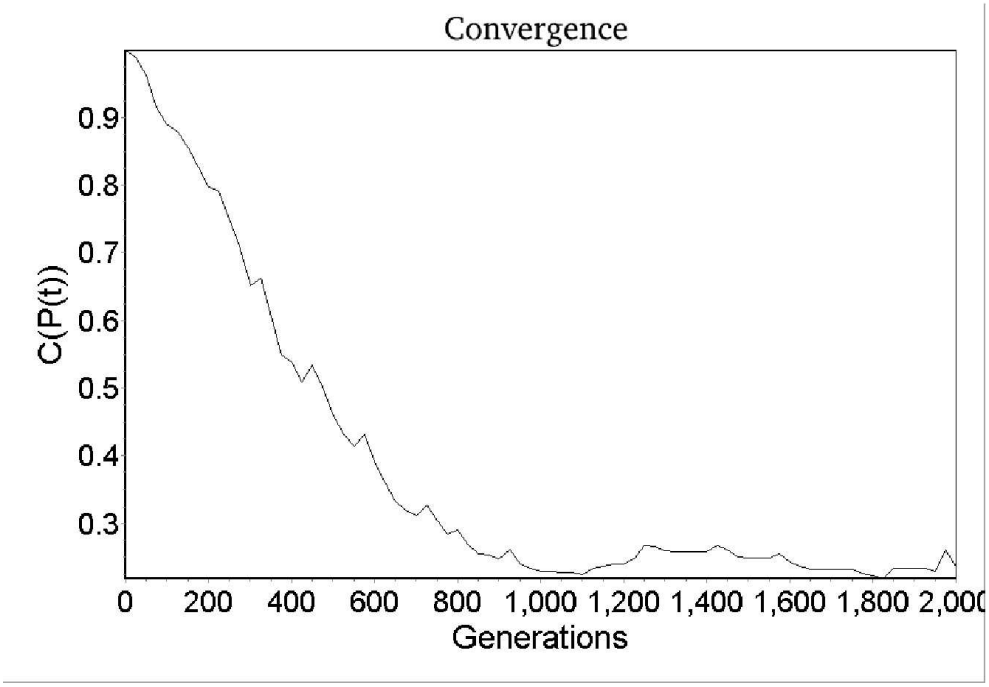

Figura 5.18: Gráfica de convergencia en una corrida típica.

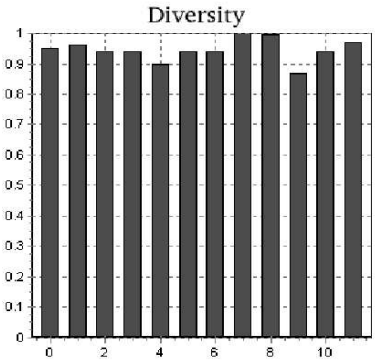

Figura 5.19: Métrica de diversidad para 12 corridas independientes.

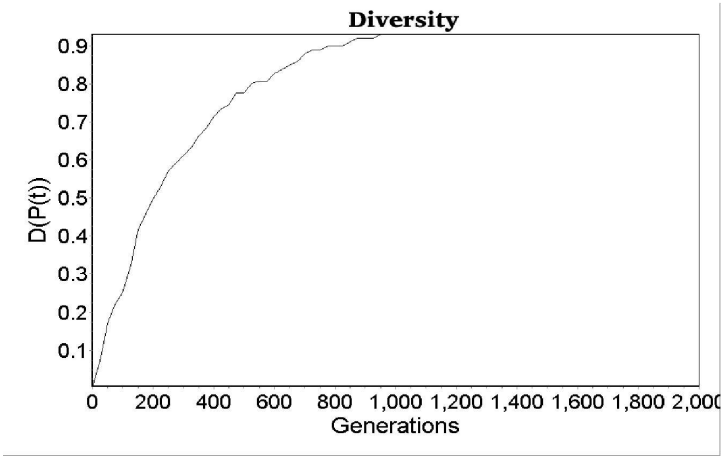

Figura 5.20: Frente de referencia (rombos) y el plano de referencia para la métrica de diversidad.

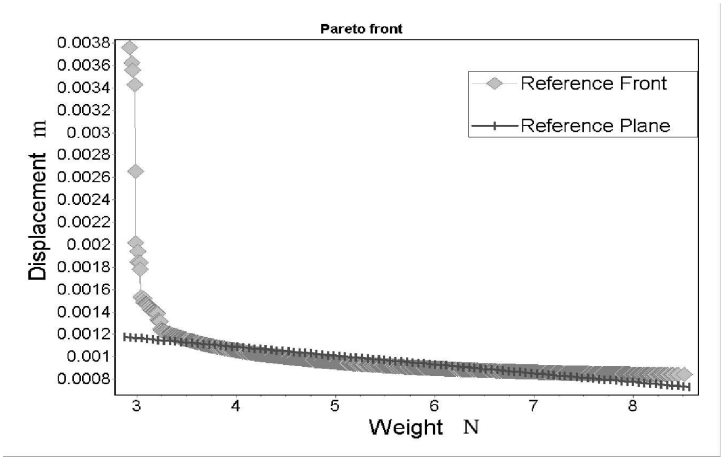

Figura 5.21: Frente de referencia (rombos) y el plano de referencia para la métrica de diversidad.

