COMPUTACIÓN EVOLUTIVA APLICADA AL DISEÑO DE ESTRUCTURAS RETICULARES

LEONARDO MORENO DE LUCA

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS

ESCUELA DE INGENIERÍA CIVIL

BUCARAMANGA

2013

COMPUTACIÓN EVOLUTIVA APLICADA AL DISEÑO DE ESTRUCTURAS RETICULARES

LEONARDO MORENO DE LUCA

Trabajo de Grado de Investigación para optar al título de

Magíster en Ingeniería Civil

Director:

OSCAR J. BEGAMBRE CARRILLO

Ingeniero Civil, M.Sc., Ph.D.

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS

ESCUELA DE INGENIERÍA CIVIL

BUCARAMANGA

2013

CONTENIDO

LISTA DE FIGURAS

LISTA DE ANEXOS

RESUMEN

TÍTULO: Computación Evolutiva Aplicada al Diseño de Estructuras Reticulares*

AUTOR: Leonardo Moreno De Luca**

PALABRAS CLAVES: optimización, estructuras reticulares, computación heurística, morfogénesis.

CONTENIDO:

Durante el proceso de optimización conducido en la fase conceptual del diseño de edificaciones, es común que surjan inconvenientes con respecto a los parámetros que definen el comportamiento estructural, y la funcionalidad y estética del objeto arquitectónico (ya que, por lo general, la optimización de dichos parámetros es conflictiva o incierta). Por consiguiente, en este proyecto de investigación se propone y se desarrolla un Procedimiento Auto-Configurado para el Diseño Multi-Objetivo de Estructuras Reticulares (PACDMOER), el cual está enfocado en generar y optimizar la geometría de dichas estructuras, teniendo como meta minimizar la energía de deformación y el peso, y optimizar el rendimiento acústico de la estructura. Para lograrlo, el procedimiento propuesto integra tres componentes (también desarrollados dentro del proyecto de investigación): un proceso de generación automática de geometría (llevado a cabo con el software Rhinoceros), unos códigos de programación para el análisis estructural y acústico (escritos en MatLab), y un algoritmo heurístico auto-configurado y multi-objetivo (basado en la combinación del algoritmo *Particle Swarm Optimization*, PSO, con el algoritmo *Unified Particle Swarm Optimization*, UPSO). Como resultado, el procedimiento propuesto fue capaz de alcanzar los objetivos, optimizando significativamente las tres funciones objetivo planteadas, y conduciendo un proceso de morfogénesis que resulta en la forma que se aproxima a ser óptima para una estructura reticular en particular. En este sentido, PACDMOER le permite al diseñador explorar distintas alternativas estéticas, mientras optimiza simultáneamente la estructura reticular.

---

*Trabajo de Grado de Investigación.

**Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería Civil. Director: Oscar J. Begambre Carrillo.

ABSTRACT

TITLE: Evolutionary Computation Applied to the design of Grid Structures*

AUTHOR: Leonardo Moreno De Luca**

KEY WORDS: optimization, grid structures, heuristic computation, morphogenesis.

CONTENT:

During the optimization process conducted in the conceptual design phase of buildings, some challenges appear regarding the parameters that define the structural behavior, and the functionality and aesthetics of an architectural object (generally, this is because the optimization of such parameters is conflictive or has a high degree of uncertainty). Therefore, in this research project we propose and develop an Auto-Configured Procedure for the Multi-Objective Design of Grid Structures (ACPMODGS). This approach is focused on the geometry generation and optimization of such structures, having as goal the minimization of the strain energy and weight, and the acoustical performance optimization of the grid structure. For doing so, the proposed procedure integrates three components (that were also developed within the research project): an automatic geometry generation process (conducted with the software Rhinoceros), some programming codes for the structural and acoustical analysis (written in MatLab), and an auto-configured multi-objective heuristic algorithm (based on the combination of the Particle Swarm Optimization algorithm, PSO, with the Unified Particle Swarm Optimization algorithm, UPSO). As result, the proposed procedure was able to reach the objectives, by optimizing the three stated objective functions in a significant way, while conducting a morphogenetic process that ends with an approximation to the optimal form, regarding a specific grid structure. In this sense, ACPMODGS allows the designer to explore different aesthetical alternatives, while simultaneously optimizes the grid structure.

---

*Research Graduation Project.

**Physical-Mechanical Engineering Faculty. Civil Engineering School. Director: Oscar J. Begambre Carrillo.

INTRODUCCIÓN

El paradigma actual del diseño arquitectónico, incluyendo todos los diseños ingenieriles y técnicos que lo componen (por ejemplo, el diseño estructural, acústico, bioclimático, hidrosanitario, etc.), se presenta como una fragmentación de diseños especializados que al sumarse conforman la totalidad del objeto arquitectónico. En este sentido, durante el proceso de diseño de una edificación, es muy común que se presenten inconvenientes, e incluso incompatibilidades, cuando se cruzan y se “integran” dichos diseños especializados. Como resultado, se limita y se restringe el rendimiento de cada componente, mientras que el proceso de diseño se convierte en un ciclo de prueba y error que termina cuando el equipo de diseño es capaz de ajustar los diseños especializados, en vez de terminar al encontrar una solución integral de alto rendimiento.

Por consiguiente, es necesario plantear soluciones que desde la fase conceptual del diseño tengan en cuenta la mayoría de los componentes que van a ser parte del diseño arquitectónico. Lo anterior se puede lograr sí, desde el inicio del planteamiento de diseño, la exploración formal y funcional (correspondiente principalmente a la arquitectura), se integra con evaluaciones de rendimiento técnico de las alternativas posibles (correspondiente principalmente a la ingeniería). Por ejemplo, sería posible pensar en desarrollar la interesante exploración plástica que lleva a cabo el arquitecto Frank Gehry en la mayoría de sus proyectos (ver la Figura 1, la Figura 2 y la Figura 3), en simultánea no sólo con consideraciones técnicas, sino procurando que éstas sean las más idóneas.

Figura 1. Maqueta del Biomuseo en Panamá, diseñado por el arquitecto Frank Gehry

Fuente: Oscar J. Begambre Carrillo

Figura 2. Vista exterior del Biomuseo en Panamá, diseñado por el arquitecto Frank Gehry

Fuente: Oscar J. Begambre Carrillo

Figura 3. Vista interior de la estructura de cubierta del Biomuseo en Panamá, diseñado por el arquitecto Frank Gehry

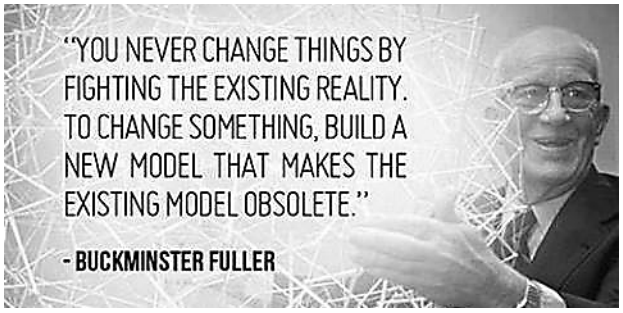
Fuente: Oscar J. Begambre Carrillo

Por los motivos anteriores, en los últimos años ha venido emergiendo una tendencia en la metodología de diseño, que pretende dejar atrás el paradigma de diseño actual descrito anteriormente. Para lograrlo, el nuevo paradigma planteado se fundamenta en la integración de procesos de formación, generación, evaluación y rendimiento, dentro de un medio digital de diseño (todo esto se vuelve posible gracias a los avances tecnológicos, especialmente a los correspondientes al campo de la computación). La idea es que esta integración se lleve a cabo desde la fase conceptual de diseño, con el fin de tener en cuenta, desde el inicio, todas las funciones que debe cumplir el objeto a diseñar, incluyendo las metas de rendimiento a las cuales se aspira llegar. De esta manera, la información obtenida de la simulación del rendimiento para cada una de las funciones, puede llegar a conducir procesos generativos y/o de formación, los

cuales definirán el resultado formal del objeto. Es claro que el diseñador debe definir los criterios de rendimiento y el proceso generativo, y debe interactuar directamente con la representación digital del objeto. Como consecuencia de dichas acciones, se presenta un cambio en el rol del diseñador, pasando de ser un usuario de herramientas y aplicaciones de diseño, a ser un generador o creador de las mismas[1].

De esta manera, el nuevo paradigma de diseño emergente le apunta a soluciones integrales de alto rendimiento (correspondiente a todas las funciones que el diseñador desee o necesite tener en cuenta, no solo a un par de ellas, como se hace actualmente), en donde ya no se debe pensar en componentes independientes, por ejemplo pensar por separado el componente formal del objeto arquitectónico y el sistema estructural, sino en sub-sistemas interdependientes e interconectados que resuelven, en conjunto, todas las necesidades de un proyecto de arquitectura.

Figura 4. “Usted nunca cambia las cosas peleando con la realidad existente. Para cambiar algo, construya un nuevo modelo que haga obsoleto al modelo existente”, Buckminster Fuller



Fuente: http://www.tumblr.com/tagged/innovation

[1] OXMAN, R.Theory and design in the first digital age. En: Design Studies. No. 27 (2006); p. 229-266.

Un acercamiento desde la Arquitectura y la Ingeniería Estructural hacia este nuevo paradigma de diseño, es la aplicación de modelos evolutivos de diseño[2], los cuales se enfocan en la optimización de los distintos parámetros que definen el comportamiento de la estructura y la funcionalidad y estética del objeto arquitectónico. En este proceso de optimización se presentan inconvenientes debido a que, generalmente, dichos parámetros son conflictivos o, la relación entre ellos presenta un comportamiento complejo, desde el punto de vista analítico, durante un proceso de optimización.

Por ejemplo, es de esperarse que la búsqueda de deformaciones mínimas entre en conflicto con la búsqueda de un peso mínimo de la estructura. Por otro lado, existe también incertidumbre en cuanto a la optimización simultánea de otros parámetros estructurales. En este caso, se puede citar, a manera de ejemplo, la búsqueda simultánea de una distribución de esfuerzos homogéneos y de un rendimiento deseado en una función arquitectónica específica como la bioclimática, la acústica o la optimización de espacios, entre otras; no es posible conocer de antemano si la optimización de estas funciones es conflictiva o, por lo contrario, van de la mano.

En resumen, la pregunta que se aborda en esta investigación, enfocada a la aproximación hacia ese nuevo paradigma de diseño, puede ser formulada de la siguiente manera: ¿Es posible desarrollar una aplicación de diseño, basada en la computación evolutiva y en la integración de procesos de formación, generación, evaluación y rendimiento, que permita llegar a obtener una edificación óptima desde el punto de vista arquitectónico y estructural?

[2] OXMAN, R.Theory and design in the first digital age. En: Design Studies. No. 27 (2006); p. 229-266.

El acercamiento propuesto para integrar lo que se expuso anteriormente, es la utilización de técnicas evolutivas de optimización multi-objetiva enfocadas al diseño integral (que satisfaga requerimientos estructurales y arquitectónicos) de una cubierta con estructura reticular[3], geometría aleatoria de apoyos[4] y superficie de forma libre[5]. Un ejemplo de este tipo de estructuras es el que se muestra en la Figura 5.

[3]Estructura reticular hace referencia a una tipología de estructura espacial compuesta por elementos lineales (los cuales están solicitados únicamente por fuerzas axiales) que forman celdas. Dependiendo de la modulación y dimensión de dichas celdas, la estructura reticular puede acercarse a formas descritas por medio de superficies curvas. Una característica de este tipo de estructuras es que las solicitaciones se aplican únicamente en los nodos, los cuales, para efectos de cálculo, se consideran como una articulación.

[4]La geometría aleatoria de apoyos hace referencia a la posible variación en la configuración de la planta del edificio, lo cual conlleva a una variación en la posición de los apoyos, bien sean columnas o muros. La idea es que la optimización se pueda llevar a cabo con cualquier tipología de planta arquitectónica. Claro está que una vez escogida (la planta arquitectónica correspondiente a una determinada edificación que se desea cubrir con la cubierta diseñada por medio de la metodología propuesta en el presente trabajo), ésta permanecerá constante durante el proceso de optimización, luego no será una variable en dicho proceso.

[5]La superficie de forma libre se fundamenta en una matriz rectangular de puntos de control y en un modelo matemático que define la manera como se creará la superficie. Por medio de dicho modelo se puede realizar la siguiente clasificación:

- Superficies de forma libre por interpolación: la superficie pasa a través de los puntos de control.
- Superficies de forma libre por aproximación: la superficie no tiene que pasar a través de los puntos de control. Se utilizan comúnmente las superficies no-racionales de Bézier y las superficies B-Spline no-racionales no-uniformes (61).

Este último tipo de superficie es el que se utilizará en el presente trabajo debido a que los software de modelado tri-dimensional disponibles se basan en él. Adicionalmente, no se utilizará una aproximación poligonal ya que el grado del polinomio necesario para ajustar una superficie curva e irregular sería bastante alto, provocando un mayor esfuerzo computacional o una descripción imprecisa de la superficie, lo que resultaría en un error en el proceso de optimización ya que las variaciones se hacen sobre los puntos de control y, si estos no son los reales, no se lograría alcanzar una forma óptima verdadera.

Figura 5. Estructura reticular de cubierta del Mannheim Multihalle – Frei Otto

Fuente:http://www.fotos.sc/PHPSESSID=21843797bf544442195a2d8bc44a330f/popi+1080906/mediafile.html

Lo que se pretende optimizar en dicho proceso de diseño integrado es la energía de deformación (buscando que sea mínima), el peso (buscando que sea mínimo) y el desempeño acústico de la cubierta (buscando una distribución uniforme en el nivel de sonido sobre el área bajo la cubierta). Dicho proceso de optimización va a generar una exploración estética durante un proceso de morfogénesis[6], brindándole al diseñador la posibilidad de escoger cierto valor formal dentro de las soluciones idóneas para un determinado problema. Lo anterior se llevará a cabo con la interacción entre el diseñador y el proceso de optimización. De esta manera es posible controlar, bajo ciertos criterios estéticos, el resultado formal de la

[6]Debido a que las variables del proceso de optimización serán los puntos de control que definen la forma de la cubierta, la superficie se generará a partir del desarrollo de dicho proceso. Esta generación de forma es llamada morfogénesis y su desarrollo implica una exploración estética (varias posibilidades formales que satisfacen los objetivos del proceso de optimización).

cubierta (es claro que esta parte del proceso de diseño es subjetiva y dependerá de los criterios propios del diseñador).

El proceso que se pretende llevar a cabo se fundamenta en la búsqueda de una determinada geometría, la cual influye directamente en el rendimiento estructural (determinará los desplazamientos de los nodos y el peso de la estructura) y acústico (influirá significativamente en la distribución del nivel de sonido sobre el área bajo la cubierta) de la cubierta a diseñar, así como en la forma definitiva de la cubierta.

Lo anterior se basa en que, adicional a la búsqueda de un determinado valor estético (subjetivo), el rol clave de la geometría arquitectónica (entendida como la definición de la forma de un proyecto y sus componentes principales) es satisfacer al mismo tiempo diferentes requerimientos técnicos que surgen de la interdisciplinariedad y de la complejidad que, por lo general, se presenta en los diseños arquitectónicos[7]. Como argumento adicional al enfoque que se propone, es interesante exponer el planteamiento de la geometría que presenta la arquitectura performativa[8], la cual sintetiza un rango amplio de evaluaciones de rendimiento e integra aspectos ingenieriles desde las fases tempranas del diseño,

[7] TURRIN, M., VON BUELOW, P. y STOUFFS, R. Design explorations of performance driven geometry in architectural design using parametric modeling and genetic algorithms. En: Advanced Engineering Informatics. No. 25 (2011); p. 656-675.

[8]"Arquitectura Performativa: La arquitectura no es algo sólido, estático, es algo vivo, que actúa, performa, respira. Como la naturaleza que también performa: la biosíntesis, la fotosíntesis. Mientras la arquitectura de Gaudí miraba la naturaleza por su geometría, por su forma, la arquitectura performativa la mira por su química, su física y su ciencia interna: los árboles recogen agua, producen oxígeno, queman CO2, son fábricas de alimentos, de plantas, de medicamentos. Si queremos que la arquitectura sea naturaleza, debe ser performativa". – Arquitecto Enric Ruiz Geli

siendo esenciales estos elementos y estas ideas para el proyecto que se propone en el presente documento.

Por otro lado, la elección de la computación evolutiva para llevar a cabo el presente proyecto se fundamenta en lo propuesto por Pugnale y Sassone[9]. Ellos plantean que el acercamiento evolutivo hacia la arquitectura es una estrategia poderosa, capaz de manejar problemas complejos, involucrando el ajuste de distintos requerimientos, desde aspectos económicos y estructurales hasta los correspondientes a lo constructivo, funcional y estético. Desde este punto de vista, el acercamiento multidisciplinario tradicional, en el que distintos profesionales están involucrados resolviendo cada uno un problema de diseño en específico (como generalmente funciona el paradigma de diseño actual, expuesto en el inicio de esta sección), se pone en discusión debido a los desarrollos recientes de la tecnología computacional. Por este motivo, muchos arquitectos sienten que su rol en el equipo de diseño está cambiando, como lo demuestra el interés global por la arquitectura no-estándar[10] y el correspondiente a la interacción entre arquitectura, ingeniería, matemáticas y computación.

Luego, lo que se pretende con el presente proyecto de investigación es aportar a la iniciativa y al desarrollo existente encaminado en la integración de las múltiples disciplinas involucradas en el diseño de un objeto arquitectónico, necesitando para esto el uso de las técnicas de computación evolutiva que permitan realizar una

[9] PUGNALE, A. y SASSONE, M. Morphogenesis and structural optimization of shell structures with the aid of a Genetic Algorithm. En: Journal of the International Association for Shell and Spatial Structures. No. 48 (2007); p. 161-166.

[10] CARPO, M. The Demise of the Identical Architectural Standarization in the Age of Digital Reproducibility. En: FIRST INTERNATIONAL CONFERENCE ON THE HISTORIES OF MEDIA ART, SCIENCE AND TECHNOLOGY. Banff New Media Institute, 2005.

exploración de alternativas de diseño[11]. Lo anterior corresponde con lo planteado por Woodbury y Burrow[12], los cuales argumentan que existen dos beneficios principales relacionados con dicha exploración de alternativas de diseño: la revelación y la comparación. Por un lado, las alternativas revelan cosas que no han sido consideradas, por ende sugieren futuras direcciones de exploración, haciendo que nuevas partes del espacio solución de diseño sean accesibles para investigaciones posteriores. Por otro lado, la comparación juega un rol clave en el entendimiento de cuándo un diseño satisface ciertos criterios y cuándo es el mejor entre los que han sido considerados, en vez de simplemente asegurar que satisface dichos criterios.

Es claro que, por el momento, se están teniendo en cuenta solo ciertos objetivos de diseño estructural y arquitectónico, pero se espera que este sea el inicio de una metodología de diseño que irá aumentando en complejidad a través del conocimiento que se genera por medio de investigaciones como la que se plantea en este documento. Adicionalmente, como lo plantea Turrin et al.[13], integrar conocimiento interdisciplinario y evaluaciones numéricas de rendimiento en las fases tempranas del diseño, con el fin de alcanzar direcciones que guíen el proceso, así como para favorecer descubrimientos de diseño, es identificado como una dirección clave para la inversión de esfuerzos en investigación.

[11]GASPAR-CUNHA, A. Modeling and Optimization of Single Screw Extrusion. Minho, 2000. Tesis doctoral. University of Minho.

[12] WOODBURY, L. y BURROW, A.L. Whither design space?. En: Artificial Intelligence for Engineering Design, Analysis and Manufacturing. No. 20 (2006); p. 63-82.

[13] TURRIN, M., VON BUELOW, P. y STOUFFS, R. Design explorations of performance driven geometry in architectural design using parametric modeling and genetic algorithms. En: Advanced Engineering Informatics. No. 25 (2011); p. 656-675.

1. COMPUTACIÓN HEURÍSTICA MULTI-OBJETIVA APLICADA AL DISEÑO ESTRUCTURAL Y ARQUITECTÓNICO

Por lo general, la intención de quien diseña una edificación es desarrollar un objeto arquitectónico creativo, innovador y óptimo. Durante este proceso, es frecuente que surjan conflictos entre los objetivos que pretende alcanzar el diseño estructural y los correspondientes que se desean lograr con el diseño arquitectónico. Como consecuencia, un diseño tiende a limitar el desarrollo óptimo del otro, y viceversa. Una de las opciones para resolver este tipo de problemas, caracterizados por interacciones complejas entre parámetros de diseño, es la aplicación de técnicas heurísticas de optimización multi-objetiva, las cuales son, en términos generales, una analogía matemática del proceso evolutivo darwiniano, de un comportamiento social, o de un proceso físico, representado por medio de un algoritmo.

Aparte del rendimiento exitoso en la resolución de múltiples objetivos numéricos, estas técnicas le permiten al diseñador explorar y generar soluciones creativas[14][15][16] e innovadoras[17]; de esta manera, la intención inicial de diseño pasa a ser una realidad. Algunas definiciones y conceptos acerca de optimización, innovación, y creatividad, así como de su integración, pueden encontrarse en un artículo científico de Koza et al.[18] y de Chen et al.[19].

[14] GERO, J.S. Computers and creative design. En: The global design studio. No. 1 (1996); p. 11-19.
[15] BODEN, M.A. The creative mind: myths and mechanisms. New York: Basic Books, 1992.
[16] ROSENMAN, M. The generation of form using evolutionary approach. En: DASGUPTA, D. y MICHALEWICZ, Z. Evolutionary algorithms in engineering applications. New York: Springer, 1997. p. 69-86.
[17] ALTSHULLER, G. The innovation algorithm: TRIZ, systematic innovation and technical creativity. Technical Innovation Center, 1999.
[18] KOZA, J.R., BENNETT III, F.H., ANDRE, D. y KEANE, M.A. Genetic Programming: Biologically Inspired Computation That Exhibits Creativity in Producing Human-Competitive Results. En: BENTLEY, P.J. y CORNE, D.W. Creative Evolutionary Systems. Londres: Academic Press, 2002. p. 275-298.
[19] CHEN, Y., LIU, Z.L. y XIE, Y.B. A knowledge-based framework for creative conceptual design of multi-disciplinary systems. En: Computer-Aided Design. No. 44 (2012); p. 146-153.

Los planteamientos presentados anteriormente, junto con la evolución de la ciencia computacional, son algunos de los motivos por los cuales la Computación Heurística (CH) se ha vuelto un paradigma atractivo para los diseñadores. Este fenómeno es parte de la revolución tecnológica y de información que ha venido en desarrollo en las últimas décadas, presentando una relación directa con el cambiante uso que se le ha otorgado a los computadores, pasando de ser herramientas exclusivas de análisis y modelamiento, a utilizarse holísticamente en el proceso de diseño, incluyendo el diseño conceptual y el diseño integral, entre otros[20].

1.1. COMPUTACIÓN MULTI-OBJETIVA INSPIRADA EN LA EVOLUCIÓN DARWINIANA (CMOIED)

La intención de CMOIED es manejar múltiples objetivos conflictivos que usualmente aparecen en problemas de diseño de la vida real, comunes en el diseño estructural y arquitectónico. Ejemplos de este tipo de situaciones de conflicto son presentados por Pugnale y Sassone[21]. Ellos plantean que los requisitos arquitectónicos como el comportamiento estructural, los costos, la estética, y la funcionalidad, generalmente son conflictivos. Otros ejemplos pueden encontrarse en el trabajo de investigación de Byrne et al.[22] y Gaspar-Cunha et al.[23].

[20] KICINGER, R., ARCISZEWSKI, T. y DE JONG, K. Evolutionary computation and structural design: A survey of the state-of-the-art. En: Computers and Structures. No. 83 (2005); p. 1943-1978.

[21] PUGNALE, A. y SASSONE, M. Morphogenesis and structural optimization of shell structures with the aid of a Genetic Algorithm. En: Journal of the International Association for Shell and Spatial Structures. No. 48 (2007); p. 161-166.

[22] BYRNE, J., FENTON, M., HEMBERG, E., MCDERMOTT, J., O'NEIL, M., SHOTTON, E. y NALLY, C. Combining structural analysis and multi-objective criteria for evolutionary architectural design. En: Applications of Evolutionary Computation. No. 1 (2011); p. 204-213.

[23] GASPAR-CUNHA, A., LOYENS, D. y VAN HATTUM, F. Aesthetic Design Using Multi-Objective Evolutionary Algorithms. En: 6th INTERNATIONAL CONFERENCE ON EVOLUTIONARY MULTI-CRITERION OPTIMIZATION. Proceedings of the 6th International Conference on Evolutionary Multi-Criterion Optimization. Heidelberg: Springer-Verlag Berlin, 2011. p. 374-388.

La meta de los procesos de diseño que se llevan a cabo con CMOIED es alcanzar un número significativo de soluciones de Pareto[24], ampliamente diferenciadas[25], para un problema específico. En este sentido, CMOIED es una aplicación de la Computación Evolutiva (EC) para manejar múltiples objetivos.

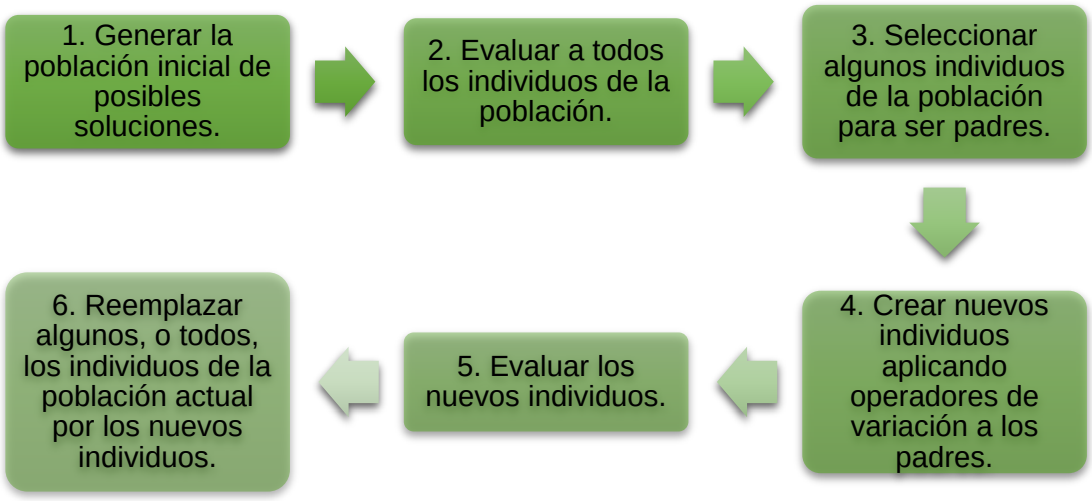
La EC es una técnica moderna de búsqueda, basada en modelos computacionales desarrollados por medio de un algoritmo. El algoritmo está inspirado en procesos evolutivos y de selección natural, tomando de ellos conceptos y mecanismos, los cuales son codificados y utilizados para resolver problemas en muchos campos de la ingeniería y la ciencia. A estos algoritmos se les conoce como Algoritmos Evolutivos (EA) y se clasifican en tres ramas principales de desarrollo[26]: Estrategias Evolutivas (ES)[27][28], Programación Evolutiva (EP)[29], y Algoritmos Genéticos (GA)[30]. Desde el punto de vista ingenieril, la EC puede entenderse como un proceso de búsqueda y optimización en el cual una población de soluciones pasa a través de una serie de cambios graduales, dependiendo del valor de idoneidad (de cada solución) definido por el entorno (una función objetivo).

A continuación se presentan los pasos necesarios para desarrollar un EA[31]:

---

[24] PARETO, V. Cours D'Economie Politique. Lausanne: F. Rouge, 1896.
[25] DEB, K. Evolutionary algorithms for multi-criterion optimization in engineering design. En: MIETTINEN, K. et al. Evolutionary algorithms in engineering and computer science: recent advances in genetic algorithms, evolution strategies, evolutionary programming, and industrial applications. New York: John Wiley & Sons, 1999.
[26] KICINGER, R., ARCISZEWSKI, T. y DE JONG, K. Evolutionary computation and structural design: A survey of the state-of-the-art. En: Computers and Structures. No. 83 (2005); p. 1943-1978.
[27] RECHENBERG, I. Cybernetic solution path of an experimental problem. Farnborough, 1965. Library Translation 1122. Royal Aircraft Establishment.
[28] SCHWEFEL, H.P. Kybernetische Evolution als Strategie der experimentelen Forschung in der Stromungstechnik. Berlín, 1965. Tesis de maestría. Technical University of Berlin. Hermann Fottinger Institute for Hydrodynamics.
[29] FOGEL, L.J., OWENS, A.J. y WALSH, M.J. Artificial Intelligence through simulated evolution. Chichester: John Wiley, 1966.
[30] HOLLAND, J.H. Adaptation in natural and artificial systems. Ann Arbor: University of Michigan Press, 1975.
[31] KICINGER, R., ARCISZEWSKI, T. y DE JONG, K. Op. cit.

Figura 6. Proceso de desarrollo de un EA



Fuente: Autor

Los pasos del 3 al 6 deben repetirse hasta que el criterio de parada sea alcanzado. Este criterio puede ser un número predefinido de generaciones, un valor de idoneidad deseado, o un cambio mínimo predefinido entre las soluciones de dos generaciones consecutivas (i.e. cuando el algoritmo converge). La definición del criterio de parada es problemática. Si el número de generaciones es utilizado como indicador, existe una alta posibilidad que la solución final no sea la óptima; es posible que al parar el algoritmo, el proceso evolutivo esté aún en desarrollo y no haya alcanzado la mejor solución. Algo similar ocurre si el criterio de parada se fija en base a un valor deseado de idoneidad; existe una posibilidad de no explotar el potencial del algoritmo al restringirlo a una meta predefinida, en vez de permitirle encontrar la solución óptima. Finalmente, si el criterio se basa en la convergencia del algoritmo, es posible que el esfuerzo computacional sea significativamente alto, especialmente en un problema de diseño complejo de

optimización multi-objetiva; es común que este tipo de problemas requieran varias iteraciones para poder alcanzar la solución óptima, resultando en un alto número de evaluaciones de rendimiento, generalmente complejas y consumidoras de tiempo.

1.1.1. Técnicas de Computación Multi-Objetiva Inspiradas en la Evolución Darwiniana (TCMOIED)

El primer trabajo de investigación en el que se aplicaron técnicas evolutivas (darwinianas) para resolver problemas multi-objetivos, fue desarrollado por Rosenberg[32]. El sugirió, pero no implementó, un método genético de búsqueda que involucraba múltiples propiedades bioquímicas y objetivos, de una población de organismos unicelulares. La primera implementación real fue hecha por Schaffer[33], en la cual él propuso, y aplicó con éxito, el Algoritmo Genético Evaluado Vectorialmente (VEGA)[34] para resolver tareas de discriminación de patrones multi-clase, en el campo de aprendizaje de máquinas. El siguiente paso significativo fue realizado por Goldberg[35] al proponer un procedimiento de clasificación basado en las soluciones no-dominadas. Desde ese momento, muchos investigadores han venido desarrollando distintas versiones de algoritmos de optimización multi-objetiva basados en principios evolutivos darwinianos[36].

[32] ROSENBERG, R.S. Simulation of genetic populations with biochemical properties. Ann Harbor, 1967. Tesis doctoral. University of Michigan.
[33] SCHAFFER, J.D. Some experiments in machine learning using vector evaluated genetic algorithms. Nashville, 1984. Tesis doctoral. Vanderbilt University.
[34] SCHAFFER, J.D. Multiple objective optimization with vector evaluated genetic algorithms. En: FIRST INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS. Proceedings on the First International Conference on Genetic Algorithms (ICGA'85). Pittsburgh: Grefenstette JJ, 1985. p. 93-100.
[35] GOLDBERG, D.E. Genetic algorithms in search, optimization, and machine learning. Reading: Addison-Wesley, 1989.
[36] KICINGER, R., ARCISZEWSKI, T. y DE JONG, K. Evolutionary computation and structural design: A survey of the state-of-the-art. En: Computers and Structures. No. 83 (2005); p. 1943-1978.

Recientemente, Zhou et al.[37] propusieron una clasificación para los Algoritmos Evolutivos Multi-Objetivos, MOEA (en los cuales se basa TCMOIED), con seis marcos teóricos (dependiendo del criterio de selección):

1. MOEAs basados en descomposición (MOEA/D)[38]:

   Se basa en los acercamientos convencionales de agregación en donde un problema multi-objetivo se descompone en un número de problemas escalares de optimización de un solo objetivo. El objetivo de cada problema escalar es la agregación con pesos de los objetivos individuales. Cada sub-problema individual guarda una solución en la memoria, que puede ser la mejor solución encontrada hasta el momento para dicho sub-problema. Para cada sub-problema, el algoritmo genera una nueva solución, ejecutando operadores genéticos en varios individuos de los sub-problemas vecinos. Luego, actualiza su memoria si la nueva solución es mejor que la anterior.

2. MOEAs basados en preferencia[39]:

   Se desarrolla a partir de la situación en la cual el diseñador puede estar interesado únicamente en soluciones de su preferencia, en vez de en todas las soluciones óptimas de Pareto. Para encontrar esas soluciones preferidas, es necesaria la información de preferencia para guiar la búsqueda hacia una región de interés del frente de Pareto, por parte del diseñador.

[37] ZHOU, A., QU, B.Y., LI, H., ZHAO, S.Z., NAGARATNAM SUGANTHAN, P. y ZHANG, Q. Multiobjective evolutionary algorithms: A survey of the state of the art. En: Swarm and Evolutionary Computation. No. 1 (2011); p. 32-49.

[38] ZHANG, Q. y LI, H. MOEA/D: a multiobjective evolutionary algorithm based on decomposition. En: IEEE CONGRESS ON EVOLUTIONARY COMPUTATION. Proceedings of the IEEE Congress on Evolutionary Computation. 2007. p. 712-731.

[39] FONSECA, C.M. y FLEMING, P.J. Genetic algorithms for multi-objective optimization: formulation, discussion, and generalization. En: 5th INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS. Proceedings of the fifth international conference on genetic algorithms. Los Altos: Morgan Kauffman, 1993. p. 416-423.

3. MOEAs basados en indicadores[40]:

   Se fundamenta en la medición, por medio de un indicador escalar como la distancia generacional y el híper-volumen, de la calidad de un frente de Pareto aproximado. Los MOEA's basados en indicadores usan dicho parámetro para guiar la búsqueda, particularmente para llevar a cabo el proceso de selección de individuos.

4. MOEAs híbridos[41][42][43]:

   En los Algoritmos Evolutivos Multi-Objetivos, existen muchas técnicas con distintas características y ventajas luego, hibridar dichas técnicas es una elección natural para utilizar las ventajas de cada una, con el fin de resolver problemas complejos de optimización multi-objetiva. Los problemas principales que se deben resolver al llevar a cabo un diseño de un MOEA híbrido serían la elección de las técnicas a utilizar y cómo se podrían hibridar.

5. MOEAs meméticos[44]:

   Son un caso especial de los MOEAs híbridos, incorporando métodos de búsqueda local. Los algoritmos meméticos son capaces de ofrecer mejores velocidades de convergencia al acercamiento evolutivo y, también, una mayor exactitud en las soluciones finales.

[40] ZITZLER, E. y KÜNZLI, S. Indicator-based selection in multi-objective search. En: Parallel Problem Solving from Nature. No. 7 (2004); p. 832-842.
[41] ZHOU, A., QU, B.Y., LI, H., ZHAO, S.Z., NAGARATNAM SUGANTHAN, P. y ZHANG, Q. Multiobjective evolutionary algorithms: A survey of the state of the art. En: Swarm and Evolutionary Computation. No. 1 (2011); p. 32-49.
[42] ELHOSSINI, A., AREIBI, S. y DONY, R. Strength Pareto particle swarm optimization and hybrid EA-PSO for multi-objective optimization. En: Evolutionary Computation. No. 18 (2010); p. 127-156.
[43] YANG, D., JIAO, L. y GONG, M. Adaptive multi-objective optimization based on nondominated solutions. En: Computational Intelligence. No. 25 (2009); p. 84-108.
[44] LARA, A., SÁNCHEZ, G., COELLO COELLO, C.A. y SCHUTZE, O. HCS: a new local search strategy for memetic multiobjective evolutionary algorithms. En: IEEE Transactions on Evolutionary Computation. No. 14 (2010); p. 112-132.

6. MOEAs basados en co-evolución[45][46]:

   Se basan en la evolución simultánea de múltiples sub-poblaciones para resolver un problema complejo. Los algoritmos que usan una estrategia de archivo se incluyen en esta categoría ya que ellos evolucionan una población y un archivo al mismo tiempo para aproximar el frente de Pareto de un problema de optimización multi-objetiva. Existe otra explicación de co-evolución haciendo uso de la idea *divide y vencerás*. Siguiendo esta idea, un algoritmo co-evolutivo divide el problema en un grupo de sub-problemas al nivel de codificación individual y, evoluciona múltiples sub-poblaciones. Las sub-poblaciones pueden ser competitivas y/o cooperativas (entre ellas). Los componentes de distintas sub-poblaciones se combinan para formar una solución completa.

Dentro de estos marcos teóricos, los acercamientos más populares encontrados en la literatura son los siguientes[47][48][49]:

- Funciones de agregación[50][51]:

  Múltiples objetivos son combinados en uno solo, utilizando adición, multiplicación, o cualquier otra combinación de operaciones aritméticas. La utilización de pesos se utiliza con frecuencia, en donde los objetivos se

[45] DEB, K., MOHAN, M. y MISHRA, S. Evaluating the epsilon-domination based multi-objective evolutionary algorithm for a quick computation of Pareto-optimal solutions. En: Evolutionary Computation. No. 13 (2005); p. 501-525.
[46] TAN, K.C., YANG, Y.L. y GOH, C.K. A distributed cooperative coevolutionary algorithm for multiobjective optimization. En: IEEE Transactions on Evolutionary Computation. No. 10 (2006); p. 527-549.
[47] KICINGER, R., ARCISZEWSKI, T. y DE JONG, K. Evolutionary computation and structural design: A survey of the state-of-the-art. En: Computers and Structures. No. 83 (2005); p. 1943-1978.
[48] COELLO, C.A. y CHRISTIANSEN, A.D. Multiobjective optimization of trusses using genetic algorithms. En: Computers and Structures. No. 75 (2000); p. 647-660.
[49] PARMEE, I.C. y BONHAM, C.R. Improving cluster oriented genetic algorithms for high-performance region identification. En: US UNITED ENGINEERING FOUNDATION'S OPTIMISATION IN INDUSTRY CONFERENCE. Proceedings US United Engineering Foundation's Optimisation in Industry Conference. Tuscany: Springer-Verlag, 2002.
[50] SYSWERDA, G. y PALMUCCI, J. The application of genetic algorithms to resource scheduling. En: FOURTH INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS. Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA'91). San Diego: Belew, R.K. y Booker, L.B., 1991. p. 502-508.
[51] COELLO COELLO, C.A. An updated survey of GA-based multiobjective optimization techniques. En: ACM Comput Surveys. No. 32 (2000); p. 109-143.

multiplican por coeficientes (peso) que representan la importancia relativa de estos. Dentro de las mayores desventajas de este método está la dificultad de determinar de manera apropiada los pesos y, el hecho de generar, posiblemente, soluciones de Pareto incorrectas en la presencia de espacios de búsqueda no-convexos, sin importar los pesos utilizados.

- Algoritmo Genético Evaluado Vectorialmente (VEGA)[52]:

  Maneja múltiples objetivos modificando el mecanismo de selección de supervivencia del algoritmo genético simple. La selección es proporcional a cada función objetivo (sub-poblaciones), luego se combinan dichas sub-poblaciones para generar una nueva población general.

- Orden lexicográfico[53]:

  Se realiza un ranking de los objetivos de acuerdo a la importancia de cada uno, luego se optimiza la función objetivo que corresponda al más importante y después, se optimizan las demás en el orden correspondiente al ranking. También se puede seleccionar aleatoriamente la función a optimizar en cada generación.

- Sumatoria con pesos[54]:

  Se le asigna un peso a cada objetivo y se promueve la diversidad de la población, compartiendo la idoneidad de cada solución (fitness). Como resultado se generan simultáneamente familias óptimas de Pareto, correspondientes a distintos coeficientes de pesos en una sola corrida del Algoritmo Genético.

---

[52] SCHAFFER, J.D. Multiple objective optimization with vector evaluated genetic algorithms. En: FIRST INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS. Proceedings on the First International Conference on Genetic Algorithms (ICGA'85). Pittsburgh: Grefenstette JJ, 1985. p. 93-100.

[53] RAO, S.S. Multiobjective optimization in structural design with uncertain parameters and stochastic processes. En: AIAA Journal. No. 22 (1984); p. 1670-1678.

[54] HAJELA, P. y LIN, C.Y. Genetic search strategies in multicriterion optimal design. En: Structural optimization. No. 4 (1992); p. 99-107.

- Algoritmo Genético Multi-Objetivo (MOGA)[55]:

  Se hace un ranking entre individuos, basado en el número de cromosomas que son dominados por la población presente. Este ranking se utiliza posteriormente para el proceso de selección.

- Algoritmo Genético con Clasificación No-dominada (NSGA)[5657]:

  Antes de la selección, se hace un ranking entre individuos, clasificando en una categoría todos los que sean no-dominados. Para mantener la diversidad en la población, se comparten algunos valores bajos de idoneidad (fitness) entre los individuos.

- Algoritmo Genético Pareto Anidado (NPGA)[58]:

  Se realiza una selección por torneo que determina la dominancia de Pareto. En vez de limitar la comparación a dos individuos, se usan otros individuos para determinar la dominancia.

- Algoritmo Genético Grupo Orientado mejorado (improved COGA) [59]:

  Se busca una región del espacio de búsqueda en donde las soluciones presenten alto rendimiento. En esta técnica es posible introducir conceptos de co-evolución para la optimización multi-objetiva. Se espera, y se han

---

[55] FONSECA, C.M. y FLEMING, P.J. Genetic algorithms for multi-objective optimization: formulation, discussion, and generalization. En: 5th INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS. Proceedings of the fifth international conference on genetic algorithms. Los Altos: Morgan Kauffman, 1993. p. 416-423.

[56] SRINIVAS, N. y DEB, K. Multiobjective optimization using non-dominated sorting in genetic algorithms. Kanpur, 1993. Reporte técnico. Indian Institute of Technology. Department of mechanical engineering.

[57] GOLDBERG, D.E. Genetic algorithms in search, optimization, and machine learning. Reading: Addison-Wesley, 1989.

[58] HORN, J. AND NAFPLIOTIS, N. Multiobjective optimization using the Niched Pareto genetic algorithm. Urbana, 1993. Reporte técnico. University of Illinois at Urbana Champaigne.

[59] PARMEE, I.C. y BONHAM, C.R. Improving cluster oriented genetic algorithms for high-performance region identification. En: US UNITED ENGINEERING FOUNDATION'S OPTIMISATION IN INDUSTRY CONFERENCE. Proceedings US United Engineering Foundation's Optimisation in Industry Conference. Tuscany: Springer-Verlag, 2002.

demostrado, mejoras en muestreo, selección, filtros, adaptabilidad y robustez en los algoritmos.

- Acercamientos de vector objetivo[60][61][62][63]:
  Las metas para cada objetivo son definidas por el diseñador (quien toma las decisiones). Este grupo de enfoques incluye programación de metas, consecución de metas, y enfoque de mínimos-máximos.

- Algoritmos Evolutivos Pareto-Fuertes (SPEA)[64]:
  Integra ideas de distintos métodos existentes de optimización evolutiva multi-objetiva y adiciona algunos elementos nuevos al algoritmo evolutivo multi-objetivo.

- Algoritmo Genético con Grupo de Pareto Reducido (RPSGA)[65][66]:
  Este algoritmo incorpora una técnica para agrupar las soluciones[67], con el fin de reducir su número en el frente eficiente, mientras se mantienen intactas sus características. En cada generación, todos los individuos de la población se reducen a un número predefinido de rankings. Luego, el valor de la función objetivo se calcula utilizando una función de ranking. Adicionalmente, una

[60] COELLO COELLO, C.A. An updated survey of GA-based multiobjective optimization techniques. En: ACM Comput Surveys. No. 32 (2000); p. 109-143.
[61] CHARNES, A. y COOPER, W.W. Management models and industrial applications of linear programming. New York: John Wiley, 1961.
[62] CHEN, Y.L. y LIU, C.C. Multiobjective VAR planning using the goal-attainment method. En: IEEE Proc Generat Transm Distrib. No. 141 (1994); p. 227-232.
[63] COELLO COELLO, C.A. y CHRISTIANSEN, A.D. Two new GA-based methods for multiobjective optimization. En: Civil Eng Syst. No. 15 (1998); p. 207-243.
[64] ZITZLER, E. AND THIELE, L. An evolutionary algorithm for multi-objective optimization: the strength Pareto approach. Zurich, 1998. Reporte técnico. Swiss Federal Institute of Technology. Computer engineering and Communication Networks Laboratory.
[65] GASPAR-CUNHA, A. y COVAS, J.A. RPSGAe - A Multiobjective Genetic Algorithm with Elitism: Application to Polymer Extrusion. En: Metaheuristics for Multi-objective Optimisation. Heidelberg: Springer, 2004.
[66] GASPAR-CUNHA, A. Modeling and Optimization of Single Screw Extrusion. Minho, 2000. Tesis doctoral. University of Minho.
[67] ROSEMAN, M.A. y GERO, J.S. Reducing the Pareto Optimal Set in Multicriteria Optimization. En: Eng. Optim. No. 8 (1985); p. 189-206.

población elitista externa es mantenida simultáneamente para preservar las mejores soluciones que se han encontrado y, para incorporar periódicamente algunas de estas en la población principal.

La posibilidad de manejar restricciones con las técnicas anteriores es un factor determinante. En casi todos los problemas de diseño de la vida real, las restricciones son propósitos que deben alcanzarse. Existen distintos métodos para cumplir con este tipo de requisitos, y Coello Coello los clasificó en cinco categorías[68]: funciones de penalización, representaciones y operadores especiales, algoritmos de reparación, separación de objetivos y restricciones, y métodos híbridos. Otra opción es la que presenta Zhou et al.[69], en la cual las restricciones son tratadas como objetivos adicionales durante el proceso de optimización. Una clasificación de este tipo de métodos puede encontrarse en Cai et al.[70].

Aparte del manejo de las restricciones, otra habilidad que TCMOIED (respecto al diseño arquitectónico y estructural) debe tener es el manejo de una cantidad significativa de objetivos. Para esto, existen algunas opciones: la primera es modificar la relación de la dominancia de Pareto y la definición de la clasificación[71] con el fin de incrementar la presión de selección hacia el frente de Pareto; otra

[68] COELLO COELLO, C.A. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. En: Computer Methods in Applied Mechanics and Engineering. No. 191 (2002); p. 1245-1287.
[69] ZHOU, A., QU, B.Y., LI, H., ZHAO, S.Z., NAGARATNAM SUGANTHAN, P. y ZHANG, Q. Multiobjective evolutionary algorithms: A survey of the state of the art. En: Swarm and Evolutionary Computation. No. 1 (2011); p. 32-49.
[70] CAI, Z. y WANG, Y. A multiobjective optimization-based evolutionary algorithm for constrained optimization. En: IEEE Transactions on Evolutionary Computation. No. 10 (2006); p. 658-675.
[71] CORNE, D. y KNOWLES, J.D. Techniques for highly multiobjective optimisation: Some nondominated points are better than others. En: Conference on Genetic and Evolutionary Computation. No. 1 (2007); p. 773-780.

opción es una propuesta de un MOEA dinámico basado en conceptos de la termodinámica[72].

1.1.2. TCMOIED aplicadas al diseño estructural y arquitectónico

Los trabajos de investigación enfocados en la aplicación de EC para resolver problemas de diseño en ingeniería empezaron en Europa con Rechenberg[73], en el campo de mecánica de fluidos, diseño de tuberías, e ingeniería estructural. Las aplicaciones más tempranas en el último campo[74][75] utilizaban ES, la cual evolucionó del acercamiento para la optimización estructural desarrollado a comienzos de la década de 1960[76].

Los métodos formales (métodos tradicionales, no heurísticos, de optimización) de optimización estructural que suponen una continuidad, funcionaban de manera correcta en problemas relativamente bien formulados, donde la configuración estructural de los elementos se asume como fija durante un proceso de optimización que pretende encontrar las dimensiones óptimas de dichos elementos, mientras satisface requisitos de diseño y restricciones. La simple generalización de este problema, al permitir variaciones en la configuración del sistema, incrementa significativamente la complejidad de la tarea de optimización, y expone como inadecuados muchos de los métodos tradicionales. Este tipo de problemas pueden dividirse en tres grupos, cada uno con distinta complejidad. El

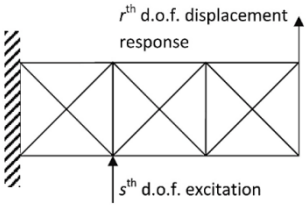
[72] ZOU, X., CHEN, Y., LIU, M. y KANG, L. A new evolutionary algorithm for solving many-objective optimization problems. En: IEEE Transactions on Systems, Man and Cybernetics. No. 38 (2008); p. 1402-1412.
[73] RECHENBERG, I. Evolutionsstrategie: Optimierung TechnischerSysteme nach Prinzipien der biologischen Evolution. Stuttgart-Bad Cannstatt: Frommann-Holzboog, 1973.
[74] HOEFFLER, A., LEYSNER, U. y WEIDERMANN, J. Optimization of the layout of trusses combining strategies based on Mitchel's theorem and on biological principles of evolution. En: II SYMPOSIUM ON STRUCTURAL OPTIMIZATION. Milan: s.e., 1973.
[75] LAWO, M. y THIERAUF, G. Optimal design for dynamic stochastic loading: a solution by random search. En: Optimization in structural design. University of Siegen (1982); p. 346-352.
[76] KICINGER, R., ARCISZEWSKI, T. y DE JONG, K. Evolutionary computation and structural design: A survey of the state-of-the-art. En: Computers and Structures. No. 83 (2005); p. 1943-1978.

primero y más complejo, es la optimización topológica, seguido por la optimización de forma (manteniendo fija la topología), y por último la optimización dimensional (manteniendo fija la topología y la forma). Cada uno de estos problemas puede ser desarrollado dentro de un sistema continuo o discreto[77].

A continuación se presentan algunos de los trabajos de investigación más relevantes del campo de estudio, enfocados en ejemplos particulares:

Noilublao y Bureerat[78] presentaron un acercamiento para alcanzar simultáneamente una optimización topológica, de forma, y dimensional, para una cercha tri-dimensional. Los objetivos planteados fueron el peso, la rigidez, las frecuencias naturales, la función de respuesta de frecuencia (FRF), y la transmisibilidad de fuerzas (FT). Con estos objetivos se desarrollaron cuatro problemas de optimización multi-objetiva: peso – rigidez, peso – frecuencia, peso – FRF, y peso – FT.

Figura 7. Medición de FRF



Fuente: NOILUBLAO, N. y BUREERAT, S. Simultaneous topology, shape and sizing optimisation of a three-dimensional slender truss tower using multiobjective evolutionary algorithms. En: Computers and Structures. No. 89 (2011); p. 2531-2538.

[77] KICINGER, R., ARCISZEWSKI, T. y DE JONG, K. Evolutionary computation and structural design: A survey of the state-of-the-art. En: Computers and Structures. No. 83 (2005); p. 1943-1978.

[78] NOILUBLAO, N. y BUREERAT, S. Simultaneous topology, shape and sizing optimisation of a three-dimensional slender truss tower using multiobjective evolutionary algorithms. En: Computers and Structures. No. 89 (2011); p. 2531-2538.

Para el problema de optimización se utilizaron dos MOEAs, el SPEA y el PBIL (Population-Based Incremental Learning)[79], los cuales presentan el mejor rendimiento en este tipo de problemas[80]. El algoritmo heurístico Simulated Annealing (se expondrá en la siguiente sección), basado en archivos, también fue utilizado ya que el acercamiento propuesto no ha sido probado. Las conclusiones fueron que el PBIL es superior cuando el objetivo es la rigidez, mientras que el SPEA es mejor cuando la optimización está enfocada en la FRF y la FT. En el caso de la maximización de frecuencias naturales, o maximizar la rigidez dinámica estructural, tanto el PBIL como el SPEA tuvieron resultados similares. Uno tiene un mejor frente de Pareto y el otro presenta un frente más avanzado.

El acercamiento propuesto por Noilublao y Bureerat puede convertirse en una herramienta poderosa para el diseño de cerchas y marcos. Con una sola corrida del algoritmo, el diseñador puede obtener varias alternativas de estructuras óptimas, las cuales serán muy útiles durante el proceso de toma de decisiones.

Gaspar-Cunha et al.[81] desarrollaron un método computacional para optimizar interactivamente un diseño bajo ciertos requerimientos funcionales, utilizando herramientas disponibles para simulación y diseño interactivo. Este método fue aplicado en la optimización de una estructura genérica de cubierta, enfocada en condiciones de asoleamiento y en la minimización del área de la superficie. Se utilizaron MOEAs en conjunto con una metodología para la toma de decisiones, lo cual condujo a una interacción crítica entre el diseñador y el procedimiento.

[79] BUREERAT, S. y SRIWORAMAS, K. Population-based incremental learning for multi-objective optimisation. En: Soft Computing in Industrial Applications. No. 39 (2007); p. 223-232.
[80] NOILUBLAO, N. y BUREERAT, S. Simultaneous topology, shape and sizing optimisation of skeletal structures using multiobjective evolutionary algorithms. En: Evolutionary Computation. No. 24 (2009); p. 487-580.
[81] GASPAR-CUNHA, A., LOYENS, D. y VAN HATTUM, F. Aesthetic Design Using Multi-Objective Evolutionary Algorithms. En: 6th INTERNATIONAL CONFERENCE ON EVOLUTIONARY MULTI-CRITERION OPTIMIZATION. Proceedings of the 6th International Conference on Evolutionary Multi-Criterion Optimization. Heidelberg: Springer-Verlag Berlin, 2011. p. 374-388.

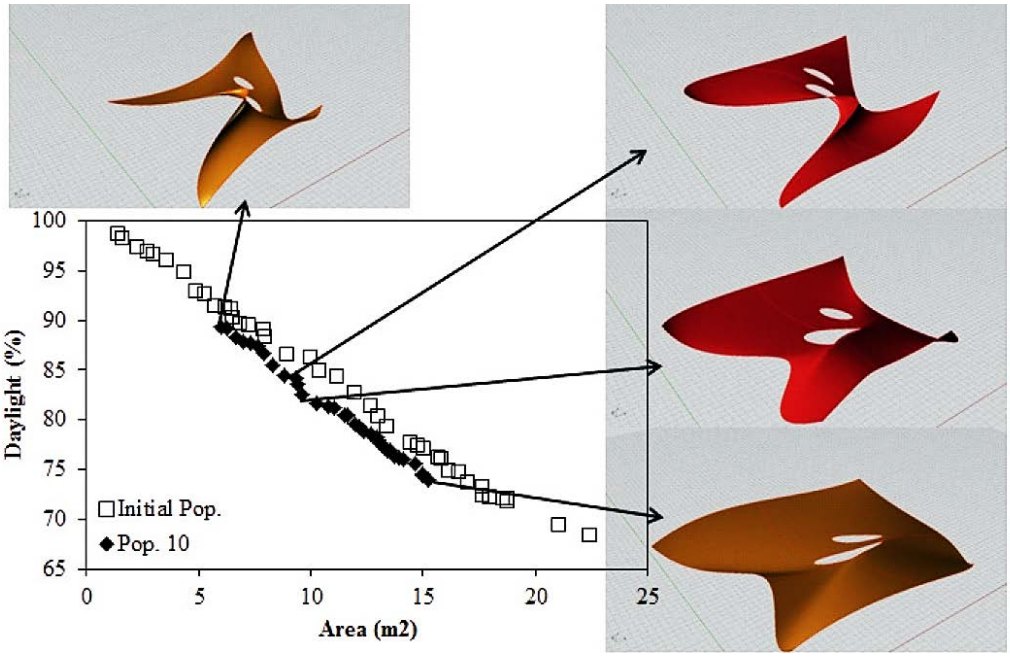
Un aspecto importante de este trabajo es que el método propuesto permite la integración de complejos requerimientos cuantitativos y cualitativos durante la fase inicial del proceso de diseño. Por ejemplo, como punto de partida se tomó la geometría genérica de una estructura de cubierta, representada con una única superficie y descrita por medio de Non-Uniform Rational B-Splines (NURBS)[82], utilizando un software de modelado tri-dimensional llamado Rhinoceros[83]. A lo largo de la superficie se definieron 20 puntos de control con grados de libertad en los ejes *x*, *y*, y *z*.

Debido a la subjetividad y a la imposibilidad de cuantificar el objetivo estético, se adoptó una estrategia que tiene en cuenta las preferencias del diseñador. Dicho procedimiento utiliza una función de agregación para manejar los múltiples objetivos, mientras que para la estrategia de selección se utilizó el RPSGA y el acercamiento de ruleta.

En todos los casos el algoritmo corrió solo durante 10 generaciones debido al tiempo computacional requerido por el software de modelamiento. Los resultados obtenidos mostraron que el algoritmo era capaz de evolucionar las soluciones con éxito, generando un frente de Pareto bien definido en cada caso. Al mismo tiempo, los resultados clarificaron la utilidad del método y las técnicas desarrolladas en la fase temprana del proceso de diseño, conduciendo en conjunto hacia la obtención de mejores soluciones para un determinado problema.

[82] PIEGL, L. y TILLER, W. The NURBS book. Berlin: Springer, 1997.
[83] ROBERT MCNEEL & ASSOCIATES. Rhinoceros [en línea]. <http://www.rhino3d.com/> [citado el 14 de Febrero de 2013].

Figura 8. Frente de Pareto considerando una combinación de pesos de 0.5 para los dos objetivos (asoleamiento y área)



Fuente: GASPAR-CUNHA, A., LOYENS, D. y VAN HATTUM, F. Aesthetic Design Using Multi-Objective Evolutionary Algorithms. En: 6th INTERNATIONAL CONFERENCE ON EVOLUTIONARY MULTI-CRITERION OPTIMIZATION. Proceedings of the 6th International Conference on Evolutionary Multi-Criterion Optimization. Heidelberg: Springer-Verlag Berlin, 2011. p. 374-388.

La estrategia propuesta por Gaspar-Cunha et al. le permite al diseñador tener el control, de forma interactiva, del resultado del elemento arquitectónico y guiar el proceso hacia una solución estéticamente agradable. De esta manera, el diseñador puede dejar a un lado la intuición a la hora de resolver requerimientos conflictivos y complejos de diseño, y enfocar su atención y sus esfuerzos en la creación de soluciones innovadoras y estéticamente agradables.

Winslow et al.[84] trabajaron en el problema de crear una estructura reticular eficiente, teniendo en cuenta todas las dificultades y la complejidad que emerge debido al rango de requerimientos presente en los proyectos de ingeniería y arquitectura.

En su trabajo, ellos presentaron un método innovador para sintetizar estructuras reticulares a lo largo de superficies de forma libre, utilizando MOGAs para encontrar la orientación óptima de las barras que conforman la estructura. El procedimiento utiliza un algoritmo basado en el NSGA para el proceso de selección.

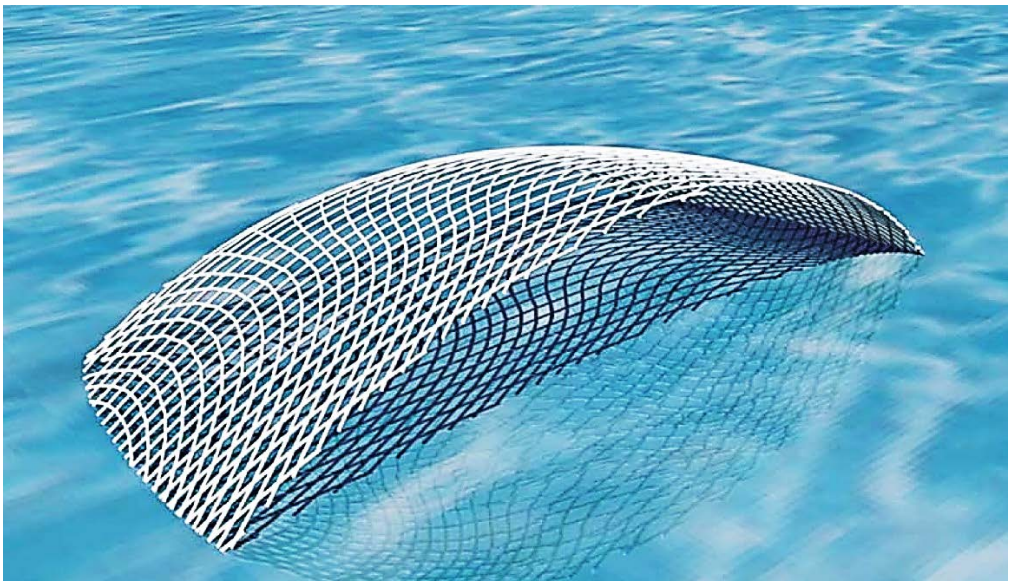
Así la estructura final deba ser un grupo discreto de elementos, el procedimiento planteado desarrolla una homogenización que permite representar, durante todo el proceso de optimización, el entramado de barras como una cáscara continua anisotrópica de elementos finitos. Adicionalmente, se creó una parametrización del problema, permitiendo reducir el número de variables de diseño y, haciendo más viable el uso del MOGA por medio de la mejora del factor constructivo y el atractivo visual de los diseños optimizados.

Dos ejemplos fueron expuestos: el primero era un domo sujeto a dos casos de carga excéntrica, teniendo como objetivos la minimización de las deformaciones bajo la acción de la carga asimétrica de viento combinada con la carga de peso propio, y la minimización de las deformaciones bajo la combinación de cargas de peso propio y una aceleración lateral de 2m/s; el segundo ejemplo era un arco con doble curvatura y luz de 54m, teniendo como objetivos la minimización de deformaciones bajo la combinación de cargas de peso propio y una carga asimétrica de viento, y minimizar las deformaciones bajo la combinación de cargas de peso propio y una carga simétrica de viento.

[84] WINSLOW, P., PELLEGRINO, S. y SHARMA, S.B. Multi-objective optimization of free-form grid structures. En: Struct Multidisc Optim. No. 40 (2010); p. 257-269.

Los resultados mostraron que, dada una superficie y una retícula deseada, la orientación de las barras que conforman la estructura reticular puede evolucionar con éxito, con el fin de crear una población de diseño óptima.

Figura 9. Resultado del segundo ejemplo de optimización (arco)



Fuente: WINSLOW, P., PELLEGRINO, S. y SHARMA, S.B. Multi-objective optimization of free-form grid structures. En: Struct Multidisc Optim. No. 40 (2010); p. 257-269.

De los resultados obtenidos, puede concluirse que al correr el procedimiento propuesto con un número moderado de iteraciones, se puede obtener un grupo de soluciones bien diferenciadas que parecen ser construibles y estéticamente racionales.

Otra aplicación de la EC multi-objetiva en el diseño arquitectónico está enfocada en el diseño sostenible y bioclimático. Por ejemplo, Cagne y Andersen[85] plantearon una herramienta basada en un GA que facilita la exploración de diseños de fachadas, teniendo en cuenta los objetivos que se pretenden lograr: iluminación y/o deslumbramiento.

El método le permite al usuario ingresar las metas requeridas de rendimiento y un modelo original tri-dimensional de la masa de la edificación, asumiendo que su forma es constante y que sus elementos de fachada son presentados como las variables.

