COMPUTACIÓN EN PARALELO APLICADA A LA OPTIMIZACIÓN DEL DISEÑO ESTRUCTURAL: ESTADO DEL ARTE

YERMAN DAVID CUJIA MEZA

CÓDIGO: 2060720

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERIAS FISICO-MECANICAS

ESCUELA DE INGENIERIA CIVIL

BUCARAMANGA

2010

COMPUTACIÓN EN PARALELO APLICADA A LA OPTIMIZACIÓN DEL DISEÑO ESTRUCTURAL: ESTADO DEL ARTE

YERMAN DAVID CUJIA MEZA

Trabajo de Grado Modalidad Investigación

Para Optar al Título de:

Ingeniero Civil

Director:

OSCAR BEGAMBRE

Ingeniero Civil, (Msc, PhD)

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERIAS FISICO-MECANICAS

ESCUELA DE INGENIERIA CIVIL

BUCARAMANGA

2010

AGRADECIMIENTOS

Agradezco a mi maestro y director de proyecto, al ing. Msc. PhD Oscar Begambre por todo su apoyo, comprensión, colaboración y paciencia en todo este proceso, el cual, sin su esfuerzo, no hubiese sido realidad.

Al grupo de Investigación INME, el cual me facilito el acceso a información y conocimientos valiosos con respecto al tema a tratar.

A la Universidad Industrial de Santander, y a su biblioteca que me permitió el acceso a bases de datos y proveyó de artículos, que fueron necesarios para la realización de este proyecto.

A mis Padres, mis hermanas y toda mi familia por su apoyo durante toda mi vida, y por su confianza puesta en mí, la cual me motivo a seguir y mejorar cada día.

A mis compañeros, los cuales me recibieron en una cálida estadía en esta ciudad e hicieron de esta experiencia una memorable y digna de recordar.

DEDICATORIA

A Dios principalmente porque su amor y misericordia me permitió la llegada a esta etapa de mi vida.

A mi padre Yerman Augusto Cujia Guerra (Q.E.P.D), porque sus sueños me permitieron soñar, sus enseñanzas y su esfuerzo crearon el carácter de lo que soy.

A mi abuela María Dolores de Castilla (Q.E.P.D), porque sé que cada una de las oraciones que hizo por mi fueron escuchadas, y todos sus sabios consejos quedaron grabados en mi corazón.

Yerman David Cujia Meza

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ECUACIONES

RESUMEN

TITULO:

COMPUTACIÓN EN PARALELO APLICADA A LA OPTIMIZACIÓN DEL DISEÑO ESTRUCTURAL: ESTADO DEL ARTE *

AUTOR:

YERMAN DAVID CUJIA MESA **

PALABRAS CLAVES:

Computación en paralelo, optimización estructural, métodos clásicos, métodos heurísticos, arquitecturas paralelas.

DESCRIPCION:

La computación en paralelo nos brinda hoy en día las mejoras exigidas en los procesos de cómputo por la optimización estructural. Por esta razón este trabajo se enfoca en el reporte de procedimientos realizados y/o aplicaciones de técnicas de computación en paralelo a algún procedimiento de optimización estructural.
Dichos problemas de optimización son aplicados a problemas, que son agrupados en dos grandes grupos, discretos y continuos; pero además también se agrupan dependiendo del tipo de optimización a realizar, las más representativas en el diseño estructural son la optimización de forma, de dimensionamiento directo y de topología, siendo esta ultima la más reciente en investigación.

La paralelización de estos métodos se realiza a preferencia del autor aplicando cualquier arquitectura o modelos de computación en paralelo, entre los cuales se encuentra los Pthreads, OpenMP, CUDA, MPI, UPC, Fortress entre otros; y dependiendo del problema puede ser cambiadas algunas de sus características.
Sin embargo el factor crucial para la paralelización depende de la naturaleza del problema como sus componentes, por ejemplo, la presencia de restricciones y naturaleza matemática de estas; la naturaleza matemática de las función objetivo.

Es de esperar que en los próximos años la investigación de este campo aumente de manera considerable, ya sea aplicado a un problema estructural específico o al mejoramiento de una componente de algún método de optimización ya explicado.

---

*Proyecto de grado modalidad investigación.
** Facultad Físico-Mecánicas. Escuela Ingeniería Civil. Director Oscar Begambre

ABSTRACT

TITLE:

PARALLEL COMPUTING APPLIED TO STRUCTURAL DESIGN OPTIMIZATION: STATE OF THE ART *

AUTHOR:

YERMAN DAVID CUJIA MESA **

KEYWORDS:

Parallel computing,structural optimization, gradient-based methods, heuristic methods, parallel architectures.

DESCRIPTION:

Nowadays parallel computing provides the required improvements in computation's processes for structural optimization. For this reason the aim of this paper is on reporting procedures performed and/or applications of parallel computing techniques to a procedure of structural optimization.

These problems are applied to optimization problems, which are grouped into two major groups, discrete and continuous, but alsothey are grouped depending on the type of optimization that is going to be performed, the most representative in the structural design arethe shape optimization, sizing and topology, the latter is the most recent in research fields.

The parallelization of these methods is made with the author's preference using any architecture or parallel computing models, among which are the Pthreads, OpenMP, CUDA, MPI, UPC, Fortress and others. Also, depending on the problem can be changed some of its characteristics.
However, the crucial factor for parallelization depends on the nature of the problem and its components, e. g. If the problem has constraints or not, the mathematical nature of these constraints also its objective function's mathematical nature.

Hopefully in the coming years, researches in this area increased substantially, whether applied to a specific structural problem or improving a component of any optimization method already explained.

---

*Degree Project – Research modality.

** Faculty of Physical-Mechanical Sciences – School of Civil Engineering – Director Oscar Begambre

INTRODUCCION

El realizar cualquier clase de proceso trae consigo un gasto de recursos. En todas las áreas de la sociedad de hoy, estos recursos causan cierto grado de preocupación y el diseño estructural no es la excepción. Este proceso de mejoramiento u optimización exige cada vez más esfuerzos de cálculos los cuales son realizados por equipos de cómputo, y a medida que estos mejoramientos se acercan más a realidad, suma más esfuerzo de cálculo.

La optimización como todos los procesos hoy en día está en busca de herramientas más eficientes que le permita ampliar su aplicación y aumentar su exactitud en respuestas. La computación por su parte está tratando de satisfacer todas estas necesidades de muchas maneras como por ejemplo mejorando la calidad de los materiales que conforman sus partes físicas; pero también se están realizando esfuerzos en encontrar un mayor aprovechamiento de las herramientas presentes utilizando la programación en paralelo.

Esta programación esta poco a poco satisfaciendo las necesidades de computo de los problemas de optimización estructural de una manera sencilla comparada con el mejoramiento que le podría brindar en estos momentos el mejoramiento del hardware. Es así como la atención se ha basado en realizar los cálculos de optimización estructural aplicando conceptos, técnicas de la computación en paralelo en toda su extensión, aunque dicha aplicación requiere de ciertas modificaciones o adaptaciones (no en todos los casos), de los procedimientos ya existentes en la resolución de un problema de optimización estructural.

La computación está avanzando a pasos gigantes en el área investigativa, mejorando los modelos existentes como el CUDA el cual se está implementando para mejoras futuras.

En este trabajo se podrá observar casos de la unión de la optimización estructural a la computación en paralelo, y tendencias en esta forma de resolución de

problemas de optimización estructural. Para lograr este objetivo, en el Capítulo I se presentaran los componentes básicos de problemas de optimización así como los métodos utilizados. En el capítulo II se realiza un resumen sobre la computación en paralelo, especificando conceptos, y el establecimiento de modelos y técnicas para lograr la paralelización de un problema. En el capítulo III se presentan casos de optimización estructural, que involucran en su metodología de realización computación en paralelo con la respectiva ubicación del autor, recopilándose al final en una tabla. Y finalmente se presentan las conclusiones y estimaciones personales de la investigación observada con propuestas para trabajos futuros.

OBJETIVOS

OBJETIVO GENERAL

Realizar una rigurosa revisión de la literatura internacional y nacional relevante relacionado con la computación en paralelo aplicada a la optimización del diseño estructural.

OBJETIVOS ESPECÍFICOS

- Clasificar y agrupar los métodos de optimización estructural y de programación en paralelo, en la revisión de acuerdo con su fundamentación matemática.

- A partir del levantamiento bibliográfico, clasificar según su aplicación los métodos de optimización y programación utilizados.

1. OPTIMIZACIÓN

En cual cualquier ámbito de la vida, ya sea empresarial, industrial etc., es necesaria la toma de cualquier tipo de decisión, normalmente para la resolución de este problema se inicia contando, o suponiendo para un mejor resultado, con una serie de recursos escasos o unos requisitos mínimos los cuales se deben cumplir y condicionan la elección de la mejor solución para nuestro problema de decisión, esto es lo que se conoce como la optimización [1].

La optimización ha estado presente desde el inicio de la humanidad, en las decisiones que hubo que tomar como por ejemplo la elección de la forma de la rueda, hasta la elección de utilizar cierto material para la fabricación de algo, aunque se podría decir que al inicio solo bastaba con una elección correcta fue necesario entrar a evaluar de esa decisión correcta, es ahí donde la optimización llego a ser parte de lo que somos y aun de todos nuestros avances.

De manera general la optimización consiste en la selección de una mejor alternativa, en cierto sentido, que las demás alternativas posibles. Dicha solución y dichas alternativas están compuestas por ciertos partes que determinan su comportamiento y que serán brevemente descritas a continuación:

1.1 COMPONENTES DE LOS PROBLEMAS DE OPTIMIZACIÓN[2]

1.1.1 FUNCIÓN OBJETIVO

La función objetivo (función de coste o función de mérito), es la función que mide cuantitativamente el funcionamiento del sistema en un proceso de optimización esto quiere decir, que se busca una maximización o minimización de esta, y es la base para seleccionar dentro de una gama de diseños aceptables. La función objetivo es una función vectorial de las variables de diseño. Representa la propiedad más importante de un diseño, como por ejemplo el costo o el peso, pero también es posible representar la función objetivo como una función del conjunto de propiedades que se quieren optimizar. Debido a la versatilidad de la función objetivo para

adaptarse al problema propuesto, esta función puede ser continua, discreta o mezclada en aquellos casos en donde entre ciertos intervalos la función se define como discreta y en otros intervalos se define como continua. Además de esto, la convexidad de la función objetivo determina la existencia de un único punto óptimo (Figura 1) o caso contrario, la existencia de múltiples puntos óptimos (Figura 2).

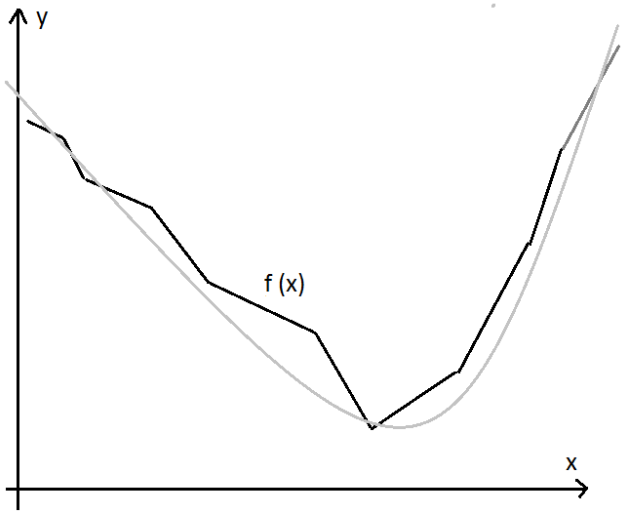


Figura 1: Función Convexa

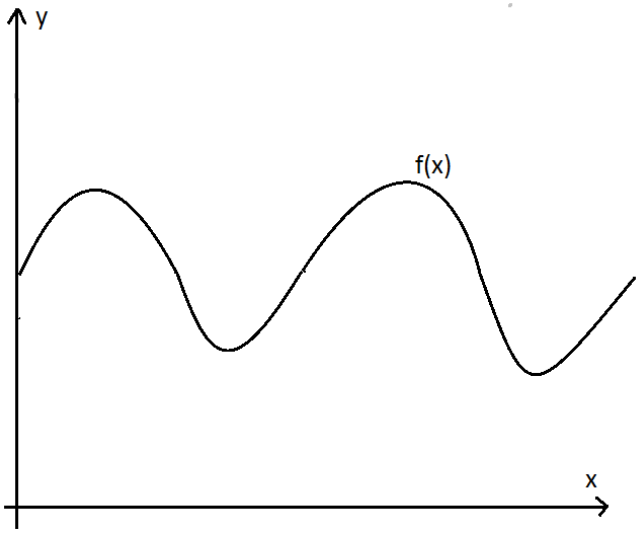


Figura 2: Función No Convexa

Como ejemplo de esta función se pueden mencionar, entre otros, la maximización de los beneficios netos de ventas de ciertos productos, la minimización del peso de cada uno de los elementos que están configurando una cercha, la disminución del volumen de una viga de concreto.

1.1.2 RESTRICCIONES

Restricción según la real academia de la lengua es la acción y efecto de ceñir, circunscribir, reducir a menores límites en cualquier contexto; entonces se podría decir que es una limitación, representada en un conjunto de relaciones, existente que debe satisfacerse para que el diseño sea valedero, Dicha limitación puede ser impuesta directamente por una o varias variables (restricción explícita) o puede ser una representación de una limitación sobre las cantidades que aunque no se pueda determinar una relación directa de estas con las variables de diseño son afectadas por estas (restricción implícita). Dentro de las restricciones se pueden encontrar las restricciones de igualdad, restricción de desigualdad, restricciones laterales y restricciones en comportamiento.

Las restricciones en comportamiento en diseño estructural son generalmente limitaciones en los esfuerzos o en los desplazamientos, pero también pueden tomar la forma de restricciones en frecuencias de vibración o probabilidad al colapso. Otros ejemplos de restricciones son, la capacidad que tiene una fábrica para producir cierto producto.

1.1.3 VARIABLES.

Representan todas las decisiones que se pueden tomar para evaluar la función objetivo, funcionalmente se pueden clasificar en variables independientes (principales o de control), y variables dependientes

(auxiliares o de estado).En un problema de diseño estructural óptimo, estas variables pueden ser:

- Tamaño de los elementos
- Características del material como limitaciones de propiedades físicas, y las limitaciones de propiedades mecánicas.
- O cualquier otro aspecto que se pueda cuantificar en el diseño.

Dentro de las variables de diseño existe un orden de complejidad de estas. La variable de diseño más simple seria el tamaño de cada elemento, y su función objetivo podría ser la representación el área de la sección de una barra, el momento de inercia de un elemento en flexión, o el grosor de una placa. De ahí que la mayor parte de los trabajos en el diseño óptimo se concentren en la determinación del tamaño de los elementos, ya que como lo dijimos anteriormente es la variable más simple y también debida a que en la práctica los elementos que nos brinda la industria común tienen geometría y materiales fijos.

Entonces se podría decir que la resolución de un problema de optimización se trata de encontrar los valores de las variables para hacer óptima la función objetivo cumpliendo con el conjunto de restricciones. La representación gráfica de esta definición se observa en la Figura 3[1]

1.2 CLASES DE OPTIMIZACIÓN SEGÚN VARIACIONES EN SUS COMPONENTES.

La definición esquematizada anteriormente no siempre se mantiene; los problemas en la realidad pueden llegar a poseer, como sabemos, muchos estilos de complejidad y variaciones. Así mismo, pasa con la optimización, existen problemas en los que podría existir una o varias funciones objetivos (Una o varias propiedades a optimizar), otros problemas en donde no existen restricciones.

[1] Imagen modificada [3]

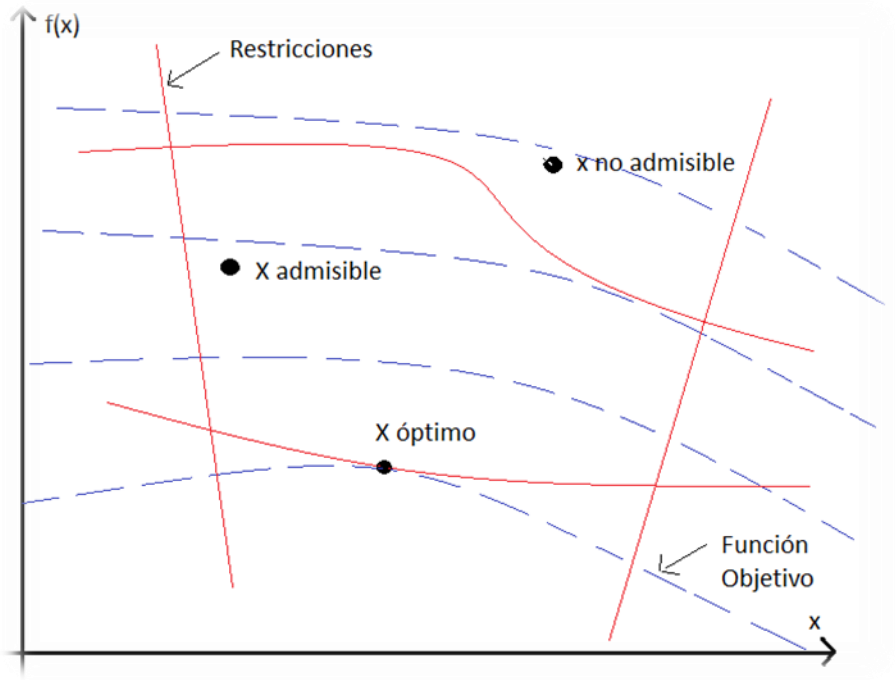


Figura 3 : Espacio de Diseño de un Problema de Optimización

1.2.1 OPTIMIZACIÓN CON ÚNICO OBJETIVO

Este tipo de optimización es aquel donde existe una sola función objetivo y restricciones, y sus variables no implican ecuaciones sistemas de ecuaciones lineales-no lineales en términos generales se podría decir que es el modelo ideal de un problema de optimización. Su planteamiento matemático se presenta a continuación

$$Encontrar\ el\ valor\ x$$

$$que\ optimice\ F(x)$$

$$Sujeto\ a\ g_j(x) \leq 0, \qquad j = 1, 2, \ldots, m,$$

$$h_l(x) = 0, \quad l = 1, 2, \dots, e,$$

Ecuación 1. Planteamiento matemático de la optimización único objetivo

1.2.2 OPTIMIZACIÓN MULTIOBJETIVO

Es cuando existe más de una función objetivo.

Según [4] un problema general de optimización Multiobjetivo se plantea así:

$$Encontrar\ el\ vector\ \overrightarrow{x^*} = [x^*_1,\ x^*_2, \dots,\ x^*_n]^T$$

$$que\ optimice\ F(x) = [F_1(x), F_2(x), \dots \dots, F_k(x)]^T$$

$$Sujeto\ a\ g_j(x) \leq 0, \quad j = 1, 2, \dots, m,$$

$$h_l(x) = 0, \quad l = 1, 2, \dots, e,$$

Ecuación 2. Planteamiento matemático de la optimización Multiobjetivo.

Donde k es el número de funciones objetivo, m es el número de restricciones de desigualdad, y e es el número de restricciones de igualdad.

Normalmente no hay una sola solución global, sino que es necesario obtener un conjunto de puntos que cumplan la definición de óptimo. El concepto predominante para definir un punto óptimo es el de la optimización de Pareto.

Óptimo de Pareto (Enunciado por Vilfredo Pareto) es aquella situación en la cual se cumple que no es posible beneficiar a más elementos de un sistema sin perjudicar a otros [5]; y la condición para que un punto sea un punto de Pareto es:

$$Optimo\ de\ Pareto: Un\ punto, x* \in X, is$$

$$es\ un\ optimo\ de\ Pareto\ si\ no\ hay\ otro\ punto,$$

$$x \in X, tal\ que\ F(x) \leq F(x*), y\ Fi\ (x) < Fi\ (x*)\ para\ al\ menos\ una\ función.$$

Ecuación 3. Condición de un óptimo de Pareto

Como en todo el tema de la optimización, la utilización de algoritmos es común, aunque alguna veces estos algoritmos proveen soluciones que podrían no ser Óptimos de Pareto pero podrían satisfacer otro criterio, haciéndolos significantes para una aplicación.

