Ing. Carlos Iván García Obregón

Diseño por desempeño de edificación vertical mediante algoritmos genéticos

2025


Universidad Autónoma de Querétaro

Facultad de Ingeniería

# DISEÑO POR DESEMPEÑO DE EDIFICACIÓN VERTICAL MEDIANTE ALGORITMOS GENÉTICOS

Tesis

Que como parte de los requisitos para obtener el Grado de

Maestro en Ciencias (Estructuras)

Presenta:

Ing. Carlos Iván García Obregón

Dirigido por:

Dr. Luis Francisco Pérez Moreno

Querétaro, Qro., junio de 2025

Universidad Autónoma de Querétaro
Facultad de Ingeniería
Maestría en Ciencias (Estructuras)

# Diseño por desempeño de edificación vertical mediante algoritmos genéticos

Tesis

Que como parte de los requisitos para obtener el Grado de
Maestro en Ciencias (Estructuras)

Presenta:
Ing. Carlos Iván García Obregón

Dirigido por:
Dr. Luis Francisco Pérez Moreno

Dr. Luis Francisco Pérez Moreno
Presidente

Dr. Miguel Ángel Pérez Lara y Hernández
Secretario

Dr. Jaime Moisés Horta Rangel
Vocal

Dr. Enrique Rico García
Suplente

M. en C. Iván Fermín Arjona Catzim
Suplente

Centro Universitario, Querétaro, Qro.
Junio de 2025
México

*A mis padres, mis ejemplos a seguir y pilares en este camino*

*A mi novia Alejandra, mi apoyo incondicional*

---

## AGRADECIMIENTOS

A mis padres, por todo el apoyo, cariño, comprensión, ánimo y ejemplo que me brindaron en el trascurso de mi vida.

A mis hermanos, por siempre mantener la admiración mutua por nuestros logros.

A mi novia Alejandra, por la paciencia, apoyo y consolación en días complicados.

A mis abuelos, por estar pendientes de mi avance y por brindarme apoyo.

A mis tíos, Alejandro y Leticia, por brindarme un hogar durante todos estos años.

A mi amigo Guillermo, porque la amistad sigue siendo fuerte aun cuando la respuesta siempre fue “No puedo”.

Al Dr. Luis Francisco Pérez Moreno por su asesoramiento riguroso en este camino.

A mis compañeros de generación, por su acompañamiento y apoyo en esta etapa.

Al Dr. Iván Arjona Catzim, por su asesoramiento en mis dudas, revisiones y disposición.

A la Universidad Autónoma de Querétaro, por sus programas educativos.

Agradezco a la Secretaría de Ciencia, Humanidades, Tecnología e Innovación (SECIHTI) por otorgarme la beca que hizo posible este proyecto. Su apoyo fue esencial para mi formación y el desarrollo de esta investigación.

# ÍNDICE GENERAL

**AGRADECIMIENTOS** ..... **II**

**ÍNDICE GENERAL** ..... **III**

**ÍNDICE DE FIGURAS** ..... **VI**

**ÍNDICE DE TABLAS** ..... **VII**

**RESUMEN** ..... **VIII**

**ABSTRACT** ..... **IX**

**1. INTRODUCCIÓN** ..... **1**

1.1 JUSTIFICACIÓN ..... 4

1.2 DESCRIPCIÓN DEL PROBLEMA ..... 5

**2. ANTECEDENTES** ..... **7**

2.1 DISEÑO POR DESEMPEÑO ..... 7

*2.1.1 Análisis Pushover* ..... *8*

*2.1.2 Fases del diseño por desempeño* ..... *10*

*2.1.3 Nivel de desempeño* ..... *12*

*2.1.4 Niveles de elementos estructurales* ..... *13*

*2.1.5 Niveles de elementos no estructurales* ..... *14*

*2.1.6 Niveles de desempeño para las estructuras.* ..... *15*

*2.1.7 Estudios previos* ..... *16*

*2.1.8 Método de coeficientes de desplazamiento (MCD)* ..... *18*

2.2 EDIFICACIÓN VERTICAL ..... 24

2.3 ALGORITMOS EVOLUTIVOS ..... 25

2.4 ALGORITMOS GENÉTICOS ..... 27

*2.4.1 Función objetivo ..... 27*
*2.4.2 Selección ..... 29*
*2.4.3 Cruce ..... 30*
*2.4.4 Mutación ..... 31*
*2.4.5 Reducción ..... 32*
**3 HIPÓTESIS Y OBJETIVOS ..... 35**
3.1 HIPÓTESIS ..... 35
3.2 OBJETIVO GENERAL ..... 35
3.3 OBJETIVOS ESPECÍFICOS ..... 35
**4 METODOLOGÍA ..... 36**
4.1 SELECCIÓN DE LA ESTRUCTURA A ANALIZAR ..... 36
*4.1.1 Selección de la topología de la estructura que se analizará ..... 37*
*4.1.2 Ubicación y uso de la estructura para determinar cargas según la normativa. 38*
4.2 MODELACIÓN DE LA ESTRUCTURA ..... 40
*4.2.1 Modelación y análisis de la topología seleccionada en SAP 2000 ..... 40*
4.3 IMPLEMENTACIÓN DEL ALGORITMO GENÉTICO PARTIENDO DEL DISEÑO POR DESEMPEÑO .... 41
*4.3.1 Ejecución de análisis Pushover de la estructura. ..... 41*
*4.3.2 Determinación del nivel de desempeño ..... 43*
*4.3.3 Definición de la función objetivo ..... 44*
*4.3.4 Implementación y codificación del Algoritmo Genético. ..... 45*
4.4 COMPARACIÓN DE LOS RESULTADOS OBTENIDOS ..... 51
*4.4.1 Ejecución de análisis de la estructura seleccionada con métodos tradicionales. 51*
*4.4.2 Comparación del modelo general con la implementación del Algoritmo Genético y la metodología tradicional. ..... 51*
**5 RESULTADOS Y DISCUSIÓN ..... 52**
5.1 ESTRUCTURA SELECCIONADA ..... 52
*5.1.1 Estructura tipo ..... 52*

5.2 MODELO DE LA ESTRUCTURA ..... 52
*5.2.1 Modelo y análisis de la topología seleccionada en SAP 2000 ..... 52*
5.3 ALGORITMO GENÉTICO BASADO EN DISEÑO POR DESEMPEÑO ..... 55
*5.3.1 Herramienta "TESTSG". ..... 55*
*5.3.2 Herramienta "AG" ..... 58*
*5.3.3 Integración entre TESTSG y AG ..... 62*
*5.3.4 Resultados generales ..... 62*
5.4 COMPARACIÓN DE LOS RESULTADOS OBTENIDOS ..... 67
**6 CONCLUSIONES Y RECOMENDACIONES ..... 69**
**7 REFERENCIAS ..... 71**
**8 APÉNDICE ..... 74**
8.1 RESUMEN DE SIMULACIÓN 1. ..... 74
8.2 RESUMEN DE SIMULACIÓN 2 ..... 74
8.3 RESUMEN DE SIMULACIÓN 3 ..... 75
8.4 EVOLUCIÓN DE ÁREAS DE SIMULACIÓN 2 ..... 75
8.5 EVOLUCIÓN DE ÁREAS DE SIMULACIÓN 3 ..... 76
8.6 HERRAMIENTA "TESTSG" ..... 76
8.7 HERRAMIENTA "AG" ..... 92

## ÍNDICE DE FIGURAS

Figura 2.1 Análisis *Pushover* ..... 9

Figura 2.2 Representación bilineal de curva de capacidad ..... 19

Figura 2.3 Diagrama de flujo ..... 23

Figura 2.4 Cruce de puntos ..... 31

Figura 2.5 Mutación ..... 32

Figura 4.1 Diagrama de flujo de metodología ..... 36

Figura 4.2 Hospital Ángeles, Querétaro ..... 37

Figura 4.3 Espectro de respuesta de Ciudad de México ..... 38

Figura 5.1 Estructura modelada en SAP 2000 ..... 53

Figura 5.2 Distribución de perfiles estructúrales en modelación ..... 54

Figura 5.3 Elementos mecánicos de la modelación ..... 55

Figura 5.4 Evolución de desplazamiento por generación en Simulación 1 . 64

Figura 5.5 Evolución de desplazamiento por generación en Simulación 2 . 65

Figura 5.6 Evolución de desplazamiento por generación en simulación 3 . 66

Figura 5.7 Evolución de áreas de simulación 1 ..... 67

Figura 8.1 Evolución de áreas de simulación 2 ..... 75

Figura 8.2 Evolución de áreas de simulación 2 ..... 76

# ÍNDICE DE TABLAS

Tabla 1 Niveles de desempeño según la FEMA ..... 13

Tabla 2 Combinación de niveles de desempeño estructural ..... 15

Tabla 3 Valores de Co ..... 20

Tabla 4 Valore de C2 ..... 21

Tabla 5 Comparación de perfiles iniciales y optimizados (Simulación 1).... 63

Tabla 6 Comparación de perfiles iniciales y optimizados en Simulación 2 . 64

Tabla 7 Comparación de perfiles iniciales y optimizados (Simulación 3).... 65

Tabla 8 Resumen extendido de simulación 1 ..... 74

Tabla 9 Resumen extendido de simulación 2 ..... 74

Tabla 10 Resumen extendido de simulación 3 ..... 75

## RESUMEN

El diseño sísmico de edificaciones verticales de elementos de acero enfrenta el reto de controlar los desplazamientos laterales bajo cargas sísmicas, un aspecto que no se ha abordado ampliamente con técnicas de optimización avanzadas en estudios previos. La presente investigación propone minimizar los desplazamientos en marcos rígidos de acero mediante un Algoritmo Genético, optimizando las secciones de los elementos estructurales. Para lograrlo, se desarrolló un código que simula la evolución de soluciones bajo cargas sísmicas estáticas equivalentes, ajustando parámetros como la selección, el cruce y la mutación a lo largo de múltiples iteraciones. El modelo se parametrizó para estructuras regulares, utilizando un conjunto definido de perfiles de acero y evaluando la respuesta estructural con base en el diseño por desempeño. Los resultados muestran una reducción significativa en los desplazamientos laterales en comparación con diseños convencionales, validando la hipótesis de que los Algoritmos Genéticos permiten mejorar el comportamiento sísmico de estas estructuras. Sin embargo, el enfoque se limitó a geometrías regulares y a un grupo específico de secciones, lo que sugiere áreas de mejora. Se concluye que el método es efectivo para optimizar el desempeño sísmico, pero podría beneficiarse al incluir más tipos de perfiles y adaptarse a estructuras irregulares. Además, se recomienda ajustar el diseño a las Normas Técnicas Complementarias (NTC) 2023 de acero para detallar los elementos según requisitos actuales. Este trabajo aporta una alternativa práctica para el diseño sísmico, con potencial para extenderse a casos más complejos mediante ajustes en la parametrización de los códigos fuente.



**Palabras clave:** Análisis Pushover, optimización sísmica, ingeniería estructural computacional, sostenibilidad, resiliencia.

## ABSTRACT

The seismic design of vertical steel structures faces the challenge of controlling lateral displacements under seismic loads, an aspect that has not been widely addressed with advanced optimization techniques in previous studies. This research proposes minimizing displacements in rigid steel frames using a genetic algorithm, optimizing the sections of structural elements. To achieve this, a code was developed to simulate the evolution of solutions under equivalent static seismic loads, adjusting parameters such as selection, crossover, and mutation across multiple iterations. The model was parameterized for regular structures, employing a defined set of steel profiles and evaluating the structural response based on performance-based design.

The results demonstrate a significant reduction in lateral displacements compared to conventional designs, validating the hypothesis that genetic algorithms can enhance the seismic behavior of these structures. However, the approach was limited to regular geometries and a specific group of sections, suggesting areas for improvement. It is concluded that the method effectively optimizes seismic performance but could benefit from incorporating more profile types and adapting to irregular structures. Additionally, it is recommended to align the design with the 2023 Technical Complementary Standards (NTC) for steel to detail elements according to current requirements. This work provides a practical alternative for seismic design, with potential to extend to more complex cases through adjustments in the source code parameterization.



**Keywords:** Pushover analysis, seismic optimization, computational structural engineering, sustainability, resilience.

# 1. INTRODUCCIÓN

Una estructura, entendida como un sistema, es un conjunto de elementos conectados entre sí para cumplir una función. Los ejemplos más claros de este concepto son los edificios, puentes y torres, en el caso de la ingeniería civil, pero se pueden mencionar ejemplos en otras ramas de la ingeniería, como estructuras de barcos, aviones, tanques, recipientes de presión e inclusive líneas eléctricas.

En ingeniería civil una estructura es diseñada para desempeñar una función en específico, por lo que la persona encargada del diseño debe tener en cuenta aspectos fundamentales como la seguridad, la estética y el fácil mantenimiento, que dependen de las limitaciones económicas y ambientales de la región (Hibbeler, 2012). Así, el diseño estructural puede definirse como el conjunto de actividades que se realizan para determinar las características físicas de una estructura, de manera que cumpla con la función para la cual fue proyectada y que sea capaz de resistir los efectos de los agentes externos que le afectarán durante su vida útil.

Debido al mal comportamiento sísmico de algunas construcciones como el colapso del Hotel Prado, La Torre 4 del Conjunto Pino Suarez, el Edificio de Berlín y Liverpool, Conjunto Habitacional Nuevo León, durante el sismo de 1985 y aproximadamente 38 edificios colapsados y 12,253 construcciones dañadas a consecuencia del terremoto del 19 de septiembre de 2017 (Castillo, 2019) se han propuesto nuevos enfoques de diseño estructural. El diseño por desempeño es una de las metodologías más importantes por su énfasis el control explícito de la reacción o comportamiento de las estructuras frente a fuerzas o movimientos causados por sismos, lo cual también se conoce como respuesta dinámica (Sánchez & Terán , 2008).

Si bien la consideración de la respuesta dinámica antes mencionada es de importancia, también lo es la preocupación social y el poco espacio para viviendas

disponible en la actualidad. En este aspecto, la falta de zonas necesarias para desarrollos habitacionales es un resultado del crecimiento acelerado de la población y de los espacios inmobiliarios, creando una tendencia inclinada hacia construcciones que aprovechen mejor los espacios existentes.

La edificación vertical podría ser una solución a esta problemática, ya que puede mejorar la calidad de vida de las personas, incrementando la conservación de áreas verdes y el frenado de la construcción de zonas residenciales que estén demasiado alejadas de las zonas urbanizadas (Martínez Gil, 2021).

En este sentido, una herramienta que contribuye al desarrollo de proyectos de mayor calidad es la optimización de procesos ya que, aunque no es algo exclusivo de la edificación vertical, su implementación en el diseño estructural permite asegurar un aprovechamiento eficiente de recursos constructivos.

Los Algoritmos Genéticos en particular son métodos de optimización basados en la selección natural, cuyo objetivo es crear individuos cada vez más aptos para la búsqueda de soluciones óptimas de un problema que mediante un objetivo ya definido, dará la posibilidad de evaluar la calidad de todas las propuestas (Gonzales, 2020).

En la literatura se pueden encontrar numerosos estudios en los que se utilizan estos principios, como el realizado por Pezeshk et al., (1997), en el que se llevó a cabo el diseño óptimo de marcos de acero planos combinando Algoritmos Genéticos con un análisis geométricamente no lineal y se compararon los resultados con los obtenidos a través de análisis lineales elásticos y métodos elásticos AISC-LRFD. Por otra parte, Foley y Schinler (2004) desarrollaron un algoritmo de diseño automatizado para marcos de acero con restricciones parciales y totales, implementado a través de un algoritmo evolutivo orientado a objetos y un modelo de plasticidad distribuida para considerar la no linealidad del material de los miembros y de las conexiones en el análisis. Asimismo, Xu et al., (2006) aplicaron

un método de optimización multicriterio para el diseño sísmico basado en el rendimiento de marcos de edificios de acero bajo cargas sísmicas estáticas equivalentes.

En los estudios mencionados no se ha considerado el uso de Algoritmos Genéticos para minimizar desplazamientos en marcos de acero bajo cargas sísmicas, ni se han explorado optimizaciones que prioricen las deformaciones laterales sobre otros parámetros como el costo, resistencia y peso. Por lo tanto, en este trabajo se aplicó este método para optimizar secciones, esperando encontrar una estructura con menor desplazamiento. Los resultados obtenidos muestran una reducción en el desplazamiento lateral, lo cual permite establecer que con la metodología propuesta se mejora el desempeño sísmico en edificaciones verticales con un valor añadido como lo es la reducción del peso total.

A continuación, se presenta la justificación para la realización de este trabajo y el planteamiento del problema desde la perspectiva del diseño por desempeño, la edificación vertical y la optimización de procesos como tendencias actuales.

Después, en el capítulo 2, se mencionan los antecedentes más relevantes del diseño por desempeño, partiendo del contexto histórico y de los objetivos de esta metodología. Además, en el mismo capítulo se habla de la edificación vertical como una tendencia al alza en la actualidad y de los Algoritmos Genéticos como método popular para la optimización de procesos.

En el capítulo 3 se presentan la hipótesis y los objetivos que se definieron para el desarrollo de esta investigación. Posteriormente, en el capítulo 4 se describen detalladamente las etapas para la metodología planteada con base en los objetivos específicos.

Finalmente, en los capítulos 5 y 6 se hace una discusión exhaustiva de los resultados obtenidos y se presentan las conclusiones y recomendaciones derivadas de esta investigación.

## 1.1 Justificación

La optimización en el diseño estructural tiene el recurrente desafío de garantizar la eficiencia, seguridad y sostenibilidad de la construcción, además de que la complejidad del mismo diseño sumando la gran cantidad de variables y restricciones a tomar en cuenta, hacen que la búsqueda de soluciones óptimas sea costosa y que sus ventajas sean insignificantes mediante métodos tradicionales. Estos métodos normalmente requieren grandes cantidades de tiempo y recursos para explorar las posibles combinaciones de diseño, por lo cual se dificulta la ejecución de proyectos de gran magnitud o con plazos de tiempo ya establecidos y ajustados.

Al introducir los Algoritmos Genéticos como un método de optimización se encuentran ventajas tales como la consideración de múltiples objetivos simultáneamente, adaptabilidad del proyecto a los cambios imprevistos y la altamente eficiente, pero poco intuitiva, generación de soluciones. Además de que la versatilidad de los Algoritmos Genéticos hace que su adaptación e inclusión a métodos tradicionales sea una herramienta de apoyo de gran valor para el modelado y análisis de estructuras.

Aun con el prometedor potencial que este tipo de algoritmos representan, su aplicación presenta varios desafíos y limitaciones entre los cuales están la correcta definición de la función objetivo, las variables de diseño que se buscan optimizar y la validación de las soluciones obtenidas.

En la presente investigación se buscaron soluciones de optimización mediante la investigación y desarrollo basado en Algoritmos Genéticos para la optimización del diseño estructural en edificación vertical. Se propone partir del diseño por desempeño estructural, así como explorar la efectividad y aplicabilidad de la innovadora metodología algorítmica en la vida real, con el objetivo general de

aumentar la calidad de las construcciones verticales y, adicionalmente, contribuir al desarrollo de nuevas metodologías en el campo del diseño estructural.

## 1.2 Descripción del problema

En el área de la construcción es posible observar una extensa cantidad de edificaciones o proyectos donde la estructura propuesta cumple con los mínimos de diseño estructural de los códigos o normativas locales. Aun con validaciones, estas estructuras no están diseñadas para resistir los efectos de algún agente externo imprevisto, lo que puede provocar colapsos estructurales, daños materiales, interrupción de servicios y riesgos en seguridad pública.

Para abordar estos desafíos, el diseño estructural optimizado mediante Algoritmos Genéticos puede ofrecer una mayor cantidad de ventajas reconocibles en comparación con métodos o tradicionales, pues permiten explorar un amplio espacio de posibles soluciones, permitiendo la identificación de mejores propuestas de diseños estructurales.

Por otra parte, los Algoritmos Genéticos pueden manejar, de forma simultánea, múltiples objetivos y restricciones de diseño como la resistencia estructural, la eficiencia en las propiedades de los materiales, así como los impactos económicos y ambientales. Hablando de la parte ambiental en ingeniería civil, el concepto es especialmente relevante para esta área en la actualidad, donde la sostenibilidad y la eficiencia energética tienen un papel fundamental y crítico en el diseño y construcción de infraestructura.

Adicionalmente, la aplicación de estos métodos de optimización contribuye al desarrollo de soluciones innovadoras que pueden convertirse en vanguardia hoy en día por la implementación de prácticas más avanzadas y eficientes para la industria de la construcción. El aprovechamiento eficiente de recursos en el diseño estructural no solo reduce costos, sino que también minimiza el volumen usado de materiales, permite redimensionar los elementos con geometrías óptimas y

aumentar la seguridad pública para las actividades que se desarrollan en las construcciones. Dicho aprovechamiento impacta positivamente en la competitividad de las futuras propuestas de diseños, brindando mayor calidad y optimización de proyectos.

# 2. ANTECEDENTES

En las últimas décadas, el diseño estructural ha experimentado una transformación significativa hacia la sostenibilidad y la eficiencia energética, evidenciando una mayor conciencia ambiental. Este cambio se ha visto respaldado por la integración de metodologías avanzadas, entre las cuales destaca el diseño por desempeño. Esta metodología, clave en la ingeniería estructural, se enfoca en la capacidad de una estructura para resistir cargas y eventos extremos, superando la simple conformidad con requisitos mínimos de diseño, como rigidez o desplazamientos permitidos.

## 2.1 Diseño por desempeño

El interés en el campo de la ingeniería estructural ha incrementado debido al deficiente comportamiento sísmico de ciertas construcciones que se han erigido siguiendo las normativas actuales como bien podrían ser las Normas Técnicas Complementarias. Este interés se ha intensificado tras las importantes pérdidas materiales y económicas ocasionadas por eventos sísmicos significativos, como los ocurridos en México en 1985, Loma Prieta en 1989, Northridge en 1994 y Kobe en 1995. Ante el inesperado nivel de estas pérdidas, se han propuesto puntos de vista más integrales de diseño sísmico, como el diseño por desempeño.

El diseño por desempeño ha evolucionado con el tiempo al incorporar avances en el análisis dinámico y las normativas sísmicas modernas. Esto permite analizar estructuras considerando múltiples niveles de respuesta ante eventos extremos, lo que lo ha posicionado como una opción destacada en regiones propensas a sismos. Su desarrollo refleja un esfuerzo por integrar herramientas analíticas más precisas y adaptadas a las necesidades de la ingeniería contemporánea (Plevris, et al., 2017).