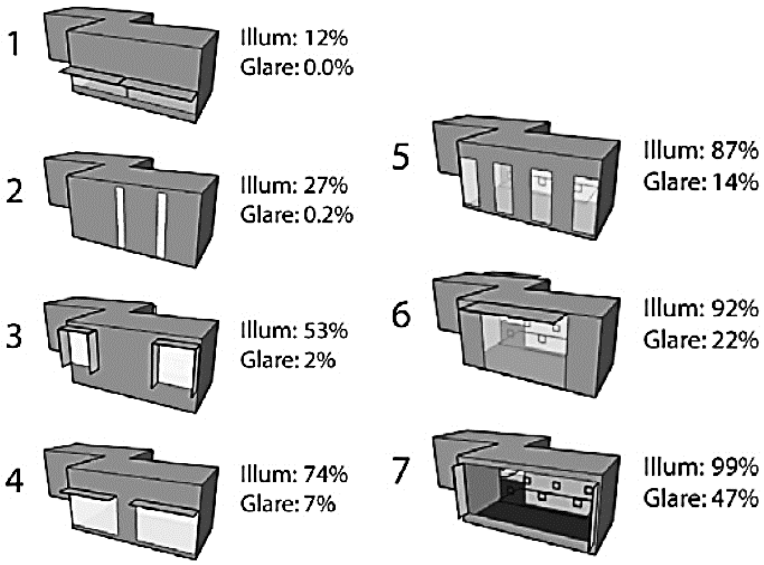
El procedimiento considera 10 parámetros de fachada, incluyendo los materiales de las ventanas, las características geométricas de las aperturas de las mismas, y los elementos generadores de sombra.

Para este propósito, un modelo de información de la edificación (BIM) es utilizado con el fin de generar automáticamente un modelo tri-dimensional para cada individuo.

Los resultados de los casos de optimización sencilla y multi-objetiva (iluminación y deslumbramiento) mostraron que el proceso de diseño es exitoso, guiando las soluciones para alcanzar los objetivos predeterminados.

[85] GAGNE, J.M.L. y ANDERSEN, M. Multi-objective facade optimization for daylighting design using a Genetic Algorithm. En: SIMBUILD 2010 - 4TH NATIONAL CONFERENCE OF IBPSA-USA. Proceedings of SimBuild 2010 - 4th National Conference of IBPSA-USA. New York: s.e., 2010.

Figura 10. Siete ejemplos de diseño del frente de Pareto (objetivos: iluminación y deslumbramiento)



Fuente: GAGNE, J.M.L. y ANDERSEN, M. Multi-objective facade optimization for daylighting design using a Genetic Algorithm. En: SIMBUILD 2010 - 4TH NATIONAL CONFERENCE OF IBPSA-USA. Proceedings of SimBuild 2010 - 4th National Conference of IBPSA-USA. New York: s.e., 2010.

Luego de su trabajo de investigación, Cagne y Andersen plantearon que los procedimientos basados en GAs presentan limitaciones significativas. Ellos argumentan que las soluciones encontradas no tienen consistencia debido a la importancia y al significado del rol que juegan las soluciones de diseño iniciales, generadas aleatoriamente. El acercamiento para solucionar, hasta cierto punto, esta limitación, es correr el algoritmo durante muchas generaciones, sabiendo que esto incrementará el tiempo y el esfuerzo computacional. Ellos también exponen otra limitación relacionada con la tendencia que tienen los GAs de quedarse atrapados en un mínimo o máximo local. Sin embrago, ellos argumentan que debido a los propósitos que tiene la exploración de diseños basados en rendimiento, no es necesario encontrar el óptimo global. En lugar de eso, la meta

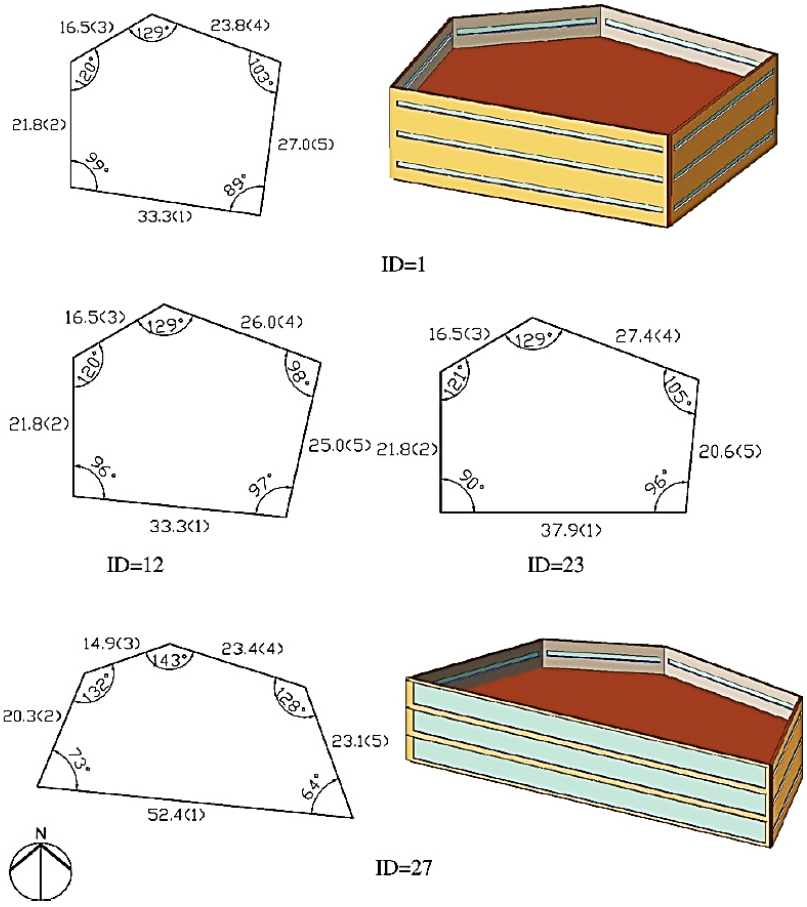
de esta metodología de diseño es ofrecerle al usuario un grupo de soluciones de diseño de alta calidad, con el fin de utilizarlas como alternativas iniciales de diseño, en vez de tomarlas como un diseño final.

Por otro lado, Wang et al.[86] plantearon la desventaja evidente que aparece cuando la toma de decisiones acerca de la forma de una edificación, se basa únicamente en consideraciones estéticas, lo cual es una práctica común. En este sentido, el potencial que tienen las soluciones formales para mejorar el rendimiento del edificio es limitado.

Por consiguiente, ellos proponen el uso de un MOGA, con una representación poligonal generalizada para la optimización de la forma de la planta, considerando dos objetivos conflictivos: minimizar el costo del ciclo de vida y el impacto medio-ambiental que este produce. Las variables incluidas en el proceso de optimización estaban relacionadas con la forma, la estructura, la configuración del recubrimiento, y los aleros o elementos generadores de sombra. El algoritmo propuesto fue probado en el diseño de un edificio de oficinas en altura, ubicado en Montreal, Canadá. El área de la planta del edificio era un pentágono de 1000m^2, y la altura del entre-piso fue de 3.60m. La temporada de calefacción tenida en cuenta para la simulación energética del edificio se fijó entre Noviembre y Marzo. La temporada de enfriamiento, para el mismo propósito, se fijó de Junio a Agosto. Las temperaturas de diseño para el interior de la edificación fueron de 21°C para la calefacción, y de 23°C para el enfriamiento. En el análisis del ciclo de vida se utilizó un periodo de 40 años. Las variables que describen la forma del edificio, la configuración del recubrimiento, el sistema estructural, y los aleros, son discretas, lo cual da como resultado una lista de posibles opciones.

[86] WANG, W., RIVARD, H. y ZMEUREANU, R. Floor shape optimization for green building design. En: Advanced Engineering Informatics. No. 20 (2006); p. 363-378.

Figura 11. Formas de la planta del edificio, seleccionadas del Frente de Pareto



Fuente: WANG, W., RIVARD, H. y ZMEUREANU, R. Floor shape optimization for green building design. En: Advanced Engineering Informatics. No. 20 (2006); p. 363-378.

Los resultados obtenidos mostraron mejoras significativas en las soluciones encontradas, alcanzando números bastante buenos para el costo del ciclo de vida y el impacto medio-ambiental de este. Este trabajo es un ejemplo de cómo este tipo de procedimiento de optimización resuelve las desventajas que se presentan en los métodos tradicionales de diseño que se basan en prueba y error, al explorar distintas formas de edificios para el diseño de edificaciones sostenibles. Por

último, este tipo de problema se plantea como una rama de investigación importante y con futuro potencial.

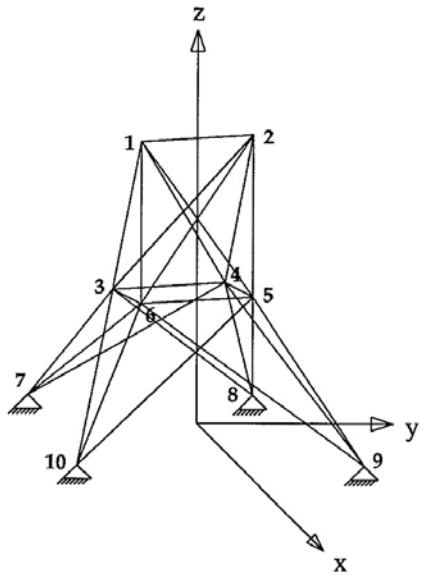
Coello Coello y Christiansen[87] desarrollaron un método para la optimización multi-objetiva de cerchas, utilizando GAs. Como primer paso, ellos generaron aleatoriamente la población inicial, conformada únicamente por individuos viables. Posteriormente, se asignó un vector de pesos, el cual lleva a que cada proceso se convierta en un GA individual. En cada GA, la combinación dada de pesos se utilizó en conjunto con el acercamiento min-max[88] con el fin de generar una única solución. Como paso siguiente, el valor de idoneidad de cada cromosoma se calculó, de acuerdo con el acercamiento min-max. Los operadores de mutación y crossover fueron modificados posteriormente para garantizar únicamente soluciones viables. Por último, se generó un archivo final en el que se incluían las soluciones no-dominadas que se habían encontrado durante el proceso (este paso se realiza luego de terminar los *n* procesos, siendo *n* el número de combinaciones de pesos dadas por el usuario).

El método propuesto se aplicó en el desarrollo de dos ejemplos: el primero era una cercha tri-dimensional con 25 barras, y el segundo era una cercha en dos dimensiones con 200 barras. Los objetivos en los dos ejemplos fueron la minimización del peso, de las deformaciones máximas, y de los esfuerzos, utilizando el área de la sección transversal de cada barra como las variables de diseño. El acercamiento propuesto es robusto ya que transforma el problema de optimización multi-objetiva en varios problemas de optimización con un solo objetivo, que pueden resolverse con un menor costo y esfuerzo computacional. En cuanto a la representación, el método parece funcionar mejor, más rápido, y con mayor precisión, cuando se utiliza la representación de punto flotante, contrario a

[87] COELLO, C.A. y CHRISTIANSEN, A.D. Multiobjective optimization of trusses using genetic algorithms. En: Computers and Structures. No. 75 (2000); p. 647-660.
[88] Ibid.

los demás acercamientos considerados en el trabajo de investigación de Coello Coello y Christiansen.

Figura 12. Cercha tri-dimensional de 25 barras, utilizada en el primer ejemplo



Fuente: COELLO, C.A. y CHRISTIANSEN, A.D. Multiobjective optimization of trusses using genetic algorithms. En: Computers and Structures. No. 75 (2000); p. 647-660.

La mayor desventaja de este acercamiento es que requiere un vector ideal y un grupo de pesos con el fin de delinear el grupo de soluciones óptimas del frente de Pareto. De todas formas, si el vector ideal no se conoce de antemano, se le puede otorgar al proceso un grupo de valores para las metas deseadas (para cada objetivo), en lugar de dicho vector. Por otro lado, una de las ventajas más significativas es que el procedimiento planteado garantiza siempre la producción de puntos viables en la generación inicial, mientras que los operadores modificados de mutación y crossover fuerzan al algoritmo para generar únicamente soluciones factibles. Esta propiedad hace único a este acercamiento

ya que ninguna de las otras técnicas basadas en GAs, y revisadas en el artículo[89], considera este aspecto importante.

Fialho et al.[90] intentaron resolver el conflicto que aparece en un proceso de optimización enfocado en la eficiencia energética de una edificación y sus costos de construcción. Un edificio energéticamente más eficiente tiene un proceso constructivo mucho más costoso.

En su trabajo, la optimización de los diseños de un edificio se logró combinando un software de simulación de consumo de energía, EnergyPlus[91], con Hype[92], un EA multi-objetivo.

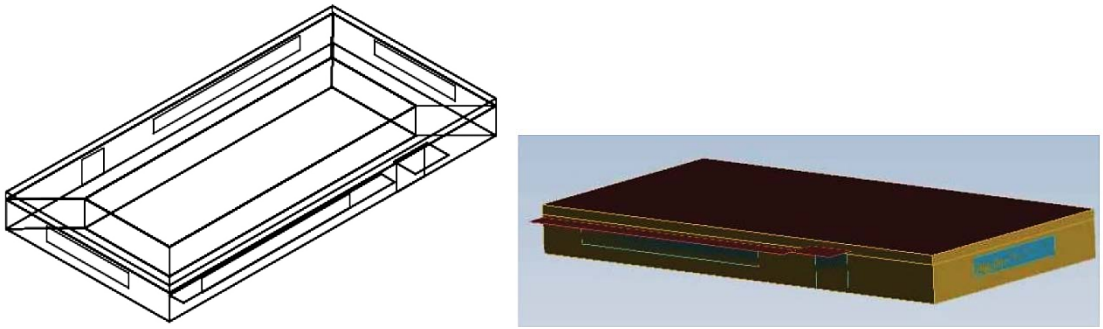
Los objetivos del procedimiento fueron la minimización del consumo de energía y de los costos de construcción, mientras que las variables fueron el ángulo de orientación del edificio y el material utilizado en cada una de las capas aislantes de los muros exteriores (para un total de 3 variables).

Como problema de prueba, el procedimiento de optimización propuesto se aplicó en el diseño de un edificio comercial de un solo piso, tomado de[93], con una planta rectangular de 30.5m por 15.2m, una altura promedio de entre-piso de 3.0m, y

[89] COELLO, C.A. y CHRISTIANSEN, A.D. Multiobjective optimization of trusses using genetic algorithms. En: Computers and Structures. No. 75 (2000); p. 647-660.
[90] FIALHO, A., HAMADI, Y. y SCHOENAUER, M. Optimizing Architectural and Structural Aspects of Buildings towards Higher Energy Efficiency. En: 13TH ANNUAL CONFERENCE COMPANION ON GENETIC AND EVOLUTIONARY COMPUTATION. Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation GECCO'11. New York: ACM, 2011. p. 727-732.
[91] STRAND, R., CRAWLEY, D., PEDERSEN, C., LIESEN, R., LAWRIE, L., WINKELMANN, F., BUHL, W., HUANG, Y. y FISHER, D. EnergyPlus: A new-generation energy analysis and load calculation engine for building design. En: Proc. Association of Collegiate Schools of Architecture Technology Conference. 2000.
[92] BADER, J. y ZITZLER, E. Hype: An algorithm for fast hypervolume-based many-objective optimization. En: Evolutionary Computation. No. 19 (2011); p. 45-76.
[93] Getting started with EnergyPlus. 2010. Reporte técnico. Lawrence Berkeley National Laboratory.

ventanas en las cuatro fachadas (para una descripción más detallada, ver[94], ejercicio 2C).

Figura 13. Vistas del edificio utilizado en el ejemplo de optimización



Fuente: FIALHO, A., HAMADI, Y. y SCHOENAUER, M. Optimizing Architectural and Structural Aspects of Buildings towards Higher Energy Efficiency. En: 13TH ANNUAL CONFERENCE COMPANION ON GENETIC AND EVOLUTIONARY COMPUTATION. Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation GECCO'11. New York: ACM, 2011. p. 727-732.

Aunque el trabajo está en su fase inicial, los resultados son prometedores al alcanzar los objetivos planteados. Una conclusión interesante que Fialho et al. plantean en su trabajo de investigación, es que el marco conceptual que ellos presentan es bastante productivo y útil para los arquitectos, no solo por un ahorro en el tiempo de diseño, sino también por proporcionar soluciones energéticamente más eficientes, que quizá no se hubiesen podido explorar con una metodología tradicional de diseño.

[94] Getting started with EnergyPlus. 2010. Reporte técnico. Lawrence Berkeley National Laboratory.

Pugnale y Sassone[95] plantearon un proceso de optimización estructural y morfogenético para estructuras tipo cascarón, basado en un GA. La morfogénesis estructural puede ser interpretada como una optimización multi-objetiva en la cual el valor de idoneidad es asignado sobre la base de un proceso de decisión con múltiples criterios. De esta manera, la configuración final será una solución aceptable, en vez de óptima, en donde todos los objetivos serán alcanzados o, al menos, parcialmente cumplidos.

Dicho proceso morfogenético de optimización fue aplicado al diseño estructural y arquitectónico de cascarones delgados de concreto, con superficie de forma libre.

Para desarrollar la generación computacional de forma y el proceso de optimización para este tipo de estructuras, fueron necesarias tres herramientas básicas: primero, es necesario definir la forma estructural desde el punto de vista geométrico; segundo, la estructura debe ser analizada con el fin de estudiar su comportamiento estructural; por último, el GA es necesario con el fin de desarrollar el proceso de optimización evolutiva de la forma estructural. Consecuentemente, la solución final óptima será el resultado de la interacción entre la herramienta propuesta de generación geométrica y el trabajo del arquitecto e ingeniero.

En el ejemplo de aplicación, la forma de la planta del edificio, la posición y la altura de las columnas, y la proyección *x-y* del contorno de la cubierta, fueron asumidos como valores fijos. El espesor de la cáscara de concreto se asumió de 0.15m, y la única aplicación de carga para el análisis estructural, por medio del método de elementos finitos, fue la del peso propio de la cubierta. La forma del cascarón fue descrita como una superficie NURBS[96] de tercer orden con una retícula de 10x10 puntos de control. La posición vertical de los puntos de control fue asumida como

[95] PUGNALE, A. y SASSONE, M. Morphogenesis and structural optimization of shell structures with the aid of a Genetic Algorithm. En: Journal of the International Association for Shell and Spatial Structures. No. 48 (2007); p. 161-166.
[96] PIEGL, L. y TILLER, W. The NURBS book. Berlin: Springer, 1997.

las variables durante el proceso morfogenético. Ciertos puntos de control se asumieron como fijos durante dicho proceso con el fin de representar los nodos entre el cascarón y las columnas.

Figura 14. Crematorio en Kakamigahara, ejemplo de aplicación

Fuente: http://openbuildings.com/buildings/crematorium-in-kakamigahara-profile-45079/media?group=image.

De este trabajo de investigación, se puede concluir que la optimización estructural, enfocada en la minimización de las deformaciones máximas verticales, bajo la acción de la carga de peso propio, mejora el comportamiento estructural en alrededor de 10 veces, al correr el algoritmo durante 75 generaciones. Esto es alcanzado por medio de modificar selectivamente las partes de la estructura que presentan el peor comportamiento, haciendo posible el proceso morfogenético, en donde la forma y la función van de la mano.

Los autores anteriores también trabajaron en el diseño óptimo de cascarones reticulares de vidrio[97], utilizando un GA. El procedimiento presentado en su artículo está basado en un proceso de optimización que puede ser aplicado para resolver un amplio rango de problemas, sin estar limitado por la forma global o por el número de lados del polígono que define la geometría del cascarón. La convergencia hacia la solución óptima se garantiza por la robustez del algoritmo. Cuando el problema es indeterminado (existiendo más de una solución óptima), el procedimiento puede ser utilizado para explorar empíricamente el rango de soluciones óptimas.

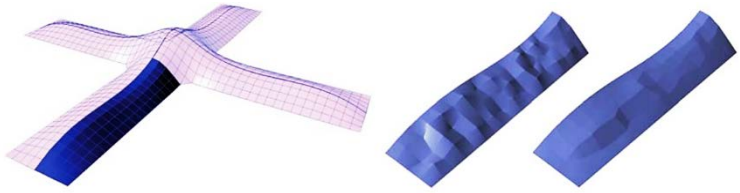
Por otro lado, cuando el problema está enfocado en la búsqueda de una configuración espacial, descrita por un grupo de coordenadas, puntos de control, u otros parámetros geométricos, la representación de la solución dentro del algoritmo debe mantener la estructura topológica del problema, siendo ésta una de las razones por las cuales se utilizó una representación NURBS[98]. De esta manera, la forma del cascarón reticular es descrita por medio de una matriz bidimensional de coordenadas, en la cual cada elemento almacena la coordenada *x*, *y*, y *z* de los nodos de la retícula como números reales (el uso de números reales al codificar las variables de diseño es parte de la estrategia evolutiva adoptada). El método de selección de ruleta rusa se utilizó como regla para seleccionar la población de individuos que iba a generar la cría para la siguiente generación, así como para seleccionar el número de nuevos individuos. Con el fin de garantizar una mejora continua en el rendimiento de cada individuo, se implementó también el método de selección elitista. La medida de idoneidad, o de rendimiento, de cada solución se basó en la cuantificación de la no-planaridad de un grupo de cuatro puntos (que definían una celda de la retícula). Para este

[97] SASSONE, M. y PUGNALE, A. Optimal design of glass grid shells with quadrilateral elements by means of a genetic algorithm. En: 6TH INTERNATIONAL CONFERENCE ON COMPUTATION OF SHELL AND SPATIAL STRUCTURES. Proceedings of the 6th International Conference on Computation of Shell and Spatial Structures IASS-IACM 2008: Spanning Nano to Mega. Ithaca: J.F. Abel and J.R Cooke, 2008.
[98] PIEGL, L. y TILLER, W. The NURBS book. Berlin: Springer, 1997.

propósito, se calculó la distancia entre el cuarto punto y el plano descrito por los otros tres. Luego, la meta era la minimización de dicha distancia. La coordenada z de cada punto se asumió como las variables de diseño para el proceso de optimización.

El ejemplo de aplicación presentado en el trabajo de Sassone y Pugnale, es un cascarón reticular que cubre una galería en forma de cruz. La forma global que define la cubierta de vidrio fue pre-definida por los arquitectos, debido a la intención que tenían de enfatizar el espacio central y de lograr un efecto de máxima transparencia. Por consiguiente, el procedimiento es considerado por Sassone y Pugnale como un proceso de mejora de forma, en lugar de considerarlo como uno de búsqueda de forma. El algoritmo corrió durante 4500 generaciones, dando como resultado una forma de cubierta mucho más suavizada y construible (se garantizó la planaridad de cada celda que conforma la cubierta de vidrio), en comparación con la forma aleatoria inicial.

Figura 15. Cascarón reticular de vidrio; forma aleatoria; forma resultante, después de 4500 iteraciones



Fuente: SASSONE, M. y PUGNALE, A. Optimal design of glass grid shells with quadrilateral elements by means of a genetic algorithm. En: 6TH INTERNATIONAL CONFERENCE ON COMPUTATION OF SHELL AND SPATIAL STRUCTURES. Proceedings of the 6th International Conference on Computation of Shell and Spatial Structures IASS-IACM 2008: Spanning Nano to Mega. Ithaca: J.F. Abel and J.R Cooke, 2008.

Siguiendo con la investigación llevada a cabo por Sassone y Pugnale, está vez en conjunto con Méndez[99], se presenta un proceso computacional morfogenético, aplicado al diseño y optimización de una cubierta de un teatro. La estructura era tipo cascarón, con un concepto arquitectónico basado en la técnica de placas plegadas. Las dimensiones de cada placa eran distintas, configurando una estructura irregular. Adicionalmente, cada placa no era plana, sino suavemente curveada con el fin de formar paraboloides hiperbólicos.

Empezando con el concepto geométrico de la cubierta, se aplicó un proceso de optimización y de búsqueda de forma (morfogenético) para lograr un determinado rendimiento acústico (en términos de la distribución del nivel de presión acústica dentro del teatro). Para describir la superficie de placas plegadas se utilizó una representación NURBS[100], en la cual se asumieron como variables de diseño la posición de los puntos de control.

El rendimiento acústico se evaluó por medio de un algoritmo capaz de calcular el nivel de la presión acústica en todos los puntos del teatro (se introdujo un acercamiento desarrollado por Krokstad[101]), considerando la reflexión del sonido en la cubierta. De esta manera, la meta del proceso de optimización (por medio de un GA) fue planteada como la búsqueda de un nivel de presión acústica uniforme dentro del teatro. En dicho proceso, es posible controlar interactivamente la evolución de la forma cuando se está buscando una solución óptima, con el fin de evitar la convergencia del algoritmo hacia configuraciones geométricas indeseables.

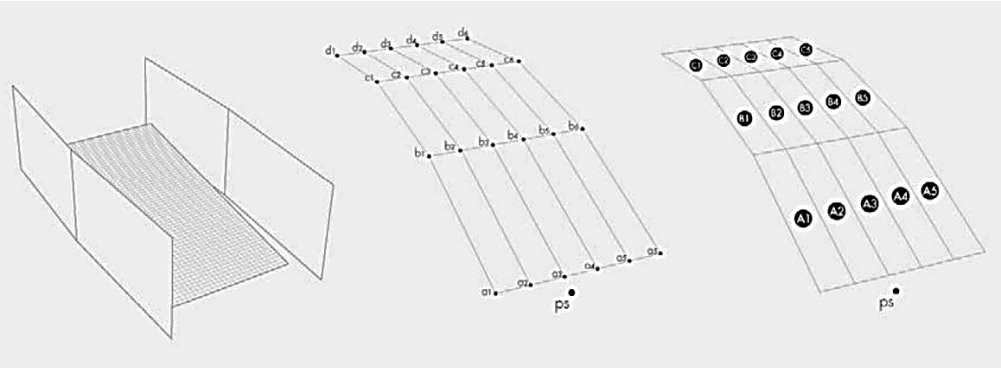
---

[99] SASSONE, M., MÉNDEZ, T. y PUGNALE, A. On the interaction between architecture and engineering: the acoustic optimization of a reinforced concrete shell. En: 6TH INTERNATIONAL CONFERENCE ON COMPUTATION OF SHELL AND SPATIAL STRUCTURES. Proceedings of the 6th International Conference on Computation of Shell and Spatial Structures IASS-IACM 2008: Spanning Nano to Mega. Ithaca: J.F. Abel and J.R Cooke, 2008.

[100] PIEGL, L. y TILLER, W. The NURBS book. Berlin: Springer, 1997.

[101] KROKSTAD, A., STROM, S. y SORSDAL, S. Calculation of the acoustical room response by the use of a ray tracing technique. En: Journal of Sound and Vibrations. No. 8 (1968); p. 118-125.

Figura 16. Área de las sillas y muros laterales; puntos de control de la superficie de cubierta; elementos de la superficie de cubierta



Fuente: SASSONE, M., MÉNDEZ, T. y PUGNALE, A. On the interaction between architecture and engineering: the acoustic optimization of a reinforced concrete shell. En: 6TH INTERNATIONAL CONFERENCE ON COMPUTATION OF SHELL AND SPATIAL STRUCTURES. Proceedings of the 6th International Conference on Computation of Shell and Spatial Structures IASS-IACM 2008: Spanning Nano to Mega. Ithaca: J.F. Abel and J.R Cooke, 2008.

El GA adoptado para el proceso morfogenético y de optimización, en su búsqueda de nuevas, más eficientes, y mejores soluciones, asume las variables codificadas de diseño como números reales, utiliza el operador clásico de ruleta para seleccionar los mejores individuos, un operador de crossover bi-dimensional, y un operador aleatorio de mutación.

Los autores de la investigación presentada anteriormente plantean que el siguiente paso en este campo de estudio debería ser la aplicación de estrategias computacionales distintas, con el fin de mejorar la eficiencia y la robustez del algoritmo. Ellos también proponen que otro paso importante sería el de combinar

la optimización acústica y estructural, al aplicar el acercamiento presentado en Pugnale et al.[102], y por medio de un proceso de optimización multi-objetiva.

Para concluir esta sección de ejemplos acerca de la aplicación de TCMOIED en el diseño estructural y arquitectónico, se presenta la investigación de Byrne et al. [103]. Ellos trabajaron en el diseño arquitectónico evolutivo, desarrollando una combinación entre el análisis estructural y algún criterio multi-objetivo.

En su trabajo, se menciona que por medio de gramáticas de diseño y de una función objetivo interactiva, la Evolución Gramatical (GE)[104] es capaz de crear diseños sorprendentes e innovadores. El ejemplo de prueba llevado a cabo pretendía probar si una búsqueda evolutiva era capaz de generar diseños que minimizaran los esfuerzos y la cantidad de material utilizado en una estructura.

En el trabajo de Byrne et al.[105], el acercamiento formal arquitectónico es combinado con las restricciones de ingeniería en el desarrollo de un puente, dando como resultado dos ventajas importantes: la posibilidad de optimizar diseños conceptuales con el fin de mejorar su rendimiento mientras se reduce la cantidad de material utilizado, y la posibilidad de agrupar los diseños al asignarle un valor de idoneidad a cada solución. De esta forma, el diseñador puede seleccionar las zonas del espacio de búsqueda que son más interesantes desde el punto de vista personal y, basado en esto, acelerar la convergencia del algoritmo hacia los diseños estéticamente placenteros.

---

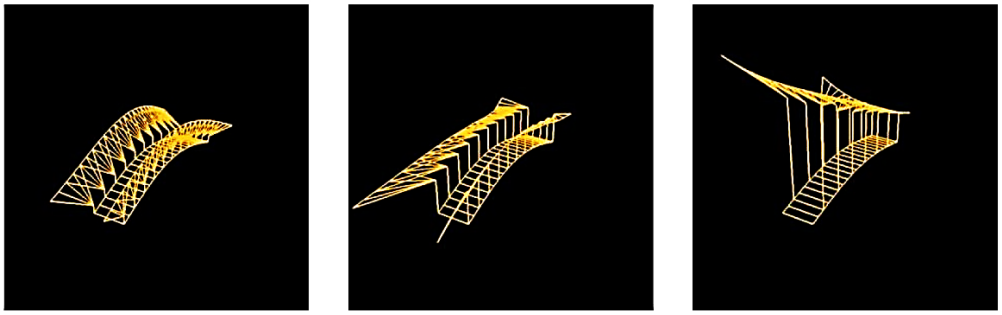
[102] PUGNALE, A. y SASSONE, M. Morphogenesis and structural optimization of shell structures with the aid of a Genetic Algorithm. En: Journal of the International Association for Shell and Spatial Structures. No. 48 (2007); p. 161-166.

[103] BYRNE, J., FENTON, M., HEMBERG, E., MCDERMOTT, J., O'NEIL, M., SHOTTON, E. y NALLY, C. Combining structural analysis and multi-objective criteria for evolutionary architectural design. En: Applications of Evolutionary Computation. No. 1 (2011); p. 204-213.

[104] O'NEIL, M. Automatic Programming in an Arbitrary Language: Evolving Programs with Grammatical Evolution. Limerick, 2001. Tesis doctoral. University of Limerick.

[105] BYRNE, J., FENTON, M., HEMBERG, E., MCDERMOTT, J., O'NEIL, M., SHOTTON, E. y NALLY, C., Op. cit.

Figura 17. Ejemplos de puentes con distintos criterios estéticos



Fuente: BYRNE, J., FENTON, M., HEMBERG, E., MCDERMOTT, J., O'NEIL, M., SHOTTON, E. y NALLY, C. Combining structural analysis and multi-objective criteria for evolutionary architectural design. En: Applications of Evolutionary Computation. No. 1 (2011); p. 204-213.

Para llevar a cabo el procedimiento anterior, el sistema propuesto está compuesto por cuatro partes: un EA, una gramática de diseño (74), un software de análisis estructural, y una función multi-objetiva para evaluar la idoneidad de cada solución (basada en una modificación del NSGA como mecanismo de selección). Como resultado, después de 50 generaciones, se obtuvo una reducción del 43% en el material utilizado y una del 41% en el esfuerzo máximo promedio de los elementos estructurales. Estos resultados hacen del procedimiento, un mecanismo de optimización efectivo. Adicionalmente, la intención de su implementación va más allá de la optimización, al permitirle al arquitecto evolucionar interactivamente diseños estéticamente placenteros, y crear de esta manera una Computación Evolutiva Interactiva (IEC)[106] (75).

[106] TAKAGI, H. Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. En: Proc. of the IEEE. No. 89 (2001); p. 1275-1296.

1.2. COMPUTACIÓN MULTI-OBJETIVA META-HEURÍSTICA (CMOMH)

Los EAs presentados en la sección anterior no son la única técnica de búsqueda que se ha utilizado en el manejo de problemas de optimización multi-objetiva. También existe otro campo computacional, inspirado en el comportamiento social y/o procesos físicos, en el cual están incluidas técnicas de optimización exitosas y prometedoras (i.e. Taboo Search, Simulated Annealing, PSO, etc.). Por este motivo, la posibilidad de extender estos acercamientos al campo multi-objetivo ha tomado gran interés en la última década de desarrollo.

Adicionalmente, en varios ejemplos de aplicación de EAs en problemas de diseño, se plantea el uso de las técnicas alternativas meta-heurísticas, presentadas en esta sección, como uno de los pasos siguientes en el presente campo de investigación. Algunos autores argumentan que este tipo de técnicas podría mejorar el rendimiento y la eficiencia del proceso de diseño y optimización. Un ejemplo puede encontrarse en el trabajo de Sassone et al.[107].

1.2.1. Técnicas de Computación Multi-Objetiva Meta-Heurística (TCMOMH)

Simulated Annealing:

Kirkpatrick et al.[108] propusieron un algoritmo llamado Simulated Annealing (SA), el cual está basado en las similitudes entre el proceso iterativo de mejora de soluciones (común en procesos de optimización) y el proceso de reconfiguración microscópica que ocurre cuando un material caliente se enfría. En este caso, la función objetivo del problema de optimización hace el rol de la energía, buscando

[107] SASSONE, M. y PUGNALE, A. Optimal design of glass grid shells with quadrilateral elements by means of a genetic algorithm. En: 6TH INTERNATIONAL CONFERENCE ON COMPUTATION OF SHELL AND SPATIAL STRUCTURES. Proceedings of the 6th International Conference on Computation of Shell and Spatial Structures IASS-IACM 2008: Spanning Nano to Mega. Ithaca: J.F. Abel and J.R Cooke, 2008.

[108] KIRKPATRICK, S., GELATT, C.D. y VECCHI, M.P. Optimization by Simulated Annealing. En: Science. No. 220 (1983); p. 671-680.

su valor mínimo. El procedimiento desarrollado por Metropolis[109] provee una generalización del proceso iterativo de optimización, permitiendo la introducción de pasos controlados (cambios suaves de temperatura) durante la búsqueda de una mejor solución.

En síntesis, dicho proceso se podría resumir de la siguiente manera: en cada paso del algoritmo, se le asigna un desplazamiento pequeño y aleatorio a cada átomo (punto de solución), calculando el cambio resultante de energía, $\Delta \mathrm{E}$. Si $\Delta \mathrm{E} \leq 0$, el desplazamiento asignado es aceptado, y la configuración con el átomo (punto de solución) desplazado es usada como el punto de partida del siguiente paso. El caso en el cual $\Delta \mathrm{E} > 0$, es tratado probabilísticamente. La probabilidad de aceptar la nueva configuración está dada por:

$$P(\Delta \mathrm{E}) = \exp(-\Delta \mathrm{E}/k_B T) \quad (1)$$

Siendo ΔE la diferencia en el valor de la función objetivo evaluada en el nuevo punto y en el anterior; k_B la constante de Boltzman que, generalmente por simplicidad, toma el valor de 1; y T la temperatura actual del proceso (irá disminuyendo a medida que avanza la optimización).

Esta probabilidad es comparada con un número aleatorio entre 0 y 1. Si dicho número es menor que $P(\Delta E)$, se acepta la nueva ubicación (desplazamiento asignado); de lo contrario, la configuración original es utilizada para empezar el paso siguiente. La elección de $P(\Delta E)$ trae como consecuencia, una evolución del sistema en una distribución de Boltzman.

[109] METROPOLIS, N., ROSENBLUTH, A., ROSENBLUTH, M., TELLER, A. y TELLER, E. Equation of State Calculations by Fast Computing Machines. En: Journal of Chemical Physics. No. 21 (1953); p. 1087-1092.

El proceso anterior se repite varias veces, simulando el movimiento térmico de átomos en contacto térmico con un foco calórico a una temperatura T, hasta alcanzar el criterio de parada.

Koulamas et al.[110] enfocaron su investigación en la aplicación de un SA en el manejo de procesos de administración, producción y operación, así como en el campo de investigación de operaciones. En su trabajo, se encuentran discusiones sobre problemas tradicionales como el *single machine*, almacenamiento de flujos y planeación de trabajo, el problema del viajero, etc. De igual forma, se presentan algunos problemas no-tradicionales con el fin de incluir el campo de imágenes de color y particiones de números. Una de las conclusiones de este trabajo es que el SA es una herramienta efectiva para resolver muchos problemas del campo de investigación de operaciones, con una precisión que puede ser controlada por el usuario, en términos del número de iteraciones y de las funciones de vecindad.

Por otro lado, Henderson et al.[111] plantean que el algoritmo SA ha emergido en la ingeniería como una herramienta alternativa para resolver problemas de difícil resolución por medio de técnicas matemáticas convencionales, incluidas en el campo de la programación. De todas formas, ellos también mencionan una desventaja significativa correspondiente a la solución de problemas que presentan sistemas altamente complejos. Al intentar resolver dichos problemas con el SA, es posible que el tiempo computacional sea bastante largo, lo que se refleja en un mayor esfuerzo por parte del procesador, en comparación con algunos de los algoritmos convencionales.

[110] KOULAMAS, C., ANTONY, S.R. y JAEN, R. A survey of simulated annealing applications to operations-research problems. En: OMEGA - International Journal of Management Science. No. 22 (1994); p. 41-56.
[111] HENDERSON, D., JACOBSON, S.H. y JOHNSON, W. The theory and practice of simulated annealing. En: GLOVER, F. y KOCHENBERGER, G.A. Handbook of Metaheuristics. Kluwer Academic Publishers, 2003. p. 287-319.

Para extender el SA al campo multi-objetivo, se presenta el trabajo de Serafini[112]. Su propuesta es utilizar un acercamiento de vector objetivo (target vector) enfocado en la solución de un problema que presente dos objetivos (se proponen varias reglas posibles de transición). De esta manera, una solución x' es generada en el vecindario de la solución actual x. Si $f(x')$ es no-dominada con respecto a $f(x)$, x'es aceptada como el estado actual y, se actualiza un grupo de soluciones no-dominadas.

El grupo, o archivo de soluciones no-dominadas, constituye la "memoria" del acercamiento propuesto y permite la generación de varios elementos del grupo óptimo de Pareto en una sola corrida. Sin embargo vale la pena resaltar que, en este caso, se utiliza únicamente una no-dominancia local para llenar el archivo de soluciones, por ende se requiere un procedimiento posterior de filtro para reducir el número de soluciones no-dominadas que se le presentan al diseñador en la fase final del proceso.

El aspecto principal del procedimiento expuesto anteriormente, está enfocado en determinar la manera de calcular la probabilidad de aceptar a un individuo x' donde $f(x')$ es dominada con respecto a $f(x)$. Para esto, Serafini propuso el uso de la norma $L_\infty - Tchebycheff$:

$$P(x', x, T) = min\left\{1, e^{max_j\left\{\lambda_j\left(f_j(x) - f_j(x')\right)/T\right\}}\right\} \quad (2)$$

En donde $P(x', x, T)$ es la probabilidad de aceptar a x', dado x, y la temperatura T. Los pesos λ_j son inicializados con uno, y modificados durante el proceso de búsqueda. Serafini propuso muchas otras reglas, incluyendo *cone ordering*, la cual es similar al orden lexicográfico.

[112] SERAFINI, P. Simulated Annealing for Multiple Objective Optimization Problems. En: TZENG, G. et al. Multiple Criteria Decision Making: Expand and Enrich the Domains of Thinking and Application. Berlin: Springer Verlag, 1994. p. 283-292.

Particle Swarm Optimization:

Otra técnica meta-heurística es el algoritmo Particle Swarm Optimization (PSO), desarrollado por Kennedy et al.[113]. El algoritmo está inspirado en la coreografía de una bandada (o enjambre) de pájaros (población o grupo) que tiene el propósito de encontrar comida. Este acercamiento puede verse como un algoritmo comportamental distribuido, que desarrolla una búsqueda multi-dimensional (en su versión más general). El proceso de optimización es guiado por las mejores soluciones que cada partícula ha encontrado y por la mejor solución que todo el enjambre ha identificado.

A continuación se resume el procedimiento del algoritmo PSO, incluyendo sus conceptos básicos. Como primer paso, se debe generar un vector de posiciones iniciales y otro de velocidades (de vuelo) iniciales, para cada partícula. Dichos vectores pueden ser definidos por el usuario o creados aleatoriamente. Con el vector de posiciones iniciales, se evalúa la función objetivo para cada uno de los puntos.

Con estos valores, se inicia un vector que almacena la mejor posición (correspondiente al mejor valor de la función objetivo) que ha tenido cada partícula y otro vector en donde se guarda la mejor posición global que ha conseguido el enjambre de partículas. Con estos datos se actualiza la velocidad con el fin de generar una nueva posición para cada partícula. Lo anterior se lleva a cabo por medio de las siguientes ecuaciones[114]:

$$v_{k+1}^{i} = w v_{k}^{i} + c_1 r_1 \left(\frac{p_k^i - x_k^i}{\Delta t} \right) + c_2 r_2 \left(\frac{p_k^g - x_k^i}{\Delta t} \right) \quad (3)$$

[113] KENNEDY, J. y EBERHART, R. Particle Swarm Optimization. En: IEEE INTERNATIONAL CONFERENCE ON NEURAL NETWORKS. Proceedings of the 1995 IEEE International Conference on Neural Networks. Piscataway: IEEE Service Center, 1995. p. 1942-1948.

[114] PEREZ, R.E. y BEHDINAN, K. Particle swarm approach for structural design optimization. En: Computers & Structures. No. 85 (2007); p. 1579-1588.

$$x_{k+1}^{i} = x_{k}^{i} + v_{k+1}^{i}\Delta t \quad (4)$$

Siendo v_{k+1}^{i} la velocidad de la partícula i para la iteración $k+1$; w el peso inercial; v_{k}^{i} la velocidad de la partícula i para la iteración k; c_1 y c_2 el parámetro cognitivo (confianza que la partícula tiene en sí misma) y el parámetro social (confianza que la partícula tiene en el enjambre), respectivamente; r_1 y r_2 números aleatorios entre 0 y 1; p_k^i y p_k^g es la mejor posición encontrada por la partícula i hasta la iteración k y, la mejor posición global encontrada por todo el enjambre de partículas hasta la iteración k, respectivamente; x_k^i es la posición de la partícula i en la iteración k; x_{k+1}^i es la posición de la partícula i en la iteración $k+1$; y Δt es un paso de tiempo que, generalmente, se asume igual a 1.

Este proceso de actualización de velocidades, posiciones, y vectores de mejores posiciones particulares y globales, se actualiza hasta que se alcanza el criterio de parada que puede ser un número determinado de iteraciones, una diferencia mínima entre las soluciones, determinada por el usuario, etc.

Como ejemplo de aplicación, Hu et al.[115] presentaron un PSO modificado para problemas de optimización con restricciones, en el campo ingenieril. El algoritmo empieza con un grupo de soluciones viables, y utiliza una función de factibilidad con el fin de probar si las nuevas soluciones exploradas satisfacen las restricciones. De esta manera, todas las partículas almacenan en su memoria únicamente soluciones viables. Como ejemplos de aplicación, se presentaron algunos problemas de diseño en ingeniería como: el diseño de un tanque de presión, el diseño de una viga metálica soldada, la minimización del peso de un resorte a compresión, y el problema de optimización no-lineal de Himmelblau. De

[115] HU, X., EBERHART, R.C. y SHI, Y. Engineering Optimization with Particle Swarm. En: IEEE SWARM INTELLIGENCE SYMPOSIUM. Proceedings of the 2003 IEEE Swarm Intelligence Symposium. Indianapolis: IEEE Service Center, 2003. p. 53-57.

los resultados de estos problemas de aplicación, es posible concluir que el algoritmo propuesto es un acercamiento general y eficiente para resolver varios problemas ingenieriles no-lineales de optimización, con restricciones de desigualdad.

Toscano Pulido et al.[116] hicieron una contribución significativa para manejar problemas multi-objetivos con el algoritmo PSO. Ellos realizaron un estudio de los mecanismos básicos de los optimizadores existentes multi-objetivos basados en el PSO (MOPSOs), con el fin de proponer nuevos mecanismos que produjeran un MOPSO mucho más eficiente. De esta manera se logró un algoritmo que sólo lleva a cabo 2000 evaluaciones de función objetivo al solucionar problemas de prueba con hasta 30 variables de decisión. Los autores argumentan que este es el menor número de evaluaciones de función objetivo jamás reportado por algún MOPSO en la literatura especializada.

Con el fin de mejorar el rendimiento y la eficiencia de un MOPSO que habían propuesto anteriormente los autores, se realizaron las siguientes modificaciones:

1. Incorporación de un mecanismo para distribuir las soluciones no-dominadas obtenidas por el algoritmo. Se proponen dos mecanismos para reducir el número de soluciones no-dominadas generadas por un MOEA:

   - Una retícula adaptable[117]:

     Es un espacio formado por híper-cubos, los cuales tienen tantos componentes como funciones objetivo tenga el problema a resolver. Cada híper-cubo puede ser interpretado como una región geográfica que contiene

[116] TOSCANO-PULIDO, G., SANTANA-QUINTERO, L.V. y COELLO COELLO, C.A. EMOPSO: A Multi-Objective Particle Swarm Optimizer with Emphasis on Efficiency. En: EMO 2007. Proceedings of EMO 2007. Heidelberg: Springer, 2007. p. 272-285.
[117] KNOWLES, J.D. y CORNE, D.W. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. En: Evolutionary Computation. No. 8 (2000); p. 149-172.

n número de individuos. De esta manera, la retícula adaptable permite almacenar soluciones no-dominadas y redistribuirlas cuando se alcanza su capacidad máxima.

- Una forma relajada de la dominancia de Pareto (ε-dominance)[118]:
  El llamado grupo ε-Pareto es una estrategia basada en un archivo que mantiene un sub-grupo de soluciones generadas, garantizando la convergencia y la diversidad según una definición correcta del parámetro ε, el cual define la resolución de la retícula que se va a adoptar para la población secundaria. La idea general de este mecanismo es dividir el espacio de la función objetivo entre cajas de tamaño ε. Cada caja puede ser interpretada como una región geográfica que contiene una única solución. El acercamiento acepta una nueva solución dentro del grupo ε-Pareto si: es la única solución de la caja a la cual pertenece, si domina a otras soluciones, o si compite contra otras soluciones no-dominadas dentro de la caja, pero está más cerca del vértice de origen de esta. Con el fin de lograr el mejor rendimiento, es necesario proveer el tamaño de la caja (el parámetro ε) el cual depende del problema y, normalmente no es conocido antes de ejecutar un MOEA (posible desventaja).

Adicionalmente, se propone un mecanismo llamado *distribución de híper-plano* para distribuir soluciones no-dominadas. La idea central es llevar a cabo una buena distribución del espacio híper-plano definido por el mínimo (asumiendo minimización) de los objetivos, y utilizar dicha distribución para seleccionar un sub-grupo representativo de todo el grupo de soluciones no-dominadas. El algoritmo funciona de la siguiente manera: primero, requiere como entrada, un grupo de soluciones no-dominadas y la cantidad n de soluciones finales deseadas. Luego, el algoritmo selecciona aquellas soluciones que tienen el

[118] LAUMANNS, M., THIELE, L., DEB, K. y ZITZLER, E. Combining Convergence and Diversity in Evolutionary Multi-objective Optimization. En: Evolutionary Computation. No. 10 (2002); p. 263-282.

valor mínimo en cada objetivo. De esta manera se calcula un híper-plano entre todas las soluciones mínimas. Después, el algoritmo divide dicho espacio en $n-1$ regiones. Posteriormente, se calcula una línea perpendicular al híper-planoen el vértice de cada región. Por último, el algoritmo sólo acepta aquellas soluciones que estén más cercanas a cada línea.

2. Uso de un operador de turbulencia con el fin de evitar una convergencia prematura:

   Este operador consiste en una alteración de la velocidad de vuelo de una partícula. Esta modificación se lleva a cabo en todas las dimensiones (i.e., en todas las variables de decisión), de tal forma que la partícula puede moverse hacia una región completamente aislada. El operador de turbulencia actúa basado en un factor de probabilidad que considera la generación actual y el número total de iteraciones que se llevarán a cabo. La idea es tener una probabilidad de perturbación de vuelo mucho mayor al principio de la búsqueda, e irla reduciendo al progresar en las iteraciones. La turbulencia puede ser vista como un operador de mutación y está basada en la siguiente expresión:

$$prob_{turbulencia} = temp^{1.7} - 2.0 * temp + 1.0 \quad (5)$$

   siendo,

$$temp = {iteración\ actual}/{total\ de\ iteraciones} \quad (6)$$

   Los valores utilizados en esta expresión fueron deducidos empíricamente luego de una serie de experimentos que se exponen en[119].

[119] TOSCANO-PULIDO, G. On the Use of Self-Adaptation and Elitism for Multiobjective Particle Swarm Optimization. Mexico, 2005. Tesis doctoral. CINVESTAV-IPN. Department of Electrical Engineering. Computer Science Section.

3. Desarrollo de un experimento para fijar el número de sub-swarms (sub-enjambres) a adoptar:

   Esto se lleva a cabo debido a que la selección apropiada de líderes es esencial para el buen rendimiento de un MOPSO, si una partícula escoge un líder inapropiado (i.e., un líder que está muy lejos en el espacio de búsqueda), casi todo su vuelo será infructuoso ya que la partícula no visitará regiones promisorias del espacio de búsqueda. Por este motivo, se propone utilizar no sólo uno sino varios sub-swarms[120]. La cantidad correcta de sub-swarms va a depender del problema, luego se debe seleccionar de forma apropiada dicho número.

4. Incorporación de un mecanismo para manejar restricciones:

   Se incorpora el mecanismo propuesto en[121]. Este acercamiento no requiere ningún parámetro definido por el usuario y lleva a cabo menos evaluaciones de las funciones objetivo que cualquier otro acercamiento con los que se comparó, obteniendo resultados similares.

5. Desarrollo de un estudio empírico de la influencia de los parámetros C_1, C_2 y w, y planteamiento de una metodología simple para auto-adaptar dichos parámetros:

   Se llevó a cabo un minucioso análisis de los parámetros (similar a un análisis de varianza), con el fin de encontrar la mejor configuración posible de dichos parámetros (considerando el grupo de funciones de prueba adoptado). Para analizar el impacto de los parámetros en el acercamiento propuesto por los

[120] TOSCANO-PULIDO, G. y COELLO COELLO, C.A. Using Clustering Techniques to Improve the Performance of a Particle Swarm Optimizer. En: GENETIC AND EVOLUTIONARY COMPUTATION-GECCO 2004. Proceedings of the Genetic and Evolutionary Computation Conference. Seattle: Springer-Verlag, 2004. p. 225-237.

[121] TOSCANO-PULIDO, G. y COELLO COELLO, C.A. A Constraint-Handling Mechanism for Particle Swarm Optimization. En: CONGRESS ON EVOLUTIONARY COMPUTATION 2004. Proceedings of the Congress on Evolutionary Computation 2004 (CEC'2004). Piscataway: IEEE Service Center, 2004. p. 1396-1403.

autores, se consideraron varias configuraciones y se desarrolló un exhaustivo análisis. Las configuraciones que se adoptaron fueron:

$$w = \{0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0\}$$
$$C_1 = \{1.0,1.2,1.4,1.6,1.8,2.0,2.2,2.4,2.6,2.8,3.0\}$$
$$C_2 = \{1.0,1.2,1.4,1.6,1.8,2.0,2.2,2.4,2.6,2.8,3.0\}$$

Para todos los experimentos se adoptaron 40 partículas. Sin embargo, se analizaron tres escenarios distintos:

- Experimento 1: uso de una población inicial generada aleatoriamente y un número bajo de evaluaciones de función objetivo ($G_{max} = 25$, lo que da un total de 1000 evaluaciones de función objetivo).

- Experimento 2: uso de una buena aproximación del frente de Pareto para la población inicial. En todos los experimentos llevados a cabo en este caso, la misma aproximación se introdujo como población inicial en cada algoritmo. En este caso solo se llevaron a cabo 600 evaluaciones de función objetivo ($G_{max} = 15$) ya que el interés estaba en analizar la capacidad (o posibles dificultades) del algoritmo propuesto para alcanzar el verdadero frente de Pareto de un problema, cuando una buena (y suficientemente cercana) aproximación se ha producido.

- Experimento 3: uso de un gran número de evaluaciones de función objetivo ($G_{max} = 250$, el cual da un total de 10000 evaluaciones de función objetivo) con el fin de evaluar el rendimiento del acercamiento en un largo término.

Se adoptaron 11 funciones de prueba para el experimento 1 y para el 2. Debido al gran tiempo requerido para cada corrida, sólo se adoptaron 6 funciones de prueba para el experimento 3. Como se necesitaba evaluar el

rendimiento en cada caso, se escogió el inverso de la distancia generacional como medida, ya que este puede medir qué tan cerca se está del frente de Pareto verdadero y, la dispersión de las soluciones. Después de combinar los resultados de los tres experimentos, se concluyó lo siguiente:

- El mejor rango para C_1 y C_2 es entre 1.2 y 2.0. Dentro de este rango, se puede decir que $C_1 = C_2 = 1.4$ provee el mejor resultado general.

- Los mejores valores para w son aquellos menores o iguales a 0.5.

- De todas formas, los valores anteriores no producen el mejor rendimiento en todos los casos. Esto es lo que precisamente motivó a los autores a proponer, en su acercamiento, un mecanismo para que los parámetros se auto-adaptaran. De esta manera, el algoritmo automáticamente ajusta los parámetros de acuerdo a las características del espacio de búsqueda que está siendo explorado.

6. Mecanismo de auto-adaptación:
   Se propone el uso de un tradicional mecanismo de selección proporcional para seleccionar la combinación más apropiada de los valores de los parámetros a adoptar. Se adoptó la selección de ruleta para dicho propósito. En vez de calcular la idoneidad de cada combinación de valores, se cuenta el número de soluciones no-dominadas generadas por cada combinación.

Luego de implementar las modificaciones anteriores, se llevaron a cabo problemas de prueba para poder realizar una comparación de resultados. Con el fin de permitir una evaluación cuantitativa de los resultados, se adoptaron las siguientes

medidas de rendimiento: la Distancia Generacional Inversa[122], Two Set Coverage[123] y Spread Metric[124]. Los resultados se compararon con respecto al NSGA-II utilizando 12 funciones estándar de prueba, 7 de ellas con restricciones y 5 sin restricciones. Los resultados obtenidos por EMOPSO fueron superiores a los generados por NSGA-II.

Al concluir el trabajo, los autores encontraron que el uso de sub-swarms promueve la búsqueda local como un comportamiento emergente en EMOPSO. Consecuentemente, el rendimiento de este acercamiento fue mejorado por el uso de sub-swarms, particularmente en presencia de frentes de Pareto conectados. También encontraron que el uso de un mecanismo de perturbación en EMOPSO es crítico para controlar su alta presión de selección así como para evitar una convergencia prematura. Por último, se recalca la dificultad de encontrar valores fijos para los tres parámetros más significativos del acercamiento propuesto (w, C_1 y C_2), siendo este el argumento principal del diseño del mecanismo de auto-adaptación para dichos parámetros.

Taboo Search:

Glover[125] desarrolló un acercamiento meta-heurístico adicional, llamado Taboo Search (TS). Esta técnica está basada en una estrategia de prohibición de ciertos movimientos, la cual previene un comportamiento cíclico en la búsqueda de mejores soluciones. Particularmente, el TS procede de acuerdo con la suposición que la selección de un movimiento pobre, por accidente o diseño, no tiene ningún valor, excepto si tiene el propósito de evitar un camino ya explorado. Destinado

[122] VELDHUIZEN, D.A.V. Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. Ohio, 1999. Reporte técnico. Air Force Institute of Technology. Graduate School of Engineering. Department of Electrical and Computer Engineering.

[123] ZITZLER, E., DEB, K. y THIELE, L. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. En: Evolutionary Computation. No. 8 (2000); p. 173-195.

[124] DEB, K., PRATAP, A., AGARWAL, S. y MEYARIVAN, T. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. En: IEEE Transactions on Evolutionary Computation. No. 6 (2002); p. 182-197.

[125] GLOVER, F. Future paths for integer programming and links to artificial intelligence. En: Comput. & Ops. Res. No. 13 (1986); p. 533-549.

por esta excepción, el método busca el mejor movimiento posible en cada paso. Con el fin de no seguir el rastro de un camino que ya ha sido examinado, el procedimiento almacena información acerca de los movimientos más recientes, utilizando una o más listas tabú.

Kargahi et al.[126] estudiaron un tipo de técnicas de búsqueda para problemas discretos de optimización, y algunos métodos heurísticos de búsqueda, incluyendo la habilidad de estos en optimización estructural. En su trabajo, se desarrolló un software que utiliza el algoritmo TS para la minimización del peso de estructuras bi-dimensionales tipo marco. El software conduce la búsqueda, el análisis estructural, y el diseño estructural en un procedimiento iterativo. Por ejemplo, el algoritmo fue aplicado en la optimización de tres marcos metálicos, resistentes a momentos. El primero tenía tres pisos y tres luces, el segundo tenía nueve pisos y cinco luces, y el tercero tenía veinte pisos y cinco luces. Por medio de estos ejemplos, el software demostró su capacidad para optimizar el peso de los marcos en un tiempo razonable, sin requerir la interfaz ingenieril durante el proceso de búsqueda. El resultado fue una disminución promedio del 23.4% en el peso, en comparación con el diseño original de los marcos.

Un acercamiento multi-objetivo basado en el Taboo Search, el Multi-Objective Taboo Search (MOTS), fue desarrollado por Hansen[127]. Esta propuesta trabaja con un grupo de soluciones actuales, las cuales son optimizadas simultáneamente hacia el frente no-dominado. El objetivo es cubrir todo el frente con las soluciones actuales, y dirigir el proceso de optimización de cada solución de tal forma que la solución tienda a alejarse de las demás, mientras se conduce al frente no-

[126] KARGAHI, M., ANDERSON, J.C. y DESSOUKY, M.M. Structural weight optimization of frames using tabu search: Optimization procedure. En: Journal of Structural Engineering ASCE. No. 132 (2006); p. 1858-1868.

[127] HANSEN, M.P. Tabu Search for Multiobjective Optimization: MOTS. En: 13TH INTERNATIONAL CONFERENCE ON MULTIPLE CRITERIA DECISION MAKING. Proceedings of the 13th International Conference on Multiple Criteria Decision Making (MCDM'97). Cape Town: s.e., 1997.

dominado. Las soluciones toman su dirección aplicando un movimiento de acuerdo a una búsqueda heurística Tabú y, cada solución almacena su propia lista tabú. El procedimiento del algoritmo propuesto puede resumirse de la siguiente manera:

1. Cada solución actual se asigna como una solución viable aleatoria y se vacía la lista tabú.

2. El grupo actual de puntos no-dominados se vacía, un contador de iteración se resetea y el rango de los factores de igualación (π) se asigna a un vector unitario. Luego se inicia el ciclo, el cual permite, continuamente, que cada solución actual realice un movimiento hacia una solución vecina hasta que el criterio de parada se alcance.

3. Se determina el vector de pesos (λ) para el punto. Se desea asignar los pesos para que el punto se aleje de los demás, idealmente teniendo los puntos dispersados equidistantemente sobre la frontera. Por tanto, cada elemento en el vector de pesos se asigna de acuerdo a la proximidad de otros puntos para un determinado objetivo. De todas formas, sólo se compara un punto con los puntos de la solución actual para la cual es no-dominado. Entre más cerca esté de otro punto, más debe influenciar el vector de pesos. La proximidad se mide con una función de distancia (d) basada en algunas medidas en el espacio de la función objetivo y, utilizando el rango de igualación de pesos. La influencia es dada por una función (g) decreciente y positiva que evalúa la proximidad basada en la distancia. En la práctica, la función de proximidad g(d)=1/d, ha demostrado un funcionamiento correcto.

4. El procedimiento estándar de búsqueda tabú (Taboo Search) se utiliza para reemplazar una solución actual con la mejor solución vecina viable (generada por la función vecindario N), la cual puede alcanzarse desde la solución sin

violar la lista tabú. Cómo establecer una lista tabú es, como siempre, una pregunta abierta; en general el elemento tabú es un tipo de atributo (A) para el rango de las soluciones, con el fin de prevenir movimientos de retroceso. El mejor vecino es determinado por el producto escalar entre el vector de pesos y el vector de la función objetivo.

5. El nuevo punto es insertado en el grupo no-dominado si es no-dominado con respecto a este, y los puntos que ya están allí que se vuelven dominados, si es que hay alguno, se remueven. Si se desea, aquí también se puede grabar la solución misma. El rango de los factores de igualación se asigna de acuerdo a los rangos de los puntos en el grupo no-dominado y, por tanto, tal vez sea necesario actualizarlos. Utilizar los rangos de los puntos en el grupo no-dominado, es una sugerencia general en caso de que no esté disponible un conocimiento acerca de estos.

6. Se reemplaza una solución aleatoriamente seleccionada por otra, igualmente aleatoria, cada vez que se alcance el criterio de desviación, el cual, por el momento, puede ser después de un número fijo de incrementos en la variable del contador. Esto se hace con el fin de asegurar una desviación en el movimiento de los puntos sobre la frontera no-dominada, explorándola así por completo.

7. Finalmente, el contador de iteración se incrementa por 1, y se continúa con la siguiente de las soluciones actuales.

Ant Colony:

Una alternativa meta-heurística distinta fue planteada por Dorigo y Colomi[128], el acercamiento Ant Colony (AC). Este algoritmo está inspirado en colonias reales de

[128] DORIGO, M. y COLOMI, A. The Ant System: Optimization by a colony of cooperating agents. En: IEEE Transactions on Systems, Man, and Cybernetics. No. 26 (1996); p. 1-13.

hormigas: las hormigas depositan en el suelo una sustancia química llamada feromonas, la cual influencia el comportamiento de las demás, siguiendo el camino que tiene la mayor cantidad de esta sustancia. Las huellas de feromonas pueden verse como un mecanismo de comunicación indirecta entre los individuos. Desde el punto de vista de la ciencia computacional, el *Ant System* es un sistema multi-agente en donde interacciones de bajo nivel entre agentes individuales, resultan en un comportamiento complejo de la totalidad de la colonia.

El procedimiento del algoritmo propuesto podría resumirse de la siguiente manera: en el tiempo cero, toma lugar la fase de inicio del algoritmo, en la cual las hormigas se ubican en distintos *pueblos* y se asignan los valores iniciales de intensidad de rastreo $\tau_{ij}(0)$ sobre los bordes. El primer elemento de la lista tabú de cada hormiga es igual a su *pueblo* de inicio. Seguidamente, cada hormiga se mueve del *pueblo i* al *pueblo j*, escogiendo el *pueblo* al cual moverse por medio de una probabilidad que está en función de (con parámetros α y β, los cuales controlan la importancia relativa entre el rastreo y la visibilidad) dos medidas deseadas. La primera, el rastro $\tau_{ij}(t)$, brinda información acerca de cuántas hormigas han escogido ese mismo borde en el pasado (*i,j*); la segunda, la visibilidad η_{ij}, plantea que entre más cerca esté un *pueblo*, es más deseable. Analizando la fórmula para calcular la probabilidad, es evidente que si se asigna α=0, el nivel de rastreo no se considera, luego se obtendría un algoritmo estocástico ávido con múltiples puntos iniciales.

$$p_{ij}^{k}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum [\tau_{ik}(t)]^{\alpha} \cdot [\eta_{ik}]^{\beta}} & si\ j \in viable_k \\ 0 & de\ otra\ manera \end{cases} \quad (7)$$

donde $viable_k$ es igual al espacio solución menos las restricciones impuestas por la lista tabú.

Después de *n* iteraciones, todas las hormigas han completado un tour y su lista de tabú habrá sido llenada; en este punto, para cada hormiga *k* se calcula el valor de L_k (longitud del tour de la hormiga k) y se actualizan los valores de $\Delta\tau_{ij}^{k}$ (cantidad de sustancia de rastreo, por unidad de longitud, puesta sobre el borde (*i,j*) por la k-ésima hormiga entre el tiempo *t* y *t+n*, equivalente a las feromonas de las hormigas reales).