Dentro de la optimización Multiobjetivo tenemos la optimización Multiobjetivo evolucionaría (Evolutionary multiobjective Optimization EMMO) el cual hoy en día es uno de la investigaciones más activas en cuanto a la computación evolucionaría [6]; esto métodos también son altamente relevante en los problemas de ingeniería porque estos fueron diseñados para afrontar conflictos Multiobjetivos los cuales usualmente ocurren en el mundo real.

Hay dos grandes metas de la optimización Multiobjetivo, la primera es buscar el mayor número de soluciones de Óptimos-Pareto a determinado problema, y el segundo es que las soluciones al problema deben ser altamente diferenciadas.

Las búsquedas y métodos de optimización clásicos no son eficientes para problemas Multiobjetivos porque la mayoría de ellos no pueden encontrar soluciones múltiples en una prueba e incluso si se ejecutara varias veces no garantizaría diferentes soluciones óptimas. Por otro lado los algoritmos genéticos [7] están bien adaptados para resolver esta clase de problemas porque se basan en la población y esta propiedad les permite encontrar un grupo de soluciones Optimo- Pareto en una sola ejecución; además estos son considerablemente más robustos comparados con los métodos clásicos.

El pionero en el uso de los métodos evolucionarios para la resolución de problemas Multiobjetivos fue Rosenberg [8], desde ese entonces muchos

investigadores han desarrollado muchas versiones de algoritmos para optimización Multiobjetivos, entre los más importantes tenemos:

- Funciones Agregadas, [9] en donde los múltiples objetivos están combinados en uno solo usando la suma, multiplicación o cualquier otra combinación de operaciones aritméticas. Usualmente es adoptado el enfoque de suma ponderada multiplicando por coeficientes de ponderación los cuales representan la importancia relativa de los objetivos. El mayor problema de este método es la dificultad para determinar las importancias apropiadas.

- Algoritmo Genético del Vector Evaluado, (Vector evaluated genetic algorithm VEGA) [10] fue propuesto por Schaffer, este maneja múltiples objetivos con la modificación del mecanismo de selección deun algoritmo genético simple. Solo considera problemas sin restricciones, crea divisiones de la población original (Subpoblaciones) dentro de una generación dada, una subpoblación es creada por medio de la evaluación de una función objetivo al tiempo que es agregada a todas las funciones, el proceso de selección es compuesto de una seria de lazos computacionales y durante cada lazo, la utilidad de cada miembro de la población es evaluada mediante un algoritmo de objetivo único.

  Hoy en día tenemos muchas variaciones del original VEGA, estas han sido propuestas para una amplia gama de problemas como por ejemplo la contención de contaminación en aguas subterráneas y el diseño de armaduras de avión.

- Enfoques vector objetivo, (Target vector approaches), En este métodos las metas han de ser definidas por un fabricante de decisión en cada objetivo. Dentro de este grupo tenemos

programación objetivo, Cumplimiento de objetivo y propuesta min – max (Min-maxapproach), este último ha sido utilizado en la optimización de una cercha plana en donde el peso y el desplazamiento estaban siendo minimizados. En cuando a este último método también ha tenido estudios como el que podemos ver [11].

- Algoritmos genéticos Multiobjetivo (Multi objetive genetic algorithm MOGA) [12], en este método se define un rango de individuos basados en el número de individuos en la población presente por la cual este es dominado.

  Los algoritmos genéticos Multiobjetivos están siendo utilizados en muchas aplicaciones de ingeniería como por ejemplo un controlador de turbina de gas, alas supersónicas, etc. Después de esto se ha ampliado este método al diseño conceptual de edificios cuyo nombre es MGA.

- Non-dominated sorting genetic algorithm (NSGA) este método fue definido por Srinivas y Deb [13] y está basado en la noción de clasificación no denominado (non- dominated sorting) de Goldbergs[14] con un lugar y un método especial. Una versión mejorada de este algoritmo es el llamado NSGA-II el cual es sin parámetros y comparte el nuevo acercamiento aplicado a la problema de optimización de diseño topológico, en ambos l masa y el máximo desplazamiento de un voladizo fueron minimizados.

- Algoritmo Evolutivo Fuerza Pareto (Strength Pareto evolutionary algorithms SPEA), [15] donde se integran ideas de varios métodos de optimización evolutivos Multiobjetivos y adiciona nuevos elementos de los algoritmos evolutivos multiobjetivos.

- Colonia de Hormigas: Extensiones Multiobjetivo

  El método de colonias de hormigas, se amplía al Multiobjetivo, esta técnica usa el ordenamiento lexicográfico ya que las funciones objetivos se tienen que organizar de manera jerarquiza y así se resuelven de manera incremental. Aquí se usa el concepto de no dominancia y una población secundaria para almacenar a los individuos que fueron catalogados como no globales, en el 2001 se propuso un algoritmo basado en la colonia de hormigas que usa varias poblaciones, esta idea de usar varias colonias heterogéneas de hormigas cada una de las cuales considera diferentes pesos para una función agregativa, le mecanismo cooperativo utilizado en este caso es un intercambio de soluciones entre poblaciones o colonias diferentes, también se considera dos poblaciones en donde cada una de ellas posee su propia matriz de feromonas. Esta técnica presenta algunos parecidos con PAES, ya que segmenta el frente de Pareto con el objetivo de crear mejores distribuciones de las soluciones.

- Algoritmos Miméticos: Extensión Multiobjetivo

  Como el método anterior los algoritmos miméticos (Ver 1.3.2.7), para problemas multiobjetivos, en 1998 Hartmann propuso una técnica que consiste del NSGA unido a un algoritmo memético que mejora las características de las soluciones encontradas previamente buscando en la vecindad de ellas, el método fue llamado Nondominated Sorting Memetic Algorithm (NSMA).

1.2.3 OPTIMIZACIÓN SIN RESTRICCIONES

En este tipo de problemas al no presentar ningún tipo de restricciones, el objetivo se concentra en buscar el grupo de valores que representa para nuestra función el máximo o mínimo según sea el caso de interés. Para este tipo de optimización existen algunas técnicas de programación no lineal. Entre estos métodos encontramos los métodos Primal-Dual Interior, y los métodos barrier. [16]

1.2.4 OPTIMIZACIONES CON SISTEMAS DE ECUACIONES LINEALES – NO LINEALES

Para este caso no existe una función objetivo propiamente, solo se concentra en encontrar una solución factible al problema con su conjunto de restricciones, este tipo de planteamientos se puede convertir en un planteamiento de problemas variacionales.

1.3 MÉTODOS DE OPTIMIZACIÓN.

Todos métodos de optimización involucran procesos matemáticos, por ende según la clase de procesos a realizar los métodos de optimización se pueden clasificar en dos grandes grupos, en nuestro caso el factor de clasificación va a ser la aplicación o no de derivadas en el procedimiento de optimización.

1.3.1 MÉTODOS CLÁSICOS[17]

Al referirse a métodos clásicos, se podría percibir la idea de antigüedad entre otras cosas debido a que la palabra clásicos posee muchas formas de interpretación, para el desarrollo de este trabajo un método de optimización clásico es aquel que usa las derivadas (gradientes).

Se puede decir que estos métodos buscan y garantizan un óptimo local, pero con sin un tiempo determinado.

Los métodos con derivadas se basan en tres algoritmos fundamentales: pendiente máxima (steepest descent), el método de Newton (Newton's method) y el método Levenberg-Marquart. Sin embargo no son los únicos métodos y algoritmos, pero todos los demás resultan de mejoras y combinaciones de estos.

1.3.1.1 DESCENSO DE LA PENDIENTE MÁXIMA (STEEPEST DESCENT) [18]

Es un algoritmo de optimización de primer orden, consiste en encontrar un mínimo local de una función, usando el gradiente, los pasos se toman proporcionales al negativo del gradiente de la función en el punto actual de análisis, o una aproximación de este.

Se denota el negativo del gradiente, ya que indica descenso, pero si el objetivo fuera conseguir el máximo, el paso seguiría proporcional al gradiente y no al negativo de este.

Matemáticamente el método es basado en la existencia de una función ***f(x)***, la cual es diferenciable y definida en un vecindario de un punto ***p***, entonces ***f(x)*** decrecerá más rápidamente si se va en la dirección negativa al gradiente de ***f(x)*** en el punto ***p***, ($-\nabla f(x)$). Luego se define el siguiente punto ***p'***

$$p' = p - \gamma \nabla f(x)$$

Ecuación 4. Definición del punto siguiente p'

Donde $\gamma > 0$; el cual es un escalar que determina el tamaño del paso, entonces $f(p) \geq f(p')$.

De manera general los puntos serán determinados por:

$$p_{n+1} = p_n - \gamma_n \nabla f(p_n) \; ; n \geq 0$$

Ecuación 5. Determinación general del punto siguiente.

Por consiguiente tendremos que:

$$f(p_0) \geq f(p_1) \geq f(p_2) \geq f(p_3), \ldots \ldots$$

Ecuación 6. Comportamiento esperado de la función

Esperamos que la secuencia converja en p_n, en un mínimo local. Ejemplo gráfico ideal de este método se puede observar en la figura 4

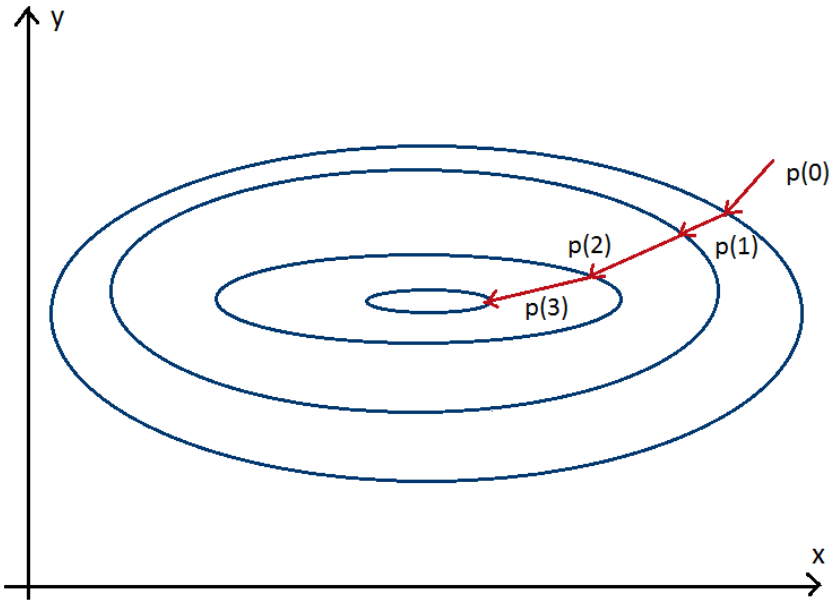


Figura 4: Avance hacia mínimo local

Donde las líneas azules con curvas de nivel, esto quiere decir donde ***f(x)*** es constante, las líneas rojas muestran la dirección donde la pendiente es negativa, al ser esta producto del gradiente debe ser perpendicular a la curva de nivel

1.3.1.2 MÉTODO DE NEWTON[19]

En matemáticas el método de newton es ampliamente utilizado, su función es encontrar las raíces de ecuaciones en una o más dimensiones. Hoy día su aplicación ha crecido en gran manera, en la optimización por ejemplo se puede utilizar para hallar el máximo o mínimo de la función, ya que estos puntos estacionarios son en realidad las raíces de la función derivada.

Este método requiere que la función ***f(x)*** sea dos veces diferenciable, se inicia con un punto ***x,*** el cual es un supuesto o una aproximación al punto estacionario de la raíz de la ecuación ***f '(x) = 0***, desde este punto se construye una aproximación lineal de la función ***f '(x)***y el punto en el cual se desvanece la aproximación lineal se toma como la siguiente aproximación. (Figura 5, Figura sacada de [19])

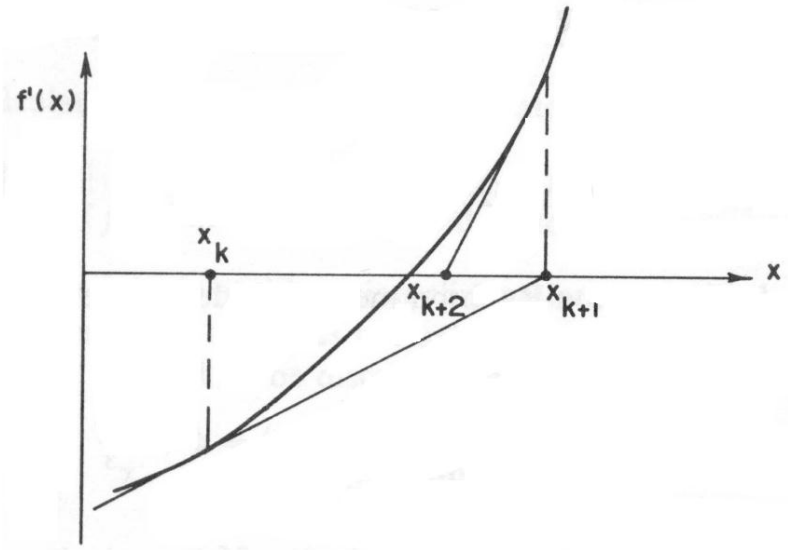


Figura 5: * Pasos método de Newton

Por último, dado el punto x_k el cual es la aproximación actual del punto estacionario, tenemos la aproximación lineal de la función ***f '(x)*** en x_k , el cual según serie de Taylor está definido como:

$$f'(x) = f'(x_k) + f''(x_k)(x - x_k)$$

Ecuación 7. Calculo de la primera derivada según serie de Taylor

Igualando esta expresión a cero, podemos obtener la aproximación del siguiente punto estacionario.

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)};\ n = 0, 1, 2, 3, 4 \ldots$$

Ecuación 8. Determinación del punto siguiente en el método de Newton.

Como en todos los métodos, posee ciertas condiciones que pueden provocar el fracaso de esta ósea una divergencia(Figura 6, Figura sacada de [19]). En este caso se podría haber evitado esta situación si se hubiera iniciado a izquierda del punto x_0.

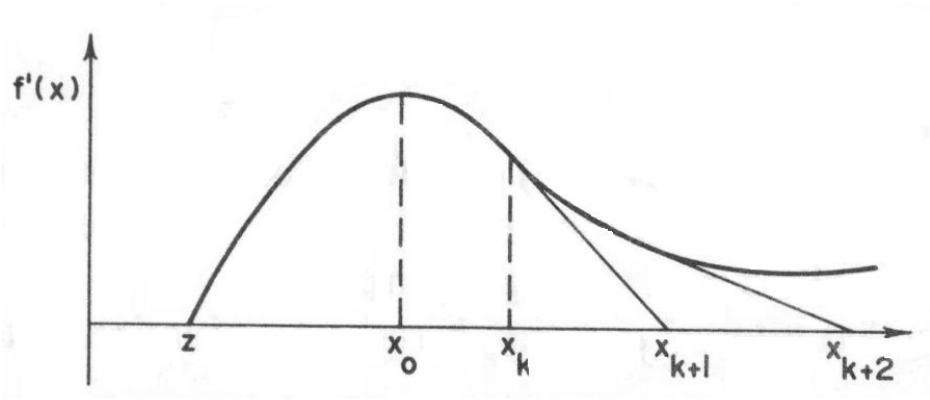


Figura 6: * Situación especial, método Newton

1.3.1.3 MÉTODO LEVENBERG-MARQUART (THE LEVENBERG–MARQUARDT ALGORITHM "LMA")[20]

Este método proporciona una solución numérica al problema de minimizar una función, generalmente no lineal sobre el espacio delimitado por la función.

Este método interpola entre el método de Newton y el método de descenso por gradiente, pero es más robusto que ellos, lo que implica en muchos casos que encuentra una solución aunque comience lejos del mínimo final, pero también tiende a hacer más lento que el método de Newton.

Este algoritmo es un algoritmo iterativo de optimización donde como se dijo anteriormente presenta una ligera modificación, sobre el método

tradicional de Newton. Las ecuaciones normales $N\Delta = J^T J\Delta = J^T \varepsilon$ donde J representa el jacobiano de la función, Δ los incrementos de los parámetros y ε el vector de errores residuales del ajuste. Estas ecuaciones son reemplazadas por las ecuaciones aumentadas normalmente $N'\Delta = J^T \varepsilon$; donde $N'_{ii} = (1 + \lambda)N_{ii}$ y $N'_{ii} = N_{ii}$ para $i \neq j$. El valor de λ es inicialmente puesto a algún valor, normalmente $\lambda=10^{-3}$, si el valor del incremento obtenido resolviendo las ecuaciones aumentadas conduce a una reducción de error, entonces se acepta, y λ es dividido por diez para la siguiente iteración. Por el contrario, si el valor del incremento da un aumento de error (nos aleja del mínimo o máximo), entonces λ es multiplicado por diez y se repiten las iteraciones hasta lograr un decremento del error.

1.3.2 MÉTODOS HEURISTICOS[21]

Según la real academia de la lengua la heurística está definida como:

Heurística

Es en algunas ciencias, manera de buscar la solución de un problema mediante métodos no rigurosos, como por tanteo, reglas empíricas, etc.

Un método heurístico es aquel que no utiliza metodología común, y rigurosa para obtener un resultado. De manera general estos métodos dan un resultado aceptable en tiempo aceptable, además de su aplicabilidad cuando se presenta una función objetivos con múltiples puntos óptimos o cuando estas funciones están compuestas por intervalos continuos y discretos. Para la clasificación presentada todos aquellos métodos que no usan derivadas de la función objetivo (método riguroso), sino que usan la función objetivo como tal. Su manera de buscar soluciones se fundamenta en el uso de conceptos intuitivos basados en sistemas naturales, como por ejemplo la evolución. El solo hecho de no usar derivadas que en ciertos casos pueden ser tediosas o

aun casi imposibles de obtener, le da a estos métodos una flexibilidad, y una gran extensión de aplicabilidad cuando las condiciones del modelo son complejas.

La manera de encontrar la solución de dichos métodos es a través de la iteración. Algunos ejemplos de estos métodos son:

1.3.2.1 ALGORITMOS GENÉTICOS[7]

Esta técnica se basa en la teoría de la evolución de Darwin, que dice que los individuos más aptos de la población lograran sobrevivir a los cambios que realiza el entorno, para lograr esta adaptación en el individuo se crean cambios en su unidad básica (genes), y lo genes más deseados que son los que van permitiendo su adaptabilidad se transmiten a sus descendientes cuando este tiene reproducción sexual.

La idea fue desarrollada por John Holland a finales de los 60's, donde desarrollo una técnica en la cual pudo incorporarla a un programa, lo que le llamo "planes reproductivos", que más adelante nombro como algoritmos genéticos.

Una definición bastante precisa fue enunciada por John Koza[7]:

"Es un algoritmo matemático altamente paralelo que transforma un conjunto de objetos matemáticos individuales con respecto al tiempo usando operaciones modeladas de acuerdo al principio Darwiniano de reproducción y supervivencia del más apto, y tras haberse presentado de forma natural una serie de operaciones genéticas de entre las que destaca la recombinación sexual. Cada uno de estos objetos matemáticos suele ser una cadena de caracteres (letras o números) de longitud fija que se ajusta al modelo de las cadenas de cromosomas, y se les asocia con una cierta función matemática que refleja su aptitud. "

Los algoritmos genéticos son métodos adaptativos utilizados para resolver problemas de búsquedas o de optimización, los algoritmos genéticos usan dicha analogía con el mundo natural, se trabaja con

una población de individuos, donde cada uno de ellos es una solución factible a un problema dado, este individuo obtiene un valor o una puntuación de acuerdo a la bondad de la solución que este brinde, comparado al mundo natural estos nos diría cuan apto está el individuo para competir, entre mayor sea esta adaptabilidad, mayor será su probabilidad de reproducirse cruzando su material genético con otro individuo también seleccionado. Esta acción crea descendientes de los anteriores (lo que indica que poseen características de sus padres); y estos de acuerdo a sus genes así tendrán la posibilidad de reproducirse, provocando una eliminación progresiva de los genes menos aptos en la población. SI el algoritmo ha sido bien diseñado este convergerá en una solución óptima.