El diseño por desempeño subraya la importancia de controlar de manera explícita la respuesta dinámica de las estructuras, pasando de enfocarse en la

resistencia a centrarse en el desempeño de la respuesta dinámica, entendida como el comportamiento de las estructuras ante movimientos por acciones externas como sismos (Gholizadeh, 2013).

Existen variables que influyen en el rendimiento de las estructuras y es complicado determinar si el diseño actual es el óptimo o si existen soluciones más económicas. Dado el aumento de los precios de los materiales, encontrar diseños estructurales rentables con mejor rendimiento se ha convertido en una prioridad. En respuesta, se han desarrollado metodologías de optimización estructural en las últimas décadas. El diseño basado en el desempeño de estructuras de acero está ganando interés, utilizando análisis no lineales para evaluar la respuesta sísmica (Gholizadeh, 2013).

La esencia del diseño por desempeño radica en evaluar el comportamiento estructural frente a acciones extremas como las resultantes de sismos, viento y asentamientos mediante herramientas avanzadas, como análisis dinámicos no lineales, análisis de elementos finitos y simulaciones de terremotos.

El objetivo fundamental es garantizar que la estructura es capaz de resistir las acciones esperadas y cumplir con los requisitos de resistencia, rigidez y durabilidad. Se habla de una optimización lograda por el uso los recursos disponibles de manera eficiente sin exceder los límites de diseño en condiciones extremas, lo cual implica el uso de metodologías enfocadas en la aplicación de cargas sísmicas y en el comportamiento de la estructura.

### 2.1.1 Análisis *Pushover*

El análisis de *Pushover* es un método simplificado que aplica cargas sísmicas de manera incremental hasta que se produce un colapso plástico, siguiendo la propagación de la inelasticidad a través de articulaciones plásticas en los elementos estructurales.

Este método permite evaluar cómo responderá una estructura a cargas laterales crecientes, lo que es fundamental para garantizar la seguridad y funcionalidad durante un evento sísmico.

El análisis *Pushover* se ha consolidado como un método para estudiar la respuesta no lineal de estructuras mediante la aplicación gradual de cargas hasta alcanzar puntos críticos. Este procedimiento toma en cuenta la deformación asociada al primer modo de vibración, lo que permite una evaluación práctica del comportamiento bajo condiciones sísmicas. Su uso se ha extendido gracias a su capacidad para modelar efectos dinámicos de manera simplificada (Chopra, 2001).

En la práctica, el análisis de *Pushover* implica aplicar una carga lateral que aumenta de forma continua hasta que se alcanza un estado de colapso plástico. La carga se distribuye de acuerdo con la forma del primer modo de vibración, que es la forma en la que la estructura se deformaría más fácilmente bajo un evento sísmico ( Figura 2.1**)**. Esto es especialmente relevante porque las estructuras suelen tener su mayor deformación y desplazamiento en esta configuración (Ghorbanie-Asl, 2007).

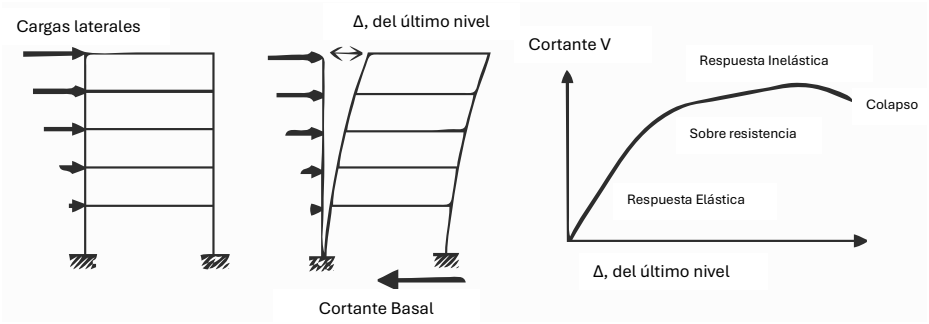


**Figura 2.1. Análisis *Pushover.***

La metodología no solo ayuda a identificar los puntos de falla potenciales y las debilidades en el diseño estructural, sino que también permite realizar ajustes antes de que la estructura sea construida. De esta manera, el análisis de *Pushover* contribuye significativamente a la creación de edificaciones más seguras y

resilientes frente a los sismos, optimizando tanto el rendimiento estructural como la eficiencia de costos.

En México, las Normas Técnicas Complementarias (NTC) han evolucionado haciendo énfasis en la resistencia desde 1987 hasta integrar conceptos de diseño por desempeño en 2023, reflejando la necesidad de controlar el daño en edificaciones tras eventos como el sismo de 2017.

La evolución de las normativas sísmicas en México refleja lecciones de eventos como el sismo de 1985 y posteriores. Inicialmente centradas en resistencia mínima en 1987, las Normas Técnicas Complementarias (NTC) de 2023, del Gobierno de la Ciudad de México, incorporan diseño por desempeño con análisis no lineales y niveles de daño aceptable. Estas actualizaciones son relevantes para el diseño de estructuras en zonas urbanas densas.

### 2.1.2 Fases del diseño por desempeño

Cubriendo las problemáticas antes mencionadas se ha propuesto un proceso integral de diseño por desempeño que abarca tres etapas principales: conceptual, numérica y de implementación.

La **fase conceptual** consiste en dar una solución tanto estructural como no estructural para el problema de diseño planteado. Comienza con la definición clara de los objetivos de diseño, los cuales deben reflejar las expectativas derivadas de la construcción.

Como siguiente paso, se evalúa la viabilidad técnica y económica de la obra en función de la sismicidad del sitio. De ser viable, se procede a desarrollar el diseño conceptual conforme a los objetivos y las características de los movimientos sísmicos anticipados. Durante esta fase, se establecen la configuración global de la construcción, la configuración estructural, los sistemas y materiales estructurales, el

sistema de cimentación, así como el tipo de elementos no estructurales y su conexión a la estructura.

La **fase numérica** consta de tres etapas:

1. Prediseño global: Utilizando espectros de respuesta, se determinan las características mecánicas relevantes de la estructura a nivel global para garantizar que su respuesta dinámica durante la excitación sísmica no exceda los límites establecidos según los criterios de desempeño.
2. Diseño local preliminar: Basado en las características mecánicas globales establecidas, se realiza el diseño detallado de la estructura, definiendo las dimensiones y detalles de los elementos estructurales.
3. Revisión del diseño: Se establecen pautas para revisar el diseño preliminar de la estructura mediante análisis estructurales detallados, especialmente si se espera que la estructura exhiba comportamiento plástico.

En la **fase de implementación**, se garantiza la calidad del diseño mediante una revisión exhaustiva e imparcial. Además, se reconoce que el éxito del diseño por desempeño depende de una supervisión continua y adecuada durante el mantenimiento, la ocupación y el funcionamiento de la estructura, así como de un control de calidad apropiado durante la construcción (Sánchez & Terán, 2008).

Con lo anterior mencionado, el desempeño sísmico se evaluará según la cantidad de daño recibido por la estructura y como este afectará a las actividades posteriores al terremoto.

Durante un evento sísmico, el comportamiento esperado de la estructura se debe establecer de forma cualitativa en la fase conceptual, antes de comenzar con la fase numérica y finalizar con la fase de implementación donde, por un control de calidad en la ejecución de obras, se garantiza la calidad de diseño. Cabe mencionar que en la fase conceptual el desempeño sísmico estará basado en el nivel de

desempeño, el nivel de amenaza y el desempeño esperado de la estructura (Sánchez & Terán, 2008).

### 2.1.3 Nivel de desempeño

Siendo un estado límite de daño, el nivel de desempeño representa la extensión de las afectaciones y se considerará el estado de elementos estructurales y no estructurales. Existen clasificaciones para esta variable y son expresados cualitativamente según el impacto en el usuario y la degradación de los elementos que conforman a la estructura.

Los métodos de ingeniería basada en el rendimiento de primera generación, como FEMA 356 y ASCE 41, clasifican el rendimiento en niveles discretos: operativo (OP), ocupación inmediata (IO), seguridad vital (LS) y previo al colapso (CP). Aunque esta categorización puede no ser relevante para los propietarios, es útil para ingenieros e investigadores, ya que les permite vincular estos niveles con parámetros de demanda de ingeniería (EDPs) y estimaciones de pérdidas sísmicas (Ghasemof, 2022).

La *Federal Emergency Management Agency (FEMA)* sugiere que existen cuatro niveles de desempeño:

A- Totalmente funcional: Los daños y efectos en los usuarios son despreciables.
B- Operacional: Los daños ocasionados por el evento sísmico se enfocan en elementos no estructurales y daños despreciables en elementos estructurales.
C- Seguridad: En este nivel los daños ocurren tanto en elementos estructurales como no estructurales que son capaces de degradar la rigidez y capacidad de resistencia del sistema.
D- Pre-colapso: Los daños ocasionados serán de suficiente magnitud como para que la estructura llegue al colapso.

En la Tabla 1 se muestra un resumen de todos los niveles de desempeño propuestos.

**Tabla 1. Niveles de desempeño según la FEMA.**

| Estado de daño | Nivel de desempeño | Características principales |
|---|---|---|
| Despreciable | Totalmente operacional | Daño estructural y no estructural despreciable o nulo. Las instalaciones continúan prestando sus servicios y funciones después del sismo. |
| Ligero | Operacional | Daños ligeros. Las instalaciones esenciales continúan en servicio y las no esenciales pueden sufrir interrupciones de inmediata recuperación. |
| Moderado | Seguridad | Daños moderados. La estructura sufre daños, pero permanece estable. Seguridad de ocupantes. Algunos elementos no estructurales pueden dañarse. |
| Severo | Pre-Colapso | Daño estructural severo, en la proximidad del colapso estructural. Falla de elementos no estructurales. Seguridad de ocupantes comprometida. |
| Completo | Colapso | Colapso estructural. |

### 2.1.4 Niveles de elementos estructurales

El nivel de desempeño proporciona información sobre el estado de daño de la estructura en tres límites y dos rangos intermedios, abreviados por SP-n ("*Strucutral Performance*").

SP-1. Inmediata ocupación: Presenta daño estructural limitado, riesgo para el usuario casi nulo y la estructura funciona con normalidad. (Limite)

SP-2. Daño controlado: El peligro para el ocupante es despreciable, pero pueden ser afectados en la rutina diaria. (Intermedio)

SP-3. Seguridad: La mayor parte de elementos estructurales presentan daños significativos y se presentan peligros contra la vida del usuario además de que el costo de reparaciones puede ser elevado. (Limite)

SP-4. Seguridad limitada: Rango intermedio entre el límite de seguridad y el límite de estabilidad y representan un peligro alto para el usuario. (Intermedio)

SP-5. Estabilidad estructural: Estado donde la estructura esta próximo o muy cerca de presentar un colapso debido a daños estructurales significativos donde la vida de los ocupantes está en riesgo alto y las reparaciones estructurales son sustanciales. (Limite)

SP-6. No considerado: Como tal, no representa un nivel desempeño de la estructura, si no que hace referencia a una evaluación sísmica de elementos no estructurales.

### 2.1.5 Niveles de elementos no estructurales

Se definen cuatro niveles o estados de daño que son utilizados para definir criterios en procesos de evaluación de estructuras y serán representados con la abreviación NP-n ("*Nonstructural Performance*").

Operacional NP-A: Los elementos no han cambiado de posición y siguen funcionando con normalidad.

Ocupación inmediata NP-B: Aun cuando los elementos permanecen en su sitio, no siguen cumpliendo con las funciones propias con normalidad e incluso algunos servicios externos pueden estar ausentes.

Seguridad NP-C: Se presentan daños considerables en la estructura, pero sin llegar al colapso y no pone en riesgo la vida del ocupante.

Amenaza reducida NP-D: Existen daños de gran magnitud con los que, aun con su presencia, otros elementos no llegaran al fallo o al colapso.

### 2.1.6 Niveles de desempeño para las estructuras.

Se implementan combinaciones de niveles de los elementos estructurales y no estructurales en las cuales se puede divisar una representación más específica del comportamiento global del edificio. En la Tabla 2 se pueden apreciar todas las combinaciones de niveles de desempeño (FEMA440, 2005).

**Tabla 2. Combinación de niveles de desempeño estructural.**

| Niveles de desemeño no estructural | Niveles de desempeño estructural | | | | | |
|---|---|---|---|---|---|---|
| | SP1 | SP2 | SP3 | SP4 | SP5 | SP6 |
| NP-A | 1-A Operacional | 2-A | NR | NR | NR | NR |
| NP-B | 1-B Ocupacion Inmediata | 2-B | 3-B | NR | NR | NR |
| NP-C | 1-C Seguridad | 2-C | 3-C | 4-C | 5-C | 6-C |
| NP-D | NR | 2-D | E-D | 4-D | 5-D | 6-D |
| NR: Combinación no recomendada | | | | | | |

Operacional 1-A: Daños estructurales limitados y los daños en elementos no estructurales no interfieren en el funcionamiento de la estructura.

Ocupación inmediata 1-B: Se espera que las estructuras que se encuentran en este nivel puedan seguir funcionando después del evento sísmico.

Seguridad 3-C: La probabilidad de pérdidas humanas es casi nula y es el desempeño esperado cuando se aplican códigos comunes.

Antes de FEMA 356, el documento FEMA 273 (1997) sentó las bases del diseño por desempeño al introducir un marco para evaluar y rehabilitar estructuras existentes, con énfasis en el análisis Pushover como herramienta clave para predecir el comportamiento no lineal. Esto influyó en este trabajo, que combina análisis Pushover con Algoritmos Genéticos para optimizar el desempeño sísmico de edificaciones verticales.

Un marco sistemático para rehabilitar edificios existentes bajo el diseño por desempeño fue introducido por FEMA 273 (1997). El documento detalla niveles

como Operacional o Seguridad Vital, similares a los de FEMA 356, y promueve el análisis Pushover para evaluar el colapso plástico mediante cargas incrementales. Este trabajo histórico ofrece una base para entender cómo se integran métodos no lineales en la optimización estructural moderna.

### 2.1.7 Estudios previos

Para la evaluación sísmica se utiliza el análisis inelástico, el cual predice el comportamiento de las estructuras en futuros eventos sísmicos, lo que es crucial para el análisis por desempeño ya que con él se pueden tomar decisiones en términos de seguridad por la misma predicción de daños en la estructura. Los análisis inelásticos, a diferencia de las técnicas elásticas lineales, permiten estimar directamente las deformaciones y distorsiones inelásticas. Este proceso implica crear un modelo estructural y someterlo a simulaciones de movimientos sísmicos anticipados, utilizando los resultados para evaluar el desempeño según criterios de aceptación.

Existen varios métodos que permiten analizar el desempeño sísmico de una estructura como el procedimiento simplificado de análisis estático no lineal multimodal (NMP) que se centra en la normalización de las demandas de deformación para evaluar la exigencia sísmica de estructuras con efectos significativos de modos superiores fundamentado conceptualmente en el método N2 extendido (Kreslin, 2011).

En Zarrin, et al, (2021) se presenta un procedimiento de *Pushover* simplificado, llamado procedimiento de *Pushover* multimodal normalizado (NMP), diseñado para abordar las limitaciones de los métodos convencionales. Su desarrollo es directo y reduce la complejidad en la evaluación sísmica de estructuras, utilizando el método N2 extendido para considerar los efectos de modos superiores bajo la suposición de que la estructura permanece elástica. A diferencia de las combinaciones modales cuadráticas tradicionales, que ignoran la reversión

de signo en los desplazamientos, el NMP utiliza un método de suma directa para combinar respuestas modales máximas de diferentes modos. Las respuestas finales se obtienen modificando los resultados de un análisis de *Pushover* convencional con factores de corrección que comparan las derivas de las historias normalizadas.

También existen métodos donde se lleva a cabo un análisis multi objetivo, donde el diseño por desempeño se enfoca en equilibrar dos objetivos conflictivos: minimizar el costo de construcción inicial y reducir el costo anual de reparación del edificio, dentro de ciertas limitaciones de diseño. Para lograr esto, se utiliza el método FEMA P-58, que es una herramienta ampliamente reconocida para evaluar el riesgo sísmico y estimar pérdidas en edificios. Este método examina el rendimiento del edificio de forma probabilística, analizando indicadores como costos de reparación, duración de las reparaciones, emisiones de carbono, energía incorporada, lesiones y fatalidades (Ghasemof, 2022).

Actualmente se entiende que el desempeño de una estructura durante un terremoto depende de la demanda de ductilidad a la que está expuesta, así como de los desplazamientos que experimenta (Ghorbanie-Asl, 2007).

El método de diseño basado en fuerzas, que se centra en las resistencias y aborda los desplazamientos de manera indirecta, no logra garantizar un nivel de rendimiento consistente. Esto ha generado inquietudes sobre la fiabilidad y la economía de diseño vigente.

Los métodos de diseño basados en desplazamientos (DBD) son más eficaces para asegurar un rendimiento uniforme, lo que ha llevado a un aumento significativo en su interés en los últimos años, haciendo que sean consideradas como las metodologías recomendadas para el diseño sísmico en los futuros códigos de construcción (Ghorbanie-Asl, 2007).

Priestley (2000) destaca que el diseño por desempeño evoluciona hacia métodos basados en desplazamientos (DBD), donde el control explícito de deformaciones mejora la predictibilidad del comportamiento sísmico. Esta perspectiva refuerza la elección del método de coeficientes de desplazamiento en esta investigación, que busca optimizar desplazamientos máximos en edificaciones verticales mediante Algoritmos Genéticos.

El diseño basado en desplazamientos representa un avance en las metodologías de diseño sísmico al enfocarse en el control directo de las deformaciones estructurales. Este tipo de diseño busca predecir con mayor precisión el comportamiento ante eventos sísmicos, marcando una transición hacia técnicas que priorizan la respuesta dinámica sobre la resistencia tradicional (Priestley et al, 2007).

Controlar directamente desplazamientos y deformaciones inelásticas es la ventaja del diseño por desempeño frente a métodos basados en fuerzas, según Priestley (2000). En su propuesta, el diseño basado en desplazamientos (DBD) ajusta la rigidez para cumplir objetivos específicos, complementando métodos como el de coeficientes de desplazamiento. Esta teoría amplía la perspectiva sobre cómo optimizar el comportamiento sísmico de estructuras.

### 2.1.8 Método de coeficientes de desplazamiento (MCD)

Para la presente investigación se pretende implementar el “Método de coeficiente de desplazamiento” (MCD) de la FEMA 440, el cual se enfoca en la aproximación de desplazamientos para estimar el punto de desempeño mediante un procedimiento numérico directo y se limitará a estructuras que no tengan efecto de torsión (SEAOC, 1995).

Para esta metodología, la Agencia Federal para el Manejo de Emergencias, FEMA, propone una serie de pasos para encontrar el punto de desempeño:

1. Representación bilineal de la curva de capacidad.

   Dicha curva se usa para estimar el amortiguamiento viscoso equivalente $\beta_{eq.}$

   - Primeramente, se dibuja una línea recta con inicio en el origen cuya pendiente será igual la rigidez inicial $K_i$ de la estructura en el rango elástico.
   - Después se propone un punto de desempeño "B" ($d_{pi}$, $a_{pi}$), el cual servirá para obtener el espectro de demanda reducido.
   - Acto seguido, se traza una línea desde el punto B hasta cortar con la línea trazada en el primer paso, esta recta debe intersecar al punto A (Dy, Ay) y las áreas (A1 y A2) que quedan por arriba y por debajo del espectro de capacidad deberán de ser iguales. El punto A hace referencia a la cedencia de la estructura en el formato bilineal.
   - La línea OAB será la representación bilineal de la curva de capacidad, ( Figura 2.2).

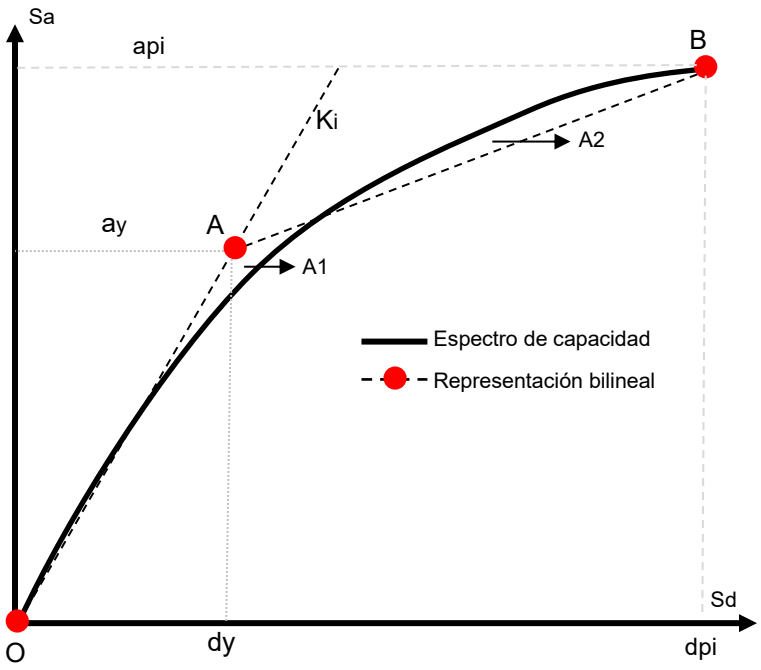


**Figura 2.2. Representación bilineal de curva de capacidad.**

2. Cálculo del periodo fundamental efectivo se observa en la ecuación ( 1 )

$$T_e = T_i \sqrt{\frac{K_i}{K_e}} \qquad (1)$$

Donde:
$T_e$ es el periodo fundamental,
$T_i$ es el periodo fundamental elástico,
$K_i$ es la rigidez lateral elástica y
$K_e$ Es la rigidez lateral efectiva.

3. Cálculo del punto de desempeño, ecuación ( 2 ),

$$D_t = C_0 C_1 C_2 C_3 S_a \frac{Sa(Te^2)}{4\pi^2} \qquad (2)$$

Donde:
$D_t$ Punto de desempeño,
$S_a$ es el valor de la aceleración espectral del periodo fundamental efectivo,
$C_n$ son factores modificadores:
C0: representa la relación entre el desplazamiento inelástico máximo probable y el desplazamiento espectral.