Si la hormiga *k-ésima* utiliza el borde (*i,j*) en su tour (entre el tiempo *t* y *t+1*):

$$\Delta\tau_{ij}^{k} = \frac{Q}{L_k} \tag{8}$$

donde Qes una constante. De otra manera:

$$\Delta\tau_{ij}^{k} = 0 \tag{9}$$

Adicionalmente, se guarda el camino más corto encontrado por las hormigas (i.e., min$L_k, k = 1, \dots, m$), y se vacían todas las listas de tabú. Este proceso se itera hasta que se alcance el máximo número de ciclos NC_{max}, definido por el usuario, o hasta que todas las hormigas realicen el mismo tour. A este último caso se le llama comportamiento de estancamiento, ya que denota una situación en la cual el algoritmo deja de buscar soluciones alternativas.

Kaveh y Talatahari[129] desarrollaron una propuesta interesante en la cual se propone un optimizador mejorado basado en el AC (Improved Ant Colony Optimizer, IACO), para resolver problemas de diseño con restricciones en ingeniería. IACO tiene la capacidad de manejar problemas discretos y continuos, utilizando un mecanismo de sub-optimización (SOM). Este mecanismo está

[129] KAVEH, A. y TALATAHARI, S. An improved ant colony optimization for constrained engineering design problems. En: Engineering Computations. No. 27 (2010); p. 155-182.

basado en el principio del método de elementos finitos, trabajando como una técnica de actualización del espacio de búsqueda.

Una extensión del AC para manejar problemas multi-objetivos es el Multi-Objective Ant Q (MOAQ)[130], desarrollado por Mariano y Morales. En este caso, la idea básica es tener una familia de agentes para cada objetivo. Cada familia trata de optimizar un objetivo considerando las soluciones encontradas para los otros objetivos. En el algoritmo presentado, todas las familias deben tener el mismo número de agentes, *m*. Como en Ant-Q, todos los agentes en una familia intentan encontrar una solución al mismo tiempo. Las *m* soluciones encontradas para un objetivo, en un ciclo, influencian los *m* puntos iniciales sucesivos de la siguiente familia (objetivo). Este proceso continua con todas las familias (objetivos).

Una vez se hayan evaluado todas las familias, se le concede una recompensa a las soluciones no-dominadas (a todos los agentes de cada familia que son responsables de dicha solución) que satisfacen todas las restricciones. Los resultados que violan restricciones son castigados, y el resto de las soluciones se ignoran. Todo el proceso se repite varias veces hasta que se encuentren únicamente soluciones no-dominadas (grupo de Pareto) que satisfagan todas las restricciones o, hasta que se alcance un número máximo predeterminado de iteraciones.

Distributed Reinforcement Learning:

Para concluir esta sección, se presenta el trabajo llevado a cabo por Mariano y Morales. Ellos propusieron un acercamiento multi-objetivo llamado Distributed

[130] MARIANO, C.E. y MORALES, E. MOAQ an Ant-Q Algorithm for Multiple Objective Optimization Problems. En: GENETIC AND EVOLUTIONARY COMPUTING CNFERENCE. Proceedings of the Genetic and Evolutionary Computing Cnference (GECCO 99). San Francisco: Morgan Kaufmann, 1999. p. 894-901.

Reinforcement Learning (MDQL)[131]. El algoritmo planteado presenta alguna similitud con el MOAQ pero, en vez de requerir una función heurística para la regla de selección, el algoritmo propuesto, Multi-objective Distibuted Q-Learning (MDQL), utiliza un algoritmo llamado Distributed Q-Learning (DQL).

La idea principal detrás de DQL es permitir la interacción entre varios agentes en un entorno común, cooperando para establecer una política óptima sobre *pueblos* y acciones que permita alcanzar una meta común. El mecanismo de comunicación entre los agentes es un *mapa* del entorno. Este *mapa* es construido con base en las actualizaciones de la función de valor adoptada para el problema. Dichas actualizaciones se llevan a cabo cada vez que un agente visita un *pueblo* y selecciona una acción (i.e., cuando deja un rastro para los otros agentes). Con el fin de decidir su siguiente movimiento, un agente considera aquellos rastros anteriormente dejados por otros agentes.

El concepto principal del MDQL es el siguiente: una familia de agentes es asignada a cada uno de los objetivos del problema que se quiere resolver. Las soluciones obtenidas por los agentes de una familia se comparan con las soluciones obtenidas por las otras familias. Para llevar a cabo esta comparación, se adopta un mecanismo de negociación.

El mecanismo propuesto pretende producir elementos del grupo real de Pareto, favoreciendo aquellas soluciones que son no-dominadas con respecto a todas las funciones objetivo. Los *pueblos* incluidos en tales soluciones son recompensados (esto se hace dinámicamente, utilizando una comparación de valores de refuerzo) y guardados para futuros usos. Se utiliza un archivo externo P para mantener los vectores de soluciones no-dominadas, generados durante la búsqueda.

[131] MARIANO, C.E. y MORALES, E. A New Distributed Reinforcement Learning Algorithm for Multiple Objective Optimization Problems. Mexico, 2000. Reporte técnico. Instituto Mexicano de Tecnología del Agua.

1.2.2. TCMOMH aplicadas al diseño estructural y arquitectónico

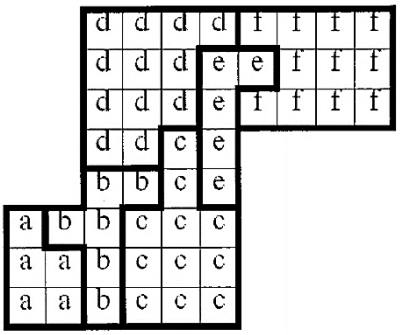
Una aplicación interesante hace referencia a la optimización del diseño de plantas arquitectónicas, desarrollado por Michalek et al.[132]. La meta de dicho procedimiento era encontrar configuraciones espaciales óptimas y viables, discretizando los espacios funcionales en unidades rectangulares con el fin de obtener una representación de diseño apropiada. La optimización fue enfocada en dos ramas principales, la geometría y la topología.

En la optimización geométrica, las variables eran parámetros que definían cada unidad: la ubicación del punto de referencia (coordenadas), las distancias desde este punto hasta cada muro que lo rodea, y el tamaño de las ventanas contenidas en las paredes exteriores.

Los objetivos eran: minimizar el costo por calefacción (depende de la forma del contorno de las unidades que conforman el edificio, el volumen, el área de la superficie, y el material), minimizar el costo por enfriamiento (depende del área neta de las ventanas y de los muros exteriores, y de la orientación de las unidades), minimizar el costo por iluminación (depende del tamaño de las ventanas, la geometría de los espacios, la orientación de las unidades, y el material), minimizar los espacios perdidos (haciendo referencia a los espacios del edificio que no están enfocados en las actividades de la vivienda; depende del área del contorno del edificio y del área utilizada como espacio para la vivienda), minimizar los pasillos de acceso (con el fin de juntar las unidades conectadas; depende de la definición de dichos pasillos), y finalmente, minimizar los vestíbulos (con el fin de proveer espacio extra para la vivienda, cuando sea posible; depende de la unidad funcional de los vestíbulos).

[132] MICHALEK, J.J., CHOUDHARY, R. y PAPALAMBROS, P.Y. Architectural Layout Design Optimization. En: Engineering Optimization. No. 34 (2002); p. 461-484.

Figura 18. Discretización del espacio en unidades funcionales (a,b,c,d,e,f)



Fuente: MICHALEK, J.J., CHOUDHARY, R. y PAPALAMBROS, P.Y. Architectural Layout Design Optimization. En: Engineering Optimization. No. 34 (2002); p. 461-484.

En la optimización geométrica, se plantearon las restricciones de la siguiente manera: forzar las unidades funcionales dentro del contorno principal del edificio, prevenir que dos unidades ocupen el mismo espacio, garantizar los accesos, forzar a unas determinadas unidades hacia el borde del contorno cuando una ventana o una puerta exterior sea necesaria, garantizar un área mínima y/o un mínimo y máximo en la relación largo/ancho, mantener un esquema estético deseado o prevenir espacios largos y angostos que seguramente no serán útiles, mantener los costos de construcción por debajo de algún valor, y asegurar un mínimo de iluminación natural para determinados espacios.

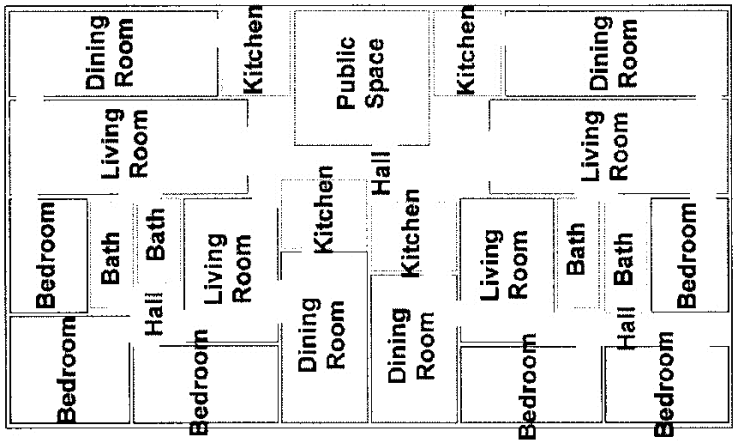
Para manejar la multiplicidad de objetivos, se utilizó un acercamiento de sumatoria con pesos, el cual combina las funciones objetivo en una sola función. Teniendo este planteamiento, el problema de optimización fue resuelto utilizando una implementación C del acercamiento Feasible Sequential Quadratic Programming (CFSQP)[133] pero, se descubrió que, en ocasiones, el algoritmo se quedaba

[133] ZHOU, J.L. y TITS, A.L. An SQP algorithm for finely discretized continuous minimax problems and other minimax problems with many objective functions. En: SIAM Journal of Optimization. No. 6 (1995); p. 461-487.

atrapado, en parte por algunas restricciones no suavizadas. Sin embargo, se encontró que CFSQP es bastante rápida y relativamente estable para problemas con dimensiones moderadas, convergiendo velozmente casi siempre.

Por estas razones, se implementó el uso de un SA y de un GA para que actuaran como métodos globales de optimización. De todas formas, ninguno de los dos pudo alcanzar soluciones viables de diseño para problemas con dimensiones moderadas. Luego, la solución final fue una estrategia de búsqueda híbrida, SA/SQP[134], la cual toma las ventajas del SA para la búsqueda global, y la eficiencia del SQP. Al hacer esto, el propósito de la estrategia fue el de generar soluciones óptimas locales con calidad global.

Figura 19. Diseño geométrico completo, utilizando el algoritmo CFSQP



Fuente: MICHALEK, J.J., CHOUDHARY, R. y PAPALAMBROS, P.Y. Architectural Layout Design Optimization. En: Engineering Optimization. No. 34 (2002); p. 461-484.

[134] MICHALEK, J.J., CHOUDHARY, R. y PAPALAMBROS, P.Y. Architectural Layout Design Optimization. En: Engineering Optimization. No. 34 (2002); p. 461-484.

En lo correspondiente a la optimización topológica, las variables incluidas fueron: la posición inicial de cada espacio sobre la retícula, y la conectividad entre cada espacio y todos los demás espacios y los muros exteriores.

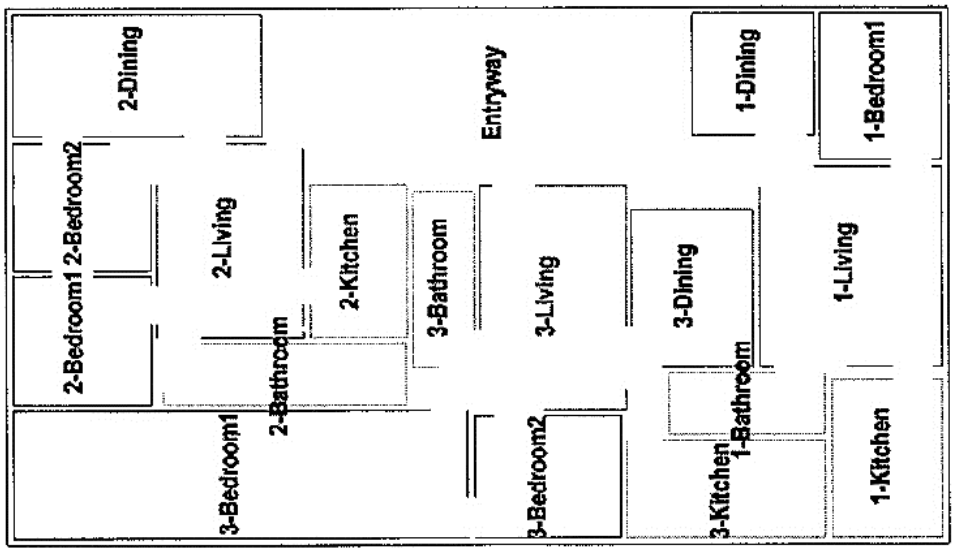
El objetivo era minimizar el valor objetivo de la solución encontrada por el proceso de optimización geométrica, la cual también ha sido formada por la topología. Las restricciones adoptadas estaban enfocadas en el traslapo de las unidades, en las conectividades deseadas, en las circulaciones deseadas, en la planaridad de las soluciones, y en su contención dentro de la envolvente principal del edificio.

Teniendo en cuenta que el espacio discreto de diseño de la topología es combinatorio, multi-modal y altamente restringido, y que el propósito de la optimización es generar un grupo de alternativas de diseño de alta calidad en vez de una única solución óptima (en parte por el aspecto estético), se seleccionó un GA para el proceso de optimización topológica. Con el fin de probar todo el procedimiento de optimización de plantas arquitectónicas, se llevó a cabo el diseño de un complejo pequeño de apartamentos, formado por tres apartamentos independientes por planta. El algoritmo corrió durante 20000 generaciones, con 100 soluciones de diseño en cada una.

De los resultados obtenidos se puede observar que, en el proceso de optimización topológica, el algoritmo tiende a reorganizar todas las soluciones de acuerdo con la primera solución viable encontrada, haciendo que éste sea dependiente del punto de partida. En este sentido, el procedimiento genera variaciones de un tipo de solución en vez de converger dentro de un frente de Pareto mucho más diferenciado. Por estas razones, Michalek et al. plantearon que el algoritmo es más útil para encontrar soluciones viables de diseño que para servir como un optimizador real. Por otro lado, se encontró que el procedimiento de optimización topológica aplicado en problemas más pequeños, es capaz de desarrollar una búsqueda más amplia dentro del espacio de diseño, llegando a obtener soluciones

de calidad. Con respecto a la optimización geométrica, se encontró que el algoritmo propuesto es capaz de manejar problemas mucho más grandes, en comparación con el proceso de optimización topológica. En el ejemplo desarrollado se incluyeron 312 variables y 1578 restricciones, correspondientes a 23 espacios, 3 vestíbulos, 1 contorno, y 25 accesos (llegando a 52 unidades).

Figura 20. Geometría final generada por la herramienta automática de diseño



Fuente: MICHALEK, J.J., CHOUDHARY, R. y PAPALAMBROS, P.Y. Architectural Layout Design Optimization. En: Engineering Optimization. No. 34 (2002); p. 461-484.

Otra aplicación de TCMOMH es presentada por Paya et al.[135]. Ellos desarrollaron una optimización multi-objetiva de pórticos de concreto utilizando un SA. El propósito del procedimiento era minimizar el costo de la estructura, en simultánea con la optimización de una función objetivo secundaria, como la facilidad en la

[135] PAYA, I., YEPES, V., GONZÁLEZ-VIDOSA, F. y HOSPITALER, A. Multiobjective Optimization of Concrete Frames by Simulated Annealing. En: Computer-Aided Civil and Infraestructure Engineering. No. 23 (2008); p. 596-610.

construcción de la estructura (el número de barras de refuerzo se toma como un indicador para esto), la sostenibilidad (basada en el Live Cycle Assessment, LCA, de los materiales utilizados en la estructura), o la seguridad global de la estructura (depende de la relación entre la resistencia de la estructura y las acciones resultantes que afectan dicha resistencia para los distintos estados límites planteados en el Código Español para concreto estructural).

Como resultado, el procedimiento es una optimización bi-objetiva con 3 escenarios. Las restricciones adoptadas fueron los estados límites de servicio y los estados últimos que la estructura debe satisfacer de acuerdo con el Código Español para concreto estructural.

Adicionalmente, fueron incluidas algunas restricciones correspondientes a la geometría y al aspecto constructivo de la estructura. Las variables para el procedimiento de optimización son discretas, haciendo que el problema sea combinatorio. Estas variables son las que definen la geometría de las secciones transversales de vigas y columnas, el tipo de concreto y acero que será utilizado, y el refuerzo de los pórticos. Para manejar los múltiples objetivos, Paya et al. implementaron una extensión del SA llamada SMOSA, propuesta por Suppapitnarm et al.[136].

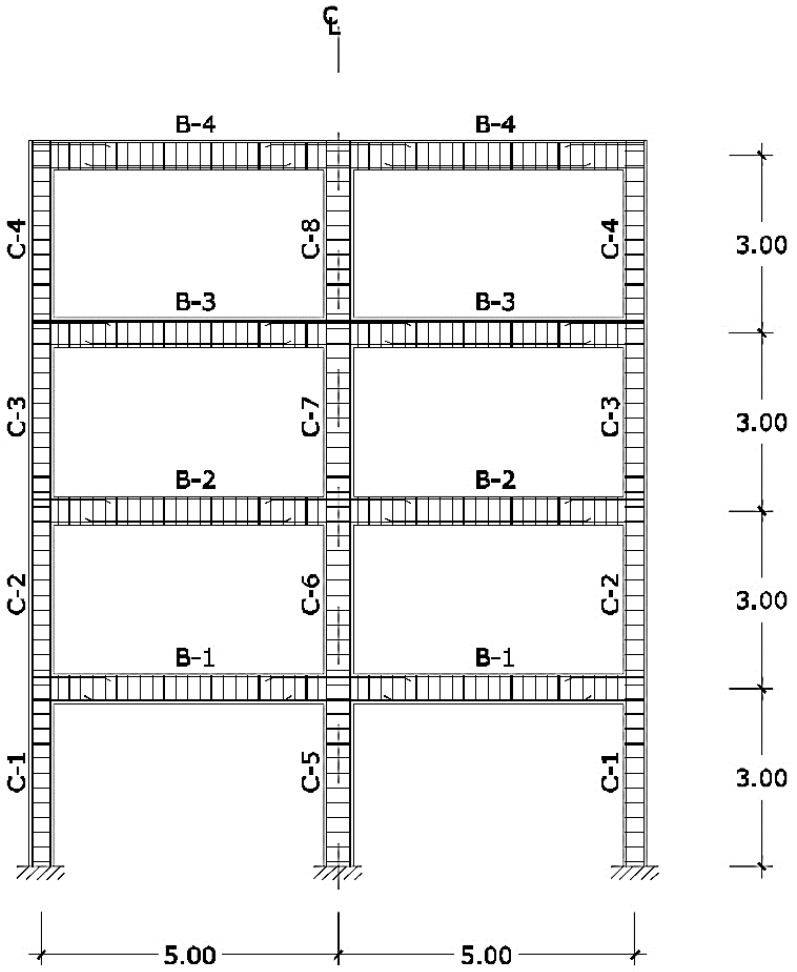
El procedimiento fue aplicado en un pórtico de concreto reforzado, cuatro pisos de altura y dos luces, resultando 77 variables de diseño que llevan a tener un número de combinaciones del orden de 10116 (las cuales exceden en tamaño a muchos de los estudios de optimización de estructuras en concreto reforzado).

Los resultados mostraron que un incremento del 5% en el costo de la estructura conlleva a una mejora del 32.25% en la función del aspecto constructivo, a una

[136] SUPPAPITNARM, A., SEFFEN, K.A., PARKS, G.T. y CLARKSON, P.J. A simulated annealing algorithm for multiobjective optimization. En: Engineering Optimization. No. 33 (2000); p. 59-85.

mejora del 24.04% en la función de sostenibilidad, y a una mejora del 6.11% en la función de la seguridad global del edificio.

Figura 21. Pórtico de concreto reforzado, utilizado en el ejemplo de aplicación



Fuente: PAYA, I., YEPES, V., GONZÁLEZ-VIDOSA, F. y HOSPITALER, A. Multiobjective Optimization of Concrete Frames by Simulated Annealing. En: Computer-Aided Civil and Infraestructure Engineering. No. 23 (2008); p. 596-610.

A través de sus aplicaciones, SMOSA probó ser una herramienta robusta y efectiva para generar un grupo de soluciones de alta calidad. Estas soluciones proveen estructuras en concreto reforzado más sostenibles, construibles y seguras, con un incremento pequeño en el costo y con un tiempo computacional razonable para su generación.

Retornando a las aplicaciones en arquitectura, Liu et al.[137] desarrollaron una investigación enfocada en el problema del zoning en el uso del suelo, tomando como caso de estudio el condado de Yicheng, en China central, el cual tiene una población de 515000 habitantes.

En este caso, el zoning en el uso del suelo pretende asignarle a cada zona (actualmente existen 47851 zonas) una zona de uso de suelo. En total, hay 8 zonas de uso de suelo: áreas básicas rurales de preservación (Basic Farmland Preservation Areas, BFPA), áreas de suelos para agricultura general (General Agricultural Land Areas, GALA), áreas de suelos para bosques (Forestry Land Areas, FLA), suelos de construcción para pueblos (Construction Land for Towns, CLT), suelos de construcción para villas (Construction Land for Villages, CLV), áreas de suelos para industrias independientes y minería (Independent Industrial and Mining Land Areas, IIMLA), áreas de suelos para turismo (Tourism Land Areas, TLA), y áreas de preservación del paisaje natural y humanístico (Natural and Humanistic Landscape Preservation Areas, NHLPA). Esta asignación depende de las características de cada zona de uso de suelo, de los objetivos a alcanzar, y de las restricciones.

[137] LIU, Y., WANG, H., JI, Y., LIU, Z. y ZHAO, X. Land Use Zoning at the County Level Based on a Multi-Objective Particle Swarm Optimization Algorithm: A Case Study from Yicheng, China. En: International Journal of Environmental Research and Public Health. No. 9 (2012); p. 2801-2826.

Las características de cada zona de uso de suelo incluían la calidad del suelo, la topografía, la infraestructura disponible para las actividades rurales, la ubicación, y el tipo de uso del suelo.

Por otro lado, los objetivos planteados eran 4: la diferencia de atributos entre zonas de uso de suelo (maximizar las diferencias entre todas las zonas de uso de suelo), la compactación espacial de las zonas de uso de suelo (reducir las zonas fragmentarias beneficia el control espacial), el grado de armonía espacial de las zonas de uso de suelo (la distribución apropiada de dos zonas de uso de suelo adyacentes promoverá el grado de convivencia de las actividades sociales de producción de las personas), y los beneficios ecológicos de las zonas de uso de suelo (distintos patrones de uso de suelo proveen un Ecosystem Services Value, ESV[138], distinto, luego, el objetivo es mejorar el ESV global, basado en las zonas de uso de suelo óptimas desde un punto de vista ecológico).

En cuanto a las restricciones, se estipuló un área mínima de parcela, la limitación del área de varias zonas de uso de suelo, y las regulaciones generales del zoning de uso de suelo, específicas de Yicheng.

Para manejar los múltiples objetivos, se utilizó un acercamiento lineal de sumatoria con pesos, con el fin de convertir los cuatro objetivos en una sola función. Luego, para resolver el problema de optimización, se utilizó un PSO Multi-Objetivo con factor de Constricción y operadores de Mutación y Crossover (MOPSO-CCM)[139].

[138] KONG, X.S., LIU, Y.F. y TAN, C.F. Correlation analysis of the changes of land use structure and land use efficiency: A case study of Jiayu County in Hubei Province. En: Resources Science. No. 31 (2009); p. 1195-1101.
[139] LIU, Y., WANG, H., JI, Y., LIU, Z. y ZHAO, X. Land Use Zoning at the County Level Based on a Multi-Objective Particle Swarm Optimization Algorithm: A Case Study from Yicheng, China. En: International Journal of Environmental Research and Public Health. No. 9 (2012); p. 2801-2826.

El factor de constricción[140] se implementó para mejorar el rendimiento del PSO convencional[141], mientras que los operadores de mutación y crossover se utilizaron para evitar que el PSO quedara atrapado en un mínimo o máximo local.
En su trabajo, Liu et al. plantearon que la situación de tener múltiples objetivos dentro de un contexto espacial, conlleva a problemas complejos de optimización espacial, los cuales, generalmente, presentan una complejidad computacional significativa.

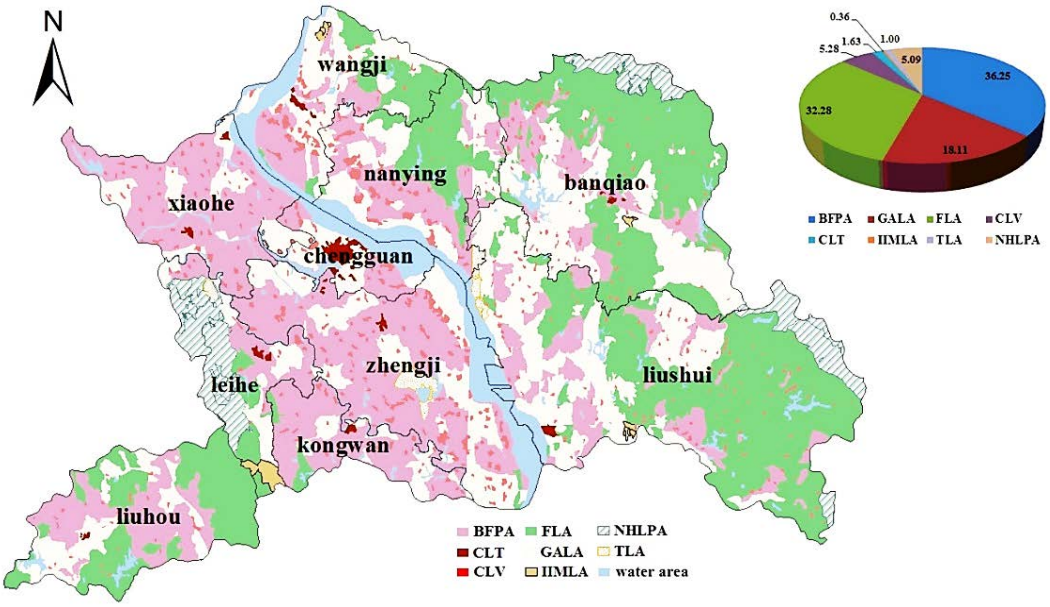
Esta situación, resalta a los métodos exactos de optimización (como el Linear Integer Programming, LIP) como inviables para manejar un gran número de unidades de suelo[142][143]; por otro lado, las técnicas heurísticas (como el SA y los GAs) son más efectivas en la resolución de este tipo de problemas[144].

De los resultados obtenidos, se puede concluir que la aplicación de MOPSO-CCM en conjunto con un Sistema de Información Geográfica (GIS), es una herramienta favorable y capaz en el desarrollo de problemas de zoning para el uso del suelo.

Adicionalmente, Liu et al. plantearon que la función objetivo podría ser más amplia, con el fin de incluir condiciones singulares de un territorio específico, como beneficios económicos o contigüidad espacial, entre otros.

[140] CLERC, M. The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization. En: ICEC. Proceedings of ICEC. Washington: s.e., 1999. p. 1951-1957.
[141] EBERHART, R.C. y SHI, Y. Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization. En: CONGRESS ON EVOLUTIONARY COMPUTING. Proceedings of 2000 Congress on Evolutionary Computing (CEC2000). La Jolla: s.e., 2000. p. 84-88.
[142] AERTS, J.C.J.H., EISINGER, E., HEUVELINK, G.B.M. y STEWART, T.J. Using linear integer programming for multi-site land-use allocation. En: Geographical Analysis. No. 25 (2003); p. 148-169.
[143] STEWART, T.J., JANSSEN, R. y VAN HERWIJNEN, M. A genetic algorithm approach to multi-objective land use planning. En: Computers and Operations Research. No. 31 (2004); p. 2293-2313.
[144] DUH, J.D. y BROWN, D.G. Knowledge-informed Pareto simulated annealing for multi-objective spatial allocation. En: Computers, Environment and Urban Systems. No. 31 (2007); p. 253-281.

Figura 22. Distribución especial y proporción entre las áreas (%) de las ocho zonas de uso de suelo (siendo los 4 pesos iguales a 0.25)



Fuente: LIU, Y., WANG, H., JI, Y., LIU, Z. y ZHAO, X. Land Use Zoning at the County Level Based on a Multi-Objective Particle Swarm Optimization Algorithm: A Case Study from Yicheng, China. En: International Journal of Environmental Research and Public Health. No. 9 (2012); p. 2801-2826.

Sin embargo, existe una debilidad que afecta, en general, a todos los acercamientos que utilizan un método convencional de sumatoria con pesos para el manejo de múltiples objetivos. Con una sola corrida del algoritmo, se obtiene una sola solución óptima, por consiguiente, el paso siguiente en la investigación sería la aplicación de un acercamiento dinámico de sumatoria con pesos (Dynamic Weighted Aggregation, DWA), para poder generar una serie de soluciones óptimas de Pareto[145]. Aparte de este futuro paso, se sugiere llevar a cabo estudios

[145] LIU, Y., WANG, H., JI, Y., LIU, Z. y ZHAO, X. Land Use Zoning at the County Level Based on a Multi-Objective Particle Swarm Optimization Algorithm: A Case Study from Yicheng, China. En: International Journal of Environmental Research and Public Health. No. 9 (2012); p. 2801-2826.

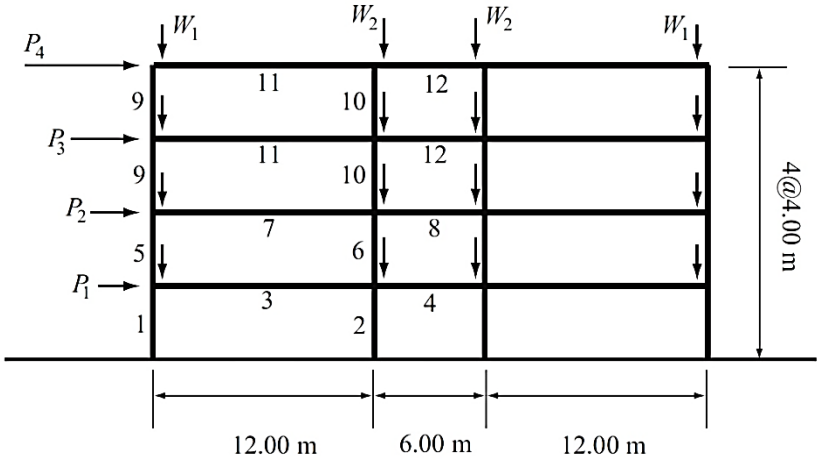
posteriores con el fin de determinar los parámetros apropiados que definen el comportamiento y el rendimiento de los algoritmos multi-objetivos aplicados en el problema de zoning para el uso del suelo.

Una aplicación adicional de TCMOMH en el diseño estructural fue la desarrollada por Ohsaki y Kinoshita[146]. Ellos plantearon que, debido a los costos computacionales significativos en la evaluación de las funciones objetivo de los problemas de optimización estructural, los acercamientos multi-punto (como los GAs) podrían no ser las estrategias apropiadas, más aún cuando el proceso de optimización es desarrollado con una estructura grande y compleja. Por este motivo, ellos llevaron a cabo una aplicación para optimización estructural multi-objetiva, con dos de las estrategias más comunes de un solo punto: el SA y el TS.

El ejemplo de aplicación desarrollado, con el fin de probar el rendimiento de estas estrategias, y compararlas, fue un marco plano con 3 luces y 4 pisos. Sobre él, fueron aplicadas cargas horizontales, que incrementaban su magnitud en proporción a la altura de aplicación, y cargas verticales fijas, aplicadas en los nodos del marco simétrico. Todos los elementos fueron divididos en 12 grupos, teniendo en cuenta las condiciones de simetría. Durante el problema de optimización, las variables fueron las áreas de las secciones transversales de los elementos de cada grupo. Luego, se definió el segundo momento de inercia y el módulo plástico de la sección, en función de dichas variables. Adicionalmente, se implementó una definición para las áreas de la sección transversal, con el fin de garantizar que la sección de un miembro fuese mayor que la correspondiente al miembro ubicado inmediatamente en un nivel superior.

[146] OHSAKI, M. y KINOSHITA, T. Single-Point Search Heuristic Methods for Multiobjective Structural Optimization. En: FCS/TECHNO-SYMPO/MPS SYMPOSIUM 2005. Proceedings of Computational Science Symposium. p. 59-66.

Figura 23. Marco en el que se desarrolló el ejemplo de aplicación



Fuente: OHSAKI, M. y KINOSHITA, T. Single-Point Search Heuristic Methods for Multiobjective Structural Optimization. En: FCS/TECHNO-SYMPO/MPS SYMPOSIUM 2005. Proceedings of Computational Science Symposium. p. 59-66.

Posteriormente, se definieron dos funciones objetivo: el volumen estructural total, con el fin de minimizarlo, y la energía plástica disipada, con el fin de maximizarla. La evaluación de estos objetivos se llevó a cabo en el estado final de carga, cuando el desplazamiento horizontal en el nivel de la cubierta alcanzó un valor de 2.5% de la altura del marco. Para el análisis estructural, se utilizó CLAP[147].

De los resultados obtenidos, se puede concluir que ambos métodos pueden generar eficientemente un grupo de soluciones óptimas de Pareto. Para alcanzar esto, se realizó una modificación importante en el SA, con el fin de garantizar que la solución del vecindario, en la dirección de la solución rechazada en el paso anterior, fuese automáticamente rechazada y reemplazada por otra solución del vecindario.

[147] OGAWA, K. y TADA, M. Computer program for static and dynamic analysis of steel frames considering the deformation of joint panel. En: 17TH SYMPOSIUM ON COMPUTER TECHNOLOGY OF INFORMATION. Proceedings of the Seventeenth Symposium on Computer Technology of Information. Systems and Applications. p. 79-84.

A pesar de la eficiencia de las dos estrategias, se encontró que el TS tuvo una ligera ventaja, correspondiente a la diversidad de soluciones. Ohsaki y Kinoshita argumentaron que la introducción de algunas técnicas en GAs multi-objetivos (como funciones para compartir información o nichos) pueden ser importantes en la generación de un grupo óptimo de Pareto, con la diversidad suficiente en sus soluciones. Sin embargo, los autores no presentan un ejemplo de comparación con GAs para probar sus planteamientos.

Para concluir con este capítulo, vale la pena comentar que, con base en el estado del arte que se presentó, se elaboró un artículo titulado “Multi-Objective Heuristic Computation Applied to Architectural and Structural Design: A Review”, el cual ha sido aceptado para publicación en el columen especial de la revista *International Journal of Architectural Computing*.

2. PROPUESTA DE UN ACERCAMIETO DE OPTIMIZACIÓN MULTI-OBJETIVA

El acercamiento de optimización multi-objetiva está basado en la integración de un algoritmo PSO[148] con uno UPSO[149]. La selección de estos dos algoritmos corresponde a dos factores principales: por un lado, los principales autores de referencia, Sassone y Pugnale[150], en una de sus publicaciones sugieren como futuro paso en la investigación del tema en estudio, aplicar el algoritmo PSO en el tipo de problemas que en esta investigación se pretenden resolver (argumentando que dicho algoritmo ha tenido muy buenos resultados en distintos campos de aplicación); adicionalmente, Parsopoulos y Vrahatis[151] demostraron la eficiencia del algoritmo UPSO, por medio de funciones de prueba como la Esfera, Rosenbrock, Rastrigin, y Griewank, con 30 dimensiones (de especial interés para el desarrollo de presente proyecto de investigación ya que es necesario trabajar con un gran número de variables), y con la función bi-dimensional de Schaffer.

En síntesis, la propuesta es utilizar el PSO para optimizar los parámetros que definen el comportamiento y el rendimiento del UPSO, el cual conducirá el proceso de optimización multi-objetiva. Esto se propone debido a que los valores asignados a dichos parámetros son inciertos y dependen del problema que se

[148] KENNEDY, J. y EBERHART, R. Particle Swarm Optimization. En: IEEE INTERNATIONAL CONFERENCE ON NEURAL NETWORKS. Proceedings of the 1995 IEEE International Conference on Neural Networks. Piscataway: IEEE Service Center, 1995. p. 1942-1948.
[149] PARSOPOULOS, K.E. y M.N., VRAHATIS. UPSO: A Unified Particle Swarm Optimization Scheme. En: Lecture Series on Computer and Computational Sciences. No. 1 (2004); p. 868-873.
[150] SASSONE, M. y PUGNALE, A. Optimal design of glass grid shells with quadrilateral elements by means of a genetic algorithm. En: 6TH INTERNATIONAL CONFERENCE ON COMPUTATION OF SHELL AND SPATIAL STRUCTURES. Proceedings of the 6th International Conference on Computation of Shell and Spatial Structures IASS-IACM 2008: Spanning Nano to Mega. Ithaca: J.F. Abel and J.R Cooke, 2008.
[151] PARSOPOULOS, K.E. y M.N., VRAHATIS. UPSO: A Unified Particle Swarm Optimization Scheme. En: Lecture Series on Computer and Computational Sciences. No. 1 (2004); p. 868-873.

pretende resolver, como puede verse en el trabajo de Toscano-Pulido et al.[152] y en el de Liu et al.[153].

Es claro que el(la) usuario(a) debe definir los parámetros para el PSO y, es posible que aquellos que seleccione no sean los mejores. Sin embargo, el acercamiento que se propone explora un espacio de búsqueda mucho más amplio de combinaciones de parámetros, en comparación con la exploración manual que, por lo general, debe llevar a cabo el(la) diseñador(a).

El UPSO es un acercamiento que combina las capacidades de exploración y explotación del PSO. Para esto, la velocidad de las partículas (posibles soluciones) es actualizada en base a un variante global, $\mathcal{G}_i^{(k+1)}$, y a uno local, $\mathcal{L}_i^{(k+1)}$, del PSO[154]:

$$\mathcal{G}_i^{(k+1)} = \mathcal{X}\left[v_i^{(k)} + c_1 r_1\left(p_i^{(k)} - x_i^{(k)}\right) + c_2 r_2\left(p_g^{(k)} - x_i^{(k)}\right)\right] \quad (10)$$

$$\mathcal{L}_i^{(k+1)} = \mathcal{X}\left[v_i^{(k)} + c_1 r_1'\left(p_i^{(k)} - x_i^{(k)}\right) + c_2 r_2'\left(p_{gi}^{(k)} - x_i^{(k)}\right)\right] \quad (11)$$

donde i es el número de la partícula (posible solución); k es el número de la iteración; $\mathcal{X}$ es el factor de constricción (un parámetro que controla la magnitud de la velocidad); $v_i^{(k)}$ es la velocidad de la partícula $i - ésima$ en la iteración k; $p_i^{(k)}$ es la mejor posición que la partícula $i - ésima$ ha encontrado hasta la iteración k; $x_i^{(k)}$ es la posición de la partícula $i - ésima$ en la iteración k; c_1 y c_2 son dos constantes

[152] TOSCANO-PULIDO, G, SANTANA-QUINTERO, LV y COELLO COELLO, CA. EMOPSO: A Multi-Objective Particle Swarm Optimizer with Emphasis on Efficiency. En: Evolutionary Multi-Criterion Optimization. No. 1 (2007); p. 272-285.
[153] LIU, Y., WANG, H., JI, Y., LIU, Z. y ZHAO, X. Land Use Zoning at the County Level Based on a Multi-Objective Particle Swarm Optimization Algorithm: A Case Study from Yicheng, China. En: International Journal of Environmental Research and Public Health. No. 9 (2012); p. 2801-2826.
[154] PARSOPOULOS, K.E. y M.N., VRAHATIS. UPSO: A Unified Particle Swarm Optimization Scheme. En: Lecture Series on Computer and Computational Sciences. No. 1 (2004); p. 868-873.

positivas (son parámetros de aceleración llamados, respectivamente, parámetro cognitivo y parámetro social); r_1 y r_2 son dos números aleatorios contenidos en el intervalo $[0,1]$; $p_g^{(k)}$ es la mejor posición que ha encontrado la totalidad del enjambre hasta la iteración k (variante global); y $p_{gi}^{(k)}$ es la mejor posición que ha encontrado el vecindario de $x_i^{(k)}$ hasta la iteración k (variante local). Luego, la actualización de la velocidad de cada partícula es una combinación de las ecuaciones (10) y (11), junto con un parámetro u, llamado el factor de unificación (con el fin de determinar la influencia que tienen los componentes globales y locales); si u=1, la actualización de la velocidad es equivalente al variante global del PSO, mientras que si u=0, será equivalente al variante local del PSO[155]:

$$U_i^{(k+1)} = u\mathcal{G}_i^{(k+1)} + (1-u)\mathcal{L}_i^{(k+1)} \quad (12)$$

Finalmente, la actualización de la posición de la partícula estará determinada por[156]:

$$x_i^{(k+1)} = x_i^{(k)} + U_i^{(k+1)} \quad (13)$$

De las ecuaciones (10) a (13), se puede observar que el UPSO depende de 4 parámetros inciertos: c_1, c_2, $\mathcal{X}$ y u. Por consiguiente, lo que se propone es aplicar un PSO de cuatro dimensiones con el fin de optimizar dichos parámetros. Por otro lado, se propone aplicar el acercamiento Conventional Weighted Aggregation (CWA)[157] para manejar la multiplicidad de objetivos. En este acercamiento, todos los objetivos, $f_i(x)$, se suman en una combinación con pesos fijos, de la siguiente manera:

[155] PARSOPOULOS, K.E. y M.N., VRAHATIS. UPSO: A Unified Particle Swarm Optimization Scheme. En: Lecture Series on Computer and Computational Sciences. No. 1 (2004); p. 868-873.
[156] Ibid.
[157] PARSOPOULOS, K.E. y VRAHATIS, M.N. Particle Swarm Optimization method in multiobjective problems. En: Proceedings ACM Symposium on Applied Computing (SAC 2002). 2002. pp. 603-607.

$$F = \sum_{i=1}^{k} w_i f_i(x) \quad (14)$$

donde w_i, con $i = 1, \dots, k$, son pesos no negativos; usualmente se asume que $\sum_{i=1}^{k} w_i = 1$. Luego, la función a optimizar sería F.

En cuanto a la aplicación de restricciones, éstas se manejaron por medio del acercamiento de función de penalización[158]. Este acercamiento se fundamenta en redefinir la función objetivo, en este caso F, por medio de la inclusión de una función que penaliza, proporcionalmente, las soluciones que incumplen con las restricciones impuestas; entre más distante esté la solución del cumplimiento de la restricción, el valor de penalización será mayor. Para llevar a cabo este acercamiento, es necesario definir las restricciones de la siguiente manera:

$$g_i(x) \leq 0, \quad i = 1, \dots, m \quad (15)$$

donde i hace referencia al número de la restricción. Si la restricción está planteada como $g_i(x) \geq 0$, se debe replantear como $-g_i(x) \leq 0$; en cambio, si la restricción es de igualdad, $g_i(x) = 0$, debe replantearse como dos restricciones: $g_i(x) \leq 0$ y $-g_i(x) \leq 0$. Luego de definir las restricciones, se replantea la función objetivo de la siguiente manera:

$$F(x) = f(x) + h(k)H(x), \quad x \in S \subset \mathbb{R}^n \quad (16)$$

Donde $f(x)$ es la función objetivo original (sin restricciones); $h(k)$ es un valor de penalización dinámicamente modificado, siendo k el número de la iteración actual del algoritmo; y $H(x)$ es el factor de penalización, definido por:

[158] PARSOPOULOS, KE y VRAHATIS, MN. Particle Swarm optimization method for constrained optimization problems. En: 2nd EURO-INTERNATIONAL SYMPOSIUM ON COMPUTATIONAL INTELLIGENCE. Proceedings of the second Euro-International Symposium on Computational Intelligence. Kosice: Kvasnicka, V. et al., 2002. p. 214-220.

$$H(x) = \sum_{i=1}^{m} \theta\big(q_i(x)\big) q_i(x)^{\gamma(q_i(x))} \quad (17)$$

donde, $q_i(x) = max\{0, g_i(x)\}$, $i = 1, \dots, m$. La función $q_i(x)$ es una función de violación relativa de las restricciones; $\theta\big(q_i(x)\big)$ es una función de asignación multi-fase; y $\gamma\big(q_i(x)\big)$ es la potencia de la función de penalización.

Para la definición de cada una de las anteriores funciones se planteó lo expuesto por Parsopoulos y Vrahatis[159]. En este sentido, dichas funciones se fijaron de la siguiente forma:

$$\gamma\big(q_i(x)\big) = \begin{cases} 1 \; si \; q_i(x) < 1 \\ 2 \; si \; q_i(x) \geq 1 \end{cases} \quad (18)$$

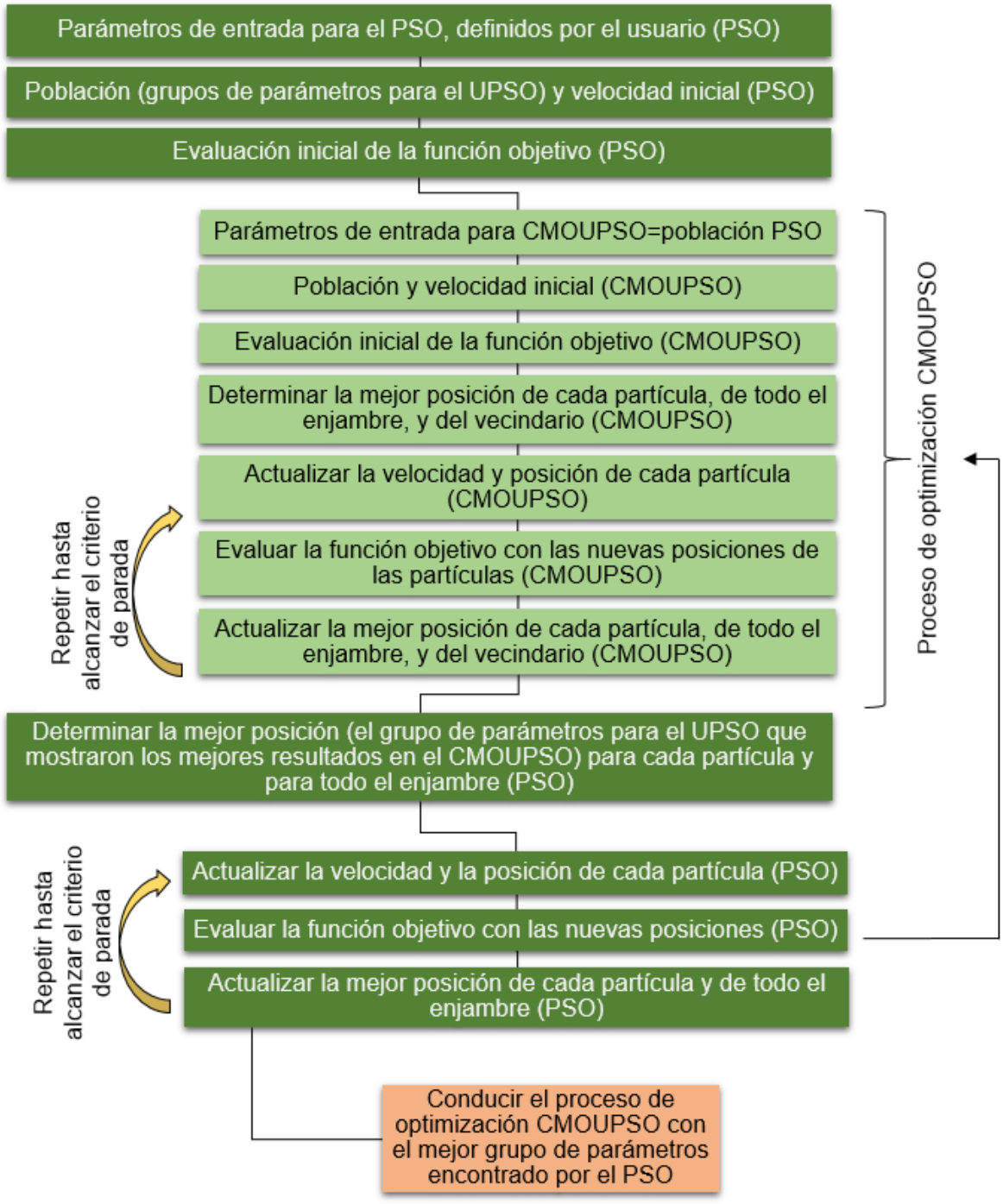
$$\theta\big(q_i(x)\big) = \begin{cases} 10 \; si \; q_i(x) < 0.001 \\ 20 \; si \; q_i(x) \leq 0.100 \\ 100 \; si \; q_i(x) \leq 1.000 \\ 300 \; si \; q_i(x) > 1.000 \end{cases} \quad (19)$$

$$h(k) = \sqrt{k} \quad (20)$$

Como resultado, el acercamiento propuesto de optimización es un UPSO Multi-Objetivo Auto-Configurado con Restricciones (ACCMOUPSO por sus siglas en inglés, Auto-Configured Constrained Multi-Objective UPSO), el cual se resume en la Figura 24.

[159] PARSOPOULOS, KE y VRAHATIS, MN. Particle Swarm optimization method for constrained optimization problems. En: 2nd EURO-INTERNATIONAL SYMPOSIUM ON COMPUTATIONAL INTELLIGENCE. Proceedings of the second Euro-International Symposium on Computational Intelligence. Kosice: Kvasnicka, V. et al., 2002. p. 214-220.

Figura 24. Diagrama de ACCMOUPSO



Fuente: Autor

2.1. VALIDACIÓN DEL ACERCAMIENTO PROPUESTO DE OPTIMIZACIÓN MULTI-OBJETIVA

Para validar el acercamiento propuesto de optimización multi-objetiva (i.e. el ACCMOUPSO), al igual que en el trabajo de Szollos et al.[160], se tomaron como funciones de prueba las funciones ZDT1, ZDT2 y ZDT3, propuestas por Zitzler et al.[161][162], y definidas a continuación:

- ZDT1: esta función presenta un frente de Pareto convexo.

$$f_1(x) = x_1 \tag{21}$$

$$g(\vec{x}) = 1 + 9 * \sum_{i=2}^{m} x_i/(m-1) \tag{22}$$

$$h(f_1, g) = 1 - \sqrt{f_1/g} \tag{23}$$

donde $m = 30$, y $x_i \in [0,1]$.

- ZDT2: esta función presenta un frente de Pareto cóncavo.

$$f_1(x) = x_1 \tag{24}$$

$$g(\vec{x}) = 1 + 9 * \sum_{i=2}^{m} x_i/(m-1) \tag{25}$$

$$h(f_1, g) = 1 - (f_1/g)^2 \tag{26}$$

donde $m = 30$, y $x_i \in [0,1]$.

[160] SZOLLOS, A., SMID, M. y HAJEK, J. Aerodynamic optimization via multi-objective micro-genetic algorithm with range adaptation, knowledge-based reinitialization, crowding and e-dominance. En: Advances in Engineering Software. No. 40 (2009); p. 419-430.
[161] ZITZLER, E., DEB, K. y THIELE, L. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. En: Evolutionary Computation. No. 8 (2000); p. 173-195.
[162] DEB, K. Multi-objective genetic algorithms: problem difficulties and construction of test problems. En: Evolutionary Computation. No. 7 (1999); p. 205-230.

- ZDT3: esta función presenta un frente de Pareto discontinuo, conformado por varios trozos convexos no-contiguos.

$$f_1(x) = x_1 \tag{27}$$

$$g(\vec{x}) = 1 + 9 * \sum_{i=2}^{m} x_i/(m-1) \tag{28}$$

$$h(f_1, g) = 1 - \sqrt{f_1/g} - (f_1/g) * sin(10\pi f_1) \tag{29}$$

donde $m = 30$, y $x_i \in [0,1]$.

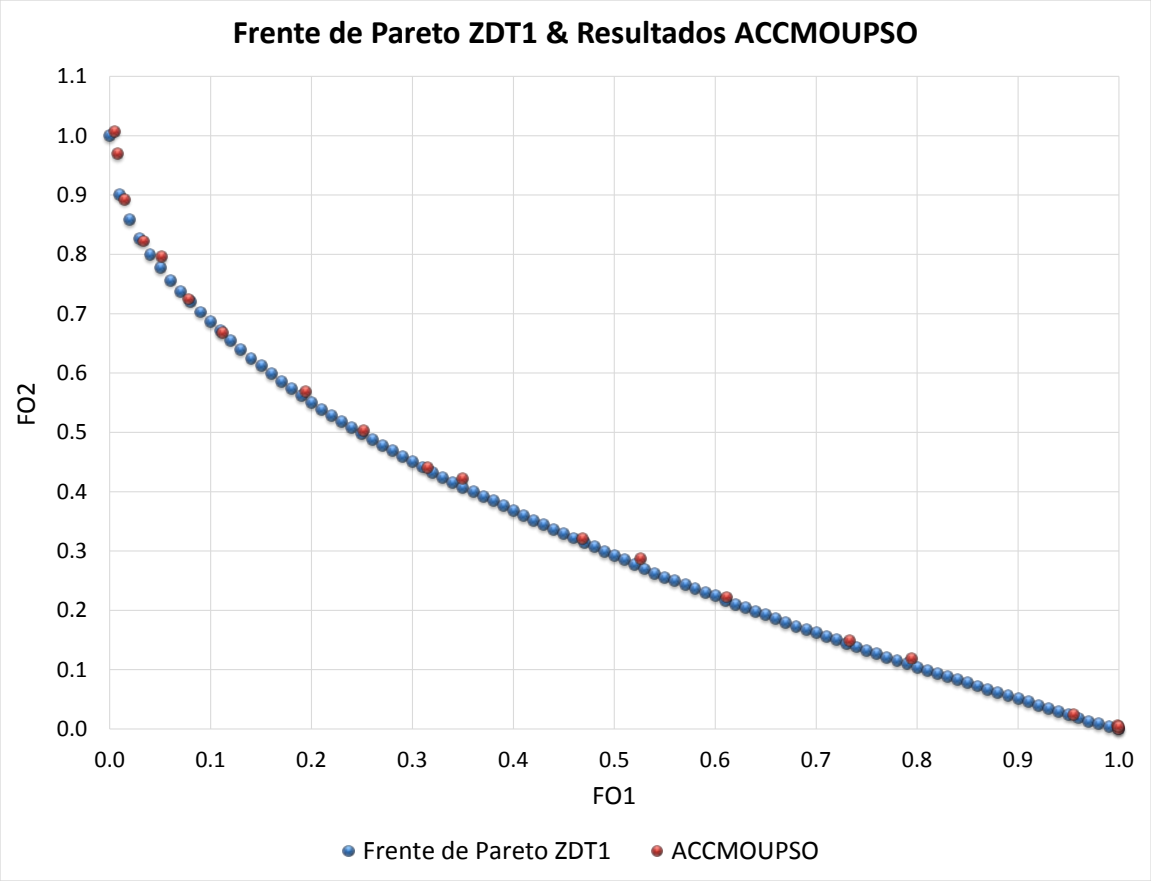
Para las tres funciones, el PSO corrió, dentro de ACCMOUPSO, con 50 partículas, durante 50 iteraciones, con el factor de constricción (X) igual a 0.5, y los parámetros c_1 y c_2 iguales a 1.2 y 1.8, respectivamente. Por otro lado, el UPSO multi-objetivo corrió dentro del PSO, de igual forma, con 50 partículas y 50 iteraciones. Posteriormente, el UPSO multi-objetivo se ejecutó 20 veces, con combinaciones de pesos distintas, con los parámetros encontrados por el PSO, y con 40000 partículas durante 125 iteraciones.

Los resultados obtenidos con ACCMOUPSO para cada una de las funciones (ZDT1, ZDT2 y ZDT3) son los que se muestran en el Cuadro 1, Cuadro 2 y Cuadro 3, y en la Figura 25, Figura 26 y Figura 27. A partir de ellos, es posible observar que ACCMOUPSO es capaz de aproximarse con significativa precisión al frente de Pareto de cada una de las funciones de prueba, tanto en el valor obtenido en cada una de las corridas (correspondiente a una determinada combinación de pesos), como en la distribución de los mismos sobre el frente de Pareto.

Cuadro 1. Resultados de ACCMOUPSO para la función ZDT1

Corrida	ACCMOUPSO		Parámetros encontrados para el UPSO				Tiempo (s)
	F.O. 1	F.O. 2	c1	c2	X	u	
1	1.0000	0.0008	0.9354	2.4663	0.6607	0.5595	535.89
2	0.9989	0.0062	1.1395	2.0295	0.7004	0.4319	523.17
3	0.9999	0.0046	1.3956	1.5780	0.6933	0.4837	520.07
4	0.9550	0.0247	1.4719	1.7680	0.6707	0.5263	532.85
5	0.7941	0.1195	0.9555	1.2199	0.7713	0.4874	533.04
6	0.7334	0.1490	0.6064	1.6942	0.6797	0.4078	519.17
7	0.6109	0.2224	1.0080	2.5528	0.6392	0.4750	522.21
8	0.5259	0.2874	0.5236	1.7073	0.6706	0.3684	518.13
9	0.4684	0.3207	1.0929	0.7883	0.7601	0.4597	525.30
10	0.3496	0.4216	0.8716	1.9615	0.6705	0.3471	520.56
11	0.3146	0.4403	1.1478	1.8858	0.6599	0.2287	524.26
12	0.2517	0.5032	0.1702	3.1114	0.5963	0.3829	521.11
13	0.1937	0.5681	0.9983	2.0428	0.6771	0.4560	528.74
14	0.1119	0.6676	1.3815	1.8342	0.6458	0.5651	523.73
15	0.0778	0.7249	0.7841	2.3577	0.6486	0.4915	528.13
16	0.0511	0.7959	1.2457	2.6660	0.5882	0.3488	522.82
17	0.0332	0.8226	1.3493	1.4863	0.7476	0.4120	541.32
18	0.0148	0.8930	0.7793	1.5170	0.7387	0.5302	542.34
19	0.0077	0.9697	0.7375	0.7858	0.7570	0.5864	536.83
20	0.0044	1.0078	0.7424	1.1832	0.7458	0.5392	528.11
Promedio:			0.9668	1.8318	0.6861	0.4544	527.39

Figura 25. Comparación entre el frente de Pareto de la función ZDT1 y los resultados obtenidos con ACCMOUPSO (ver Cuadro 1)

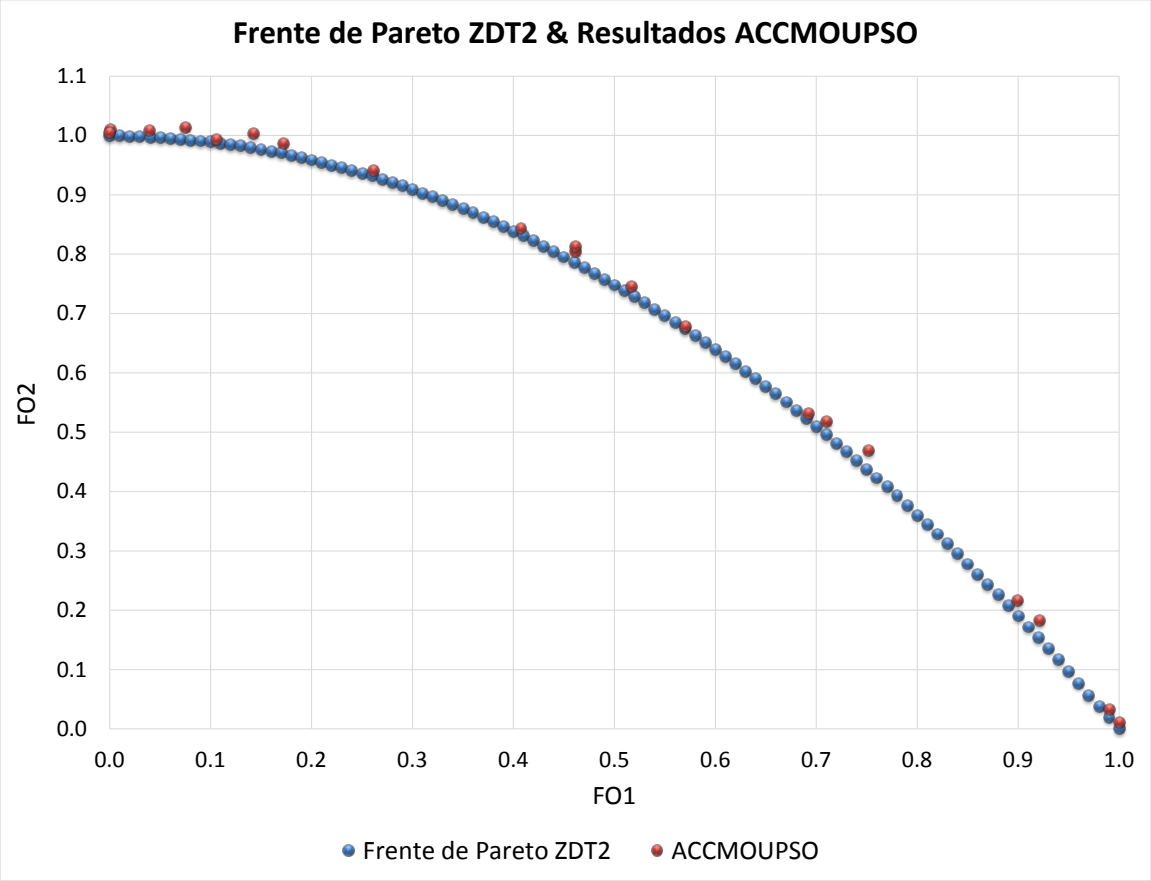


Fuente: Autor

Cuadro 2. Resultados de ACCMOUPSO para la función ZDT2.

Corrida	ACCMOUPSO		Parámetros encontrados para el UPSO				Tiempo (s)
	F.O. 1	F.O. 2	c1	c2	X	u	
1	1.0000	0.0105	0.3425	3.0290	0.5943	0.2924	526.53
2	0.9906	0.0324	1.2879	2.2543	0.6667	0.4035	540.80
3	0.9210	0.1822	0.7010	1.6255	0.7462	0.6413	531.87
4	0.8993	0.2165	0.6893	1.2330	0.7263	0.5041	538.41
5	0.7518	0.4692	1.3652	1.8278	0.7274	0.6224	536.30
6	0.7106	0.5179	0.0871	1.3966	0.7289	0.4596	521.61
7	0.6922	0.5307	1.4646	1.8268	0.7131	0.7072	527.50
8	0.5702	0.6787	1.2611	1.5199	0.7164	0.7933	539.58
9	0.5166	0.7460	0.8156	2.3970	0.6783	0.4987	523.61
10	0.4619	0.8041	0.7024	2.8516	0.6272	0.3002	538.37
11	0.4619	0.8131	0.7024	2.8516	0.6272	0.3002	523.01
12	0.4075	0.8439	0.7854	3.1777	0.6127	0.3558	533.57
13	0.2609	0.9403	0.8733	1.9819	0.6681	0.5107	522.03
14	0.1721	0.9856	0.7385	0.8042	0.7522	0.4643	522.17
15	0.1420	1.0031	1.1623	1.2523	0.7291	0.5256	531.76
16	0.1056	0.9929	1.0633	1.8328	0.6686	0.3412	522.48
17	0.0747	1.0142	1.1110	1.0444	0.7129	0.4612	529.22
18	0.0398	1.0082	0.3212	1.4050	0.7188	0.3968	514.48
19	0.0010	1.0099	0.5817	1.9194	0.6670	0.3617	528.84
20	0.0000	1.0048	0.7001	2.5810	0.6194	0.2938	527.89
Promedio:			0.8378	1.9406	0.6850	0.4617	529.00

Figura 26. Comparación entre el frente de Pareto de la función ZDT2 y los resultados obtenidos con ACCMOUPSO (ver Cuadro 2)

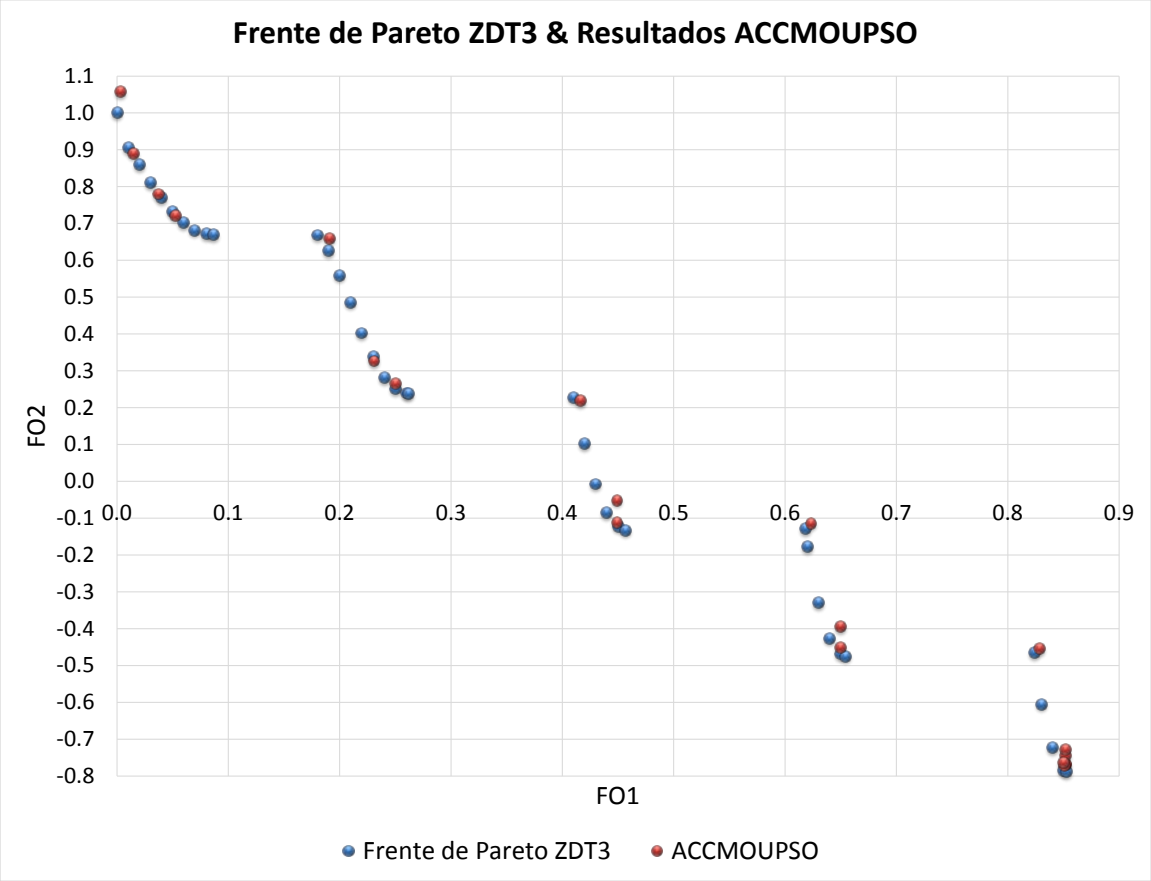


Fuente: Autor

Cuadro 3. Resultados de ACCMOUPSO para la función ZDT3.