Los algoritmos genéticos operan de forma simultánea, en vez de trabajar de forma secuencial como las técnicas tradicionales, de ahí a que algunos autores vean implícito en estos algoritmos trabajo en paralelo [22].

Estos algoritmos son menos afectados (que las técnicas tradicionales) cuando se trata de maximizar una función objetivo y existen varios máximos locales (falsas soluciones), al ser su naturaleza de cierto modo paralela por lo que nombramos anteriormente resulta fácil ejecutarlos en las masivamente modernas arquitecturas paralelas.

El modelo de Islas, nos permitirá observar tres maneras diferentes de explorar el paralelismo de los algoritmos genéticos.

Modelo De Islas.[8]

La idea es simple, consiste en tomar la población total y dividirla en varias subpoblaciones a las cuales se le aplicara un algoritmo genético, cada cierto número de generaciones se efectúa una migración (intercambio de información entre dichas subpoblaciones), lo cual también le da la oportunidad a las islas (subpoblaciones) de comparar

su diversidad genética, y compartir material genético con las otras islas, este intercambio es controlado por una tasa de migración evitando así convergencias prematuras, la manera en que las islas se comunica crea modelos, entre los típicos tenemos:

- Comunicación Estrella, en este modelo se selecciona una isla maestra, el criterio para la elección de esta se basa en la media más alta de la bondad perteneciente a la función objetivo, y el resto de islas se consideran esclavas. Cada una de las esclavas manda sus mejores hijos a la maestra y esta a su vez manda su mejores hijos a cada una de las esclavas, (Figura 7)

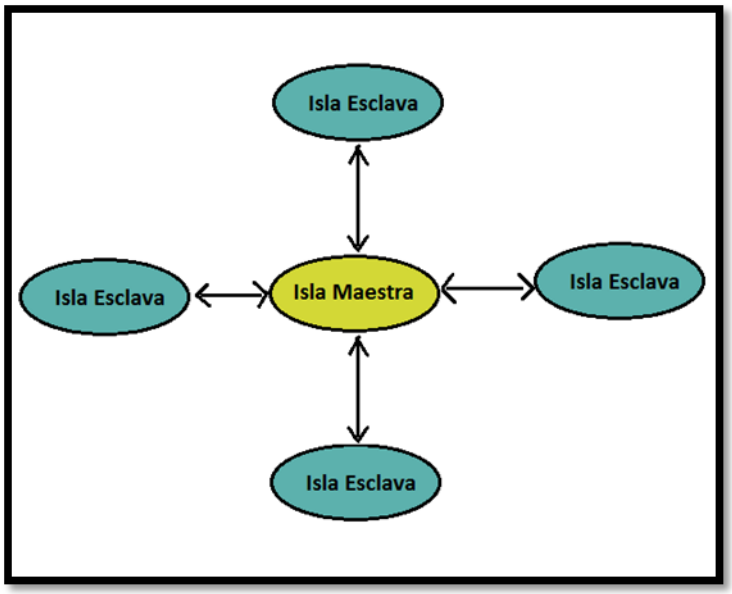


Figura 7: Comunicación estrella

- Comunicación en Red, en este caso no existe jerarquía todas las islas mandan sus mejores individuos a todas las demás islas (Figura8).

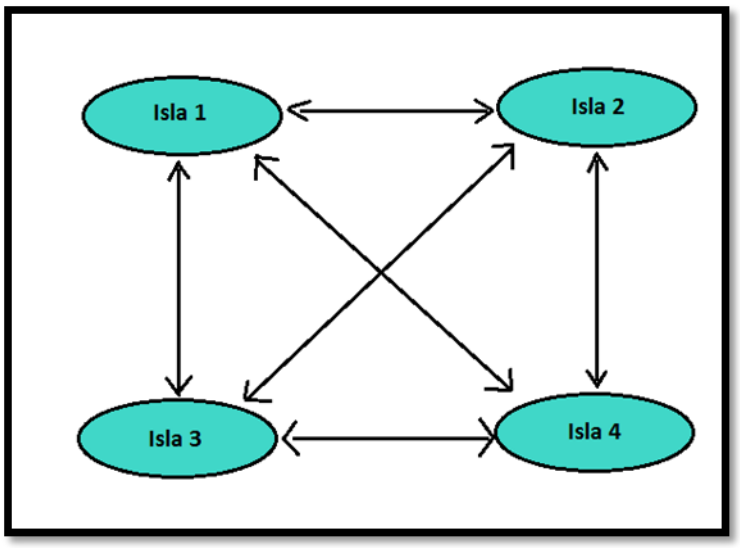


Figura 8: Comunicación en red

- Comunicación anillo, en este caso las islas envían sus mejores individuos a una isla vecina, solo realizándose flujo de individuos en un solo sentido (Figura 9).

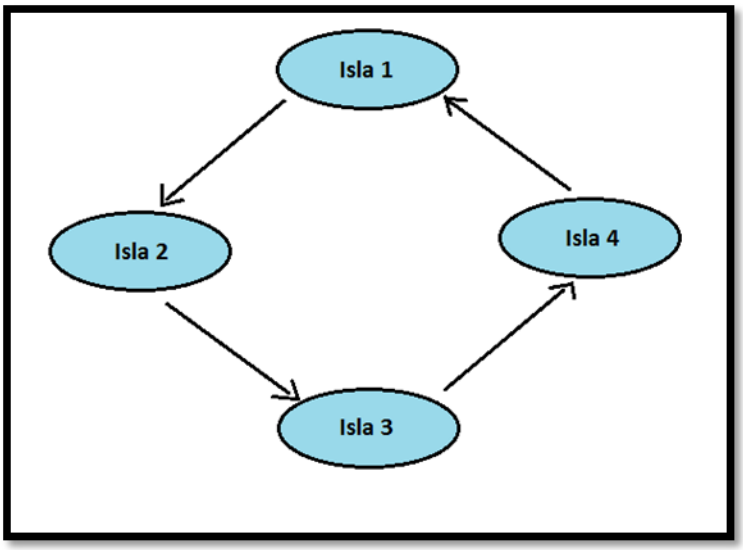


Figura 9 : Comunicación anillo

1.3.2.2 RECOCIDO SIMULADO (SIMULATED ANNEALING SA) [21]

Este método se inspira en el proceso de recocido del acero, donde se calienta y se enfría controladamente el material con el fin de aumentar el tamaño de sus cristales y por ende reducir sus defectos. El calor hace que los átomos salgan de sus posiciones, y ya que la materia siempre tiende al menor estado de energía, el enfriamiento lento aumenta las probabilidadesde que se organicen en un estado de energía menor al anterior, lo importante es controlar la temperatura para asegurar que los átomos se configuren en el menor estado de energía (que caigan al mínimo absoluto y no a un mínimo local).

En este método se considera unos vecinos***e'***de un estado actual ***e*** y de manera probabilística decide si moverse a este nuevo vecino ***e'*** o permanecer en el estado ***e***, las probabilidades son elegidas debido a que en últimas instancias el sistema siempre se moverá al menos estado de energía, normalmente este paso se repite hasta que la aplicación encuentra un estado lo suficientemente bueno o hasta cuando se agotan los recursos de computo.

1.3.2.3 DOWNHILL SIMPLEX SEARCH (METODO NELDER–MEAD Ó METODO AMEBA)[23]

Este método utiliza en concepto de simplex, el cual es un politopo (la generalización a cualquier dimensión de un polígono bidimensional, o un poliedro tridimensional) de *N+1* vértices en *N* dimensiones, el concepto es simple la idea es variar el punto más alto del simplex por uno más bajo, si este proceso resulta exitoso el simplex se agrandara, si no es exitoso este se achicara.

El simplex se adapta al entorno de la función objetivo como una ameba y de esta forma encuentra un mínimo local cercano. El paso simple consiste en reemplazar el peor punto con un punto que se refleja a través del centro de gravedad de los puntos de N restantes. Si este

punto es resulta mejor que el mejor punto actual, entonces podemos tratar de estirar de forma exponencial a lo largo de esta línea. Mientras que si este nuevo punto no es mucho mejor que el valor anterior, entonces se intensificara a través de un valle, por lo que reducir el simple hacia un punto mejor.

De manera general el simplexreemplazara el punto a través de 4operaciones: la reflexión, la expansión, la contracción de una soladimensión, y la contracción múltiples(Figura 10,* Figura sacada de [23])

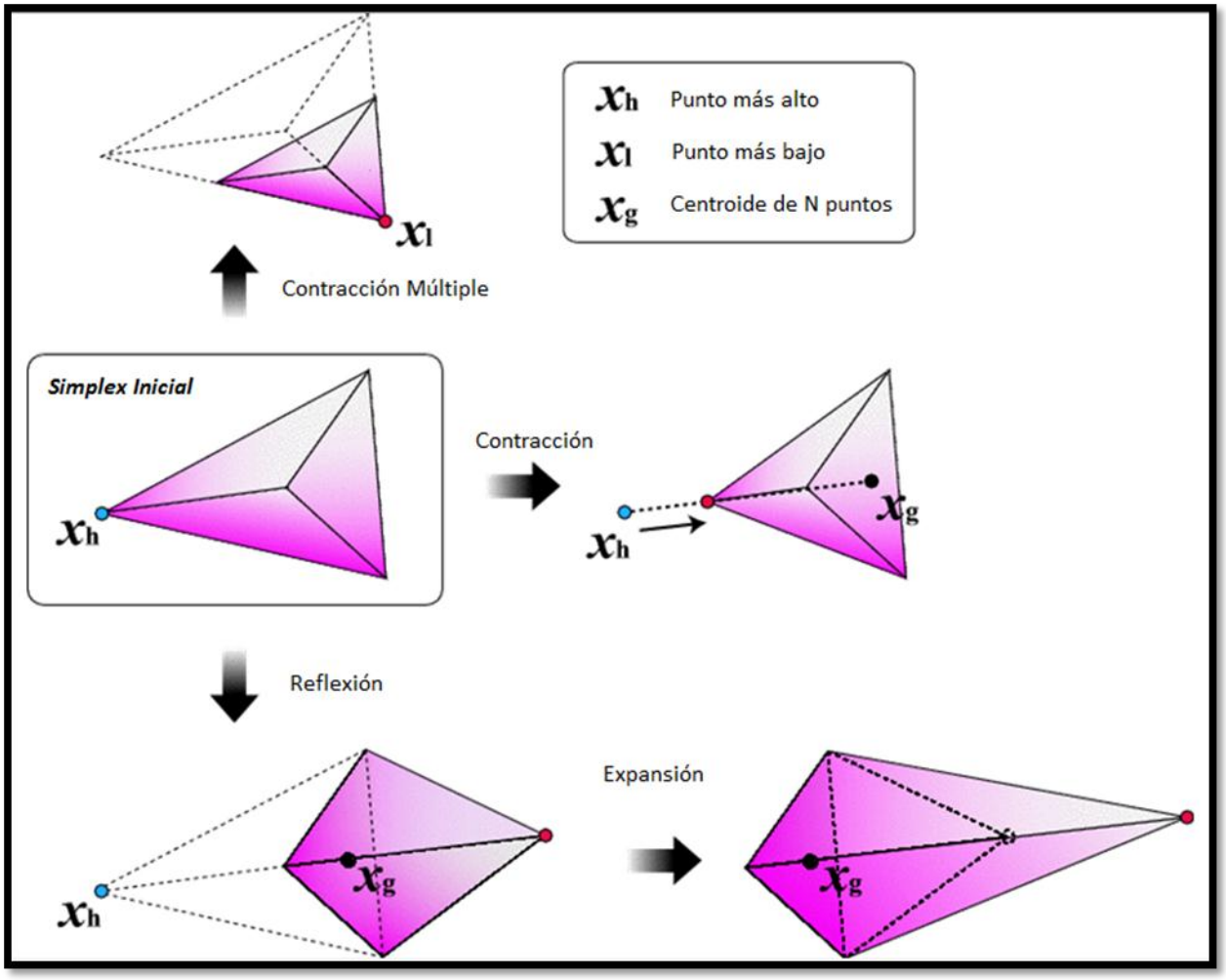


Figura 10 : Operaciones básicas en el método downhill simplex *

1.3.2.4 BUSQUEDA ALEATORIA (RANDOM SEARCH)[21]

O búsqueda aleatoria explora el espacio paramétrico de manera supuestamente aleatoria para encontrar el punto ***x*** que minimiza o máxima la función ***f(x),*** para realizarlo este método elige un punto de partida ***x***, luego a este punto le suma un valor aleatorio ***δx***y se evalúa la función nuevamente la función ***f(x+ δx).***

Luego se compara entre los dos valores de la funciones si ***f(x+ δx) <f(x)***, en el caso de búsqueda de mínimo, se designa como nuevo punto ***x+ δx,*** y si ***f(x+ δx) >f(x),*** en el caso de búsqueda de máximo, también se designara como nuevo punto ***x+ δx.*** Se detiene con el máximo número de iteraciones o nos regresamos a elegir otro ***δx.***

Como podemos observar este método se podría considerar uno de los más sencillos en cuanto a procedimiento de cálculo se refiere, pero igual que los demás métodos este también ha tenido avances y mejoras, y esta consiste en que la búsqueda no se hace en una sola dirección, por el contrario se realiza en dos, esto quiere decir que el punto seleccionado no solo se le adicionara ese valor aleatorio sino que también le será restado, (***x+ δx***y ***x- δx***).

Este método es muy útil con optimizaciones globales en estructuras con variables discretas y/o continuas. Para mayor información consultar [24]

1.3.2.5 OPTIMIZACIÓN POR ENJAMBRE DE PARTICULAS (PARTICLE SWARM OPTIMIZATION “PSO”[25]

Este método fue desarrollado por Dr. Eberhart y Dr. Kennedy en 1995[26], y fue inspirada por el comportamiento de las bandadas de aves, el movimiento de los bancos de peces y los enjambres de insectos en la naturaleza.

Este método comparte muchas similitudes con las técnicas evolutivas, como los algoritmos genéticos (GA), empieza con una población aleatoria y la búsqueda de optimo es actualizando las generaciones.
Sin embargo, a diferencia de GA, PSO no tiene operadores de la evolución, tales como cruce y mutación. En PSO, las posibles soluciones, unas partículas llamadas, volar por el espacio del problema siguiendo las partículas óptimas actuales.
Cada partícula realiza un seguimiento de las coordenadas en el espacio del problema que cree que es la mejor solución que ha logrado hasta ahora ósea según el conocimiento de su entorno (fitness), y el valor de aptitud de este "mejor punto" es almacenado. El optimizador de enjambre de partículas sigue al "mejor valor" de los mejores valores que hasta el momento hay por cualquier partícula.

Figura 11 : Fotografía Enjambre de abejas

Ejemplo de esta metodología lo vemos en las abejas (Figura 11), las cuales a la hora de buscar polen, lo buscan por aquella región del

espacio donde hay más densidad de flores, y por ende será allí donde exista más, una sola abeja (una partícula del enjambre vuela de manera aleatoria sobre el espacio, y lo único que recuerda la abeja es donde ha percibido mayor densidad de flores. Al tiempo el enjambre también sabe de algún modo la región del espacio donde el enjambre ha encontrado la mayor densidad de flores. Cada abeja al ver que el enjambre apunto en otra dirección dudara y variara individualmente su movimiento en dirección intermedia entre las dos direcciones de información (la individual y la del enjambre). Puede ser que la abeja en ese sobrevuelo pueda encontrar una región con más densidad de flores de la que ha visto hasta ese momento, o aún con más densidad que la el enjambre ha visto, si ese es el caso la dirección del enjambre variara hacia la exploración de esta nueva zona. Pasado el tiempo si se sigue encontrando regiones con más densidad el enjambre explorara esa zona.

En los últimos años, la optimización por enjambre de partículas se ha aplicado con éxito en muchas áreas de investigación, lo que demuestra que este método obtiene mejores resultados en una manera más rápida y más barata en comparación con otros métodos.

También se podría decir que este éxito de aplicabilidad se debe que hay algunos parámetros ajustables, pequeñas variaciones podría funcionar buen para una amplia gama de aplicaciones.

1.3.2.6 STOCHASTIC HILLCLIMBING[27]

Pertenece a la familia de los métodos de búsquedas locales (se le considera local ya que toma la decisión solo en base de las consecuencias inmediatas de sus opciones), es relativamente simple de implementar y esto lo hace una popular primera elección, este método fue introducido primeramente por Ackley en 1987, y puede ser utilizado para problemas que tienen muchas soluciones, a lo que se le llama

búsqueda espacial, en esta búsqueda usualmente diferentes soluciones tienen diferentes valores.

La escalada empieza con una solución aleatoria la cual es generalmente pobre, y por medio de la iteración realiza pequeños cambios a la solución creando así un vecindario de solución.

Si dicha solución vecindario es mejor que la actual, entonces la utilizaremos para reemplazar la solución actual hasta cuando esta no pueda ser mejorada lo que implicara la finalización del algoritmo. Idealmente este punto debe ser muy cercano al óptimo, pero no se garantiza su acercamiento con el punto óptimo.

Matemáticamente el método realiza una búsqueda en un espacio discreto S con el fin de encontrar un estado que tenga una bondad (fitness) lo más alto o bajo posible. Esto se logra haciendo pequeñas sucesivas mejoras a un estado σϵ S.

Dentro de este método tenemos, la escalada simple que es un método de búsqueda local que no mantiene registro de sus estados anteriores solo se caracteriza por sus movimientos los cuales están determinados por ser mejores que los previos.

Y la escalada por máxima pendiente, donde no busca solo un estado mejor que el actual, sino uno mejor a todos los posibles esto es lo que es representado como una mayor pendiente.

Para ayudar al generador a decidirse por cual decisión moverse en el espacio se realimenta del procedimiento de prueba, por lo que si no se encuentra una razón razonable es preferible abandonar el proceso de búsqueda.

Existe la posibilidad que nunca se llegue a encontrar la solución, esto pasaría si el método se quedara atrapado en estados que no son el objetivo, desde los cuales no se pueden hallar mejores estados como por ejemplo: un máximo local, donde refleja un estado mejor que los vecino pero no mejor que otros que se encuentran alejados, una buena

forma de evitar esto (pero no garantizado) es devolverse a un estado anterior y realizar la búsqueda en otra dirección; una meseta, es aquel espacio de búsqueda en donde todo el conjunto de estados tienen el mismo valor, para tratar de evitar esto lo recomendable es dar un gran salto, en alguna dirección para encontrar así algún estado mejor; y por último los riscos que se podría definir como un tipo especial de máximo local, en el cual es imposible escapar con pasos simples por lo que para tratar de este estado lo que se podría intentar es moverse en varias direcciones a la vez.

En todos los casos que nombramos anteriormente el algoritmo llega a un punto más allá del cual no se logra ningún avance, por lo que es necesario empezar de nuevo desde otro punto de partida, por lo que el método fue mejorado de tal forma a ascenso de cima con reinicio aleatorio, en el cual se realizan muchas series de búsquedas de ascenso de cima desde estados iníciales elegidos aleatoriamente.

El objetivo es esperar que cada uno de ellos encuentre el fin de su iteración, esto quiere decir cuando cada uno no logre ningún avance significativo, y hace recolectar los mejores resultados que se hayan obtenido hasta cierto momento dado, está en la libertad de utilizar un número fijo de iteraciones o hasta cuando el criterio diga que es el mejor resultados ya almacenados no han sido mejorados en cierta cantidad de iteraciones.