Su valor se puede definir por el factor de participación del primer modo de vibración y siguiendo los valores apropiados según la Tabla 3:

**Tabla 3. Valores de Co.**

| Número de Niveles | Valor de $C_0$ |
|---|---|
| 1 | 1.0 |
| 2 | 1.2 |
| 3 | 1.3 |
| 5 | 1.4 |
| Más de 10 | 1.5 |

$C_1$: es la relación entre el desplazamiento inelástico máximo esperado y el desplazamiento calculado mediante la ecuación ( 3 ).

$$C_1 = 1.0, \qquad T_e \geq T_c \; 1.0 + (R-1)\frac{T_c}{T_e}, \qquad T_e < T_c \; x, \qquad T_e < 0.1 \qquad (3)$$

$T_c$ es el periodo del espectro de respuesta y define al punto de transición del segmento de aceleración constante al segmento de velocidad constante.

R es la relación entre la demanda de resistencia inelástica y el coeficiente de resistencia de cadencia y se define según la ecuación ( 4 ).

$$R = \frac{\frac{S_a}{g}}{\frac{v_y}{W}} \frac{1}{C_0} \qquad (4)$$

$V_y$ es el cortante de cadencia de la representación bilineal de la curva de capacidad y W es el peso total de la estructura.

En la Tabla 4 se muestra los valores de $C_2$, los cuales hacen referencia al efecto de degradación de rigidez en dos tipos de estructuras, donde el Tipo 1 corresponde a estructuras con más del 30% del cortante en cualquier nivel resistido por los elementos y el Tipo 2 corresponde a las estructuras no incluidas en el Tipo 1.

**Tabla 4. Valores de C2.**

| Nivel de Desempeño Estructural | T = 0.1 s | | T ≥ Tc | |
|---|---|---|---|---|
| | Sistema Tipo 1 | Sistema Tipo 2 | Sistema Tipo 3 | Sistema Tipo 4 |
| Ocupación Inmediata | 1.0 | 1.0 | 1.0 | 1.0 |
| Seguridad | 1.3 | 1.0 | 1.1 | 1.0 |
| Prevención al Colapso | 1.5 | 1.0 | 1.2 | 1.0 |

$C_3$: Según la ecuación ( 5 ), es el incremento del desplazamiento según los efectos de segundo orden, si la estructura presenta rigidez mayor al 5% de la rigidez elástica $K_i$, entonces $C_3 = 1.0$, de lo contrario:

$$C_3 = 1 + \frac{|\alpha|(R-1)^{3/2}}{T_e} \qquad (5)$$

Donde:

α es la relación entre rigidez post-cedencia $K_s$ y la rigidez elástica $K_i$.

En la Figura 2.3 se resume el diseño por desempeño a manera de diagrama de flujo.

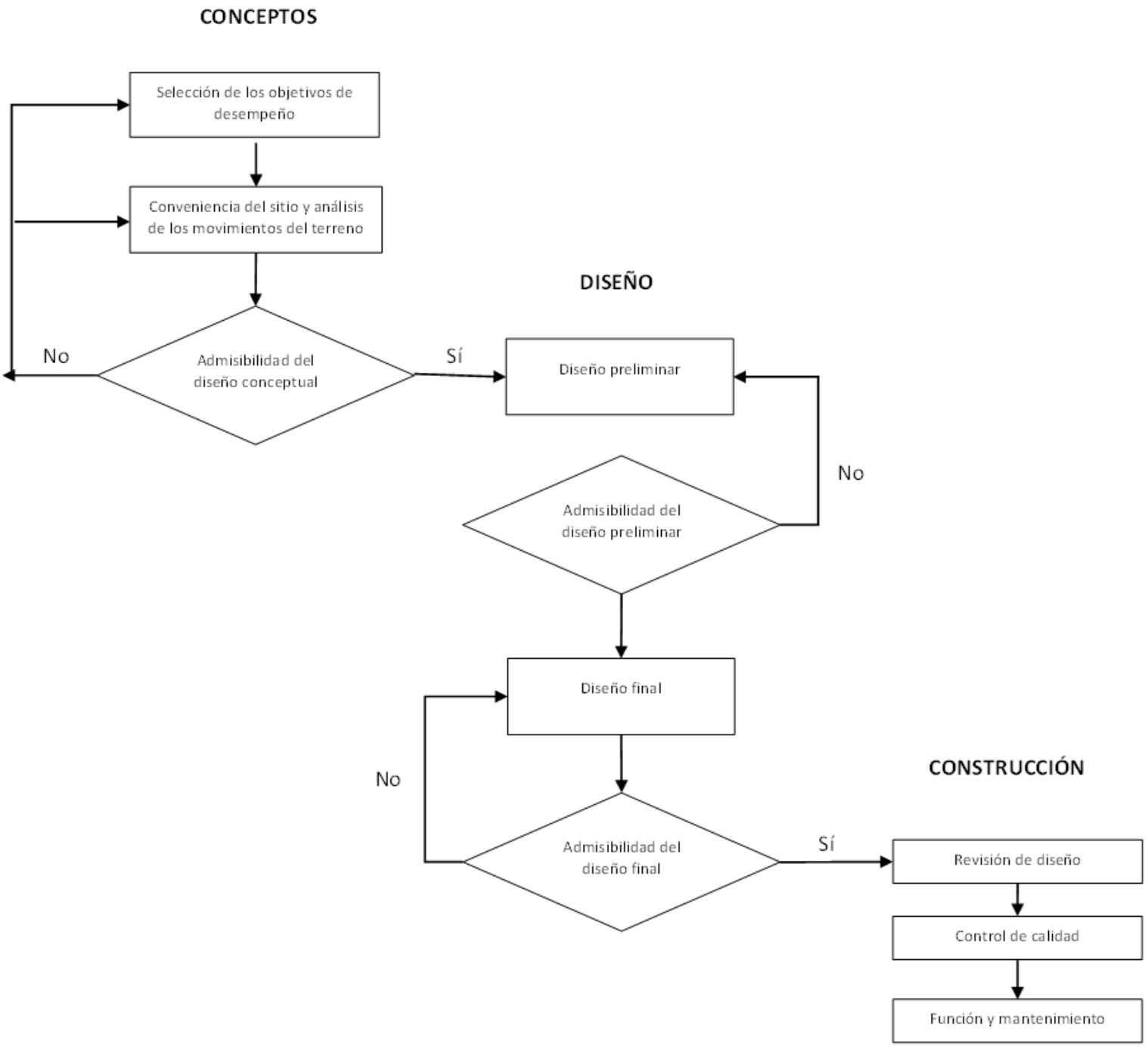


**Figura 2.3. Diagrama de flujo.**

## 2.2 Edificación vertical

Ahora bien, se debe considerar que para 2050 se proyecta que al menos un 75% de la población mundial residirá en ciudades y, por falta de espacio para desarrollo de proyectos inmobiliarios, dos aspectos importantes tomarán mayor fuerza. Por un lado, incrementará el desafío inherente a la conservación de áreas verdes y desarrollo de proyectos sostenibles y, por otro lado, el aumento en la tendencia de la construcción de vertical (Martínez Gil, 2021).

La edificación vertical ha surgido como una respuesta a la urbanización acelerada, integrando innovaciones tecnológicas y de diseño para adaptarse a entornos densamente poblados. Este tipo de construcción refleja un cambio en la planificación urbana, donde la funcionalidad debe equilibrarse con las restricciones impuestas por el contexto dando como resultado su evolución por la necesidad de optimizar el uso del espacio en ciudades modernas (Whitehead & Leslie, 2007).

Por lo que representa, la edificación vertical es una solución que reduce considerablemente los costos de redes de servicios como agua y luz eléctrica, mantenimiento e incluso seguridad. Aunado a esto, la edificación vertical mejora la calidad de vida de los habitantes, pues se ha convertido en la forma más fácil de acercarlos a sus fuentes de trabajo en comparación con la creación de zonas habitacionales de interés social que cada vez están más retiradas de la urbanización. El alejamiento de las zonas residenciales tiene consecuencias considerables, donde las más importantes son: el aumento de la delincuencia y el tráfico de automóviles por las horas de traslado (Martínez Gil, 2021).

Reducir la expansión urbana mediante edificaciones verticales es una solución sostenible que en Al-Kodmany (2018) se mencionan rascacielos con sistemas estructurales avanzados y materiales ecológicos como ejemplos clave, especialmente en megaciudades. Esta visión conecta con la necesidad de optimizar diseños para mejorar la eficiencia en contextos urbanos densos.

Yeang (2007) aboga por rascacielos ecológicos que combinen eficiencia estructural y sostenibilidad, una visión que complementa la optimización de secciones mediante Algoritmos Genéticos para reducir el uso de materiales y mejorar el desempeño sísmico en edificaciones verticales.

La alta sismicidad y la densidad urbana son desafíos clave para la edificación vertical en México. Diseños optimizados que reduzcan el impacto ambiental se presentan como una necesidad para lograr sostenibilidad, destacando Gómez y Rodríguez (2020) ejemplos de Guadalajara que también son relevantes para Ciudad de México.

## 2.3 Algoritmos evolutivos

La optimización estructural se refiere a garantizar que un conjunto de materiales pueda resistir las acciones que dentro del análisis estructural se están tomando en consideración. En otras palabras, se trata de diseñar estructuras de manera eficiente para que puedan resistir las fuerzas aplicadas y minimizar desplazamientos según la teoría de análisis estructural (Christensen & Klarbring, 2009).

Durante las últimas cinco décadas, se ha investigado ampliamente la optimización estructural. Aunque inicialmente se empleaba la programación matemática como la técnica predominante, en la actualidad ha sido sustituida por otras estrategias metaheurísticas. Entre estas alternativas, los algoritmos evolutivos se destacan como los más usados (Sánchez, et al., 2012).

Los algoritmos evolutivos han ganado terreno en la resolución de problemas multidimensionales, marcando un cambio desde las matemáticas tradicionales hacia estrategias metaheurísticas más versátiles. Esta transición comenzó a consolidarse en las últimas décadas del siglo XX, mostrando su potencial para adaptarse a desafíos computacionales complejos (Dasgupta & Michalewicz, 1997).

Las técnicas metaheurísticas han diversificado sus fuentes de inspiración, incluyendo un método que emplea principios físicos como la interacción entre partículas cargadas para optimizar soluciones. Estas propuestas, basadas en leyes de electrostática y dinámica, ofrecen una alternativa a los algoritmos biológicos, destacándose por su eficiencia en problemas complejos. Así, se amplía el abanico de herramientas heurísticas disponibles para la resolución de desafíos computacionales (Kaveh & Talatahari, 2010).

Los algoritmos evolutivos constituyen una categoría de algoritmos de optimización que encuentran su inspiración en los procesos evolutivos biológicos. En el ámbito del diseño estructural, estos algoritmos se emplean de manera común para mejorar el rendimiento de las estructuras ante cargas extremas mediante la optimización de su geometría y topología (Eiben et al., 2015).

La metodología de los algoritmos evolutivos implica la creación de una población inicial de soluciones candidatas, a las cuales se les aplican operadores de selección, cruce y mutación para generar nuevas soluciones. Estas soluciones son evaluadas a través de una función de aptitud que cuantifica el desempeño de la estructura frente a cargas extremas, y aquellas más aptas son seleccionadas para la siguiente generación. Este proceso se repite iterativamente hasta alcanzar una solución óptima (Gonzalez , 2020).

En el diseño estructural, los algoritmos evolutivos se colocan como una herramienta sumamente eficaz y se combinan con el diseño por desempeño para crear estructuras más eficientes y capaces de resistir cargas extremas. Estos algoritmos son particularmente valiosos en problemas con espacios de búsqueda extensos y no lineales, donde otros métodos podrían no lograr encontrar soluciones en un tiempo razonable. Su aplicación se extiende más allá del ámbito estructural, abarcando campos como la medicina y la programación, donde se utilizan para optimizar diversos procesos y resolver problemas en espacios de búsqueda complejos y no lineales (Sánchez et al., 2012).

## 2.4 Algoritmos Genéticos

Los Algoritmos Genéticos surgieron como una herramienta de optimización inspirada en la evolución natural, proponiendo un marco donde la selección y los operadores genéticos generan soluciones progresivamente mejores (Holland, 1975).

Los Algoritmos Genéticos (AG) son el tipo de algoritmo evolutivo con más reconocimiento. Están basados en el comportamiento de la naturaleza por el principio de selección natural y son reconocidos como métodos de optimización de funciones.

El principal objetivo de los Algoritmos Genéticos es el de crear individuos cada vez más aptos, y con esta característica se pretende que el fin sea la optimización de cuando se buscar la mejor solución posible a un problema (Delyová, et al., 2021).

En esencia el algoritmo se convierte en un sistema de optimización que brinda una solución óptima de forma iterativa en un espacio donde se encuentran varias posibles soluciones y mediante una función objetivo previamente definida, se llega a la evaluación de calidad de las soluciones desarrolladas.

Los Algoritmos Genéticos se fundamentan en la selección de individuos y la aplicación de operadores de variación para evolucionar soluciones a lo largo de iteraciones. Este proceso, inspirado en la biología, permite explorar espacios de búsqueda amplios y ha sido estudiado como una alternativa efectiva para problemas de optimización (Mitchell, 1996).

### 2.4.1 Función objetivo

Dos aspectos cruciales en los Algoritmos Genéticos son la determinación de una función de adaptación adecuada y la codificación utilizada. Idealmente, las

funciones objetivo deben ser regulares, es decir, que, para dos individuos cercanos en el espacio de búsqueda, sus valores sean similares.

Un problema común es la presencia de muchos óptimos locales y un óptimo global aislado. Una buena función objetivo debe reflejar el valor real del individuo, pero en problemas de optimización combinatoria con muchas restricciones, muchos individuos pueden no ser válidos.

Para abordar individuos con restricciones, se proponen varias soluciones. Una es eliminar los individuos que no cumplen las restricciones o asignarles un valor objetivo de cero. Otra opción es reconstruir los individuos no válidos mediante un operador reparador.

Otro modo es penalizar la función objetivo de los individuos que violan restricciones, dividiendo el valor de la función por una cantidad relacionada con las restricciones violadas.

Existe una estructuración canónica que desarrollan a los Algoritmos Genéticos de esta índole en 6 pasos:

- Inicialización de población
- Evaluación de los individuos
- Selección de individuos como ancestros
- Generación de nuevos individuos
- Evaluación de nuevos individuos
- Remplazo de individuos en población

Generalizando, el primer paso será generar, de forma aleatoria, una población de individuos que serán las posibles soluciones y que serán evaluados mediante una función objetivos ya definida con anterioridad la cual reflejara la calidad de soluciones para el problema (Gonzalez, 2020).

Los algoritmos evolutivos pueden optimizar secciones estructurales para cumplir niveles de desempeño sísmico específicos, como ocupación inmediata. Lo

explorado por Fragiadakis y Lagaros (2011), integra análisis no lineales y restricciones múltiples, alineándose con métodos como el de coeficientes de desplazamiento. Su aplicación práctica destaca el potencial de las técnicas computacionales en el diseño sísmico.

La efectividad de los Algoritmos Genéticos depende de cómo se codifican y evalúan las soluciones, un proceso que permite medir su aptitud en problemas complejos. Esta metodología práctica ha facilitado su aplicación en la optimización de estructuras y en la reducción de desplazamientos (Goldberg, 1989).

### 2.4.2 Selección

La función de selección de padres más común, ecuación ( 6 ), es la proporcional a la función objetivo, donde cada individuo tiene una probabilidad de ser seleccionado como padre que corresponde al valor de su función objetivo.

$$p_{j,t}^{prop} = \frac{g(I_i^j)}{\sum_{j=1} \quad g(I_i^j)} \qquad (6)$$

Donde: $p_{j,t}^{prop}$ es la probabilidad de que el individuo $I_i^j$ sea seleccionado.

La selección de soluciones óptimas en Algoritmos Genéticos permite abordar problemas con comportamiento no lineal, como se demostró en un estudio donde se optimizaron marcos bajo condiciones complejas. Este trabajo destacó la capacidad de los Algoritmos Genéticos para manejar múltiples restricciones y explorar espacios de búsqueda amplios, mejorando la eficiencia de los procesos iterativos. Tales avances subrayan la versatilidad de estas técnicas en diversas áreas técnicas (Liu & Stephen, 2004).

Los individuos que mejor respuesta o comportamiento presenten serán seleccionados para convertirse en parte de la nueva generación de individuos. Acto seguido se definen los operadores de variación (cruce, mutación y reducción), estos

operadores tienen la función de ejecutar modificaciones en las características de copias generadas de los ancestros y una vez modificados se concentran en descendencia de la nueva generación. Esta dependencia será evaluada de la misma forma que los ancestros para finalmente regresar a la población inicial y reemplazar a los individuos por población mezclada de ancestros y descendientes, variando el porcentaje de los individuos de la población inicial (Gonzalez , 2020).

Resolver problemas con varios objetivos, como desplazamiento y peso, es posible gracias al algoritmo NSGA-II. Este método, desarrollado por Deb et al. (2002), usa selección elitista y no dominancia para mantener diversidad en las soluciones, mejorando la eficiencia respecto a Algoritmos Genéticos tradicionales. Su modelo multiobjetivo podría enriquecer estudios estructurales que buscan equilibrar múltiples factores.

### 2.4.3 Cruce

El Algoritmo Genético Canónico emplea un cruce de un punto, donde dos individuos seleccionados como padres se recombinan eligiendo un punto de corte e intercambiando las secciones a la derecha de dicho punto. Se han investigado otros operadores, concluyendo que con más de un punto de cruce se mejora el rendimiento del algoritmo.

La ventaja de usar múltiples puntos de cruce es una mejor exploración del espacio de búsqueda, aunque también aumenta la probabilidad de romper buenos esquemas. En el cruce de dos puntos, los cromosomas se consideran como un circuito donde se eligen aleatoriamente dos puntos para el cruce como se puede observar en la Figura 2.4.

El cruce en Algoritmos Genéticos combina características de soluciones existentes mediante variantes como el cruce de un punto o multipunto, generando diversidad en las poblaciones. Este operador ha sido objeto de estudio por su capacidad para influir en la exploración del espacio de búsqueda, ofreciendo

diferentes niveles de complejidad según su implementación. Su uso ha contribuido a entender mejor la dinámica evolutiva (Man et al., 1999).

Desde esta perspectiva, el cruce de un punto es un caso especial del cruce de dos puntos, donde uno de los puntos de corte está fijado al inicio de la secuencia que representa al individuo.

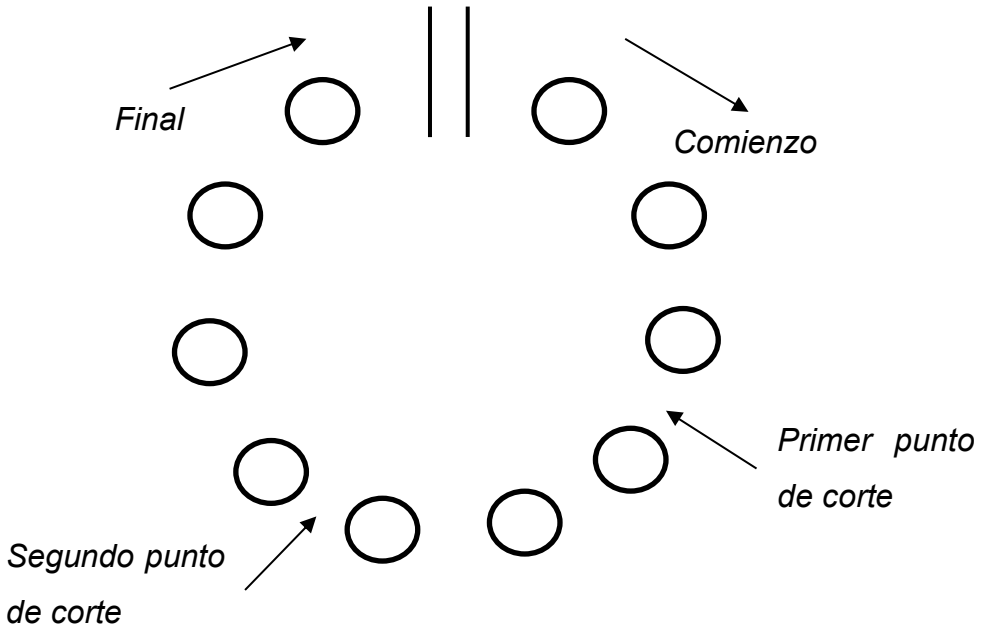


**Figura 2.4. Cruce de puntos.**

### 2.4.4 Mutación

La mutación es un operador básico que añade un pequeño elemento de aleatoriedad en el entorno de los individuos de la población. Aunque el cruce es el principal responsable de la búsqueda en el espacio de soluciones, varios experimentos indican que la importancia de la mutación aumenta a medida que la población converge.

Aunque en la mayoría de las implementaciones de Algoritmos Genéticos se asume que las probabilidades de cruce y mutación son constantes, algunos autores

han logrado mejores resultados ajustando la probabilidad de mutación a medida que aumentan las iteraciones.

La mutación introduce variaciones aleatorias en los Algoritmos Genéticos, lo que permite diversificar las soluciones dentro de una población en evolución. Este operador, ha sido analizado por su influencia en la adaptación de sistemas computacionales a entornos en constante cambio un ejemplo se muestra en la Figura 2.5, donde una cadena de números es mutada y adquiere nuevos valores al formar al nuevo individuo (Fogel, 2006).

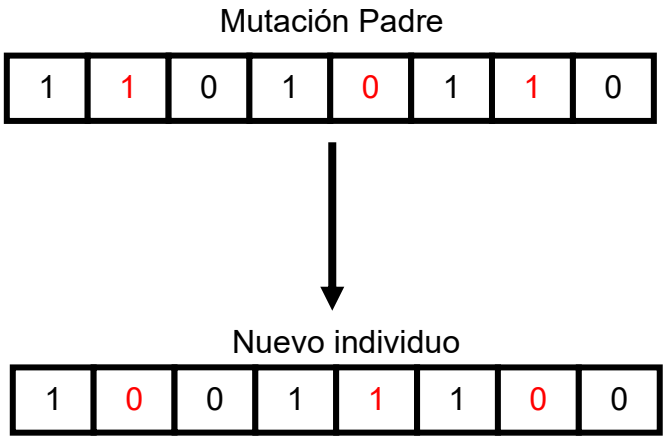


**Figura 2.5. Mutación.**

### 2.4.5 Reducción

La evolución de los Algoritmos Genéticos incluyó propuestas que los combinaron con técnicas matemáticas, como la integración del método Lagrange aumentado para mejorar la eficiencia. Esta fusión permitió optimizar soluciones bajo restricciones complejas, acelerando la convergencia y manejando penalizaciones de forma precisa. Tales desarrollos han ampliado las posibilidades de los métodos evolutivos en la resolución de problemas técnicos (Adeli & Cheng, 1994).