Corrida	ACCMOUPSO		Parámetros encontrados para el UPSO				Tiempo (s)
	F.O. 1	F.O. 2	c1	c2	X	u	
1	0.8521	-0.7699	1.2154	2.5524	0.6250	0.3586	525.67
2	0.8517	-0.7672	0.9422	1.9023	0.7495	0.5559	518.00
3	0.8515	-0.7446	0.5175	1.0893	0.7544	0.8128	519.95
4	0.8514	-0.7273	0.4911	1.1141	0.7420	0.6072	520.97
5	0.8509	-0.7716	1.1394	2.1094	0.6994	0.3632	517.31
6	0.8500	-0.7645	1.0497	2.7484	0.6373	0.5396	518.42
7	0.8284	-0.4543	0.5707	2.7021	0.6169	0.5068	521.98
8	0.6501	-0.4515	0.6414	1.4242	0.7804	0.5657	516.23
9	0.6497	-0.3947	0.4133	1.2625	0.7520	0.5196	520.30
10	0.6232	-0.1139	1.5618	2.4370	0.6319	0.5098	518.62
11	0.4491	-0.0511	0.3213	1.3212	0.6301	0.4356	535.45
12	0.4490	-0.1113	0.6429	2.3057	0.6732	0.4553	529.36
13	0.4159	0.2198	1.0962	1.1011	0.6313	0.6452	523.76
14	0.2499	0.2655	0.6104	1.0310	0.7300	0.4717	520.40
15	0.2307	0.3264	1.1401	1.0400	0.7467	0.4059	519.32
16	0.1906	0.6600	0.5369	2.2649	0.6899	0.6302	526.80
17	0.0525	0.7221	0.8710	1.6161	0.7259	0.4239	531.73
18	0.0370	0.7799	1.3307	1.5247	0.7004	0.4567	542.68
19	0.0149	0.8904	0.8712	2.4985	0.6926	0.4053	542.22
20	0.0033	1.0576	1.3307	1.5247	0.7004	0.4567	542.68
Promedio:			0.8647	1.7785	0.6955	0.5063	525.59

Figura 27. Comparación entre el frente de Pareto de la función ZDT3 y los resultados obtenidos con ACCMOUPSO (ver Cuadro 3).



Fuente: Autor

Por último, vale la pena resaltar que con la ejecución de ACCMOUPSO, fue posible aproximarse a los valores óptimos para los parámetros que definen el comportamiento y el rendimiento del UPSO utilizado, dejando a un lado la incertidumbre que se produce cada vez que se deben definir dichos valores[163][164] (ver Cuadro 1, Cuadro 2 y Cuadro 3, en los cuales se encuentra el promedio de los valores encontrados por el PSO para cada uno de los parámetros del UPSO). En este caso, no se presentó una variación significativa en los parámetros del UPSO, correspondiente a la solución de las tres funciones utilizadas: ZDT1, ZDT2 y ZDT3; por ejemplo, el promedio del coeficiente c_1 varió entre 0.8378 y 0.9668, el del coeficiente c_2 entre 1.7785 y 1.9406, el del factor de constricción (X) entre 0.6850 y 0.6955, y el correspondiente al factor de unificación (u) entre 0.4544 y 0.5063.

[163] TOSCANO-PULIDO, G, SANTANA-QUINTERO, LV y COELLO COELLO, CA. EMOPSO: A Multi-Objective Particle Swarm Optimizer with Emphasis on Efficiency. En: Evolutionary Multi-Criterion Optimization. No. 1 (2007); p. 272-285.

[164] LIU, Y., WANG, H., JI, Y., LIU, Z. y ZHAO, X. Land Use Zoning at the County Level Based on a Multi-Objective Particle Swarm Optimization Algorithm: A Case Study from Yicheng, China. En: International Journal of Environmental Research and Public Health. No. 9 (2012); p. 2801-2826.

3. PROCEDIMIENTO AUTO-CONFIGURADO PARA EL DISEÑO MULTI-OBJETIVO DE ESTRUCTURAS RETICULARES (PACDMOER)

El procedimiento que se propone pretende aproximarse al nuevo paradigma de diseño que se plantea en la introducción del presente documento, por medio de fundamentar su funcionamiento en la integración de procesos de formación, generación, evaluación y rendimiento, dentro de un medio digital de diseño.

Específicamente, lo que se desea lograr con el procedimiento propuesto es que el diseñador pueda obtener una estructura reticular de cubierta que se aproxime a ser óptima desde el punto de vista estructural (enfocado en la energía de deformación y en el peso de la estructura) y acústico (enfocado en la distribución de la intensidad del sonido sobre el área bajo la estructura).

Para lograr estos cometidos, se planteó una integración entre distintos procesos (ver Figura 28):

- Por un lado, es necesario tener un procedimiento que genere automáticamente la geometría de la estructura reticular, pues de no tenerlo, el tiempo que tomaría definir manualmente dicha geometría, sería significativamente alto.

  Como resultado, se planteó y se desarrolló un proceso automático de generación de geometría, llevado a cabo por medio de la integración del software Rhinoceros[165]

[165] ROBERT MCNEEL & ASSOCIATES. Rhinoceros [en línea]. <http://www.rhino3d.com/> [citado el 14 de Febrero de 2013].

+ Grasshopper[166][167] + Lunchbox[168], con procedimientos ejecutados en MatLab[169].

- Adicionalmente, para llevar a cabo los procesos de evaluación del rendimiento estructural y acústico, se programaron códigos en MatLab que, tomando como base la información que se extrae del proceso automático de generación de geometría, son capaces de calcular la energía de deformación y el peso de una estructura reticular de cubierta, junto con la distribución de la intensidad de sonido que dicha estructura genera sobre el área que cubre.

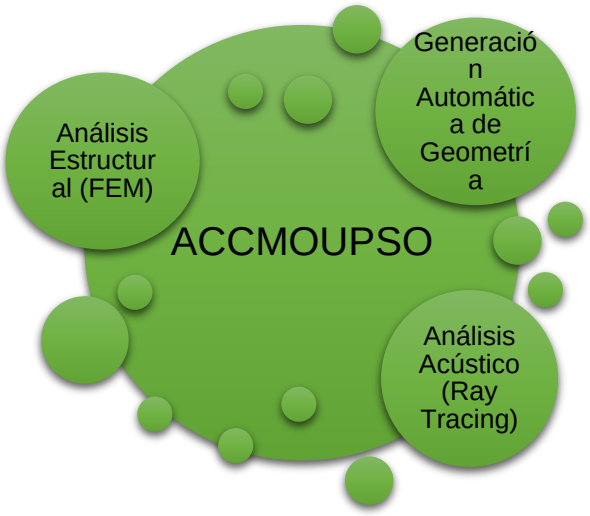
- Por último, era necesario incluir el acercamiento propuesto de optimización multi-objetiva (ACCMOUPSO), por medio del cual se deben integrar los procesos de generación automática de geometría, y los correspondientes a la evaluación del rendimiento estructural y acústico de la estructura. Todo esto con el fin de generar, por medio del procedimiento de optimización, la geometría de una cubierta reticular que se aproxime a ser óptima (con respecto a los objetivos considerados).

  El algoritmo de optimización genera dicha forma al tomar como variables la coordenada *z* de todos los nodos de la estructura reticular (sin incluir a los nodos de apoyo que permanecen fijos). Como es de esperarse, ACCMOUPSO va a ir variando y optimizando estas posiciones, y por consiguiente, va generando nuevas estructuras, con distintas geometrías, las cuales pasarán por el procedimiento de evaluación de rendimiento estructural y acústico. Como

[166] Para los diseñadores que desean explorar nuevas formas utilizando algoritmos generativos, Grasshopper es un editor gráfico de algoritmos, estrechamente integrado con las herramientas de modelado 3-D de Rhinoceros. Grasshopper no requiere un mayor conocimiento en el campo de la programación o en el desarrollo de scripts pero, de todas maneras, le permite a los diseñadores construir generadores de formas que pueden ser simples o llegar hasta la complejidad requerida o deseada (121).
[167] Grasshopper [en línea]. <http://www.grasshopper3d.com> [citado el 8 de Julio de 2013].
[168] LunchBox [en línea]. <http://www.grasshopper3d.com/group/lunchbox> [citado el 8 de Julio de 2013].
[169] MATHWORKS. MatLab [en línea]. <http://www.mathworks.com/> [citado el 8 de Julio de 2013].

resultado, este proceso de optimización se convierte en un procedimiento de morfogénesis que resulta en una estructura reticular con una configuración geométrica que presenta un alto rendimiento con respecto a todos los objetivos considerados. Vale la pena resaltar que el resultado estético al cual se llega, está controlado por el criterio estético del diseñador (como se explicará posteriormente).

Figura 28. Componentes de la herramienta de optimización



Fuente: Autor

Antes de pasar a explicar el funcionamiento del procedimiento propuesto, es necesario comentar que la intención es aplicarlo en la fase conceptual del diseño de la estructura reticular de cubierta. Es decir, a lo que se pretende llegar es a definir la configuración geométrica más idónea para una estructura que cubre una determinada planta arquitectónica (esto implica que se deben pre-definir las secciones transversales de los elementos y el material a utilizar, así éstas no vayan a ser exactamente las del diseño definitivo). Luego, posteriormente será necesario entrar a definir con precisión las secciones transversales de los

elementos, y corroborar que se cumplan las condiciones de diseño estructural que plantea la norma sismo-resistente que aplique en cada caso.

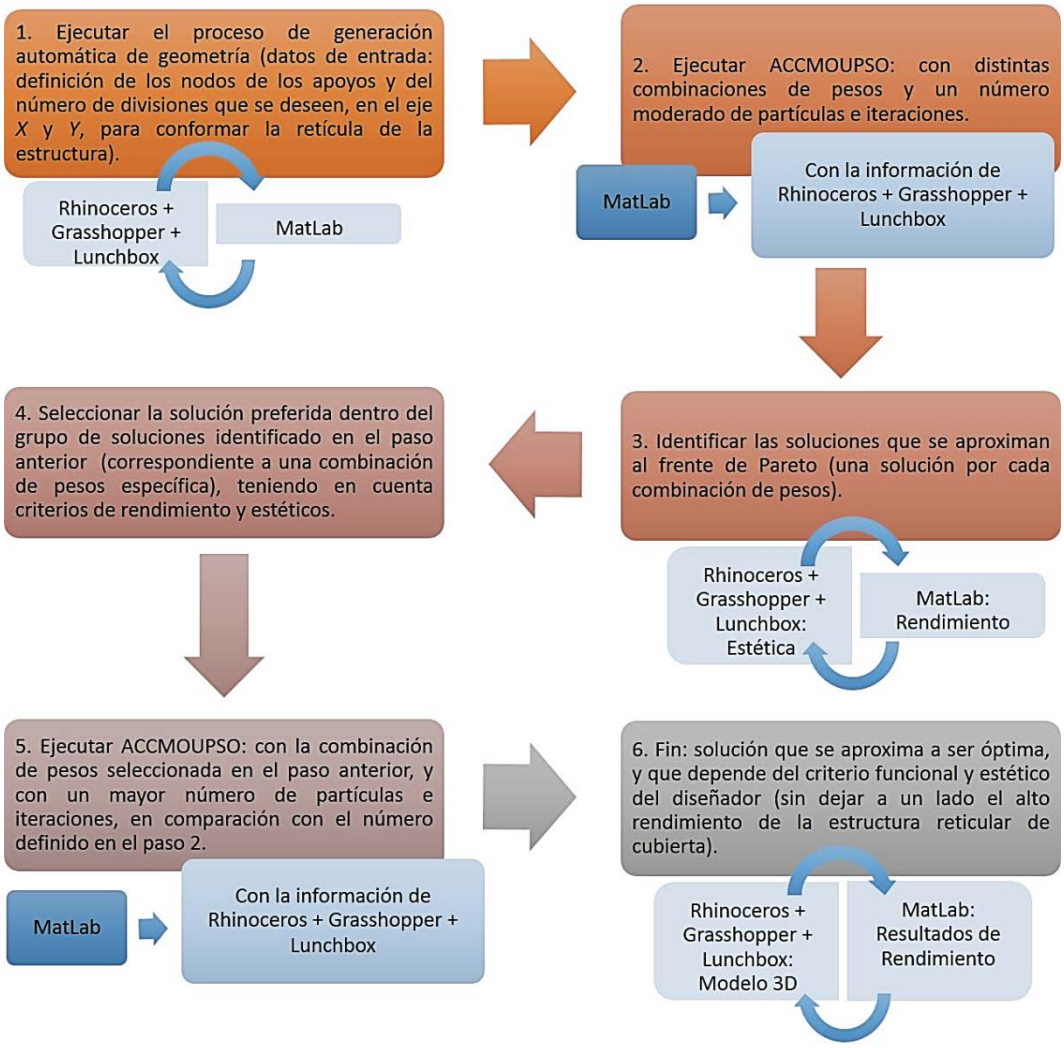
Ahora bien, el funcionamiento del procedimiento propuesto (PACDMOER) se explica a continuación, y se resume en la Figura 29 (en donde los recuadros numerados corresponden a cada uno de los pasos a seguir, mientras que los diagramas azules representan la correspondiente interacción entre los software utilizados en cada uno de ellos):

Con el fin de acercarse al frente de Pareto, ACCMOUPSO corre por primera vez con distintas combinaciones de pesos, pocas partículas, y durante un moderado número de iteraciones. Las soluciones inicialmente identificadas tendrán aproximadamente el mismo rendimiento global (para la combinación de los tres objetivos: energía de deformación + peso + rendimiento acústico), pero una configuración geométrica distinta.

Luego, el(la) diseñador(a) debe seleccionar la combinación de pesos preferida, teniendo en cuenta la configuración geométrica estéticamente más placentera, y la importancia que el(ella) desea darle a cada objetivo.

Como siguiente paso, ACCMOUPSO corre una vez más con la combinación de pesos seleccionada, muchas más partículas, y durante más iteraciones. En este sentido, la solución final dependerá del criterio funcional y estético del(de la) diseñador(a), sin dejar a un lado el rendimiento de la estructura reticular de cubierta.

Figura 29. Diagrama del Procedimiento Auto-Configurado para el Diseño Multi-Objetivo de Estructuras Reticulares (PACDMOER)



Fuente: Autor

Adicionalmente, en el Anexo E se encuentra el código en MatLab utilizado para llevar a cabo el procedimiento propuesto (PACDMOER); en dicha sección se expone el procedimiento de cálculo de manera más detallada.

3.1. GENERACIÓN AUTOMÁTICA DE LA GEOMETRÍA

La geometría inicial de la estructura reticular de cubierta es generada con Grasshopper[170] + LunchBox[171], un plug-in de Rhinoceros[172] y un plug-in de Grasshopper, respectivamente.

Con estos plug-ins, es posible programar la creación de los nodos de la estructura y la conectividad de Delaunay[173] entre ellos (la cual definirá la retícula de la estructura de cubierta), necesitando como datos de entrada, únicamente las coordenadas de los puntos de apoyo y el número deseado de divisiones en el plano *x-y* (con el fin de crear la retícula de la estructura).

En el desarrollo de este proceso, también es necesario el uso de MatLab[174], el cual interactúa con Grasshopper durante el proceso de generación automática de geometría, y genera aleatoriamente la coordenada *z* de cada uno de los nodos (estos valores serán las variables durante el proceso de optimización).

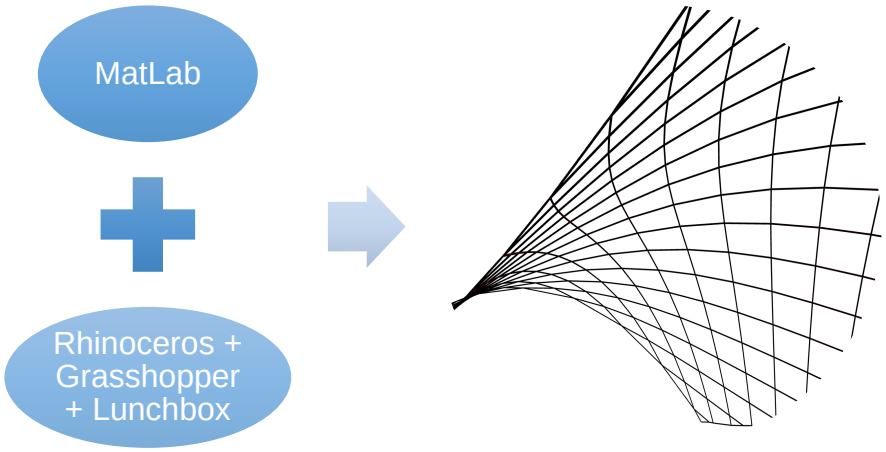
Consecuentemente, el resultado de este proceso son una serie de matrices, guardadas en MatLab, con la información de los nodos ordenados, sus coordenadas, y la información de la conectividad de Delaunay.

Estos datos serán utilizados durante el proceso de optimización multi-objetiva para el análisis estructural y acústico.

---

[170] Grasshopper [en línea]. <http://www.grasshopper3d.com> [citado el 8 de Julio de 2013].
[171] LunchBox [en línea]. <http://www.grasshopper3d.com/group/lunchbox> [citado el 8 de Julio de 2013].
[172] ROBERT MCNEEL & ASSOCIATES. Rhinoceros [en línea]. <http://www.rhino3d.com/> [citado el 14 de Febrero de 2013].
[173] DE BERG, M. Computational Geometry: Algorithms and Applications. Berlín: Springer, 2008.
[174] MATHWORKS. MatLab [en línea]. <http://www.mathworks.com/> [citado el 8 de Julio de 2013].

Figura 30. Componentes del proceso de generación automática de geometría.



Fuente: Autor

La definición desarrollada en Grasshopper + LunchBox se presenta en el Anexo A, mientras que el código escrito en MatLab se presenta en el Anexo B; este proceso está dividido en distintas partes para poder lograr la interacción entre las dos programaciones.

3.2. ANÁLISIS ESTRUCTURAL

En este caso, el propósito del análisis estructural es calcular la energía de deformación de cada estructura reticular de cubierta (correspondiente a cada partícula del ACCMOUPSO) con el fin de minimizarla durante el proceso de optimización multi-objetiva. Para esto, se programó en MatLab el Método de Elementos Finitos (FEM), siguiendo los planteamientos presentados por Chandrupatla y Belegundu[175] para elementos sometidos a cargas axiales (para mayores detalles, ver el Anexo C).

[175] CHANDRUPATLA, T. y BELEGUNDU, A. Introduction to finite elements in engineering. Upper Saddle River: Prentice Hall, 2011.

Finalmente, la energía de deformación se calculó con la ecuación (30), en la cual se fundamenta la primera función objetivo (minimizando la expresión allí expuesta):

$$E_s = (\{D\}[K])\{D\}' \tag{30}$$

donde $\{D\}$ es el vector de desplazamientos; $[K]$ la matriz de rigidez de la estructura reticular; y $\{D\}'$ es la transpuesta de $\{D\}$. Esta ecuación también puede verse de la siguiente manera:

$$E_s = \{F\}\{D\}' \tag{31}$$

donde $\{F\} = \{D\}[K]$ es el vector de fuerzas nodales.

Otro valor que debe ser calculado es el peso de la estructura, con el fin de minimizarlo. La información resultante del proceso automático de generación de geometría, es utilizada para obtener el volumen total de material utilizado en las barras de la estructura reticular. Luego, este valor es multiplicado por la densidad del acero y la aceleración de la gravedad, calculando así el peso total de la estructura, en el cual se fundamenta la segunda función objetivo (minimizando dicha expresión):

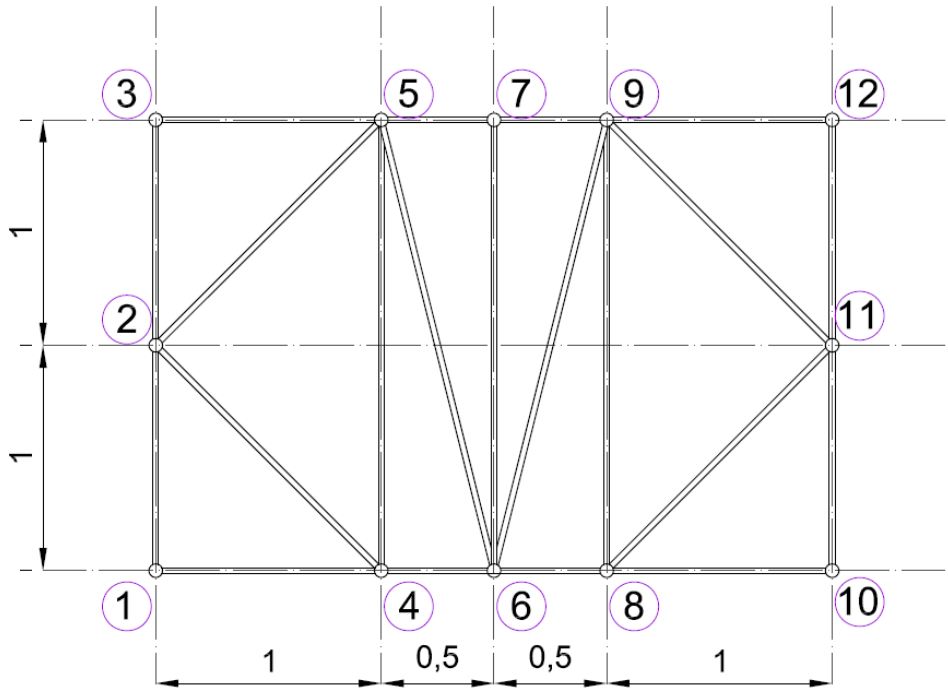
$$W = g * \rho_{steel} \sum_{i=1}^{n} V_{r_i} \tag{32}$$

donde g es la aceleración de la gravedad; ρ_{steel} es la densidad del acero; n es el número de barras; y V_{r_i} es el volumen de la $i - ésima$ barra. En el Anexo C se encuentra el código en MatLab utilizado para calcular las dos funciones objetivo, tanto la energía de deformación de la estructura como el peso de la misma. En dicha sección se expone el procedimiento de cálculo de manera más detallada.

3.2.1. Validación del análisis estructural

Para la validación del procedimiento de análisis estructural llevado a cabo en el código de programación, se realizó un ejemplo de comparación para contrastar los resultados obtenidos con los resultados provenientes del software de análisis estructural SAP2000[176]. El ejemplo consiste en una estructura reticular de 12 nodos, con apoyos articulados en todos aquellos de la periferia, exceptuando los nodos centrales (para un total de 10 apoyos); en dichos nodos (6 y 7), la estructura está cargada con una fuerza vertical en sentido de la gravedad, de 3433.5 N (ver Figura 31 y Figura 32).

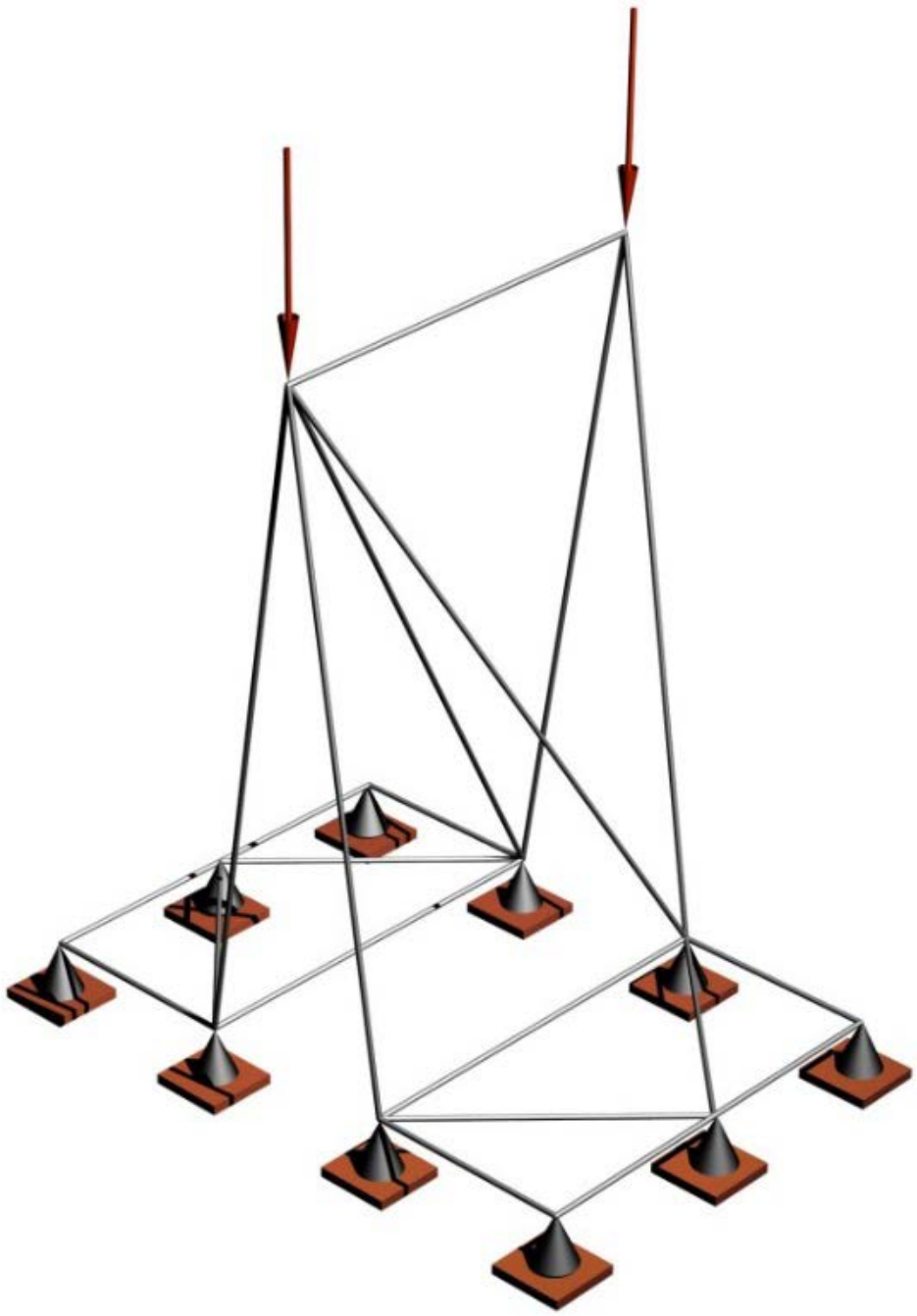
Figura 31. Planta de la estructura utilizada en la validación del análisis estructural - Dimensiones y numeración de nodos



Fuente: Autor

[176] COMPUTERS & STRUCTURES, INC. Sap2000 [en línea]. <http://www.csiamerica.com/sap2000> [citado el 24 de Septiembre de 2013].

Figura 32. Modelo 3D de la estructura utilizada en la validación del análisis estructural



Fuente: Autor

Los datos de entrada tanto para el modelo desarrollado en el código de programación, como para el modelo realizado en SAP2000, fueron los siguientes:

- Coordenadas de los nodos:

Cuadro 4. Coordenadas de los nodos de la estructura utilizada para la validación del análisis estructural.

Nodo	Coordenadas		
	X (m)	*Y (m)*	*Z (m)*
1	0.00	0.00	0.00
2	0.00	1.00	0.00
3	0.00	2.00	0.00
4	1.00	0.00	0.00
5	1.00	2.00	0.00
6	1.50	0.00	4.56
7	1.50	2.00	8.24
8	2.00	0.00	0.00
9	2.00	2.00	0.00
10	3.00	0.00	0.00
11	3.00	1.00	0.00
12	3.00	2.00	0.00

- Conectividad de los elementos:

Cuadro 5. Conectividad de los nodos de la estructura utilizada para la validación del análisis estructural.

Elemento	Nodo	
	i	*j*
1	2	1
2	4	1
3	3	2
4	4	2
5	5	2
6	5	3
7	5	4
8	6	4
9	6	5
10	7	5
11	7	6
12	8	6
13	9	6
14	9	7
15	9	8
16	10	8
17	11	8
18	11	9
19	12	9
20	11	10
21	12	11

- Material:

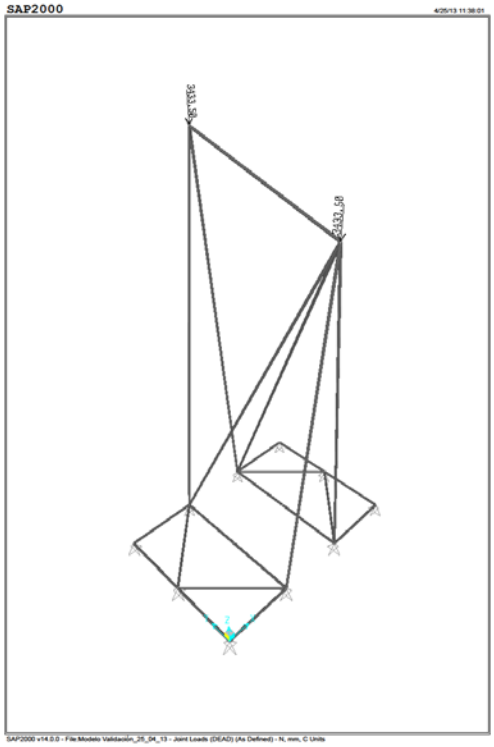
  Se utilizó un acero con módulo de elasticidad igual a 200000MPa, correspondiente al acero A992Fy50 (predeterminado en el software SAP2000).

- Sección transversal:

  La sección transversal utilizada es maciza y circular, con un diámetro de 2” (área de 506.7 mm^2).

Luego de llevar a cabo el análisis estructural utilizando el código de programación, y el software SAP2000 (ver Figura 33), los resultados a comparar se enfocaron en los desplazamientos de los nodos, debido a que éstos son el interés principal del análisis estructural en el proceso de optimización (ya que lo que se busca es poder obtener la energía de deformación de la estructura). En el Cuadro 6 se presentan dichos resultados.

Figura 33. Modelo de la estructura en el software de análisis estructural SAP2000.



Fuente: Autor

Cuadro 6. Resultados de la validación del análisis estructural (desplazamientos obtenidos con el código de programación y los correspondientes al software SAP2000).

Nodo	Grado de Libertad	Desplazamientos (mm)		Diferencia (%)
		Código de Programación	SAP2000	
6	*X*	0.000000	0.000000	0.0000
	Y	-0.179191	-0.179233	-0.0234
	Z	-0.078620	-0.078639	-0.0242
7	*X*	0.000000	0.000000	0.0000
	Y	-0.175303	-0.175345	-0.0240
	Z	-0.090005	-0.090027	-0.0244

En los resultados obtenidos, es posible observar que la diferencia entre el análisis estructural contenido en el código y el que se desarrolló por medio del software SAP2000, es prácticamente despreciable, estando entre el 0% y el -0.0244%. De esta manera, queda validado el análisis estructural programado y la exactitud de sus resultados.

3.3. ANÁLISIS ACÚSTICO

El análisis acústico se desarrolló siguiendo el trabajo de investigación realizado por Yang y Shield[177].

El primer paso es generar los receptores, los cuales son modelados como esferas de 0.60m de diámetro, lo que correspondería al espacio ocupado por una silla de un auditorio o de un teatro (sin embargo, Yang y Shield modelan los receptores

[177] YANG, L. y SHIELD, B. Development of a ray tracing computer model for the prediction of the sound field in long enclosures. En: Journal of Sound and Vibration. No. 229 (2000); p. 133-146.

como esferas de 1.00m de diámetro). Estos receptores son distribuidos uniformemente dentro del área cubierta por la estructura reticular.

El siguiente paso, es calcular el número requerido de rayos, el cual está determinado por[178]:

$$N_{ray} = \frac{10 * V_{space}}{V_{receiver}} \quad (33)$$

donde V_{space} es el volumen del espacio cubierto, y $V_{receiver}$ es el volumen de un receptor. Basados en este número, los rayos deben ser generados con direcciones aleatorias (hay otro método propuesto por Krokstad et al.[179] para generar las direcciones de los rayos). Por otro lado, se plantea que todos los rayos iniciales tienen su origen en las coordenadas pre-definidas por el usuario (lo que correspondería a las coordenadas de la fuente de sonido), mientras que su energía inicial, definida para una fuente omni-direccional (es decir, en donde el sonido se genera con igual intensidad en todas las direcciones), está determinada por la ecuación (34)[180]:

$$E_0 = \frac{10^{Lw/10}}{N_{ray}} * 10^{-12} \quad (34)$$

donde L_w (medido en dB) es el nivel de potencia del sonido de la fuente omni-direccional.

Por otro lado, los planos triangulares definidos por la conectividad de Delaunay (es decir, los planos de la cubierta), los planos verticales que definen los muros, y el

[178] ONDET, A. y BARBRY, J. Modelling of sound propagation in fitted workshops using ray tracing. En: Journal of the Acoustical Society of America. No. 85 (1989); p. 787-796.
[179] KROKSTAD, A., STROMS, S. y SOORSDAL, S. Calculating the acoustical room response by the use of a ray tracing technique. En: Journal of Sound and Vibration. No. 8 (1968); p. 118-125.
[180] YANG, L. y SHIELD, B. Development of a ray tracing computer model for the prediction of the sound field in long enclosures. En: Journal of Sound and Vibration. No. 229 (2000); p. 133-146.

plano horizontal (podría tomarse un plano inclinado, si así se desea) de piso, deben ser generados con la información resultante del procedimiento de generación automática de la geometría.

Para poder continuar con el proceso, calculando los puntos de intersección y la reflexión de los rayos, es necesario plantear las ecuaciones de todos los planos que definen el espacio, las de las esferas que definen a los receptores, y las de los planos que definen cada rayo (teniendo en cuenta que cada rayo es definido por la intersección de dos planos).

En este sentido, se calculan los puntos iniciales de intersección rayo-plano, y la energía inicial de sonido acumulada en los receptores. Para esto, la intensidad sentida por uno de los volúmenes receptores está definida por la ecuación (35)[181]:

$$E(t) = \frac{E' * d_{cell}}{V_{receiver}} \quad (35)$$

donde d_{cell} es la distancia que viaja el rayo dentro del volumen del receptor, y E' es la energía del rayo cuando éste llega al receptor, definida por la ecuación (36)[182]:

$$E' = E_0 e^{-h*d} \prod_i (1 - \alpha_i) \quad (36)$$

donde h es el coeficiente de absorción por atenuación del aire; d es la distancia total que viaja el rayo, desde la fuente hasta el punto en donde se desea medir la energía actual del rayo (incluyendo todos los caminos de todas las reflexiones); y α_i es el coeficiente de absorción de la superficie del $i - ésimo$ plano, en donde se ha reflejado el rayo.

[181] YANG, L. y SHIELD, B. Development of a ray tracing computer model for the prediction of the sound field in long enclosures. En: Journal of Sound and Vibration. No. 229 (2000); p. 133-146.
[182] Ibid.

Luego, el propósito es generar una tabla (propuesta en el presente proyecto de investigación, ver Cuadro 7) por cada reflexión de los rayos. La tabla deberá tener una fila por rayo, y la siguiente información en las columnas: coordenadas del punto de intersección, E_0, E', los componentes del vector director de cada rayo, los componentes del vector normal del plano en el cual se reflejó el rayo, y las coordenadas del punto de origen de cada rayo.

Cuadro 7. Cuadro propuesto para el desarrollo del algoritmo Ray-Tracing.

Rayo	Punto de Intersección Actual			E_0	E'	Vector Director del Rayo			Vector Normal del Plano de Reflexión			Punto de Origen del Rayo		
	x	y	z			x	y	z	x	y	z	x	y	z
1														
2														
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
N_{rayo}														

Con la información del Cuadro 7, se puede calcular el siguiente punto de intersección; luego, será posible actualizar toda la tabla para cada una de las reflexiones rayo-plano. Después de la reflexión inicial, el punto de origen de cada rayo será el punto de intersección anterior, y la nueva dirección de cada rayo se puede calcular sabiendo que el ángulo de incidencia es igual al ángulo de reflexión.

Adicionalmente, otra tabla debe ser generada (propuesta en el presente proyecto de investigación, ver Cuadro 8) con la información de los receptores. En este caso, la tabla debe tener una fila por receptor y la energía acumulada como columna. La energía puede ser calculada con la información del Cuadro 7, y las ecuaciones

(34), (35) y (36). Esta tabla debe ser actualizada con cada reflexión, sumando la energía correspondiente de cada intersección rayo-receptor.

Cuadro 8. Cuadro propuesto para la acumulación de energía en los receptores

Receptor	Energía Acumulada
1	
2	
⋮	⋮
$N_{receptor}$	

Finalmente, el criterio para finalizar este proceso se basa en el porcentaje de discontinuidad de energía (Energy Discontinuity Percentage, EDP)[183]. Esta medida representa el porcentaje de la pérdida de energía de un rayo antes de terminar el seguimiento del mismo[184]. En este caso, el EDP va a ser del 90% (Yang y Shield plantean que el rango para este valor debe estar entre el 90% y el 99%). Como resultado, el proceso correrá mientras que el valor de la energía actual de los rayos, E', sea mayor que $0.1 * E_0$ (el 10% de la energía inicial de los rayos).

De esta manera, la tercera función objetivo se plantea en la ecuación (37):

$$minimizar \sum_i^n \left| E_{receiver_i} - \bar{E}_{receivers} \right| \quad (37)$$

[183] DANCE, S. The development of computer models for the prediction of sound distribution in fitted non-diffuse spaces. Londres, 1993. Tesis doctoral. South Bank University.
[184] YANG, L. y SHIELD, B. Development of a ray tracing computer model for the prediction of the sound field in long enclosures. En: Journal of Sound and Vibration. No. 229 (2000); p. 133-146.

donde $E_{receiver_i}$ es la energía acumulada en el $i - ésimo$ receptor, luego de haber concluido el proceso de seguimiento de los rayos, y $\bar{E}_{receivers}$ es el promedio de la energía acumulada en los receptores, luego de finalizar el mismo proceso anterior. Por consiguiente, la tercera función objetivo busca una distribución uniforme de la energía acumulada en los receptores.

En el Anexo D se encuentra el código en MatLab utilizado para calcular la función objetivo expuesta anteriormente (sumatoria de las diferencias entre la energía acumulada en los receptores y el promedio de la misma). En dicha sección se expone el procedimiento de cálculo de manera más detallada.

3.3.1. Validación del análisis acústico

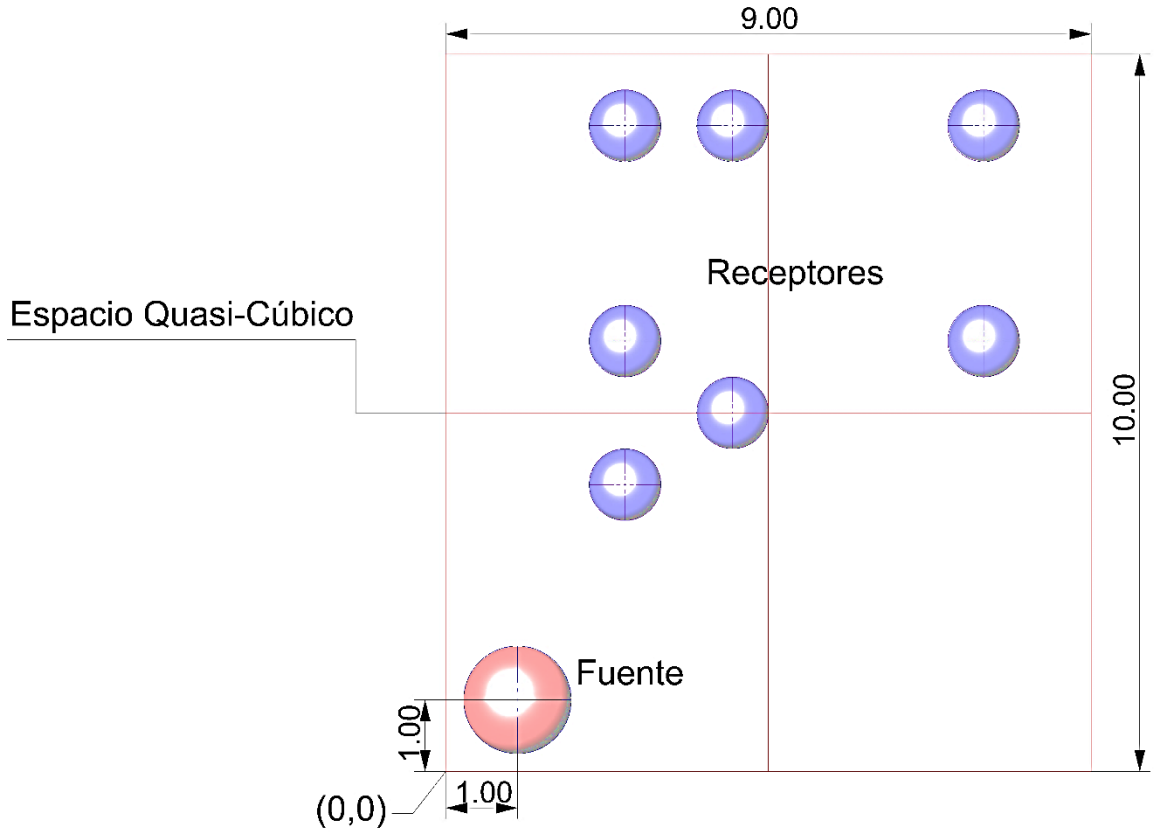
Con el fin de validar el código desarrollado para el análisis acústico de la estructura reticular de cubierta, se llevó a cabo un modelo de ejemplo presentado por Yang y Shield[185]. En este caso, el espacio corresponde a un volumen quasi-cúbico, con dimensiones en planta de 10mx9m, y una altura de 8m (ver Figura 34 y Figura 35). Por otro lado, el coeficiente de absorción del material de las seis caras del prisma, se definió igual a 0.10 (el coeficiente de atenuación del aire no estaba definido en el ejemplo de Yang y Shield, luego se tomó un coeficiente de 2.77 dB/km), mientras que la fuente de sonido, definida con 90dB de nivel de sonido, se ubicó a 1m de distancia de los tres planos más cercanos que la rodean (ver Figura 34 y Figura 35). En cuanto a los receptores, estos se modelaron como esferas de 0.50m de radio, y se ubicaron en las coordenadas que se muestran en el Cuadro 9.

[185] YANG, L. y SHIELD, B. Development of a ray tracing computer model for the prediction of the sound field in long enclosures. En: Journal of Sound and Vibration. No. 229 (2000); p. 133-146.

Cuadro 9. Coordenadas de los receptores para el ejemplo de la validación acústica.

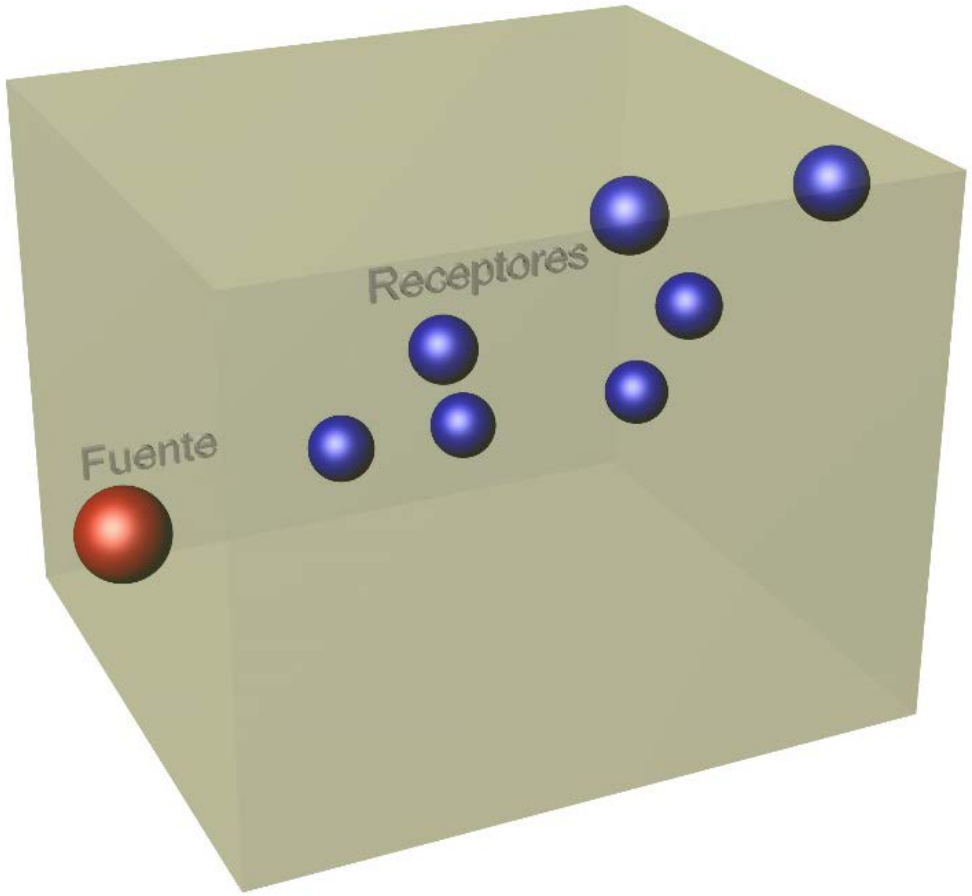
Coordenada [m]	Receptores						
	A	B	C	D	E	F	G
X	4.00	4.00	2.50	2.50	2.50	7.50	7.50
Y	5.00	9.00	4.00	9.00	6.00	6.00	9.00
Z	4.50	4.50	2.50	2.50	2.50	7.50	7.50

Figura 34. Planta del espacio quasi-cúbico, utilizado como ejemplo en la validación acústica.



Fuente: Autor

Figura 35. Perspectiva del espacio quasi-cúbico, utilizado como ejemplo en la validación acústica.



Fuente: Autor

Luego de correr el código desarrollado para el análisis acústico, se obtuvieron los resultados que se muestran en el Cuadro 10, en donde se comparan con los resultados presentados por Yang y Shield[186].

[186] YANG, L. y SHIELD, B. Development of a ray tracing computer model for the prediction of the sound field in long enclosures. En: Journal of Sound and Vibration. No. 229 (2000); p. 133-146.

Cuadro 10. Resultados del ejemplo de validación acústica, obtenidos por Yang y Shield, y por el código de análisis acústico desarrollado.

Receptor	**Nivel de la Potencia de Sonido [dB]**		**Diferencia [%]**
	Yang y Shield	Código Desarrollado	
A	78.82	77.61	-1.53
B	78.75	77.63	-1.42
C	79.03	77.75	-1.62
D	78.76	77.68	-1.37
E	78.85	77.63	-1.55
F	78.76	77.60	-1.47
G	78.72	77.61	-1.41

Como se puede observar, las diferencias entre los resultados son mínimas, del orden de 1.50%, y probablemente se presentan debido al coeficiente de atenuación del aire, ya que se tomó dicho valor arbitrariamente debido a que en el ejemplo de los autores de referencia, este coeficiente no estaba definido (de todas formas, es posible ver que la influencia de este coeficiente no es significativa). En conclusión, basándose en los resultados obtenidos, puede decirse que el código para el análisis acústico queda validado.

4. APLICACIÓN DEL PROCEDIMIENTO AUTO-CONFIGURADO PARA EL DISEÑO MULTI-OBJETIVO DE ESTRUCTURAS RETICULARES (PACDMOER): EJEMPLOS NUMÉRICOS

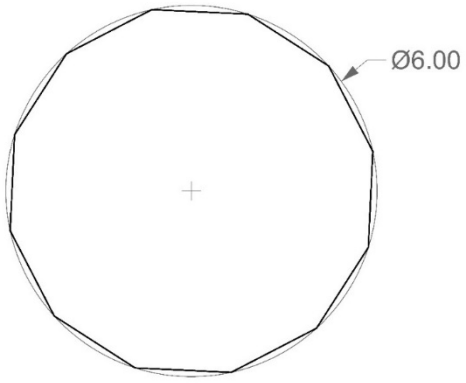
Con el fin de mostrar la convergencia y el funcionamiento del procedimiento de diseño propuesto, se plantearon 6 ejemplos, o problemas de prueba. Los tres primeros están enfocados en la optimización de un solo objetivo, es decir, un problema de prueba en donde se busca minimizar únicamente el peso de la estructura, otro en donde la minimización hace referencia a la energía de deformación, y un tercero que pretende minimizar la diferencia de la intensidad de sonido (energía) acumulada en los receptores distribuidos uniformemente sobre el área que cubre la estructura. Posteriormente, se llevaron a cabo dos ejemplos más en donde se empiezan a combinar los objetivos: un problema de prueba en donde se minimiza, simultáneamente, el peso y la energía de deformación de la estructura, y otro en donde se incluyen los tres objetivos, i.e. peso + energía de deformación + rendimiento acústico.

La intención de estos cinco primeros ejemplos es mostrar la convergencia del algoritmo, por lo tanto, no se hicieron variaciones en los pesos de los objetivos; por consiguiente, tampoco se llevó a cabo la interacción entre el diseñador y el código de programación. Adicionalmente, estos problemas de prueba se basaron en la misma definición geométrica y en las mismas condiciones estructurales. Finalmente, el último ejemplo se realizó con una configuración geométrica distinta a la utilizada en los cinco problemas de prueba anteriores. En él se tuvieron en cuenta los tres objetivos (peso + energía de deformación + rendimiento acústico), se corrió el algoritmo con distintas combinaciones de peso, y se llevó a cabo la interacción entre el diseñador y el procedimiento de diseño. En resumen, este último problema de prueba se hizo para mostrar el uso de PACDMOER en su totalidad.

4.1. EJEMPLOS DE CONVERGENCIA

Para los ejemplos en donde se muestra la convergencia del algoritmo, se utilizó una configuración geométrica definida por una planta en forma de dodecaedro, el cual está inscrito en un círculo con un diámetro de 6.00m (ver Figura 36).

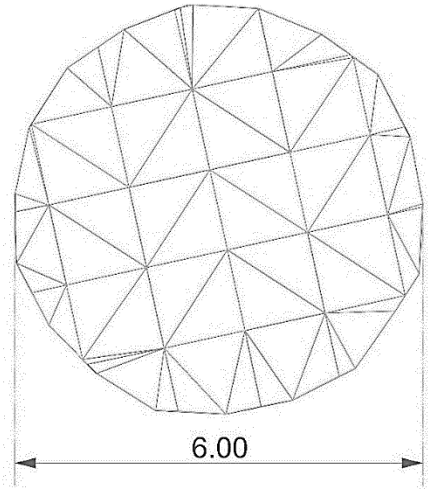
Figura 36. Perímetro que define la forma de la planta de la estructura reticular de cubierta – Ejemplos de convergencia.



Fuente: Autor

La estructura se definió con 72 nodos y 122 barras (ver Figura 37), a las cuales se les asignó una sección transversal circular de $1/2''$ de diámetro (correspondiente a un área de 126.7 mm^2), y un acero con módulo de elasticidad de 200000 MPa y densidad de 7800 kg/m^3. En cuanto a las cargas aplicadas, todos los nodos fueron solicitados con una carga vertical de 1000 N en dirección de la gravedad.

Figura 37. Planta de la estructura reticular de cubierta – Ejemplos de convergencia.

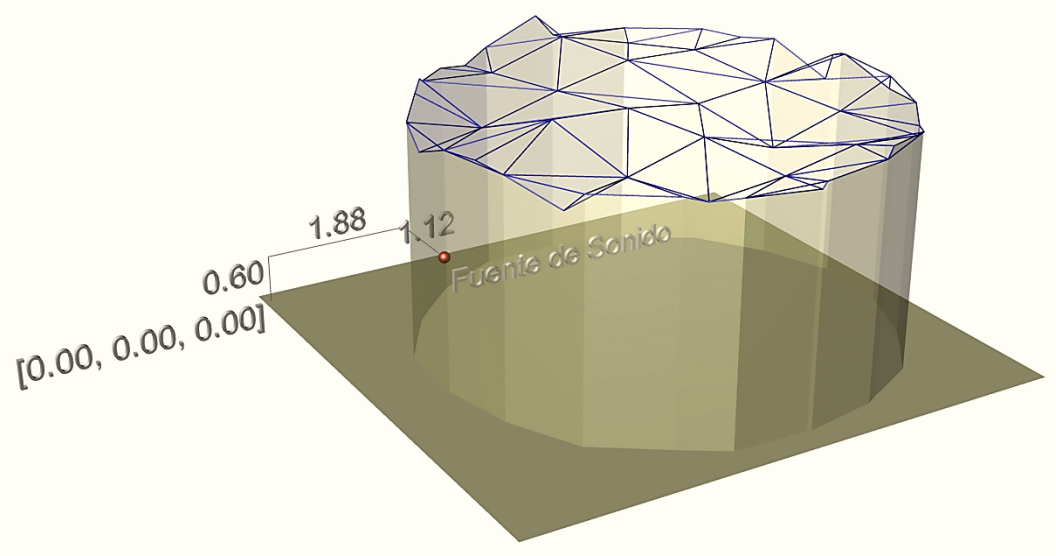


Fuente: Autor

Adicionalmente, para la evaluación del rendimiento acústico se definieron las coordenadas de la fuente de sonido, en metros, como $[1.12, 1.88, 0.60]$ (ver Figura 38), mientras que el nivel de la potencia de sonido (Sound Power Level – SPL) de ésta fue de 90 dB. Por otro lado, el coeficiente de absorción del aire se tomó como 2.77 dB/km (correspondiente a una temperatura de 20°C, una humedad relativa del 80%, y una frecuencia de preferencia de 500 Hz)[187], y el coeficiente de absorción del material de los distintos planos (muros, piso y cubierta) se definió como 0.17 (correspondiente a paneles de plywood para una frecuencia de preferencia de 500 Hz)[188].

[187] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION - ISO. Attenuation of sound during propagation outdoors, Part 1: Calculation of the absorption of sound by the atmosphere. Geneve: International Organization for Standardization, 1993. (ISO 9613-1).
[188] MONKS, M.C. Audioptimization: Goal-Based Acoustic Design. Massachusetts, 1999. Tesis doctoral. Massachusetts Institute of Technology.

Figura 38. Ubicación de la fuente de sonido dentro del espacio cubierto por la estructura reticular.



Fuente: Autor

Para finalizar con la definición geométrica de los ejemplos en donde se muestra la convergencia del algoritmo, las restricciones aplicadas a la coordenada z de cada nodo (las variables durante el proceso de morfogénesis), se plantearon para delimitar un rango espacial, i.e. una altura máxima y una mínima. En los dos ejemplos, ésta restricción se definió en función de las dimensiones en planta, para que el resultado formal final tuviera la libertad de formar tipologías de bóvedas o cúpulas (dejando un margen de movilidad considerable).

Por último, el PSO corrió dentro del ACCMOUPSO con 10 partículas durante 100 iteraciones y con la siguiente definición de parámetros, tomada de Toscano Pulido et al.[189]**:**

[189] TOSCANO-PULIDO, G., SANTANA-QUINTERO, L.V. y COELLO COELLO, C.A. EMOPSO: A Multi-Objective Particle Swarm Optimizer with Emphasis on Efficiency. En: EMO 2007. Proceedings of EMO 2007. Heidelberg: Springer, 2007. p. 272-285.

- $X = 0.5$
- $C_1 = 1.4$
- $C_2 = 1.4$

En cambio, el UPSO multiobjetivo corrió dentro del PSO descrito anteriormente, con 10 partículas durante 10 iteraciones. Esta asignación de un número moderado de partículas e iteraciones corresponde al alto tiempo computacional requerido en la ejecución de ACCMOUPSO.

4.1.1. Minimización del peso

Para llevar a cabo este ejemplo, se corrió el ACCMOUPSO diez veces (con el fin de mostrar la convergencia) con una definición de pesos de la siguiente manera: w1 (el peso correspondiente al objetivo de energía de deformación) = 0, w2 (el peso correspondiente al objetivo de peso) = 1, y w3 (el peso correspondiente al objetivo de rendimiento acústico) = 0.

Para este caso, el PSO dentro del ACCMOUPSO se demoró 1162.84 segundos en encontrar los parámetros que se aproximan a ser óptimos para el UPSO (en su aplicación específica en este problema de prueba), siendo éstos los siguientes:

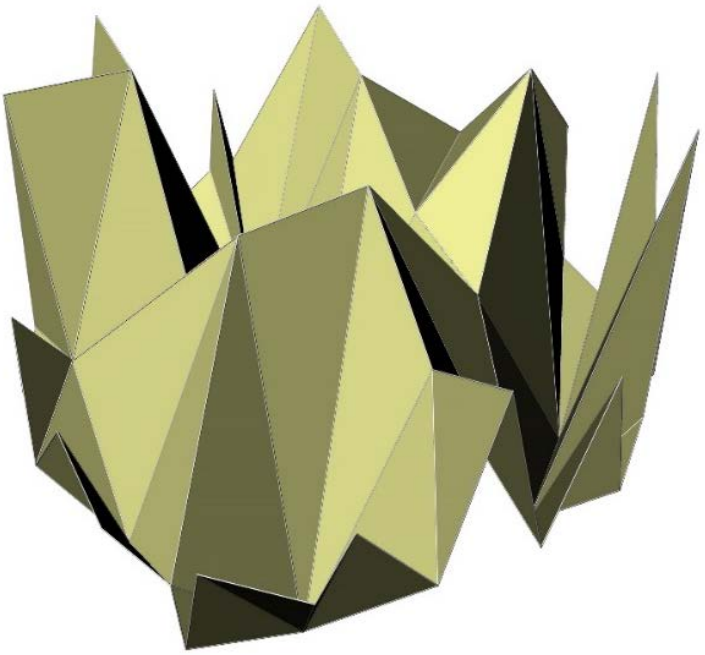
- $X = 0.4677$
- $C_1 = 1.8024$
- $C_2 = 1.3156$
- $u = 0.5456$

Luego de haber encontrado estos parámetros, el UPSO multi-objetivo (fase final de ACCMOUPSO) corrió con 1000 partículas durante 50 iteraciones y con la misma definición de pesos que se describió al inicio. Los resultados obtenidos son los que muestran en el Cuadro 11 y en la Figura 40.

A partir de ellos, es posible observar que el ACCMOUPSO efectivamente converge, presentando una desviación estándar de 6.61 N (0.59% del promedio) en los resultados finales obtenidos.

Adicionalmente, también es posible ver la efectividad en el proceso de optimización del peso de la estructura, alcanzando una reducción promedio de 55.70% (en comparación con las estructuras reticulares iniciales generadas aleatoriamente, ver Figura 39), en un tiempo promedio de 511.23 segundos.

Figura 39. Ejemplo de una de las estructuras iniciales en el proceso de optimización (generada aleatoriamente).

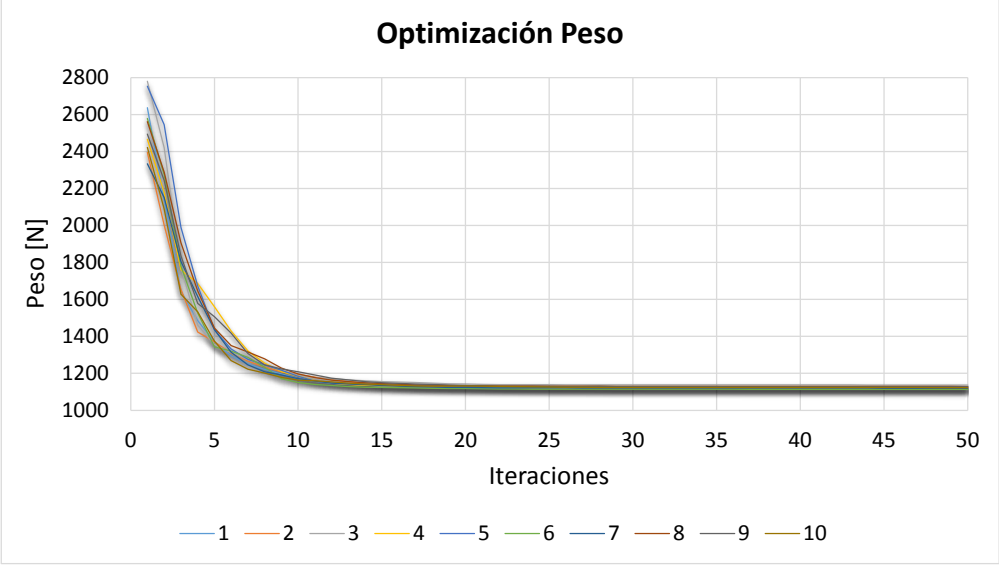


Fuente: Autor

Cuadro 11. Resultados de la convergencia de ACCMOUPSO al optimizar únicamente el peso de la estructura (correspondiente a 10 corridas distintas).