Este método es ventajoso ya que no explora todo el espacio de estados sino que como máximo da una sola solución, y precisamente esto es lo que se desea cuando se aplica un modelo heurístico obtener resultados razonables (Además de cumplir con diseño, ser un poco mejor) en tiempos razonables, y esto al tiempo se define como eficiencia.

1.3.2.7 ALGORITMOS MEMÉTICOS[28]

A finales de los años ochenta ya comenzaba a afianzarse la computación evolutiva, y fue el escenario para los orígenes de los algoritmos memético (MA) cuya idea básica es combinar conceptos y estrategias de diferentes heurísticas o metaheurísticas para intentar unir las ventajas de las mismas.

El término “meme” fue definido en 1976 por Dawkins como “la unidad básica de una transición cultural.

Los algoritmos memético en todo momento cuentan con una población de soluciones diferentes al problema analizado, cada una de estas soluciones son llamadas agentes, los cuales tienen una interrelación entre ellos en los escenarios de competición y de cooperación.

Cuando se considera la población de soluciones en todo su conjunto, y estos cambios en ellas se llaman generaciones, cada una de estas generaciones se logra con actualización de su población de soluciones, dicha actualización se hace recombinando las características de algunos de agentes seleccionados.

Es justamente esta selección lo que provoca la competencia entre cada una de las soluciones, más concretamente esta selección se realiza a base de elegir las mejores soluciones presentes en la población actual, para poder medir este grado de bondad, se emplea una función guía que cuantifica cuán bueno es cada una de las soluciones de resolución del problema abordado.

Después de la selección continua la actualización, encargándose de la más importante tarea: colocar límites en el tamaño de la población a través de la eliminación de agentes y la entrada de agentes nuevos enfocándose en la tarea de búsqueda.

La creación de nuevos agentes es responsabilidad de la fase de reproducción, la cual tiene lugar mediante la aplicación de ciertos números de operadores reproductivos, este proceso tiene parecido con los algoritmos genéticos ya que los procesos típicos para la respectiva

reproducción son la recombinación y mutación; primero la recombinación es el responsable de llevar a cabo la cooperación entre los agentes, esta cooperación se realiza mediante la construcción de nuevos agentes solamente empleando información de las soluciones presentes y quizás alguna información externa.

La mutación es realizada por llamados meta operadores, los cuales iteran la aplicación iterativa de un operador de mutaciónarbitrario sobre un solo agente, este hecho de solo mejorar la bondad de un agente hacen que este proceso sea llamado de optimización local. Aunque vale la pena resaltar que esto hace parte de un elemento diferenciador de los algoritmos memético, pero ni es el único, ni es imprescindible encontrarlo de esta manera; se debe procurar evitar la falsa igualdad que los algoritmos memético son iguales a los algoritmos evolucionarios más una búsqueda local.

1.4 OPTIMIZACIÓN ESTRUCTURAL.

El diseño estructural no es la excepción a la problemática de tomas de decisiones, desde el mismo momento en el cual se decide empezar a diseñar nos enfrentamos a preguntas que por lo general requieren cierto estudio previo para poder resolverlas, pero no todas estas respuestas tienen un valor único, sino que pueden ser una gama de valores los cuales de producen a lo que llamamos de cierta manera “sobre-diseño”, es aquí donde la optimización estructural juega su rol.

La optimización estructural ha tenido gran aceptación, se podría decir que casi toda su historia el desarrollo de esta rama ha sido de carácter analítico, cuyo objetivo se puede generalizar en “minimizar el costo, o minimizar el peso”, de tal manera que se satisfaga con el diseño y con los requerimientos de este. Pero hoy en día se puede observar que todos los métodos modernos para el diseño optimo

son tan abiertamente generales que se puede realizar el análisis con una cantidad bastante extensa de hipótesis de “carga”, o “situaciones” de colapso; esto se ha podido realizar gracias a la oportunidad que le ha dado la computación a esta rama de la optimización, al reducir el tiempo en todos sus procesos analíticos, le brindó la oportunidad de extender el estudio de esta.

1.4.1 TIPOS DE OPTIMIZACIÓN ESTRUCTURAL SEGÚN EL TIPO DE PROBLEMA.

En el diseño estructural, la optimización varía según el problema de diseño que se está tratando; algunos de estos problemas se han venido trabajando desde cierto tiempo atrás, mientras que otros de ellos son recientes o totalmente nuevos en sus estudios y aplicaciones, podríamos así clasificar los tipos de optimización estructural endos grandes categorías: Problemas Clásicos de Optimización estructural, (teniendo en cuenta que esta vez clásico hace referencia a algo que ha venido trabajando durante un tiempo considerable de tiempo alrededor de 30 años) entre estos tenemos optimización de la forma (Busca el contorno o forma óptima de un sistema estructural pero dejando fija su topología) y Optimización de dimensionamiento directo (Busca por la óptima sección transversal, dimensiones o elementos de un sistema estructural, pero teniendo fijo su forma y topología). Y los Problemas especiales de Optimización estructural entre los cuales tenemos la Optimización de la topología (búsqueda de una distribución de material óptimo para el sistema), Optimización de la detección de daños en estructuras, y la Optimización en ajustes de modelos (Modelupdating)[29].

Un problema ubicado en cualquiera de las categorías mencionadas anteriormente también puede ser clasificado en problemas de optimización continua o discreta.Los problemas discretos de optimización son fácilmente representados en una cercha, ya que sus componentes son unidades que

se pueden diferenciar fácilmente teniendo como objetivo la elección de alguna propiedad en particular de las unidades o en su defecto propiedad de sus puntos de unión (Figura 13, 14a y 16)[2].

Un problema continuo es aquel donde no se pueden distinguir unidades de conformación, siendo la estructura como su nombre lo indica totalmente continua, un claro ejemplo seria el diseño de un elemento estructural específico (Figura 12, 14b y 15)[3].

1.4.1.1 OPTIMIZACIÓN DE LA FORMA (SHAPE OPTIMIZATION "SO")

En los problemas de optimización de forma se busca alcanzar cierto rendimiento satisfaciendo las restricciones, como en la optimización general, algunos se solucionan en base del gradiente, derivadas mientras que otro grupo no.

La optimización de forma (SO) mantiene en el diseño una topología fija, pero cambia su forma o localización de sus nodos [6].

De manera general, como se nombró anteriormente los problemas presentes en este caso pueden dividir en problemas continuos y discretos:

SO CONTINUOS.

En este caso se encuentran problemas dentro el contexto de estructuras continuas 2D o 3D. Normalmente en un problema planteado en 2 Dimensiones la forma es delimitada por curvas, y en 3 Dimensiones la forma de este es delimitada por superficies.

En la Figura 12 se puede observar un problema de una estructura en forma de T, donde es delimitado por curvas, a las cuales se les ubica

[2y 2] Figuras Modificadas de [6]

nodos imaginarios para poder ejecutar la optimización, entre más nodos tenga más cercano a la realidad será (conceptos de elementos finitos).

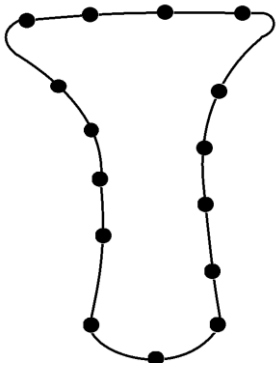


Figura 12. Problema de Optimización de forma continua

Para la solución de este tipo de problemas se ha utilizado aproximaciones de elementos finitos[30], procedimiento común en los métodos basados en gradientes, donde el cálculo de la sensibilidad (parámetro que representa que tan impactada será una variable dependiente con respecto a una variables independiente, la cual matemáticamente es representada por las derivadas de la función objetivo y de las restricciones respecto de las variables de diseño) es la parte que más consume tiempo de computo.

El método de homogenización ha sido aplicado a la resolución de problemas de este tipo [31], este método consiste básicamente en reducir el problema original el cual posee factibilidad de solución a un problema simplificado que mantenga y refleje las características importantes para su solución, esta simplificación se realiza por medio de coeficientes que dependen significativamente de la estructura a tratar, matemáticamente se rige por ecuaciones en derivadas parciales con los coeficientes nombrados anteriormente.

La optimización estructural evolucionaria (ESO) y sus derivados como BESO y GESO (ver 1.4.2.3) también han sido aplicados a la resolución de problemas continuos de optimización de forma.

El uso de algoritmos genéticos también ha sido muy común, se ha hecho optimización de forma para una estructura continua en 2 dimensiones usando algoritmos genéticos a través de funciones B-spline [32], *"es una función spline (curva definida en porciones mediante polinomios) es aquella que tiene el mínimo apoyo con respecto a un determinado grado, suavidad y partición del dominio. Un teorema fundamental establece que cada función spline de un determinado grado, suavidad y partición del dominio, se puede representar como una combinación lineal de B-splines del mismo grado y suavidad, y sobre la misma partición"* definición sacada de [33]; y se ha ido ampliando a la optimización de estructuras ovalas simétricamente axiales, y la optimización de modelos de elementos finitos en 3 dimensiones. [34-35].

SO DISCRETOS

La optimización discreta de forma se logra a través de variaciones en la geometría de armaduras, y a través del cambio de la posición de nodos en las estructuras.

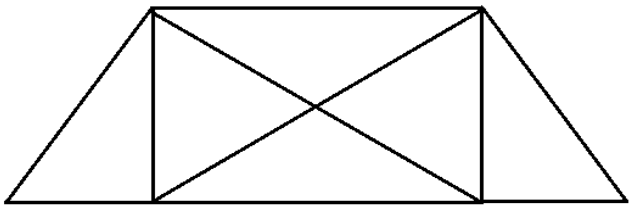


Figura 13. Problema de Optimización de forma discreta

En la Figura 13 se puede observar el ejemplo de una cercha donde el objetivo es determinar cambios de posición en los nodos.

Para la solución de este tipo de problemas se han utilizado varios métodos de programación matemática como:

La programación lineal [36], el cual es un algoritmo matemático, (Clásico según previa clasificación), en donde el problema consiste en

optimizar una función objetivo lineal, cuyas restricciones también deben ser lineales, por medio de sistemas de ecuaciones lineales, su expresión matemática general (definición tomada de [2])

$$\min c^T x$$

$$Ax = b$$

$$x \geq 0$$

$$x \epsilon R^n, c \epsilon R^n, A \epsilon R^{mxn}, b \epsilon R^m$$

Ecuación 9. Definición matemática de la programación lineal

La programación no lineal [37], donde debe contener algúntérmino no lineal, ya sea en sus restricciones o cualquiera de los componentes del problema, según [2] su definición matemática es la siguiente.

$$\min f(x)$$

$$g(x) = 0$$

$$h(x) \leq 0$$

$$l \leq x \leq u$$

$$f: R^n \rightarrow R$$

$$g, h : R^n \rightarrow R^m$$

Ecuación 10. Definición matemática de la programación no lineal

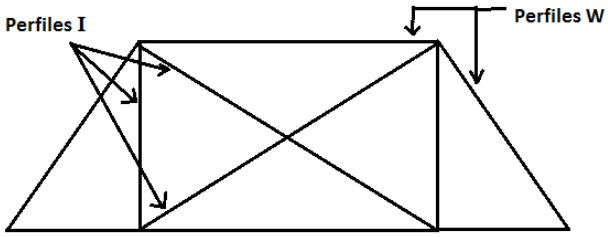
La programación dinámica [38], es utilizada en problemas donde el sistema evoluciona con el tiempo, en otras palabras se busca un conjunto de soluciones que optimizan un comportamiento polietápico (conformado por varias etapas de desarrollo).

Existen varios tipos de programación dinámica: la homogénea, no homogénea, la determinista y la aleatoria [38].

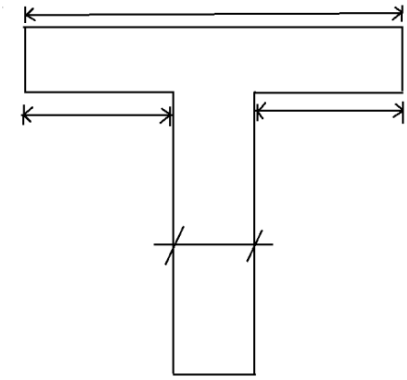
Para la optimización discreta de forma también se ha visto influenciada por la computación evolucionaria, se han utilizado los algoritmos genéticos controlado por lógica confusa para optimizar la forma de armaduras planares y espaciales[39], otra técnica heurística que se ha

utilizado para este tipo de optimización son las estrategias evolucionaras, utilizado para optimizar grandes cerchas como se muestra en [40].

1.4.1.2 OPTIMIZACIÓN DEL DIMENSIONAMIENTO DIRECTO (SIZING OPTIMIZATION)



a. Problema discreto de dimensionamiento directo



b. Problema continuo de dimensionamiento directo

Figura 14. Problemas de Optimización de dimensionamiento directo

Características como la sección transversal, o dimensiones óptimas de un sistema estructural en general son las que son tratadas por la optimización de dimensionamiento directo.

En la Figura 14 se observan ejemplos de problemas del tipo de optimización de dimensionamiento directo, en la Figura 14a se observa un problema discreto, en donde se busca la determinación de la sección

transversal de la cercha, mientras que en la Figura 14b se puede observar un problema continuo donde lo que se busca es establecer las dimensiones optimas de una estructura con forma establecida.

Se considera en este tipo de planteamientos una topología y forma determinada, en consideración de esto se podría decir que este tipo de optimización es la más fácil de determinar.

Según [41]la optimización de dimensionamiento directo en los sistemas estructurales consiste en llegar a los valores óptimos de área de sección A de los miembros que minimice la función objetivo F, que por lo general en la optimización estructural es el peso W, dicho diseño también tienen que satisfacer mínimas limitaciones de desigualdad que limitan los valores de la variables de diseño y la respuesta estructural. Por lo tanto, el problema de diseño se puede expresar como

$$Minimizar\ F = WA = \gamma \sum_{i=1}^{n} L_i A_i$$

$$Sujeto\ a\ G_j \leq G_j(A) \leq G_j,\quad j = 1,2,\dots,q$$

Ecuación 11. Planteamiento matemático de un problema de optimización estructural de dimensionamiento directo.

Donde L_i es la longitud del elemento, y $\gamma = densidad\ del\ material$

Investigaciones en la aplicación de métodos clásicos de optimización a este tipo de optimización se pueden encontrar en [42] y el cálculo de sensibilidades de estos métodos en [43], en este caso de manera paralela.

Los métodos heurísticos también han sido utilizados en este tipo de problemas; muestra de esto es el uso de los algoritmos genéticos para la determinación de secciones transversales óptimas para barras [44], cuya investigación se ha ampliado a armaduras generales y al diseño de secciones I soldadas pertenecientes a marcos, realizándose al

tiempo comparaciones del rendimiento de estos con otros métodos y algoritmos de optimizaciones no lineales [45].

También se ha usado el método de armonía[41],este es un método heurístico ya que se basa en un fenómeno natural, dicho algoritmo está basado en los procesos naturales de la interpretación musical que se produce cuando un músico busca un mejor estado de armonía como por ejemplo la improvisación de jazz. Esta improvisación busca encontrar la armonía musical más agradable (perfecto estado) lo cual se determina por una norma estética, así como el proceso de optimización trata de encontrar una solución global (perfecto estado) determinada por una función objetivo. El tono de cada instrumento musical determina la calidad estética, así como el valor de la función objetivo es determinado por el conjunto de los valores asignados a cada variable de decisión.

Más métodos heurísticos, como las estrategias evolucionarias se ven aplicadas a la resolución de este tipo de optimización [40], y similarmente a la optimización de forma, la optimización estructural evolucionaria (ESO, BESO y GESO) también es aplicada en esta rama.

1.4.1.3 OPTIMIZACIÓN DE LA TOPOLOGÍA (TOPOLOGY OPTIMIZATION)

También conocido como Diseño Optimo Topológico (TOD, “TopologicalOptimumDesign”) de manera general modifica la geometría y la topología, la idea de este es buscar una distribución de materia en un dominio dado (determinando bordes, perforando la estructura etc.) se llegue al diseño más óptimo, ejemplos de optimización de estructuras sencillas [46] Se puede decir que es una disciplina reciente dentro del campo de la optimización estructural por lo que ha sido un área de ardua investigación en los últimos cuarenta años donde los primeros investigadores (entre 1970 y 1980) usaban métodos clásicos

de optimización (Ver 1.2.1). Al igual que SO, la optimización topológica se puede dividir en dos grandes grupos.

TOD CONTINUOS.

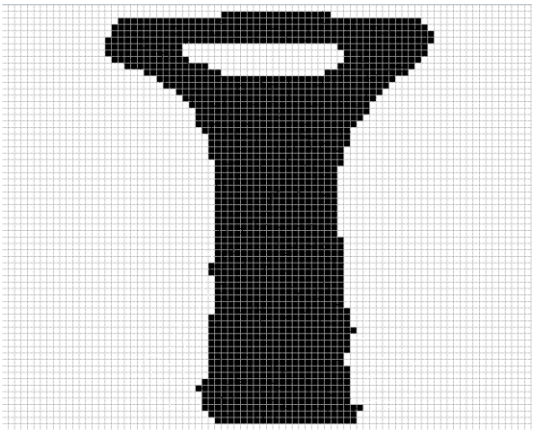


Figura 15. Problema Continuo de Optimización de topología

En este caso el diseño del dominio es discretizado en muchos elementos pequeños y rectangulares (Cuadricula) donde cada rectángulo será llenado de material o de vacío como se puede observar en la Figura 15, cuyo ejemplo planteado consiste en la determinación topológica de un elemento en forma de T. Esta optimización se diferencia principalmente a la de forma en que la topológica permite realizar perforaciones a la estructura mientras que en la de forma no es posible.Se podría decir que posiblemente los métodos de homogenización dentro de los métodos clásicos son los más utilizados en este tipo de problemas[31], donde cada elemento en la grilla está compuesto de un material con densidad variable continuo en la dirección vertical y orientación positiva y la idea consiste crear una caracterización topología a partir de la densidad, es decir, se le atribuye un valor de densidad a cada rectángulo de la grilla, en el caso de los espacios vacíos se titulan con densidad 0;así se consigue una distribución ficticia del material donde alguna veces para traerlo a una

solución ingenieril se utilizan métodos de penalización. Los métodos heurísticos, han profundizado más en este campo, el método de optimización estructural evolucionaria [47] (Evolutionary Structural Optimization ESO), propuesto por Xie y Steve en 1992, está basado en una simple idea de una estructura optima, con máxima rigidez pero con mínimo peso, sigue el concepto de remover ligeramente ciertos elementos, aunque el nombre de este método es algo confuso debido a que no está basado en los principios de computación evolucionaria (Evolutionary Computing “EC”) pero como evolución puede ser entendido de manera general también como una remoción que se realiza lentamente del material que forma parte del diseño cuando este es ineficaz para la estructura.

Dicha eficacia se determina a partir de la tensión presente en el elemento, esto significa que los elementos que presenten una tensión muy baja resultan ineficientes. Básicamente la aplicación del método sigue los siguientes pasos: Primeramente se decide el volumen que va a ocupar la estructura original, luego se divide este dominio en una malla de elementos finitos, se aplican las cargas, propiedades de apoyo, se establece propiedades del material. Luego se analiza la respuesta de cada elemento a estas y se procede a eliminar los que no cumplan con los parámetros de mínima tensión, (criterio de esta se puede encontrar en [48]).

Se han creado muchos derivados de este método, los cuales añaden ciertas características nuevas como la optimización estructural evolucionaria aditiva, (Addittive Evolutionary Structural Optimization “AESO”), que como su nombre lo indica los elementos se incorporan donde son necesitados en la estructura, la idea consiste primeramente determinar el volumen máximo que esta pueda ocupar, luego se establece un número mínimo de elementos suficientes para conectar

los apoyos y poder cargar y analizar sus resultados y con el criterio optimizador (tensión en este caso) añadir a los que necesiten.