Después de obtener los descendientes de una población en el tiempo t el proceso de reducción al tamaño original consiste en elegir λ individuos de entre los λ que conforman la población en t, junto con los λ descendientes. Esto se realiza principalmente de dos formas distintas. En una, los λ descendientes son aquellos que formarán parte de la población en el tiempo t+1, conocido como reducción simple. En la otra, se seleccionan λ individuos más aptos entre los λ individuos, utilizando un criterio de reducción elitista de grado λ También se pueden considerar otros métodos de reducción, como elegir los λ mejores entre padres y descendientes, y luego seleccionar los λ - $\lambda_1$ restantes de entre los descendientes no seleccionados hasta ese momento. El concepto de reducción está asociado con la tasa de reemplazo generacional $t_{rg}$, que indica el porcentaje de descendientes generados en comparación con el tamaño total de la población.

Si los objetivos del problema son satisfechos se obtiene una gama de soluciones que ya están adaptadas a las características del problema y se podrá realizar una selección a través de un criterio cualitativo, pero si los objetivos aún no están cumplidos se deberá repetir el proceso desde la etapa de selección. (Gonzalez , 2020).

Para el crecimiento de las zonas urbanas en la actualidad, la edificación vertical y su optimización es de vital importancia. Se habla de beneficios como la reducción de huella de carbono de los proyectos futuros a desarrollar, mitigación del cambio climático y un mejor aprovechamiento de recursos como agua o minerales disminuyendo su sobreexplotación.

El poco espacio que queda en zonas urbanas y la introducción de la edificación vertical optimizada se verá reflejada en una exhaustiva búsqueda de soluciones enfocadas en la eficiencia, sostenibilidad e innovación. Esto permitirá crear competitividad entre empresas que cada vez se verán obligadas a la adopción de nuevas tecnologías y procesos constructivos que beneficiarán a la sociedad en conjunto.

Al término de la presente investigación se desarrolló un Algoritmo Genético basado en el diseño por desempeño para la optimización de elementos estructurales utilizados en la construcción de edificios. Esto teniendo especial atención en minimizar los desplazamientos resultantes de la configuración estructural propuesta. Mediante este algoritmo y su evaluación se generaron soluciones optimizadas en comparación con procesos de diseño actuales.

# 3 HIPÓTESIS Y OBJETIVOS

## 3.1 Hipótesis

El uso de un Algoritmo Genético que tome en cuenta la resistencia, propiedades de materiales y dimensiones de elementos, inherentes al diseño por desempeño, permite una minimización de los desplazamientos máximos de una estructura de edificación vertical.

## 3.2 Objetivo general

Implementar un Algoritmo Genético basado en el método de coeficientes de desplazamiento para una estructura de edificación vertical.

## 3.3 Objetivos específicos

1. Seleccionar la estructura a analizar considerando el número de niveles.
2. Modelar la estructura tipo en algún software de diseño estructural.
3. Implementar Algoritmo Genético partiendo del diseño por desempeño.
4. Comparar los resultados obtenidos con los métodos de diseño comunes.

# 4 METODOLOGÍA

En esta sección se describe el enfoque, técnicas y procedimientos utilizados para abordar el tema de estudio, que permitió dar validez a los resultados obtenidos. En la Figura 4.1. Diagrama de flujo de metodología., se muestra un diagrama de flujo ejemplificando, de forma general, el proceso de desarrollo de la presente investigación.

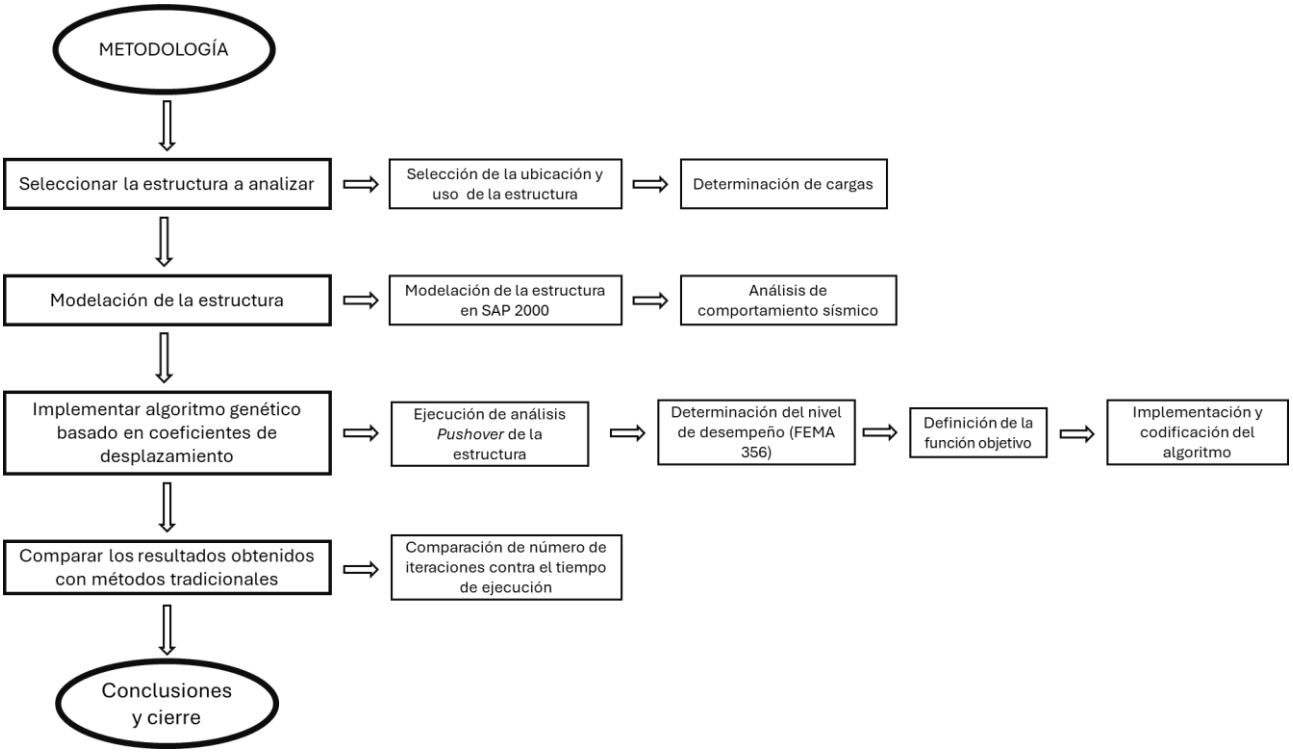


**Figura 4.1. Diagrama de flujo de metodología.**

## 4.1 Selección de la estructura a analizar

Esta etapa se enfocó en la selección de la estructura de análisis bajo cargas sísmicas para el algoritmo realizado. La metodología desarrollada estuvo basada en el método de los coeficientes de desplazamiento, método propio del diseño por desempeño sísmico.

### 4.1.1 Selección de la topología de la estructura que se analizará

Para la selección de la estructura se buscó la regularidad y simetría, pues al tener este tipo de configuración se minimiza la irregularidad planimétrica y vertical que pudo haber dificultado la interpretación y realización del análisis sísmico.

La propuesta de esta topología se inspiró en edificios representativos de México con características geométricas y estructurales similares, como el Hospital Ángeles Centro Sur en Querétaro ( *Figura* 4.2). Este hospital, desarrollado en un terreno de 46,954.59 m² con una construcción total de 52,065.99 m², se caracterizó por una geometría regular y simétrica, diseñada con pórticos rígidos y sistemas estructurales robustos para resistir las fuerzas sísmicas de la región.

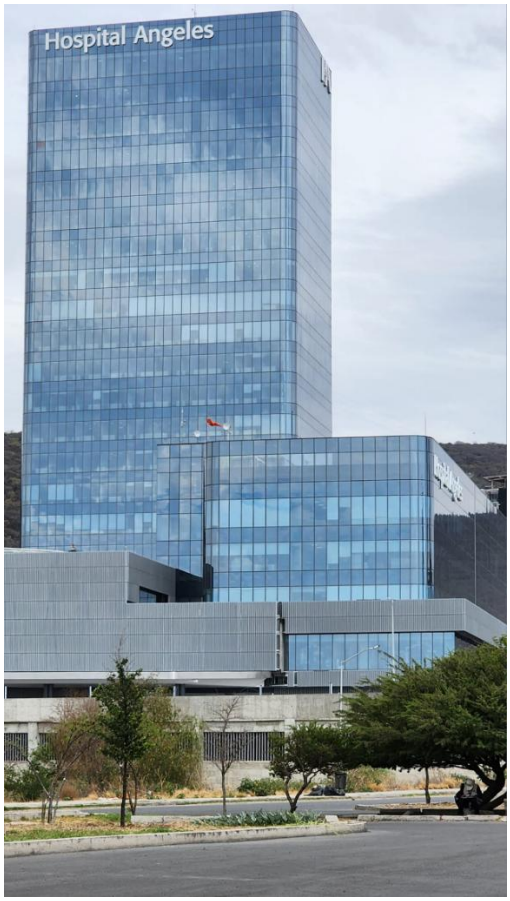


**Figura 4.2. Hospital Ángeles, Querétaro.**

La estructura tipo fue representada como un modelo de marcos, con elementos estructurales (vigas y columnas) distribuidos uniformemente en planta y en elevación**.** Este diseño inicial sirvió como punto de partida para las posteriores iteraciones realizadas resultado de su inclusión en el Algoritmo Genético descrito en la sección 4.3.

### 4.1.2 Ubicación y uso de la estructura para determinar cargas según la normativa.

Para la definición de cargas lo primero que se realizó fue la selección de ubicación de la estructura tipo, la cual fue el área de Ciudad de México, específicamente en las coordenadas de longitud -99.1319° O y latitud 19.196° N. Esta ubicación es relevante debido a su frecuente actividad sísmica, el espectro de respuesta fue extraído de Prodisis y se muestra en la Figura 4.3**.**

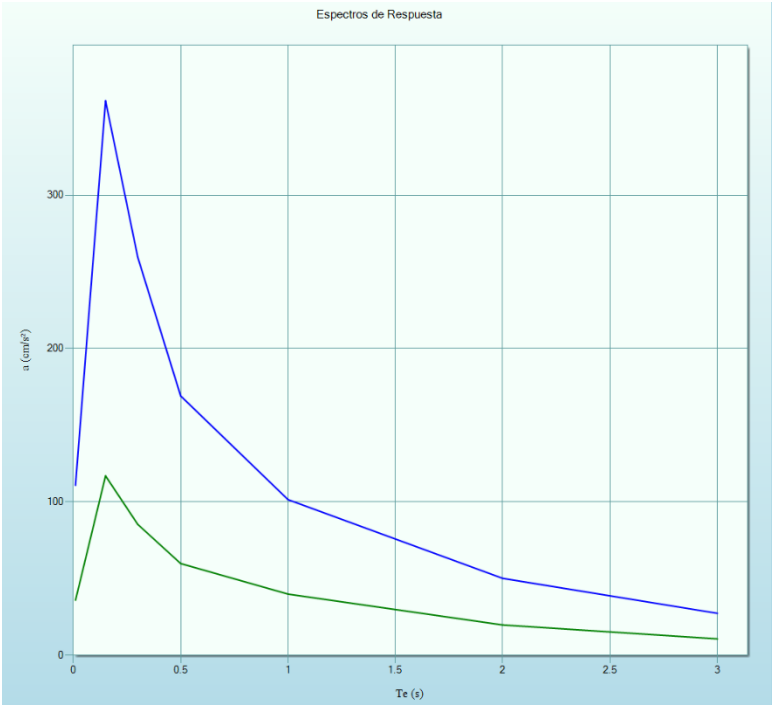


**Figura 4.3. Espectro de respuesta de Ciudad de México.**

En esta etapa también se determinaron las cargas muertas y vivas, además de las combinaciones de carga empleadas para el análisis símico, basándose en las Normas Técnicas Complementarias edición 2023(NTC-2023).

Las cargas muertas se consideraron como:

Losa aligerada:

- Análisis de carga losa entrepiso: Se determinó una carga muerta de 432.5 kg/m².
- Análisis de carga losa azotea: Se estableció una carga muerta de 554 kg/m².

Las cargas vivas se determinaron según el uso del edificio que en el caso de análisis será para uso habitacional (salas, dormitorios, comedores, pasillos) y azoteas, de acuerdo con las normativas aplicables:

- Cargas vivas en pisos habitables (salas, dormitorios, comedores, pasillos): 0.19 ton/m².
- Cargas vivas instantáneas en pisos habitables (salas, dormitorios, comedores, pasillos): 0.1 ton/m².
- Cargas vivas en azotea: 0.1 ton/m².
- Cargas vivas instantáneas en azoteas (salas, dormitorios, comedores, pasillos): 0.07 ton/m².

Finalmente, se establecieron las combinaciones de cargas conforme a las normativas, para evaluar el comportamiento estructural bajo diferentes escenarios de carga:

- Combinación 1: 1.3 CM + 1.5 CV.
- Combinación 2: 1.1 CM + 1.1 CV + 1.1 SX + 0.33 SY.
- Combinación 3: 1.1 CM + 1.1 CV + 0.33 SX + 1.1 SY.

Estas combinaciones permitieron determinar las demandas máximas en la estructura, considerando tanto las cargas muertas (CM), vivas (CV) como las

sísmicas en las direcciones X (SX) e Y (SY), asegurando que el diseño cumpliera con los criterios de seguridad y desempeño sísmico establecidos.

El método de coeficientes de desplazamiento, empleado para el diseño y evaluación sísmica de estructuras, se centra exclusivamente en determinar la respuesta de la estructura ante cargas sísmicas, considerando desplazamientos inelásticos y el comportamiento dinámico bajo sismo. Este método, descrito en normativas como el *Federal Emergency Management Agency* (FEMA 440), no requiere la incorporación de cargas de viento, ya que su objetivo principal es evaluar el desempeño sísmico de la estructura bajo condiciones extremas de sismo, sin combinar otros tipos de solicitaciones. En este contexto, las combinaciones de cargas presentadas están diseñadas específicamente para capturar las demandas máximas derivadas de las cargas gravitacionales y sísmicas, asegurando que la estructura cumpla con los criterios de seguridad y desempeño sísmico establecidos.

Por lo tanto, la omisión de combinaciones de carga de viento es adecuada y justificada, dado que el método de coeficientes de desplazamiento no las contempla ni las requiere para su aplicación, priorizando el análisis sísmico sobre otras solicitaciones. En caso de que las condiciones del proyecto o la normativa local exigieran considerar el viento, se habrían definido combinaciones adicionales específicas para ese propósito, lo cual no aplica en este caso.

## 4.2 Modelación de la estructura

En esta etapa se procedió a realizar el modelado virtual de la topología seleccionada en el software especializado SAP 2000, en el cual se realizó el análisis sísmico y estructural bajo las condiciones de carga propuestas en la sección 4.1.

### 4.2.1 Modelación y análisis de la topología seleccionada en SAP 2000

En esta etapa se llevó a cabo la modelación virtual de la estructura seleccionada utilizando el software SAP 2000, versión 22, elegido por su capacidad

para realizar análisis sísmicos y estructurales complejos. Se configuró el software para emplear el método de elementos finitos, asumiendo condiciones de empotramiento en la base de las columnas y modelando vigas y columnas como elementos *frame* con propiedades de sección correspondientes a perfiles W (Wide Flange) del AISC 360-16.

Previamente, en la etapa 1, se seleccionó una topología de marcos rígidos debido a su eficiencia para resistir cargas laterales en zonas sísmicas, consistente en un edificio de 10 niveles, con una altura total de 40 metros y dimensiones en planta de 20 m x 30 m, con una crujía en dirección X y dos en dirección Y.

Los perfiles iniciales se seleccionaron aleatoriamente dentro de rangos adecuados, verificándose mediante los criterios de diseño del software, que incluyen resistencia, estabilidad lateral y deformaciones máximas según la norma ASCE 7-16. Las cargas consideradas incluyeron cargas muertas y cargas vivas según NTC para uso habitacional. Para el análisis sísmico, se utilizó el espectro de respuesta correspondiente a la Ciudad de México.

Se realizaron análisis estáticos equivalentes para verificar esfuerzos y deformaciones, complementados con análisis modales para determinar periodos y modos de vibración, y un análisis Pushover para evaluar la capacidad sísmica no lineal.

## 4.3 Implementación del Algoritmo Genético partiendo del diseño por desempeño

### 4.3.1 Ejecución de análisis *Pushover* de la estructura.

En esta subsección, se detalla la ejecución del análisis Pushover para evaluar la respuesta sísmica no lineal de la estructura modelada. El análisis Pushover es un procedimiento estático no lineal que aplica cargas laterales crecientes, simulando fuerzas sísmicas, hasta alcanzar un desplazamiento objetivo

o la inestabilidad estructural. Este método permite construir la curva de capacidad (cortante basal vs. desplazamiento), proporcionando información sobre la secuencia de fluencia, ductilidad y capacidad global de la estructura frente a sismos.

El análisis se realizó utilizando SAP 2000, configurado para modelar el comportamiento no lineal mediante articulaciones plásticas asignadas en los extremos de vigas y columnas, donde se espera la formación de regiones plásticas.

Para capturar la influencia de los modos de vibración, se realizaron múltiples análisis Pushover, cada uno con un patrón de carga proporcional a uno de los primeros tres modos de vibración, obtenidos de un análisis modal previo en SAP 2000. Los patrones de carga se definieron en el módulo *"Define* > *Load Patterns* > *Pushover"*, seleccionando la opción "*Mode*" para distribuir las fuerzas laterales según las formas modales. Este enfoque multimodal, inspirado en métodos como el análisis Pushover modal (MPA) de Chopra (2001), permite evaluar la contribución de modos superiores, relevantes en edificios de 10 niveles como el modelado.

En cada análisis, se monitorearon la cortante basal, calculada como la suma de las reacciones en los apoyos de la base, y el desplazamiento en un nodo de control ubicado en el centro del nivel superior (techo). Los análisis se configuraron con un número suficiente de pasos (por ejemplo, 100 pasos) y criterios de convergencia estrictos para garantizar la precisión de los resultados no lineales.

Los resultados de cada análisis se utilizaron para construir curvas de capacidad, graficando la cortante basal frente al desplazamiento del techo para cada modo de vibración.

Posteriormente, se aplicó el método del coeficiente de desplazamiento, según FEMA 440, para determinar el desplazamiento objetivo correspondiente a los niveles de desempeño: Ocupación Inmediata, Seguridad y Prevención de Colapso. Este método calcula coeficientes (C0, C1, C2, C3) que ajustan el desplazamiento elástico para considerar el comportamiento no lineal, efectos de modos superiores

y efectos P-Δ. En este estudio, el método se aplicó principalmente a la curva de capacidad del primer modo, siguiendo la práctica estándar, aunque las curvas de modos superiores se analizaron para comprender su contribución a la respuesta sísmica.

Durante el análisis, se asumieron uniones rígidas en las conexiones viga-columna y se simplificaron efectos secundarios como la interacción suelo-estructura, lo que podría limitar la precisión en ciertos casos. Estas suposiciones se justifican por el enfoque en el comportamiento global de la estructura.

### 4.3.2 Determinación del nivel de desempeño.

El método empleado para que la estructura entrara en un nivel de clasificación fue el de “Coeficientes de desplazamiento” que, por su nombre, evalúa coeficientes para estimar el desplazamiento máximo de un edificio durante un evento sisimico y esta descrito en la sección 2.1.8, de este modo el resultado es el nivel de desempeño que indica si los usuarios de la edificación pueden continuar con sus actividades diarias sin sufrir riesgo o no.

Los valores de los coeficientes se determinaron siguiendo las especificaciones de FEMA 440, considerando las características de la estructura (por ejemplo, altura de 35 m, 10 niveles) y el sismo de diseño definido en la sección 4.1. El periodo efectivo, Te, se obtuvo mediante la bilinearización de la curva de capacidad derivada del análisis Pushover, ajustando el período inicial según la ductilidad de la estructura.

Una vez calculado el desplazamiento objetivo, se evaluó la respuesta de la estructura en ese punto utilizando la curva de capacidad obtenida del análisis Pushover descrito en la sección 4.3.1. Se verificó si la estructura cumple con los criterios de aceptación para el nivel de desempeño deseado, cabe mencionar que para esta investigación se buscó que la estructura cumpla con el nivel de ocupación inmediata.

### 4.3.3 Definición de la función objetivo

La función objetivo estableció la minimización del desplazamiento máximo de la estructura para alinear los objetivos del diseño por desempeño, buscando reducir las deformaciones laterales bajo cargas sísmicas, lo que mejora la estabilidad y reduce el riesgo de daños estructurales importantes.

Partiendo del uso de Algoritmos Genéticos, cada individuo es una representación de una configuración o selección de perfiles estructurales asignados a los diferentes grupos contenidos en la estructura. Esta asignación permitió evaluar cómo diferentes combinaciones de perfiles afectaban el desplazamiento máximo, facilitando la comparación entre soluciones.

Se establecieron varias restricciones para garantizar que los perfiles seleccionados fueran viables y cumplieran con los criterios de diseño. Una de las restricciones principales fue que las áreas de los perfiles debían disminuir de acuerdo con la ubicación del grupo, es decir, los perfiles en niveles superiores (grupos más altos) debían tener áreas menores que los de niveles inferiores, reflejando la reducción de cargas en la parte superior de la estructura. Esto es común en diseño estructural, ya que las columnas y vigas en niveles superiores soportan menos carga y pueden ser más ligeras.

Otra condición importante fue que los perfiles iniciales se propusieron con la relación de la longitud del claro entre 20, asegurando un peralte de perfil adecuado evitando perfiles excesivamente altos que podrían comprometer la estabilidad o la economía del diseño. Esta condición se verificó en las iteraciones iniciales, asegurando que los perfiles cumplieran con criterios de proporcionalidad y funcionalidad.

### 4.3.4 Implementación y codificación del Algoritmo Genético.

En esta sección se describe el desarrollo y la implementación de dos herramientas clave: TESTSG y AG, herramientas que empleando el lenguaje de programación Python realizan tareas específicas, las cuales se describen a continuación.

#### *4.3.4.1 Herramienta "TESTSG"*

Es una herramienta diseñada para automatizar la modelación, análisis y extracción de datos de una estructura realizada en SAP2000, su objetivo es evaluar el comportamiento sisimico utilizando el método de coeficientes de desplazamiento (DCM), que es un método del diseño por desempeño sísmico. El código fuente se puede visualizar en la sección 8 (Apéndice) - Herramienta "TESTSG".