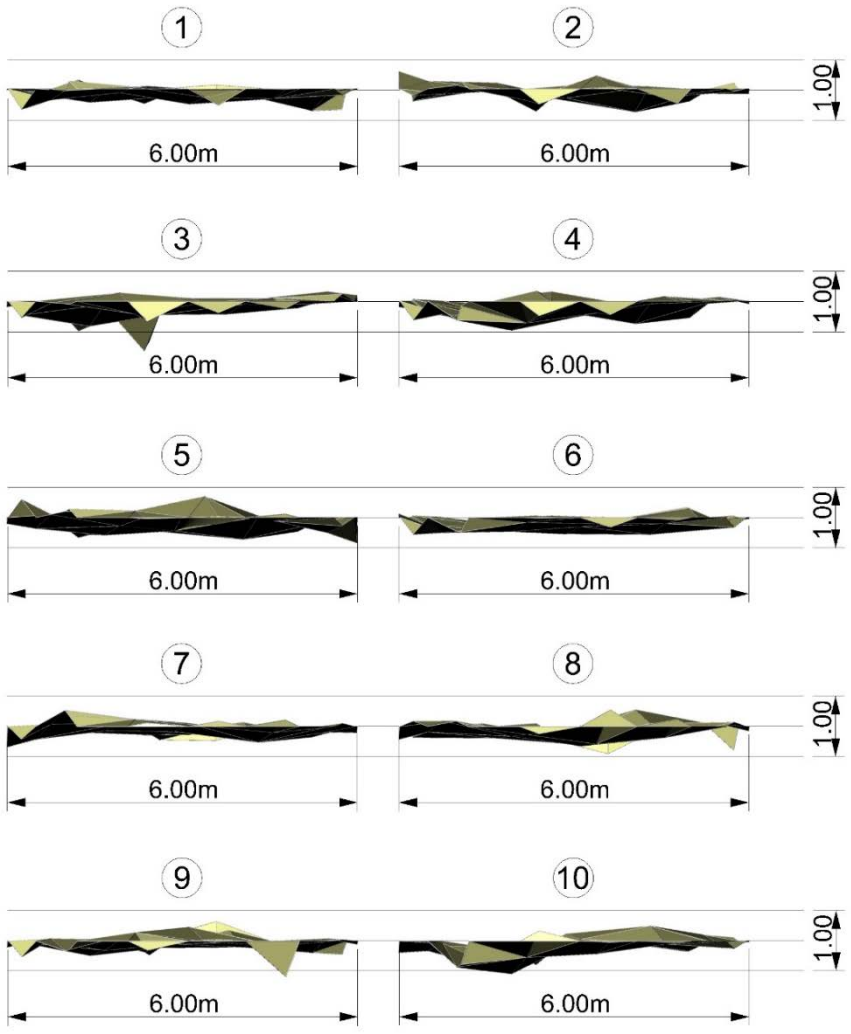
Corrida	Peso Inicial [N]	Peso Final [N]	Diferencia [%]	Tiempo [seg]
1	2636.73	1117.32	-57.62	502.95
2	2395.80	1121.81	-53.18	499.65
3	2780.36	1136.74	-59.12	486.79
4	2466.12	1119.74	-54.60	471.04
5	2753.98	1124.52	-59.17	469.87
6	2578.63	1113.27	-56.83	483.26
7	2334.30	1119.35	-52.05	492.10
8	2561.17	1128.76	-55.93	494.23
9	2494.82	1127.11	-54.82	571.68
10	2422.72	1122.86	-53.65	640.76
Promedio:	2542.46	1123.15	-55.70	511.23

Figura 40. Convergencia de ACCMOUPSO al optimizar únicamente el peso de la estructura (correspondiente a 10 corridas distintas, ver Cuadro 11).



Fuente: Autor

Figura 41. Vista frontal de las estructuras de cubierta correspondientes al resultado final de la convergencia de ACCMOUPSO para el objetivo de peso (correspondientes a 10 corridas distintas, ver Cuadro 11 y Figura 40).

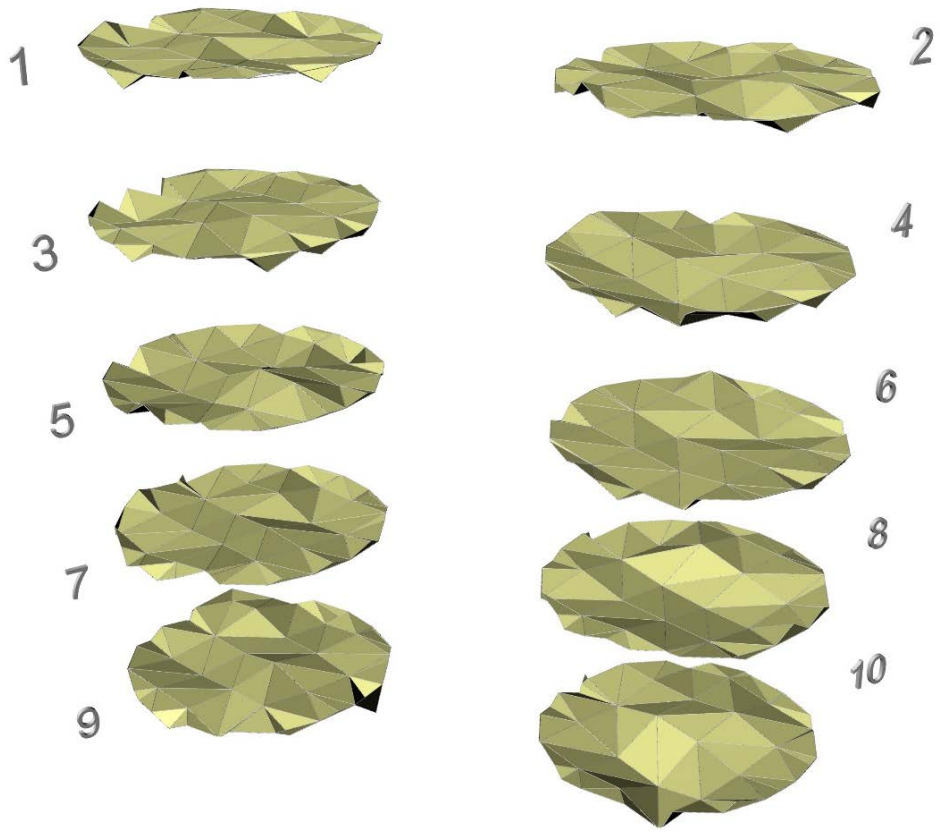


Fuente: Autor

En cuanto al resultado formal de este primer ejemplo, las configuraciones geométricas de las estructuras reticulares de cubierta obtenidas al ejecutar ACCMOUPSO, son las que se muestran en la Figura 41 y en la Figura 42. En

estas imágenes es posible observar que las estructuras tienden a ser planas, lo cual tiene coherencia con la función objetivo del peso (ver ecuación (32)), ya que depende de la sumatoria de los volúmenes de las barras y ésta está en función de las longitudes de las mismas; por ende, al minimizar la función objetivo del peso, lo que se hace es buscar las menores longitudes de las barras, y esta condición se logra cuando las estructuras tienden a ser planas.

Figura 42. Perspectiva de las estructuras de cubierta correspondientes al resultado final de la convergencia de ACCMOUPSO para el objetivo de peso (correspondientes a 10 corridas distintas, ver Cuadro 11 y Figura 40).



Fuente: Autor

4.1.2. Minimización de la energía de deformación

Para el desarrollo de este ejemplo, se corrió el ACCMOUPSO diez veces (con el fin de mostrar la convergencia) con una definición de pesos de la siguiente manera: w1 (el peso correspondiente al objetivo de energía de deformación) = 1, w2 (el peso correspondiente al objetivo de peso) = 0, y w3 (el peso correspondiente al objetivo de rendimiento acústico) = 0.

En este caso, el PSO dentro del ACCMOUPSO se demoró 1385.88 segundos en encontrar los parámetros que se aproximan a ser óptimos para el UPSO (en su aplicación específica en este problema de prueba), siendo éstos los siguientes:

- $X = 0.5627$
- $C_1 = 1.1247$
- $C_2 = 1.1552$
- $u = 0.6434$

Después de encontrar estos parámetros, el UPSO multi-objetivo (fase final de ACCMOUPSO) corrió con 1000 partículas durante 50 iteraciones y con la misma definición de pesos que se describió al inicio.

Los resultados obtenidos son los que muestran en el Cuadro 12, en la Figura 43 y en la Figura 44 (en esta figura se puede evidenciar de manera más clara la convergencia del algoritmo, debido a que en la Figura 43 la escala no permite ver con precisión el comportamiento de éste objetivo a lo largo de las 50 iteraciones).

A partir de estos resultados, se puede observar que el ACCMOUPSO efectivamente converge, presentando una desviación estándar de 368.30 N*mm (12.2% del promedio) en los resultados finales obtenidos; el anterior valor es significativamente mayor que el correspondiente a la convergencia del peso de la estructura (ver numeral 4.1.1), y esto se debe a que la energía de deformación

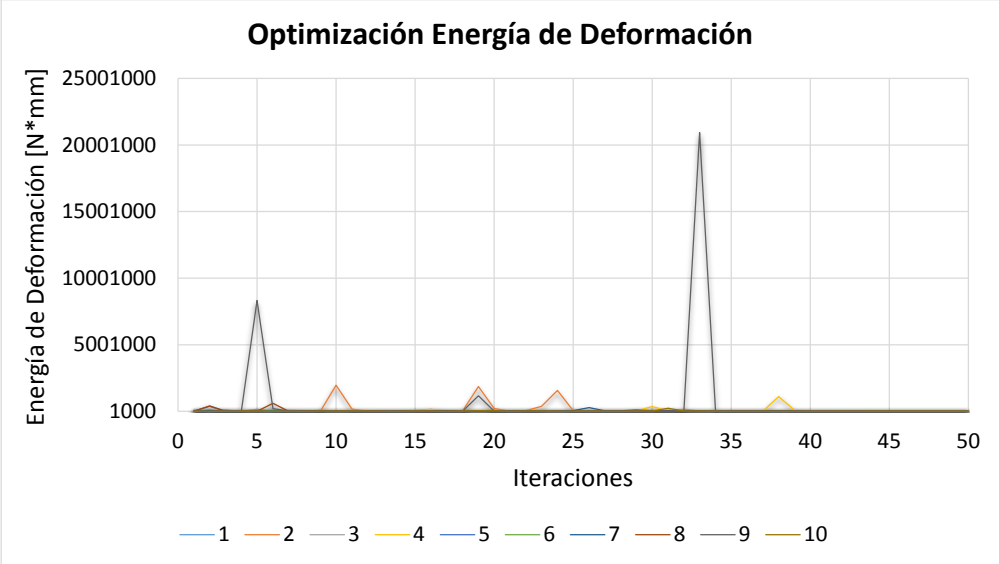
presenta un comportamiento mucho más variable, tanto en escala como en tendencia.

Adicionalmente, también es posible ver la efectividad en el proceso de optimización de la energía de deformación de la estructura, alcanzando una reducción promedio de 80.26% (en comparación con las estructuras reticulares iniciales generadas aleatoriamente, ver Figura 39), en un tiempo promedio de 509.56 segundos.

Cuadro 12. Resultados de la convergencia de ACCMOUPSO al optimizar únicamente la energía de deformación de la estructura (correspondiente a 10 corridas distintas).

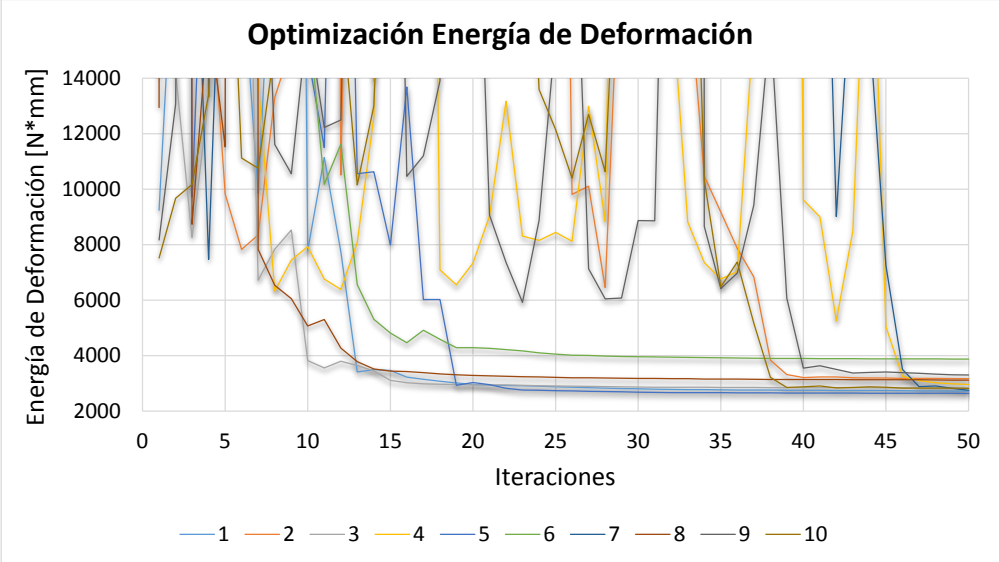
Corrida	Energía de Deformación Inicial [N*mm]	Energía de Deformación Final [N*mm]	Diferencia [%]	Tiempo [seg]
1	9241.40	2721.84	-70.55	490.87
2	32550.22	3161.88	-90.29	497.05
3	149347.39	2830.93	-98.10	494.92
4	18800.67	2959.17	-84.26	500.84
5	31288.18	2639.53	-91.56	494.95
6	36465.16	3874.67	-89.37	502.43
7	14106.01	2755.45	-80.47	553.65
8	12944.07	3114.01	-75.94	501.16
9	8173.25	3302.39	-59.60	531.34
10	7516.71	2823.58	-62.44	528.40
Promedio:	32043.31	3018.35	-80.26	509.56

Figura 43. Convergencia de ACCMOUPSO al optimizar únicamente la energía de deformación de la estructura (correspondiente a 10 corridas distintas, ver Cuadro 12).



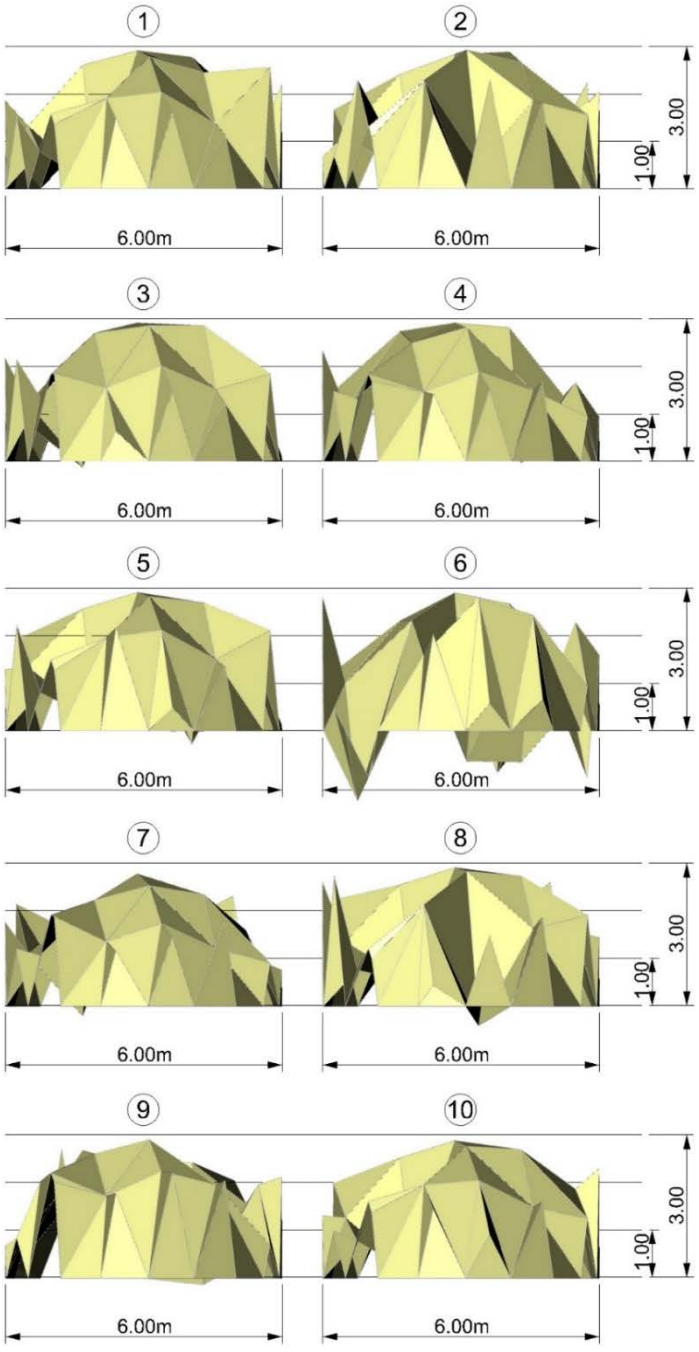
Fuente: Autor

Figura 44. Acercamiento de la Figura 43.



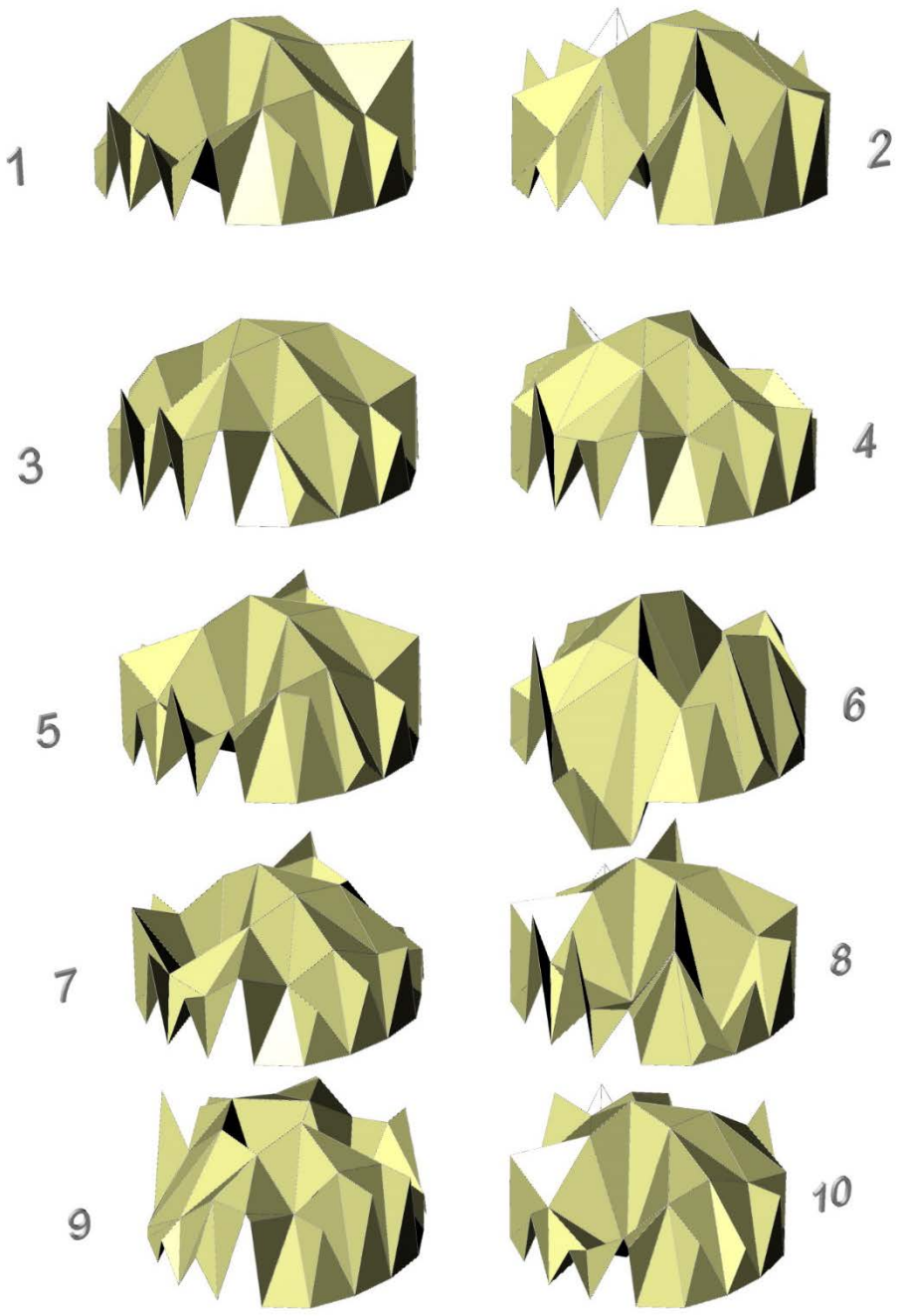
Fuente: Autor

Figura 45. Vista frontal de las estructuras de cubierta correspondientes al resultado final de la convergencia de ACCMOUPSO para el objetivo de energía de deformación (correspondientes a 10 corridas distintas, ver el Cuadro 12, la Figura 43 y la Figura 44).



Fuente: Autor

Figura 46. Perspectiva de las estructuras de cubierta correspondientes al resultado final de la convergencia de ACCMOUPSO para el objetivo de energía de deformación (correspondientes a 10 corridas distintas, ver el Cuadro 12, la Figura 43 y la Figura 44).



Fuente: Autor

En lo referente al resultado formal del ejemplo de convergencia de la energía de deformación, las configuraciones geométricas de las estructuras reticulares de cubierta obtenidas al ejecutar ACCMOUPSO, son las que se muestran en la Figura 45 y en la Figura 46.

En estas imágenes es posible observar que las estructuras tienden hacia la tipología estructural de domo, lo cual corresponde con los resultados encontrados por Richardson et al.[190], y al reconocimiento que se le ha dado a dicha tipología estructural, debido a su eficiencia, a lo largo de la historia: desde el Panteón romano, pasando por la basílica de Santa María del Fiore, y llegando a ejemplos más recientes como el Eden Project.

4.1.3. Optimización del rendimiento acústico

Debido al alto costo computacional en la evaluación del rendimiento acústico de las estructuras de cubierta, en este ejemplo se corrió el ACCMOUPSO únicamente tres veces (con el fin de mostrar la convergencia), con una definición de pesos de la siguiente manera: w1 (el peso correspondiente al objetivo de energía de deformación) = 0, w2 (el peso correspondiente al objetivo de peso) = 0, y w3 (el peso correspondiente al objetivo de rendimiento acústico) = 1.

En este problema de prueba se cambió el número de partículas y de iteraciones para la primera fase de ACCMOUPSO (debido al alto costo computacional), i.e. para la búsqueda de los mejores parámetros para el UPSO por parte del PSO; el PSO corrió con 4 partículas durante 20 iteraciones, mientras que el UPSO corrió, dentro del PSO, con 3 partículas durante 6 iteraciones.

[190] RICHARDSON, J.N., ADRIAENSSENS, S., COELHO, R.F. y BOUILLARD, P. Coupled form-finding and grid optimization approach for single layer grid shells. En: Engineering Structures. No. 52 (2013); p. 230-239.

Como resultado, después de 864123.25 segundos, el PSO encontró los siguientes parámetros que se aproximan a ser óptimos para el UPSO (en su aplicación específica en este problema de prueba):

- $X = 0.5767$
- $C_1 = 0.7729$
- $C_2 = 0.9435$
- $u = 0.5700$

Luego de haber encontrado estos parámetros, el UPSO multi-objetivo (fase final de ACCMOUPSO) corrió con 10 partículas durante 20 iteraciones y con la misma definición de pesos que se describió al inicio.

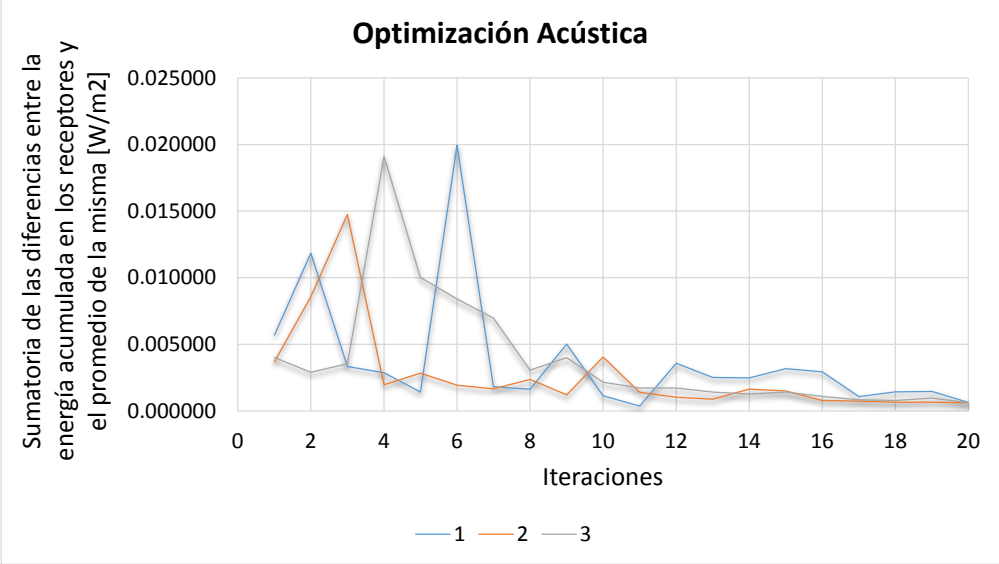
Los resultados obtenidos son los que muestran en el Cuadro 13 y en la Figura 47. A partir de ellos, es posible observar que el ACCMOUPSO efectivamente converge, presentando una desviación estándar de 1.21E-05 W/m^2 (1.94% del promedio) en los resultados finales obtenidos.

Adicionalmente, también es posible ver la efectividad en el proceso de optimización del rendimiento acústico de la estructura (correspondiente a la sumatoria final de las diferencias entre la energía acumulada en los receptores y el promedio de la misma), alcanzando una reducción promedio de 85.61% (en comparación con las estructuras reticulares iniciales generadas aleatoriamente, ver Figura 39), en un tiempo promedio de 177566.61 segundos.

Cuadro 13. Resultados de la convergencia de ACCMOUPSO al optimizar únicamente el rendimiento acústico de la estructura (correspondiente a 3 corridas distintas).

Corrida	Sumatoria inicial de las diferencias entre la energía acumulada en los receptores y el promedio de la misma [W/m^2]	Sumatoria final de las diferencias entre la energía acumulada en los receptores y el promedio de la misma [W/m^2]	Diferencia [%]	Tiempo [seg]
1	0.00568742	0.00063446	-88.84	160901.86
2	0.00368260	0.00061067	-83.42	217551.02
3	0.00401301	0.00061898	-84.58	154246.96
Promedio:	0.00446101	0.00062137	-85.61	177566.61

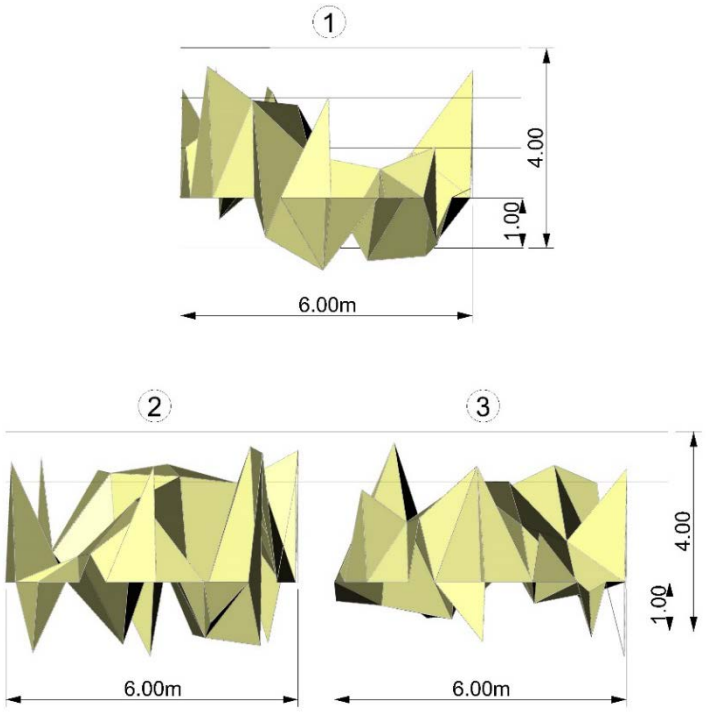
Figura 47. Convergencia de ACCMOUPSO al optimizar únicamente el rendimiento acústico de la estructura (correspondiente a 3 corridas distintas, ver Cuadro 13).



Fuente: Autor

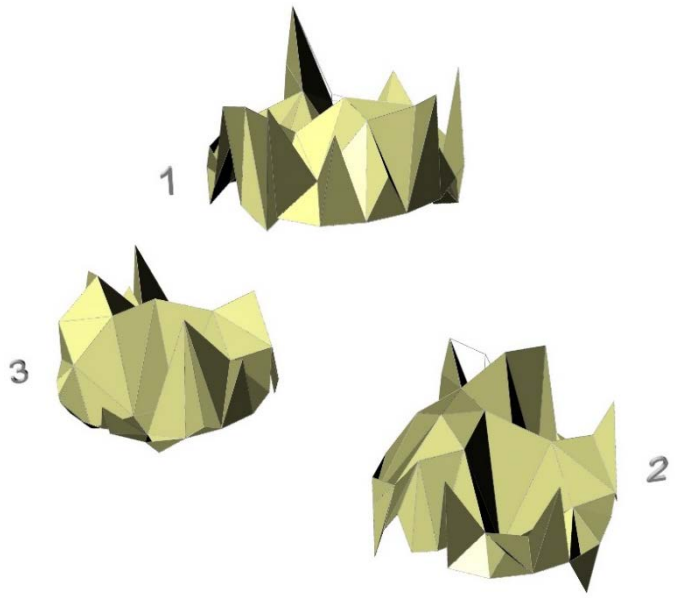
En cuanto al resultado formal de este ejemplo, las configuraciones geométricas de las estructuras reticulares de cubierta obtenidas al ejecutar ACCMOUPSO, son las que se muestran en la Figura 48 y en la Figura 49. En estas imágenes es posible observar que las estructuras tienden a ser significativamente irregulares, y esto se debe a la necesidad de tener distintas orientaciones e inclinaciones en los planos en donde se van a reflejar los rayos que modelan el recorrido del sonido (ver numeral 3.3); entre más diversidad se tenga en dichos planos, los rayos podrán reflejarse hacia una mayor cantidad de área bajo la estructura de cubierta, i.e. se repartirán con mayor uniformidad (lo cual corresponde con el objetivo a alcanzar).

Figura 48. Vista frontal de las estructuras de cubierta correspondientes al resultado final de la convergencia de ACCMOUPSO para el objetivo de rendimiento acústico (correspondientes a 3 corridas distintas, ver el Cuadro 13 y la Figura 47).



Fuente: Autor

Figura 49. Perspectiva de las estructuras de cubierta correspondientes al resultado final de la convergencia de ACCMOUPSO para el objetivo de rendimiento acústico (correspondientes a 3 corridas distintas, ver el Cuadro 13 y la Figura 47).



Fuente: Autor

4.1.4. Optimización simultánea del peso y de la energía de deformación

En este ejemplo, se corrió el ACCMOUPSO diez veces (con el fin de mostrar la convergencia) con una definición de pesos de la siguiente manera: w1 (el peso correspondiente al objetivo de energía de deformación) = 0.5, w2 (el peso correspondiente al objetivo de peso) = 0.5, y w3 (el peso correspondiente al objetivo de rendimiento acústico) = 0.

Para este ejemplo, el PSO dentro del ACCMOUPSO se demoró 1252.95 segundos en encontrar los parámetros que se aproximan a ser óptimos para el UPSO (en su aplicación específica en este problema de prueba), siendo éstos los siguientes:

- $X = 0.4709$
- $C_1 = 1.0830$
- $C_2 = 1.7534$
- $u = 0.6197$

Luego de encontrar estos parámetros, el UPSO multi-objetivo (fase final de ACCMOUPSO) corrió con 1000 partículas durante 50 iteraciones y con la misma definición de pesos que se describió al inicio.

Los resultados obtenidos son los que muestran en el Cuadro 14, en la Figura 50, Figura 51 y Figura 52 (en esta última figura se puede evidenciar de manera más clara la convergencia del algoritmo, correspondiente a la energía de deformación, debido a que en la Figura 51, la escala no permite ver con precisión el comportamiento de éste objetivo a lo largo de las 50 iteraciones).

A partir de estos resultados, es posible ver que el ACCMOUPSO efectivamente converge, presentando una desviación estándar, correspondiente al peso, de 37.14 N (2.53% del promedio de este valor) en los resultados finales obtenidos, mientras que la correspondiente a la energía de deformación fue de 264.33 N*mm (8.54% del promedio de este valor); en este caso, se puede observar que la convergencia del peso empeora, en comparación al ejemplo en donde se optimizó únicamente dicho objetivo (numeral 4.1.1), mientras que la convergencia de la energía de deformación mejora, en comparación al ejemplo en donde la optimización se enfocó exclusivamente en dicho objetivo (ver numeral 4.1.2).

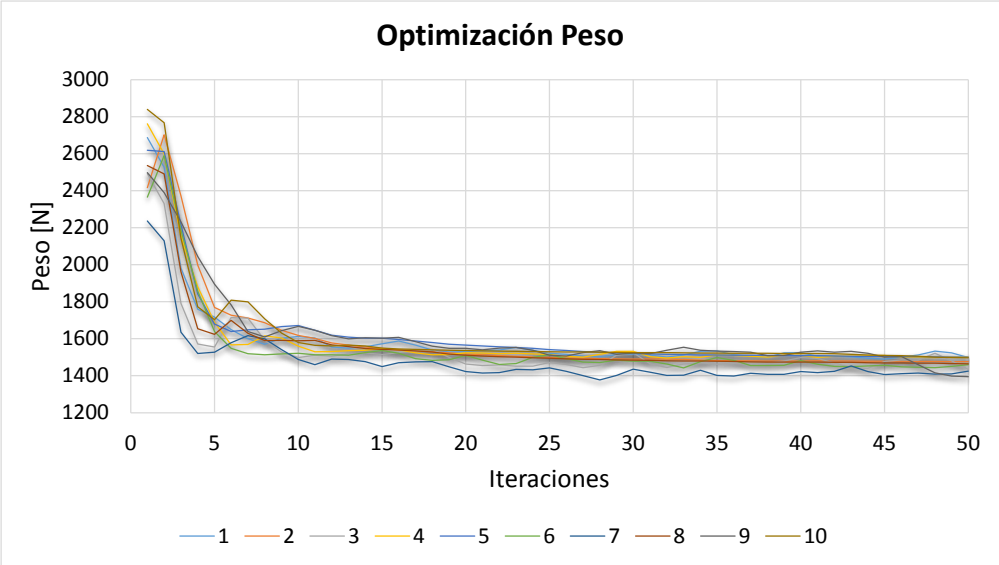
Adicionalmente, también es posible ver la efectividad en el proceso de optimización tanto del peso como de la energía de deformación de la estructura, alcanzando una reducción promedio de 42.22% y 85.08%, respectivamente (en comparación con las estructuras reticulares iniciales generadas aleatoriamente, ver Figura 39), en un tiempo promedio de 499.30 segundos.

En los anteriores resultados se puede analizar la influencia que tiene la optimización simultánea de los dos objetivos en el resultado final de cada uno de ellos: por un lado, el resultado final del peso empeora (en comparación al ejemplo en donde solo se optimiza este objetivo, ver numeral 4.1.1), mientras que el resultado final de la energía de deformación mejora (en comparación al ejemplo en donde solo se optimiza este objetivo, ver numeral 4.1.2).

Cuadro 14. Resultados de la convergencia de ACCMOUPSO al optimizar simultáneamente el peso y la energía de deformación de la estructura (correspondiente a 10 corridas distintas).

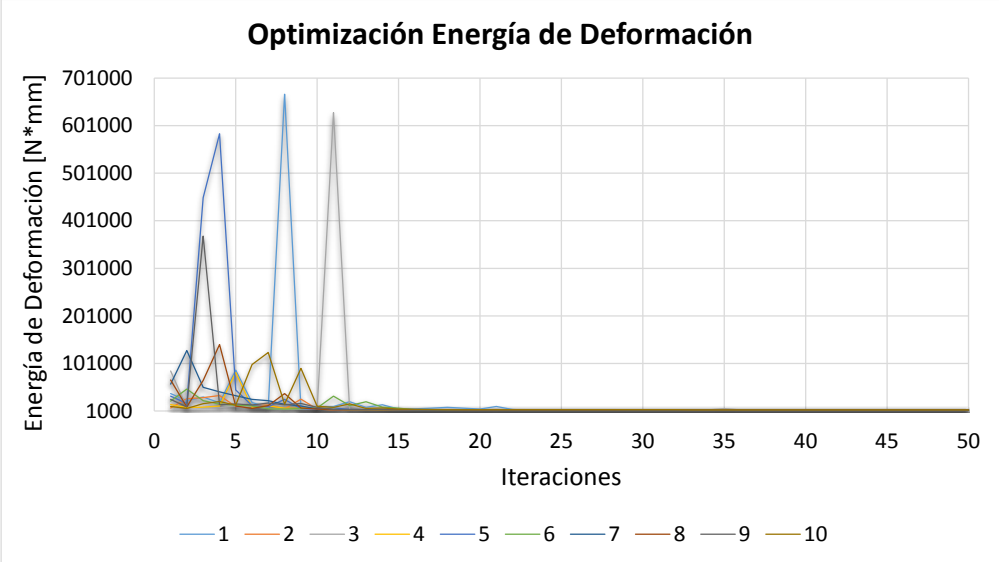
Corrida	Peso Inicial [N]	Peso Final [N]	Diferencia [%]	Energía de Deformación Inicial [N*mm]	Energía de Deformación Final [N*mm]	Diferencia [%]	Tiempo [seg]
1	2686.81	1497.76	-44.26	37706.93	2939.57	-92.20	520.95
2	2416.30	1476.46	-38.90	7238.28	2960.73	-59.10	538.34
3	2492.26	1433.93	-42.46	85068.64	3284.39	-96.14	529.31
4	2762.05	1501.99	-45.62	14771.75	2670.01	-81.92	485.98
5	2619.09	1500.89	-42.69	32457.32	2940.03	-90.94	479.92
6	2365.26	1459.58	-38.29	22030.84	3041.46	-86.19	477.25
7	2236.23	1425.07	-36.27	57686.40	3380.80	-94.14	478.82
8	2536.40	1464.94	-42.24	66335.61	3327.78	-94.98	483.30
9	2497.57	1394.66	-44.16	25507.17	2908.96	-88.60	479.89
10	2839.30	1496.86	-47.28	10514.72	3510.70	-66.61	519.28
Promedio:	2545.13	1465.21	-42.22	35931.77	3096.44	-85.08	499.30

Figura 50. Convergencia de ACCMOUPSO (específicamente del objetivo de peso) al optimizar simultáneamente el peso y la energía de deformación de la estructura (correspondiente a 10 corridas distintas, ver Cuadro 14)



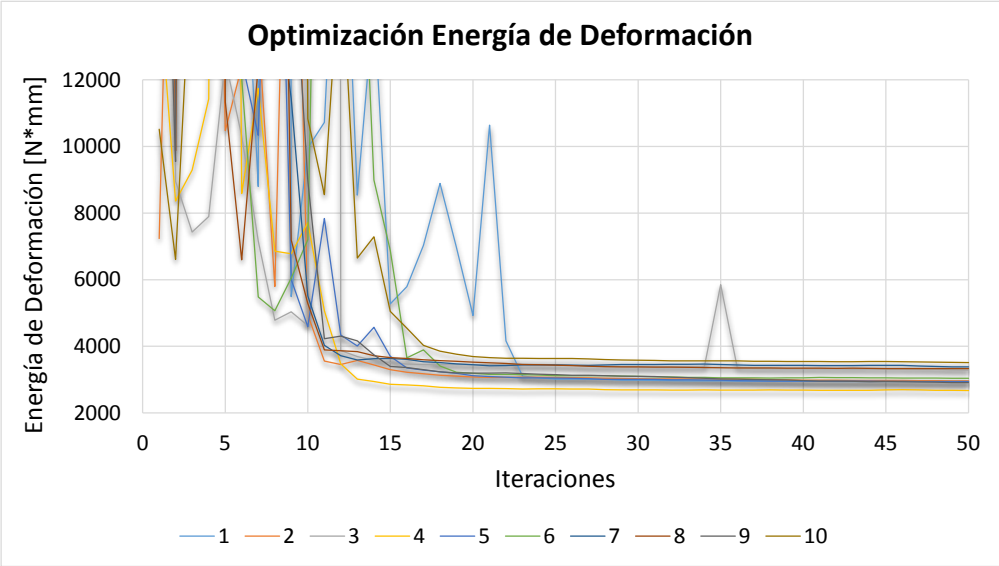
Fuente: Autor

Figura 51. Convergencia de ACCMOUPSO (específicamente de la energía de deformación) al optimizar simultáneamente el peso y la energía de deformación de la estructura (correspondiente a 10 corridas distintas, ver Cuadro 14)



Fuente: Autor

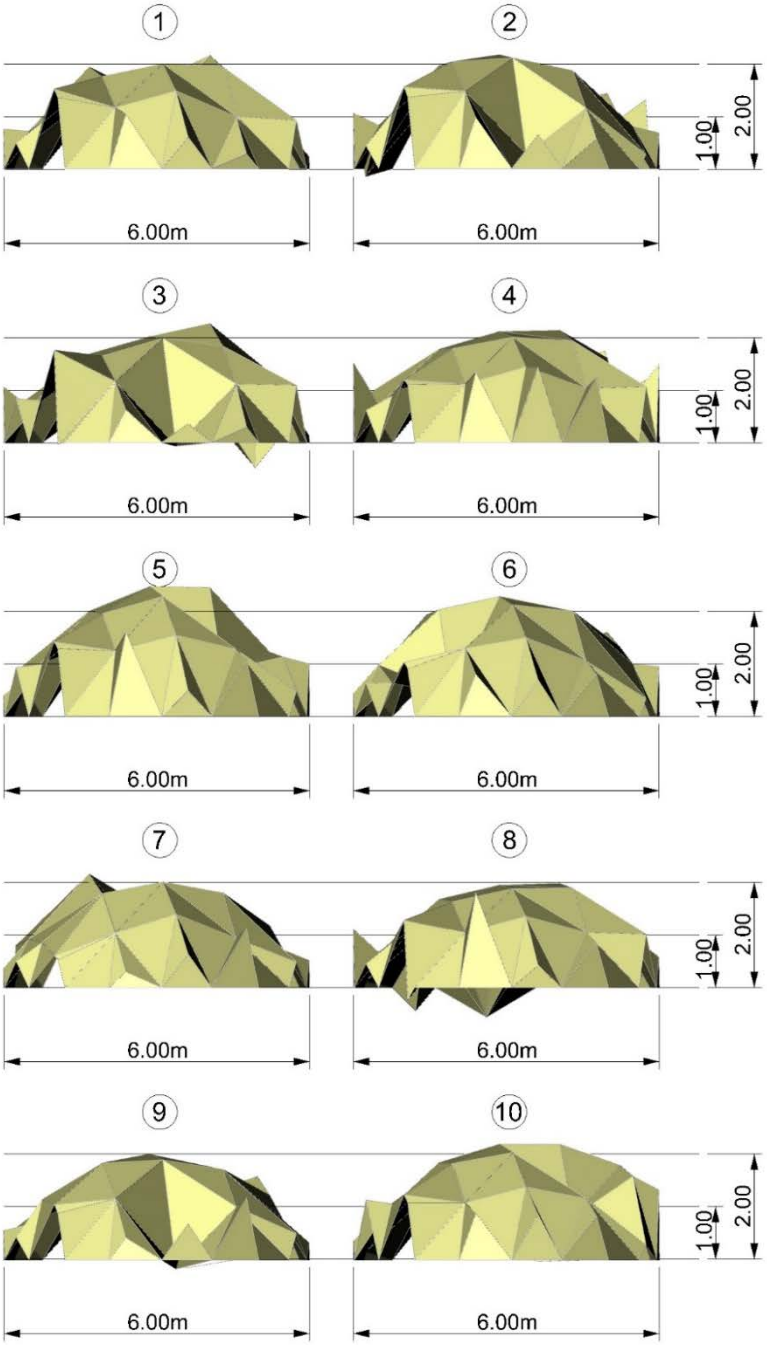
Figura 52. Acercamiento de la Figura 51



Fuente: Autor

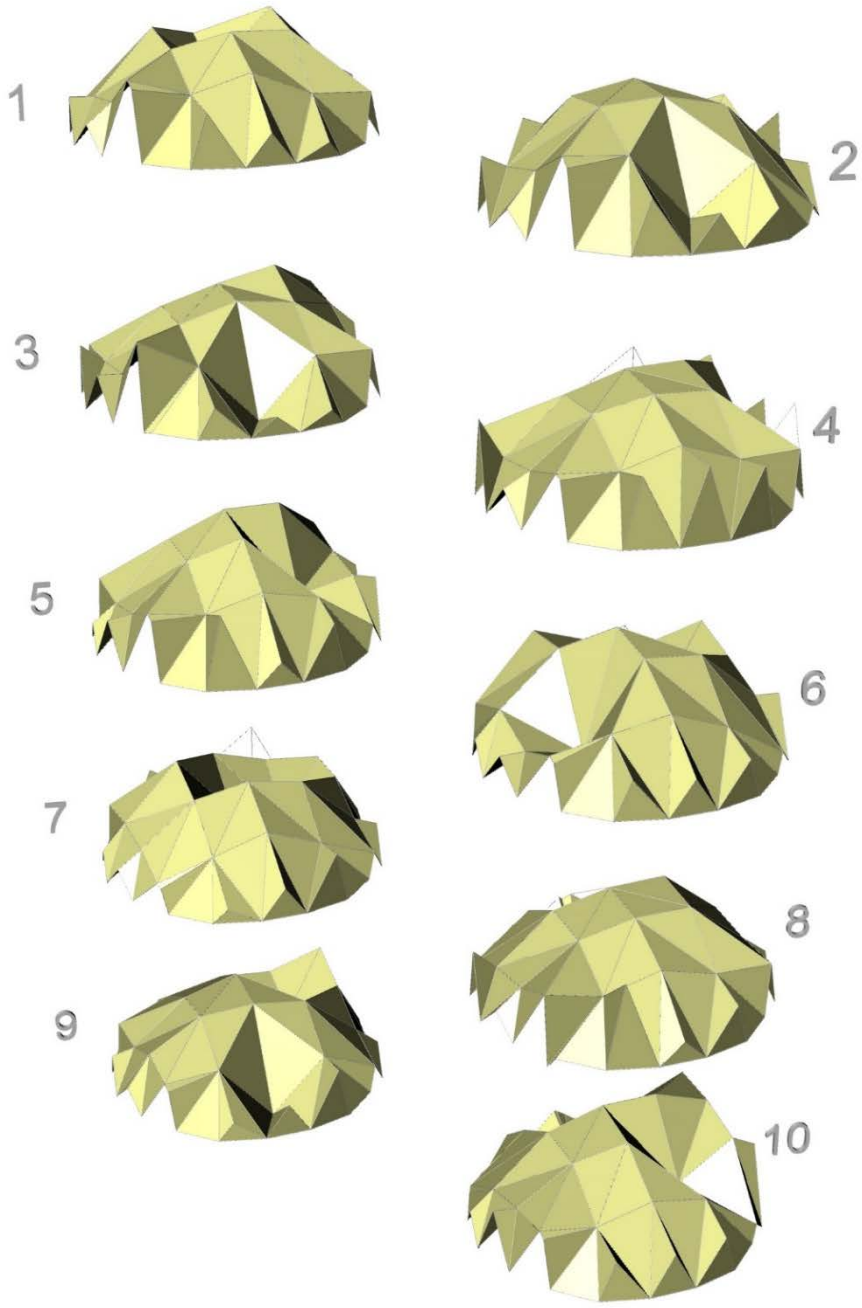
En lo referente al resultado formal de este ejemplo en donde se optimizan simultáneamente el peso y la energía de deformación, las configuraciones geométricas de las estructuras reticulares de cubierta obtenidas al ejecutar ACCMOUPSO, son las que se muestran en la Figura 53 y en la Figura 54. En estas imágenes es posible observar que las estructuras tienden hacia la tipología estructural de domo (resultado similar al ejemplo en donde únicamente se optimizó la energía de deformación, ver numeral 4.1.2), pero debido a la tendencia que el objetivo de peso presenta hacia una configuración geométrica que tiende a ser plana (ver numeral 4.1.1), los domos resultantes surgen con una altura intermedia entre las estructuras planas del ejemplo 4.1.1 (considerando únicamente el objetivo de peso) y los domos profundos del ejemplo 4.1.2 (considerando únicamente el objetivo de la energía de deformación).

Figura 53. Vista frontal de las estructuras de cubierta correspondientes al resultado final de la convergencia de ACCMOUPSO para los objetivos de peso y de energía de deformación (correspondientes a 10 corridas distintas, ver Cuadro 14, Figura 50, Figura 51 y Figura 52).



Fuente: Autor

Figura 54. Perspectiva de las estructuras de cubierta correspondientes al resultado final de la convergencia de ACCMOUPSO para los objetivos de peso y de energía de deformación (correspondientes a 10 corridas distintas, ver Cuadro 14, Figura 50, Figura 51 y Figura 52).



Fuente: Autor

4.1.5. Optimización simultánea del peso, de la energía de deformación, y del rendimiento acústico

Para el desarrollo de este ejemplo, se corrió el ACCMOUPSO tres veces (debido al alto esfuerzo computacional de cada ejecución, y con el fin de mostrar la convergencia) con una definición de pesos de la siguiente manera: w1 (el peso correspondiente al objetivo de energía de deformación) = $1/3$, w2 (el peso correspondiente al objetivo de peso) = $1/3$, y w3 (el peso correspondiente al objetivo de rendimiento acústico) = $1/3$.

De manera similar al ejemplo de convergencia del rendimiento acústico (sección 4.1.3), en este problema de prueba se cambió el número de partículas y de iteraciones para la primera fase de ACCMOUPSO (debido al alto costo computacional), i.e. para la búsqueda de los mejores parámetros para el UPSO por parte del PSO; el PSO corrió con 4 partículas durante 20 iteraciones, mientras que el UPSO corrió, dentro del PSO, con 3 partículas durante 6 iteraciones.

Como resultado, después de 981384.77 segundos, el PSO encontró los siguientes parámetros que se aproximan a ser óptimos para el UPSO (en su aplicación específica en este problema de prueba):

- $X = 0.3994$
- $C_1 = 1.2685$
- $C_2 = 1.4922$
- $u = 0.6801$

Luego de haber encontrado los anteriores parámetros, el UPSO multi-objetivo (fase final de ACCMOUPSO) corrió con 10 partículas durante 20 iteraciones y con la misma definición de pesos que se describió al inicio.

Los resultados obtenidos son los que muestran en el Cuadro 15, en la Figura 55, Figura 56, Figura 57 y Figura 58 (en la Figura 57 se puede evidenciar de manera más clara la convergencia del algoritmo, correspondiente a la energía de deformación, debido a que en la Figura 56, la escala no permite ver con precisión el comportamiento de éste objetivo a lo largo de las 20 iteraciones).

De estos resultados, es posible ver que el ACCMOUPSO efectivamente converge, presentando una desviación estándar, correspondiente al peso, de 113.39 N (6.64% del promedio de este valor) en los resultados finales obtenidos, correspondiente a la energía de deformación, de 403.24 N*mm (9.89% del promedio de este valor), mientras que la correspondiente a la sumatoria de las diferencias entre la energía acumulada en los receptores y el promedio de la misma (rendimiento acústico), fue de 0.00016 W/m^2 (11.60% del promedio de este valor).

Es evidente que los anteriores valores aumentaron, en comparación con los demás ejemplos expuestos hasta el momento, y esta situación se debe a que la convergencia del algoritmo se da, en general, para la función objetivo total (la que combina a los tres objetivos, ver ecuación (14)), la cual puede tender hacia ciertos valores finales similares teniendo diferencias entre los valores que toman las funciones objetivo que la componen. Por ejemplo, es posible que la función objetivo total llegue a valores muy similares con un valor bajo para la energía de deformación y unos valores moderados para las otras dos funciones (peso y rendimiento acústico), o con un valor bajo para el rendimiento acústico y unos valores medios para los otros dos objetivos (peso y energía de deformación).

La situación descrita anteriormente, es justamente lo que sucede en las tres corridas que se ejecutaron para mostrar la convergencia de ACCMOUPSO al optimizar simultáneamente los tres objetivos considerados (ver Cuadro 15).

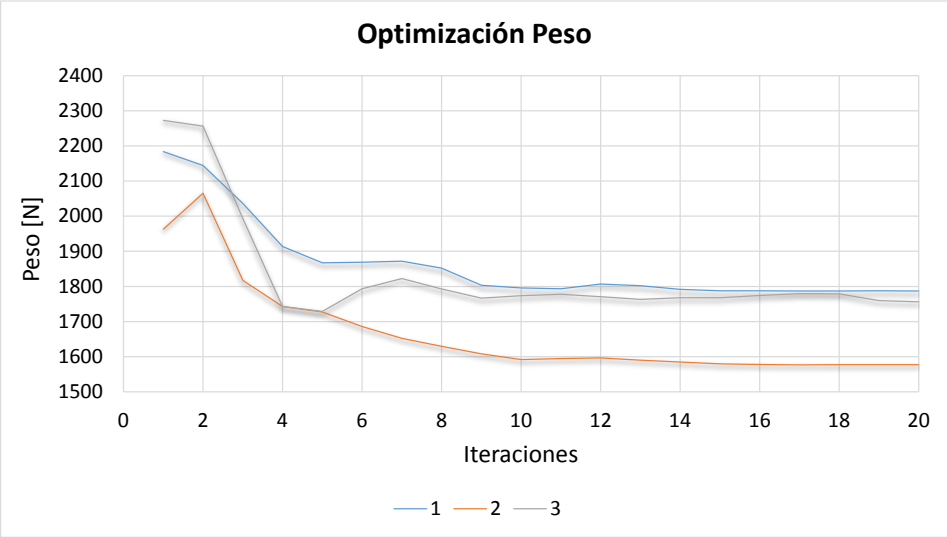
Por otro lado, con dichos resultados también se evidencia la efectividad en el proceso de optimización, tanto en el peso como en la energía de deformación, como en el rendimiento acústico de la estructura reticular de cubierta, alcanzando una reducción promedio de 20.19%, 63.43% y 94.07%, respectivamente (en comparación con las estructuras reticulares iniciales generadas aleatoriamente, ver Figura 39), en un tiempo promedio de 203291.80 segundos.

Cuadro 15. Resultados de la convergencia de ACCMOUPSO al optimizar simultáneamente el peso, la energía de deformación, y el rendimiento acústico de la estructura (correspondiente a 3 corridas distintas).

Corrida	Peso Inicial [N]	Peso Final [N]	Diferencia [%]	Energía de Deformación Inicial [N*mm]	Energía de Deformación Final [N*mm]	Diferencia [%]
1	2183.62	1787.01	-18.16	13832.44	4349.44	-68.56
2	1963.16	1577.07	-19.67	11540.97	4263.21	-63.06
3	2273.08	1756.32	-22.73	8737.69	3611.91	-58.66
Promedio:	2139.96	1706.80	-20.19	11370.37	4074.85	-63.43

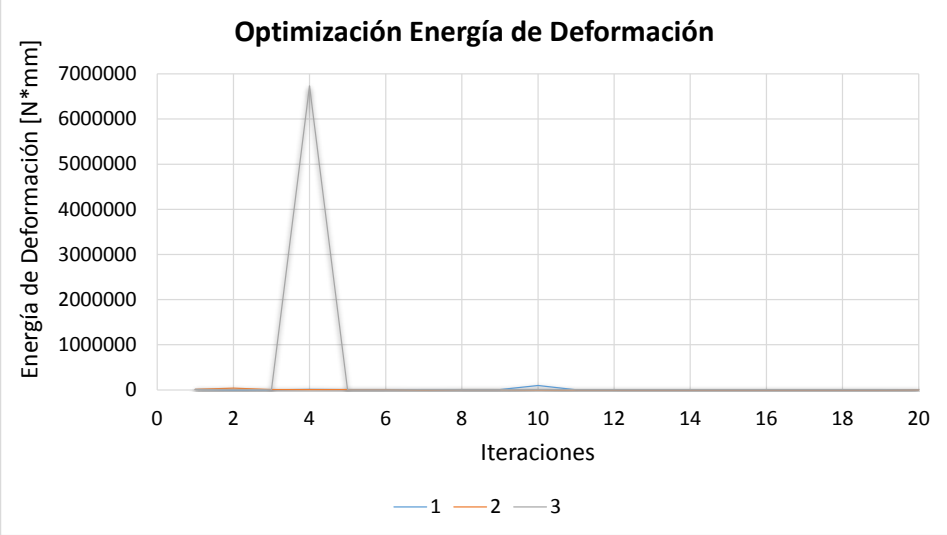
Corrida	Sumatoria inicial de las diferencias entre la energía acumulada en los receptores y el promedio de la misma [W/m2]	Sumatoria final de las diferencias entre la energía acumulada en los receptores y el promedio de la misma [W/m2]	Diferencia [%]	Tiempo [seg]
1	0.02836846	0.00121185	-95.73	148924.87
2	0.02343883	0.00151522	-93.54	202003.23
3	0.02076377	0.00146183	-92.96	258947.29
Promedio:	0.02419035	0.00139630	-94.07	203291.80

Figura 55. Convergencia de ACCMOUPSO (específicamente del objetivo de peso) al optimizar simultáneamente el peso, la energía de deformación, y el rendimiento acústico de la estructura (correspondiente a 3 corridas distintas, ver Cuadro 15).



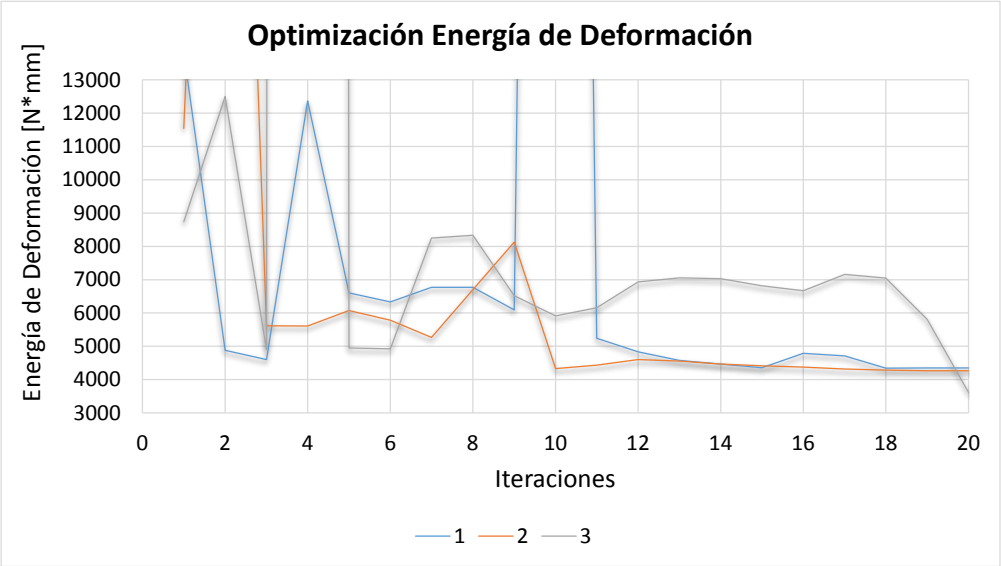
Fuente: Autor

Figura 56. Convergencia de ACCMOUPSO (específicamente del objetivo de energía de deformación) al optimizar simultáneamente el peso, la energía de deformación, y el rendimiento acústico de la estructura (correspondiente a 3 corridas distintas, ver Cuadro 15).



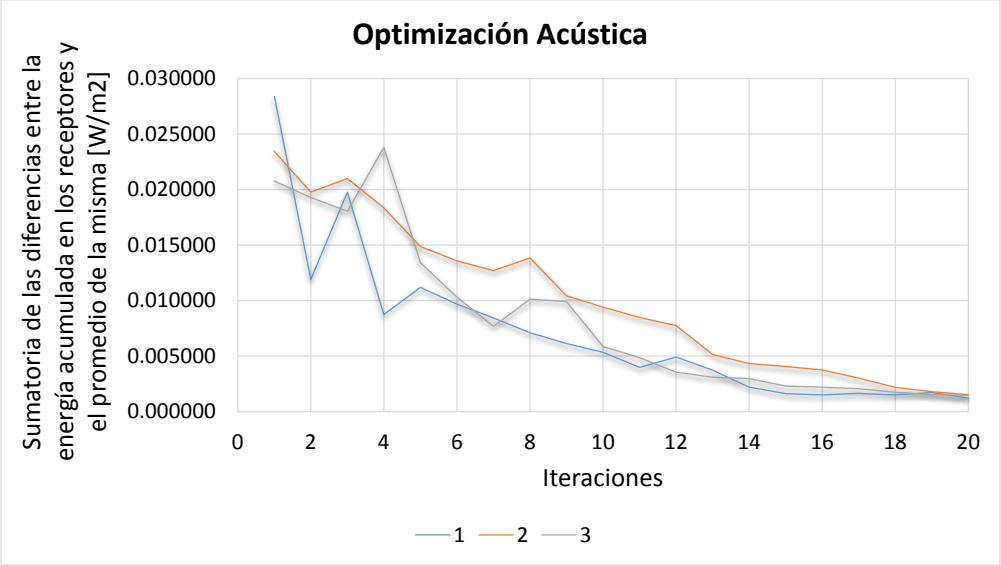
Fuente: Autor

Figura 57. Acercamiento de la Figura 56.



Fuente: Autor

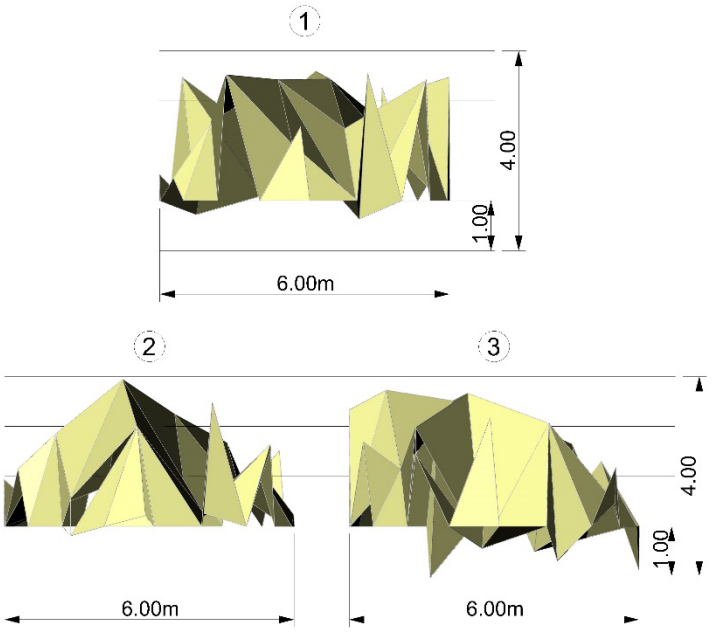
Figura 58. Convergencia de ACCMOUPSO (específicamente del objetivo de rendimiento acústico) al optimizar simultáneamente el peso, la energía de deformación, y el rendimiento acústico de la estructura (correspondiente a 3 corridas distintas, ver Cuadro 15).



Fuente: Autor

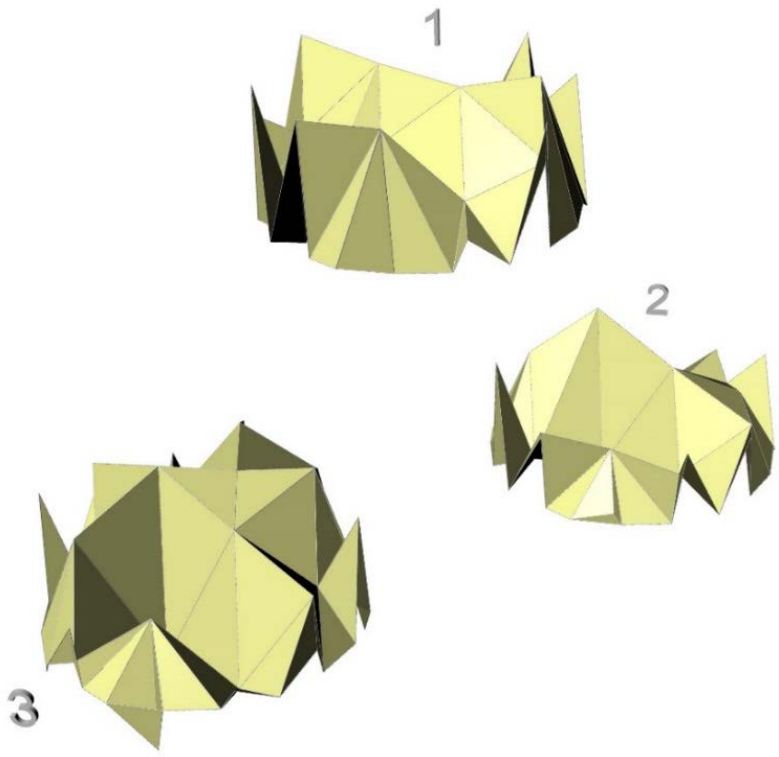
En cuanto al resultado formal de este ejemplo de convergencia, las configuraciones geométricas de las estructuras reticulares de cubierta obtenidas al ejecutar ACCMOUPSO, son las que se muestran en la Figura 59 y en la Figura 60. En estas imágenes es posible observar que la tipología a la cual tienden las estructuras, responde a la combinación de los resultados formales obtenidos al optimizar independientemente cada uno de los objetivos considerados (ver numerales 4.1.1, 4.1.2 y 4.1.3). Esto quiere decir que la geometría resultante tiende a ser una aproximación hacia una tipología de cúpula (la cual responde a la combinación de los objetivos de peso y energía de deformación), distorsionada con planos irregulares por su elevación y orientación (lo cual corresponde al objetivo de rendimiento acústico).