Otro derivado es la optimización estructural bidireccional (BESO) [49], que consiste en una combinación de las dos metodologías nombradas anteriormente, esto significa que en este caso es posible remover o añadir material.

Luego fue desarrollada la optimización estructural evolucionaria morfológica (Morphing Evolutionary Structural Optimization “MESO”), que se aplica en problemas donde se presenten variables como el espesor (Variables con múltiples posibilidades, que tomen más de dos valores), ya que el método clásico no está preparado para trabajar con este tipo de variables porque en su formulación solo existen dos opciones, el elemento está presente o no, sin embargo en este caso se utilizan un conjunto de variables discretas para definir el elemento.

La última modificación fue realizada en el 2008, propuesta por XiaLiu[50] es la optimización estructural genética evolucionaria (Genetic Evolutionary Structural Optimization “GESO”), la cual se considera el más reciente avance en cuando a optimización estructural evolucionaria se refiere. Se basa en la integración de algoritmos genéticos, en el desarrollo de este método cada elemento perteneciente a la malla de los elementos finitos es independiente y posee su propio número de aptitud el cual depende de su sensibilidad, el principio se tendrá un dominio que constituye toda la población del algoritmo genético, después de cierto número de generaciones convergirá en un resultado óptimo.

Los algoritmos genéticos en general logran un diseño topológico óptimo de material o vacío, reasentándolo por medio de una matriz de manera que el peso de la estructura se reduce al mínimo estando está sometido a restricciones de desplazamiento y/o a restricciones de esfuerzo.

TOD DISCRETOS

Consiste en determinar la óptima conectividad de los elementos dentro de las posibles conexiones, este número aunque es finito es bastante extenso según el problema a analizar.

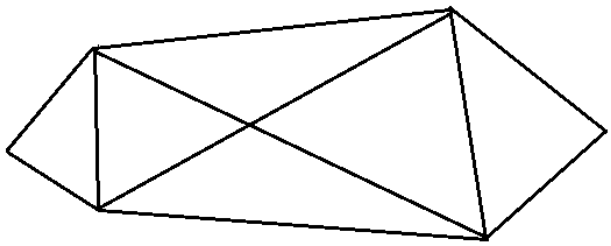


Figura 16. Problema discreto de Optimización de topología

En la Figura 16 se intenta representar los efectos de la optimización topológica a una cercha, la idea consiste en crear varios tipos de conexiones posibles, es decir que cierto elemento no se conecte a cierto nodo, sino a otro dentro de la estructura.

La aplicabilidad de los métodos clásicos como la programación lineal ha demostrado buen desempeño para pequeños problemas, caso contrario, cuando se trata de un problema extenso donde el aumento de variables de diseño o partes de la grilla de elementos finitos es significativo provocando en estos métodos una toma de tiempo y de recursos bastante extensa; además de esto estos métodos presentan otra desventaja la cual consiste en la naturaleza discontinua de las funciones de los problemas de diseño, sin embargo se siguen realizando investigaciones como mejorar estos métodos (programación lineal, cuadrática) para la optimización topológica a través de las mejoras que se pueden realizar a las restricciones de los problemasy el cálculo de sensibilidades aplicando programación en paralelo [51], esta

aplicación ha permitido que estos métodos puedan resolver problemas de optimización de grandes estructuras en tres dimensiones [52].

De ahí que la utilización de métodos heurísticos para la resolución de problemas de este tipo sean extensas comparadas con las clásicas como por ejemplo el uso de computación evolucionaria para la optimización de estructuras de tres dimensiones [53], la aplicación de algoritmos genéticos es basta, se han utilizado para optimizar armaduras en polinomios, armaduras en tres dimensión, se ha combinado con lógica difusa. [54, 55, 56].

2. PROGRAMACIÓN EN PARALELO

Los problemas modernos de ingeniería los cuales poseen más semejanza con la realidad implican soluciones más complejas y difíciles de calcular. En los últimos años en todas las ramas de la ciencia, estos problemas se han proliferado creando una necesidad de intentar mejorar los procedimientos ya existentes.

Las herramientas computacionales han jugado un papel importante en este proceso de cambio y de satisfacción en esta necesidad ya que desde la aparición de dichas herramientas se abrió un nuevo mundo lleno de posibilidades de procedimientos y les permitió a los diseñadores poder probar más hipótesis sin necesidad de arduas jornadas de cálculo.

El software normalmente se ha realizado con procesos en serie, en los cuales para resolver un problema se construye un algoritmo y se implementa en un flujo de instrucciones en serie, esto quiere decir que avanzan linealmente, y que se realiza un instrucción al tiempo(hasta que termine una instrucción puede empezar la otra).

Todas estas instrucciones son desarrolladas por un centro de procesamiento; el hecho de que los problemas sean más complejos ha obligado a que esta unidad de procesamiento sea mejorada para que dichos problemas sean solucionados en un tiempo adecuado.

Estas mejoras se han realizado a través de técnicas como la segmentación y también a través del aprovechamiento de ciertos avances tecnológicos como los que involucran integración de circuitos los cuales permiten acelerar la velocidad de cómputo, actualmente se ha alcanzado un límite en este tipo de mejoras, y para poder sobrepasarlos se requerirá una inversión bastante extensa de recursos, lo cual nos obliga a buscar otras alternativas de mejoras en nuestro procedimiento de cálculos.

La computación en paralelo es una de estas alternativasque nos puede brindar más rendimiento con pocos recursos. Básicamente se basa en que varios computadores (de bajo rendimiento), pueden conseguir un buen rendimiento cuando conforman un multi computador, ya que un computador secuencial, la velocidad de estará limitada por la tecnología disponible.

Es fácil aumentar el rendimiento de un multi computador aumentando el número de procesadores, comparado con tratar de aumentar el rendimiento de un computador con un solo procesador. Los multi computadores también son llamados clusters de estaciones de trabajo y ofrecen una excelente relación beneficio/Costo comparado con un supercomputador. Aunque todas estas ventajas en este tipo de técnica han sido permitidas por el desarrollo en las interconexiones, que nos dan buenas velocidades de comunicación.

Los problemas paralelizables son comúnmente clasificados de acuerdo al manejo de la comunicación entre sus partes, los cuales son problema de grano grueso, que es aquel en donde muchos de los cálculos se pueden manejar en paralelo y la comunicación está siendo a menudo mantenida entre unos pocos, pero con grandes mensajes, y los problemas de grano fino, que por el contrario requieren una comunicación frecuente, y los mensajes son en su mayoría pequeños, ejemplo de este último seria la realización de una transformada rápida de Fourier.

Dependiendo del número de paralelizaciones realizadas en el problema, se define el nivel del paralelismo. Siendo paralelismo de un nivel aquel en donde se aplica un método de paralelización y cada una de las divisiones trabaja linealmente hasta ahorrar resultados, como se puede observar en la Figura 17, donde solo se aplica paralelización una sola vez.

Por el contrario la paralelización multinivel es aquella en donde el problema inicial se paraleliza, y luego cada una de esas paralelizaciones puede ser nuevamente paralelizada, el límite de estas paralelizaciones serán impuesta por el método o por el diseñador, este comportamiento se puede observar en la Figura 18.

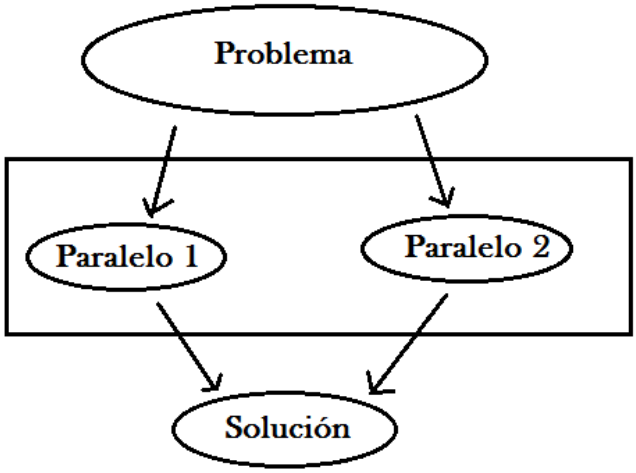


Figura 17. Paralelismo único nivel

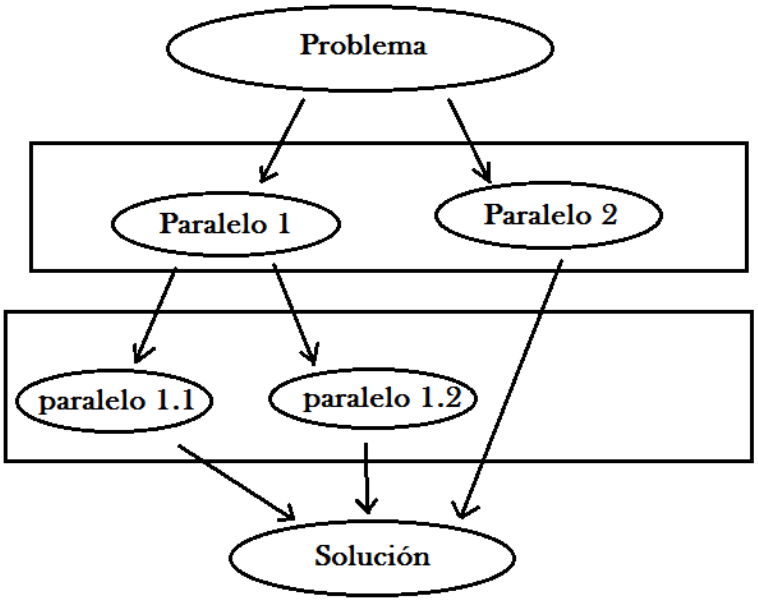


Figura 18. Paralelismo multinivel

2.1 EVALUACION DE ALGORITMOS PARALELOS

Una pregunta común en cualquier proceso es ¿Cómo mido su eficiencia?, en el caso de los algoritmos paralelos, se tienen en cuenta cierto tipos de características para poder evaluar su rendimiento.

2.1.1 TIEMPO DE EJECUCIÓN

Según [57] se podría decir que el tiempo de ejecución es el índice de prestaciones más instintivo. Se podría definir como un parámetro absoluto ya que permite medir la rapidez del algoritmo sin necesidad de cualquier tipo de comparación con otros algoritmos.

Si se habla de un programa secuencial, el tiempo de ejecución es el tiempo transcurrido desde que se inicia su ejecución hasta que finaliza, pero en el caso de un programa en paralelo este tiempo se define como el tiempo que transcurre desde el comienzo de la ejecución del programa en el sistema en paralelo hasta que finalice el último procesador.

En el caso de sistemas paralelos con memoria distribuida el tiempo de ejecución paralelo con *p* procesadores T_p, se puede determinar de modo aproximado mediante la fórmula

$$T_p \approx T_A + T_C - T_{SO}$$

Ecuación 12. Calculo del tiempo de ejecución en paralelo

Donde T_A es el *tiempo aritmético*, que se define como el tiempo que tarda el sistema multiprocesador en realizar todas las operaciones aritméticas, este tiempo se expresa en FLOPs, el cual es una medida que indica la velocidad que tarda el procesador en hacer una operación aritmética en un punto aleatorio; este tiempo depende de dos parámetros, uno es el tiempo de envío de una palabra (τ) y el otro es que transcurre desde que dicho mensaje hasta que circula y se denota (β).

$T_C$es el *tiempo de comunicación* que se define como el tiempo que tarda el sistema en ejecutar transferencias de datos, y T_{SO} es el *tiempo de soplamiento* que se define como el tiempo que pasa cuando las operaciones aritméticas y de transferencias están siendo ejecutadas al mismo tiempo.

Muchas veces el tiempo de soplamiento puede ser difícil de calcular tanto teóricamente como experimentalmente, y por el contrario puede tener gran relevancia en el tiempo total de ejecución del algoritmo paralelo. Sin embargo su ausencia obliga a que se haga la aproximación de la expresión sin este término por lo que la expresión recalcula como:

$$T_P \approx T_A + T_C$$

Ecuación 13. Calculo del tiempo de ejecución sin tiempo de soplamiento

2.1.2 GANACIA DE VELOCIDAD (Speed-Up)

Indica la ganancia que se ha obtenido que se ha obtenido con una ejecución en paralelo [57], dicha ganancia de velocidad (S_P) para *p* procesadores, se calcula por medio de la división entren el tiempo de ejecución de un programa secuencialmente (T_S), como es usual pueden haber muchas versiones de este tiempo secuencialmente en este caso se toma la ejecución más rápida, y la versión paralela de dicho programa en *p* procesadores (T_P).

$$S_p = \frac{T_S}{T_P}$$

Ecuación 14. Cálculo de la ganancia de velocidad.

En el mejor de los casos el tiempo en paralelo será *p* veces inferior al de su ejecución secuencialmente, considerando que en el caso del paralelo que todos los procesadores tienen igual potencia de cálculo; lo que significa que al ser el mejor caso es el máximo valor que puede alcanzar el Speed-Up es

el número de procesadores, pero este caso se considera poco frecuente ya que es necesario contar la sobre carga que implica un problema resuelto en paralelo como lo es las sincronizaciones y dependencias entre ellos.

2.1.3 EFICIENCIA

En [57] la eficiencia es el cociente entre la ganancia de velocidad y el número de procesadores utilizados; esto representa el grado de aprovechamiento de los procesadores en la resolución del problema.

$$E = \frac{S_p}{p}$$

Ecuación 15. Cálculo de la eficiencia

Ya que el valor máximo que puede tomar la ganancia es el número de procesadores utilizados (p), el mayor número de eficiencia que se puede alcanzar es 1, lo que implica que se aprovechó 100% los procesadores.

2.1.4 ESCALABILIDAD

Es habilidad de cierto algoritmo o red de ampliar su número de operaciones sin perder calidad, en otras palabras mantener su mismo rendimiento cuando se aumente el número de procesadores y el tamaño del problema en la misma proporción. En conclusión la escalabilidad nos muestra que tanto ha aprovechado el algoritmo un incremento en recursos computacionales.

En [57] se puede observar que para caso de un algoritmo paralelo, es lo mismo; cuando es escalable mantiene su eficiencia a pesar de ampliaciones en los procesadores y en el problema, por el contrario cuando no el algoritmo no es escalable aunque se incremente el número de procesadores no se conseguirá aumentara la eficiencia, lo que significa que aprovechara menos la potencia ofrecida por el procesador.

Dicha medida se puede evaluar por diferentes métricas, y para poder elegir la métrica ideal es necesario tener en cuenta las características del problema con el que se está tratando. Un ejemplo es el Speed-Up escalado,

el cual aumenta su facilidad cuando hay resultados experimentales disponibles, su expresión:

$$S_P = \frac{T_S(kW)}{T_{KP}(kW)}$$

Ecuación 16. Cálculo de la Escalabilidad.

Donde W es el costo teórico computacional del algoritmo secuencial. El comportamiento del escalado al variar k nos muestra cómo cambia el Speed-Up cuando se cambia el número de procesadores y el tamaño del problema.

2.2 HARDWARE

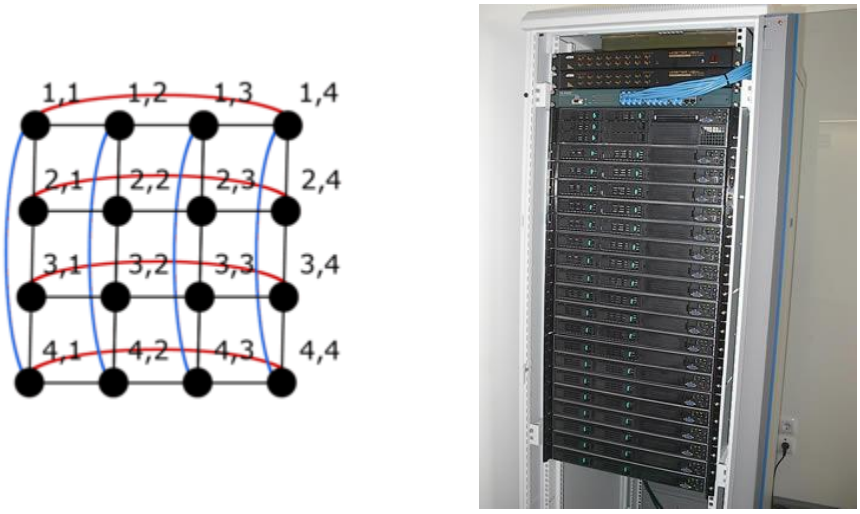
La estructura del hardware para ejecutar programación en paralelo se basa en las dos posibilidades de trabajo de sus memorias: memoria compartida y memoria distribuida; cuando se realiza el uso de memoria compartida se trabaja por ende con sistemas en los cuales todos los procesadores comparten una sola dirección espacial; para el caso de la memoria distribuida sus nodos (componentes) son computadores personales o estaciones de trabajo las cuales conectadas mediante una red de interconexión que permiten el paso de mensajes, perteneciente a este grupo están los clusters conformados por nodos multiprocesadores, esto implica un paralelismo de dos niveles desde el punto de vista lógica: el primero, que corresponde a un esquema distribuido, los nodos están conectados con una red de interconexión, el segundo es un nivel local que aplica a un esquema compartido, ya que los procesadores comparten una misma memoria física.

En [57] podemos observar dos ejemplos de dos clustersutilizados en cierto problema: máquina de Kefren(Figura 17)[4]

El *kefren* es un clusters de memoria compartida, el cual trabaja bajo el ambiente Linux, Red Hat 8. En compuesto por 20 nodos biprocesadores marca Pentium

[4] Figura sacada de [57]

Xeon a 2 Ghz con 1 Gigabyte de memoria RAM, y están conectados por medio de SCI con topología de Toro 2D en malla de 4 x 5 (Figura 17a); cada uno de los nodos está disponible para cálculo científico.



a. Red de interconexión SCI b. Vista Frontal

Figura 19. Máquina de Kefren

Máquina de Rosebud, esta máquina paralela es un clusters heterogéneo compuesto por 6 ordenadores conectados por una red Fast-Ethernet. Estos 6 ordenadores se agrupan en grupos de dos ordenadores, y poseen diferencias en su potencia de cálculo y en su arquitectura.

El primer par de computadores, se les otorga los nombres de rosebud01 y rosebud02, con una memoria RAM de 1 Gbyte, los cuales tienen procesadores Pentium IV con velocidades de 1.6 GHz y 1.7 GHz respectivamente.

El segundo par cuyos nombres son rosebud03 y rosebud04, están compuestos por biprocesadores Xeon con 3.5 Gbyte de memoria RAM y procesador 2.2 GHz.

2.3 SOFTWARE

Para la aplicación de la computación en paralelo se necesita tener unas herramientas de software:el ambiente paralelo de programación, el lenguaje de programación y las librerías secuenciales y/o paralelas.

Con el paso del tiempo se han creado modelos los cuales poseen las características nombradas anteriormente, entre esos modelo tenemos:

2.3.1 PTHREADS[58]

Llamado también sistema de operación interface portable (Portable Operating System Interface “POSIX”), utiliza un conjunto de lenguajes de programación “C” con una memoria compartida, esto quiere decir que todos los procesadores van a la misma memoria a retirar información y a depositarla. Los hilos son su unidad de trabajo, técnicamente un hilo es una serie de instrucciones definidas que pueden ser programadas a correr como un sistema operativo y para la administración de estos, desde la creación hasta la destrucción es necesaria una programación explicita donde una orden especifique estos procesos.

Cuando varios hilos entran a la memoria compartida, se debe tener cuidado con las secciones criticas de esta, para ello los programadores utilizan el sistema de mutex, que permiten la entrada de un solo hilo a la vez; o el semáforo donde se permite la entrada de varios hilos a la zona critica.