TESTSG tiene dos funciones principales:

- Permite generar un modelo estructural desde cero en SAP2000, asignar materiales, secciones, cargas, patrones de carga, espectros de diseño y combinaciones de carga. Luego realiza análisis estáticos y no lineales (como *Pushover*) para obtener datos necesarios para calcular gráficas de esfuerzo-deformación y determinar el punto de desempeño sísmico mediante el método de coeficientes de desplazamiento.
- Revisar un modelo ya creado en SAP2000 y realizar los mismos análisis (*Pushover*, gráficas de esfuerzo-deformación, cálculo del punto de desempeño, etc.) para evaluar su comportamiento sísmico.

I. Configuración inicial del modelo estructural:

A. Se establecieron las bases para interactuar con el software de diseño estructural mediante un archivo de ayuda de HTML compilado de SAP2000.

B. Se definieron las unidades del modelo (fuerza, longitud, temperatura) para garantizar consistencia en los cálculos.

C. Se importaron materiales estructurales básicos (acero y concreto) y se añadieron bases de datos de secciones estándares para su uso en el diseño.

II. Definición de patrones de carga y espectros de diseño:

A. Se definieron patrones de carga muerta, viva, sísmicas (en direcciones X e Y).

B. Se incorporó un espectro de diseño sísmico basado en datos de una ubicación específica (Ciudad de México).

C. Se crearon combinaciones de carga para análisis gravitacional y sísmico, incluyendo una combinación envolvente para considerar los casos más desfavorables.

III. Modelado geométrico de la estructura:

A. Se generaron elementos estructurales basados en las dimensiones del edificio (número de niveles, altura entre pisos, número de crujías y longitud de crujías).

B. Se organizaron los elementos estructurales en grupos cada 4 niveles (vigas, columnas y losas) para facilitar la asignación de propiedades y el análisis por niveles.

IV. Asignación de propiedades iniciales a los elementos estructurales:

A. Se estimaron secciones iniciales para vigas y columnas utilizando criterios preliminares basados en las dimensiones de la estructura y las cargas definidas.

B. Las secciones se seleccionaron de una base de datos predefinida, asegurando que cumplieran con restricciones iniciales como límites de área y peralte.

V. Análisis estructural y *Pushover*:

A. Se configuró un análisis estático no lineal (*Pushover*) para evaluar la respuesta de la estructura bajo cargas sísmicas.

B. Se identificó el nodo de monitorización de desplazamientos y fuerzas durante el análisis.

C. Se procesaron los datos obtenidos para generar curvas de capacidad (desplazamiento vs cortante) y se derivó una representación bilineal para determinar parámetros como la rigidez inicial, la rigidez secundaria y la fuerza de fluencia.

VI. Aplicación del método de coeficientes de desplazamiento:

A. Se calcularon parámetros fundamentales como el peso total de la estructura, el período inicial y el período efectivo, utilizando los resultados del análisis *Pushover*.

B. Se determinaron los coeficientes del método (C0, C1, C2, C3) según las especificaciones de FEMA 440, considerando tres niveles de desempeño: "Ocupación Inmediata", "Seguridad" y "Prevención al Colapso".

C. Se estimó el desplazamiento objetivo para cada nivel de desempeño mediante la combinación de los coeficientes y el desplazamiento elástico, siguiendo las ecuaciones del método.

VII. Preparación de datos para la optimización:

A. Se extrajeron métricas clave como el desplazamiento máximo de la estructura y los desplazamientos objetivo para cada nivel de desempeño.

B. Se organizaron los datos en forma de tablas y gráficas para su uso posterior en el proceso de optimización.

#### *4.3.4.2 Herramienta “AG”*

Se desarrolló un Algoritmo Genético para optimizar las secciones de vigas y columnas de la estructura, utilizando el método de coeficientes de desplazamiento como criterio de evaluación del desempeño sísmico. El código fuente se puede visualizar en la sección 8 (Apéndice) - Herramienta “AG”.

I. Definición de la población inicial:

A. Se estableció un conjunto inicial de soluciones posibles (población), donde cada solución representaba una combinación de secciones de vigas y columnas para los diferentes grupos de niveles del edificio.

B. Se aseguraron restricciones iniciales, como que las áreas de las secciones fueran decrecientes entre grupos de niveles (por ejemplo, secciones más grandes en niveles inferiores y más pequeñas en niveles superiores).

II. Definición de la función objetivo:

A. Se diseñó un criterio de evaluación (función objetivo) para medir la calidad de cada solución.

B. El objetivo principal fue minimizar el desplazamiento máximo de la estructura bajo cargas sísmicas.

C. Se incorporaron penalizaciones para soluciones que no cumplieran con restricciones prácticas, como áreas no decrecientes entre grupos o áreas excesivamente grandes que resultaran en diseños poco realistas.

D. Se utilizó el desplazamiento objetivo del nivel de desempeño más exigente ("Ocupación Inmediata") como referencia para penalizar soluciones que excedieran este límite.

III. Selección de individuos para la reproducción:

A. Se implementó un método de selección basado en torneos, donde se comparaban subgrupos de soluciones y se elegían las mejores (con menor aptitud) para generar nuevas soluciones.

IV. Aplicación de cruce y mutación:

A. Se desarrolló un mecanismo de cruce para combinar características de dos soluciones seleccionadas (padres) y generar nuevas soluciones (hijos). El cruce se diseñó para preservar las restricciones de áreas decrecientes.

B. Se implementó un operador de mutación para introducir variaciones aleatorias en las soluciones, permitiendo explorar nuevas combinaciones de secciones dentro de los límites establecidos.

C. Las mutaciones se controlaron para evitar que las áreas de las secciones violaran las restricciones de diseño (como áreas decrecientes entre grupos).

V. Generación iterativa de nuevas poblaciones:

A. Se estableció un proceso iterativo donde, en cada generación, se creaba una nueva población combinando las mejores soluciones de la generación anterior (elitismo), los hijos generados por cruce y mutación, y algunas soluciones aleatorias para completar el tamaño de la población.

B. Este proceso se repitió por un número predefinido de generaciones, ajustado para balancear la convergencia con el tiempo computacional.

VI. Evaluación del desempeño sísmico:

A. Para cada solución generada, se evaluó su comportamiento sísmico utilizando los datos obtenidos del análisis estructural y el método de coeficientes de desplazamiento.

B. Se comparó el desplazamiento máximo de la estructura con los desplazamientos objetivo para los diferentes niveles de desempeño, priorizando el cumplimiento del nivel más exigente ("Ocupación Inmediata").

VII. Selección de la solución óptima:

A. Al finalizar el proceso iterativo, se identificó la solución con la mejor aptitud que cumpliera con las restricciones de diseño (áreas decrecientes, límites de área, etc.).

B. Se verificó el nivel de desempeño alcanzado por esta solución (por ejemplo, "Ocupación Inmediata", "Seguridad" o "Prevención al Colapso") para determinar su viabilidad según los criterios de diseño sísmico.

#### *4.3.4.3 Integración de ambas herramientas*

En esta etapa se diseñó un flujo de trabajo integrado para combinar el análisis estructural con la optimización mediante el Algoritmo Genético.

I. Interacción entre análisis y optimización:

A. El proceso de análisis estructural se utilizó como base para evaluar cada solución propuesta por el Algoritmo Genético.

B. Para cada combinación de secciones propuesta, se realizó un análisis *Pushover* y se aplicó el método de coeficientes de desplazamiento para determinar su desempeño sísmico.

II. Iteración continua:

A. El Algoritmo Genético propuso nuevas combinaciones de secciones en cada iteración, las cuales fueron evaluadas mediante el análisis estructural.

B. Los resultados del análisis (desplazamiento máximo, nivel de desempeño alcanzado) se retroalimentaron al Algoritmo Genético para guiar la búsqueda hacia soluciones óptimas.

III. Optimización guiada por el desempeño sísmico:

A. El proceso iterativo se diseñó para buscar una solución que minimizara el desplazamiento máximo mientras cumpliera con los criterios de diseño sísmico establecidos por el método de coeficientes de desplazamiento.

B. Se priorizó el cumplimiento del nivel de desempeño más exigente, ajustando las secciones estructurales para lograr un diseño eficiente y seguro.

## 4.4 Comparación de los resultados obtenidos

### 4.4.1 Ejecución de análisis de la estructura seleccionada con métodos tradicionales.

### 4.4.2 Comparación del modelo general con la implementación del Algoritmo Genético y la metodología tradicional.

En esta sección la eficiencia de optimización será el factor de comparación mediante el tiempo de ejecución y la cantidad de estructuras analizadas necesarias para la reducir desplazamientos máximos y el peso de la estructura.

# 5 RESULTADOS Y DISCUSIÓN

## 5.1 Estructura seleccionada

### 5.1.1 Estructura tipo

Se propuso una estructura tipo edificio vertical de 10 niveles, caracterizada por una geometría regular y simétrica, con una disposición de dos crujías en la dirección X y dos crujías en la dirección Y. Esta topología resultó en un modelo estructural formando dimensiones uniformes, lo que permitió un análisis eficiente del comportamiento sísmico en ambas direcciones (X e Y). La estructura se diseñó con marcos rígidos, integrando columnas y vigas distribuidas uniformemente en planta y elevación, ( Figura 5.1). Este diseño geométrico regular minimizó efectos de irregularidad vertical, facilitando la evaluación posterior bajo cargas sísmicas.

## 5.2 Modelo de la estructura

### 5.2.1 Modelo y análisis de la topología seleccionada en SAP 2000

El modelo estructural se configuró en SAP 2000 para simular las cargas muertas, vivas y sísmicas determinadas en la sección 5.1, aplicando el espectro de diseño de Ciudad de México obtenido de Prodisis.

Los perfiles estructurales iniciales se asignaron de manera aleatoria, creándose grupos cada 4 niveles como se aprecia en la Figura 5.2, asegurando que cumplieran con los criterios de diseño de acero del software SAP 2000.

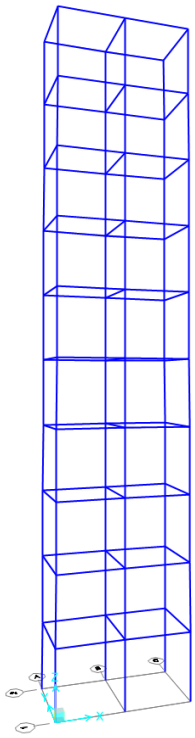

**Figura 5.1. Estructura modelada en SAP 2000.**

Los perfiles seleccionados incluyeron secciones W (*Wide Flange*) de acero, distribuidos como sigue en las columnas y vigas:

- Columnas: Se asignaron perfiles como W6X12, W14X74, W14X90, W14X109, W14X120, W14X132, W14X145, W14X159, W14X176, W14X193, W14X211, W14X233, W14X257 y W14X283, con áreas y resistencias variando según la ubicación en los niveles superiores e inferiores, siguiendo la restricción de áreas decrecientes en niveles más altos.
- Vigas: Se asignaron perfiles similares dentro de la serie W, como W6X12, distribuidos uniformemente para mantener la rigidez del sistema de marcos rígidos.

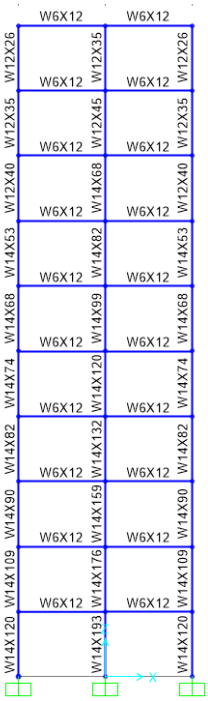


**Figura 5.2. Distribución de perfiles estructúrales en modelación.**

Como resultado del análisis de comportamiento sísmico implementado en SAP 2000, se obtuvieron los valores máximos de las fuerzas y deformaciones en los elementos estructurales, representados gráficamente en la Figura 5.3. Estos resultados incluyeron:

- Fuerza axial máxima: Se registró un valor de 125.54 toneladas en las columnas, indicando la carga máxima soportada en compresión, distribuida principalmente en las columnas de los niveles inferiores debido a la acumulación de cargas gravitacionales y sísmicas.
- Fuerza cortante basal: Se obtuvo un valor de 59 toneladas, reflejando las fuerzas laterales causadas por el sismo, distribuidas a lo largo de la altura del edificio.
- Momento máximo: Se obtuvo un valor de 2.26 ton-m, presente en las conexiones entre vigas y columnas, así como en los elementos sometidos a flexión.

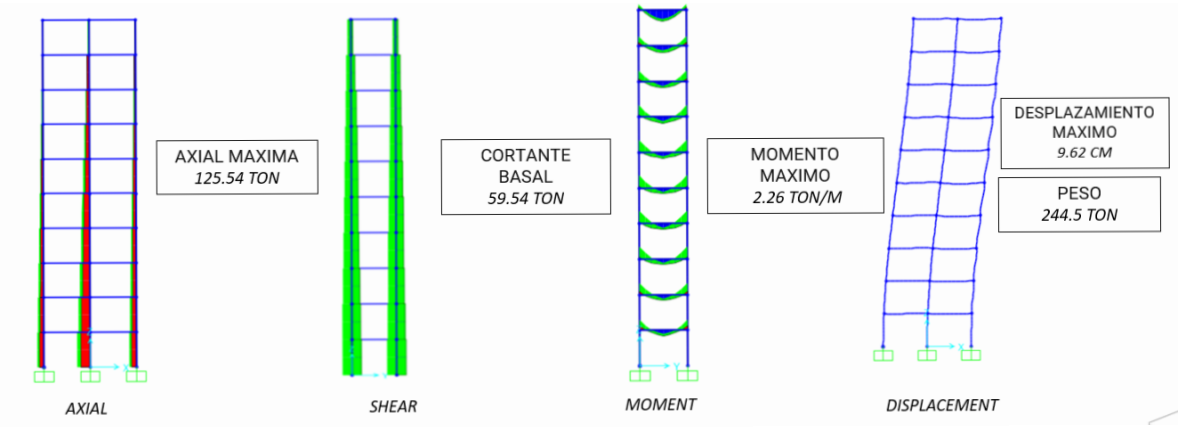


**Figura 5.3. Elementos mecánicos de la modelación.**

Desplazamiento máximo: Se registró un valor de 9.62 cm en la parte superior de la estructura, bajo la acción sísmica, representando la deformación lateral máxima en el último nodo del edificio, un indicador clave del desempeño sísmico inicial.

## 5.3 Algoritmo Genético basado en diseño por desempeño

### 5.3.1 Herramienta “TESTSG”.

I. Inicialización y configuración del modelo en SAP2000:

A. Utiliza la API de SAP2000 (*comtypes.client*) para interactuar con el software.

B. Abre un archivo .sdb existente o permite trabajar con un modelo en blanco.

C. Establece las unidades del modelo (Ton, m, °C).

D. Define materiales como "ACERO" y "CONCRETO", e importa secciones de una base de datos (AISC15.xml) para asignarlas a elementos estructurales.

II. Definición de patrones de carga y espectros de diseño:

A. Agrega patrones de carga como "DEAD" (carga muerta), "SISMO X", "SISMO Y" (cargas sísmicas), y "VIVA" (carga viva).

B. Importa un espectro de diseño desde un archivo de texto (ESPECTRO CIUDAD DE MEXICO.txt) y lo asigna a los casos de respuesta espectral ("SISMO X" y "SISMO Y").

C. Crea combinaciones de carga como "1.3D + 1.5V" (para carga gravitacional), "COMB SISMO X" y "COMB SISMO Y" (para carga sísmica), y una combinación envolvente ("ENVOLVENTE").

III. Modelado geométrico de la estructura:

A. Genera coordenadas para los nodos de la estructura basándose en las dimensiones proporcionadas (número de niveles, altura entre pisos, número de crujías, longitud de crujías).

B. Define grupos para vigas (VG), columnas (CG), y losas (LOSAS) para facilitar la asignación de propiedades y análisis.

IV. Asignación de secciones iniciales:

A. Utiliza la función *calcular_perfiles_iniciales* para estimar secciones iniciales de vigas y columnas basadas en las dimensiones de la estructura y las cargas aproximadas.

B. Las secciones se seleccionan de una base de datos (DB.xlsx) considerando criterios como el área de la sección y el peralte (profundidad).

V. Análisis estructural y *Pushover*:

A. Realiza un análisis estático no lineal (*Pushover*) utilizando el caso "INELASTICO".

B. Extrae datos de desplazamientos y fuerzas en puntos clave (como el punto de control *point_name*) para generar curvas de capacidad (*Pushover*).

C. Calcula una representación bilineal de la curva *Pushover* para obtener parámetros como la rigidez inicial (Ki), la rigidez secundaria (Ks), y la fuerza de fluencia (Vy).

VI. Cálculo del punto de desempeño mediante el método de coeficientes de desplazamiento:

A. Determina parámetros clave como el peso de la estructura (W), los períodos iniciales (Ti) y efectivo (Te), y la aceleración espectral (Sa).

B. Calcula los coeficientes del método de coeficientes de desplazamiento (C0, C1, C2, C3) según FEMA 440, considerando diferentes niveles de desempeño ("Ocupación Inmediata", "Seguridad", "Prevención al Colapso").

C. Obtiene el desplazamiento objetivo (*delta_d*) para cada nivel de desempeño.

D. Compara el desplazamiento máximo (*max_u1*) de la estructura con los desplazamientos objetivo para determinar si cumple con los niveles de desempeño.

VII. Generación de gráficas y resultados:

A. Genera gráficas de la curva *Pushover* y su representación bilineal, mostrando desplazamientos y fuerzas.

B. Produce tablas con los desplazamientos objetivo para cada nivel de desempeño.

C. Devuelve métricas como *max_u1* (desplazamiento máximo), (desplazamiento objetivo para "Ocupación Inmediata"), y el archivo de la gráfica (*plot_path*).

VIII. Relación con el método de coeficientes de desplazamiento.

TESTSG implementa el método de coeficientes de desplazamiento para determinar el punto de desempeño sísmico de la estructura. Este método, basado en FEMA 440, estima el desplazamiento objetivo (*delta_d*) mediante la ecuación 2.

TESTSG calcula estos coeficientes para tres niveles de desempeño y compara el desplazamiento máximo (*max_u1*) con los desplazamientos objetivo para determinar si la estructura cumple con los criterios de diseño sísmico.

### 5.3.2 Herramienta “AG”

I. Inicialización de la población (*crear_poblacion_inicial*):

A. Tamaño de la población: Se define un tamaño inicial (por defecto, 30 individuos).

B. Individuos: Cada individuo representa una solución potencial y está compuesto por lista de vigas y columnas.

C. Generación inicial: Las secciones iniciales se generan utilizando la función *calcular_perfiles_iniciales* de TESTSG, que proporciona una estimación basada en las dimensiones y cargas. Luego, se generan variaciones aleatorias asegurando que las áreas de las secciones sean decrecientes entre grupos (por ejemplo, el área de la viga del grupo 1 debe ser menor que la del grupo 2).

D. Restricciones: Se aplican límites iniciales al peralte de las secciones (por ejemplo, máximo 30 in para vigas y 50 in para columnas en el grupo 1) para garantizar soluciones factibles.

II. Evaluación de aptitud (*evaluar_aptitud*):

A. Objetivo principal: Minimizar el desplazamiento máximo (*max_u1*) de la estructura mientras se asegura que cumpla con los niveles de desempeño sísmico.

B. Proceso:

Cada individuo (combinación de secciones de vigas y columnas) se pasa a TESTSG mediante la función *ejecutar_analisis*.

TESTSG realiza un análisis *Pushover* y calcula el desplazamiento máximo (*max_u1*) y el desplazamiento objetivo (*delta_d1*) para el nivel de "Ocupación Inmediata" y la función de aptitud se define como: Aptitud = *max_u1_cm* (desplazamiento en centímetros).

III. Penalizaciones.

A. Si *max_u1* excede *delta_d1*, se agrega una penalización proporcional a la diferencia:

(*max_u1*_cm − *delta_d1*_cm)(50)

(*max_u1*_cm−*delta_d1*_cm)(50).

B. Si las áreas de las secciones no son estrictamente decrecientes entre grupos, se agrega una penalización de 5000.

C. Se penalizan áreas excesivamente grandes (por ejemplo, áreas mayores a 50 in²) para evitar secciones poco prácticas.

D. Salida: La función devuelve la aptitud, el desplazamiento máximo (*max_u1*), el desplazamiento objetivo (*delta_d1*), el tiempo de evaluación, el archivo de la gráfica *Pushover*, y una tabla con los desplazamientos objetivo para cada nivel de desempeño.

IV. Selección de padres (*seleccionar_padres*):

A. Utiliza un método de selección por torneo: se eligen aleatoriamente *tamano_torneo* individuos (por defecto 3), y se selecciona el de menor aptitud.

B. Se seleccionan *num_padres* individuos (la mitad del tamaño de la población) para generar la siguiente generación.

V. Cruce:

A. Se generan dos hijos a partir de dos padres mediante un cruce uniforme.

B. Para cada grupo de niveles, se elige aleatoriamente (con probabilidad 0.7) una sección de viga o columna de uno de los padres.

C. Se asegura que las áreas de las secciones sean decrecientes entre grupos. Si una sección seleccionada viola esta restricción, se elige una sección válida de la base de datos (secciones).

VI. Mutación:

A. Cada sección de un individuo tiene una probabilidad (*tasa_mutacion*, por defecto 0.2) de mutar.
B. Si se decide mutar, se elige una nueva sección de la base de datos, asegurando que cumpla con la restricción de áreas decrecientes.
C. Para el grupo 1, no se aplican restricciones adicionales; para grupos posteriores, la nueva sección debe tener un área menor que la del grupo anterior.

VII. Generación de nuevas poblaciones:

A. Se crea una nueva generación
B. Se selecciona al mejor individuo de la generación anterior (elitismo).
C. Se generan hijos mediante cruce y mutación.
D. Si faltan individuos para completar el tamaño de la población, se seleccionan aleatoriamente individuos de la generación anterior.
E. El proceso se repite por un número definido de generaciones (*num_generaciones*, por defecto 30).

VIII. Convergencia y selección del mejor individuo:

A. Al final de todas las generaciones, se selecciona el individuo con la menor aptitud que cumpla con la restricción de áreas decrecientes.
B. Se evalúa si el desplazamiento máximo (*max_u1*) es menor o igual al desplazamiento objetivo (*delta_d1*) para el nivel de "Ocupación Inmediata". También se verifica si cumple con otros niveles de desempeño ("Seguridad", "Prevención al Colapso").

IX. Generación de resultados y visualización:

A. Tablas: Genera tablas con los perfiles iniciales, los perfiles del mejor diseño, y los desplazamientos objetivo para cada nivel de desempeño.
B. Gráficas: Produce gráficas de la evolución del desplazamiento máximo (*max_u1*) y las áreas promedio de vigas y columnas por generación.
C. Exportación a Excel: Guarda los resultados en un archivo Excel (*max_u1*_evolution.xlsx), incluyendo gráficas y tablas.