Figura 59. Vista frontal de las estructuras de cubierta correspondientes al resultado final de la convergencia de ACCMOUPSO para los objetivos de peso, energía de deformación y rendimiento acústico (correspondientes a 3 corridas distintas, ver Cuadro 15, Figura 55, Figura 56, Figura *57* y Figura 58).



Fuente: Autor

Figura 60. Perspectiva de las estructuras de cubierta correspondientes al resultado final de la convergencia de ACCMOUPSO para los objetivos de peso, energía de deformación y rendimiento acústico (correspondientes a 3 corridas distintas, ver Cuadro 15, Figura 55, Figura 56, Figura *57* y Figura 58).

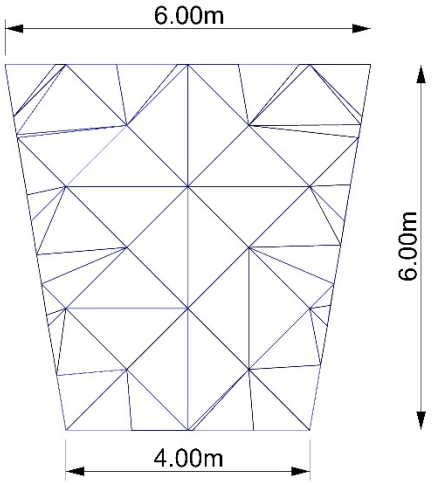


Fuente: Autor

4.2. RESULTADOS DEL PACDMOER EN EL DISEÑO ESTRUCTURAL Y ARQUITECTÓNICO DE UNA ESTRUCTURA RETICULAR DE CUBIERTA

Para el desarrollo de este ejemplo, se utilizó una configuración geométrica definida por una planta en forma de trapecio, el cual tiene una base mayor de 6.00m y una menor de 4.00m, mientras que su altura, es de 6.00m (ver Figura 61).

Figura 61. Planta trapezoidal de la estructura reticular de cubierta, utilizada mostrar el funcionamiento de PACDMOER en su totalidad.



Fuente: Autor

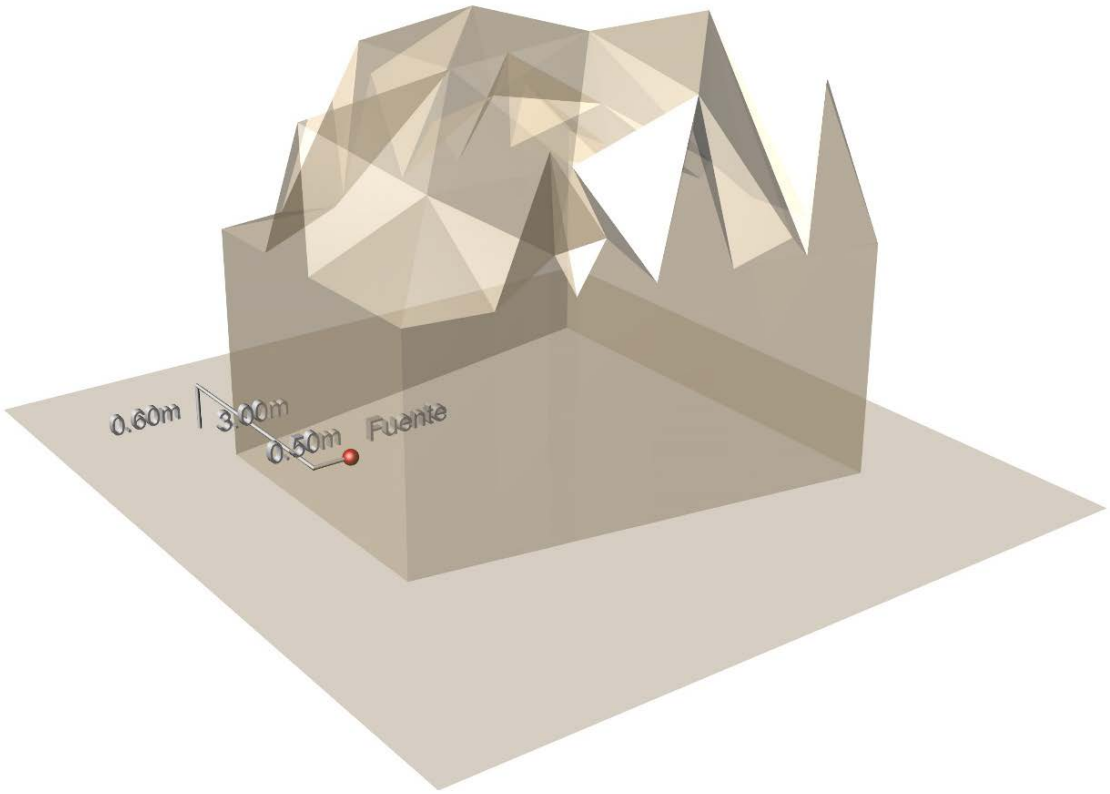
La estructura se definió con 48 nodos y 106 barras (ver Figura 61), a las cuales se les asignó una sección transversal circular de 1⁄2" de diámetro (correspondiente a un área de $126.7mm^2$), y un acero con módulo de elasticidad de 200000 MPa y densidad de 7800 kg/m^3. En cuanto a las cargas aplicadas, todos los nodos fueron solicitados con una carga vertical de 1000 N en dirección de la gravedad.

Adicionalmente, para la evaluación del rendimiento acústico se definieron las coordenadas de la fuente de sonido, en metros, como $[3.00, 0.50, 0.60]$ (ver Figura 62), mientras que el nivel de la potencia de sonido (Sound Power Level – SPL) de ésta fue de 90 dB. Por otro lado, el coeficiente de absorción del aire se tomó como 2.77 dB⁄km (correspondiente a una temperatura de 20°C, una humedad relativa del 80%, y una frecuencia de preferencia de 500 Hz)[191], y el coeficiente de

[191] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION - ISO. Attenuation of sound during propagation outdoors, Part 1: Calculation of the absorption of sound by the atmosphere. Geneve: International Organization for Standardization, 1993. (ISO 9613-1).

absorción del material de los distintos planos (muros, piso y cubierta) se definió como 0.17 (correspondiente a paneles de plywood para una frecuencia de preferencia de 500 Hz)[192].

Figura 62. Ubicación de la fuente de sonido dentro del espacio cubierto por la estructura reticular, utilizada mostrar el funcionamiento de PACDMOER en su totalidad.



Fuente: Autor

Para finalizar con la definición geométrica de este ejemplo, las restricciones aplicadas a la coordenada z de cada nodo (las variables durante el proceso de morfogénesis), se plantearon para delimitar un rango espacial, i.e. una altura

[192] MONKS, M.C. Audioptimization: Goal-Based Acoustic Design. Massachusetts, 1999. Tesis doctoral. Massachusetts Institute of Technology.

máxima y una mínima. En este caso, la restricción se definió en función de las dimensiones en planta (las dimensiones del trapecio), para que el resultado formal final tuviera la libertad de formar hasta tipologías de bóvedas o cúpulas (dejando un margen de movilidad considerable).

Por último, el PSO corrió dentro del ACCMOUPSO con 4 partículas durante 20 iteraciones y con la siguiente definición de parámetros, tomada de Toscano Pulido et al.[193]:

- $X = 0.5$
- $C_1 = 1.4$
- $C_2 = 1.4$

En cambio, el UPSO multiobjetivo corrió dentro del PSO descrito anteriormente, con 3 partículas durante 6 iteraciones. Esta asignación de un número moderado de partículas e iteraciones corresponde al alto tiempo computacional requerido en la ejecución de ACCMOUPSO (sobre todo cuando se incluye el objetivo de rendimiento acústico, como es el caso de este ejemplo).

Para iniciar con la aplicación del procedimiento propuesto para el diseño multi-objetivo de estructuras reticulares, se corrió la primera parte de ACCMOUPSO (la correspondiente a optimizar los parámetros que definen el comportamiento y el rendimiento del UPSO) con el número de partículas y de iteraciones definido en los párrafos anteriores, y con una combinación de pesos en donde los tres valores (w1, w2 y w3) eran iguales a $1/3$.

[193] TOSCANO-PULIDO, G., SANTANA-QUINTERO, L.V. y COELLO COELLO, C.A. EMOPSO: A Multi-Objective Particle Swarm Optimizer with Emphasis on Efficiency. En: EMO 2007. Proceedings of EMO 2007. Heidelberg: Springer, 2007. p. 272-285.

Como resultado, después de 698541.37 segundos, el PSO encontró los siguientes parámetros que se aproximan a ser óptimos para el UPSO (en su aplicación específica en este problema de prueba):

- $X = 0.1991$
- $C_1 = 1.3371$
- $C_2 = 1.6374$
- $u = 0.9068$

Luego de haber encontrado estos parámetros, el UPSO multi-objetivo (siguiente paso en ACCMOUPSO) corrió con 10 partículas durante 20 iteraciones y con las siguientes combinaciones de pesos:

1. $w1 = 0.80;\ \ w2 = 0.10;\ \ \ w3 = 0.10$
2. $w1 = 0.10;\ \ w2 = 0.80;\ \ \ w3 = 0.10$
3. $w1 = 0.10;\ \ w2 = 0.10;\ \ \ w3 = 0.80$
4. $w1 = {}^{1}/_{3};\ \ w2 = {}^{1}/_{3};\ \ \ w3 = {}^{1}/_{3}$

Los resultados de rendimiento obtenidos son los que muestran en el Cuadro 16. A partir de ellos, es posible observar que ACCMOUPSO es efectivo en el proceso de optimización, ya que la reducción en el peso osciló entre el 6.66% y el 21.59%, la correspondiente a la energía de deformación estuvo entre el 15.38% y el 54.96%, mientras que la reducción en la sumatoria de las diferencias entre la energía acumulada en los receptores y el promedio de la misma (correspondiente al objetivo acústico) osciló entre el 84.39% y el 97.88% (vale la pena aclarar que estas comparaciones se hicieron con respecto a las estructuras de cubierta iniciales, generadas aleatoriamente). Por otro lado, el tiempo promedio que se tardaron las corridas fue de 203489.79 segundos.

Cuadro 16. Resultados de ACCMOUPSO para cuatro combinaciones de pesos distintas.

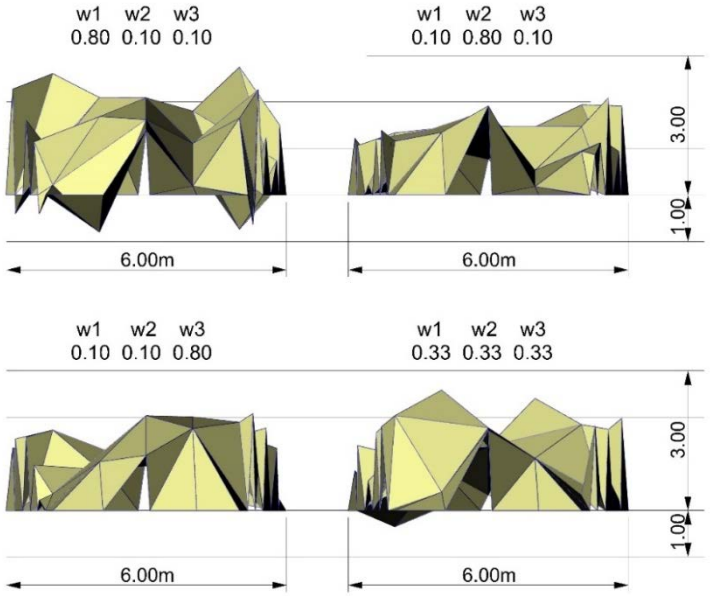
Combinación de Pesos			Peso Inicial [N]	Peso Final [N]	Diferencia [%]	Energía de Deformación Inicial [N*mm]	Energía de Deformación Final [N*mm]	Diferencia [%]
w1	w2	w3						
0.80	0.10	0.10	1892.76	1766.67	-6.66	5174.80	4378.69	-15.38
0.10	0.80	0.10	1886.59	1479.19	-21.59	7920.18	6620.61	-16.41
0.10	0.10	0.80	1878.61	1563.02	-16.80	10934.90	4925.35	-54.96
0.33	0.33	0.33	1816.73	1539.00	-15.29	8954.01	4638.99	-48.19
Promedio:			1868.67	1586.97	-15.09	8245.97	5140.91	-33.74

Combinación de Pesos			Sumatoria inicial de las diferencias entre la energía acumulada en los receptores y el promedio de la misma [W/m2]	Sumatoria final de las diferencias entre la energía acumulada en los receptores y el promedio de la misma [W/m2]	Diferencia [%]	Tiempo [seg]
w1	**w2**	**w3**				
0.80	0.10	0.10	0.02546091	0.00397571	-84.39	175431.87
0.10	0.80	0.10	0.04401324	0.00388662	-91.17	177994.75
0.10	0.10	0.80	0.03991655	0.00084736	-97.88	281281.96
0.33	0.33	0.33	0.01461306	0.00131863	-90.98	179250.59
Promedio:			0.03100094	0.00250708	-91.10	203489.79

Aparte de los resultados de rendimiento, en la Figura 63 y en la Figura 64, se puede observar el resultado formal de las estructuras reticulares de cubierta, correspondiente a cada una de las combinaciones de pesos considerada.

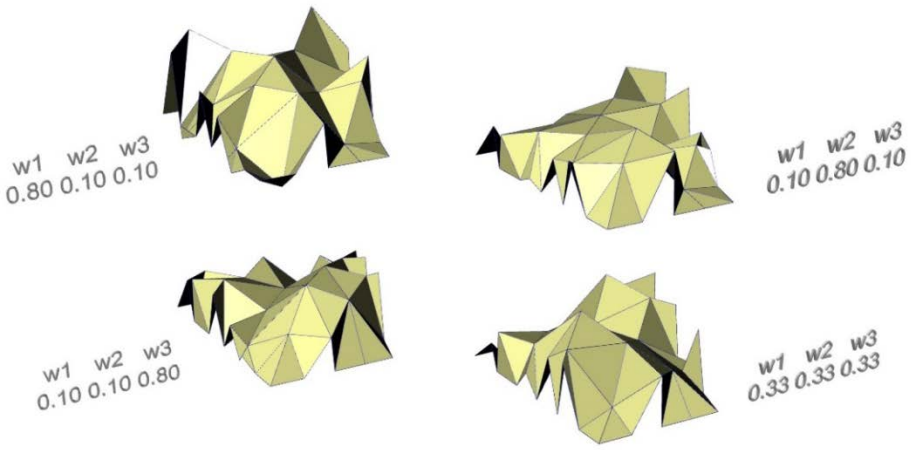
Partiendo de la combinación de los resultados expuestos hasta el momento (rendimiento y estética), el paso a seguir es seleccionar la solución preferida, teniendo en cuenta criterios de rendimiento y criterios estéticos. Por consiguiente, en este caso se seleccionó la solución que corresponde a la combinación de pesos número 4, en donde los tres pesos, w1, w2, y w3, son iguales a $1/3$.

Figura 63. Vista frontal de las estructuras reticulares de cubierta que se obtuvieron con ACCMOUPSO para las cuatro combinaciones de pesos consideradas, ver Cuadro 16.



Fuente: Autor

Figura 64. Perspectiva de las estructuras reticulares de cubierta que se obtuvieron con ACCMOUPSO para las cuatro combinaciones de pesos consideradas, ver Cuadro 16.



Fuente: Autor

Con la combinación de pesos seleccionada, se procedió a ejecutar por última vez la última fase de ACCMOUPSO, con la misma definición de parámetros para el PSO que se utilizó anteriormente, y con los mismos parámetros para el UPSO encontrados por el PSO; pero ésta vez, se incrementó el número de iteraciones a 40 (debido al alto costo computacional, no se incrementó en mayor medida el número de evaluaciones de las funciones objetivo). Los resultados de rendimiento obtenidos son los que se exponen en el Cuadro 17, en donde se puede observar que la solución mejoró con respecto a la obtenida con la misma combinación de pesos en el paso anterior (ver Cuadro 16), con respecto a los tres objetivos.

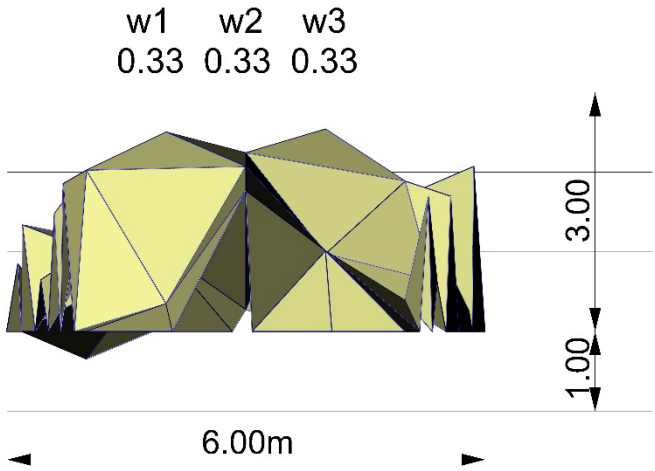
Cuadro 17. Resultados de ACCMOUPSO para la última fase de PACDMOER.

Combinación de Pesos			**Peso Inicial [N]**	**Peso Final [N]**	**Diferencia [%]**	**Energía de Deformación Inicial [N*mm]**	**Energía de Deformación Final [N*mm]**	**Diferencia [%]**
w1	w2	w3						
0.33	0.33	0.33	1830.68	1487.85	-18.73	9337.14	4488.62	-51.93

Combinación de Pesos			**Sumatoria inicial de las diferencias entre la energía acumulada en los receptores y el promedio de la misma [W/m2]**	**Sumatoria final de las diferencias entre la energía acumulada en los receptores y el promedio de la misma [W/m2]**	**Diferencia [%]**	**Tiempo [seg]**
w1	**w2**	**w3**				
0.33	0.33	0.33	0.03641998	0.00119056	-96.73	362776.86

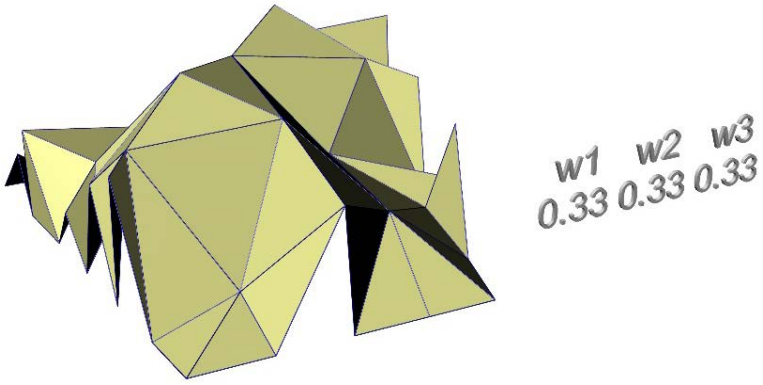
Adicionalmente, en la Figura 65 y en la Figura 66, puede verse el resultado formal de la cubierta finalmente obtenida por medio de PACDMOER.

Figura 65. Vista frontal de la cubierta final obtenida por medio de PACDMOER.



Fuente: Autor

Figura 66. Perspectiva de la cubierta final obtenida por medio de PACDMOER.



Fuente: Autor

Por último, es interesante resaltar la manera en como la forma resulta de la combinación de las tipologías estructurales que se aproximan a ser óptimas al considerar un solo objetivo. Es decir, la forma que se obtuvo, es la combinación de la aproximación a un domo (correspondiente al objetivo de la energía de

deformación) achatado (debido al objetivo del peso) y distorsionado con planos irregulares (correspondientes al objetivo del rendimiento acústico).

Para concluir este capítulo, vale la pena comentar que, aparte de los ejemplos aquí mostrados, se elaboró un artículo titulado “Desarrollo de un Proceso de Morfogénesis Basado en Computación Heurística Multi-Objetiva para una Estructura Reticular de Cubierta”, en donde se exponen ejemplos adicionales enfocados en la búsqueda de forma de la estructura reticular, teniendo en cuenta los objetivos de la energía de deformación y el peso. Este artículo ha sido aceptado para publicación en la Revista M, de arquitectura.

5. CONCLUSIONES

En la sección correspondiente al marco teórico y a los antecedentes correspondientes al presente tema de investigación (sección 1), se presentaron las técnicas heurísticas de computación multi-objetiva como una herramienta de diseño, especialmente útil en la fase conceptual. Allí se revisaron varios ejemplos desarrollados dentro del campo del diseño estructural y arquitectónico (incluyendo un ejemplo de planeación urbana), discutiendo la metodología de diseño basada en la computación heurística multi-objetiva, y la posibilidad que tiene de llegar a ser un proceso eficiente y creativo, el cual ayudaría al diseñador en la generación de un amplio grupo de soluciones diferenciadas y con gran calidad. En este sentido, el diseñador puede expandir su creatividad mientras garantiza soluciones de alto rendimiento.

Como sugerencia para futuras investigaciones en este campo, se plantea la posibilidad de incorporar un mayor número de objetivos en el proceso de diseño basado en computación heurística multi-objetiva. Por ejemplo, se podrían combinar procesos de optimización estructural, acústica, de iluminación, bioclimática, y de sostenibilidad, dentro de un solo procedimiento integral de optimización y morfogénesis. Esto llevaría a la obtención de soluciones holísticas de diseño, con el mejor rendimiento posible (de acuerdo a las funciones que el(la) diseñador(a) quiera incluir) y, por qué no, con una reducción en el costo de construcción. Adicionalmente, esto permitiría una exploración mucho más amplia de la forma resultante del objeto arquitectónico, satisfaciendo el criterio del(de la) diseñador(a) mientras se mantiene, o se mejora, la idoneidad de la solución.

Otro paso potencial es la implementación de procesos de optimización para encontrar los mejores valores de los parámetros que definen el rendimiento de las distintas técnicas heurísticas (lo cual se lleva a cabo en la presente investigación). Este proceso de optimización correría en cada iteración del algoritmo principal, asegurando su eficiencia y evitando cuestionamientos en este sentido.

El acercamiento propuesto de optimización multi-objetiva (ACCMOUPSO), en el cual se integran el PSO convencional con el UPSO, con el fin de generar un algoritmo que se auto-configure (con respecto a la definición de los parámetros que definen el comportamiento y el rendimiento del UPSO, i.e. los parámetros de aceleración c_1 y c_2, el factor de constricción, X, y el factor de unificación u), demostró ser eficiente en la búsqueda de soluciones para problemas de optimización multi-objetiva. Lo anterior se pudo verificar al comparar las soluciones encontradas por ACCMOUPSO con el frente de Pareto de las funciones de prueba ZDT1, ZDT2 y ZDT3 (las cuales tienen 30 variables y dos funciones objetivo). Los valores de los resultados obtenidos se acercaron con gran precisión a los correspondientes al frente de Pareto, y lograron una distribución aproximadamente uniforme a lo largo de éste.

Vale la pena resaltar el avance que se logró con ACCMOUPSO con respecto a la definición, por parte del usuario, de los valores para los parámetros inciertos del UPSO (c_1, c_2, X y u), ya que en este caso, el diseñador no debe definirlos, sino que el mismo algoritmo es capaz de optimizar dichos valores para el problema específico de optimización que se esté intentando resolver.

En este sentido, es posible dejar a un lado la incertidumbre, junto con las posibles críticas que de ella se generen, de la implementación de una técnica heurística de optimización como la que en el presente trabajo de investigación se llevó a cabo.

Sin lugar a duda, el acercamiento propuesto toma más tiempo en comparación con los acercamientos convencionales (que deben ser configurados manualmente por el usuario), pero debido a que la naturaleza de la implementación que se propone está enfocada en el diseño conceptual estructural y arquitectónico, éste tiempo adicional no se toma como un problema, ya que no se necesita una respuesta inmediata por parte del algoritmo, sino una serie de alternativas de diseño con alto rendimiento de donde se podrá partir para el desarrollo de una estructura o de la totalidad de un objeto arquitectónico.

En el Procedimiento Auto-Configurado para el Diseño Multi-Objetivo de Estructuras Reticulares (PACDMOER) que se propone, fue posible integrar un proceso que se creó para generar automáticamente la geometría de una estructura reticular de cubierta (teniendo como datos de entrada únicamente los nodos de la base que definen el perímetro de la estructura, y la cantidad de divisiones deseada para la generación de la retícula), con un código de programación que se realizó para calcular la energía de deformación y el peso de la estructura, y con otro semejante para llevar a cabo el análisis acústico de la misma; todos dentro del funcionamiento del acercamiento propuesto de optimización multi-objetiva (ACCMOUPSO), el cual permite una interacción entre el usuario y el procedimiento de optimización por medio de la manipulación de la combinación de pesos, correspondiente a la importancia que se le desea dar a cada objetivo considerado.

Ésta integración, de la cual se deriva una interdependencia y una interconexión entre procesos formales, de análisis técnicos de rendimiento, y de optimización, se plantea como una rama potencial de desarrollo en el campo del diseño arquitectónico (enfocada hacia el nuevo paradigma planteado en la introducción del presente documento), en el cual se incluyen todos los componentes ingenieriles, funcionales y artísticos. En este sentido, fue posible evidenciar la capacidad que puede llegar a tener una técnica heurística de optimización (o una combinación de ellas) para servir como plataforma integradora de procesos de diseño arquitectónicos e ingenieriles, permitiendo aproximarse a soluciones integrales y óptimas, en donde el rendimiento del objeto arquitectónico va de la mano con el criterio estético del usuario, sin dejar de lado ninguno de los dos.

Vale la pena resaltar que por medio de PACDMOER, es posible considerar tantas alternativas de diseño como se desee (claro está, teniendo en cuenta que un número mayor de alternativas corresponderá a un tiempo mayor de ejecución), permitiendo una exploración formal significativamente amplia mientras se mejora el rendimiento de la solución (correspondiente a los objetivos considerados), actividad que no sería posible llevar a cabo manualmente por parte del diseñador, o por lo menos no con la complejidad ni con la misma amplitud del campo de exploración.

Por medio de los ejemplos de convergencia llevados a cabo en la sección 4.1, es posible concluir que el algoritmo de optimización incluido en PACDMOER, presenta un comportamiento en el cual todas las soluciones tienden a encontrar valores similares correspondientes a los objetivos planteados, es decir, las soluciones convergen.

Dicho comportamiento se evidencia al considerar un solo objetivo (energía de deformación, peso, o rendimiento acústico), la combinación de dos de ellos (energía de deformación + peso), o la combinación de los tres objetivos en los que se enfocó el procedimiento propuesto de optimización (energía de deformación + peso + rendimiento acústico). Adicionalmente, de esta convergencia se deriva la robustez del acercamiento propuesto, ya que las distintas corridas siempre se iniciaron con diferentes poblaciones iniciales (generadas aleatoriamente), y aun así todas las soluciones tendieron hacia valores similares de rendimiento.

Vale la pena resaltar que el comportamiento de los tres objetivos incluidos en PACDMOER, durante el transcurrir de las iteraciones, no fue el mismo. Por ejemplo, el comportamiento del peso de la estructura tuvo siempre una tendencia constante de decrecimiento, manteniendo el orden numérico en los distintos valores que iba adquiriendo, mientras que la energía de deformación tuvo un comportamiento mucho más aleatorio en las primeras 45 iteraciones (aproximadamente en esta iteración convergen las soluciones), en donde se presentaban picos que podían llegar a ser hasta 50 veces mayores a los valores finales de dicha función objetivo. En cuanto al rendimiento acústico, este objetivo tuvo un comportamiento similar al del peso.

Es importante aclarar que el logro de estas condiciones de convergencia y de robustez, no es una tarea sencilla en este tipo de problemas, debido al gran número de variables (en los ejemplos de convergencia se tenían 72 variables), a la infinidad de posibles valores que cada una de ellas podía tomar, al incontable número de combinaciones que entre ellas se podía realizar, a las significativas variaciones en la magnitud de los valores correspondientes a algunas funciones objetivo (por ejemplo la correspondiente a la energía de deformación), y a la incertidumbre de la influencia que cada objetivo podía ejercer sobre el resultado

de las estructuras reticulares de cubierta al llevar a cabo una optimización simultánea de las tres funciones consideradas.

Adicionalmente, aparte de la convergencia de las soluciones para la geometría de la estructura, también se presentó este mismo comportamiento en la búsqueda del grupo óptimo de parámetros para el funcionamiento del UPSO, que llevó a cabo el PSO incluido en ACCMOUPSO. En este caso, el promedio de los valores encontrados para c_1, c_2, X y u, fue similar para los distintos problemas de convergencia, en donde se consideraron distintas prioridades en la combinación de los objetivos.

Por último, se puede concluir que el Procedimiento Auto-Configurado para el Diseño Multi-Objetivo de Estructuras Reticulares (PACDMOER), propuesto y desarrollado en este proyecto de investigación, funciona correctamente, permitiéndole al diseñador generar y optimizar una estructura reticular de cubierta correspondiente a una determinada planta arquitectónica (ver la sección 4.2). Adicionalmente, por medio de este procedimiento se pueden explorar múltiples soluciones formales, mientras se optimiza el rendimiento de la estructura reticular de cubierta (en relación a los objetivos planteados).

En este sentido, PACDMOER se enfoca hacia el nuevo paradigma de diseño planteado en la introducción del documento, caracterizado por la pretensión de llegar a proponer soluciones arquitectónicas integrales, en donde el componente técnico, funcional (desde la arquitectura), y formal del proyecto que se diseñe, en realidad sea uno solo. Así, será posible aproximarse a exploraciones plásticas interesantes (por ejemplo, como las del arquitecto Frank Gehry, mostradas en la introducción del documento), mejorando continuamente la funcionalidad y la

técnica de la totalidad del objeto, en vez de aproximarse a ellas de manera, por lo general, caprichosa.

BIBLIOGRAFÍA

AERTS, J.C.J.H., EISINGER, E., HEUVELINK, G.B.M. y STEWART, T.J. Using linear integer programming for multi-site land-use allocation. En: Geographical Analysis. No. 25 (2003); p. 148-169.

ALTSHULLER, G. The innovation algorithm: TRIZ, systematic innovation and technical creativity. Technical Innovation Center, 1999.

BADER, J. y ZITZLER, E. Hype: An algorithm for fast hypervolume-based many-objective optimization. En: Evolutionary Computation. No. 19 (2011); p. 45-76.

BODEN, M.A. The creative mind: myths and mechanisms. New York: Basic Books, 1992.

BUREERAT, S. y SRIWORAMAS, K. Population-based incremental learning for multi-objective optimisation. En: Soft Computing in Industrial Applications. No. 39 (2007); p. 223-232.

BYRNE, J., FENTON, M., HEMBERG, E., MCDERMOTT, J., O'NEIL, M., SHOTTON, E. y NALLY, C. Combining structural analysis and multi-objective criteria for evolutionary architectural design. En: Applications of Evolutionary Computation. No. 1 (2011); p. 204-213.

CAGNE, J.M.L. y ANDERSEN, M. Multi-objective facade optimization for daylighting design using a Genetic Algorithm. En: SIMBUILD 2010 - 4TH NATIONAL CONFERENCE OF IBPSA-USA. Proceedings of SimBuild 2010 - 4th National Conference of IBPSA-USA. New York: s.e., 2010.

CAI, Z. y WANG, Y. A multiobjective optimization-based evolutionary algorithm for constrained optimization. En: IEEE Transactions on Evolutionary Computation. No. 10 (2006); p. 658-675.

CARPO, M. The Demise of the Identical Architectural Standarization in the Age of Digital Reproducibility. En: FIRST INTERNATIONAL CONFERENCE ON THE HISTORIES OF MEDIA ART, SCIENCE AND TECHNOLOGY. Banff New Media Institute, 2005.

CHANDRUPATLA, T. y BELEGUNDU, A. Introduction to finite elements in engineering. Upper Saddle River: Prentice Hall, 2011.

CHARNES, A. y COOPER, W.W. Management models and industrial applications of linear programming. New York: John Wiley, 1961.

CHEN, Y., LIU, Z.L. y XIE, Y.B. A knowledge-based framework for creative conceptual design of multi-disciplinary systems. En: Computer-Aided Design. No. 44 (2012); p. 146-153.

CHEN, Y.L. y LIU, C.C. Multiobjective VAR planning using the goal-attainment method. En: IEEE Proc Generat Transm Distrib. No. 141 (1994); p. 227-232.

CLERC, M. The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization. En: ICEC. Proceedings of ICEC. Washington: s.e., 1999. p. 1951-1957.

COELLO COELLO, C.A. An updated survey of GA-based multiobjective optimization techniques. En: ACM Comput Surveys. No. 32 (2000); p. 109-143.

COELLO COELLO, C.A. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. En: Computer Methods in Applied Mechanics and Engineering. No. 191 (2002); p. 1245-1287.

COELLO, C.A. y CHRISTIANSEN, A.D. Multiobjective optimization of trusses using genetic algorithms. En: Computers and Structures. No. 75 (2000); p. 647-660.

COELLO COELLO, C.A. y CHRISTIANSEN, A.D. Two new GA-based methods for multiobjective optimization. En: Civil Eng Syst. No. 15 (1998); p. 207-243.

CORNE, D. y KNOWLES, J.D. Techniques for highly multiobjective optimisation: Some nondominated points are better than others. En: Conference on Genetic and Evolutionary Computation. No. 1 (2007); p. 773-780.

DANCE, S. The development of computer models for the prediction of sound distribution in fitted non-diffuse spaces. Londres, 1993. Tesis doctoral. South Bank University.

DE BERG, M. Computational Geometry: Algorithms and Applications. Berlín: Springer, 2008.

DEB, K. Evolutionary algorithms for multi-criterion optimization in engineering design. En: MIETTINEN, K. et al. Evolutionary algorithms in engineering and computer science: recent advances in genetic algorithms, evolution strategies, evolutionary programming, and industrial applications. New York: John Wiley & Sons, 1999.

DEB, K. Multi-objective genetic algorithms: problem difficulties and construction of test problems. En: Evolutionary Computation. No. 7 (1999); p. 205-230.

DEB, K., MOHAN, M. y MISHRA, S. Evaluating the epsilon-domination based multi-objective evolutionary algorithm for a quick computation of Pareto-optimal solutions. En: Evolutionary Computation. No. 13 (2005); p. 501-525.

DEB, K., PRATAP, A., AGARWAL, S. y MEYARIVAN, T. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. En: IEEE Transactions on Evolutionary Computation. No. 6 (2002); p. 182-197.

DORIGO, M. y COLORNI, A. The Ant System: Optimization by a colony of cooperating agents. En: IEEE Transactions on Systems, Man, and Cybernetics. No. 26 (1996); p. 1-13.

DUH, J.D. y BROWN, D.G. Knowledge-informed Pareto simulated annealing for multi-objective spatial allocation. En: Computers, Environment and Urban Systems. No. 31 (2007); p. 253-281.

EBERHART, R.C. AND SHI, Y. Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization. En: CONGRESS ON EVOLUTIONARY COMPUTING. Proceedings of 2000 Congress on Evolutionary Computing (CEC2000). La Jolla: s.e., 2000. p. 84-88.

ELHOSSINI, A., AREIBI, S. y DONY, R. Strength Pareto particle swarm optimization and hybrid EA-PSO for multi-objective optimization. En: Evolutionary Computation. No. 18 (2010); p. 127-156.

FIALHO, A., HAMADI, Y. y SCHOENAUER, M. Optimizing Architectural and Structural Aspects of Buildings towards Higher Energy Efficiency. En: 13TH ANNUAL CONFERENCE COMPANION ON GENETIC AND EVOLUTIONARY COMPUTATION. Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation GECCO'11. New York: ACM, 2011. p. 727-732.

FOGEL, L.J., OWENS, A.J. y WALSH, M.J. Artificial Intelligence through simulated evolution. Chichester: John Wiley, 1966.

FONSECA, C.M. y FLEMING, P.J. Genetic algorithms for multi-objective optimization: formulation, discussion, and generalization. En: 5th INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS. Proceedings of the fifth international conference on genetic algorithms. Los Altos: Morgan Kauffman, 1993. p. 416-423.

GASPAR-CUNHA, A. Modeling and Optimization of Single Screw Extrusion. Minho, 2000. Tesis doctoral. University of Minho.

GASPAR-CUNHA, A., LOYENS, D. y VAN HATTUM, F. Aesthetic Design Using Multi-Objective Evolutionary Algorithms. En: 6th INTERNATIONAL CONFERENCE ON EVOLUTIONARY MULTI-CRITERION OPTIMIZATION. Proceedings of the 6th International Conference on Evolutionary Multi-Criterion Optimization. Heidelberg: Springer-Verlag Berlin, 2011. p. 374-388.

GASPAR-CUNHA, A. y COVAS, J.A. RPSGAe - A Multiobjective Genetic Algorithm with Elitism: Application to Polymer Extrusion. En: Metaheuristics for Multi-objective Optimisation. Heidelberg: Springer, 2004.

GERO, J.S. Computers and creative design. En: The global design studio. No. 1 (1996); p. 11-19.

GLOVER, F. Future paths for integer programming and links to artificial intelligence. En: Comput. & Ops. Res. No. 13 (1986); p. 533-549.

GOLDBERG, D.E. Genetic algorithms in search, optimization, and machine learning. Reading: Addison-Wesley, 1989.

HAJELA, P. y LIN, C.Y. Genetic search strategies in multicriterion optimal design. En: Structural optimization. No. 4 (1992); p. 99-107.

HANSEN, M.P. Tabu Search for Multiobjective Optimization: MOTS. En: 13TH INTERNATIONAL CONFERENCE ON MULTIPLE CRITERIA DECISION MAKING. Proceedings of the 13th International Conference on Multiple Criteria Decision Making (MCDM'97). Cape Town: s.e., 1997.

HENDERSON, D., JACOBSON, S.H. y JOHNSON, W. The theory and practice of simulated annealing. En: GLOVER, F. y KOCHENBERGER, G.A. Handbook of Metaheuristics. Kluwer Academic Publishers, 2003. p. 287-319.

HOEFFLER, A., LEYSNER, U. y WEIDERMANN, J. Optimization of the layout of trusses combining strategies based on Mitchel's theorem and on biological principles of evolution. En: II SYMPOSIUM ON STRUCTURAL OPTIMIZATION. Milan: s.e., 1973.

HOLLAND, J.H. Adaptation in natural and artificial systems. Ann Arbor: University of Michigan Press, 1975.

HORN, J. AND NAFPLIOTIS, N. Multiobjective optimization using the Niched Pareto genetic algorithm. Urbana, 1993. Reporte técnico. University of Illinois at Urbana Champaigne.

HU, X., EBERHART, R.C. y SHI, Y. Engineering Optimization with Particle Swarm. En: IEEE SWARM INTELLIGENCE SYMPOSIUM. Proceedings of the 2003 IEEE Swarm Intelligence Symposium. Indianapolis: IEEE Service Center, 2003. p. 53-57.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION - ISO. Attenuation of sound during propagation outdoors, Part 1: Calculation of the absorption of sound by the atmosphere. Geneve: International Organization for Standardization, 1993. (ISO 9613-1).

KARGAHI, M., ANDERSON, J.C. y DESSOUKY, M.M. Structural weight optimization of frames using tabu search: Optimization procedure. En: Journal of Structural Engineering ASCE. No. 132 (2006); p. 1858-1868.

KAVEH, A. y TALATAHARI, S. An improved ant colony optimization for constrained engineering design problems. En: Engineering Computations. No. 27 (2010); p. 155-182.

KENNEDY, J. y EBERHART, R. Particle Swarm Optimization. En: IEEE INTERNATIONAL CONFERENCE ON NEURAL NETWORKS. Proceedings of the 1995 IEEE International Conference on Neural Networks. Piscataway: IEEE Service Center, 1995. p. 1942-1948.

KICINGER, R., ARCISZEWSKI, T. y DE JONG, K. Evolutionary computation and structural design: A survey of the state-of-the-art. En: Computers and Structures. No. 83 (2005); p. 1943-1978.

KIRKPATRICK, S., GELATT, C.D. y VECCHI, M.P. Optimization by Simulated Annealing. En: Science. No. 220 (1983); p. 671-680.

KNOWLES, J.D. y CORNE, D.W. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. En: Evolutionary Computation. No. 8 (2000); p. 149-172.

KONG, X.S., LIU, Y.F. y TAN, C.F. Correlation analysis of the changes of land use structure and land use efficiency: A case study of Jiayu County in Hubei Province. En: Resources Science. No. 31 (2009); p. 1195-1101.

KOULAMAS, C., ANTONY, S.R. y JAEN, R. A survey of simulated annealing applications to operations-research problems. En: OMEGA - International Journal of Management Science. No. 22 (1994); p. 41-56.

KOZA, J.R., BENNETT III, F.H., ANDRE, D. y KEANE, M.A. Genetic Programming: Biologically Inspired Computation That Exhibits Creativity in Producing Human-Competitive Results. En: BENTLEY, P.J. y CORNE, D.W. Creative Evolutionary Systems. Londres: Academic Press, 2002. p. 275-298.

KROKSTAD, A., STROM, S. y SORSDAL, S. Calculation of the acoustical room response by the use of a ray tracing technique. En: Journal of Sound and Vibrations. No. 8 (1968); p. 118-125.

LARA, A., SÁNCHEZ, G., COELLO COELLO, C.A. y SCHUTZE, O. HCS: a new local search strategy for memetic multiobjective evolutionary algorithms. En: IEEE Transactions on Evolutionary Computation. No. 14 (2010); p. 112-132.

LAUMANNS, M., THIELE, L., DEB, K. y ZITZLER, E. Combining Convergence and Diversity in Evolutionary Multi-objective Optimization. En: Evolutionary Computation. No. 10 (2002); p. 263-282.

LAWO, M. y THIERAUF, G. Optimal design for dynamic stochastic loading: a solution by random search. En: Optimization in structural design. University of Siegen (1982); p. 346-352.

LIU, Y., WANG, H., JI, Y., LIU, Z. y ZHAO, X. Land Use Zoning at the County Level Based on a Multi-Objective Particle Swarm Optimization Algorithm: A Case Study from Yicheng, China. En: International Journal of Environmental Research and Public Health. No. 9 (2012); p. 2801-2826.

MARIANO, C.E. y MORALES, E. MOAQ an Ant-Q Algorithm for Multiple Objective Optimization Problems. En: GENETIC AND EVOLUTIONARY COMPUTING CNFERENCE. Proceedings of the Genetic and Evolutionary Computing Cnference (GECCO 99). San Francisco: Morgan Kaufmann, 1999. p. 894-901.

MARIANO, C.E. y MORALES, E. A New Distributed Reinforcement Learning Algorithm for Multiple Objective Optimization Problems. Mexico, 2000. Reporte técnico. Instituto Mexicano de Tecnología del Agua.

METROPOLIS, N., ROSENBLUTH, A., ROSENBLUTH, M., TELLER, A. y TELLER, E. Equation of State Calculations by Fast Computing Machines. En: Journal of Chemical Physics. No. 21 (1953); p. 1087-1092.

MICHALEK, J.J., CHOUDHARY, R. y PAPALAMBROS, P.Y. Architectural Layout Design Optimization. En: Engineering Optimization. No. 34 (2002); p. 461-484.

MONKS, M.C. Audioptimization: Goal-Based Acoustic Design. Massachusetts, 1999. Tesis doctoral. Massachusetts Institute of Technology.

NOILUBLAO, N. y BUREERAT, S. Simultaneous topology, shape and sizing optimisation of a three-dimensional slender truss tower using multiobjective evolutionary algorithms. En: Computers and Structures. No. 89 (2011); p. 2531-2538.

NOILUBLAO, N. y BUREERAT, S. Simultaneous topology, shape and sizing optimisation of skeletal structures using multiobjective evolutionary algorithms. En: Evolutionary Computation. No. 24 (2009); p. 487-580.

OGAWA, K. y TADA, M. Computer program for static and dynamic analysis of steel frames considering the deformation of joint panel. En: 17TH SYMPOSIUM ON COMPUTER TECHNOLOGY OF INFORMATION. Proceedings of the Seventeenth

Symposium on Computer Technology of Information. Systems and Applications. p. 79-84.

OHSAKI, M. y KINOSHITA, T. Single-Point Search Heuristic Methods for Multiobjective Structural Optimization. En: FCS/TECHNO-SYMPO/MPS SYMPOSIUM 2005. Proceedings of Computational Science Symposium. p. 59-66.

ONDET, A. y BARBRY, J. Modelling of sound propagation in fitted workshops using ray tracing. En: Journal of the Acoustical Society of America. No. 85 (1989); p. 787-796.

O'NEIL, M. Automatic Programming in an Arbitrary Language: Evolving Programs with Grammatical Evolution. Limerick, 2001. Tesis doctoral. University of Limerick.

OXMAN, R.Theory and design in the first digital age. En: Design Studies. No. 27 (2006); p. 229-266.

PARETO, V. Cours D'Economie Politique. Lausanne: F. Rouge, 1896.

PARMEE, I.C. y BONHAM, C.R. Improving cluster oriented genetic algorithms for high-performance region identification. En: US UNITED ENGINEERING FOUNDATION'S OPTIMISATION IN INDUSTRY CONFERENCE. Proceedings US United Engineering Foundation's Optimisation in Industry Conference. Tuscany: Springer-Verlag, 2002.

PARSOPOULOS, K.E. y M.N., VRAHATIS. UPSO: A Unified Particle Swarm Optimization Scheme. En: Lecture Series on Computer and Computational Sciences. No. 1 (2004); p. 868-873.

PARSOPOULOS, K.E. y VRAHATIS, M.N. Particle Swarm Optimization method in multiobjective problems. En: Proceedings ACM Symposium on Applied Computing (SAC 2002). 2002. pp. 603-607.

PARSOPOULOS, KE y VRAHATIS, MN. Particle Swarm optimization method for constrained optimization problems. En: 2nd EURO-INTERNATIONAL SYMPOSIUM ON COMPUTATIONAL INTELLIGENCE. Proceedings of the second Euro-International Symposium on Computational Intelligence. Kosice: Kvasnicka, V. et al., 2002. p. 214-220.

PAYA, I., YEPES, V., GONZÁLEZ-VIDOSA, F. y HOSPITALER, A. Multiobjective Optimization of Concrete Frames by Simulated Annealing. En: Computer-Aided Civil and Infraestructure Engineering. No. 23 (2008); p. 596-610.

PEREZ, R.E. y BEHDINAN, K. Particle swarm approach for structural design optimization. En: Computers & Structures. No. 85 (2007); p. 1579-1588.

PIEGL, L. y TILLER, W. The NURBS book. Berlin: Springer, 1997.

PUGNALE, A. y SASSONE, M. Morphogenesis and structural optimization of shell structures with the aid of a Genetic Algorithm. En: Journal of the International Association for Shell and Spatial Structures. No. 48 (2007); p. 161-166.

RAO, S.S. Multiobjective optimization in structural design with uncertain parameters and stochastic processes. En: AIAA Journal. No. 22 (1984); p. 1670-1678.

RECHENBERG, I. Cybernetic solution path of an experimental problem. Farnborough, 1965. Library Translation 1122. Royal Aircraft Establishment.

RECHENBERG, I. Evolutionsstrategie: Optimierung TechnischerSysteme nach Prinzipien der biologischen Evolution. Stuttgart-Bad Cannstatt: Frommann-Holzboog, 1973.

REYES-SIERRA, M. y COELLO COELLO, C.A. Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art. En: International Journal of Computational Intelligence Research. No. 2 (2006); p. 287-308.

RICHARDSON, J.N., ADRIAENSSENS, S., COELHO, R.F. y BOUILLARD, P. Coupled form-finding and grid optimization approach for single layer grid shells. En: Engineering Structures. No. 52 (2013); p. 230-239.

ROSENBERG, R.S. Simulation of genetic populations with biochemical properties. Ann Harbor, 1967. Tesis doctoral. University of Michigan.

ROSENMAN, M. The generation of form using evolutionary approach. En: DASGUPTA, D. y MICHALEWICZ, Z. Evolutionary algorithms in engineering applications. New York: Springer, 1997. p. 69-86.

ROSEMAN, M.A. y GERO, J.S. Reducing the Pareto Optimal Set in Multicriteria Optimization. En: Eng. Optim. No. 8 (1985); p. 189-206.

SASSONE, M., MÉNDEZ, T. y PUGNALE, A. On the interaction between architecture and engineering: the acoustic optimization of a reinforced concrete shell. En: 6TH INTERNATIONAL CONFERENCE ON COMPUTATION OF SHELL AND SPATIAL STRUCTURES. Proceedings of the 6th International Conference on Computation of Shell and Spatial Structures IASS-IACM 2008: Spanning Nano to Mega. Ithaca: J.F. Abel and J.R Cooke, 2008.

SASSONE, M. y PUGNALE, A. Optimal design of glass grid shells with quadrilateral elements by means of a genetic algorithm. En: 6TH INTERNATIONAL CONFERENCE ON COMPUTATION OF SHELL AND SPATIAL STRUCTURES. Proceedings of the 6th International Conference on Computation of Shell and Spatial Structures IASS-IACM 2008: Spanning Nano to Mega. Ithaca: J.F. Abel and J.R Cooke, 2008.

SCHAFFER, J.D. Some experiments in machine learning using vector evaluated genetic algorithms. Nashville, 1984. Tesis doctoral. Vanderbilt University.

SCHAFFER, J.D. Multiple objective optimization with vector evaluated genetic algorithms. En: FIRST INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS. Proceedings on the First International Conference on Genetic Algorithms (ICGA'85). Pittsburgh: Grefenstette JJ, 1985. p. 93-100.

SCHWEFEL, H.P. Kybernetische Evolution als Strategie der experimentelen Forschung in der Stromungstechnik. Berlín, 1965. Tesis de maestría. Technical University of Berlin. Hermann Fottinger Institute for Hydrodynamics.

SERAFINI, P. Simulated Annealing for Multiple Objective Optimization Problems. En: TZENG, G. et al. Multiple Criteria Decision Making: Expand and Enrich the Domains of Thinking and Application. Berlin: Springer Verlag, 1994. p. 283-292.

SRINIVAS, N. y DEB, K. Multiobjective optimization using non-dominated sorting in genetic algorithms. Kanpur, 1993. Reporte técnico. Indian Institute of Technology. Department of mechanical engineering.

STEWART, T.J., JANSSEN, R. y VAN HERWIJNEN, M. A genetic algorithm approach to multi-objective land use planning. En: Computers and Operations Research. No. 31 (2004); p. 2293-2313.

STRAND, R., CRAWLEY, D., PEDERSEN, C., LIESEN, R., LAWRIE, L., WINKELMANN, F., BUHL, W., HUANG, Y. y FISHER, D. EnergyPlus: A new-generation energy analysis and load calculation engine for building design. En: Proc. Association of Collegiate Schools of Architecture Technology Conference. 2000.

SUPPAPITNARM, A., SEFFEN, K.A., PARKS, G.T. y CLARKSON, P.J. A simulated annealing algorithm for multiobjective optimization. En: Engineering Optimization. No. 33 (2000); p. 59-85.

SYSWERDA, G. y PALMUCCI, J. The application of genetic algorithms to resource scheduling. En: FOURTH INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS. Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA'91). San Diego: Belew, R.K. y Booker, L.B., 1991. p. 502-508.

SZOLLOS, A., SMID, M. y HAJEK, J. Aerodynamic optimization via multi-objective micro-genetic algorithm with range adaptation, knowledge-based reinitialization, crowding and e-dominance. En: Advances in Engineering Software. No. 40 (2009); p. 419-430.

TAKAGI, H. Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. En: Proc. of the IEEE. No. 89 (2001); p. 1275-1296.

TAN, K.C., YANG, Y.L. y GOH, C.K. A distributed cooperative coevolutionary algorithm for multiobjective optimization. En: IEEE Transactions on Evolutionary Computation. No. 10 (2006); p. 527-549.

TOSCANO-PULIDO, G. On the Use of Self-Adaptation and Elitism for Multiobjective Particle Swarm Optimization. Mexico, 2005. Tesis doctoral. CINVESTAV-IPN. Department of Electrical Engineering. Computer Science Section.

TOSCANO-PULIDO, G. y COELLO COELLO, C.A. Using Clustering Techniques to Improve the Performance of a Particle Swarm Optimizer. En: GENETIC AND

EVOLUTIONARY COMPUTATION-GECCO 2004. Proceedings of the Genetic and Evolutionary Computation Conference. Seattle: Springer-Verlag, 2004. p. 225-237.

TOSCANO-PULIDO, G. y COELLO COELLO, C.A. A Constraint-Handling Mechanism for Particle Swarm Optimization. En: CONGRESS ON EVOLUTIONARY COMPUTATION 2004. Proceedings of the Congress on Evolutionary Computation 2004 (CEC'2004). Piscataway: IEEE Service Center, 2004. p. 1396-1403.

TOSCANO-PULIDO, G., SANTANA-QUINTERO, L.V. AND COELLO COELLO, C.A. EMOPSO: A Multi-Objective Particle Swarm Optimizer with Emphasis on Efficiency. En: EMO 2007. Proceedings of EMO 2007. Heidelberg: Springer, 2007. p. 272-285.

TURRIN, M., VON BUELOW, P. y STOUFFS, R. Design explorations of performance driven geometry in architectural design using parametric modeling and genetic algorithms. En: Advanced Engineering Informatics. No. 25 (2011); p. 656-675.

VELDHUIZEN, D.A.V. Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. Ohio, 1999. Reporte técnico. Air Force Institute of Technology. Graduate School of Engineering. Department of Electrical and Computer Engineering.

WANG, W., RIVARD, H. y ZMEUREANU, R. Floor shape optimization for green building design. En: Advanced Engineering Informatics. No. 20 (2006); p. 363-378.

WINSLOW, P., PELLEGRINO, S. y SHARMA, S.B. Multi-objective optimization of free-form grid structures. En: Struct Multidisc Optim. No. 40 (2010); p. 257-269.

WOODBURY, L. y BURROW, A.L. Whither design space?. En: Artificial Intelligence for Engineering Design, Analysis and Manufacturing. No. 20 (2006); p. 63-82.

YANG, D., JIAO, L. y GONG, M. Adaptive multi-objective optimization based on nondominated solutions. En: Computational Intelligence. No. 25 (2009); p. 84-108.

YANG, L. y SHIELD, B. Development of a ray tracing computer model for the prediction of the sound field in long enclosures. En: Journal of Sound and Vibration. No. 229 (2000); p. 133-146.

ZHANG, Q. y LI, H. MOEA/D: a multiobjective evolutionary algorithm based on decomposition. En: IEEE CONGRESS ON EVOLUTIONARY COMPUTATION. Proceedings of the IEEE Congress on Evolutionary Computation. 2007. p. 712-731.

ZHOU, A., QU, B.Y., LI, H., ZHAO, S.Z., NAGARATNAM SUGANTHAN, P. y ZHANG, Q. Multiobjective evolutionary algorithms: A survey of the state of the art. En: Swarm and Evolutionary Computation. No. 1 (2011); p. 32-49.

ZHOU, J.L. y TITS, A.L. An SQP algorithm for finely discretized continuous minimax problems and other minimax problems with many objective functions. En: SIAM Journal of Optimization. No. 6 (1995); p. 461-487.

ZITZLER, E., DEB, K. y THIELE, L. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. En: Evolutionary Computation. No. 8 (2000); p. 173-195.

ZITZLER, E. y KÜNZLI, S. Indicator-based selection in multi-objective search. En: Parallel Problem Solving from Nature. No. 7 (2004); p. 832-842.

ZITZLER, E. AND THIELE, L. An evolutionary algorithm for multi-objective optimization: the strength Pareto approach. Zurich, 1998. Reporte técnico. Swiss Federal Institute of Technology. Computer engineering and Communication Networks Laboratory.

ZOU, X., CHEN, Y., LIU, M. y KANG, L. A new evolutionary algorithm for solving many-objective optimization problems. En: IEEE Transactions on Systems, Man and Cybernetics. No. 38 (2008); p. 1402-1412.

ANEXOS

ANEXO A: Definición desarrollada en Grasshopper + LunchBox

En la plancha que se presenta a continuación, se expone la definición que se creó en Grasshopper para llevar a cabo la generación automática de la geometría de la estructura reticular de cubierta (ver sección 3.1).

Para ésta definición, fue necesaria la inclusión del plug-in LunchBox. La definición está dividida en secciones, o en partes, las cuales corresponden a las partes del código que se presenta en el Anexo B. Es importante resaltar que cada vez que se genera o se lee un archivo .txt en ésta definición de Grasshopper, se está generando una interacción con el código desarrollado en MatLab, presentado en los anexos B, C, D y E.