La programación paralela con esta librería se hace a un bajo nivel, por ende se vuelve más complejo realizar y llevar a un punto alto de rendimiento y adaptabilidad, de ahí que su uso se haya restringido principalmente para la programación de sistemas operativos.

2.3.2 OPENMP[59]

Es una interfaz para la programación multiproceso de un paralelismo de memoria compartida. Consiste en un conjunto de directivas de compilador rutinas de biblioteca, y variables de entorno que modifican el tiempo de ejecución; dichas variables de entorno que se extienden a Fortran, C y C++ y su unidad de trabajo son los hilos.

Este es un modelo portable que proporciona a los programadores una interfaz simple para el manejo de muchas aplicaciones paralelas.

No es necesario para los programadores administrar el tiempo de vida de los hilos, solo se especifica el número de hilos a usar.

Para este caso la partición de cargas de trabajo requiere un esfuerzo relativamente mayor, OpenMP también abstrae la carga de trabajo (por ejemplo una matriz) y la divide en tareas, y estas son asignadas a los respectivos hilos.

Esta arquitectura soporta varias construcciones donde los programadores tienen la libertad de especificar donde se produce la sincronización, estas sincronizaciones son con el fin de proteger los conflictos de datos que se presentan cuando ha existido un inadecuado uso de variables compartidas.

Comparada con Pthreads se pude programar a un nivel superior, y es muy sencilla de usar ya que sólo 3 o 4 directivas o pueden implicar un paralelismo significativo.

2.3.3 CUDA[60]

Siglas de Compute Unified Device Architecture, el cual hace referencia a cierta arquitectura con un conjunto de herramientas de desarrollo creadas por Nvidia, es una extensión de lenguaje de programación C diseñado para soportar procesamiento en paralelo.

CUDA busca encontrar las ventajas de las GPUs (Unidades de procesamiento gráfico), al utilizar el paralelismo que ofrecen los núcleos múltiples, que por su estructura permite el lanzamiento de gran número de hilos múltiples. Por ende si una aplicación está diseñada utilizando gran número de hilos que realizan tareas independientes, este modelo ofrecerá un gran rendimiento.

Este modelo de computación en paralelo contiene tres puntos clave, los cuales son la jerarquía de grupos de hilos, las memorias compartidas y las

barreras de sincronización; dicha jerarquía se compone de dos niveles, el bloque y la grilla, donde un bloque es un conjunto de temas cercanamente acoplados, cada hilo perteneciente es identificado con un respectivo identificador de hilo y la grilla es un conjunto de bloques flexibles con tamaños y dimensiones similares, pueden haber bloques de hilos conformados por máximo 512 hilos distintos.

En este modelo la creación y destrucción de hilos está implícita, esto quiere decir que los programadores no deben especificar la creación y destrucción de estos, solo deben especificar las dimensiones de los bloques y grillas requeridas para el proceso de cierta tarea.

2.3.4 MPI (Message Passing Interface)[61]

Es una especificación para el paso de mensajes operacionales, diseñada para ser utilizada en programas que usen la existencia de múltiples procesadores, ya que aporta sincronización entre cada uno de los procesos. Entre sus características está que define cada unidad de trabajo como un proceso, hace uso de memoria distribuida y no de memoria compartida, esto quiere decir que cada procesador posee su propio espacio en la memoria.

Si el proceso que envía el mensaje espera que dicho mensaje sea recibido se habla de un paso síncrono, caso contrario, se habla de paso asíncrono si el proceso no espera que el mensaje sea recibido si no que por el contrario continúa su ejecución, creando la posibilidad de que se produzca un nuevo mensaje, crea la necesidad de buzones, donde se almacenan mensajes para esperar que un proceso los reciba.

En otras palabras es la comunicación estándar entre los nodos que ejecutan un programa en un sistema en paralelo con memoria distribuida, para la realización de esta arquitectura se utilizan lenguajes C, C++, Fortran y Ada.

Tiene grandes ventajas debido a su disponibilidad, distribuciones prácticamente en cualquier arquitectura, también permite comunicaciones punto a punto mediante operaciones de envío y recepción. Permite además operaciones colectivas.

Su administrador de trabajo de cargas es similar al utilizado en POSIX.

En el diseño de este modelo, se tuvo en cuenta todas las características más atractivas de los pasos de mensajes ya existentes.

2.3.5 UPC[58]

Paralelo Unificado C, es un lenguaje de programación en paralelo para arquitectura de memoria compartida y distribuida,

El espacio global se divide en ramificaciones de hilos, en donde cada hilo posee dos tipos de accesos a memoria, un acceso a su propio espacio de direcciones privadas, y otro acceso a espacios comunes a todos los hilos, para el acceso a estos dos espacios se utiliza la misma sintaxis, y el desempeño de estos accesos se mejoran con el concepto de afinidad de subprocesos.

En UPC, el administrador de carga de trabajo es implícito, mientras la partición de dicha carga de trabajo puede ser explícita o implícita, los programadores solo necesitan especificar el número de hilos requeridos durante la llamada de comando de línea.

La comunicación entre hilos es adoptado por la partición global de la dirección espacial (PGAS) realizando el uso de punteros, hay 3 tipos de comunes de punteros que se usan en UPC: el puntero privado donde el punto privado posee su dirección espacial privada, puntero a compartir privado donde el puntero del punto privado ha compartido su dirección espacial, y el puntero compartido donde los punteros compartidos provienen de un espacio compartido.

2.3.6 FORTRESS[58]

Es un lenguaje de programación diseñado para computación de alto rendimiento, en esto caso la unidad de procesamiento son los hilos. Su administrador de carga de trabajo puede ser implícito o explicito, para el caso en el cual es implícito los programadores no tiene que especificar que hilos tiene que ser corridos en cada iteración, en el caso explicito la creación de hilos pueden ser hechos usando una palabra clave.

Para evitar un comportamiento anormal en las carreras de datos en un programa, los programadores tienen que especificar la sincronización explícitamente.

El nombre de Fortress viene del lenguaje Fortran, aunque su lenguaje no está diseñado para ser similar a Fortran.

Fortress está diseñada para ser altamente paralelas y tiene una rica funcionalidad contenida dentro de las bibliotecas a partir de Java, y sintácticamente es lo más parecido a Scala.

2.4 AMBIENTE DE COMPUTACIÓN EN PARALELO BASADOS EN INTERNET EMPLEANDO ELEMENTOS FINITOS

Hoy en día se nos ofrece más posibilidades para la resolución de problemas aun cuando no se cuente con herramientas de hardware y software, se puede acceder a metodologías a través del internet.

Vale la pena resaltar que en el ámbito colombiano existe la red RENATA[5] (Red Nacional Académica de Tecnología Avanzada) de tecnología avanzada que conecta comunica y propicia la colaboración de la comunidad académica y científica del país con la comunidad académica internacional, dentro de los servicio prestado por RENATA está el procesamiento masivo y distribuido C hecho posible mediante cooperación y paralelización de recursos informáticos que

[5] Mayor información http://www.renata.edu.co/

aportan las distintas instituciones que integran la red, algunos de ejemplo de procesamiento masivo distribuido son: malla computacionales (Grids), almacenamiento distribuido, servidores espejos (Mirroring), súper computación y clusters. El uso de RENATA en la ciudad de Bucaramanga se hace a través de Unired a 100 Mbps.

Adicionalmente están disponibles vía internet programa de análisis basados en elementos finitos que trabajan en ambiente en computación en paralelo haciendo que cualquier PC pueda convertirse en un Súper computador mientras está conectado al servicio. De esta forma, los usuarios se ven libres del mantenimiento del hardware y Software y siempre tienen acceso a la última versión de la aplicación a través del internet como el propuesto en [77].

Pero no solamente se está ligado a programas de elementos finitos, si no que estas herramientas de soporte se utilizar para realizar una extensa gama de simulaciones. En [78] se encuentra una completa revisión de herramientas de soporte y simulación basadas en internet.

Dentro de las ventajas de este enfoque se pueden nombrar la facilidad de uso, su carácter colaborativo y la reutilización de modelos entre otro. Sin embargo, existen algunas desventajas que incluyen la pérdida en velocidad, la vulnerabilidad en las seguridad y la estabilidad del sistema. De forma general un gran número de estas desventajas están siendo solucionadas a medidas que las tecnologías de la red evolucionan.

2.5 TÉCNICAS DE PROCESAMIENTO EN PARALELO PARA LA INGENIERIA ESTRUCTURAL

En [62] se presentan técnicas de procesamiento en paralelo, que son aplicados a los elementos finitos lo que de manera general puede significar que son aplicados en la ingeniería estructural, y estos son:

2.5.1 DESCOMPOSICIÓN DE DOMINIO

Esta técnica está basada en el concepto de la técnica “divide y vencerás”, donde la tarea a desarrollar es dividida en pares de tareas totalmente independientes, y por consiguiente la comunicación entre procesadores es reducida, lo cual también minimiza las necesidades de coordinación que en cierta forma se podría decir que poseen más dificultad al momento de implementar una técnica paralela.

La idea consiste en una descomposición de dominio o espacial, es decir, el dominio se divide en regiones, por ende el problema se descompone en la solución de problemas de contorno de subdominios. La solución de la interfaz de los subdominios es lo que diferencia a los diversos algoritmos.

Dentro de la descomposición de dominio existen muchas vertientes, ya que dicha descomposición puede incluir traslapos entre subdominios o no.

2.5.2 SUBESTRUCTURACIÓN

Esta técnica se puede ser clasificada como una vertiente de la descomposición de dominio que no posee traslapo entre sus subdominios. Fue introducida en la década de 1960 con el fin de reducir la dependencia en la memoria fuera de los núcleos en aplicaciones de gran escala en ingeniería estructural. La idea básica consiste en subdividir una estructura en una serie de subestructuras que comparten en común solo los nodos interiores. Estas subestructuras poseen grados de libertad (DOF)que se clasifican en grados libertad interna y grados de libertad interfaz. El proceso consiste en la aplicación de condensación estática para reducir el número de grados de libertad en cada estructura, y se renombran las incógnitas de los grados de libertades interiores de todas las subestructuras y por últimos los grados de libertad de interfaz.

La condensación estática es cuando la matriz de rigidez condensada de una subestructura permite expresar las fuerzas elásticas de los bordes en función de estos mismos límites.

2.5.3 OPERADOR DE DIVISIÓN

Esta técnica es una generalización de la subestructuración (técnica nombrada anteriormente), lo que indica que pertenece a la filosofía de “divide y vencerás”, para este caso las estrategias de división pueden ser desarrolladas de múltiples maneras y es además lo que le da particularidad al método. Un ejemplo de estas formas es aplicando el método de Hughes (alternar direcciones), en donde un problema multidimensional se reduce a una serie de problemas de una sola dimensión.

2.5.4 ELEMENTO POR ELEMENTO

En esta técnica se aprovecha el hecho de que los elementos de cálculos se pueden realizar de forma independiente, y por ende es un paralelizado trivial. Al estar los elementos de cálculos separados la paralelización solo consiste en atribuir cada uno de estos elementos a un procesador respectivo. Con el objetivo de mejorar la eficacia de las estrategias de EBE, se acompaña con resolución de ecuaciones en paralelo. Para mayor información ver [63].

3. OPTIMIZACIÓN ESTRUCTURAL Y PROGRAMACIÓN EN PARALELO

La programación en paralelo es una herramienta tan versátil que permite su aplicación de diferentes maneras. En la optimización estructural dicha programación ha sido aplicada a todo un método de optimización, o a cierto cálculo especial perteneciente a un método. Creando de esta manera un campo de aplicación tan extenso que los autores (ejemplo P. K. Umesha) solo la han aplicado a ciertas áreas específicas de la optimización estructural. El hecho de que la programación en paralelo sea un área de investigación cada vez más necesaria por la reducción de recursos con la que se cuenta en la actualidad, crea por ende una necesidad de solo enfocarse en los métodos de optimización estructural más representativo, entendiéndose por representativo aquello que hayan mostrado un mejor desempeño y una alta recurrencia global de aplicación.

De manera similar a la gran acogida que han tenido los métodos heurísticos en los últimos años para la solución de problemas de optimización, así también ha sido la acogida de la aplicación de la programación en paralelo a este tipo de optimización, sin embargo no ha sido nula la aplicación de la computación en paralelo a los métodos clásicos de optimización, pero comparados con los métodos heurísticos su énfasis ha sido menor.

De igual manera la particularidad de las respuestas brindadas por cada uno de estos métodos es el punto clave para la selección de uno de ellos ya que esta selección depende de las características del problema estudiado y de los límites de precisión en las respuestas deseadas por el diseñador.

3.1 PROGRAMACIÓN EN PARALELO EN METODOS CLÁSICOS

Como se explicó anteriormente, para efectos de este trabajo, un método clásico es aquel que presenta el uso de derivadas para la resolución de problemas de optimización.

Los métodos basados en gradientes para encontrar un diseño óptimo de una estructura no son fácilmente adaptables para la estrategia de paralelización del tipo “dividir y vencerás”, la cual, como se explicó anteriormente, calcula diferentes diseños independientes, basándose en su repartición de dominio. Casi siempre esta repartición de dominio se realiza con la utilización de un método heurístico (búsqueda aleatoria) el cual mejora los rendimientos del sistema pero no lo hace indispensable.

En [64] K.L. Chan y D. Kennedy, describen nuevas estrategias de distribución de cargas de trabajo, para realizar una paralelización individual de cada uno de los pasos del procedimiento del diseño basado en gradientes (derivadas) utilizando un software llamado viconopt; se realiza el uso de este programa debido a su gran complejidad cuyo enfoque puede ser utilizado en varios tipos de estructuras, pero para el caso de este procedimiento se procedió a realizar el banco de pruebas con un marco simple.

Básicamente viconopt se basa en 7 pasos los cuales son: donde el paso 1 y 7 de este proceso de diseño encuentra un factor crítico de pandeo de carga para los diseños (en el caso del 1 inicial, para el 7 final), luego en el paso 2 estabiliza el diseño inicial en uno “posible” (cumpla mínimas condiciones) reduciendo la variable de diseño que para este caso es el espesor, esta reducción se realiza iterativamente probando diferentes valores de factores y la convergencia de estos se realiza por bisección. Paso 3, se calcula las sensibilidades de las restricciones y de la función objetivo (en este caso masa), a pequeños cambios en las variables de diseño, dichas sensibilidades se utilizan para guiar la secuencia de las modificaciones a realizar al diseño. En el paso 4 se calcula el movimiento de diseño moviéndose en el espacio límite dado por el optimizador clásico (programación lineal) CONMIN. En el paso 5 se encuentra un diseño óptimo por el método de direcciones posibles y en el paso 6 vuelve a estabilizar el diseño aplicando la metodología del paso 2.

En este artículo se usa la combinación e hibridación paralelizando el método actual de análisis (paralelización de grano fino), sobretodo en el procedimiento en el cual se involucra el CONMIN, como se nombró anteriormente para el análisis el programa trabajo con un marco plano obteniendo eficiencias altamente razonables aun cuando el número de procesadores es bastante extenso, obteniéndose en este caso el tiempo de solución más rápido. Los resultados mostraron que la eficacia relativa baja drásticamente al aumentar el número de procesadores.

En [52] Thomas Borrvall y Joakim Petersson realizan una optimización topológica a gran escala usando la computación en paralelo, en este procedimiento se emplea una método de solución iterativa (método del gradiente conjugado), básicamente la iteración es realizada a un gran sistema de ecuaciones lineales y los problemas pequeños de optimización se deben resolver en cada iteración, esta forma de solución posee la ventaja que asintóticamente se requiere menos cálculos, por ende requiere menos almacenamiento y por consiguiente mucha menos memoria que los métodos directos o no iterativos, esto se debe a que los factores de la matriz de rigidez es más densa que la matriz de rigidez en sí, y como el almacenamiento de estos factores no es necesario cuando se utiliza un método iterativo por ende se requerirá menos almacenamiento como se nombró anteriormente.

En cuanto a la paralelización se utiliza la técnica de descomposición de dominio, cuyo código esta principalmente escrito en C++.El sistema de equilibrio adecuado es generalmente mal condicionado por los valores de diferentes densidades que resultan de los cambios en los modelos topológicos, pero esto se resuelve con un acondicionamiento previo del sistema utilizando la diagonal pre condicionadora Jacobi. En cuanto a la optimización consiste en el análisis de sensibilidad y la creación de soluciones a los sub problemas, los cuales están basados en la descomposición de dominio de la ejecución en paralelo.

Este método es confiable en el sentido de que produce diseños en blanco y negro, y el mismo modelo se utiliza para calcular la respuesta elástica y propiedades geométricas además que está libre de anomalías como la malla de dependencia y la inestabilidad del tablero de ajedrez. El código que se utiliza es completamente en paralelo, tanto las partes de equilibrio como las partes de optimización

La mayor parte del tiempo se gasta en la parte de equilibrio, (alrededor del 97%) aunque la eficiencia en paralelo ha demostrado ser alta en esta parte. Concluyen que el método debería ser más útil que solo conceptos de diseños.

En [65] A. Migdalas y G. Toraldo informan de la computación en paralelo aplicada a optimización no lineal, dicha optimización no necesariamente debe ser clásica (en este caso la programación cuadrática), sino que también se presta para optimizaciones heurísticas.

En el caso de los métodos clásicos los problemas de programación cuadrática son muy interesantes debido a que muchas aplicaciones requieren soluciones de esta clase. Por otra parte en general para la optimización no lineal los códigos de la programación cuadrática son componentes básicos de cualquiera de sus algoritmos. Unos de los enfoques paralelos para la programación cuadrática se basa en la propuesta de resolver los grandes problemas que se plantean en un segundo grado de formación de las máquinas, este básicamente consiste en la separación de subconjuntos de puntos de datos de entrenamiento dado que se separa por una superficie al margen de ellos.

En [65] podemos observar que la computación en paralelo puede ser aplicada en los problemas no lineales en 3 niveles de paralelización:

1. *Paralelización de la función y/o de la evaluación de la derivada en el algoritmo.*
2. *Paralelización de los núcleos algebraicos lineales*

3. *Modificación de los algoritmos básicos los cuales incrementan el grado de paralelismo intrínseco y por consiguiente rendimiento de múltiples funciones.*

Para el caso de los 2 primeros han llevado gran cantidad de trabajo cuyo objetivo era primordialmente generar un software paralelo de propósito general, con un alto nivel de robustez, fiabilidad y portabilidad, para garantizar su generalidad.

En cuanto al tercer nivel, se ha venido realizando continuamente una investigación para el uso de la computación en paralelo en la optimización de las necesidades crecientes de resolver problemas cada vez más grandes y aplicados a diferentes campos.

Pero como se nombró anteriormente la programación en paralelo no solo esta aplicada a un método enteramente sino que puede ser aplicada a una sola fase de cálculos de los métodos de optimización, teniendo en cuenta que no todos los problemas son paralelizables.

En [43] P.K. Umesha y M.T Venuraju proponen técnicas de computación en paralelo para el análisis de la sensibilidad en el diseño óptimo de estructuras, ya que entre todas las diferentes actividades y cálculos que requiere una optimización estructural por medio de un método clásico (basado en gradiente), el diseño del análisis de la sensibilidad es el que más tiempo computacional requiere el cual es un componente importante en todos los métodos de optimización por programaciones matemáticas , por ende en el que más recursos se invierte, el cálculo del gradiente en cada iteración involucra incontables cálculos, lo que abre la puerta a la paralelización.

Para el cálculo de las sensibilidades usaron dos formas de cálculo: el primero de ellos es la forma analítica, la cual está basada en la diferenciación directa de las ecuaciones obtenidas cuando las ecuaciones continuas son discretizadas por un método de elementos finitos, esto trae como consecuencia que algunas veces no sea posible diferenciar las ecuaciones obtenidas de un elemento de orden mayor o

de un elemento no lineal. La otra forma es las diferencias finitas, este método es menos eficiente computacionalmente y puede no ser exacto, esta forma tiene una ventaja ya que permite calcular los gradientes de los elementos no importa el orden o su linealidad.