X. Relación con el método de coeficientes de desplazamiento

El Algoritmo Genético utiliza los resultados del método de coeficientes de desplazamiento (calculados por TESTSG) para evaluar la aptitud de cada individuo. Específicamente:

A. Función de aptitud: El desplazamiento máximo (*max_u1*) se compara con el desplazamiento objetivo (*delta_d1*) para el nivel de "Ocupación Inmediata". El objetivo es minimizar *max_u1* y asegurar que sea menor o igual a *delta_d1*.
B. Niveles de desempeño: AG también considera otros niveles de desempeño ("Seguridad", "Prevención al Colapso") al reportar los resultados finales, verificando en qué nivel de desempeño encaja el mejor diseño.
C. Optimización: El Algoritmo Genético ajusta las secciones de vigas y columnas iterativamente para encontrar una configuración que cumpla con los criterios de diseño sísmico establecidos por el método de coeficientes de desplazamiento.

### 5.3.3 Integración entre TESTSG y AG

I. TESTSG proporciona los datos necesarios:
   A. TESTSG realiza el análisis estructural y calcula métricas clave como *max_u1* y *delta_d1* para cada combinación de secciones propuesta por AG.
   B. También genera gráficas y tablas que AG utiliza para reportar resultados.

II. AG optimiza las secciones:
   A. AG propone diferentes combinaciones de secciones (individuos) y las evalúa utilizando TESTSG.
   B. Basado en los resultados de TESTSG (aptitud, desplazamientos, cumplimiento de niveles de desempeño), AG ajusta las secciones mediante operadores genéticos (selección, cruce, mutación) para encontrar una solución óptima.

III. Iteración hasta la convergencia:

Este proceso iterativo continúa hasta que AG encuentra un diseño que minimiza *max_u1*, cumple con los niveles de desempeño sísmico, y satisface las restricciones prácticas (áreas decrecientes, límites de área).

### 5.3.4 Resultados generales

Se realizaron tres simulaciones del Algoritmo Genético para optimizar el diseño estructural: una con 20 generaciones y 20 individuos (Simulación 1), otra con 30 generaciones y 30 individuos (Simulación 2), y una tercera con 50 generaciones y 50 individuos (Simulación 3).

Además, se incluyó el cálculo para el porcentaje de mejora en lo que se refiere al desplazamiento máximo de la estructura propuesta al ser comparado con

el individuo más optimo de cada una de las simulaciones, el cálculo es representado por la ecuación 7.

$$\eta = \frac{max\_u1_{base} - max\_u1_{optimizada}}{max\_u1_{base}} x100 \qquad (7)$$

Simulación 1 (20 generaciones, 20 individuos)

La Simulación 1 evaluó 440 configuraciones y 23 perfiles distintos. La Tabla 5 compara los perfiles iniciales con los del mejor diseño optimizado.

**Tabla 5. Comparación de perfiles iniciales y optimizados (Simulación 1).**

| Grupo | Pefiles iniciales | Área ($in^2$) | d (in) | Perfiles Optimizados | Desplazamiento (cm) | Desplazamiento Objetivo (Ocupación Inmediata, cm) |
|---|---|---|---|---|---|---|
| Grupo 1 (Niveles 0-3) | Viga: W6X20 | 5.87 | 6.2 | Viga: W10X49 | 8.95 | 9.35 |
| | Columna: W10X112 | 32.9 | 11.4 | Columna: W18X119 | | |
| Grupo 2 (Niveles 4-7) | Viga: W5X19 | 5.56 | 5.15 | Viga: W21X48 | | |
| | Columna: W10X88 | 26 | 10.8 | Columna: W10X77 | | |
| Grupo 3 (Niveles 8-9) | Viga: W5X16 | 4.71 | 5.01 | Viga: W12X35 | | |
| | Columna: W8X67 | 19.7 | 9 | Columna: W10X49 | | |

El desplazamiento máximo del mejor diseño fue de 8.35 cm, cumpliendo con el nivel de "Ocupación Inmediata" (desplazamiento objetivo de 9.35 cm). Comparado con el valor inicial de 9.62 cm, esto representa una reducción del 13.2%. El peso del diseño optimizado fue de 204.46 ton. La evolución de max_u1 se muestra en la Figura 5.4, donde el desplazamiento del mejor individuo disminuyó de 10.5 cm a 8.35 cm, y el promedio de la población se estabilizó alrededor de 9 cm.

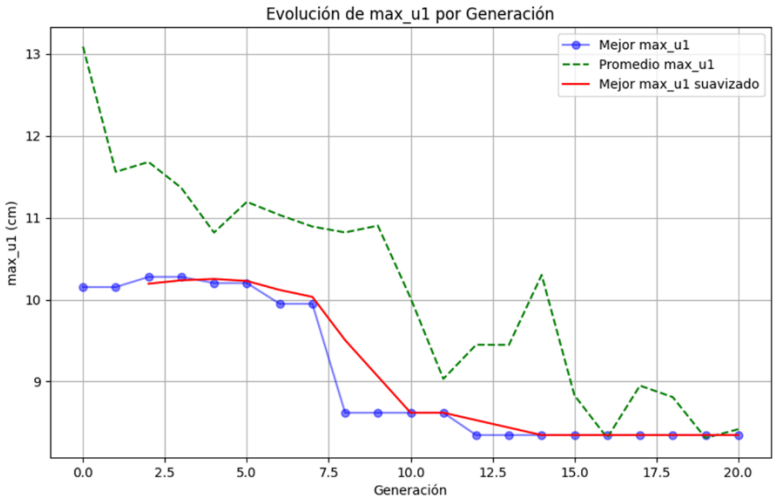


**Figura 5.4. Evolución de desplazamiento por generación en Simulación 1.**

Simulación 2 (30 generaciones, 30 individuos)

La Simulación 2 evaluó 960 configuraciones y 28 perfiles distintos. La Tabla 6 presenta los resultados del mejor diseño optimizado.

**Tabla 6. Comparación de perfiles iniciales y optimizados en Simulación 2.**

| Grupo | Pefiles iniciales | Área ($in^2$) | d (in) | Perfiles Optimizados | Desplazamiento (cm) | Desplazamiento Objetivo (Ocupación Inmediata, cm) |
|---|---|---|---|---|---|---|
| Grupo 1 (Niveles 0-3) | Viga: W6X20 | 5.87 | 6.2 | Viga: W4X13 | 9.89 | 10.62 |
| | Columna: W10X112 | 32.9 | 11.4 | Columna: W30X124 | | |
| Grupo 2 (Niveles 4-7) | Viga: W5X19 | 5.56 | 5.15 | Viga: W10X12 | | |
| | Columna: W10X88 | 26 | 10.8 | Columna: W16X89 | | |
| Grupo 3 (Niveles 8-9) | Viga: W5X16 | 4.71 | 5.01 | Viga: W8X10 | | |
| | Columna: W8X67 | 19.7 | 9 | Columna: W12X58 | | |

El desplazamiento máximo del mejor diseño fue de 9.89 cm, cumpliendo con el nivel de "Ocupación Inmediata" (desplazamiento objetivo de 10.62 cm). Comparado con el valor inicial de 9.62 cm, el desplazamiento aumentó ligeramente (2.8%), pero el peso se redujo a 193.4 ton, un 5.4% menos que en la Simulación 1 (204.46 ton). La Figura 5.5 muestra la evolución de max_u1 con una disminución

del mejor individuo de 10.5 cm a 9.89 cm, y un promedio de la población que fluctúa alrededor de 10 cm.

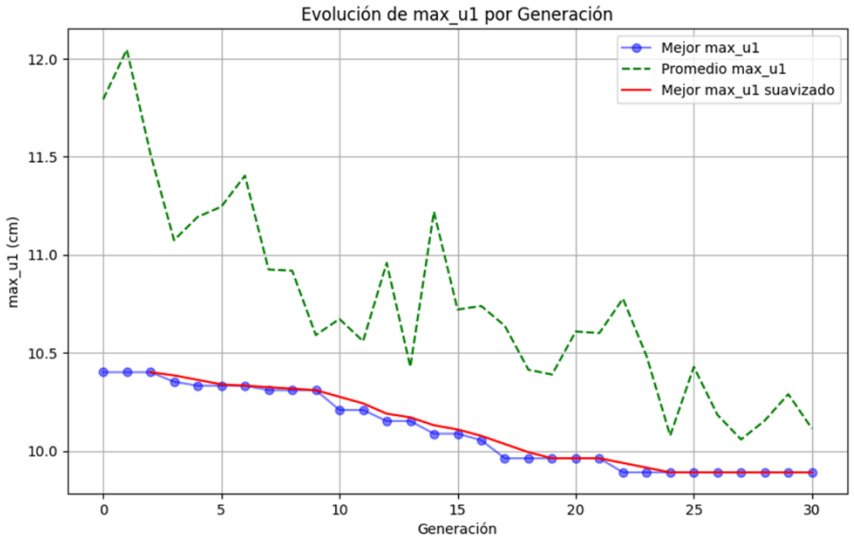


**Figura 5.5. Evolución de desplazamiento por generación en Simulación 2.**

Simulación 3 (50 generaciones, 50 individuos)

La Simulación 3 evaluó 2600 configuraciones y 63 perfiles distintos. La Tabla 7 presenta los resultados del mejor diseño optimizado.

**Tabla 7. Comparación de perfiles iniciales y optimizados (Simulación 3).**

| Grupo | Pefiles iniciales | Área ($in^2$) | d (in) | Perfiles Optimizados | Desplazamiento (cm) | Desplazamiento Objetivo (Ocupación Inmediata, cm) |
|---|---|---|---|---|---|---|
| Grupo 1 (Niveles 0-3) | Viga: W6X20 | 5.87 | 6.2 | Viga: W18X97 | 5.77 | 7.66 |
| | Columna: W10X112 | 32.9 | 11.4 | Columna: W14X145 | | |
| Grupo 2 (Niveles 4-7) | Viga: W5X19 | 5.56 | 5.15 | Viga: W10X60 | | |
| | Columna: W10X88 | 26 | 10.8 | Columna: W14X109 | | |
| Grupo 3 (Niveles 8-9) | Viga: W5X16 | 4.71 | 5.01 | Viga: W18X46 | | |
| | Columna: W8X67 | 19.7 | 9 | Columna: W12X65 | | |

El desplazamiento máximo del mejor diseño fue de 5.77 cm, cumpliendo con el nivel de "Ocupación Inmediata" (desplazamiento objetivo de 7.66 cm). Comparado con el valor inicial de 9.62 cm, esto representa una reducción del 40%,

la mayor entre las tres simulaciones. Sin embargo, el peso aumentó a 224.76 ton, un 16.2% más que en la Simulación 2 (193.4 ton) y un 9.9% más que en la Simulación 1 (204.46 ton). La  Figura 2.2 muestra la evolución de max_u1 con una disminución del mejor individuo de 9.5 cm a 5.77 cm. Sin embargo, entre las generaciones 20 y 40, se observa un comportamiento errático, con fluctuaciones significativas en el mejor y promedio max_u1.

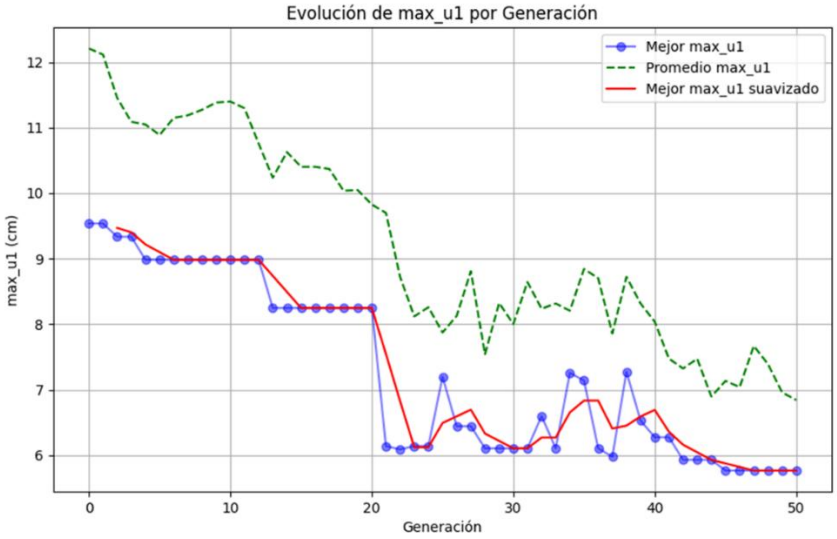


**Figura 5.6. Evolución de desplazamiento por generación en simulación 3.**

El diseño inicial (9.62 cm) no cumplía con el nivel de "Ocupación Inmediata" en ninguna de las simulaciones (9.35 cm, 10.62 cm, y 7.66 cm, respectivamente). La Simulación 1 logró una reducción significativa del desplazamiento (13.2%), la Simulación 2 priorizó la reducción de peso (193.4 ton), y la Simulación 3 alcanzó el menor desplazamiento (5.77 cm), aunque con un peso mayor (224.76 ton). Esto indica que el Algoritmo Genético supera los métodos tradicionales al cumplir con los niveles de desempeño sísmico, pero los resultados dependen de los parámetros de optimización.

El enfoque propuesto optimizó la configuración de la estructura considerando como restricciones principales la disminución del desplazamiento, el cumplimiento

de los criterios de ocupación inmediata y la disminución progresiva de las áreas de los perfiles estructurales en función de la altura de los niveles, sin embargo, se analizó la evolución del promedio de áreas de vigas y columnas. La gráfica en la Figura 5.7 mostró que el promedio de áreas de las vigas (línea azul) presentó fluctuaciones significativas, con valores que oscilaron entre 10 y 15 ln², lo que reflejó la iteratividad del algoritmo para optimizar las restricciones declaradas en la función objetivo. Por otro lado, el promedio de áreas de las columnas (línea amarilla) exhibió un comportamiento más variable, con picos que alcanzaron los 25 ln² alrededor de la generación 12, para luego estabilizarse cerca de 20 ln² hacia las últimas generaciones. Este comportamiento sugirió que el Algoritmo Genético inicialmente exploró soluciones con áreas más grandes para las columnas, posiblemente para garantizar estabilidad estructural, antes de que hubiera convergido hacia valores más moderados que cumplieron con las restricciones de diseño sísmico.

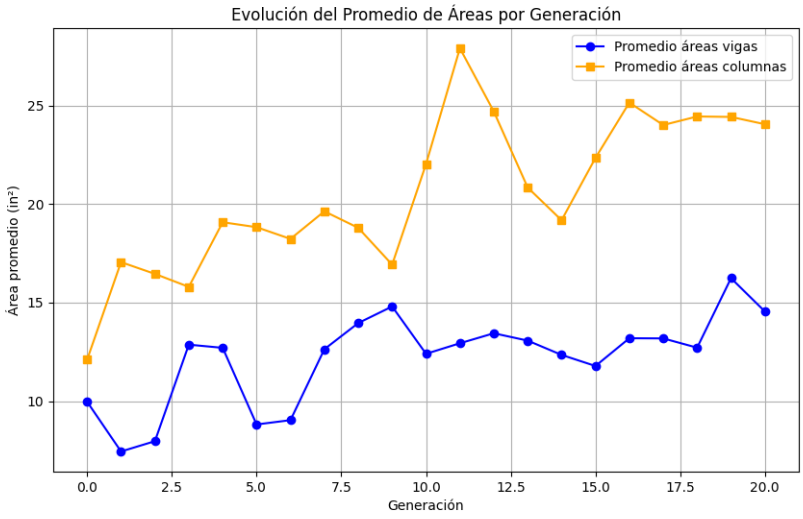


**Figura 5.7. Evolución de áreas de simulación 1.**

## 5.4 Comparación de los resultados obtenidos

El diseño inicial (9.62 cm) no cumplía con el nivel de "Ocupación Inmediata" en ninguna de las simulaciones (9.35 cm, 10.62 cm, y 7.66 cm, respectivamente).

La Simulación 1 logró una reducción significativa del desplazamiento (13.2%), la Simulación 2 priorizó la reducción de peso (193.4 ton), y la Simulación 3 alcanzó el menor desplazamiento (5.77 cm), aunque con un peso mayor (224.76 ton). Esto indica que el Algoritmo Genético supera los métodos tradicionales al cumplir con los niveles de desempeño sísmico, pero los resultados dependen de los parámetros de optimización.

La Simulación 1 (8.35 cm, 204.46 ton) ofrece un balance entre reducción de desplazamiento y peso, la Simulación 2 (9.89 cm, 193.4 ton) prioriza la sostenibilidad con el menor peso, y la Simulación 3 (5.77 cm, 224.76 ton) maximiza la seguridad sísmica con el menor desplazamiento. Todos los diseños cumplen con "Ocupación Inmediata", un aspecto crítico en zonas urbanas de alta sismicidad como Ciudad de México. Estos resultados son consistentes con estudios previos, como los de Pezeshk et al. (1997), que lograron optimizaciones similares en marcos de acero.

El comportamiento errático observado en la Simulación 3 (Figura 3, generaciones 20 a 40) puede atribuirse al elevado número de iteraciones (2600 evaluaciones) en comparación con el número limitado de perfiles disponibles (63). Esto probablemente causó una exploración excesiva del espacio de soluciones, agotando la diversidad de perfiles y llevando a fluctuaciones en la selección de configuraciones. Ajustar el número de generaciones o aumentar la lista de perfiles podría mitigar este efecto, mejorando la estabilidad del proceso de optimización.

Abordar múltiples objetivos, como peso y desplazamientos, es el foco de este estudio sobre Algoritmos Genéticos en diseño estructural. Hajela y Lin (1992) sugieren estrategias para equilibrar estos criterios, mostrando la versatilidad de las técnicas evolutivas en problemas ingenieriles. Las ideas expuestas podrían abrir caminos para considerar aspectos adicionales en la optimización de edificaciones verticales.

# 6 CONCLUSIONES Y RECOMENDACIONES

La presente investigación permitió evaluar la eficacia de los Algoritmos Genéticos como herramienta de optimización en el diseño sísmico de edificaciones verticales con elementos de acero, enfocándose en la minimización de los desplazamientos.

Los resultados obtenidos confirman que la hipótesis planteada se cumple en su totalidad, ya que la aplicación del Algoritmo Genético logró reducir significativamente los desplazamientos laterales de la estructura bajo cargas sísmicas estáticas equivalentes y como valor agregado se logró disminuir el peso total mediante la optimización de las secciones de los elementos estructurales. Esta doble mejora se alcanzó gracias a la capacidad del algoritmo para explorar un amplio espacio de soluciones y converger hacia configuraciones que equilibran eficiencia y seguridad.

La reducción del peso, además, implica un uso más eficiente de los materiales, lo que refuerza la viabilidad del enfoque propuesto en el contexto del diseño por desempeño sísmico.

Sin embargo, los resultados reflejan un alcance limitado por las condiciones iniciales del modelo, como la selección de perfiles estructurales y la regularidad de la estructura analizada. Esto sugiere que, aunque los objetivos principales se cumplieron, pero existe potencial para perfeccionar aún más el proceso de optimización.

La validación de los desplazamientos optimizados frente a análisis dinámicos no lineales y la comparación con diseños convencionales respaldan la efectividad del método, pero también destacan áreas de mejora en la diversidad de parámetros considerados.

Para mejorar los resultados obtenidos y ampliar la aplicabilidad de la metodología desarrollada, se proponen las siguientes mejoras:

En primer lugar, se sugiere incorporar una mayor variedad de tipos de perfiles estructurales en el proceso de optimización.

La inclusión de secciones adicionales, como perfiles compuestos o de geometrías no convencionales, podría enriquecer el espacio de búsqueda del Algoritmo Genético, permitiendo soluciones más diversificadas y potencialmente más eficientes en términos de peso y resistencia sísmica. Esta ampliación requeriría ajustar la base de datos de entrada y evaluar el impacto computacional de un mayor número de variables.

En cuanto a la continuidad de la investigación, se recomienda modificar la parametrización del código para que sea adaptable a estructuras irregulares, más allá de las configuraciones regulares analizadas en este estudio. Esto implicaría flexibilizar las funciones de entrada del Algoritmo Genético para considerar variaciones en la geometría y la distribución de masas, lo que ampliaría su utilidad en casos reales de edificaciones verticales complejas.

Se aconseja actualizar el diseño de los elementos estructurales optimizados para alinearse con las especificaciones de las Normas Técnicas Complementarias (NTC) 2023 para estructuras de acero. Esto incluye detallar las dimensiones y conexiones de los elementos según los requisitos de resistencia, ductilidad y disipación de energía establecidos en la normativa, asegurando que las soluciones optimizadas sean directamente aplicables en proyectos reales.

Finalmente, se propone realizar simulaciones adicionales que combinen análisis dinámicos no lineales con los resultados optimizados, para validar la robustez del diseño bajo sismos históricos específicos. Estas acciones no solo mejorarían la precisión de los resultados, sino que también consolidarían la metodología desarrollada como una alternativa viable para el diseño sostenible y sísmicamente eficiente de edificaciones verticales.

# 7 REFERENCIAS


Adeli, H., & Cheng, N.-T. (1994). Augmented Lagrangian genetic algorithm for structural optimization. *Journal of Structural Engineering*, 3106-3125.

Al-Kodmany, K. (2018). *The Vertical City: A Sustainable Development Model.* Southampton, UK: WIT Press.

Castillo, M. (20 de 09 de 2019). *Mexicanos contra la corrupcion y la impunidad*. (Mexicanos VS Corrupción e Impunidad. A.C.) Recuperado el 22 de 10 de 2024, de https://contralacorrupcion.mx/cdmx-sismo-19s-sin-lecciones/#:~:text=En%20el%20terremoto%20del%2019,la%20muerte%20de%20228%20personas.

Chopra, A. (2001). *Nonlinear Static Pushover Analysis of Structures.* Upper Saddle River, NJ: Prentice Hall.

Christensen, P., & Klarbring, A. (2009). *An Introduction to Structural Optimization.* Suecia: Springer.

Dasgupta, D., & Michalewicz, Z. (1997). *Evolutionary Algorithms in Engineering Applications.* Berlin, Heidelberg: Springer.

Deb, K. P. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 182-197.

Delyová, I., Frankovský, P., Bocko, J., Trebuñna, P., Živcák, J., Schürger, B., & Janigová, S. (2021). Sizing and Topology Optimization of Trusses Using. *Materials, 14*(4), 1-14.