ANEXO B: Código en MatLab para la generación automática de geometría

Parte 1 (correspondiente a la definición en Grasshopper):

```
%-------------------------------------------------------------------------
%LECTURA Y CREACIÓN DE LOS PUNTOS DE APOYO
%-------------------------------------------------------------------------
%Leer un archivo .txt con las coordenadas de los nodos de la base, creado en Grasshopper como una lista de números, no como grupos de coordenadas (corresponde al paso 1 de la definición en Grasshopper).
BasePoints=load('BasePoints.txt');

%Crear los vectores que contendrán las coordenadas X, Y y Z de los puntos de apoyo, con los datos de la lista del archivo "BasePoints.txt". En el for, el 3i-2, 3i-1 y 3i es porque debe tomar los índices correspondientes a cada coordenada, teniendo en cuenta que el archivo .txt es una lista de números, no unos grupos de coordenadas.
BasePointsX=zeros((length(BasePoints)/3),1);
BasePointsY=zeros((length(BasePoints)/3),1);
BasePointsZ=zeros((length(BasePoints)/3),1);

for i=1:(length(BasePoints)/3)
    BasePointsX(i)=BasePoints((3*i)-2);
    BasePointsY(i)=BasePoints((3*i)-1);
    BasePointsZ(i)=BasePoints(3*i);
end

%Crear un límite inferior para la generación de la coordenada Z de todos los puntos de la retícula y para, posteriormente, aplicar una restricción espacial.
MinHeight=min(BasePointsZ)-(0.5*min(BasePointsZ));

%Crear un límite superior con los mismos fines del inferior. En este caso, este límite se define para que sea posible la generación hasta de una tipología estructural de domo esférico.
```

```
if                (max(BasePointsX)-min(BasePointsX))<=(max(BasePointsY)-
min(BasePointsY))
    MaxHeight=max(BasePointsZ)+(max(BasePointsY)-min(BasePointsY))/2;
elseif             (max(BasePointsX)-min(BasePointsX))>(max(BasePointsY)-
min(BasePointsY))
    MaxHeight=max(BasePointsZ)+(max(BasePointsX)-min(BasePointsX))/2;
end

%Crear la matriz que almacenará las coordenadas de los puntos de apoyo,
con los vectores de coordenadas X, Y y Z de dichos puntos.
BasePointsCo=zeros((length(BasePoints)/3),3);

for i=1:(length(BasePoints)/3)
    BasePointsCo(i,1)=round(BasePointsX(i)*100)/100;
    BasePointsCo(i,2)=round(BasePointsY(i)*100)/100;
    BasePointsCo(i,3)=round(BasePointsZ(i)*100)/100;
end

%-------------------------------------------------------------------------
%LECTURA Y CREACIÓN DE LOS PUNTOS INICIALES DE LA RETÍCULA
%-------------------------------------------------------------------------

%Leer un archivo .txt en donde se evalúa si los puntos de la retícula
están contenidos o no dentro del perímetro que definen los puntos de
apoyo. Este archivo se crea en Grasshopper y sus convenciones son las
siguientes: 0 si el punto está fuera del perímetro, 1 si el punto
coincide con el perímetro, y 2 si el punto está dentro del perímetro
(corresponde al paso 2 de la definición en Grasshopper).
ContPoints=load('ContPoints.txt');

%Contabilizar la cantidad de puntos que quedan por fuera del perímetro
definido por los puntos de apoyo.
nncp=0;

for i=1:length(ContPoints)
    if ContPoints(i)==0
```

```
        nncp=nncp+1;
    end
end

%Ordenar descendentemente la información (vector columna) obtenida del
archivo "ContPoints.txt". De acá se obtiene una lista ordenada (vector
SortContPoints) y un vector (ItemContPoints) con los índices que le
correspondían a los números ahora ordenados, dentro del vector original
(ContPoints).
[SortContPoints,ItemContPoints]=sort(ContPoints,'descend');

%Crear un vector que contendrá los índices de los puntos que coinciden
con, o que están contenidos en, el perímetro definido por los puntos de
apoyo (este vector se necesita en Grasshopper para continuar con el
procedimiento). Por este motivo su longitud es igual a la longitud de
ContPoints menos la cantidad de números que no cumplen con la condición
anterior (nncp).
ItemCP=zeros((length(ContPoints)-nncp),1);

for i=1:(length(ContPoints)-nncp)
    ItemCP(i)=(ItemContPoints(i)-1); Este vector se llena con los índices
    de ItemContPoints (los correspondientes a los puntos que sí coinciden
    o que sí están contenidos en el perímetro; para esto fue que se
    ordenó descendentemente el vector original ContPoints y se calculó el
    número de puntos no contenidos o no coincidentes) y, a cada valor se
    le resta 1, ya que estos datos son para exportar a Grasshopper y, en
    dicho plug-in, los índices empiezan en 0, no en 1 como en MatLab.
end

%Imprimir un archivo .txt con los Items de los puntos que coinciden o que
están dentro del perímetro. Este archivo será leído en Grasshopper para
filtrar los puntos de la retícula que allí se genera (corresponde al paso
3 de la definición en Grasshopper).
dlmwrite('ItemCP.txt',ItemCP);
```

```
%Leer un archivo .txt con las coordenadas (como una lista de números, no como grupos de coordenadas) de los puntos que coinciden o que están contenidos en el perímetro definido por los puntos de apoyo. Este archivo se crea en Grasshopper con la información de los índices generados anteriormente (ItemCP) (corresponde al paso 4 de la definición en Grasshopper).
InGridPoints=load('InGridPoints.txt');

%Crear un vector que contendrá las coordenadas X, Y y Z de los puntos de la retícula que están contenidos o que coinciden con el perímetro definido por los puntos de apoyo. En el for, el 3i-2, 3i-1 y 3i es porque debe tomar los índices correspondientes a cada coordenada, teniendo en cuenta que el archivo .txt es una lista de números, no unos grupos de coordenadas.
InGridPointsX=zeros((length(InGridPoints)/3),1);
InGridPointsY=zeros((length(InGridPoints)/3),1);
InGridPointsZ=zeros((length(InGridPoints)/3),1);

for i=1:(length(InGridPoints)/3)
    InGridPointsX(i)=InGridPoints((3*i)-2);
    InGridPointsY(i)=InGridPoints((3*i)-1);
    InGridPointsZ(i)=MinHeight+rand*(MaxHeight-MinHeight);    %Llena el vector de la coordenada Z con unos valores aleatorios dentro del rango de alturas definido por MinHeight y MaxHeight (ver la sección “LECTURA Y CREACIÓN DE LOS PUNTOS DE APOYO”). Lo anterior se hace porque la retícula de puntos que se crea en Grasshopper es sobre una superficie plana, sin alturas.
end

%Crear una matriz que contendrá las coordenadas X, Y y Z de los puntos de la retícula que están contenidos o que coinciden con el perímetro definido por los puntos de apoyo. Esto se hace con los vectores de coordenadas X, Y y Z de los puntos coincidentes o contenidos en el perímetro (InGridPointsX).
InGridPointsCo=zeros((length(InGridPoints)/3),3);
```

```
for i=1:(length(InGridPoints)/3)
    InGridPointsCo(i,1)=InGridPointsX(i);
    InGridPointsCo(i,2)=InGridPointsY(i);
    InGridPointsCo(i,3)=InGridPointsZ(i);
end

%-------------------------------------------------------------------------
%LECTURA Y CREACIÓN DE LOS PUNTOS DE BORDE
%-------------------------------------------------------------------------

%Leer un archivo .txt con las coordenadas (como una lista de números, no
como grupos de coordenadas) de los puntos que se encuentran en el borde
de la retícula. Este archivo se crea en Grasshopper con las
intersecciones entre el perímetro definido por los puntos de apoyo, y las
líneas de la retícula (corresponde al paso 5 de la definición en
Grasshopper).
BorderPoints=load('BorderPoints.txt');

%Crear un vector que contendrá las coordenadas X, Y y Z de los puntos que
se encuentran en el borde, con los datos de la lista del archivo
"BorderPoints.txt". En el for, el 3i-2, 3i-1 y 3i es porque debe tomar
los índices correspondientes a cada coordenada, teniendo en cuenta que el
archivo .txt es una lista de números, no unos grupos de coordenadas.
BorderPointsX=zeros((length(BorderPoints)/3),1);
BorderPointsY=zeros((length(BorderPoints)/3),1);
BorderPointsZ=zeros((length(BorderPoints)/3),1);

for i=1:(length(BorderPoints)/3)
    BorderPointsX(i)=BorderPoints((3*i)-2);
    BorderPointsY(i)=BorderPoints((3*i)-1);
    BorderPointsZ(i)= BorderPoints(3*i);
end

%Crear la matriz que almacenará las coordenadas de los puntos que están
en el borde, con los vectores de coordenadas X, Y y Z de los puntos de
borde (BorderPointsX, BorderPointsY, BorderPointsZ).
```

```
BorderPointsCo=zeros((length(BorderPoints)/3),3);

for i=1:(length(BorderPoints)/3)
    BorderPointsCo(i,1)=BorderPointsX(i);
    BorderPointsCo(i,2)=BorderPointsY(i);
    BorderPointsCo(i,3)=BorderPointsZ(i);
end

%-------------------------------------------------------------------------
%UNIÓN DE LOS PUNTOS DE APOYO CON LOS PUNTOS INICIALES DE LA RETÍCULA, Y CON %LOS PUNTOS DE BORDE
%-------------------------------------------------------------------------
%Crear una matriz con las coordenadas de los puntos de apoyo, con las de los puntos iniciales de la retícula, y con las de los puntos de borde. Muy seguramente quedarán algunos puntos repetidos, pero estos se filtrarán en pasos posteriores.
InPopPoints=[BasePointsX,BasePointsY,BasePointsZ;BorderPointsX,BorderPointsY,BorderPointsZ;InGridPointsX,InGridPointsY,InGridPointsZ];

%Imprimir un archivo .txt con la matriz que reúne las coordenadas de los puntos de apoyo, las de los puntos iniciales de la retícula, y las de los puntos de borde. Este archivo será utilizado en Grasshopper para ordenar todos los puntos (corresponde al paso 6 de la definición en Grasshopper).
dlmwrite('GridPoints.txt',InPopPoints);
```

PARTE 2 (correspondiente a la definición en Grasshopper):

```
%-------------------------------------------------------------------------
%POBLACIÓN INICIAL DE PUNTOS
%-------------------------------------------------------------------------

%Leer un archivo .txt con las coordenadas (como una lista de números, no como grupos de coordenadas) de todos los puntos (ordenados) de la retícula. Este archivo se hace en Grasshopper ordenando los puntos que se leen del archivo "GridPoints.txt" (corresponde al paso 7 de la definición en Grasshopper).
```

```
PopPoints=load('OrderedGridPoints.txt');

%Crear los vectores que contendrán las coordenadas X, Y y Z de todos los puntos (ordenados) de la retícula. En el for, el 3i-2, 3i-1 y 3i es porque debe tomar los índices correspondientes a cada coordenada, teniendo en cuenta que el archivo .txt es una lista de números, no unos grupos de coordenadas.
PopPointsX=zeros((length(PopPoints)/3),1);
PopPointsY=zeros((length(PopPoints)/3),1);
PopPointsZ=zeros((length(PopPoints)/3),1);

for i=1:(length(PopPoints)/3)
    PopPointsX(i)=PopPoints((3*i)-2);
    PopPointsY(i)=PopPoints((3*i)-1);
    PopPointsZ(i)=PopPoints(3*i);
end
%Crear la matriz que almacenará las coordenadas de todos los puntos (ordenados) de la retícula, con los vectores de coordenadas X, Y y Z de todos los puntos (ordenados) de la retícula (PopPointsX, PopPointsY, PopPointsZ).
PopPointsCo=zeros((length(PopPoints)/3),3);

for i=1:(length(PopPoints)/3)
    PopPointsCo(i,1)=PopPointsX(i);
    PopPointsCo(i,2)=PopPointsY(i);
    PopPointsCo(i,3)=PopPointsZ(i);
end

%-------------------------------------------------------------
%Borrar los puntos que estén duplicados
%-------------------------------------------------------------

%Contabilizar la cantidad de puntos que están duplicados en la matriz de coordenadas "PopPointsCo".
NRepPopPoints=0;
```

```
for i=1:((length(PopPoints)/3)-1)
    if(PopPointsCo(i,1)==PopPointsCo((i+1),1))&&(PopPointsCo(i,2)==PopPoi
    ntsCo((i+1),2)) %Se evalúa si las coordenadas X (PopPointsCo(i,1)) y
    Y (PopPointsCo(i,2)) del punto (i) son iguales a las coordenadas X
    (PopPointsCo((i+1),1))) y Y (PopPointsCo((i+1),2))) del punto (i+1).
    Debido a que los puntos ya están ordenados, se asegura que la
    evaluación de duplicidad sea correcta.
      NRepPopPoints=NRepPopPoints+1;
    end
end

for  i=1:((length(PopPoints)/3)-1)  %El (-1) se debe a que en el
condicional se utiliza (i+1), luego el ciclo no puede ir hasta el último
(i), sino hasta el penúltimo.
    if(PopPointsCo(i,1)==PopPointsCo((i+1),1))&&(PopPointsCo(i,2)==PopPoi
    ntsCo((i+1),2)) %El mismo condicional del for anterior.
      %Se asigna un valor igual a 0 a la coordenada X, Y y Z del punto
      (i) que esté duplicado (que sea igual al siguiente). Luego se
      suprimirán los puntos que tengan coordenadas (0,0,0).
      PopPointsCo(i,1)=0;
      PopPointsCo(i,2)=0;
      PopPointsCo(i,3)=0;
    end
end

%Ordenar la matriz de las coordenadas de todos los puntos de la retícula,
en base a la columna 3 (la coordenada Z). Esto para que los primeros
puntos de la matriz sean los que están repetidos, es decir, los que
tienen coordenadas (0,0,0).
PopPointsCo=sortrows(PopPointsCo,3);

%Crear la matriz que almacenará las coordenadas de todos los puntos de la
retícula, excluyendo los puntos duplicados. Las columnas se llenan con
los vectores de coordenadas X, Y y Z de la matriz "PopPointsCo", pero se
empieza con el elemento (NRepPopPoints+i) de dicha matriz. De esta manera
se excluyen los elementos duplicados (coordenadas 0,0,0) que se
```

```
encuentran ubicados desde la posición 1 hasta la posición (NRepPopPoints)
en la matriz "PopPointsCo".
PopPointsFilt=zeros(((length(PopPoints)/3)-NRepPopPoints),3);

for  i=1:((length(PopPoints)/3)-NRepPopPoints) %El ciclo para llenar la
matriz "PopPointsFilt" va de 1 hasta la longitud de la matriz
(PopPointsCo) que contiene todos los puntos de la retícula (incluyendo
duplicados) menos la cantidad de puntos duplicados (NRepPopPoints).
    PopPointsFilt(i,1)=PopPointsCo((NRepPopPoints+i),1);
    PopPointsFilt(i,2)=PopPointsCo((NRepPopPoints+i),2);
    PopPointsFilt(i,3)=PopPointsCo((NRepPopPoints+i),3);
end

%Imprimir un archivo .txt con la matriz de las coordenadas de todos los
puntos de la retícula, después de haber filtrado los puntos duplicados.
Este archivo se leerá en Grasshopper para ordenar de nuevo dichos puntos
(corresponde al paso 8 de la definición en Grasshopper).
dlmwrite('OrderedGridPointsFilt.txt',PopPointsFilt);

%-------------------------------------------------------------
%Leer los puntos anteriormente filtrados, y reordenados en Grasshopper, y
crear %una matriz con los puntos definitivos para la retícula
%-------------------------------------------------------------

%Leer un archivo .txt con las coordenadas (como una lista de números, no
como grupos de coordenadas) de los puntos ordenados y filtrados (sin
duplicados) de la grilla o retícula. Este archivo se crea en Grasshopper,
ordenando los puntos que se leen del archivo "OrderedGridPointsFilt.txt"
(corresponde al paso 9 de la definición en Grasshopper).
PopPoints=load('OrderedGridPointsFilt_1.txt');

%Crear los vectores que contendrán las coordenadas X, Y y Z de los puntos
ordenados y filtrados (sin duplicados) de la retícula, con los datos de
la lista del archivo "OrderedGridPointsFilt_1.txt". En el for, el 3i-2,
3i-1 y 3i es porque debe tomar los índices correspondientes a cada
```

```
coordenada, teniendo en cuenta que el archivo .txt es una lista de
números, no unos grupos de coordenadas.
PopPointsX=zeros((length(PopPoints)/3),1);
PopPointsY=zeros((length(PopPoints)/3),1);
PopPointsZ=zeros((length(PopPoints)/3),1);

for i=1:(length(PopPoints)/3)
    PopPointsX(i)=PopPoints((3*i)-2);
    PopPointsY(i)=PopPoints((3*i)-1);
    PopPointsZ(i)=PopPoints(3*i);
end

%Crear la matriz que almacenará las coordenadas de los puntos ordenados y
filtrados (sin duplicados) de la retícula. Las columnas se llenan con los
vectores de coordenadas X, Y y Z de los puntos ordenados y filtrados (sin
duplicados) de la retícula (PopPointsX, PopPointsY, PopPointsZ).
PopPointsCo=zeros((length(PopPoints)/3),3);

for i=1:(length(PopPoints)/3)
    PopPointsCo(i,1)=round(PopPointsX(i)*100)/100;
    PopPointsCo(i,2)=round(PopPointsY(i)*100)/100;
    PopPointsCo(i,3)=round(PopPointsZ(i)*100)/100;
end
```

PARTE 3 (correspondiente a la definición en Grasshopper):

```
%-------------------------------------------------------------------------
%CONECTIVIDAD
%-------------------------------------------------------------------------

%Leer los archivos .txt "DelaunayConn.txt" y "NumDelaunayConn.txt", los
cuales contienen la información necesaria de la conectividad de Delaunay,
para que en MatLab se pueda obtener la información acerca de qué puntos
están conectados con cada punto de la retícula (corresponde al paso 10 de
la definición en Grasshopper).
```

```
DelaunayConn=load('DelaunayConn.txt'); %Hace referencia al índice de los puntos que están conectados con el punto i de la retícula.
NumDelaunayConn=load('NumDelaunayConn.txt'); %Hace referencia a la cantidad de puntos que están conectados con el punto i de la retícula.

%Crear la matriz que almacenará la información de conectividad de Delaunay. Es decir, una matriz en donde las dos columnas corresponden al índice de cada par de puntos conectados dentro de la retícula.
PointsConn=zeros((length(DelaunayConn)),2);

for i=1:length(DelaunayConn)
    PointsConn(i,1)=DelaunayConn(i)+1; %El +1 es necesario ya que los índices
    en Grasshopper (de donde se leyó el archivo de conectividad) empiezan en 0,
    mientras que en MatLab, empiezan en 1.
end

%Asignar el punto al cual está conectado cada punto definido en la columna 1 (en el for anterior). Para esto, se tiene en cuenta la cantidad de puntos conectados a cada punto; por este motivo se hace necesario el uso de un contador y el for anidado.
cont=0;

for i=1:length(NumDelaunayConn)
    for j=1:NumDelaunayConn(i)
        PointsConn((cont+j),2)=i;
    end
    cont=cont+NumDelaunayConn(i);
end

%Eliminar las parejas de puntos conectados que estén repetidas. Esto es necesario ya que en el resultado del paso anterior, por ejemplo, surge la pareja de puntos 3 – 5, y su equivalente 5 – 3.
[PC1,PC2]=size(PointsConn); %Se evalúa el tamaño de la matriz que contiene la información de la conectividad de Delaunay.
```

```
%Evaluar las parejas de puntos conectados que son equivalentes, y
asignarle a una de ellas un valor de -1, el cual será utilizado para
eliminar, posteriormente, todas las parejas que tengan este valor.

for i=1:PC1
    PointsConnComparison=[PointsConn(i,2),PointsConn(i,1)];
    for j=1:PC1
        if
(PointsConn(j,1)==PointsConnComparison(1,1))&&(PointsConn(j,2)==
            PointsConnComparison(1,2))
            if (i~=j)
                PointsConn(j,:)=-1;
            end
        end
    end
end

%Obtener el índice (SupPointsConn) de la pareja de puntos a la que se le
haya asignado un valor de -1, y la cantidad (NumSupPointsConn) de parejas
con esta condición.
SupPointsConn=zeros(PC1/2,1);
NumSupPointsConn=0;

for i=1:PC1
    if PointsConn(i,:)==-1
        SupPointsConn(i)=i;
        NumSupPointsConn=NumSupPointsConn+1;
    else

        SupPointsConn(i)=0;
    end
end
```

```
%Eliminar las parejas equivalentes, utilizando el índice de éstas
obtenido anteriormente (SupPointsConn).
SupPointsConn=sort(SupPointsConn);

for i=1:NumSupPointsConn
    SupPointsConn(i)=SupPointsConn(i+NumSupPointsConn);
end

PointsConn(SupPointsConn,:)=[];
```

ANEXO C: Código en MatLab para el análisis estructural

```
%-------------------------------------------------------------------------
%DATOS DE ENTRADA FEM
%-------------------------------------------------------------------------

%Definición del módulo de elasticidad (E), área de la sección transversal de las barras (Ae), densidad del acero (ro), y la magnitud de la carga que se va a aplicar en cada nodo de la estructura reticular (PunctualForce).

E=200000; %[MPa]
Ae=127; %[mm2]
ro=7800; %[kg/m3]
PunctualForce=-1000; %[N]

%-------------------------------------------------------------------------
%EVALUACIÓN DE LA FUNCIÓN OBJETIVO 1 Y 2 (ENERGÍA DE DEFORMACIÓN Y PESO)
%-------------------------------------------------------------------------

%Iniciar los vectores que contendrán el valor de las funciones objetivo 1 y 2 (fo1 y fo2), y el vector que almacenará los desplazamientos calculados (Displacements).
fo1=zeros(numpar,1);
fo2=zeros(numpar,1);
Displacements=zeros(numpar,((PP1-(length(BasePoints)/3))*3));

%Obtener los índices de los puntos que corresponden a los puntos de apoyo (esta información se necesitará para las condiciones de borde, y la generación de las alturas de los demás puntos). En el for se evalúa cuáles de los puntos de la matriz PopPointsCo tienen las mismas coordenadas X y Y que los puntos de apoyo (BasePointsCo).
BPInd=zeros((length(BasePoints)/3),1);
[PP1,PP2]=size(PopPointsCo);
```

```
for q=1:(length(BasePoints)/3)
    for i=1:PP1
        if PopPointsCo(i,1)==BasePointsCo(q,1)&&PopPointsCo(i,2)==
           BasePointsCo(q,2)
             BPInd(q)=i;
        end
    end
end

%Obtener los índices de los puntos que no corresponden a los puntos de
apoyo. Para esto, simplemente se toman los índices de todos los puntos y
se eliminan los que sí corresponden a los puntos de apoyo (BPInd).
PopZInd=zeros(PP1,1);

for i=1:PP1
    PopZInd(i,1)=i;
end

PopZInd(BPInd,:)=[];

%-------------------------------------------------------------------------
%Desarrollo del Método de Elementos Finitos (FEM)
%-------------------------------------------------------------------------

%Crear la tabla de coordenadas con los datos de la matriz PopPointsCo
(ver anexo B). La coordenada Z de los puntos de borde es la misma que se
leyó de Grasshopper, la cual está almacenada en el vector BasePointsZ
(ver anexo B). La coordenada Z de los demás puntos de la retícula
corresponde a la población del algoritmo ACCMOUPSO (xin(k,i)), donde k
hace referencia al número de la partícula que se está evaluando dentro
del algoritmo), ya que éstos valores son los que se van a modificar
durante el proceso de optimización.
CooTabPopPoints=zeros(PP1,4);

for i=1:PP1
```

```
    CooTabPopPoints(i,1)=i; %Esta primera columna es solo para numerar
    los puntos.
    CooTabPopPoints(i,2)=PopPointsCo(i,1);
    CooTabPopPoints(i,3)=PopPointsCo(i,2);
end

for i=1:(length(BasePoints)/3)
    CooTabPopPoints(BPInd(i),4)=BasePointsZ(i);
end

for i=1:(PP1-(length(BasePoints)/3))
    CooTabPopPoints(PopZInd(i),4)=xin(k,i);
end

%Crear la tabla de conectividad con los datos de la matriz PointsConn
(ver anexo B). La primera columna de esta tabla será la numeración de las
parejas de puntos, y las siguientes dos columnas serán los índices de los
puntos que están conectados.
[PCL1,PCL2]=size(PointsConn);
ConnTab=zeros(PCL1,3);

for i=1:PCL1
    ConnTab(i,1)=i;
    ConnTab(i,2)=PointsConn(i,1);
    ConnTab(i,3)=PointsConn(i,2);
end

%Crear una tabla en donde se almacenará la longitud de cada una de las
barras que conectarán los puntos de la retícula, así como los cosenos
directores de éstas. La primera columna será la numeración de las barras,
la segunda será la longitud, y la tercera, cuarta y quinta serán los
cosenos directores en X, Y y Z, respectivamente.
LenDirCosTab=zeros(PCL1,5);

for i=1:PCL1
```

```
    LenDirCosTab(i,1)=i;

    LenDirCosTab(i,2)=sqrt((CooTabPopPoints(ConnTab(i,3),2)-
    CooTabPopPoints(ConnTab(i,2),2))^2+(CooTabPopPoints(ConnTab(i,3),3)-
    CooTabPopPoints(ConnTab(i,2),3))^2+(CooTabPopPoints(ConnTab(i,3),4)-
    CooTabPopPoints(ConnTab(i,2),4))^2);

    LenDirCosTab(i,3)=(CooTabPopPoints(ConnTab(i,3),2)-
    CooTabPopPoints(ConnTab(i,2),2))/LenDirCosTab(i,2);

    LenDirCosTab(i,4)=(CooTabPopPoints(ConnTab(i,3),3)-
    CooTabPopPoints(ConnTab(i,2),3))/LenDirCosTab(i,2);

    LenDirCosTab(i,5)=(CooTabPopPoints(ConnTab(i,3),4)-
    CooTabPopPoints(ConnTab(i,2),4))/LenDirCosTab(i,2);

end

%Crear la matriz de rigidez global de la estructura reticular. El primer
paso es calcular los componentes de la matriz de rigidez de cada barra.
Para esto, se tiene la siguiente definición:
```

$$k = \frac{E_e A_e}{l_e} * \begin{array}{c|cccccc|cc} & a & b & c & d & e & f & & \\ & 3i-2 & 3i-1 & 3i & 3j-2 & 3j-1 & 3j & DOF & \\ \hline & l^2 & lm & ln & -l^2 & -lm & -ln & 3i-2 & a \\ & lm & m^2 & mn & -lm & -m^2 & -mn & 3i-1 & b \\ & ln & mn & n^2 & -ln & -mn & -n^2 & 3i & c \\ & -l^2 & -lm & -ln & l^2 & lm & ln & 3j-2 & d \\ & -lm & -m^2 & -mn & lm & m^2 & mn & 3j-1 & e \\ & -ln & -mn & -n^2 & ln & mn & n^2 & 3j & f \end{array}$$

```
%donde Ee es el módulo de elasticidad del material de la barra; Ae es el
área transversal de la barra; le es la longitud de la barra; l, m, n son
los cosenos directores de la barra; DOF es el grado de libertad al que
corresponde cada elemento, siendo i y j los índices de los puntos que
conecta la barra; y a, b, c, d, e, f son simplemente referencias que
serán utilizadas para referirse a los componentes de la matriz, por
ejemplo, el componente aa corresponde al término l2, y así sucesivamente.
[CTPP1,CTPP2]=size(CooTabPopPoints); %Calcular las dimensiones de la
tabla de coordenadas.
StMat=zeros((CTPP1*3),(CTPP1*3)); %Inicializar la matriz de rigidez.
LenDirCosPlusConn=[LenDirCosTab ConnTab(:,2) ConnTab(:,3)]; %Crear una
tabla que combine la tabla en donde se encuentra la longitud y los
cosenos directores de las barras, con la tabla de conectividad.

%Calcular los componentes de la matriz de rigidez (como se explicó en
párrafos anteriores), barra por barra, utilizando la información de la
tabla creada anteriormente (longitud, cosenos directores, y
conectividad).
for i=1:PCL1

    aa=((LenDirCosPlusConn(i,3))^2)*(Ae*E/(LenDirCosPlusConn(i,2)*1000));
    ba=(LenDirCosPlusConn(i,3))*(LenDirCosPlusConn(i,4))*(Ae*E/(LenDirCos
    PlusC
        onn(i,2)*1000));
    ca=(LenDirCosPlusConn(i,3))*(LenDirCosPlusConn(i,5))*(Ae*E/(LenDirCos
    PlusC
        onn(i,2)*1000));
    da=-(aa);
    ea=-(ba);
    fa=-(ca);

    ab=ba;
    bb=((LenDirCosPlusConn(i,4))^2)*(Ae*E/(LenDirCosPlusConn(i,2)*1000));
    cb=(LenDirCosPlusConn(i,4))*(LenDirCosPlusConn(i,5))*(Ae*E/(LenDirCos
    PlusC
```

```
    onn(i,2)*1000));
db=-(ab);
eb=-(bb);
fb=-(cb);

ac=ca;
bc=cb;
cc=((LenDirCosPlusConn(i,5))^2)*(Ae*E/(LenDirCosPlusConn(i,2)*1000));
dc=-(ac);
ec=-(bc);
fc=-(cc);


ad=da;
bd=ea;
cd=fa;
dd=aa;
ed=ba;
fd=ca;

ae=db;
be=eb;
ce=fb;
de=ab;
ee=bb;
fe=cb;

af=dc;
bf=ec;
cf=fc;
df=ac;
ef=bc;
ff=cc;
```

```
%Ubicar los componentes calculados en la matriz de rigidez global,
teniendo en cuenta el grado de libertad (DOF) al que corresponden.
Los componentes de las matrices de rigidez de las barras que tengan
la misma ubicación dentro de la matriz de rigidez global, se sumarán;
por este motivo, al ubicar un componente, éste no reemplaza al valor
pre-existente de la matriz de rigidez global, si no que se suma a él.
StMat((3*(LenDirCosPlusConn(i,6))-2),(3*(LenDirCosPlusConn(i,6))-
2))=StMat((3*(LenDirCosPlusConn(i,6))-2),(3*(LenDirCosPlusConn(i,6))-
2))+ aa;

StMat((3*(LenDirCosPlusConn(i,6))-1),(3*(LenDirCosPlusConn(i,6))-
2))=StMat((3*(LenDirCosPlusConn(i,6))-1),(3*(LenDirCosPlusConn(i,6))-
2))+ba;

StMat(3*(LenDirCosPlusConn(i,6)),(3*(LenDirCosPlusConn(i,6))-
2))=StMat(3*(LenDirCosPlusConn(i,6)),(3*(LenDirCosPlusConn(i,6))-
2))+ca;

StMat((3*(LenDirCosPlusConn(i,7))-2),(3*(LenDirCosPlusConn(i,6))-
2))=StMat((3*(LenDirCosPlusConn(i,7))-2),(3*(LenDirCosPlusConn(i,6))-
2))+da;

StMat((3*(LenDirCosPlusConn(i,7))-1),(3*(LenDirCosPlusConn(i,6))-
2))=StMat((3*(LenDirCosPlusConn(i,7))-1),(3*(LenDirCosPlusConn(i,6))-
2))+ea;

StMat(3*(LenDirCosPlusConn(i,7)),(3*(LenDirCosPlusConn(i,6))-
2))=StMat(3*(LenDirCosPlusConn(i,7)),(3*(LenDirCosPlusConn(i,6))-
2))+fa;



StMat((3*(LenDirCosPlusConn(i,6))-2),(3*(LenDirCosPlusConn(i,6))-
1))=StMat((3*(LenDirCosPlusConn(i,6))-2),(3*(LenDirCosPlusConn(i,6))-
1))+ab;
```

```
StMat((3*(LenDirCosPlusConn(i,6))-1),(3*(LenDirCosPlusConn(i,6))-
1))=StMat((3*(LenDirCosPlusConn(i,6))-1),(3*(LenDirCosPlusConn(i,6))-
1))+bb;

StMat(3*(LenDirCosPlusConn(i,6)),(3*(LenDirCosPlusConn(i,6))-
1))=StMat(3*(LenDirCosPlusConn(i,6)),(3*(LenDirCosPlusConn(i,6))-
1))+cb;

StMat((3*(LenDirCosPlusConn(i,7))-2),(3*(LenDirCosPlusConn(i,6))-
1))=StMat((3*(LenDirCosPlusConn(i,7))-2),(3*(LenDirCosPlusConn(i,6))-
1))+db;

StMat((3*(LenDirCosPlusConn(i,7))-1),(3*(LenDirCosPlusConn(i,6))-
1))=StMat((3*(LenDirCosPlusConn(i,7))-1),(3*(LenDirCosPlusConn(i,6))-
1))+eb;

StMat(3*(LenDirCosPlusConn(i,7)),(3*(LenDirCosPlusConn(i,6))-
1))=StMat(3*(LenDirCosPlusConn(i,7)),(3*(LenDirCosPlusConn(i,6))-
1))+fb;



StMat((3*(LenDirCosPlusConn(i,6))-2),3*(LenDirCosPlusConn(i,6)))=
StMat((3*(LenDirCosPlusConn(i,6))-2),3*(LenDirCosPlusConn(i,6)))+ac;

StMat((3*(LenDirCosPlusConn(i,6))-1),3*(LenDirCosPlusConn(i,6)))=
StMat((3*(LenDirCosPlusConn(i,6))-1),3*(LenDirCosPlusConn(i,6)))+bc;

StMat(3*(LenDirCosPlusConn(i,6)),3*(LenDirCosPlusConn(i,6)))=StMat(3*
(LenDirCosPlusConn(i,6)),3*(LenDirCosPlusConn(i,6)))+cc;

StMat((3*(LenDirCosPlusConn(i,7))-2),3*(LenDirCosPlusConn(i,6)))=
StMat((3*(LenDirCosPlusConn(i,7))-2),3*(LenDirCosPlusConn(i,6)))+dc;
```

```
StMat((3*(LenDirCosPlusConn(i,7))-1),3*(LenDirCosPlusConn(i,6)))=
StMat((3*(LenDirCosPlusConn(i,7))-1),3*(LenDirCosPlusConn(i,6)))+ec;

StMat(3*(LenDirCosPlusConn(i,7)),3*(LenDirCosPlusConn(i,6)))=StMat(3*
(LenDirCosPlusConn(i,7)),3*(LenDirCosPlusConn(i,6)))+fc;



StMat((3*(LenDirCosPlusConn(i,6))-2),(3*(LenDirCosPlusConn(i,7))-
2))=StMat((3*(LenDirCosPlusConn(i,6))-2),(3*(LenDirCosPlusConn(i,7))-
2))+ad;

StMat((3*(LenDirCosPlusConn(i,6))-1),(3*(LenDirCosPlusConn(i,7))-
2))=StMat((3*(LenDirCosPlusConn(i,6))-1),(3*(LenDirCosPlusConn(i,7))-
2))+bd;

StMat(3*(LenDirCosPlusConn(i,6)),(3*(LenDirCosPlusConn(i,7))-
2))=StMat(3*(LenDirCosPlusConn(i,6)),(3*(LenDirCosPlusConn(i,7))-
2))+cd;

StMat((3*(LenDirCosPlusConn(i,7))-2),(3*(LenDirCosPlusConn(i,7))-
2))=StMat((3*(LenDirCosPlusConn(i,7))-2),(3*(LenDirCosPlusConn(i,7))-
2))+dd;

StMat((3*(LenDirCosPlusConn(i,7))-1),(3*(LenDirCosPlusConn(i,7))-
2))=StMat((3*(LenDirCosPlusConn(i,7))-1),(3*(LenDirCosPlusConn(i,7))-
2))+ed;

StMat(3*(LenDirCosPlusConn(i,7)),(3*(LenDirCosPlusConn(i,7))-
2))=StMat(3*(LenDirCosPlusConn(i,7)),(3*(LenDirCosPlusConn(i,7))-
2))+fd;
```

```
StMat((3*(LenDirCosPlusConn(i,6))-2),(3*(LenDirCosPlusConn(i,7))-
1))=StMat((3*(LenDirCosPlusConn(i,6))-2),(3*(LenDirCosPlusConn(i,7))-
1))+ae;

StMat((3*(LenDirCosPlusConn(i,6))-1),(3*(LenDirCosPlusConn(i,7))-
1))=StMat((3*(LenDirCosPlusConn(i,6))-1),(3*(LenDirCosPlusConn(i,7))-
1))+be;

StMat(3*(LenDirCosPlusConn(i,6)),(3*(LenDirCosPlusConn(i,7))-
1))=StMat(3*(LenDirCosPlusConn(i,6)),(3*(LenDirCosPlusConn(i,7))-
1))+ce;

StMat((3*(LenDirCosPlusConn(i,7))-2),(3*(LenDirCosPlusConn(i,7))-
1))=StMat((3*(LenDirCosPlusConn(i,7))-2),(3*(LenDirCosPlusConn(i,7))-
1))+de;

StMat((3*(LenDirCosPlusConn(i,7))-1),(3*(LenDirCosPlusConn(i,7))-
1))=StMat((3*(LenDirCosPlusConn(i,7))-1),(3*(LenDirCosPlusConn(i,7))-
1))+ee;

StMat(3*(LenDirCosPlusConn(i,7)),(3*(LenDirCosPlusConn(i,7))-
1))=StMat(3*(LenDirCosPlusConn(i,7)),(3*(LenDirCosPlusConn(i,7))-
1))+fe;



StMat((3*(LenDirCosPlusConn(i,6))-2),3*(LenDirCosPlusConn(i,7)))=
StMat((3*(LenDirCosPlusConn(i,6))-2),3*(LenDirCosPlusConn(i,7)))+af;

StMat((3*(LenDirCosPlusConn(i,6))-1),3*(LenDirCosPlusConn(i,7)))=
StMat((3*(LenDirCosPlusConn(i,6))-1),3*(LenDirCosPlusConn(i,7)))+bf;

StMat(3*(LenDirCosPlusConn(i,6)),3*(LenDirCosPlusConn(i,7)))=StMat(3*
(LenDirCosPlusConn(i,6)),3*(LenDirCosPlusConn(i,7)))+cf;
```

```
    StMat((3*(LenDirCosPlusConn(i,7))-2),3*(LenDirCosPlusConn(i,7)))=
    StMat((3*(LenDirCosPlusConn(i,7))-2),3*(LenDirCosPlusConn(i,7)))+df;

    StMat((3*(LenDirCosPlusConn(i,7))-1),3*(LenDirCosPlusConn(i,7)))=
    StMat((3*(LenDirCosPlusConn(i,7))-1),3*(LenDirCosPlusConn(i,7)))+ef;

    StMat(3*(LenDirCosPlusConn(i,7)),3*(LenDirCosPlusConn(i,7)))=StMat(3*
    (LenDirCosPlusConn(i,7)),3*(LenDirCosPlusConn(i,7)))+ff;

end

%Definir las fuerzas nodales aplicadas. Es este caso, las fuerzas están
aplicadas en el grado de libertad Z de los nodos (por este motivo se
ubica la fuerza puntual en la ubicación 3i del vector de fuerzas, y el
resto se dejan en cero).
Forces=zeros((CTPP1*3),1);

for i=1:((CTPP1*3)/3)
    Forces(3*i)=PunctualForce;
end

%Definir condiciones de borde. En el primer for se evalúa cuáles puntos
de la tabla de coordenadas (CooTabPopPoints) coinciden con los puntos de
apoyo; en el vector BordCond se almacenarán los índices de los que así lo
hagan. Posteriormente, en el siguiente for, se crean los índices de las
filas y columnas que deben ser eliminadas por ser puntos de apoyo (o
condiciones de borde), basándose en el vector BordCond. Finalmente, se
procede a eliminar dichas filas y columnas de la matriz de rigidez
global, y del vector de fuerzas.
BorCond=zeros((length(BasePoints)/3),1);
for q=1:(length(BasePoints)/3)
    for l=1:PP1
        if BasePointsCo(q,1)==CooTabPopPoints(l,2)&&BasePointsCo(q,2)==
           CooTabPopPoints(l,3)
             BorCond(q)=CooTabPopPoints(l,1);
```

```
        end
    end
end

Sup1=zeros(length(BorCond),1);
Sup2=zeros(length(BorCond),1);
Sup3=zeros(length(BorCond),1);

for q=1:length(BorCond)
    Sup1(q)=(3*BorCond(q))-2;
    Sup2(q)=(3*BorCond(q))-1;
    Sup3(q)=3*BorCond(q);
end

Sup=[Sup1;Sup2;Sup3];

StMat(Sup,:)=[];
StMat(:,Sup)=[];
Forces(Sup)=[];

%Calcular los desplazamientos. Una vez definida la matriz de rigidez
global y el vector de fuerzas, se resuelve el sistema de ecuaciones
({Q}=[K]^-1{F}) para obtener los desplazamientos (el índice k en el vector
de desplazamientos, hace referencia al número de la partícula del
algoritmo de optimización).
Displacements(k,:)=linsolve(StMat,Forces);

%-------------------------------------------------------------------------
%Evaluación de la función objetivo 1: Energía de Deformación
%-------------------------------------------------------------------------

%Para obtener la energía de deformación, se multiplican las fuerzas (el
resultado de la multiplicación del vector de desplazamientos por la
matriz de rigidez) por los desplazamientos (el índice k del vector que
```

```
almacena los valores de la evaluación de la función objetivo 1, fo1, hace
referencia al número de la partícula del algoritmo de optimización).
fo1(k,1)=abs(Displacements(k,:)*StMat)*abs(Displacements(k,:)');

%--------------------------------------------------------------------------
%Evaluación de la función objetivo 2: Peso
%--------------------------------------------------------------------------

%Para obtener el peso total de la estructura reticular, se hace una
sumatoria en la que se van acumulando los pesos de cada barra. Estos son
calculados multiplicando la longitud de cada elemento (obtenida de la
tabla de longitud y cosenos directores, LenDirCosTab) por la densidad del
material utilizado (ro), y por el área de la sección transversal.
Posteriormente, la sumatoria obtenida se multiplica por la aceleración de
la gravedad para obtener el peso, el cual es asignado al vector que
almacena los valores de la evaluación de la función objetivo 2, fo2 (el
índice k hace referencia al número de la partícula del algoritmo de
optimización).
Weight=0;

for i=1:PCL1
    Weight=Weight+LenDirCosTab(i,2)*ro*(Ae/(1000^2));
end

fo2(k,1)=Weight*9.81;
```

ANEXO D: Código en MatLab para el análisis acústico

```
%-------------------------------------------------------------------------
%Datos de entrada para el acercamiento Ray-Tracing
%-------------------------------------------------------------------------

%El nivel de la potencia de sonido (Sound Power Level – SPL) emitido por la fuente, en dB:
Lw=90;
%Coordenadas de la ubicación de la fuente, en metros:
CooFuente=[1.12 1.88 0.6];
%Coeficiente de atenuación sonora del aire, el cual depende de la frecuencia del sonido, de la temperatura, y de la humedad (ver la norma ISO 9613-1), en dB/km:
CoefAtenAire=2.77;
%Coeficiente de atenuación sonora del material de las superficies, el cual depende de la frecuencia del sonido (este valor se obtuvo del trabajo de Monks (132)); para este caso, el material utilizado fue paneles de plywood, para una frecuencia de 500 hz:
CoefAtenMaterial=0.17;
%Porcentaje de discontinuidad de energía (Energy Discontinuity Percentage - EDP). Es el porcentaje de disminución de la energía que determina cuándo parar el seguimiento de la trayectoria de los rayos, cuando la energía inicial del rayo se haya reducido en un valor igual al EDP (%), el proceso se detiene:
EDP=90;

%-------------------------------------------------------------------------
%Generación de los receptores
%-------------------------------------------------------------------------

%Crear las coordenadas de los receptores. MatLab lee un archivo .txt con las coordenadas de los puntos de los receptores, creado en Grasshopper como una lista de números, no como grupos de coordenadas (corresponde a la sección de “Análisis Acústico” en la definición de Grasshopper).
ReceiversPoints=load('ReceiversPoints.txt');
```

```
%Crear los vectores que contendrán las coordenadas X, Y, y Z de los
receptores. En el for, el 3i-2, 3i-1 y 3i es porque debe tomar los
índices correspondientes a cada coordenada, teniendo en cuenta que el
archivo .txt es una lista de números, no unos grupos de coordenadas. Para
definir la coordenada Z, se tomó una altura de 0.60m (la cual
corresponde, aproximadamente, a la altura media de una persona sentada)
para todos los receptores (ver el siguiente for).
ReceiversPointsX=zeros((length(ReceiversPoints)/3),1);
ReceiversPointsY=zeros((length(ReceiversPoints)/3),1);
ReceiversPointsZ=zeros((length(ReceiversPoints)/3),1);

for i=1:(length(ReceiversPoints)/3)
    ReceiversPointsX(i)=ReceiversPoints((3*i)-2);
    ReceiversPointsY(i)=ReceiversPoints((3*i)-1);
    ReceiversPointsZ(i)=ReceiversPoints(3*i)+0.6;
end

%Crear la matriz que almacenará las coordenadas de los puntos de los
receptores. Esta matriz se llena con los vectores creados en el paso
anterior (ReceiversPointsX, ReceiversPointsY y ReceiversPointsZ). En el
for utilizado para crear esta matriz, también se generan las ecuaciones
de las esferas que se utilizan para modelar a los receptores
(ReceiversEc), teniendo como base la información de las coordenadas de
los puntos de los receptores (que corresponden a los centros de las
esferas).
ReceiversPointsCo=zeros((length(ReceiversPoints)/3),3);
syms x y z;
ReceiversEc=sym(zeros(length(ReceiversPointsCo),1));

for i=1:(length(ReceiversPoints)/3)

    ReceiversPointsCo(i,1)=ReceiversPointsX(i);
    ReceiversPointsCo(i,2)=ReceiversPointsY(i);
    ReceiversPointsCo(i,3)=ReceiversPointsZ(i);
```

```
    ReceiversEc(i,1)=(x-ReceiversPointsCo(i,1))^2+(y-
    ReceiversPointsCo(i,2))^2+(z-ReceiversPointsCo(i,3))^2-0.3^2;
end

%-------------------------------------------------------------------------
%Número y generación de rayos
%-------------------------------------------------------------------------

%Obtener el área de la planta que cubre la estructura reticular de
cubierta (MatLab lee un archivo .txt que contiene el área de la planta;
este paso corresponde a la sección de “Análisis Acústico” en la
definición de Grasshopper). Este valor es utilizado para calcular el
volumen del espacio contenido por la cubierta, el plano de piso, y los
planos verticales (muros).
Area=load('Area.txt');

%Calcular aproximadamente el volumen del espacio interior. Para esto se
utiliza, además del área de la planta, una altura media (debido a que
todos los nodos tienen un valor distinto como coordenada Z, definido
dentro del rango [MinHeight, MaxHeight], ver Anexo B, Parte 1).
Vspace=((MaxHeight+MinHeight)/2)*Area;

%Calcular el volumen de los receptores (todos son iguales y corresponden
a una esfera de 0.60m de diámetro).
Vreceiver=(4*pi*(0.3^3))/3;

%Calcular el número de rayos, el cual depende del volumen total del
espacio interior (Vspace) y del volumen de los receptores (Vreceiver).
Ver ecuación (33).
Nray=round((10*Vspace)/Vreceiver);

%Crear la matriz que contendrá los cosenos directores de los rayos. El
mismo for utilizado para crear dicha matriz también se utiliza para crear
las ecuaciones de los rayos, RayEc1 y RayEc2 (dos por rayo, ya que la
recta se describe como la intersección de dos planos, y a cada plano le
corresponde una ecuación).
```

```
DirRay=zeros(Nray,3);
RayEc1=sym(zeros(Nray,1));
RayEc2=sym(zeros(Nray,1));

for i=1:Nray

    DirRay(i,1)=rand; %Llena el vector del primer coseno director de
    todos los rayos, con un número aleatorio.
    DirRay(i,2)=sqrt((1-((DirRay(i,1))^2)))*rand; %Llena el vector del
    segundo coseno director de todos los rayos, con un número aleatorio
    multiplicado por lo que haría falta para que la magnitud del vector
    director sea 1.
    DirRay(i,3)=sqrt(1-(DirRay(i,1))^2-(DirRay(i,2))^2); %Llena el vector
    del tercer coseno director de todos los rayos, con lo que haría falta
    para que la magnitud del vector director sea 1.

    RayEc1(i,1)=DirRay(i,2)*x-DirRay(i,1)*y+(DirRay(i,1)*CooFuente(1,2)-
    DirRay(i,2)*CooFuente(1,1));
    RayEc2(i,1)=DirRay(i,3)*x-DirRay(i,1)*z+(DirRay(i,1)*CooFuente(1,3)-
    DirRay(i,3)*CooFuente(1,1));

end

%-------------------------------------------------------------------------
%Generación de los planos de cubierta
%-------------------------------------------------------------------------

%Debido a que en el análisis acústico se necesita la información de la
tabla de conectividad, generada en el análisis estructural (ver Anexo C),
se crea una copia de dicha tabla con el nombre de ConnTabAcust. Lo
anterior para prevenir que se modifique la información correspondiente al
análisis estructural.
ConnTabAcust=ConnTab;

%Crear una matriz que contendrá los trios de puntos que forman cada uno
de los planos triangulares de cubierta. Los dos primeros puntos son los
```

```
mismos de la tabla de conectividad (esta asignación corresponde al primer
for a continuación). El tercer punto se busca de tal manera que esté
conectado tanto con el primer punto como con el segundo, asegurándose que
éste sea el faltante para formar un triángulo determinado (esta
asignación corresponde al segundo for a continuación).
PlanePoints=zeros(PCL1,3);

for i=1:PCL1
    PlanePoints(i,1)=ConnTabAcust(i,2);
    PlanePoints(i,2)=ConnTabAcust(i,3);
end

for i=1:PCL1
    for j=1:PCL1
        if (ConnTabAcust(j,3)==ConnTabAcust(i,3))&&(i~=j)
           for l=1:PCL1
               if
               (ConnTabAcust(l,3)==ConnTabAcust(i,2))&&(ConnTabAcust(l,2)=
               =
                   ConnTabAcust(j,2))
                     PlanePoints(i,3)=ConnTabAcust(l,2);
               end
           end
        end
    end
end

%Eliminar los trios que tengan como tercer número el cero, ya que no
serían planos (existen varias parejas de puntos que ya formaron el
triángulo al que corresponden pero que siguen estando en la tabla de
conectividad).
SupPointsConnAcustTrans=zeros(length(PlanePoints),1);
NumSupPointsConnAcust=0;

for i=1:PCL1
    if PlanePoints(i,3)==0
```

```
            SupPointsConnAcustTrans(i)=i;
            NumSupPointsConnAcust=NumSupPointsConnAcust+1;
        end
    end

    SupPointsConnAcustTrans=sort(SupPointsConnAcustTrans);
    SupPointsConnAcust=zeros(NumSupPointsConnAcust,1);

    for i=1:NumSupPointsConnAcust
        SupPointsConnAcust(i)=SupPointsConnAcustTrans(i+length(PlanePoints)-
        NumSupPointsConnAcust);
    end

    PlanePoints(SupPointsConnAcust,:)=[];

    %Crear un vector con las ecuaciones de los planos de cubierta. La
    ecuación descrita dentro del for, es la correspondiente a un plano cuando
    se conocen las coordenadas de tres puntos contenidos en él.
    PlaneEc=sym(zeros(length(PlanePoints),1));

    for i=1:length(PlanePoints)
        PlaneEc(i,1)=(x-CooTabPopPoints(PlanePoints(i,1),2))*
        (((CooTabPopPoints(PlanePoints(i,2),3)-
        CooTabPopPoints(PlanePoints(i,1),3))*
        (CooTabPopPoints(PlanePoints(i,3),4)-
        CooTabPopPoints(PlanePoints(i,1),4)))-
        ((CooTabPopPoints(PlanePoints(i,2),4)-
        CooTabPopPoints(PlanePoints(i,1),4))*
        (CooTabPopPoints(PlanePoints(i,3),3)-
        CooTabPopPoints(PlanePoints(i,1),3))))-
        (y-CooTabPopPoints(PlanePoints(i,1),3))*
        (((CooTabPopPoints(PlanePoints(i,2),2)-
        CooTabPopPoints(PlanePoints(i,1),2))*
        (CooTabPopPoints(PlanePoints(i,3),4)-
        CooTabPopPoints(PlanePoints(i,1),4)))-
```

```
    ((CooTabPopPoints(PlanePoints(i,2),4)-
    CooTabPopPoints(PlanePoints(i,1),4))*
    (CooTabPopPoints(PlanePoints(i,3),2)-
    CooTabPopPoints(PlanePoints(i,1),2))))+
    (z-CooTabPopPoints(PlanePoints(i,1),4))*
    (((CooTabPopPoints(PlanePoints(i,2),2)-
    CooTabPopPoints(PlanePoints(i,1),2))*
    (CooTabPopPoints(PlanePoints(i,3),3)-
    CooTabPopPoints(PlanePoints(i,1),3)))-
    ((CooTabPopPoints(PlanePoints(i,2),3)-
    CooTabPopPoints(PlanePoints(i,1),3))*
    (CooTabPopPoints(PlanePoints(i,3),2)-
    CooTabPopPoints(PlanePoints(i,1),2))));
end

%Crear una matriz con los vectores normales a los planos (la matriz
tendrá en sus columnas los cosenos directores de los vectores). Los
cosenos directores de estos vectores se obtienen con base en las
ecuaciones de los planos generadas en el paso anterior (cada coseno es el
término que acompaña a la variable correspondiente: X, Y y Z).
PlaneNormalVec=zeros(length(PlanePoints),3);

for i=1:length(PlanePoints)

    PlaneNormalVec(i,1)=(((CooTabPopPoints(PlanePoints(i,2),3)-
    CooTabPopPoints(PlanePoints(i,1),3))*(CooTabPopPoints(PlanePoints(i,3
    ),4)-CooTabPopPoints(PlanePoints(i,1),4)))-
    ((CooTabPopPoints(PlanePoints(i,2),4)-
    CooTabPopPoints(PlanePoints(i,1),4))*(CooTabPopPoints(PlanePoints(i,3
    ),3)-CooTabPopPoints(PlanePoints(i,1),3))));

    PlaneNormalVec(i,2)=-(((CooTabPopPoints(PlanePoints(i,2),2)-
    CooTabPopPoints(PlanePoints(i,1),2))*(CooTabPopPoints(PlanePoints(i,3
    ),4)-CooTabPopPoints(PlanePoints(i,1),4)))-
    ((CooTabPopPoints(PlanePoints(i,2),4)-
```

```
    CooTabPopPoints(PlanePoints(i,1),4))*(CooTabPopPoints(PlanePoints(i,3
    ),2)-CooTabPopPoints(PlanePoints(i,1),2))));

    PlaneNormalVec(i,3)=(((CooTabPopPoints(PlanePoints(i,2),2)-
    CooTabPopPoints(PlanePoints(i,1),2))*(CooTabPopPoints(PlanePoints(i,3
    ),3)-CooTabPopPoints(PlanePoints(i,1),3)))-
    ((CooTabPopPoints(PlanePoints(i,2),3)-
    CooTabPopPoints(PlanePoints(i,1),3))*(CooTabPopPoints(PlanePoints(i,3
    ),2)-CooTabPopPoints(PlanePoints(i,1),2))));

end

%Ordenar los puntos de "PlanePoints" (asignando los puntos ordenados a
una nueva matriz, PlanePointsOrdered) para que las rectas que definen los
triángulos que definen cada uno de los planos de cubierta, queden
ordenadas, con el fin de poder verificar posteriormente si el punto de
intersección que se va a calcular, entre los rayos y los planos, se
encuentra dentro del plano triangular. Esto es necesario hacerlo ya que
al definir los planos de cubierta, éstos quedaron definidos como planos
infinitos, sin estar delimitados por los triángulos correspondientes,
luego, de no hacer esta verificación, seguramente se calcularían puntos
de intersección que en la realidad no coinciden con los planos
triangulares que definen la cubierta.
PlanePointsOrdered=zeros(length(PlanePoints),3);

for i=1:length(PlanePoints)

    if (CooTabPopPoints(PlanePoints(i,1),2)<
        CooTabPopPoints(PlanePoints(i,2),2))&&
       (CooTabPopPoints(PlanePoints(i,1),2)<
        CooTabPopPoints(PlanePoints(i,3),2))

           PlanePointsOrdered(i,1)=PlanePoints(i,1);

    elseif (CooTabPopPoints(PlanePoints(i,2),2)<
            CooTabPopPoints(PlanePoints(i,3),2))
```

```
            PlanePointsOrdered(i,1)=PlanePoints(i,2);

else
        PlanePointsOrdered(i,1)=PlanePoints(i,3);
end

if (CooTabPopPoints(PlanePoints(i,1),3)>
    CooTabPopPoints(PlanePoints(i,2),3))&&
   (CooTabPopPoints(PlanePoints(i,1),3)>
    CooTabPopPoints(PlanePoints(i,3),3))&&
   (PlanePoints(i,1)~=PlanePointsOrdered(i,1))

       PlanePointsOrdered(i,2)=PlanePoints(i,1);

elseif (CooTabPopPoints(PlanePoints(i,2),3)>
        CooTabPopPoints(PlanePoints(i,3),3))&&
       (PlanePoints(i,2)~=PlanePointsOrdered(i,1))

            PlanePointsOrdered(i,2)=PlanePoints(i,2);

else
        PlanePointsOrdered(i,2)=PlanePoints(i,3);
end

if (PlanePoints(i,1)~=PlanePointsOrdered(i,1))&&
   (PlanePoints(i,1)~=PlanePointsOrdered(i,2))

        PlanePointsOrdered(i,3)=PlanePoints(i,1);

elseif (PlanePoints(i,2)~=PlanePointsOrdered(i,1))&&
       (PlanePoints(i,2)~=PlanePointsOrdered(i,2))

            PlanePointsOrdered(i,3)=PlanePoints(i,2);

else
```

```
            PlanePointsOrdered(i,3)=PlanePoints(i,3);
    end

end

%-------------------------------------------------------------------------
%Generación de los planos verticales (muros)
%-------------------------------------------------------------------------

%Hacer un vector con las ecuaciones de los planos verticales (muros).
Para esto, se toman como base los puntos correspondientes a la base de la
estructura, los cuales definen el perímetro de la misma y, por ende, los
muros. El for utilizado para crear este vector va hasta la longitud de
“BasePointsCo” menos uno, ya que el último plano generado se crea con el
primer punto de base y el último (este plano correspondería al que
termina de cerrar el perímetro), y ésta asignación no se podría hacer
dentro de la secuencia lógica del for. Por este motivo se asigna la
ecuación de ése último plano después de haber hecho el for (corresponde a
la asignación o definición que se encuentra justo después de finalizar el
for que se presenta a continuación).
PlaneEcVer=sym(zeros(length(BasePointsCo),1));

for i=1:(length(BasePointsCo)-1)

    PlaneEcVer(i,1)=(x-BasePointsCo(i,1))*((BasePointsCo(i+1,2)-
    BasePointsCo(i,2))*(0-BasePointsCo(i,3))-(BasePointsCo(i+1,3)-
    BasePointsCo(i,3))*(BasePointsCo(i,2)-BasePointsCo(i,2)))-(y-
    BasePointsCo(i,2))*((BasePointsCo(i+1,1)-BasePointsCo(i,1))*(0-
    BasePointsCo(i,3))-(BasePointsCo(i+1,3)-
    BasePointsCo(i,3))*(BasePointsCo(i,1)-BasePointsCo(i,1)))+(z-
    BasePointsCo(i,3))*((BasePointsCo(i+1,1)-
    BasePointsCo(i,1))*(BasePointsCo(i,2)-BasePointsCo(i,2))-
    (BasePointsCo(i+1,2)-BasePointsCo(i,2))*(BasePointsCo(i,1)-
    BasePointsCo(i,1)));

end
```

```
PlaneEcVer(length(BasePointsCo),1)=(x-
BasePointsCo(length(BasePointsCo),1))*
((BasePointsCo(1,2)-BasePointsCo(length(BasePointsCo),2))*
(0-BasePointsCo(length(BasePointsCo),3))-(BasePointsCo(1,3)-
BasePointsCo(length(BasePointsCo),3))*(BasePointsCo(length(BasePointsCo),
2)-BasePointsCo(length(BasePointsCo),2)))-
(y-BasePointsCo(length(BasePointsCo),2))*((BasePointsCo(1,1)-
BasePointsCo(length(BasePointsCo),1))*
(0-BasePointsCo(length(BasePointsCo),3))-(BasePointsCo(1,3)-
BasePointsCo(length(BasePointsCo),3))*(BasePointsCo(length(BasePointsCo),
1)-BasePointsCo(length(BasePointsCo),1)))+
(z-BasePointsCo(length(BasePointsCo),3))*((BasePointsCo(1,1)-
BasePointsCo(length(BasePointsCo),1))*(BasePointsCo(length(BasePointsCo),
2)-BasePointsCo(length(BasePointsCo),2))-(BasePointsCo(1,2)-
BasePointsCo(length(BasePointsCo),2))*(BasePointsCo(length(BasePointsCo),
1)-BasePointsCo(length(BasePointsCo),1)));

%Crear una matriz con los vectores normales a los planos. Los cosenos
directores que generan dichos vectores se obtienen de los coeficientes
que acompañan a cada una de las variables (X, Y y Z) en las ecuaciones de
los planos de muro que se definieron en el paso anterior. Por el mismo
motivo que se expuso en el paso anterior, es necesario definir el vector
normal correspondiente al último plano de muro, por fuera del for que se
presenta a continuación.

PlaneVerNormalVec=zeros(length(PlaneEcVer),3);

for i=1:(length(PlaneEcVer)-1)

    PlaneVerNormalVec(i,1)=((BasePointsCo(i+1,2)-BasePointsCo(i,2))*
    (0-BasePointsCo(i,3))-(BasePointsCo(i+1,3)-
    BasePointsCo(i,3))*(BasePointsCo(i,2)-BasePointsCo(i,2)));

    PlaneVerNormalVec(i,2)=-((BasePointsCo(i+1,1)-BasePointsCo(i,1))*
```

```
    (0-BasePointsCo(i,3))-(BasePointsCo(i+1,3)-
    BasePointsCo(i,3))*(BasePointsCo(i,1)-BasePointsCo(i,1)));

    PlaneVerNormalVec(i,3)=((BasePointsCo(i+1,1)-BasePointsCo(i,1))*
    (BasePointsCo(i,2)-BasePointsCo(i,2))-(BasePointsCo(i+1,2)-
    BasePointsCo(i,2))*(BasePointsCo(i,1)-BasePointsCo(i,1)));

end

PlaneVerNormalVec(length(PlaneEcVer),1)=((BasePointsCo(1,2)-
BasePointsCo(length(BasePointsCo),2))*
(0-BasePointsCo(length(BasePointsCo),3))-(BasePointsCo(1,3)-
BasePointsCo(length(BasePointsCo),3))*(BasePointsCo(length(BasePointsCo),
2)-BasePointsCo(length(BasePointsCo),2)));

PlaneVerNormalVec(length(PlaneEcVer),2)=-((BasePointsCo(1,1)-
BasePointsCo(length(BasePointsCo),1))*
0-BasePointsCo(length(BasePointsCo),3))-(BasePointsCo(1,3)-
BasePointsCo(length(BasePointsCo),3))*(BasePointsCo(length(BasePointsCo),
1)-BasePointsCo(length(BasePointsCo),1)));

PlaneVerNormalVec(length(PlaneEcVer),3)=((BasePointsCo(1,1)-
BasePointsCo(length(BasePointsCo),1))*(BasePointsCo(length(BasePointsCo),
2)-BasePointsCo(length(BasePointsCo),2))-(BasePointsCo(1,2)-
BasePointsCo(length(BasePointsCo),2))*(BasePointsCo(length(BasePointsCo),
1)-BasePointsCo(length(BasePointsCo),1)));

%-------------------------------------------------------------------------
%Generación del plano horizontal de piso
%-------------------------------------------------------------------------

%El plano de piso se define simplemente como el plano X-Y, con Z=0.
PlaneEcHor(1,1)=z;

%Creación del vector normal al plano.
PlaneHorNormalVec=[0 0 1];
```

```
%-------------------------------------------------------------------------
%Calcular los puntos iniciales de intersección (entre los rayos y los
planos), y la energía acumulada en los receptores
%-------------------------------------------------------------------------

%Definir la energía inicial de cada rayo (ver ecuación (34)).
Eo=((10^(Lw/10))/Nray)*10^(-12);

%Iniciar las matrices y los vectores que se van a utilizar en este
proceso. InterPoints es la matriz, o tabla, en donde se almacenará toda
la información del seguimiento de los rayos, DistRecRay es un vector en
donde se almacenará la distancia recorrida por cada rayo, y EnerRec es la
matriz, o tabla, en donde se guardará la energía acumulada en cada uno de
los receptores.
InterPoints=zeros(Nray,15);
DistRecRay=zeros(Nray,1);
EnerRec=zeros(length(ReceiversEc),2);
InterPoints(:,5)=Eo; %La columna 5 de InterPoints es la energía inicial
de cada rayo, la cual es la misma para todos ellos.

%Llevar a cabo el ciclo en donde se calcularán los puntos iniciales de
intersección (entre rayos y planos), y la energía acumulada en los
receptores. El ciclo más grande corresponde a los rayos, y dentro de
este, están anidados los ciclos correspondientes a los planos: uno para
los planos de cubierta, otro para los planos de los muros, y otro para
las esferas que son utilizadas para modelar a los receptores (como el
plano de piso es sólo uno, no es necesario crear otro ciclo anidado
adicional). De esta manera se evalúa rayo por rayo la posible
intersección de éstos con alguno de los planos, con el fin de obtener los
puntos de intersección.
for i=1:Nray

    InterPoints(i,1)=i; %Se crea la primera columna del Cuadro 7,
    correspondiente a la numeración de los rayos.
```

```
%A las columnas 7, 8 y 9 del Cuadro 7 se les asignan los cosenos
directores del vector director de cada rayo.
InterPoints(i,7)=DirRay(i,1);
InterPoints(i,8)=DirRay(i,2);
InterPoints(i,9)=DirRay(i,3);

%A las columnas 13, 14 y 15 del Cuadro 7 se les asignan las
coordenadas de la fuente de sonido (las cuales corresponden al origen
de los rayos para ésta fase inicial).
InterPoints(i,13)=CooFuente(1,1);
InterPoints(i,14)=CooFuente(1,2);
InterPoints(i,15)=CooFuente(1,3);

%Obtener la intersección con los planos de cubierta. El primer if
evalúa si el producto punto entre el vector director del rayo y el
vector normal al plano (a cada plano) de cubierta es distinto de
cero, ya que si es así, existe una intersección (en algún punto del
espacio). De cumplirse ésta condición, se calcula dicho punto de
intersección, despejando las variables X, Y y Z, del sistema de
ecuaciones compuesto por las dos ecuaciones que definen al rayo y por
la ecuación que define al plano de cubierta. Luego de obtener el
punto de intersección se procede a evaluar si dicho punto se
encuentra ubicado dentro del triángulo que define el plano de
cubierta real. Para esto, se utiliza el vector ordenado de los puntos
que definen cada triángulo (PlanePointsOrdered, ver el final de la
sección “Generación de los planos de cubierta”, incluida en este
mismo anexo), y por medio de comparaciones condicionales entre las
coordenadas de dichos puntos y las correspondientes al punto de
intersección calculado anteriormente, se define si la intersección
coincide con el plano triangular de cubierta (esto corresponde a los
if anidados). Vale la pena resaltar que al encontrar el punto de
intersección real y asignarle sus coordenadas a las columnas
correspondientes del Cuadro 7 (InterPoints), se asignan los cosenos
directores del vector normal al plano correspondiente de reflexión, a
las columnas 10, 11 y 12 de la misma tabla, pues corresponden
justamente a la definición del vector normal al plano de reflexión.
```

```
for j=1:length(PlaneEc)
    if
    (DirRay(i,1)*PlaneNormalVec(j,1)+DirRay(i,2)*PlaneNormalVec(j,2)+
        DirRay(i,3)*PlaneNormalVec(j,3))~=0

          [InterPointRayCubX,InterPointRayCubY,InterPointRayCubZ]=
          solve(RayEc1(i,1),RayEc2(i,1),PlaneEc(j,1),x,y,z);
          InterPointRayCubX=double(InterPointRayCubX);
          InterPointRayCubY=double(InterPointRayCubY);
          InterPointRayCubZ=double(InterPointRayCubZ);

          if (InterPointRayCubX>=
              CooTabPopPoints(PlanePointsOrdered(j,1),2))&&
             (InterPointRayCubX<=
              max(CooTabPopPoints(PlanePointsOrdered(j,2),2),
              CooTabPopPoints(PlanePointsOrdered(j,3),2)))

             if (InterPointRayCubY>=
                ((CooTabPopPoints(PlanePointsOrdered(j,3),3)-
                 CooTabPopPoints(PlanePointsOrdered(j,1),3))/
                (CooTabPopPoints(PlanePointsOrdered(j,3),2)-
                 CooTabPopPoints(PlanePointsOrdered(j,1),2)))*
                 InterPointRayCubX+
                 CooTabPopPoints(PlanePointsOrdered(j,1),3)-
                 CooTabPopPoints(PlanePointsOrdered(j,1),2)*
                ((CooTabPopPoints(PlanePointsOrdered(j,3),3)-
                 CooTabPopPoints(PlanePointsOrdered(j,1),3))/
                (CooTabPopPoints(PlanePointsOrdered(j,3),2)-
                 CooTabPopPoints(PlanePointsOrdered(j,1),2))))&&
                (InterPointRayCubY<=
                ((CooTabPopPoints(PlanePointsOrdered(j,2),3)-
                 CooTabPopPoints(PlanePointsOrdered(j,1),3))/
                (CooTabPopPoints(PlanePointsOrdered(j,2),2)-
                 CooTabPopPoints(PlanePointsOrdered(j,1),2)))*
                 InterPointRayCubX+
                 CooTabPopPoints(PlanePointsOrdered(j,1),3)-
```

```
 CooTabPopPoints(PlanePointsOrdered(j,1),2)*
((CooTabPopPoints(PlanePointsOrdered(j,2),3)-
 CooTabPopPoints(PlanePointsOrdered(j,1),3))/
(CooTabPopPoints(PlanePointsOrdered(j,2),2)-
 CooTabPopPoints(PlanePointsOrdered(j,1),2))))

if CooTabPopPoints(PlanePointsOrdered(j,3),2)>
   CooTabPopPoints(PlanePointsOrdered(j,2),2)

   if (InterPointRayCubY<=
      ((CooTabPopPoints(PlanePointsOrdered(j,3),3)-
       CooTabPopPoints(PlanePointsOrdered(j,2),3))/
      (CooTabPopPoints(PlanePointsOrdered(j,3),2)-
       CooTabPopPoints(PlanePointsOrdered(j,2),2)))*
       InterPointRayCubX+
       CooTabPopPoints(PlanePointsOrdered(j,2),3)-
       CooTabPopPoints(PlanePointsOrdered(j,2),2)*
      ((CooTabPopPoints(PlanePointsOrdered(j,3),3)-
       CooTabPopPoints(PlanePointsOrdered(j,2),3))/
      (CooTabPopPoints(PlanePointsOrdered(j,3),2)-
       CooTabPopPoints(PlanePointsOrdered(j,2),2))))

            InterPoints(i,2)=InterPointRayCubX;
            InterPoints(i,3)=InterPointRayCubY;
            InterPoints(i,4)=InterPointRayCubZ;

            InterPoints(i,10)=PlaneNormalVec(j,1);
            InterPoints(i,11)=PlaneNormalVec(j,2);
            InterPoints(i,12)=PlaneNormalVec(j,3);

   end

else

   if (InterPointRayCubY>=
      ((CooTabPopPoints(PlanePointsOrdered(j,3),3)-
```

```
                        CooTabPopPoints(PlanePointsOrdered(j,2),3))/
                       (CooTabPopPoints(PlanePointsOrdered(j,3),2)-
                        CooTabPopPoints(PlanePointsOrdered(j,2),2)))*
                        InterPointRayCubX+
                        CooTabPopPoints(PlanePointsOrdered(j,2),3)-
                        CooTabPopPoints(PlanePointsOrdered(j,2),2)*
                       ((CooTabPopPoints(PlanePointsOrdered(j,3),3)-
                        CooTabPopPoints(PlanePointsOrdered(j,2),3))/
                       (CooTabPopPoints(PlanePointsOrdered(j,3),2)-
                        CooTabPopPoints(PlanePointsOrdered(j,2),2))))

                            InterPoints(i,2)=InterPointRayCubX;
                            InterPoints(i,3)=InterPointRayCubY;
                            InterPoints(i,4)=InterPointRayCubZ;

                            InterPoints(i,10)=PlaneNormalVec(j,1);
                            InterPoints(i,11)=PlaneNormalVec(j,2);
                            InterPoints(i,12)=PlaneNormalVec(j,3);

                    end
                end
            end
        end
    end
end

%Intersección con planos de muros. La forma en cómo se evaluaron las
intersecciones de los rayos con los planos de muros, es prácticamente
la misma que se utilizó para evaluar la intersección entre los rayos
y los planos de cubierta (ver el procedimiento inmediatamente
anterior). La única diferencia se presenta al verificar si el punto
de intersección calculado está incluido en el plano real del muro,
delimitado por las coordenadas de los puntos de base (BasePointsCo);
en este caso, para dicha evaluación, el for debe ir hasta la longitud
de PlaneEcVer (cantidad de planos de muros) menos uno, ya que el
último plano analizado se creó con el primer punto de base y el
```

```
último (este plano correspondería al que termina de cerrar el
perímetro), y éste análisis no se podría hacer dentro de la secuencia
lógica del for. Por este motivo, la evaluación de intersección con
ése último plano se hace después del for (corresponde a los
condicionales que se encuentran justo después de finalizar el for que
se presenta a continuación). Vale la pena resaltar que al encontrar
el punto de intersección real y asignarle sus coordenadas a las
columnas correspondientes del Cuadro 7  (InterPoints), se asignan los
cosenos directores del vector normal al plano correspondiente de
reflexión, a las columnas 10, 11 y 12 de la misma tabla, pues
corresponden justamente a la definición del vector normal al plano de
reflexión.
for j=1:(length(PlaneEcVer)-1)

    if (DirRay(i,1)*PlaneVerNormalVec(j,1)+DirRay(i,2)*
        PlaneVerNormalVec(j,2)+DirRay(i,3)*PlaneVerNormalVec(j,3))~=0

       [InterPointRayMurX,InterPointRayMurY,InterPointRayMurZ]=
       solve(PlaneEcVer(j,1),RayEc1(i,1),RayEc2(i,1),x,y,z);

       InterPointRayMurX=double(InterPointRayMurX);
       InterPointRayMurY=double(InterPointRayMurY);
       InterPointRayMurZ=double(InterPointRayMurZ);

       if (InterPointRayMurX>=
           min(BasePointsCo(j,1),BasePointsCo(j+1,1)))&&
          (InterPointRayMurX<=
           max(BasePointsCo(j,1),BasePointsCo(j+1,1)))&&
          (InterPointRayMurZ<=
           min(BasePointsCo(j,3),BasePointsCo(j+1,3)))&&
          (InterPointRayMurZ>=0)

              InterPoints(i,2)=InterPointRayMurX;
              InterPoints(i,3)=InterPointRayMurY;
              InterPoints(i,4)=InterPointRayMurZ;
```

```
                InterPoints(i,10)=PlaneVerNormalVec(j,1);
                InterPoints(i,11)=PlaneVerNormalVec(j,2);
                InterPoints(i,12)=PlaneVerNormalVec(j,3);

        end
    end
end

if (DirRay(i,1)*PlaneVerNormalVec(length(PlaneEcVer),1)+DirRay(i,2)*
    PlaneVerNormalVec(length(PlaneEcVer),2)+DirRay(i,3)*
    PlaneVerNormalVec(length(PlaneEcVer),3))~=0

   [InterPointRayMurX,InterPointRayMurY,InterPointRayMurZ]=
   solve(PlaneEcVer(length(PlaneEcVer),1),RayEc1(i,1),RayEc2(i,1),x,y
   ,z);

   InterPointRayMurX=double(InterPointRayMurX);
   InterPointRayMurY=double(InterPointRayMurY);
   InterPointRayMurZ=double(InterPointRayMurZ);

   if (InterPointRayMurX>=
       min(BasePointsCo(length(PlaneEcVer),1),BasePointsCo(1,1)))&&
      (InterPointRayMurX<=
       max(BasePointsCo(length(PlaneEcVer),1),BasePointsCo(1,1)))&&
      (InterPointRayMurZ<=
       min(BasePointsCo(length(PlaneEcVer),3),BasePointsCo(1,3)))&&
      (InterPointRayMurZ>=0)

           InterPoints(i,2)=InterPointRayMurX;
           InterPoints(i,3)=InterPointRayMurY;
           InterPoints(i,4)=InterPointRayMurZ;

           InterPoints(i,10)=PlaneVerNormalVec(length(PlaneEcVer),1);
           InterPoints(i,11)=PlaneVerNormalVec(length(PlaneEcVer),2);
           InterPoints(i,12)=PlaneVerNormalVec(length(PlaneEcVer),3);
   end
```

```
end
%Intersección con el plano de piso. En este caso, se evalúa el
producto punto entre el rayo y el plano de piso, si es distinto a
cero se calcula el punto de intersección existente, y por último se
evalúa si el rayo no se ha intersectado ni con los planos de
cubierta, ni con los planos de muros, pues de ser así, necesariamente
se intersectará con el plano de piso (ya que el origen de los rayos
se encuentra al interior del espacio encerrado por dichos planos,
luego si los rayos no se han intersectado ni con los planos de muros
ni con los planos de cubierta, la única opción adicional es que se
intersecten con el plano de piso). Para esta evaluación, se utilizan
los valores de las columnas 2, 3 y 4 del Cuadro 7 (InterPoints), ya
que si no ha existido la intersección con los planos de cubierta o de
muros, dichos valores serán iguales a cero (no se les habrá asignado
el valor de las coordenadas correspondientes al punto de
intersección). Vale la pena resaltar que al encontrar el punto de
intersección real y asignarle sus coordenadas a las columnas
correspondientes del Cuadro 7 (InterPoints), se asignan los cosenos
directores del vector normal al plano correspondiente de reflexión, a
las columnas 10, 11 y 12 de la misma tabla, pues corresponden
justamente a la definición del vector normal al plano de reflexión.

if (DirRay(i,1)*PlaneHorNormalVec(1,1)+DirRay(i,2)*
    PlaneHorNormalVec(1,2)+DirRay(i,3)*PlaneHorNormalVec(1,3))~=0

   [InterPointRayPisoX,InterPointRayPisoY,InterPointRayPisoZ]=
   solve(PlaneEcHor(1,1),RayEc1(i,1),RayEc2(i,1),x,y,z);

   InterPointRayPisoX=double(InterPointRayPisoX);
   InterPointRayPisoY=double(InterPointRayPisoY);
   InterPointRayPisoZ=double(InterPointRayPisoZ);

   if
   (InterPoints(i,2)==0)&&(InterPoints(i,3)==0)&&(InterPoints(i,4)==0
   )
```

```
        InterPoints(i,2)=InterPointRayPisoX;
        InterPoints(i,3)=InterPointRayPisoY;
        InterPoints(i,4)=InterPointRayPisoZ;

        InterPoints(i,10)=PlaneHorNormalVec(1,1);
        InterPoints(i,11)=PlaneHorNormalVec(1,2);
        InterPoints(i,12)=PlaneHorNormalVec(1,3);

    end
end
```