El paralelismo propuesto se basa en su aplicación al cálculo de gradientes y este puede ser aplicado en un solo nivel de paralelismo, en el cual el coeficiente de la matriz gradiente de la función objetivo ($_{\nabla f}$) y sus restricciones ($_{\nabla g}$) para toda la estructura están calculados en paralelo, donde filas individuales de ∇f y ∇g pueden ser evaluados en diferentes procesadores sin ninguna comunicación entre ellos, y después de que todos los procesadores esclavos hayan realizado el cálculo lo envían al procesador maestro el cual los ensamblara y terminara el proceso de optimización. En cuanto al paralelismo multinivel se basa en la estrategia anteriormente nombrada “divide y vencerás”, este divide el dominio del problema en subdominios, cada uno de estos subdominios es asignado a un procesador y para su solución se procede igual que la paralelización en un nivel, donde cada procesador esclavo transmite información al procesador maestro, estos muestra que la optimización secuencial será tomada por el procesador maestro después de recibir todos los gradientes de los subdominios.

Para efectos de esta propuesta P.K. Umesha realizo pruebas numéricas con armaduras espaciales planas, donde fue utilizado el software PARCAL (versión paralela del CAL). Y fue desarrollado en un modelo de arquitectura MPI. Los resultados arrojaron que el speed-Up por diferencias finitas era mayor que por la forma analítica.

En [51] se encuentra un ejemplo de esta aplicación lo más reciente posible (2010) donde J. París y F. Navarrina mejoran las restricciones de esfuerzo en la optimización estructural topológica. Aunque los principios básicos de la

optimización topológica se basaban en la máxima rigidez, desde hace algún tiempo se ha propuesto esta optimización basada en el peso mínimo con el esfuerzo necesario. Esto ha provocado que la programación matemática (métodos clásicos de optimización) se enfrente a problemas más complejos con ciertas componentes no lineales debido a que las restricciones (como esfuerzo y deformación) deben ser tomadas en cuenta.

Dichas restricciones de esfuerzo se proponen en varios enfoques; un enfoque local donde esta restricción será el máximo esfuerzo en el punto central de cada elemento de la malla (se utiliza una malla ya que el sistema se resuelve por medio de elementos finitos), y un enfoque global donde se ve reducido enormemente el esfuerzo de cálculo necesario, pero se ve afectado por la pérdida de información en el análisis de sensibilidad debido a la agregación de las limitaciones globales. De ahí el tercer enfoque, el de bloques, el donde cada bloque contiene aproximadamente el mismo número de elementos, la idea principal de este enfoque es imponer una restricción global sobre los elementos de cada bloque, esto crea el punto medio para reducir esfuerzo de cálculo sin la pérdida significativa de información.

Para la solución de este problema de optimización de topología se toma como una aproximación lineal, donde cada iteración se realiza mediante el método Simplex, ya que este algoritmo ha demostrado que tiene un buen desempeño cuando se trata en un enfoque global (con una sola restricción), no se tienen en cuenta más restricciones para poder ahorrar recursos computacionales. En este caso el análisis de sensibilidad requerida se calcula analíticamente. El sistema completo de las derivadas de primer orden pertenecientes a las restricciones se obtiene a través de un método adjunto variable con el fin de seguir reduciendo el uso de recursos computacionales. Por el contrario las derivadas de según orden de las restricciones se calculan analíticamente a través de una técnica de diferenciación directa.

En cuanto a la paralelización del algoritmo simplex (un procedimiento iterativo) no es posible del todo. Por otro lado la modificación de la matriz del problema en cada iteración interna consume casi todo el tiempo de cálculo y puede ponerse en paralelo con facilidad. El código paralelo fue desarrollado utilizando OpenMP en un código fuente Fortran.

3.2 PROGRAMACIÓN EN PARALELO EN METODOS HEURISTICOS

Los métodos heurísticos para efectos de este trabajo son todos aquellos que no presentan el cálculo de derivadas en procedimiento. Algunos de estos métodos se han paralelizado naturalmente convirtiéndose en métodos derivados, como por ejemplo el método de islas dentro de los algoritmos genéticos donde tiene una paralelización implícita.

Los algoritmos evolutivos en pocas palabras son aquellos que requieren una población de soluciones tentativas que representan los puntos del espacio de búsqueda, en donde se realizan ciertas operaciones para efectos de la búsqueda, lo que en recursos computacionales se traduce en memoria física y tiempo del procesador.

Para disminuir el tiempo de ejecución existen dos posibilidades, la primera de ella es reduciendo de alguna manera el número de cálculos o evaluaciones necesarias para llegar a la solución, y la segunda es ejecutar el algoritmo en un máquina paralela, dicho paralelismo en las operaciones de los algoritmos evolutivos da lugar a los algoritmos evolutivos paralelo (AEP) que permite utilizar poblaciones de mayor tamaño, lo que significa la resolución de problemas de mayor dimensión y complejidad, reduciendo como es el objetivo el tiempo de espera por la solución.

En [66] Georg Thierauf y JianboCai toman un método de la computación evolucionaria, y este es la estrategia evolucionaria para la resolución de problemas de optimización estructural, este es aplicado a situaciones en las

cuales no solo se cuenta con variables discretas o continuas sino un combinación de ambas, para resolver estos problemas la mayoría de métodos transforman estos problemas mixtos a una secuencia de problemas continuos que se resuelven iterativamente, y en cada iteración un problema de optimización continuo debe ser resuelto.

La principal modificación del método de las estrategias de evolución para resolver la optimización discreta y continua se basa en la aplicación de operadores de mutación diferentes. La adaptación de este en una técnica de computación en paralelo, nos muestra que es necesaria la aplicación de un paralelismo de dos niveles (sugerencia del artículo), ya que como se nombró anteriormente es necesario en primera instancia dividir el problema de optimización en dos subproblemas, para tratarlo como procesos diferentes, uno de ellos con variables de diseño discretas y el otro con variables de diseño continuas. Cada uno de estos subproblemas se resuelve simultáneamente utilizando computación en paralelo, (otro nivel de paralelismo).

En el segundo nivel cada subproblema es aún más paralelizables a través de la estrategia subevolucionaria paralela (PSES), la idea básica del PSES es dividir la población de gran tamaño en varias subpoblaciones pequeñas y en paralelo. Si se tienen un conjunto de procesadores p, el conjunto de la población del subproblema se dividirá en p subpoblaciones, cada procesador ejecutara la estrategia evolucionaria en su propia subpoblación. Periódicamente algunos de sus mejores individuos son seleccionados y copias de ellos son enviadas a los vecinos, y este también recibe copia de los vecinos, sustituyendo los elementos malos de su subpoblación.

En [67] B.H.V Topping resume los últimos desarrollos realizados a la programación en paralelo aplicado a las redes neuronales y los algoritmos genéticos.

Los algoritmos genéticos como se explicó anteriormente se basan en las leyes de evolución, este algoritmo ha sido paralelizado y han sobresalido tres modelos, los cuales se podrían considerar como la clasificación de los algoritmos genéticos paralelos: Global, Islas (Explicado en 1.3.2.1) y celular.

El algoritmo genético paralelo global cuenta con una sola población, (global) donde los compañeros son elegidos dentro de la misma población y la población evoluciona sin ningún tipo de influencia externa, por ende todos los individuos conviven en el mismo hábitat y están igualmente dispuesto para el apareamiento, el paralelismo se basa en que dicho apareamiento se puede realizar de una manera distribuida, sin embargo el control general debe permanecer en el un único procesador, los que significa que parte del algoritmos continua siendo secuencial.

Si el algoritmo corre de manera tradicional, en cada generación la ejecución se lleva a cabo en un único procesador hasta que la selección se realice, lo que muestra que el costo de la generación y selección de una gran población (realizada por único procesador) es muy alto. Por lo tanto, la paralelización es muy poco probable que sea eficiente al menos que el coste computacional de las evaluaciones sea muy alto.

En el algoritmo genético paralelo de Islas, como ya se explicó, la población se divide en islas las cuales separan los individuos, permitiéndoles el apareo solo con otros individuos de la misma isla, en este contexto un sistema en paralelo puede ser visto en un entorno global y complejos compuesto de regiones autónomas (islas), y cada procesador le corresponde una región en la cual realiza el respectivo algoritmo genético, por lo que el modelo de islas presenta un paralelismo de grano grueso, cuyo paralelismo viene de subconjuntos de procesamientos de una población al mismo tiempo en un número de procesadores.

El algoritmo genético paralelo celular, conformado por paralelización de grano fino, compuesto por una gran población subdividida en vecindarios que se superponen, en este caso se coloca un elemento en cada procesador y casa individuo tiene un vecindario en particular, donde la paralelización puede ser entendida como un cuerpo compuesto por células reproductoras, si dicho cuerpo está estructurado de tal forma que las células tienen muy limitada movilidad, y la reproducción solo se puede realizar con las células muy cercanas, y luego la cría es elegida para reemplazar a uno de sus padres, y lógicamente dos descendencias pueden reemplazar a ambos padres.

La aplicación de los algoritmos genéticos en la optimización estructuralha sido de gran recurrencia en los últimos años, sin embargo el alto costo computacional limita su aplicación a problemas en donde el espacio de diseño es pequeño, por ende los algoritmos genéticos son más eficaces cuando el espacio de diseño es grande, a pesar de todo esto el proceso de optimización es paralelizable en un alto grado.

En [68] B.H.V Topping, realiza una optimización de dimensionamiento directo utilizando un algoritmo genético paralelo, a un problema de un puente atirantado sin restricciones y con una longitud de cadena fija.

Otro de los métodos heurísticos utilizado en un enfoque paralelo es el recocido simulado (SA), en [69] J.P.B. Leite muestra como este método es aplicado de manera paralela a los problemas de optimización estructural.

El recocido simulado ha demostrado ser una buena técnica para resolver problemas de optimización combinatorios, aunque posee una desventaja la cual consiste en que sistemas muy complejos el proceso de convergencia puede tomar demasiado tiempo. De ahí múltiples intentos para mejorar el desempeño, son mediante la realización de cambios en el algoritmo o por el uso de hardware paralelo pero estos han dado lugar a problemas de conducta dependiente. LA

paralelización del recocido simulado no es una tarea trivial debido a la naturaleza secuencial del algoritmo.

La aplicación de este método a la optimización estructural ha sido considerable por ejemplo se ha utilizado para diseño de estructura de soportes, también se ha aplicado a la optimización discreta de armaduras en tres dimensiones.

Como se nombró anteriormente la desventaja más relevante es la demora en el tiempo de cómputo, por lo que el procesamiento en paralelo parece ser el único camino viable, sin embargo el rendimiento de esta aplicación puede ser dependiente, además que depende de las capacidades de la arquitectura del hardware a utilizar.

El SA en paralelo cuenta con la necesidad de identificar en forma estática o dinámica, el número de operaciones o cambios de tal manera que no se interactúe con la evaluación de la función, ya que en problemas de optimización estructural se gasta el mayor tiempo en las evaluaciones de las funciones.

En [70] Arciszewski, utiliza el método de algoritmo genético paralelo unificado (modelo de islas con migraciones entre ellas), para su respectiva aplicación a problemas de topológicos discretos y para problemas de optimización de dimensionamiento directo (reducción del peso), y el problema se plantea para estructuras de acero para grandes edificaciones.

En [71] K.C Sarma aplica de manera similar el método de islas con migraciones para la optimización del dimensionamiento directo del marco de las estructuras espaciales de varios pisos, donde es fija la longitud de los elementos.

En [72] C.K. Dimou realizo un trabajo similar al inmediatamente nombrado, una optimización del dimensionamiento directo, utilizando un algoritmo genético paralelo, y su aplicación se extiende de marcos de estructuras espaciales a sistemas armaduras planares.

En [73] T. Pullman planteo un método para problemas discretos de optimización topológica aplicando el modelo de islas y lógica difusa, aplicado a grandes edificaciones de concreto reforzado.

En [74] Travis S. Metcalfe realiza un modelamiento de una estructura de tipo laminar usando un algoritmos genético paralelo, para alcanzar un objetivo global, y diseñan un hardware completamente paralelo y distribuido, implementando un software de la sub rutina de optimización llamada Pikaia, la cual usa un algoritmo genético para proporcionar unos parámetros óptimos a nivel general. Este trabajo es aplicado especialmente a determinaciones y observaciones físicas, como por ejemplo el modelado a partir de la observación de sus frecuencias de oscilación de una enana blanca (estrella).

En [75] Kicinger R. plantea un método de islas distribuido para la optimización de problemas de tipo de topología discreta y dimensionamiento directo, y está enfocado a las estructuras de aceros en los grandes edificios.

Todos estos métodos de optimización no solo se aplican a problemas estructurales, sino que también se aplican a otras ramas de la ingeniería civil, como por ejemplo en la construcción como se observa en [76] donde el objetivo es darle robustez a los procedimiento de optimización en construcción.

En las tablas 1 y 2, muestran resumidamente algunas aplicaciones encontradas donde se especifica la clasificación matemática del método de optimización utilizado, además se encuentra la respectiva herramienta de computación en paralelo utilizada. En cuanto a su clasificación según aplicación se reduce a la especificación de tipo de problema resuelto ya sea de topología, de forma, de dimensionamiento directo, o en algunos casos se especifico a que tipo de de problemas deben ser aplicados.

Tabla 1 : Agrupación de ejemplos de aplicaciones de programación en paralelo en la optimización estructural

Referen.	Tipo de Optimización	Problema propuesto y/o aplicación	Método de Opt. Utilizado	Tipo de Paralelización	Software y/o Arquitectura
[64]	Clásica	Para estructuras ligeras con ejemplo de armaduras	Programación lineal (CONMIN)	Paralelización de Grano Fino	Versión paralelo de VICONOPT
[52]	Clásica: Topológica	Estructuras en 3D	Método del gradiente conjugado	descomposición de dominio	Código de escritura C++
[65]	Clásica	Estructuras en general, planteamiento teórico	Programación no lineal, programación cuadrática	Establece 3 niveles de acuerdo al método de optimización	No Especifica
[43]	Clásica: Dimensionamiento directo y forma	Grandes armaduras espaciales	Calculo de sensibilidad analíticamente y diferencias finitas	Único Nivel, y Multinivel (descomposición de dominio en este caso)	MPI con lenguaje CAL-PARCAL
[51]	Clásica: Topológica	Ejercicio con Vigas, pero extendido a cualquier estructura	Método Simplex	De 1 nivel aplicada al algoritmo simplex	OPEN MP con uso del FORTRAN
[67]	Heurística	Estructuras en general, ejemplo con presa	Algoritmos genéticos, y redes neuronales	De Grano Fino y grueso en los algoritmos genéticos	MPI con uso del FORTRAN, y lenguaje C, C++
[68]	Heurística: Dimensionamiento directo	Puentes atirantados con longitud de cables fijas	Algoritmos genéticos	Algoritmos Genéticos Paralelos	No Especifica
[70]	Heurística: Topología discreta y de forma continua	Estructura de Soporte de Edificios Altos	Algoritmos genéticos	Modelo de Islas	No Especifica

Tabla 2: Continuación Tabla 1

Referen.	Tipo de Optimización	Problema propuesto y/o aplicación	Método de Opt. Utilizado	Tipo de Paralelización	Software y/o Arquitectura
[71]	Heurística: Dimensionamiento directo	Marcos de Estructuras espaciales de varios pisos	Algoritmos genéticos	Modelo de Islas con migraciones	No Especifica
[72]	Heurística: Dimensionamiento directo	Armaduras Planares	Algoritmos genéticos	Algoritmos Genéticos Paralelos	No Especifica
[73]	Heurística: Topología discreta	Grandes Edificaciones de concreto reforzado	Algoritmos genéticos	Modelo de Islas, Lógica Difusa	No Especifica
[75]	Heurística: Topología discreta y dimensionamiento directo	Estructuras de acero en edificios altos	Algoritmos genéticos	Modelo de Islas Distribuido	No Especifica
[66]	Heurística	Armaduras, y ejemplo de diseño de recipiente a presión	Estrategias Evolucionarías	Estrategias subevolucionarias paralelas (PESES)	No Especifica
[69]	Heurística	Estructuras Generales	Recocido Simulado	Paralelización de Recocido Simulado, con memoria compartida y distribuida	Lenguaje de Programación en C

4. CONCLUSIONES

A través de toda la historia en el desarrollo de métodos de optimización estructural, siempre han sido un factor crucial las herramientas, ejemplo de ello es que en las dos últimas décadas del siglo pasado se puede observar y comparar que los método utilizados se basaban en cálculos que en problemas de pequeña escala estaban diseñados para ahorrar el mayor número de iteraciones utilizando para ello las derivadas. Con la aparición de los recursos computacionales, ya el problema se centra en la eficiencia a la hora de aplicar esos resultados, y ese se podría decir que es el porqué de la existencia de los métodos heurísticos.

La computación en paralelo no es nada extraña en nuestro medio aunque alguna veces ignoremos su presencia, la necesidad de estos recursos computacionales ha explotado este nuevo enfoque.

La tendencia apunta a una mayor inversión en los métodos heurísticos, y a su paralelización, pero hay que tener en cuenta que los métodos clásicos también están siendo desarrollados con estas técnicas; todo depende del tipo del problema con el que se esté tratando, debido a que es este el que determina cual es el mejor método de optimización estructural y a la vez determina cual es la mejor forma de paralelización del método escogido.

En el caso de los métodos clásicos, se ha enfocado en la paralelización de su fase de cálculo más dificultosa, el cálculo de sensibilidades con ciertos modelos de paralelización, lo que abre campo a futuras investigaciones en comparar con otros modelos de paralelización para así seguir optimizando este proceso de cálculo de las sensibilidades y por ende la optimización de los métodos clásicos.

El método de homogenización es un método clásico, en donde la aplicación de la computación en paralelo es casi nula, genera la pregunta de ¿qué tan efectivo

seria este procedimiento en paralelo? que se espera sea estudiada y respondida en tiempos futuros.

Como paralelizar un problema depende su naturaleza, como se nombró anteriormente, de hecho existen algunos problemas que pueden no ser paralelizados, para el caso de que la paralelización sea viable, se debe hacer de tal forma que la parte principal de computo sobre un procesador sea independiente de datos y/o cómputos sobre cualquier otro procesador.

El método heurístico y el método general más tratado de forma paralela es el de los algoritmos genéticos los cuales se han aplicado a una gran gama de problemas por su adaptabilidad, esto le ha sido permitido a este algoritmo ya que se diseñó con vertientes paralelizadas implícitas (modelo de islas) sin embargo, no se ha establecido un modelo de paralelización óptimo para este método heurístico, dejando para trabajos futuros la elección del mejor método de paralelización para este método.

Con respecto a los modelos de paralelización, la más usada es la arquitectura MPI en la paralelización, ya que su facilidad de acceso, y facilidad de modificación la hace muy atractiva para los autores. Y la segunda preferida es la OPENMP, cuyo principal atractivo es que es abierta los usuarios, y la más revolucionaria en investigaciones sistemáticas es la CUDA, y ya que no se encontró una aplicación directa con la optimización estructural abre campo a aplicar modelos de optimización estructural con este modelo de paralelización.

Debido a que se nota un incremento significativo [78], en el uso de herramientas de computación mediante la simulación basada en internet, sería interesante que la Escuela de Ingeniería Civil de la Universidad Industrial de Santander se comience a explorar el uso, por ejemplo, análisis de elementos finitos basados en internet.

5. BIBLIOGRAFIA

[1] FAUSTO GARCÍA MÁRQUEZ Y MANUEL LAGUNA. Optimización: Conceptos fundamentales y Tendencias Actuales. Universidad de Castilla-La Mancha, University of Colorado(1999) EN: www.terra.es/personal/faustopedro.gar/files

[2] ANDRÉS RAMOS Y PEDRO LINARES. Modelos matemáticos de optimización. Universidad Pontificia Comillas. España. Tesis Doctoral (Octubre de 2001).