Eiben, A. E., Back, T., Kok, J. N., & Spaink, H. P. (2015). *Introduction to Evolutionary Computing.* Amsterdam: Springer.

FEMA273, F. E. (1997). *NEHRP Guidelines for the Seismic Rehabilitation of Buildings.* WASHINTON,D.C.: FEMA, Washington, D.C.

FEMA440, F. E. (2005). *Improvement of NONLINEAR STATIC SEISMIC ANALYSIS PROCEDURES.* WASHINTON,D.C.

Fogel, D. B. (2006). *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence.* Piscataway, NJ: IEEE Press.

Foley, C., & Schinler, D. (2004). Automated Design of Steel Frames with Partial and Total Constraints Using an Object-Oriented Evolutionary Algorithm and Distributed Plasticity Model. *Journal of Structural Engineering*.

Fragiadakis, M., & Lagaros, N. D. (2011). Performance-based earthquake engineering using evolutionary optimization tools. *Engineering Optimization*, 283-301.

Ghasemof, A. (2022). Multi-objective optimization for probabilistic performance-based design of buildings using FEMA P-58 methodology. *Engineering Structures*, 113856.

Gholizadeh, S. (2013). Performance-based optimum seismic design of steel structures by a modified firefly algorithm and a new neural network. *Advances in Engineering Software*, 18-31.

Ghorbanie-Asl, M. (2007). *Optimal Design of Reinforced Concrete Frames Using Genetic Algorithms.* Auckland, Nueva Zelanda: University of Auckland, Department of Civil and Environmental Engineerin.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, MA: Addison-Wesley.

Gomez, A., & Rodriguez, D. (2020). Sostenibilidad en la edificación vertical: Retos y oportunidades en México. *Revista Hábitat Sustentable*, 34-45.

Gonzalez , N. (2020). *Algoritmos evolutivos para el diseno estructural estado del arte y caso estudio.* Santiago de Chile: Universidad de Chile.

Hajela, P., & Lin, C. (1992). Genetic search strategies in multicriterion optimal design. *Structural Optimization*, 99-107.

Hibbeler, R. C. (2012). *Análisis Estrucutral.* Estado de México: Pearson.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems.* Ann Arbor: University of Michigan Press.

Kaveh, A., & Talatahari, S. (2010). A novel heuristic optimization method: Charged system search. *Acta Mechanica*, 267-289.

Kreslin, M. (2011). The extended N2 method taking into account higher mode effects in elevation. *Earthquake Engineering & Structural Dynamics*, 1571-1589.

Liu, L. X. (2006). Multi-Criteria Optimization for Performance-Based Seismic Design of Steel Building Frames under Equivalent Static Loads. *Journal of Structural Engineering*.

Liu, M., & Stephen , B. (2004). Genetic algorithm optimization of steel frame structures with non-linear behavior. *Structural and Multidisciplinary Optimization*, 334-344.

Man, K.-F., Tang, K.-S., & Kwong, S. (1999). *Genetic Algorithms: Concepts and Designs.* London: Springer.

Martínez Gil, J. P. (2021). El uso mixto de suelo y la edificación vertical en el area metropolitana de Guadalajara. *Derecho GlobaL. Estudios sobre derecho y justicia, VIII*(22), 115-139.

Martins, J. P., Correia, J., Ljubinkovic, F., & Simooes da Silva, L. (2023). Cost optimisation of steel I-girder cross-sections using genetic algorithms. *Elsevier, 55*, 379-388.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms.* Cambridge, MA: MIT Press.

NTC. (1987). *Normas Técnicas Complementarias para Diseño por Sismo.* Ciudad de Mexico: Gaceta Oficial de la Ciudad de México.

NTC. (2023). *Normas Técnicas Complementarias para Diseño por Sismo.* Ciudad de Mexico: Gaceta Oficial de la Ciudad de México.

Pezeshk, S., Liu, A., & Camp, C. V. (1997). A Procedure to Estimate the Response Spectra for Central and Eastern United States Earthquakes. *Seismological Research Letters*, 749-759.

Plevris, V., Kremmeyda, G., & Fahjan, Y. (2017). *Performance-Based Seismic Design of Concrete Structures and Infrastructure.* Hershey, PA: IGI Global.

Priestley. (2000). Performance-based seismic design. *Bulletin of the New Zealand Society for Earthquake Engineering*, 325-346.

Priestley, M., Calvi, G., & Kowalsky, M. (2007). *Displacement-Based Seismic Design of Structures.* Pavia, Italy: IUSS Press.

Sánchez, A., & Terán , A. (2008). DISEÑO POR DESEMPEÑO DE ESTRUCTURAS DÚCTILES DE CONCRETO REFORZADO UBICADAS EN LA ZONA DEL LAGO DEL. *Revista de Ingeniería Sísmica*(78), 47-71.

Sánchez, S., Selles, M., Martinez, A., & Pla-Ferrando, R. (2012). Recent advances in structural optimization. *ANNALS of the ORADEA UNIVERSITY., XI*(NR1).

SEAOC. (1995). Vision 2000 Report on performance based seismic engineering of. Sacramento: Structural Engineers Association of California.

Whitehead, C., & Leslie, T. (2007). *The Skyscraper and the City: Design, Technology, and Innovation.* Chicago, IL: University of Chicago Press.

Yeang, K. (2007). *Designing the Eco-Skyscraper.* Chichester, UK: Wiley-Academy.

Zarrin, M., Poursha, M., & Gharabaghi, M. (2021). An updated consecutive modal Pushover (UCMP) procedure for estimating the ductility level earthquake design demands of jacket offshore platforms. *Soil Dynamics and Earthquake Engineering*.

# 8 APÉNDICE

## 8.1 Resumen de simulación 1.

**Tabla 8. Resumen extendido de simulación 1.**

| | | | | | | |
|---|---|---|---|---|---|---|
| Cantidad de evaluaciones totales | 440 | | | | | |
| Tiempo de evaluacion (hrs) | 1.63 | | | | | |
| Perfiles iniciales | | | | | | |
| Grupo | Perfil de viga | Área (in²) | d (in) | Perfil de columna | Área (in²) | d (in) |
| Grupo 1 | W6X20 | 5.87 | 6.2 | W10X112 | 32.9 | 11.4 |
| Grupo 2 | W5X19 | 5.56 | 5.15 | W10X88 | 26 | 10.8 |
| Grupo 3 | W5X16 | 4.71 | 5.01 | W8X67 | 19.7 | 9 |
| Resultados del mejor diseño | | | | | | |
| Grupo | Perfil de viga | Perfil de columna | Desplazamiento máximo (cm) | Desplazamiento para Ocupación Inmediata (cm) | | |
| Grupo 1 (Niveles 0-3) | W10X49 | W18X119 | 8.35 | 9.35 | | |
| Grupo 2 (Niveles 4-7) | W21X48 | W10X77 | 8.35 | 9.35 | | |
| Grupo 3 (Niveles 8-9) | W12X35 | W10X49 | 8.35 | 9.35 | | |
| Nivel de desempeño cumplido | Ocupación Inmediata | | | | | |
| Desplazamientos del mejor diseño | | | | | | |
| Nivel de Desempeño | Desplazamiento Objetivo (cm) | | | | | |
| Ocupación Inmediata | 9.35 | | | | | |
| Seguridad | 11.22 | | | | | |
| Prevención al Colapso | 14.02 | | | | | |
| Desplazamiento máximo (max_u1) | 8.35 | | | | | |
| Peso en Ton | 204.46 | | | | | |

## 8.2 Resumen de simulación 2

**Tabla 9. Resumen extendido de simulación 2.**

| | | | | | | |
|---|---|---|---|---|---|---|
| Cantidad de evaluaciones totales | 960 | | | | | |
| Tiempo de evaluacion (hrs) | 3.57 | | | | | |
| Perfiles iniciales | | | | | | |
| Grupo | Perfil de viga | Área (in²) | d (in) | Perfil de columna | Área (in²) | d (in) |
| Grupo 1 | W6X20 | 5.87 | 6.2 | W10X112 | 32.9 | 11.4 |
| Grupo 2 | W5X19 | 5.56 | 5.15 | W10X88 | 26 | 10.8 |
| Grupo 3 | W5X16 | 4.71 | 5.01 | W8X67 | 19.7 | 9 |
| Resultados del mejor diseño | | | | | | |
| Grupo | Perfil de viga | Perfil de columna | Desplazamiento máximo (cm) | Desplazamiento para Ocupación Inmediata (cm) | | |
| Grupo 1 (Niveles 0-3) | W4X13 | W30X124 | 9.89 | 10.62 | | |
| Grupo 2 (Niveles 4-7) | W10X12 | W16X89 | 9.89 | 10.62 | | |
| Grupo 3 (Niveles 8-9) | W8X10 | W12X58 | 9.89 | 10.62 | | |
| Nivel de desempeño cumplido | Ocupación Inmediata | | | | | |
| Desplazamientos del mejor diseño | | | | | | |
| Nivel de Desempeño | Desplazamiento Objetivo (cm) | | | | | |
| Ocupación Inmediata | 10.62 | | | | | |
| Seguridad | 12.74 | | | | | |
| Prevención al Colapso | 15.92 | | | | | |
| Desplazamiento máximo (max_u1) | 9.89 | | | | | |
| Peso en Ton | 193.4 | | | | | |

## 8.3 Resumen de simulación 3

**Tabla 10. Resumen extendido de simulación 3.**

| | |
|---|---|
| Cantidad de evaluaciones totales | 2600 |
| Tiempo de evaluacion (hrs) | 9.66 |

| Perfiles iniciales calculados | | | | | | |
|---|---|---|---|---|---|---|
| Grupo | Perfil de viga | Área (in²) | d (in) | Perfil de columna | Área (in²) | d (in) |
| Grupo 1 | W6X20 | 5.87 | 6.2 | W10X112 | 32.9 | 11.4 |
| Grupo 2 | W5X19 | 5.56 | 5.15 | W10X88 | 26 | 10.8 |
| Grupo 3 | W5X16 | 4.71 | 5.01 | W8X67 | 19.7 | 9 |

| Resultados del mejor diseño | | | | |
|---|---|---|---|---|
| Grupo | Perfil de viga | Perfil de columna | Desplazamiento máximo (cm) | Desplazamiento para Ocupación Inmediata (cm) |
| Grupo 1 (Niveles 0-3) | W18X97 | W14X145 | 5.77 | 7.66 |
| Grupo 2 (Niveles 4-7) | W10X60 | W14X109 | 5.77 | 7.66 |
| Grupo 3 (Niveles 8-9) | W18X46 | W12X65 | 5.77 | 7.66 |

| | |
|---|---|
| Nivel de desempeño cumplido | Ocupación Inmediata |
| Desplazamientos del mejor diseño | |
| Nivel de Desempeño | Desplazamiento Objetivo (cm) |
| Ocupación Inmediata | 7.66 |
| Seguridad | 9.19 |
| Prevención al Colapso | 11.49 |
| Desplazamiento máximo (max_u1) | 5.77 |
| Peso en Ton | 224.76 |