%Calcular la energía actual de cada rayo. Primero se calcula la distancia recorrida por el rayo (DistRecRay), desde su origen hasta el punto de intersección. Posteriormente, se utiliza la ecuación (36) para el cálculo de la energía del rayo en dicho punto.

```
DistRecRay(i,1)=sqrt((InterPoints(i,2)-CooFuente(1,1))^2+
(InterPoints(i,3)-CooFuente(1,2))^2+(InterPoints(i,4)-
CooFuente(1,3))^2)/1000; %Se divide sobre 1000 para convertir la
```

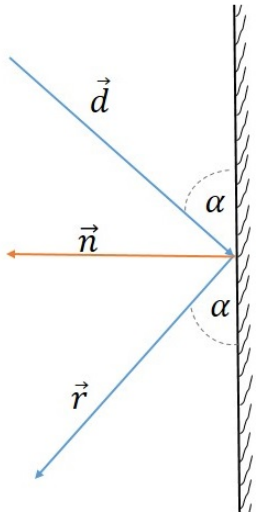
distancia a kilómetros, ya que el coeficiente de atenuación del aire está en decibeles por kilómetro.

```
InterPoints(i,6)=Eo*exp(-CoefAtenAire*DistRecRay(i,1))*
(1-CoefAtenMaterial);
```

%Calcular la energía acumulada en los receptores. El primer paso es evaluar si el rayo intersecta a las esferas que modelan a los receptores. Para esto, es utilizado el primer condicional dentro del for, en donde se evalúa si la distancia entre el centro de la esfera y el rayo (distancia perpendicular al rayo) es menor que el radio de la esfera, pues de ser así, existe dicha intersección. Una vez garantizada la intersección, se calcula la energía acumulada en los receptores utilizando la ecuación (35); esta energía se va acumulando con cada intersección que se presente entre los rayos y el receptor analizado.

```
    for j=1:length(ReceiversEc)

        EnerRec(j,1)=j;  %Llena la primera columna del Cuadro 7,
        correspondiente a la numeración de los receptores.

        if (sqrt(((ReceiversPointsCo(j,2)-CooFuente(1,2))*DirRay(i,3)-
           (ReceiversPointsCo(j,3)-CooFuente(1,3))*DirRay(i,2))^2+
           (-((ReceiversPointsCo(j,1)-CooFuente(1,1))*DirRay(i,3)-
           (ReceiversPointsCo(j,3)-CooFuente(1,3))*DirRay(i,1)))^2+
           ((ReceiversPointsCo(j,1)-CooFuente(1,1))*DirRay(i,2)-
           (ReceiversPointsCo(j,2)-CooFuente(1,2))*DirRay(i,1))^2)/
            sqrt((DirRay(i,1))^2+(DirRay(i,2))^2+(DirRay(i,3))^2))<0.3

                [InterPointRayRecX,InterPointRayRecY,InterPointRayRecZ]=
                solve(RayEc1(i,1),RayEc2(i,1),ReceiversEc(j,1),x,y,z);

                InterPointRayRecX=double(InterPointRayRecX);
                InterPointRayRecY=double(InterPointRayRecY);
                InterPointRayRecZ=double(InterPointRayRecZ);

                EnerRec(j,2)=EnerRec(j,2)+((InterPoints(i,6)*
                sqrt((InterPointRayRecX(2,1)-
                InterPointRayRecX(1,1))^2+(InterPointRayRecY(2,1)-
                InterPointRayRecY(1,1))^2+(InterPointRayRecZ(2,1)-
                InterPointRayRecZ(1,1))^2))/Vreceiver);
        end
    end
end

%-------------------------------------------------------------------------
%CICLO DE REFLEXIONES
%-------------------------------------------------------------------------

%El siguiente ciclo se lleva a cabo mientras que la enegía de los rayos
(la cual está asignada en InterPoints(:,6)), sea mayor o igual que el
```

```
criterio de parada pre-establecido, basado en el porcentaje de
discontinuidad de energía (EDP) (ver sección 3.3).
while max(InterPoints(:,6))>=(Eo*(100-EDP)/100)

    %--------------------------------------------------------------------
    %Número y generación de rayos
    %--------------------------------------------------------------------

    %Crear la matriz que contendrá los cosenos directores de los rayos,
    y sus respectivas ecuaciones. Para calcular los cosenos directores
    de los rayos, es necesario considerar la reflexión de éstos a partir
    de los puntos de intersección calculados anteriormente. Para
    calcular dicha reflexión se tiene que: r = d - (2d·n)/||n||^2 n (ver la figura a
    continuación). La anterior ecuación es la que se utiliza dentro del
    for para definir a DirRay. Una vez obtenido el vector director, se
    procede a normalizarlo. Por último, se determinan las ecuaciones que
    definen a cada rayo (son dos, ya que la línea se representa como la
    intersección de dos planos).
```




```
    DirRay=zeros(Nray,3);
    RayEc1=sym(zeros(Nray,1));
    RayEc2=sym(zeros(Nray,1));
```

```
for i=1:Nray

    DirRay(i,1)=InterPoints(i,7)-
    ((2*InterPoints(i,7)*InterPoints(i,10)+
    2*InterPoints(i,8)*InterPoints(i,11)+2*InterPoints(i,9)*
    InterPoints(i,12))/((InterPoints(i,10))^2+(InterPoints(i,11))^2+
    (InterPoints(i,12))^2))*InterPoints(i,10);

    DirRay(i,2)=InterPoints(i,8)-
    ((2*InterPoints(i,7)*InterPoints(i,10)+
    2*InterPoints(i,8)*InterPoints(i,11)+2*InterPoints(i,9)*
    InterPoints(i,12))/((InterPoints(i,10))^2+(InterPoints(i,11))^2+
    (InterPoints(i,12))^2))*InterPoints(i,11);

    DirRay(i,3)=InterPoints(i,9)-
    ((2*InterPoints(i,7)*InterPoints(i,10)+
    2*InterPoints(i,8)*InterPoints(i,11)+2*InterPoints(i,9)*
    InterPoints(i,12))/((InterPoints(i,10))^2+(InterPoints(i,11))^2+
    (InterPoints(i,12))^2))*InterPoints(i,12);

    %Normalizar el vector.
    DirRay(i,1)=DirRay(i,1)/sqrt(DirRay(i,1)^2+DirRay(i,2)^2+
    DirRay(i,3)^2);
    DirRay(i,2)=DirRay(i,2)/sqrt(DirRay(i,1)^2+DirRay(i,2)^2+
    DirRay(i,3)^2);

    DirRay(i,3)=DirRay(i,3)/sqrt(DirRay(i,1)^2+DirRay(i,2)^2+
    DirRay(i,3)^2);

    %Ecuaciones de los rayos.
    RayEc1(i,1)=DirRay(i,2)*x-DirRay(i,1)*y+(DirRay(i,1)*
    InterPoints(i,3)-DirRay(i,2)*InterPoints(i,2));

    RayEc2(i,1)=DirRay(i,3)*x-DirRay(i,1)*z+(DirRay(i,1)*
    InterPoints(i,4)-DirRay(i,3)*InterPoints(i,2));
end
```

```
%El procedimiento que sigue a continuación, hasta el título
“EVALUACIÓN FUNCIÓN OBJETIVO 3: UNIFORMIDAD EN LA DISTRIBUCIÓN DE LA
ENERGÍA SONORA EN LOS RECEPTORES”, corresponde a la actualización
del Cuadro 7, es decir, los pasos son los mismos que se llevaron a
cabo en el procedimiento anterior llamado “Calcular los puntos
iniciales de intersección (entre los rayos y los planos) y la
energía acumulada en los receptores”. La diferencia es que el origen
de los rayos, para cada reflexión, deja de ser correspondiente a las
coordenadas de la fuente de sonido, y para a ser el punto de
intersección (sobre el plano correspondiente) inmediatamente
anterior. Este procedimiento (la actualización del Cuadro 7, se
lleva a cabo con cada reflexión de cada rayo, hasta alcanzar el
criterio de parada, basado en el EDP.
for i=1:Nray

    InterPoints(i,7)=DirRay(i,1);
    InterPoints(i,8)=DirRay(i,2);
    InterPoints(i,9)=DirRay(i,3);

    InterPoints(i,13)=InterPoints(i,2);
    InterPoints(i,14)=InterPoints(i,3);
    InterPoints(i,15)=InterPoints(i,4);


    %Intersección con planos de cubierta.
    for j=1:length(PlaneEc)

        if (DirRay(i,1)*PlaneNormalVec(j,1)+DirRay(i,2)*
            PlaneNormalVec(j,2)+DirRay(i,3)*PlaneNormalVec(j,3))~=0

           [InterPointRayCubX,InterPointRayCubY,InterPointRayCubZ]=
           solve(RayEc1(i,1),RayEc2(i,1),PlaneEc(j,1),x,y,z);

           InterPointRayCubX=double(InterPointRayCubX);
           InterPointRayCubY=double(InterPointRayCubY);
           InterPointRayCubZ=double(InterPointRayCubZ);
```

```
if (InterPointRayCubX>=
    CooTabPopPoints(PlanePointsOrdered(j,1),2))&&
   (InterPointRayCubX<=
    max(CooTabPopPoints(PlanePointsOrdered(j,2),2),
    CooTabPopPoints(PlanePointsOrdered(j,3),2)))

    if (InterPointRayCubY>=
       ((CooTabPopPoints(PlanePointsOrdered(j,3),3)-
        CooTabPopPoints(PlanePointsOrdered(j,1),3))/
       (CooTabPopPoints(PlanePointsOrdered(j,3),2)-
        CooTabPopPoints(PlanePointsOrdered(j,1),2)))*
        InterPointRayCubX+
        CooTabPopPoints(PlanePointsOrdered(j,1),3)-
        CooTabPopPoints(PlanePointsOrdered(j,1),2)*
       ((CooTabPopPoints(PlanePointsOrdered(j,3),3)-
        CooTabPopPoints(PlanePointsOrdered(j,1),3))/
       (CooTabPopPoints(PlanePointsOrdered(j,3),2)-
        CooTabPopPoints(PlanePointsOrdered(j,1),2))))&&
       (InterPointRayCubY<=
       ((CooTabPopPoints(PlanePointsOrdered(j,2),3)-
        CooTabPopPoints(PlanePointsOrdered(j,1),3))/
       (CooTabPopPoints(PlanePointsOrdered(j,2),2)-
        CooTabPopPoints(PlanePointsOrdered(j,1),2)))*
        InterPointRayCubX+
        CooTabPopPoints(PlanePointsOrdered(j,1),3)-
        CooTabPopPoints(PlanePointsOrdered(j,1),2)*
       ((CooTabPopPoints(PlanePointsOrdered(j,2),3)-
        CooTabPopPoints(PlanePointsOrdered(j,1),3))/
       (CooTabPopPoints(PlanePointsOrdered(j,2),2)-
        CooTabPopPoints(PlanePointsOrdered(j,1),2))))

       if CooTabPopPoints(PlanePointsOrdered(j,3),2)>
          CooTabPopPoints(PlanePointsOrdered(j,2),2)

          if (InterPointRayCubY<=
```

```
        ((CooTabPopPoints(PlanePointsOrdered(j,3),3)-
         CooTabPopPoints(PlanePointsOrdered(j,2),3))/
        (CooTabPopPoints(PlanePointsOrdered(j,3),2)-

    CooTabPopPoints(PlanePointsOrdered(j,2),2)))*
         InterPointRayCubX+
         CooTabPopPoints(PlanePointsOrdered(j,2),3)-
         CooTabPopPoints(PlanePointsOrdered(j,2),2)*
        ((CooTabPopPoints(PlanePointsOrdered(j,3),3)-
         CooTabPopPoints(PlanePointsOrdered(j,2),3))/
        (CooTabPopPoints(PlanePointsOrdered(j,3),2)-

    CooTabPopPoints(PlanePointsOrdered(j,2),2))))

              InterPoints(i,2)=InterPointRayCubX;
              InterPoints(i,3)=InterPointRayCubY;
              InterPoints(i,4)=InterPointRayCubZ;

              InterPoints(i,10)=PlaneNormalVec(j,1);
              InterPoints(i,11)=PlaneNormalVec(j,2);
              InterPoints(i,12)=PlaneNormalVec(j,3);

        end

    else

        if (InterPointRayCubY>=
        ((CooTabPopPoints(PlanePointsOrdered(j,3),3)-
         CooTabPopPoints(PlanePointsOrdered(j,2),3))/
        (CooTabPopPoints(PlanePointsOrdered(j,3),2)-
        CooTabPopPoints(PlanePointsOrdered(j,2),2)))*
         InterPointRayCubX+
         CooTabPopPoints(PlanePointsOrdered(j,2),3)-
         CooTabPopPoints(PlanePointsOrdered(j,2),2)*
        ((CooTabPopPoints(PlanePointsOrdered(j,3),3)-
         CooTabPopPoints(PlanePointsOrdered(j,2),3))/
```

```
                    (CooTabPopPoints(PlanePointsOrdered(j,3),2)-
                    CooTabPopPoints(PlanePointsOrdered(j,2),2))))

                        InterPoints(i,2)=InterPointRayCubX;
                        InterPoints(i,3)=InterPointRayCubY;
                        InterPoints(i,4)=InterPointRayCubZ;

                        InterPoints(i,10)=PlaneNormalVec(j,1);
                        InterPoints(i,11)=PlaneNormalVec(j,2);
                        InterPoints(i,12)=PlaneNormalVec(j,3);

                end
            end
        end
    end
end

%Intersección con planos de muros.
for j=1:(length(PlaneEcVer)-1)

    if (DirRay(i,1)*PlaneVerNormalVec(j,1)+DirRay(i,2)*

    PlaneVerNormalVec(j,2)+DirRay(i,3)*PlaneVerNormalVec(j,3))~=
    0

       [InterPointRayMurX,InterPointRayMurY,InterPointRayMurZ]=
       solve(PlaneEcVer(j,1),RayEc1(i,1),RayEc2(i,1),x,y,z);

       InterPointRayMurX=double(InterPointRayMurX);
       InterPointRayMurY=double(InterPointRayMurY);
       InterPointRayMurZ=double(InterPointRayMurZ);


       if (InterPointRayMurX>=
           min(BasePointsCo(j,1),BasePointsCo(j+1,1)))&&
```

```
            (InterPointRayMurX<=
             max(BasePointsCo(j,1),BasePointsCo(j+1,1)))&&
            (InterPointRayMurZ<=
             min(BasePointsCo(j,3),BasePointsCo(j+1,3)))&&
            (InterPointRayMurZ>=0)

                InterPoints(i,2)=InterPointRayMurX;
                InterPoints(i,3)=InterPointRayMurY;
                InterPoints(i,4)=InterPointRayMurZ;

                InterPoints(i,10)=PlaneVerNormalVec(j,1);
                InterPoints(i,11)=PlaneVerNormalVec(j,2);
                InterPoints(i,12)=PlaneVerNormalVec(j,3);

        end
    end
end

if (DirRay(i,1)*PlaneVerNormalVec(length(PlaneEcVer),1)+
    DirRay(i,2)*PlaneVerNormalVec(length(PlaneEcVer),2)+
    DirRay(i,3)*PlaneVerNormalVec(length(PlaneEcVer),3))~=0

   [InterPointRayMurX,InterPointRayMurY,InterPointRayMurZ]=
   solve(PlaneEcVer(length(PlaneEcVer),1),RayEc1(i,1),RayEc2(i,1
   ),x,y,z);

   InterPointRayMurX=double(InterPointRayMurX);
   InterPointRayMurY=double(InterPointRayMurY);
   InterPointRayMurZ=double(InterPointRayMurZ);

   if (InterPointRayMurX>=

   min(BasePointsCo(length(PlaneEcVer),1),BasePointsCo(1,1)))&&
      (InterPointRayMurX<=

   max(BasePointsCo(length(PlaneEcVer),1),BasePointsCo(1,1)))&&
```

```
      (InterPointRayMurZ<=

   min(BasePointsCo(length(PlaneEcVer),3),BasePointsCo(1,3)))&&
      (InterPointRayMurZ>=0)

           InterPoints(i,2)=InterPointRayMurX;
           InterPoints(i,3)=InterPointRayMurY;
           InterPoints(i,4)=InterPointRayMurZ;

           InterPoints(i,10)=PlaneVerNormalVec(length(PlaneEcVer)
           ,1);
           InterPoints(i,11)=PlaneVerNormalVec(length(PlaneEcVer)
           ,2);
           InterPoints(i,12)=PlaneVerNormalVec(length(PlaneEcVer)
           ,3);

   end
end
%Intersección con el plano de piso.
if (DirRay(i,1)*PlaneHorNormalVec(1,1)+DirRay(i,2)*

PlaneHorNormalVec(1,2)+DirRay(i,3)*PlaneHorNormalVec(1,3))~=0

   [InterPointRayPisoX,InterPointRayPisoY,InterPointRayPisoZ]=
   solve(PlaneEcHor(1,1),RayEc1(i,1),RayEc2(i,1),x,y,z);

   InterPointRayPisoX=double(InterPointRayPisoX);
   InterPointRayPisoY=double(InterPointRayPisoY);
   InterPointRayPisoZ=double(InterPointRayPisoZ);

   if (InterPoints(i,2)==0)&&(InterPoints(i,3)==0)&&
      (InterPoints(i,4)==0)

           InterPoints(i,2)=InterPointRayPisoX;
           InterPoints(i,3)=InterPointRayPisoY;
           InterPoints(i,4)=InterPointRayPisoZ;
```

```
            InterPoints(i,10)=PlaneHorNormalVec(1,1);
            InterPoints(i,11)=PlaneHorNormalVec(1,2);
            InterPoints(i,12)=PlaneHorNormalVec(1,3);

    end
end

%Calcular la energía actual de cada rayo.
DistRecRay(i,1)=sqrt((InterPoints(i,2)-
InterPoints(i,13))^2+(InterPoints(i,3)-
InterPoints(i,14))^2+(InterPoints(i,4)-
InterPoints(i,15))^2)/1000;

InterPoints(i,6)=InterPoints(i,6)*
exp(-CoefAtenAire*DistRecRay(i,1))*(1-CoefAtenMaterial);

%Calcular la energía acumulada en los receptores.
for j=1:length(ReceiversEc)

    EnerRec(j,1)=j;

    If                          (sqrt(((ReceiversPointsCo(j,2)-
    InterPoints(i,14))*DirRay(i,3)-(ReceiversPointsCo(j,3)-
    InterPoints(i,15))*DirRay(i,2))^2+
    (-((ReceiversPointsCo(j,1)-InterPoints(i,13))*DirRay(i,3)-
    (ReceiversPointsCo(j,3)-
    InterPoints(i,15))*DirRay(i,1)))^2+((ReceiversPointsCo(j,1)-
    InterPoints(i,13))*DirRay(i,2)-(ReceiversPointsCo(j,2)-
    InterPoints(i,14))*DirRay(i,1))^2)/sqrt((DirRay(i,1))^2+(Dir
    Ray(i,2))^2+(DirRay(i,3))^2))<0.3

            [InterPointRayRecX,InterPointRayRecY,InterPointRayR
            ecZ]=
            solve(RayEc1(i,1),RayEc2(i,1),ReceiversEc(j,1),x,y,
            z);
```

```
                    InterPointRayRecX=double(InterPointRayRecX);
                    InterPointRayRecY=double(InterPointRayRecY);
                    InterPointRayRecZ=double(InterPointRayRecZ);

                    EnerRec(j,2)=EnerRec(j,2)+((InterPoints(i,6)*sqrt((
                    InterPointRayRecX(2,1)-InterPointRayRecX(1,1))^2+
                    (InterPointRayRecY(2,1)-InterPointRayRecY(1,1))^2+
                    (InterPointRayRecZ(2,1)-
                    InterPointRayRecZ(1,1))^2))/
                     Vreceiver);
            end
        end
    end
end

%EVALUACIÓN FUNCIÓN OBJETIVO 3: UNIFORMIDAD EN LA DISTRIBUCIÓN DE LA
ENERGÍA SONORA EN LOS RECEPTORES
%Este cálculo se hizo con base en la ecuación (37). Primero se calcula el
promedio de la energía acumulada en cada receptor (promedio de EnerRec).
Luego, se crea un vector (DifEnerRec) en donde se almacenará la
diferencia entre la energía acumulada en los receptores, y el promedio de
ésta (con respecto a todos los receptores). Por último, se obtiene la
función objetivo 3 al llevar a cabo la sumatoria de dicha diferencia.
MeanEnerRec=mean(EnerRec(:,2));
DifEnerRec=zeros(length(ReceiversEc),1);

for j=1:length(ReceiversEc)

    DifEnerRec(j,1)=abs(EnerRec(j,2)-MeanEnerRec); % Calcula el valor
    absoluto de la diferencia entre la energía acumulada en cada receptor
    y el promedio de la Energía acumulada en cada receptor. Si esa
    diferencia es cero, la energía en los receptores es uniforme.
end

fo3(k,1)=sum(DifEnerRec(:,1)); %El índice k corresponde al número de
partícula de ACCMOUPSO.
```

ANEXO E: Código en MatLab del Procedimiento Auto-Configurado para el Diseño Multi-Objetivo de Estructuras Reticulares (PACDMOER)

```
%-------------------------------------------------------------------------
%Generación automática de la geometría
%-------------------------------------------------------------------------

Ejecutar el Anexo B (Parte 1 + Parte 2 + Parte 3);

%-------------------------------------------------------------------------
%ACCMOUPSO
%-------------------------------------------------------------------------

%-------------------------------------------------------------------------
%Parámetros de entrada para el UPSO multiobjetivo (dentro del PSO)
%-------------------------------------------------------------------------

%Únicamente se define el número de partículas (numpar), el número de
iteraciones (numit), los pesos correspondientes a cada objetivo (w1 para
la energía de deformación, w2 para el peso, y w3 para el rendimiento
acústico), y la escala necesaria para que los tres objetivos queden
aproximadamente del mismo orden; en este caso se tomó como referencia el
orden de la energía de deformación, luego se definieron escalas
únicamente para el objetivo de peso y para el objetivo de rendimiento
acústico. Los parámetros c1, c2, X y u, no se definen ya que el PSO será
el encargado de encontrarlos.

numpar=3;
numit=6;
w1=1/3;
w2=1/3;
w3=1/3;
escalaw2=7;
escalaw3=5640171;
```

```
%-------------------------------------------------------------------------
%PSO para optimizar c1, c2, x y u del UPSO multiobjetivo
%-------------------------------------------------------------------------

%-------------------------------------------------------------------------
%Parámetros de entrada para el PSO
%-------------------------------------------------------------------------

%Para el PSO se definió el número de partículas (numparPSO), el número de
iteraciones (numitPSO), el factor de constricción (XPSO), y los
parámetros de aceleración (c1PSO y c2PSO). Adicionalmente, se definieron
unos rangos para generar la población inicial del PSO, es decir, los
parámetros del UPSO (c1, c2, X y u).
numparPSO=4;
numitPSO=20;
XPSO=0.5;
c1PSO=1.4;
c2PSO=1.4;

c1min=0;
c1max=2;
c2min=0;
c2max=2;
Xmin=0;
Xmax=1;
umin=0;
umax=1;

%-------------------------------------------------------------------------
%Población inicial del PSO
%-------------------------------------------------------------------------

%La población inicial del PSO constará de varios grupos de cuatro valores
(cada partícula es un grupo), correspondientes a los parámetros
requeridos por el UPSO (c1, c2, X y u). Para generar los primeros
```

```
valores, se tomó un número aleatorio contenido en el rango predefinido en
el paso anterior.
xinsub=zeros(numparPSO,4);

for i=1:numparPSO

    xinsub(i,1)=c1min+rand*(c1max-c1min);
    xinsub(i,2)=c2min+rand*(c2max-c2min);
    xinsub(i,3)=Xmin+rand*(Xmax-Xmin);
    xinsub(i,4)=umin+rand*(umax-umin);

end

%-------------------------------------------------------------------------
%Velocidad inicial del PSO
%-------------------------------------------------------------------------

%Los valores para la velocidad inicial de las partículas del PSO, se
generar de igual forma que los correspondientes a la población inicial.
vinsub=zeros(numparPSO,4);

for i=1:numparPSO

    vinsub(i,1)=c1min+rand*(c1max-c1min);
    vinsub(i,2)=c2min+rand*(c2max-c2min);
    vinsub(i,3)=Xmin+rand*(Xmax-Xmin);
    vinsub(i,4)=umin+rand*(umax-umin);

end

%-------------------------------------------------------------------------
%Evaluación inicial de función objetivo PSO
%-------------------------------------------------------------------------
```

```
%Iniciar los vectores correspondientes a la evaluación de la función objetivo del PSO sin restricciones (foPSO) y con restricciones (FOPSO), y los vectores que se utilizarán para las restricciones.
foPSO=zeros(numparPSO,1);
FOPSO=zeros(numparPSO,1);

%Inicio de vectores para las restricciones del PSO (ver sección 1):
g1PSO=zeros(numparPSO,1);
g2PSO=zeros(numparPSO,1);
g3PSO=zeros(numparPSO,1);
g4PSO=zeros(numparPSO,1);
g5PSO=zeros(numparPSO,1);
g6PSO=zeros(numparPSO,1);
g7PSO=zeros(numparPSO,1);
q1PSO=zeros(numparPSO,1);
q2PSO=zeros(numparPSO,1);
q3PSO=zeros(numparPSO,1);
q4PSO=zeros(numparPSO,1);
q5PSO=zeros(numparPSO,1);
q6PSO=zeros(numparPSO,1);
q7PSO=zeros(numparPSO,1);
teta1PSO=zeros(numparPSO,1);
teta2PSO=zeros(numparPSO,1);
teta3PSO=zeros(numparPSO,1);
teta4PSO=zeros(numparPSO,1);
teta5PSO=zeros(numparPSO,1);
teta6PSO=zeros(numparPSO,1);
teta7PSO=zeros(numparPSO,1);
gamma1PSO=zeros(numparPSO,1);
gamma2PSO=zeros(numparPSO,1);
gamma3PSO=zeros(numparPSO,1);
gamma4PSO=zeros(numparPSO,1);
gamma5PSO=zeros(numparPSO,1);
gamma6PSO=zeros(numparPSO,1);
gamma7PSO=zeros(numparPSO,1);
H1PSO=zeros(numparPSO,1);
```

```
H2PSO=zeros(numparPSO,1);
H3PSO=zeros(numparPSO,1);
H4PSO=zeros(numparPSO,1);
H5PSO=zeros(numparPSO,1);
H6PSO=zeros(numparPSO,1);
H7PSO=zeros(numparPSO,1);
HPSO=zeros(numparPSO,1);

%Inicia el ciclo en donde se evaluará la función objetivo del PSO, i.e.
un ciclo completo del UPSO multiobjetivo, llevado a cabo con los valores
de los parámetros c1, c2, X y u, correspondientes a cada partícula del
PSO.
for p=1:numparPSO

    %-------------------------------------------------------------------
    %Parámetros de entrada para el UPSO multiobjetivo
    %-------------------------------------------------------------------

    %Como se expuso anteriormente, los parámetros del UPSO corresponden a
    los valores de cada partícula del PSO.
    c1=xinsub(p,1);
    c2=xinsub(p,2);
    X=xinsub(p,3);
    u=xinsub(p,4);

    %-------------------------------------------------------------------
    %Población inicial del UPSO multiobjetivo
    %-------------------------------------------------------------------

    %La población inicial (xin) para el UPSO multiobjetivo corresponde a
    la coordenada Z de cada uno de los nodos de la estructura reticular
    de cubierta. El valor para dicha coordenada es asignado
    aleatoriamente dentro de un rango definido por una altura máxima y
    una mínima (ver anexo B, parte 1).
    [PP1,PP2]=size(PopPointsCo); %El tamaño calculado de ésta matriz
    (PopPointsCo) será utilizado en ciclos posteriores.
```

```
xin=zeros(numpar,(PP1-(length(BasePoints)/3)));

for i=1:numpar
    for j=1:(PP1-(length(BasePoints)/3))
        xin(i,j)=round((MinHeight+rand*(MaxHeight-
                MinHeight))*100)/100;
    end
end

%-----------------------------------------------------------------
%Velocidad inicial de las partículas del UPSO multiobjetivo
%-----------------------------------------------------------------

%Los valores para la velocidad inicial de las partículas del UPSO, se
generar de igual forma que los correspondientes a la población
inicial.
vin=zeros(numpar,(PP1-(length(BasePoints)/3)));

for i=1:numpar
    for j=1:(PP1-(length(BasePoints)/3))
        vin(i,j)=MinHeight+rand*(MaxHeight-MinHeight);
    end
end

G=vin; %La velocidad inicial es almacenada en "G" para su uso
posterior en la actualización de velocidades y posiciones del UPSO
multiobjetivo.
L=vin; %La velocidad inicial es almacenada en "L" para su uso
posterior en la actualización de velocidades y posiciones del UPSO
multiobjetivo.

%-----------------------------------------------------------------
%Evaluación inicial de las funciones objetivo del UPSO multiobjetivo
%-----------------------------------------------------------------
```

```
%Iniciar los vectores que contendrán la evaluación de las funciones objetivo para cada partícula, así como el vector que almacenará la función objetivo total (la combinación con pesos de las tres funciones objetivo). Adicionalmente se inicia la matris que contendrá los desplazamientos objetidos al ejecutar el FEM.
fo1=zeros(numpar,1);
fo2=zeros(numpar,1);
fo3=zeros(numpar,1);
FO=zeros(numpar,1);
Displacements=zeros(numpar,((PP1-(length(BasePoints)/3))*3));

%Evaluar las tres funciones objetivo (energía de deformación, peso y rendimiento acústico) para cada partícula.

Ejecutar el Anexo C (de donde se obtiene la función objetivo 1, fo1, y la función objetivo 2, fo2, para cada partícula);

Ejecutar el Anexo D (de donde se obtiene la función objetivo 3, fo3, para cada partícula);

%Aplicar las restricciones. La restricción 1 y 2 limita el valor de la coordenada Z de los nodos de la estructura reticular de cubierta, a un valor máximo y a un valor mínimo, predeterminados por el usuario (para una explicación mayor en el manejo y definición de las restricciones y de los componentes que las conforman, ver la sección 2).

    for q=1:(PP1-(length(BasePoints)/3))

        g1(k,q)=xin(k,q)-MaxHeight;
        g2(k,q)=MinHeight*0.5-xin(k,q);

        q1(k,q)=max(0,g1(k,q));
        q2(k,q)=max(0,g2(k,q));
```

```
if q1(k,q)<0.001
    teta1(k,q)=10;
else
    if q1(k,q)<=0.1
        teta1(k,q)=20;
    else
        if q1(k,q)<=1
            teta1(k,q)=100;
        else
            teta1(k,q)=300;
        end
    end
end

if q2(k,q)<0.001
    teta2(k,q)=10;
else
    if q2(k,q)<=0.1
        teta2(k,q)=20;
    else
        if q2(k,q)<=1
            teta2(k,q)=100;
        else
            teta2(k,q)=300;
        end
    end
end

if q1(k,q)<1
    gamma1(k,q)=1;
else
    gamma1(k,q)=2;
end

if q2(k,q)<1
    gamma2(k,q)=1;
```

```
        else
            gamma2(k,q)=2;
        end

        H1(k,q)=teta1(k,q)*q1(k,q)^(gamma1(k,q));
        H2(k,q)=teta2(k,q)*q2(k,q)^(gamma2(k,q));

        H(k,q)=H1(k,q)+H2(k,q);

    end
end

%Evaluar la función objetivo total, es decir, la que combina las tres
funciones objetivo, con sus respectivos pesos y escalas (para que se
maneje aproximadamente el mismo orden para las tres funciones
objetivo), y la penalización correspondiente a las restricciones.

for i=1:numpar
    FO(i,1)=(w1*fo1(i,1)+w2*fo2(i,1)*escalaw2+w3*fo3(i,1)*escalaw3)+s
    um(H(i,:));
end

%-------------------------------------------------------------------
%Definir la mejor posición de cada partícula del UPSO multiobjetivo
%-------------------------------------------------------------------

%Primero se inicia el vector que contendrá las mejores posiciones
encontradas por cada partícula. En este caso, por ser la primera
evaluación de la función objetivo que se lleva a cabo, dicho vector
se llena con las mismas posiciones iniciales (no existen otras
alternativas).
xbest=zeros(numpar,(PP1-(length(BasePoints)/3)));

for i=1:numpar
    for j=1:(PP1-(length(BasePoints)/3))
```

```
        xbest(i,j)=xin(i,j);
    end
end

FOxbest=FO; %Los valores de FO se almacenan en FOxbest para la
posterior actualización de la mejor posición de cada partícula.

%-----------------------------------------------------------------
%Definir la mejor posición del enjambre para el UPSO multiobjetivo
%-----------------------------------------------------------------

%La mejor posición global (de todo el enjambre) se obtiene buscando
el mínimo de FO, y el correspondiente índice que define la ubicación
de dicha posición dentro de la población de partículas.
xbestglob=zeros(1,(PP1-(length(BasePoints)/3)));
[bestglob,ibestglob]=min(FO);

%En el siguiente for, se almacena la mejor posición encontrada al
vector xbestglob, utilizando el índice que se obtuvo en el paso
inmediatamente anterior.

for j=1:(PP1-(length(BasePoints)/3))
    xbestglob(1,j)=xin(ibestglob,j);
end

%-----------------------------------------------------------------
%Definir la mejor posición del vecindario de cada partícula del UPSO
 multiobjetivo
%-----------------------------------------------------------------

%Primero se inicia el vector que contendrá las mejores posiciones
encontradas en el vecindario de cada partícula, xbestveci (el radio
del vecindario se definió como 1, es decir, el vecindario de la
partícula i serán las partículas i+1 e i-1). Debido a la anterior
situación de índices, el for general debe empezar en 2 y terminar en
el número total de partículas menos 1. Por este motivo, primero se
```

```
evalúa el vecindario de la partícula 1 (el cual corresponde
únicamente a la partícula 2), luego se evalúa el vecindario del resto
de partículas, hasta la penúltima, y por último se evalúa el
vecindario de la última partícula (el cual corresponde únicamente a
la penúltima partícula). Esta evaluación se lleva a cabo comparando
los valores de FO de cada partícula, en donde el menor será el mejor
del vecindario, y el valor correspondiente a dicha partícula se
almacenará en xbestveci.
xbestveci=zeros(numpar,(PP1-(length(BasePoints)/3)));

%Para la primera partícula:
if FO(1,1)<FO(2,1)
    for j=1:(PP1-(length(BasePoints)/3))
        xbestveci(1,j)=xin(1,j);
    end
else
    for j=1:(PP1-(length(BasePoints)/3))
        xbestveci(1,j)=xin(2,j);
    end
end

%Para el resto de partículas menos la última:
for i=2:(numpar-1)
    if FO(i,1)<FO(i-1,1)&&FO(i,1)<FO(i+1,1)
        for j=1:(PP1-(length(BasePoints)/3))
            xbestveci(i,j)=xin(i,j);
        end
    else
        if FO(i-1,1)<FO(i+1,1)
            for j=1:(PP1-(length(BasePoints)/3))
                xbestveci(i,j)=xin(i-1,j);
            end
        else
            for j=1:(PP1-(length(BasePoints)/3))
                xbestveci(i,j)=xin(i+1,j);
            end
```

```
        end
    end
end


%Para la última partícula:
if FO(numpar,1)<FO(numpar-1,1)
    for j=1:(PP1-(length(BasePoints)/3))
        xbestveci(numpar,j)=xin(numpar,j);
    end
else
    for j=1:(PP1-(length(BasePoints)/3))
        xbestveci(numpar,j)=xin(numpar-1,j);
    end
end

%Evaluación de la función objetivo con las mejores posiciones del
vecindario. En este paso se obtienen los valores de FO
correspondientes a dichas posiciones y se almacenan en un vector
llamado FOxbestveci (luego también se incluyen las restricciones, de
igual forma que se expuso anteriormente).

Ejecutar el Anexo C (de donde se obtiene la función objetivo 1, fo1,
y la función objetivo 2, fo2, para cada partícula);

Ejecutar el Anexo D (de donde se obtiene la función objetivo 3, fo3,
para cada partícula);

%-------------------------------------------------------------------
%Ciclo de optimización del UPSO multiobjetivo
%-------------------------------------------------------------------

%Acontinuación inicia el ciclo de optimización del UPSO
multiobjetivo, el cual va desde 1, hasta el número de iteraciones
predefinido (numit).
for j=1:numit
```

```
h=j; %Éste parámetro es utilizado en el manejo de restricciones
(ver sección 2).

%Actualizar la velocidad de las partículas y su posición, para,
posteriormente, evaluar las funciones objetivo con las nuevas
posiciones.
for i=1:numpar

    %Primero se calculan los parámetros que definen la
    actualización de la velocidad de las partículas, i.e. G y L.
    Luego, con éstos parámetros se calcula la nueva velocidad, la
    cual se utiliza para calcular la nueva posición de las
    partículas (para este proceso se utilizan las ecuaciones
    presentadas en la sección 2).
    for q=1:(PP1-(length(BasePoints)/3))

        G(i,q)=xinsub(p,3)*(G(i,q)+xinsub(p,1)*rand*(xbest(i,q)-
        xin(i,q))+xinsub(p,2)*rand*(xbestglob(1,q)-xin(i,q)));
        %Calcula el componente de la velocidad de la partícula
        que depende de la mejor posición que ha tenido la
        partícula y de la mejor posición que ha tenido el
        enjambre.

        L(i,q)=xinsub(p,3)*(L(i,q)+xinsub(p,1)*rand*(xbest(i,q)-
        xin(i,q))+xinsub(p,2)*rand*(xbestveci(i,q)-xin(i,q)));
        %Calcula el componente de la velocidad de la partícula
        que depende de la mejor posición que ha tenido la
        partícula y de la mejor posición que ha tenido el
        vecindario de la partícula.

        vin(i,q)=xinsub(p,4)*G(i,q)+(1-xinsub(p,4))*L(i,q);
        %Actualiza la velocidad de la partícula, dependiendo del
        factor de unificación (u) y de los componentes calculados
        anteriormente (G y L).
```

```
    xin(i,q)=round((xin(i,q)+vin(i,q))*100)/100;  %Calcula  la
    nueva posición de la partícula.

end

%Evaluar la función objetivo con las nuevas posiciones de las
partículas. En este paso se obtienen nuevos valores de FO,
correspondientes a dichas posiciones (incluyendo las tres
funciones objetivo y las restricciones).

Ejecutar el Anexo C (de donde se obtiene la función objetivo
1, fo1, y la función objetivo 2, fo2, para cada partícula);

Ejecutar el Anexo D (de donde se obtiene la función objetivo
3, fo3, para cada partícula);

%Evaluación función objetivo total

for t=1:numpar
    FO(t,1)=(w1*fo1(t,1)+w2*fo2(t,1)*escalaw2+
    w3*fo3(t,1)*escalaw3)+h*sum(H(t,:));
end

%Actualizar la mejor posición de cada partícula. Para esto,
se comparan los nuevos valores obtenidos para FO, con los
valores asignados a FOxbest (los mejores valores encontrados
por cada partícula). Si el nuevo valor es menor al
correspondiente a FOxbest, se reemplaza por el que estaba en
FOxbest, y también se reemplazan los valores de xbest (la
mejor posición de cada partícula) por las nuevas posiciones
(xin).
 for i=1:numpar
     if FO(i,1)<FOxbest(i,1)
         FOxbest(i,1)=FO(i,1);
         for q=1:(PP1-(length(BasePoints)/3))
             xbest(i,q)=xin(i,q);
```

```
        end
    end
end

%Actualizar la mejor posición del enjambre. En este caso, se
busca la mejor posición correspondiente a los nuevos valores
de FO. Si dicho valor es menor que el mejor valor que hasta
el momento había encontrado la totalidad del enjambre
(bestglob), se asigna éste (bestglobtrans) como el nuevo
mejor valor, y la correspondiente posición de la partícula
(xin) se asigna como la nuea mejor posición global
(xbestglob).
[bestglobtrans,ibestglobtrans]=min(FO);

if bestglobtrans<bestglob
    bestglob=bestglobtrans;
    for q=1:(PP1-(length(BasePoints)/3))
        xbestglob(1,q)=xin(ibestglobtrans,q);
    end
end

%Actualizar la mejor posición del vecindario. Por los mismos
motivos que se expusieron con anterioridad (ver, dentro de
éste mismo anexo, la definición de la mejor posición del
vecindario), ésta actualización se divide en tres partes, la
primera correspondiente a la primera partícula, la segunda al
resto de partículas hasta la penúltima, y la tercera a la
última partícula. En este caso, la comparación se hizo entre
los nuevos valores de FO, y los que hasta el momento eran los
mejores del vecindario (FOxbestveci). Si los nuevos valores
son mejores, se actualiza el vector FOxbestveci, así como la
correspondiente posición xbestveci.

%Para la primera partícula:
if FO(1,1)<FO(2,1)&&FO(1,1)<FOxbestveci(1,1)
    FOxbestveci(1,1)=FO(1,1);
```

```
    for q=1:(PP1-(length(BasePoints)/3))
        xbestveci(1,q)=xin(1,q);
    end
else
    if FO(2,1)<FOxbestveci(1,1)
        FOxbestveci(1,1)=FO(2,1);
        for q=1:(PP1-(length(BasePoints)/3))
            xbestveci(1,q)=xin(2,q);
        end
    end
end

%Para el resto de partículas hasta la penúltima:
for i=2:(numpar-1)
    if FO(i,1)<FO(i-1,1)&&FO(i,1)<FO(i+1,1)&&FO(i,1)<
       FOxbestveci(i,1)
        FOxbestveci(i,1)=FO(i,1);
        for q=1:(PP1-(length(BasePoints)/3))
            xbestveci(i,q)=xin(i,q);.
        end
    else
        if FO(i-1,1)<FO(i+1,1)&&FO(i-1,1)<FOxbestveci(i,1)
            FOxbestveci(i,1)=FO(i-1,1);
            for q=1:(PP1-(length(BasePoints)/3))
                xbestveci(i,q)=xin(i-1,q);.
            end
        else
            if FO(i+1,1)<FOxbestveci(i,1)
                FOxbestveci(i,1)=FO(i+1,1);
                for q=1:(PP1-(length(BasePoints)/3))
                    xbestveci(i,q)=xin(i+1,q);
                end
            end
        end
    end
end
```

```
    %Para la última partícula:
     if FO(numpar,1)<FO(numpar-1,1)&&FO(numpar,1)<
        FOxbestveci(numpar,1)
         FOxbestveci(numpar,1)=FO(numpar,1);
         for q=1:(PP1-(length(BasePoints)/3))
             xbestveci(numpar,q)=xin(numpar,q);
         end
     else
         if FO(numpar-1,1)<FOxbestveci(numpar,1)
             FOxbestveci(numpar,1)=FO(numpar-1,1);
             for q=1:(PP1-(length(BasePoints)/3))
                 xbestveci(numpar,q)=xin(numpar-1,q);
             end
         end
     end

end

%---------------------------------------------------------------
%Fin del ciclo de optimización del UPSO multiobjetivo
%---------------------------------------------------------------

%---------------------------------------------------------------
%Evaluación de la función objetivo del PSO
%---------------------------------------------------------------

%La evaluación de la función objetivo del PSO hace referencia al
rendimiento que tuvo el UPSO multiobjetivo con cada grupo de
parámetros que hicieron las veces de partículas del PSO. Por este
motivo, la función objetivo del PSO es el mejor valor que logró
encontrar el UPSO (bestglob), correspondiente a cada partícula
del PSO (grupo de parámetros que definen el comportamiento y el
rendimiento del UPSO, i.e. c1, c2, X y u).
foPSO(p,1)=bestglob;
```

```
%Aplicación de las restricciones correspondientes a las
partículas del PSO (al grupo de parámetros del UPSO). En este
caso, se plantearon 7 restricciones, las primeras 4 limitan los
parámetros a ser siempre positivos (mayores que cero), la quinta
limíta a un máximo los valores de c1 y c2 (la suma de éstos dos
parámetros se limita para que sea menor que 4, restricción
comúnmente encontrada en la literatura especializada del tema), y
por último, las restricciones número 6 y 7 limitan a los
parámetros X y u, para que tomen valores menor que 1 (condición
que debe cumplirse). Para una explicación más detallada sobre el
manejo y definición de las restricciones, ver la sección 2.

    g1PSO(p,1)=-xinsub(p,1);
    g2PSO(p,1)=-xinsub(p,2);
    g3PSO(p,1)=-xinsub(p,3);
    g4PSO(p,1)=-xinsub(p,4);
    g5PSO(p,1)=xinsub(p,1)+xinsub(p,2)-4;
    g6PSO(p,1)=xinsub(p,3)-1;
    g7PSO(p,1)=xinsub(p,4)-1;

    q1PSO(p,1)=max(0,g1PSO(p,1));
    q2PSO(p,1)=max(0,g2PSO(p,1));
    q3PSO(p,1)=max(0,g3PSO(p,1));
    q4PSO(p,1)=max(0,g4PSO(p,1));
    q5PSO(p,1)=max(0,g5PSO(p,1));
    q6PSO(p,1)=max(0,g6PSO(p,1));
    q7PSO(p,1)=max(0,g7PSO(p,1));

    if q1PSO(p,1)<0.001
        teta1PSO(p,1)=10;
    else
        if q1PSO(p,1)<=0.1
            teta1PSO(p,1)=20;
        else
            if q1PSO(p,1)<=1
```

```
            teta1PSO(p,1)=100;
        else
            teta1PSO(p,1)=300;
        end
    end
end

if q2PSO(p,1)<0.001
    teta2PSO(p,1)=10;
else
    if q2PSO(p,1)<=0.1
        teta2PSO(p,1)=20;
    else
        if q2PSO(p,1)<=1
            teta2PSO(p,1)=100;
        else
            teta2PSO(p,1)=300;
        end
    end
end

if q3PSO(p,1)<0.001
    teta3PSO(p,1)=10;
else
    if q3PSO(p,1)<=0.1
        teta3PSO(p,1)=20;
    else
        if q3PSO(p,1)<=1
            teta3PSO(p,1)=100;
        else
            teta3PSO(p,1)=300;
        end
    end
end

if q4PSO(p,1)<0.001
```

```
    teta4PSO(p,1)=10;
else
    if q4PSO(p,1)<=0.1
        teta4PSO(p,1)=20;
    else
        if q4PSO(p,1)<=1
            teta4PSO(p,1)=100;
        else
            teta4PSO(p,1)=300;
        end
    end
end

if q5PSO(p,1)<0.001
    teta5PSO(p,1)=10;
else
    if q5PSO(p,1)<=0.1
        teta5PSO(p,1)=20;
    else
        if q5PSO(p,1)<=1
            teta5PSO(p,1)=100;
        else
            teta5PSO(p,1)=300;
        end
    end
end

if q6PSO(p,1)<0.001
    teta6PSO(p,1)=10;
else
    if q6PSO(p,1)<=0.1
        teta6PSO(p,1)=20;
    else
        if q6PSO(p,1)<=1
            teta6PSO(p,1)=100;
        else
```

```
            teta6PSO(p,1)=300;
        end
    end
end

if q7PSO(p,1)<0.001
    teta7PSO(p,1)=10;
else
    if q7PSO(p,1)<=0.1
        teta7PSO(p,1)=20;
    else
        if q7PSO(p,1)<=1
            teta7PSO(p,1)=100;
        else
            teta7PSO(p,1)=300;
        end
    end
end



if q1PSO(p,1)<1
    gamma1PSO(p,1)=1;
else
    gamma1PSO(p,1)=2;
end

if q2PSO(p,1)<1
    gamma2PSO(p,1)=1;
else
    gamma2PSO(p,1)=2;
end

if q3PSO(p,1)<1
    gamma3PSO(p,1)=1;
else
```

```
    gamma3PSO(p,1)=2;
end

if q4PSO(p,1)<1
    gamma4PSO(p,1)=1;
else
    gamma4PSO(p,1)=2;
end

if q5PSO(p,1)<1
    gamma5PSO(p,1)=1;
else
    gamma5PSO(p,1)=2;
end

if q6PSO(p,1)<1
    gamma6PSO(p,1)=1;
else
    gamma6PSO(p,1)=2;
end

if q7PSO(p,1)<1
    gamma7PSO(p,1)=1;
else
    gamma7PSO(p,1)=2;
end

H1PSO(p,1)=teta1PSO(p,1)*q1PSO(p,1)^(gamma1PSO(p,1));
H2PSO(p,1)=teta2PSO(p,1)*q2PSO(p,1)^(gamma2PSO(p,1));
H3PSO(p,1)=teta3PSO(p,1)*q3PSO(p,1)^(gamma3PSO(p,1));
H4PSO(p,1)=teta4PSO(p,1)*q4PSO(p,1)^(gamma4PSO(p,1));
H5PSO(p,1)=teta5PSO(p,1)*q5PSO(p,1)^(gamma5PSO(p,1));
H6PSO(p,1)=teta6PSO(p,1)*q6PSO(p,1)^(gamma6PSO(p,1));
H7PSO(p,1)=teta7PSO(p,1)*q7PSO(p,1)^(gamma7PSO(p,1));
```

```
        HPSO(p,1)=H1PSO(p,1)+H2PSO(p,1)+H3PSO(p,1)+H4PSO(p,1)+H5PSO(
        p,1)+H6PSO(p,1)+H7PSO(p,1);

end



%Evaluación función objetivo para el PSO, con restricciones.
for i=1:numparPSO
    FOPSO(i,1)=foPSO(i,1)+sum(HPSO(i,:));
End

%Definir la mejor posición de cada partícula. En este caso se llena
el vector xinsubbest (mejores posiciones de las partículas) con las
mismas posiciones iniciales (por lo que es la primera iteración y no
hay más alternativas).
xinsubbest=xinsub;
FOxinsubbest=FOPSO; %los valores de la función objetivo (FOPSO) se
almacenan en el vector FOxinsubbest, el cual se utilizará,
posteriormente, en la actualización de las mejores posiciones de cada
partícula.
%Definir la mejor posición del enjambre. La mejor posición del
enjambre, corresponde a la posición de las partículas en donde FOPSO
fue mínima. En este sentido se define el mejor valor del enjambre
(bestglobPSO), y la posición a la que corresponde se asigna como la
mejor (xbestglobPSO).
[bestglobPSO,ibestglobPSO]=min(FOPSO);
xbestglobPSO=xinsub(ibestglobPSO,:);

%-------------------------------------------------------------------
%Ciclo de optimización del PSO
%-------------------------------------------------------------------

%El ciclo de optimización del PSO, va desde 1 hasta el número de
partículas predeterminado (numitPSO).
for b=1:numitPSO
```

```
    hPSO=b;  %Corresponde al parámetro h utilizado en las
restricciones.

    %Actualizar velocidades (vinsub) y posiciones (xinsub) para cada
    partícula. Las ecuaciones utilizadas corresponden al PSO
    original.
    for p=1:numparPSO

        vinsub(p,1)=XPSO*vinsub(p,1)+c1PSO*rand*(xinsubbest(p,1)-
        xinsub(p,1))+c2PSO*rand*(xbestglobPSO(1,1)-xinsub(p,1));

        vinsub(p,2)=XPSO*vinsub(p,2)+c1PSO*rand*(xinsubbest(p,2)-
        xinsub(p,2))+c2PSO*rand*(xbestglobPSO(1,2)-xinsub(p,2));

        vinsub(p,3)=XPSO*vinsub(p,3)+c1PSO*rand*(xinsubbest(p,3)-
        xinsub(p,3))+c2PSO*rand*(xbestglobPSO(1,3)-xinsub(p,3));

        vinsub(p,4)=XPSO*vinsub(p,4)+c1PSO*rand*(xinsubbest(p,4)-
        xinsub(p,4))+c2PSO*rand*(xbestglobPSO(1,4)-xinsub(p,4));

        xinsub(p,1)=xinsub(p,1)+vinsub(p,1);
        xinsub(p,2)=xinsub(p,2)+vinsub(p,2);
        xinsub(p,3)=xinsub(p,3)+vinsub(p,3);
        xinsub(p,4)=xinsub(p,4)+vinsub(p,4);

        %Evaluar la función objetivo. Como ya se dijo, la evaluación
        de la función objetivo del PSO corresponde a una ejecución
        completa del UPSO multiobjetivo, teniendo en cuenta las
        nuevas posiciones de las partículas del PSO (los nuevos
        grupos de parámetros. Por consiguiente, en este paso se debe
        ejecutar lo expuesto desde el título “Parámetros de entrada
        para el UPSO multiobjetivo“ (cerca del inicio del presente
        anexo), hasta el título “Fin del ciclo de optimización del
        UPSO multiobjetivo“ (aproximadamente 11 páginas y media
```

```
    después del título mencionado anteriormente), del presente
    anexo.

end

%Evaluación función objetivo del PSO. Como ya se dijo, la función
objetivo del PSO corresponde al rendimiento que tuvo el UPSO
multiobjetivo con cada grupo de parámetros que hicieron las veces
de partículas del PSO. Por este motivo, la función objetivo del
PSO es el mejor valor que logró encontrar el UPSO (bestglob),
correspondiente a cada partícula del PSO (grupo de parámetros que
definen el comportamiento y el rendimiento del UPSO, i.e. c1, c2,
X y u).
foPSO(p,1)=bestglob;

%Evaluación de la función objetivo del PSO con restricciones. Ver
la aplicación de restricciones en el anterior título de
“Evaluación de la función objetivo del PSO”.

for i=1:numparPSO
    FOPSO(i,1)=foPSO(i,1)+hPSO*sum(HPSO(i,:));
end

%Actualizar la mejor posición de cada partícula. Si la función
objetivo evaluada con las nuevas posiciones es menor que la
función objetivo evaluada con la mejor posición que ha tenido
hasta el momento la partícula, se actualiza la mejor posición de
la partícula (xinsubbest) con la nueva posición (xinsub).

for i=1:numparPSO
    if FOPSO(i,1)<FOxinsubbest(i,1)
        FOxinsubbest(i,1)=FOPSO(i,1);
        for q=1:4
            xinsubbest(i,q)=xinsub(i,q);
        end
    end
```

```
        end

        %Actualizar la mejor posición del enjambre. Si el nuevo mejor
        valor para la función objetivo (bestglobtransPSO), evaluado con
        las nuevas posiciones, es menor que la el correspondiente a la
        mejor posición que ha encontrado hasta el momento el enjambre, se
        actualiza la mejor posición global (xbestglobPSO) con la nueva
        posición (xinsub).
        [bestglobtransPSO,ibestglobtransPSO]=min(FOPSO);

        if bestglobtransPSO<bestglobPSO
            bestglobPSO=bestglobtransPSO;
            for q=1:4
                xbestglobPSO(1,q)=xinsub(ibestglobtransPSO,q);
            end
        end

    end %Cierra el for que se abrió justo al iniciar el título "Ciclo de
    optimización del PSO", aproximadamente hace 2 páginas y media. Es el
    for que va desde 1 hasta numitPSO.


%En este punto, el procedimiento se encuentra justo antes de iniciar la
última fase de ACCMOUPSO. En la cual se debe correr una vez más la
ejecución del UPSO multiobjetivo, tomando como parámetros de entrada los
correspondientes a la mejor posición (xbestglobPSO) que logró encontrar
el PSO que se finalizó algunas líneas atrás. Dicho procedimiento no se
volverá a presentar debido a que es exactamente igual al que se expuso
anteriormente, desde el título "Parámetros de entrada para el UPSO
multiobjetivo" (cerca del inicio del presente anexo), hasta el título
"Fin del ciclo de optimización del UPSO multiobjetivo" (aproximadamente
11 páginas y media después del título mencionado anteriormente), del
presente anexo. Vale la pena volver a resaltar que la única diferencia
sería la asignación de los parámetros c1, c2, X y u, los cuales, como ya
se dijo, corresponderán ahora a los valores de xbestglobPSO.
```

%Por último, es necesario aclarar que para el desarrollo de la herramienta de diseño propuesta, es necesario ejecutar la totalidad del procedimiento anterior (todo el anexo E), con distintas variaciones en la combinación de pesos. Cada una de ellas resultará en una configuración geométrica distinta, pero en general, todas presentarán un rendimiento similar (con respecto a los objetivos considerados). Consecuentemente, el diseñador podrá escoger una de éstas soluciones, y volver a correr el ACCMOUPSO con la combinación de pesos seleccionada, con más partículas y durante un mayor número de iteraciones. En este sentido, las primeras corridas mostrarán las posibilidades que existen para resolver el problema determinado, mientras que la última ejecución refinará el diseño (luego de haber seleccionado el resultado estético y funcional preferido) y lo acercará aún más a ser una solución óptima.