[3] LAURA BATTAGLIA Y ALBERTO CARDONA. Aplicación de métodos de optimización de forma en el diseño estructural EN: Mecánica computacional Vol. XXI pp. 2804-2823. (Octubre de 2002).

[4] R.T. MARLER AND J.S. ARORA. Survey of multi-objective optimization methods for engineering. Review Article EN: Struct Multidisc Optim 26, 369–395 (2004)

[5].http://es.wikipedia.org/wiki/Eficiencia_de_Pareto. Acceso 28 /09/2010.

[6]RAFAL KICINGER, TOMASZ ARCISZEWSKI, KENNETH DE JONG. Evolutionary computation and structural design: A surveyof the state-of-the-art EN: Computers and Structures 83 (2005) 1943–1978

[7]NATYHELEM GIL LONDOÑO. Algoritmos genéticos. Universidad Nacional de Colombia. Escuela de Estadística. Sede Medellín (2006)

[8].ROSENBERG RS. Simulation of genetic populations with biochemical properties.,Phd Dissertation. University of Michigan, Ann Harbor, MI; (1967).

[9]. SYSWERDA G, PALMUCCI J. The application of genetic algorithms to resource scheduling.EN: Belew RK, Booker LB, editors. Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA91), San Diego, CA, USA; (1991). p. 502–8.

[10]. SCHAFFER JD. Multiple objective optimization with vector evaluated genetic algorithms. EN: Grefenstette JJ, editor. Proceedings of the First International Conference on Genetic Algorithms (ICGA85), Pittsburgh, PA, USA; (1985). p. 93–100.

[11] ELENA CHERKAEV, ANDREJ CHERKAEV. Minimax optimization problem of structural design EN: Computers and Structures 86 (2008) 1426–1435

[12] FONSECA CM, FLEMING PJ. Genetic algorithms for multiobjective optimization: formulation, discussion, and generalization. EN: Forrest S, editor. Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA93), Urbana-Champaign, IL, USA; (1993). p. 416-23

[13]. SRINIVAS N, DEB K. Multiobjective optimization using nondominated sorting in genetic algorithms. Evolution Computat (1994);2(3):221–48.

[14] GOLDBERG DE. Genetic algorithms in search, optimization, and machine learning. Reading, MA: Addison-Wesley; (1989).

[15].HORN J, NAFPLIOTIS N. Multiobjective optimization using the niched Pareto genetic algorithm (Technical Report No. IlliGAl Report 93005), University of Illinois at Urbana-Champaign Urbana, Illinois, USA; 1993.

[16] ANDERS FORSGREN, PHILIP E. GILL, MARGARET H. WRIGHT. Interior Methods for Nonlinear Optimization EN: Society for Industrial and Applied Mathematics (2002), Vol. 44, No. 4, pp. 525–597

[17] TOMÁS ARREDONDO VIDAL. Introducción a la optimización. (2009) EN: http://profesores.elo.utfsm.cl/~tarredondo/info/softcomp/Introduccion%20a%20la%20optimizacion.pdf

[18] MARIA FONTALVO. Optimización geometría y minimización Energética. Departamento Química Orgánica y Farmacéutica. Universidad de Navarra. (2004).En:www.unav.es/organica/docencia/modeling.../optimizacion/optimizacion.pdf

[19] DR. CARLOS A. COELLO COELLO. Método de las Estimaciones Cuadráticas Sucesivas EN: Av. IPN No. 2508Col. San Pedro Zacatenco México, D.F. 07300. (2005)

[20] KANG SEOK LEE Y ZONG WOO GEEM B. A new structural optimization method based on the harmony search algorithm EN: Computers and Structures 82 (2004) 781–798.

[21] TOMÁS ARREDONDO VIDAL. Optimización sin derivados. (2010) EN: http://profesores.elo.utfsm.cl/~tarredondo/info/softcomp/Optimizacion%20sin%20derivados.pdf

[22].GREFENSTETTE JJ, BAKER JE. How genetic algorithms work: a critical look at implicit parallelism. EN: Schaffer JD, editor. Proceedings of the Third

International Conference on Genetic Algorithms (ICGA89), Fairfax, VA, USA; (1989). p. 20–7.

[23].http://mikilab.doshisha.ac.jp/dia/research/report/2006/0806/006/report20060806006.html Acceso 29/09/2010.

[24] ZELDA B. ZABINSKY. Random Search Algorithms, Department of Industrial and Systems Engineering, University of Washington, Seattle, WA, 98195–2650.(Abril 2009)

[25] PEDRO YOAIM SALAZAR. Algoritmo hibrido auto configurado para optimización estructural. Trabajo de Grado. Universidad Industrial de Santander. 2009

[26] http://www.swarmintelligence.org/Acceso 29/09/2010

[27] C BRIAN P. GERKEY, SEBASTIAN THRUN. Parallel stochastic hillClimbing with small teams.Stanford University, Stanford, CA 94305, USA. (2003)

[28] O P. MOSCATO Y C. COTTA. Memetic Algorithms, Newcastle Bioinformatics Initiative, University of Newcastle, Callaghan, NSW, 2308, Australia. (2005)

[29] U ZHOU FENG, YONG HAN KIM y BO-SUK YANG.Applications of hybrid optimization techniques for model updatingof rotor shaftsEN: Struct Multidisc Optim (2006) 32: 65–75

[30] HASLINGER J, NEITTAANMAKI P. Finite element approximation for optimal shape design material and topology design. Chichester, UK: John Wiley & Sons; (1996).

[31] ALLAIRE G, BONNETIER E, FRANCFORT G, JOUVE F. Shape optimization by the homogenization method. EN: Nuemer Math (1997); 76:27–68.

[32] CERROLAZA M, ANNICCHIARICO W. Genetic algorithms in shape optimization: finite and boundary element applications. EN: Miettinen K, Makela MM, Neittaanmaki P, Periaux J,1976 R. Kicinger et al. / Computers and Structures 83 (2005) 1943–1978editors. Evolutionary algorithms in engineering and computer science.Chichester, England: John Wiley &Sons;(1999).

[33] http://es.wikipedia.org/wiki/B-spline.Acceso 01/10/2010

[34]. ANNICCHIARICO W, CERROLAZA M. Structural shape optimization 3D finite-element models based on genetic algorithms and geometric modeling. Finite Elem Anal Des (2001); 37(5):403–15.

[35]. WOON SY, QUERIN OM, STEVEN GP. Structural application of a shape optimization method based on a genetic algorithm. StructMultidisciplinaryOptim (2001); 22(1):57–64.]

[36] PROGRAMACIÓN LINEAL, Capítulo 8, http://sauce.pntic.mec.es/~jpeo0002/Archivos/PDF/T08.pdf. Acceso 29/09/2010.

[37] B. CHACHUAT. Nonlinear and Dynamic Optimization: From Theory to Practice. Chapter 1, (2007) Automatic Control Laboratory, EPFL, Switzerland.

[38] PROGRAMACIÓN DINÁMICA, http://www.edicionsupc.es/ftppublic/pdfmostra/OE03503M.pdf. Acceso 03/10/2010.

[39] BOHNENBERGER O, HESSER J, MANNER R. Automatic design of truss structures using evolutionary algorithms. EN: Proceedings of the Second IEEE International Conference on Evolutionary Computation (ICEC95), Perth, Australia; (1995). p. 143–9.

[40]O. HASANC EBI. Adaptive evolution strategies in structural optimization: Enhancing their computational performance with applications to large-scale structures EN: Computers and Structures 86 (2008) 119–132.

[41] KANG SEOK LEE Y ZONG WOO GEEM B. A new structural optimization method based on the harmony search algorithm EN: Computers and Structures 82 (2004) 781–798.

[42] YUHANG CHEN, SHIWEI ZHOU, QING LI. Multiobjective topology optimization for finite periodic structures EN: Computers and Structures 88 (2010) 806–81.

[43] P. K. UMESHA, M. T. VENURAJU, D. HARTMANN Y K. R. LEIMBACH. Parallel Computing Techniques for Sensitivity Analysis in Optimum Structural Design EN: Journal of Computing in Civil Engineering, Vol. 21, No.6, November 1, 2007.©ASCE, ISSN 0887-3801/2007/6-463–477.

[44] HAJELA P. Genetic algorithms in automated structural synthesis. EN: Topping BHV, editor. Optimization and artificial intelligence in civil and structural engineering, vol. 1. Dordrecht: Kluwer Academic Press; (1992).

[45] JARMAI K, SNYMAN JA, FARKAS J, GONDOS G. Optimal design of a welded I-section frame using four conceptually different optimization algorithms. EN:Struct Multidiscipl Optimiz (2003);25: 54–61.

[46] Optimización topológica. Capítulo 4. http://bibing.us.es/proyectos/abreproy/4296/fichero/VolumenI%252F4.pdf. Acceso 03/10/2010.

[47] XIE YM, STEVEN GP. Shape and layout optimization via an evolutionary procedure.EN: Proceedings of the International Conference on Computational Engineering Science, Hong Kong University of Science and Technology, Hong Kong; (1992).

[48] MARIANO VICTORIA NICOLÁS, Optimización de forma y topología con malla fija y algoritmos genéticos. Tesis Doctoral. Universidad Politécnica de Cartagena Departamento de Estructuras y Construcción. (2006)

[49] QUERIN OM. Evolutionary structural optimization: Stress based formulation and implementation. Ph.D. thesis. Australia: Dept. of Aeronautical Engineering, The University of Sydney; (1997).

[50] XIA LIUA, WEI-JIAN YI, Q.S. LI, PU-SHENG SHEN. Genetic evolutionary structural optimization EN: Journal of Constructional Steel Research 64 (2008) 305–311.

[51] J. PARÍS, F. NAVARRINA, I. COLOMINAS, M. CASTELEIRO. Improvements in the treatment of stress constraints in structural topology optimization problems EN: Journal of Computational and Applied Mathematics 234 (2010) 2231-2238.

[52] Thomas Borrvall y Joakim Petersson.Large-Scale topology optimization in 3D using parallel computing EN: Comput. Methods Appl. Mech. Engrg. 190 (2001) 6201-6229.

[53] BRAMLETTE MF, BOUCHARD EE. Genetic algorithms in parametric design of aircraft.EN: Davis L, editor. Handbook of genetic algorithms. New York: Van Nostrand Reinhold; (1991). p. 109–23.

[54] SHYUE-JIAN WU y PEI-TSE CHOW. Integrated discrete and configuration optimization of trusses using genetic algorithms. EN: Computers and Structures Vol. 55. No. 4. pp. 695-702,(1995).

[55] OMER KELESOGLU. Fuzzy multiobjective optimization of truss-structures using genetic algorithm.EN: Advances in Engineering Software 38 (2007) 717–721.

[56] VEDAT TOˇGAN, AYSE T. DALOGLU**.** Optimization of 3d trusses with adaptive approach in genetic algorithms**.**EN: Engineering Structures 28 (2006) 1019–1027.

[57] RAFAEL ARTURO TRUJILLO RASUA. Algoritmos paralelos para la solución de problemas de optimización discretos aplicados a la decodificación de señales. Tesis Doctoral. Departamento de Sistemas Informáticos y Computación Universidad Politécnica de Valencia. (Marzo 2009).

[58] HENRY KASIM, VERDI MARCH, RITA ZHANG y SIMON SEE. Survey on Parallel Programming Model.EN: J. Cao et al. NPC (2008) pp. 266 – 275.

[59] http://openmp.org/wp/. Acceso 05/10/2010.

[60]http://www.nvidia.com/object/cuda_home_new.htmlAcceso 05/10/2010.

[61]http://www.mcs.anl.gov/research/projects/mpi/Acceso 04/10/2010.

[62] E. D. SOTELINO. Parallel Processing Techniques in Structural Engineering Applications EN: Journal of Structural Engineering, Vol. 129, No. 12, (December 1, 2003). ©ASCE, ISSN 0733-9445/(2003)/12-1698–1706.

[63] G.F. CAREY, E. BARRAGY, R. MCLAY y M. SHARMA. Element-By-Element vector and parallel computations. EN: Communications in applied numerical methods. Vol 4, 299-307 (1988).

[64] K.L. CHAN, D. KENNEDY, F.W. WILLIAMS. Parallel processing, neural networks and genetic algorithms EN: Advances in Engineering Software 31 (2000) 819-825.

[65] A. MIGDALAS, G. TORALDO B, V. KUMAR. Nonlinear optimization and parallel computing EN: Parallel Computing 29 (2003) 375–39.

[66] GEORG THIERAUF AND JIANBO CAI. Parallel evolution strategy for solving structural optimization EN: Engineering Structures Vol. 19. N° 4 pp. 318 - 324 (1997).

[67] B.H.V. TOPPING, J. SZIVERI, A. BAHREINEJAD, J.P.B. LEITE & B. CHENG. Parallel processing, neural networks and genetic algorithms EN: Advances in Engineering Software Vol. 29, No. 10, pp. 763–786, (1998).

[68] TOPPING BHV, DE BARROS LEITE JP. Parallel genetic models for structural optimization. Eng Optimiz (1998); 31(1): 65–99.

[69] J.P.B. LEITE, B.H.V. TOPPING. Parallel simulated annealing for structural optimization. EN: Computers and Structures 73 (1999) 545-564.

[70] ARCISZEWSKI T, DE JONG KA. Evolutionary computation in civil engineering: research frontiers. EN: Topping BHV, editor. Civil and Structural Computing 2001. Stiring, Scotland: Saxe-Coburg Publications; (2001).p.2001.

[71] SARMA KC, ADELI H. Bilevel parallel genetic algorithms for optimization of large steel structures. Comput-Aided Civil Infrastruct Eng(2001); 16:295–304.

[72] DIMOU CK, KOUMOUSIS VK. Genetic algorithms in competitive environments.J Comput Civil Eng (2003); 17(3):142–9.

[73] PULLMANN T, SKOLICKI Z, FREISCHLAD M, ARCISZEWSKI T, DE JONG KA, SCHNELLENBACH-HELD M. Structural design of reinforced concrete tall buildings: evolutionary computation approach using fuzzy sets. EN:Ciftcioglu O, Dado E, editors. Proceedings of the 10th International Workshop of the European Group for Intelligent Computing in Engineering (EG-ICE), Delft, The Netherlands; (2003). p. 53–61.

[74] TRAVIS S. METCALFE, PAUL CHARBONNEAU. Stellar structure modeling using a parallel genetic algorithm for objective global optimization EN: Journal of Computational Physics 185 (2003) 176–193.

[75] KICINGER R, ARCISZEWSKI T, DE JONG KA. Distributed evolutionary design: island-model based optimization of steel skeleton structures in tall buildings. EN: Beucke K, Firmenich B, Donath D, Fruchter R, Roddis K, editors. Proceedings of the 10th International Conference on Computing in Civil and Building Engineering (ICCCBEX), Weimar, Germany; (2004).p. 190.

[76] AMR KANDIL, KHALED EL-RAYES, Y OMAR EL-ANWAR. Optimization Research: Enhancing the Robustness of Large-Scale Multiobjective Optimization inConstructionEN: Journal of Construction Engineering and Management, Vol. 136,No. 1, (January 1, 2010).©ASCE, ISSN 0733-9364/2010/1-17–25.

[77]HUNG- MING CHEN y YU-CHIN LIN. Web-FEM: An internet-based finite-element analysis framework with 3D graphics and parallel computing environment EN: Advances in Engineering Software 39 (2008) 55-68.

[78]JAMES BYRNE, CATHAL HEAVEY y P.J. BYRNE. A review of Web-based simulation and supporting tools.EN: Simulation Modelling Practice and Theory 18 (2010) 253-276.

BIBLIOGRAFIA ADICIONAL

Dicha bibliografía no se nombra explícitamente en el texto, pero ha influenciado la escritura de este. Por lo que es recomendable tenerla en cuenta.

[79] H. SCHMIDT Y G. THIERAUF. A combined heuristic optimization technique EN: Advances in Engineering Software 36 (2005) 11–19

[80] B. HASSANI Y E. HINTON. A review of homogenization and topology optimization III topology optimization using optimality criteria EN: Computers and Structures 69 (1998) 739-756

[81] PRUETTHA NANAKORN Y KONLAKARN MEESOMKLIN. An adaptive penalty function in genetic algorithms for structural design optimization EN: Computers and Structures 79 (2001) 2527–2539

[82] C. JIANG, X. HAN, F.J. GUAN, Y.H. LI. An uncertain structural optimization method based on nonlinear interval number programming and interval analysis method EN: Engineering Structures 29 (2007) 3168–3177

[83] HECTOR A. JENSEN, MICHAEL BEER. Discrete–continuous variable structural optimization of systems under stochastic loading EN: Structural Safety 32 (2010) 293–304

[84] QING LI, GRANT P. STEVEN , Y.M. XIE. Evolutionary structural optimization for stress minimization problems by discrete thickness design EN: Computers and Structures 78 (2000) 769-780

[85] ENRIQUE ALBA TORRES. Análisis y Diseño de Algoritmos Genéticos Paralelos Distribuidos. Tesis Doctoral. UNIVERSIDAD DE MÁLAGA. (Febrero de 1999).
AGO CERISOLA. Optimización Estocástica. Universidad Pontificia Comillas. España.(Septiembre de 2010).

[86] DR. CARLOS A. COELLO COELLO. Introducción a la Optimización Multiobjetivo Usando Metaheurísticas. CINVESTAV-IPN. México, D.F. 07300. (Julio 2005).

[87]http://www.tdr.cesca.es/TESIS_UPC/AVAILABLE/TDX-0731108-091907//09FLnm09de13.pdf

[88] R. H. GALLAGHER. Diseño estructural óptimo - una reseña EN: Revista internacional de métodos numéricos para cálculo y diseño en ingeniería, Vol. 1, 1,3-20 (1985)

[89] RAJKUMAR ROY, SRICHAND HINDUJA, ROBERTO TETI. Recent advances in engineering design optimisation: Challenges and future trends EN: CIRP Annals - Manufacturing Technology 57 (2008) 697–715.

[90] S. ŠILIH, S. KRAVANJA, M. PREMROV. Shape and discrete sizing optimization of timber trusses by considering of joint flexibility.EN: Advances in Engineering Software 41 (2010) 286–294.

[91] PRUETTHA NANAKORN Y KONLAKARN MEESOMKLIN. An adaptive penalty function in genetic algorithms for structural design optimization EN: Computers and Structures 79 (2001) 2527–2539.

[92] KAI-UWE BLETZINGER, EKKEHARD RAMM. Structural optimization and form finding of light weight structures EN: Computers and Structures 79 (2001) 2053-2062.

[93] NIKOLAOS D. LAGAROS, MANOLIS PAPADRAKAKIS, GEORGE KOKOSSALAKIS. Structural optimization using evolutionary algorithms EN: Computers and Structures 80 (2002) 571–589.

[94] J.M. AGUINAGALDE Y M. NO. Un nuevo método para la optimización de dimensiones de estructuras modelizadas por elementos finitos EN:Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería. Vol.4, 3, 275-295(1988).

[95] ANTONY JAMESON. Gradient Based Optimization Methods EN: Department of Aeronautics and Astronautics Stanford University, Stanford, CA 94305-4035. (1996).

[96] MAGNUS ERIK HVASS PEDERSEN. Numeric & Heuristic Optimization Source-Code Library for C#, The Manual, Revision 2.0, Copyright © (2009-2010), all rights reserved by the author.

[97] GADE PANDU RANGAIAH. Multi-Objective Optimization- Techniques and Applications in Chemical Engineering; Chapter 1: World Scientific Publishing Co. Pte. Ltda. (2003)

[98] DEPARTAMENTO DE MATEMÁTICASCSI/ITESM, Optimización Con Restricciones de Igualdad, (11 de noviembre de 2009).