5.4 Conclusiones

Se presenta un nuevo algoritmo para resolver el problema de optimización de formas, utilizando dominancia de Pareto para el manejo de las restricciones. El algoritmo regresa un conjunto de soluciones que satisfacen las condiciones de servicio y minimizan el peso y el desplazamiento nodal. El comportamiento mostrado por el algoritmo de acuerdo con las métricas utilizadas muestra una buena diversidad, esparciendo las soluciones en el frente; aunque las soluciones en el lado más pesado se ven más juntas que en el lado más ligero, se debe notar que la relación entre peso y desplazamiento no es lineal, por lo tanto, el quitar un elemento cuando la estructura tiene casi el mínimo peso causa un desplazamiento mucho mayor que quitar el mismo elemento en una estructura más pesada. El comportamiento de la convergencia es descendente a través de las generaciones, y todas las corridas llegan a estar muy cerca del frente de referencia.
Se debe notar que el problema de optimización depende de la malla, porque una malla con pocos elementos (cientos) condiciona a la forma dentro de ciertas regiones con cambios brucos en el desplazamiento y peso. Por lo tanto, para trabajos futuros se debe explorarla dependencia del problema de optimización a la malla, y como inicializar las probabilidades en una estratégia multimalla, considerando la dependencia de la malla y la reducción del espacio de búsqueda. Los resultados que se presentan muestran soluciones buenas a pesar de la malla no estructurada (otros autores han utilizado en una gran mayoría mallas estructuradas). Por último el trabajo futuro debe considerar también la reducción del costo computacional; explorando la posibilidad de trabajar con un tamaño de frente fijo.

Bibliografía

[1] S. Baluja. Population based incremental learning: A method for integrating genetic search based function optimization and competitive learning. *School of Computer Science Carnegie Mellon University, Pittsburgh, Pennsylvania 1523*, CMU-CS-94-163, 1996.

[2] C. Chapman, K. Saitou, and M. Jakiela. Genetic algorithms as an approach to configuration and topology design. *Journal of Mechanical Design*, 116:1005–11, 1994.

[3] K. Deb and T. Goell. Multiobjetive evolutionary algorithms for engineering shape optimization. *KanGal report*, (200003), 2000.

[4] K. Deb and S. Jain. Running perfomance metrics for evolutionary multi-objective optimization. *KanGal report*, (2002004):1–18, 2000.

[5] A. Farhang-Mehr and S. Azarm. Diversity assesment of pareto-optimal solution sets: An entropy approach. In *Proceedings of the 1990 World Congress on Computational Intelligence*, pages 723–728, 2002.

[6] C. Kane and M. Schoenauer. Topological optimum design using genetic algorithms. *Control and Cybernetics*, 25(5), 1996.

[7] H. Li, Q. Zhang, E. Tsang, and J. Ford. Hybrid estimation of distribution algorithm for multiobjective knapsack problem. *Procedings of the 4th European COnference on Evolutionary Computation in combinatorial Optimization*, 2004.

[8] L. Malvern. *Introduction to the Mechanics of a Continuous Medium*. Prentice Hall Inc., Englewood Cliffs New Jersey, 1969.

[9] J. Marroquín, F. V. M. Rivera, and M. Nakamura. Gauss-markov measure field models for low-level vision. *IEEE Trans. On PAMI*, 23(4):337–348, 2001.

[10] H. Muhlenbein and G. PaaB. From recombination of genes to the estimation of distributions i. binary parameters. *Parallel problem Solving form Nature*, PPSN(IV):178–187, 1996.

[11] M. Pelikan, D. Goldberg, and C. Paz. Linkage problem, distribution estimation and bayesian networks. *IlliGal Report*, 1(98013), 1998.

[12] M. Pelikan, D. Goldberg, and C. Paz. Boa: The bayesian optimization algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 1999.

[13] R. A. Richards. *Zeroth-order shape optimization utilizing a learning classifier system*. Copyright by Robert A. Richards. All Rights Reserved., 1995.

[14] E. J. Sandgren, E. and J. Welton. Topological design of structural components using genetic optimization methods. In *Proceedings of the 1990 Winter Annual Meeting of the American Society of Mechanical Engineers*, pages 31–43. American Society of Mechanical Engineers, 1991.

[15] O. Zienkiewicz and R. Taylor. *El Método de los Elementos Finitos*. Mc. Graw Hill-CIMNE., 1995.