## 8.4 Evolución de áreas de simulación 2

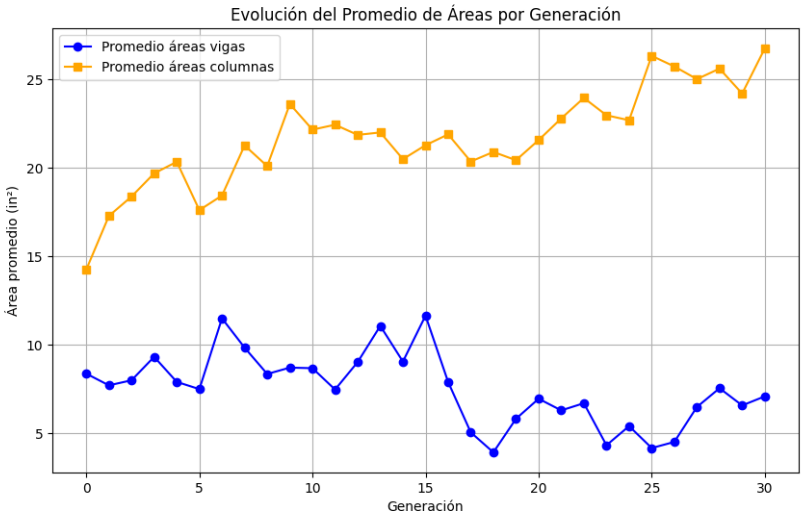


**Figura 8.1. Evolución de áreas de simulación 2.**

## 8.5 Evolución de áreas de simulación 3

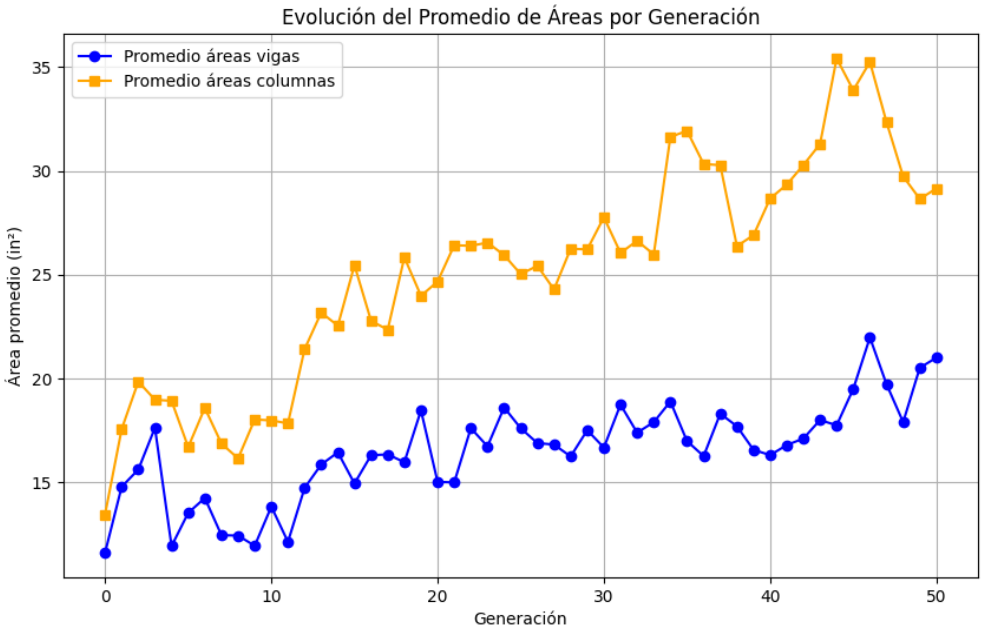


**Figura 8.2. Evolución de áreas de simulación 2.**

## 8.6 Herramienta “TESTSG”

```
import os
import sys
import comtypes.client
from ctypes import c_long, c_double, c_char_p, byref, create_string_buffer
from tabulate import tabulate
import pandas as pd
import math
import numpy as np
import matplotlib.pyplot as plt

# Dimensiones del edificio
nniv = int(10)
lentrepiso = 4.0
ncrujx = int(2)
ncrujy = int(1)
lcruj = 4.0

point_name = "66"
```

```
# Cargas para zotea
cma = 0.554
cva = 0.1
cvia = 0.07

# Cargas restantes
cm = 0.432
cv = 0.19
cvi = 0.1

# Inicializar SAP2000
helper = comtypes.client.CreateObject('SAP2000v1.Helper')
helper = helper.QueryInterface(comtypes.gen.SAP2000v1.cHelper)
mySapObject = helper.GetObject("CSI.SAP2000.API.SapObject")
SapModel = mySapObject.SapModel
SapModel.SetModelIsLocked(False)

# Abrir un archivo de SAP2000 existente
sap2000_file_path = r"C:\PYTHON\WORKSPACE\SAP\sap\test.sdb"
SapModel.File.OpenFile(sap2000_file_path)
SapModel.SelectObj.All()

# Definir unidades
SapModel.SetPresentUnits(12)

# Definir materiales
SapModel.PropMaterial.SetMaterial('ACERO', 1)
SapModel.PropMaterial.SetMaterial('CONCRETO', 2)

# Importar secciones AISC15
prop_file_path = r"C:\PYTHON\WORKSPACE\DATABASE\AISC15.xml"

secciones = [
    "W44X335", "W44X290", "W44X262", "W44X230", "W40X655", "W40X593", "W40X503", "W40X431",
"W40X397", "W40X372",
    "W40X362", "W40X324", "W40X297", "W40X277", "W40X249", "W40X215", "W40X199", "W40X392",
"W40X331", "W40X327",
    "W40X294", "W40X278", "W40X264", "W40X235", "W40X211", "W40X183", "W40X167", "W40X149",
"W36X925", "W36X853",
    "W36X802", "W36X723", "W36X652", "W36X529", "W36X487", "W36X441", "W36X395", "W36X361",
"W36X330", "W36X302",
    "W36X282", "W36X262", "W36X247", "W36X231", "W36X256", "W36X232", "W36X210", "W36X194",
"W36X182", "W36X170",
    "W36X160", "W36X150", "W36X135", "W33X387", "W33X354", "W33X318", "W33X291", "W33X263",
"W33X241", "W33X221",
    "W33X201", "W33X169", "W33X152", "W33X141", "W33X130", "W33X118", "W30X391", "W30X357",
"W30X326", "W30X292",
```

```
            "W30X261", "W30X235", "W30X211", "W30X191", "W30X173", "W30X148", "W30X132", "W30X124",
"W30X116", "W30X108",
            "W30X99", "W30X90", "W27X539", "W27X368", "W27X336", "W27X307", "W27X281", "W27X258", "W27X235",
"W27X217",
            "W27X194", "W27X178", "W27X161", "W27X146", "W27X129", "W27X114", "W27X102", "W27X94", "W27X84",
"W24X370",
            "W24X335", "W24X306", "W24X279", "W24X250", "W24X229", "W24X207", "W24X192", "W24X176",
"W24X162", "W24X146",
            "W24X131", "W24X117", "W24X104", "W24X103", "W24X94", "W24X84", "W24X76", "W24X68", "W24X62",
"W24X55", "W21X275",
            "W21X248", "W21X223", "W21X201", "W21X182", "W21X166", "W21X147", "W21X132", "W21X122",
"W21X111", "W21X101",
            "W21X93", "W21X83", "W21X73", "W21X68", "W21X62", "W21X55", "W21X48", "W21X57", "W21X50",
"W21X44", "W18X311",
            "W18X283", "W18X258", "W18X234", "W18X211", "W18X192", "W18X175", "W18X158", "W18X143",
"W18X130", "W18X119",
            "W18X106", "W18X97", "W18X86", "W18X76", "W18X71", "W18X65", "W18X60", "W18X55", "W18X50",
"W18X46", "W18X40",
            "W18X35", "W16X100", "W16X89", "W16X77", "W16X67", "W16X57", "W16X50", "W16X45", "W16X40",
"W16X36", "W16X31",
            "W16X26", "W14X873", "W14X808", "W14X730", "W14X665", "W14X605", "W14X550", "W14X500", "W14X455",
"W14X426",
            "W14X398", "W14X370", "W14X342", "W14X311", "W14X283", "W14X257", "W14X233", "W14X211",
"W14X193", "W14X176",
            "W14X159", "W14X145", "W14X132", "W14X120", "W14X109", "W14X99", "W14X90", "W14X82", "W14X74",
"W14X68", "W14X61",
            "W14X53", "W14X48", "W14X43", "W14X38", "W14X34", "W14X30", "W14X26", "W14X22", "W12X336",
"W12X305", "W12X279",
            "W12X252", "W12X230", "W12X210", "W12X190", "W12X170", "W12X152", "W12X136", "W12X120",
"W12X106", "W12X96",
            "W12X87", "W12X79", "W12X72", "W12X65", "W12X58", "W12X53", "W12X50", "W12X45", "W12X40",
"W12X35", "W12X30",
            "W12X26", "W12X22", "W12X19", "W12X16", "W12X14", "W10X112", "W10X100", "W10X88", "W10X77",
"W10X68", "W10X60",
            "W10X54", "W10X49", "W10X45", "W10X39", "W10X33", "W10X30", "W10X26", "W10X22", "W10X19",
"W10X17", "W10X15",
            "W10X12", "W8X67", "W8X58", "W8X48", "W8X40", "W8X35", "W8X31", "W8X28", "W8X24", "W8X21", "W8X18",
"W8X15",
            "W8X13", "W8X10", "W6X25", "W6X20", "W6X15", "W6X16", "W6X12", "W6X9", "W6X8.5", "W5X19", "W5X16",
"W4X13"
        ]

        for seccion in secciones:
            ret = SapModel.PropFrame.ImportProp(seccion, "ACERO", prop_file_path, seccion)

        # Agregar patrones de carga
        SapModel.LoadPatterns.Add("DEAD", 1, 1.0, True)
        SapModel.LoadPatterns.Add("SISMO X", 5, 0, True)
        SapModel.LoadPatterns.Add("SISMO Y", 5, 0, True)
```

```
SapModel.LoadPatterns.Add("VIVA", 3, 0, True)
SapModel.LoadPatterns.Add("VIVA INST", 3, 0, True)

# Importar espectro de diseño desde un archivo
espectro_file_path = r"C:\PYTHON\WORKSPACE\ESPECTRO\ESPECTRO CIUDAD DE MEXICO.txt"
with open(espectro_file_path, 'r') as file:
    lines = file.readlines()

periods = []
values = []
for line in lines:
    period, value = map(float, line.split())
    periods.append(period)
    values.append(value)

damp_ratio = 0.05
SapModel.Func.FuncRS.SetUser("ESPC", len(periods), periods, values, damp_ratio)

SapModel.LoadCases.ResponseSpectrum.SetCase("SISMO X")
SapModel.LoadCases.ResponseSpectrum.SetCase("SISMO Y")
d1 = ["U1"]
d2 = ["U2"]
espc = ["ESPC"]
sf = [9.81]
MyCSys = ["Global"]
MyAng = [0.0]
SapModel.LoadCases.ResponseSpectrum.SetLoads("SISMO X", 1, d1, espc, sf, MyCSys, MyAng)
SapModel.LoadCases.ResponseSpectrum.SetLoads("SISMO Y", 1, d2, espc, sf, MyCSys, MyAng)

SapModel.RespCombo.Add("1.3D + 1.5V", 0)
SapModel.RespCombo.SetCaseList("1.3D + 1.5V", 0, "DEAD", 1.3)
SapModel.RespCombo.SetCaseList("1.3D + 1.5V", 0, "VIVA INST", 1.5)

SapModel.RespCombo.Add("COMB SISMO X", 0)
SapModel.RespCombo.SetCaseList("COMB SISMO X", 0, "DEAD", 1.1)
SapModel.RespCombo.SetCaseList("COMB SISMO X", 0, "VIVA INST", 1.1)
SapModel.RespCombo.SetCaseList("COMB SISMO X", 0, "SISMO X", 1.1)
SapModel.RespCombo.SetCaseList("COMB SISMO X", 0, "SISMO Y", 0.3)

SapModel.RespCombo.Add("COMB SISMO Y", 0)
SapModel.RespCombo.SetCaseList("COMB SISMO Y", 0, "DEAD", 1.1)
SapModel.RespCombo.SetCaseList("COMB SISMO Y", 0, "VIVA INST", 1.1)
SapModel.RespCombo.SetCaseList("COMB SISMO Y", 0, "SISMO X", 0.3)
SapModel.RespCombo.SetCaseList("COMB SISMO Y", 0, "SISMO Y", 1.1)

SapModel.RespCombo.Add("ENVOLVENTE", 1)
SapModel.RespCombo.SetCaseList("ENVOLVENTE", 1, "1.3D + 1.5V", 1.0)
SapModel.RespCombo.SetCaseList("ENVOLVENTE", 1, "COMB SISMO X", 1.0)
SapModel.RespCombo.SetCaseList("ENVOLVENTE", 1, "COMB SISMO Y", 1.0)
```

```
            altura_nivel = float(lentrepiso)
            ancho_crujia = float(lcruj)

            for nivel in range(nniv + 1):
                for crujia_x in range(ncrujx + 1):
                    for crujia_y in range(ncrujy + 1):
                        SapModel.PointObj.AddCartesian(float(crujia_x * ancho_crujia), float(crujia_y * ancho_crujia), float(nivel *
altura_nivel))

            grupo_num = 1
            for i in range(0, nniv, 4):
                grupov = f"VG{grupo_num}"
                grupoc = f"CG{grupo_num}"
                SapModel.GroupDef.SetGroup(grupov)
                SapModel.GroupDef.SetGroup('COLUMNAS')
                SapModel.GroupDef.SetGroup(grupoc)
                SapModel.GroupDef.SetGroup('VIGAS')
                grupo_num += 1

            SapModel.GroupDef.SetGroup('LOSAS')

            def obtener_area(perfil):
                excel = r"C:\PYTHON\WORKSPACE\DATABASE\DB.xlsx"
                df_local = pd.read_excel(excel)
                seccion_data = df_local[df_local['Perfil'] == perfil]
                if not seccion_data.empty:
                    return seccion_data['Area'].values[0]
                else:
                    raise ValueError(f"El perfil {perfil} no se encontró en la base de datos")

            def calcular_perfiles_iniciales(luz_vigas, luz_columnas, num_grupos):
                excel = r"C:\PYTHON\WORKSPACE\DATABASE\DB.xlsx"
                df = pd.read_excel(excel)
                if not all(col in df.columns for col in ['Perfil', 'Area', 'd']):
                    raise ValueError("El archivo DB.xlsx debe tener las columnas 'Perfil', 'Area' y 'd'")

                # Validar y limpiar datos
                df = df.dropna(subset=['Perfil', 'Area', 'd'])
                df = df[df['Area'] > 0]  # Asegurar que las áreas sean positivas
                print(f"Datos válidos en DB.xlsx: {len(df)} filas después de limpieza")

                # Estimación inicial de carga aproximada
                area_piso = luz_vigas * luz_columnas * ncrujx * ncrujy
                carga_muerta_piso = area_piso * (cm + cma)
                carga_viva_piso = area_piso * (cv + cva)
                carga_total_piso = carga_muerta_piso + carga_viva_piso
                carga_por_viga = carga_total_piso / (ncrujx * (ncrujy + 1)) if (ncrujx * (ncrujy + 1)) > 0 else 1.0
                carga_por_columna = carga_total_piso / (ncrujx * ncrujy) if (ncrujx * ncrujy) > 0 else 1.0
```

```
        print(f"Carga por viga: {carga_por_viga:.2f} kN, Carga por columna: {carga_por_columna:.2f} kN")

        # Factores de peralte ajustados
        k_vigas_base = 30
        k_columnas_base = 15

        perfiles_vigas_iniciales = []
        perfiles_columnas_iniciales = []
        peraltes_vigas = []
        peraltes_columnas = []

        df_sorted = df.sort_values(by='Area', ascending=False).copy()

        for i in range(num_grupos):
            k_vigas = k_vigas_base + i * 2
            k_columnas = k_columnas_base + i * 1
            peralte_viga_m = luz_vigas / k_vigas
            peralte_columna_m = luz_columnas / k_columnas
            peralte_viga_in = peralte_viga_m * 39.3701
            peralte_columna_in = peralte_columna_m * 39.3701
            peraltes_vigas.append(peralte_viga_in)
            peraltes_columnas.append(peralte_columna_in)
            print(f"Grupo {i+1}: Peralte viga = {peralte_viga_in:.2f} in, Peralte columna = {peralte_columna_in:.2f} in")

        for i in range(num_grupos):
            # Filtrar por peralte con margen de 1 pulgada (ampliable si no hay candidatos)
            df_sorted['diferencia_viga'] = abs(df_sorted['d'] - peraltes_vigas[i])
            margin = 1.0
            candidatos_vigas = df_sorted[df_sorted['diferencia_viga'] < margin].copy()
            while candidatos_vigas.empty and margin <= 5.0:
                print(f"Grupo {i+1}: No se encontraron candidatos para vigas con margen {margin:.1f}. Ampliando margen...")
                margin += 0.5
                candidatos_vigas = df_sorted[df_sorted['diferencia_viga'] < margin].copy()

            perfil_viga = None
            if not candidatos_vigas.empty:
                # Depuración de carga_estimada
                candidatos_vigas['carga_estimada'] = carga_por_viga / (candidatos_vigas['Area'] * 0.00064516)
                print(f"Grupo {i+1}: Dimensiones de candidatos_vigas: {len(candidatos_vigas)} filas")
                print(f"Grupo {i+1}: Valores de carga_estimada antes de filtrar: {candidatos_vigas['carga_estimada'].tolist()}")
                # Filtrar valores no finitos y válidos
                candidatos_vigas = candidatos_vigas[candidatos_vigas['carga_estimada'].notna() &
                                                    (candidatos_vigas['carga_estimada'] > 0) &
                                                    (candidatos_vigas['carga_estimada'] < float('inf'))]
                print(f"Grupo {i+1}: Dimensiones de candidatos_vigas después de filtrar: {len(candidatos_vigas)} filas")
                print(f"Grupo {i+1}: Valores de carga_estimada después de filtrar: {candidatos_vigas['carga_estimada'].tolist()}")
                if not candidatos_vigas.empty:
```

```
                    try:
                        idx_min = candidatos_vigas['carga_estimada'].idxmin()
                        perfil_viga = candidatos_vigas.loc[idx_min, 'Perfil']
                        # Límite de área máxima (50 in² para vigas)
                        if obtener_area(perfil_viga) > 50:
                            candidatos_vigas = candidatos_vigas[candidatos_vigas['Area'] <= 50]
                            if not candidatos_vigas.empty:
                                idx_min = candidatos_vigas['carga_estimada'].idxmin()
                                perfil_viga = candidatos_vigas.loc[idx_min, 'Perfil']
                            else:
                                perfil_viga = df_sorted.loc[df_sorted['diferencia_viga'].idxmin(), 'Perfil']
                                print(f"Grupo {i+1}: Límite de área excedido, usando perfil más cercano: {perfil_viga}")
                        print(f"Grupo {i+1}: Perfil viga seleccionado: {perfil_viga}, Área: {obtener_area(perfil_viga):.2f} in²")
                    except (KeyError, ValueError) as e:
                        print(f"Grupo {i+1}: Error al seleccionar perfil por carga_estimada: {e}. Usando perfil más cercano.")
                        perfil_viga = df_sorted.loc[df_sorted['diferencia_viga'].idxmin(), 'Perfil']
                else:
                    print(f"Grupo {i+1}: No hay candidatos válidos para vigas después de filtrar carga_estimada. Usando el perfil más cercano.")
                    perfil_viga = df_sorted.loc[df_sorted['diferencia_viga'].idxmin(), 'Perfil']
            else:
                print(f"Grupo {i+1}: No se encontraron candidatos para vigas incluso con margen ampliado. Usando el perfil más cercano.")
                perfil_viga = df_sorted.loc[df_sorted['diferencia_viga'].idxmin(), 'Perfil']

            perfiles_vigas_iniciales.append(perfil_viga)
            df_sorted = df_sorted[df_sorted['Perfil'] != perfil_viga]

            # Filtrar por peralte con margen de 1 pulgada (ampliable si no hay candidatos)
            df_sorted['diferencia_columna'] = abs(df_sorted['d'] - peraltes_columnas[i])
            margin = 1.0
            candidatos_columnas = df_sorted[df_sorted['diferencia_columna'] < margin].copy()
            while candidatos_columnas.empty and margin <= 5.0:
                print(f"Grupo {i+1}: No se encontraron candidatos para columnas con margen {margin:.1f}. Ampliando margen...")
                margin += 0.5
                candidatos_columnas = df_sorted[df_sorted['diferencia_columna'] < margin].copy()

            perfil_columna = None
            if not candidatos_columnas.empty:
                candidatos_columnas['carga_estimada'] = carga_por_columna / (candidatos_columnas['Area'] * 0.00064516)
                print(f"Grupo {i+1}: Dimensiones de candidatos_columnas: {len(candidatos_columnas)} filas")
                print(f"Grupo {i+1}: Valores de carga_estimada antes de filtrar: {candidatos_columnas['carga_estimada'].tolist()}")
                # Filtrar valores no finitos y válidos
                candidatos_columnas = candidatos_columnas[candidatos_columnas['carga_estimada'].notna() &
                                                          (candidatos_columnas['carga_estimada'] > 0) &
                                                          (candidatos_columnas['carga_estimada'] < float('inf'))]
```

```
                print(f"Grupo {i+1}: Dimensiones de candidatos_columnas después de filtrar: {len(candidatos_columnas)} filas")
                print(f"Grupo {i+1}: Valores de carga_estimada después de filtrar: {candidatos_columnas['carga_estimada'].tolist()}")
                if not candidatos_columnas.empty:
                    try:
                        idx_min = candidatos_columnas['carga_estimada'].idxmin()
                        perfil_columna = candidatos_columnas.loc[idx_min, 'Perfil']
                        # Límite de área máxima (100 in² para columnas)
                        if obtener_area(perfil_columna) > 100:
                            candidatos_columnas = candidatos_columnas[candidatos_columnas['Area'] <= 100]
                            if not candidatos_columnas.empty:
                                idx_min = candidatos_columnas['carga_estimada'].idxmin()
                                perfil_columna = candidatos_columnas.loc[idx_min, 'Perfil']
                            else:
                                perfil_columna = df_sorted.loc[df_sorted['diferencia_columna'].idxmin(), 'Perfil']
                                print(f"Grupo {i+1}: Límite de área excedido, usando perfil más cercano: {perfil_columna}")
                        print(f"Grupo {i+1}: Perfil columna seleccionado: {perfil_columna}, Área: {obtener_area(perfil_columna):.2f} in²")
                    except (KeyError, ValueError) as e:
                        print(f"Grupo {i+1}: Error al seleccionar perfil por carga_estimada: {e}. Usando perfil más cercano.")
                        perfil_columna = df_sorted.loc[df_sorted['diferencia_columna'].idxmin(), 'Perfil']
                else:
                    print(f"Grupo {i+1}: No hay candidatos válidos para columnas después de filtrar carga_estimada. Usando el perfil más cercano.")
                    perfil_columna = df_sorted.loc[df_sorted['diferencia_columna'].idxmin(), 'Perfil']
            else:
                print(f"Grupo {i+1}: No se encontraron candidatos para columnas incluso con margen ampliado. Usando el perfil más cercano.")
                perfil_columna = df_sorted.loc[df_sorted['diferencia_columna'].idxmin(), 'Perfil']

            perfiles_columnas_iniciales.append(perfil_columna)
            df_sorted = df_sorted[df_sorted['Perfil'] != perfil_columna]

        areas_vigas = [obtener_area(p) for p in perfiles_vigas_iniciales]
        areas_columnas = [obtener_area(p) for p in perfiles_columnas_iniciales]
        if not (all(areas_vigas[i] > areas_vigas[i+1] for i in range(len(areas_vigas)-1)) and
                all(areas_columnas[i] > areas_columnas[i+1] for i in range(len(areas_columnas)-1))):
            print(f"Advertencia: Perfiles iniciales no decrecientes. Intentando corregir...")
            perfiles_vigas_iniciales = [p for _, p in sorted(zip(areas_vigas, perfiles_vigas_iniciales), reverse=True)]
            perfiles_columnas_iniciales = [p for _, p in sorted(zip(areas_columnas, perfiles_columnas_iniciales), reverse=True)]
            areas_vigas = sorted(areas_vigas, reverse=True)
            areas_columnas = sorted(areas_columnas, reverse=True)
            if not (all(areas_vigas[i] > areas_vigas[i+1] for i in range(len(areas_vigas)-1)) and
                    all(areas_columnas[i] > areas_columnas[i+1] for i in range(len(areas_columnas)-1))):
                raise ValueError(f"Perfiles iniciales no decrecientes después de corrección: Vigas {areas_vigas}, Columnas {areas_columnas}")
```

```
            print(f"Perfiles vigas iniciales: {perfiles_vigas_iniciales}, Áreas: {areas_vigas}")
            print(f"Perfiles columnas iniciales: {perfiles_columnas_iniciales}, Áreas: {areas_columnas}")
            return perfiles_vigas_iniciales, perfiles_columnas_iniciales

        def ejecutar_analisis(perfiles_vigas, perfiles_columnas, generacion):
            SapModel.SetModelIsLocked(False)
            grupo_num = 1
            for i in range(0, nniv, 4):
                grupov = f"VG{grupo_num}"
                grupoc = f"CG{grupo_num}"
                SapModel.FrameObj.SetSection(grupov, perfiles_vigas[grupo_num - 1], 1)
                SapModel.FrameObj.SetSection(grupoc, perfiles_columnas[grupo_num - 1], 1)
                grupo_num += 1

            SapModel.SelectObj.ClearSelection()
            SapModel.SelectObj.CoordinateRange(-1e6, 1e6, -1e6, 1e6, 0, 0, False, "Global", True, True, False, False, False,
False)
            SapModel.PointObj.SetRestraint("all", [True, True, True, True, True, True], 2)

            for nivel in range(1, nniv + 1):
                for crujia_x in range(ncrujx):
                    for crujia_y in range(ncrujy):
                        nombrelosa = f'LOSA_{(nivel - 1) * ncrujx * ncrujy + crujia_x * ncrujy + crujia_y + 1}'
                        SapModel.AreaObj.SetLoadUniform(nombrelosa, "DEAD", cm, 10, True, "Global", 0)
                        SapModel.AreaObj.SetLoadUniform(nombrelosa, "VIVA", cv, 10, True, "Global", 0)
                        SapModel.AreaObj.SetLoadUniform(nombrelosa, "VIVA INST", cvi, 10, True, "Global", 0)

            for nivel in range(1, nniv + 1):
                diafragma = f"DIAFRAGMA_{nivel}"
                SapModel.ConstraintDef.SetDiaphragm(diafragma, 3, "Global")
                SapModel.SelectObj.ClearSelection()
                SapModel.SelectObj.CoordinateRange(-1e6, 1e6, -1e6, 1e6, float(nivel * altura_nivel), float(nivel *
altura_nivel), False, "Global", True, True, True, True, True, True)
                SapModel.PointObj.SetConstraint("all", diafragma, 2, True)

            SapModel.LoadCases.StaticNonlinear.SetCase("INELASTICO")
            load_type = ["Accel"]
            load_name = ["UX"]
            scale_factor = [-1.0]
            SapModel.LoadCases.StaticNonlinear.SetLoads("INELASTICO", 1, load_type, load_name, scale_factor)

            load_control = 2
            disp_type = 2
            displ = 1.0
            monitor_type = 1
            dof = 1
            gdispl = ""
            SapModel.LoadCases.StaticNonlinear.SetLoadApplication("INELASTICO", load_control, disp_type, displ,
monitor_type, dof, point_name, gdispl)
```

```
            SapModel.LoadCases.StaticNonlinear.SetResultsSaved("INELASTICO", True, 100, 100, True)

            SapModel.File.Save(sap2000_file_path)
            SapModel.Analyze.RunAnalysis()

            SapModel.DesignSteel.SetGroup('All', False)
            grupo_num = 1
            for nivel in range(1, nniv + 1, 4):
                grupov = f"VG{grupo_num}"
                grupoc = f"CG{grupo_num}"
                SapModel.DesignSteel.SetGroup(grupov, True)
                SapModel.DesignSteel.SetGroup(grupoc, True)
                grupo_num += 1
            SapModel.DesignSteel.SetComboDeflection('ENVOLVENTE', True)

            MyLCase = ["ENVOLVENTE"]
            XMax = ncrujx * lcruj
            YMax = ncrujy * lcruj
            ZMax = nniv * lentrepiso

            SapModel.SelectObj.CoordinateRange(XMax, XMax, YMax, YMax, ZMax, ZMax, False, "Global", True, True,
False, False, False, False)
            ret, target, point_names, constraint_names = SapModel.PointElm.GetConstraint("Selection", 0, [], [], 3)
            MyPoint = target[0] if point_names else None

            if MyPoint is None:
                print("Advertencia: No se pudo obtener el punto objetivo")
                raise ValueError("Punto objetivo no encontrado")

            MyDispl = [0.5]
            SapModel.DesignSteel.SetTargetDispl(1, MyLCase, [MyPoint], MyDispl)
            SapModel.DesignSteel.StartDesign()

            SapModel.Results.Setup.DeselectAllCasesAndCombosForOutput()
            SapModel.Results.Setup.SetComboSelectedForOutput("ENVOLVENTE")
            group_name = f"VG{(nniv - 1) // 4 + 1}"
            number_results, obj, elm, load_case, step_type, step_num, u1, u2, u3, r1, r2, r3 = 0, [], [], [], [], [], [], [], [], [], [], [], []
            ret, number_results, obj, elm, load_case, step_type, step_num, u1, u2, u3, r1, r2, r3 =
SapModel.Results.JointDispl(group_name, 2, number_results, obj, elm, load_case, step_type, step_num, u1, u2, u3, r1, r2,
r3)
            max_u1 = max(u1) if u1 and len(u1) > 0 else 0.0

            SapModel.SelectObj.ClearSelection()
            SapModel.SelectObj.CoordinateRange(XMax, XMax, YMax, YMax, ZMax, ZMax, False, "Global", True, True,
False, False, False, False)

            number_results, obj, elm, load_case, step_type, step_num, u1, u2, u3, r1, r2, r3 = 0, [], [], [], [], [], [], [], [], [], [], [], []
            SapModel.Results.Setup.DeselectAllCasesAndCombosForOutput()
            SapModel.Results.Setup.SetCaseSelectedForOutput("INELASTICO")
```

```
SapModel.Results.Setup.SetOptionNLStatic(2)
ret, number_results, obj, elm, load_case, step_type, step_num, u1, u2, u3, r1, r2, r3 = SapModel.Results.JointDispl(point_name, 3, number_results, obj, elm, load_case, step_type, step_num, u1, u2, u3, r1, r2, r3)
D = step_num if step_num else [0.0]

number_results, load_case, step_type, step_num, Fx, Fy, Fz, Mx, My, Mz, gx, gy, gz = 0, [], [], [], [], [], [], [], [], [], [], 0.0, 0.0, 0.0
ret, number_results, load_case, step_type, step_num, Fx, Fy, Fz, Mx, My, Mz, gx, gy, gz = SapModel.Results.BaseReact(number_results, load_case, step_type, step_num, Fx, Fy, Fz, Mx, My, Mz, gx, gy, gz)
V = [abs(val) for val in step_num] if step_num else [0.0]

desplazamientos = np.array(D[:len(V)])
fuerzas = np.array(V)

def calcular_representacion_bilineal(desplazamientos, fuerzas):
    if len(desplazamientos) < 3 or len(fuerzas) < 3:
        print("Advertencia: Datos insuficientes para la curva bilineal, usando valores por defecto")
        return [(0.0, 0.0), (0.1, 10.0), (0.2, 20.0)]

    d = np.array(desplazamientos)
    v = np.array(fuerzas)

    idx_vy = np.argmax(v)
    vy = v[idx_vy]
    dy = d[idx_vy]

    ki = vy / dy if dy > 0 else 1e-6
    ks = ki * 0.05

    area_real = np.trapezoid(v, d)

    def area_bilineal(du):
        return 0.5 * vy * du + (du - dy) * vy * (ks / ki)

    du_min, du_max = dy, 2.0 * d[-1]
    du = (du_min + du_max) / 2
    for _ in range(100):
        area_bil = area_bilineal(du)
        if abs(area_bil - area_real) < 1e-6:
            break
        if area_bil < area_real:
            du_min = du
        else:
            du_max = du
        du = (du_min + du_max) / 2

    du = max(du, dy)
    if du > du_max:
        du = du_max
```

```
        vu = vy + ks * (du - dy)
        return [(0.0, 0.0), (dy, vy), (du, vu)]

    puntos_bilineal = calcular_representacion_bilineal(desplazamientos, fuerzas)

    def calcular_rigideces(puntos_bilineal):
        x1, y1 = puntos_bilineal[0]
        x2, y2 = puntos_bilineal[1]
        x3, y3 = puntos_bilineal[2]

        Ki = (y2 - y1) / (x2 - x1) if (x2 - x1) > 0 else 1e-6
        Ks = (y3 - y2) / (x3 - x2) if (x3 - x2) > 0 else 0.05 * Ki
        Ke = Ki
        Vy = y2

        return Ki, Ke, Ks, Vy

    Ki, Ke, Ks, Vy = calcular_rigideces(puntos_bilineal)

    def peso():
        number_results, load_case, step_type, step_num, Fx, Fy, Fz, Mx, My, Mz, gx, gy, gz = 0, [], [], [], [], [], [], [], [], [], [], 0.0, 0.0, 0.0
        SapModel.Results.Setup.DeselectAllCasesAndCombosForOutput()
        SapModel.Results.Setup.SetCaseSelectedForOutput("DEAD")
        ret, number_results, load_case, step_type, step_num, Fx, Fy, Fz, Mx, My, Mz, gx, gy, gz = SapModel.Results.BaseReact(number_results, load_case, step_type, step_num, Fx, Fy, Fz, Mx, My, Mz, gx, gy, gz)
        W = sum(abs(val) for val in Fy) if Fy else 0.0
        return W

    W = peso()

    def calcular_Ti(W, Ki):
        g = 9.81
        M = W / g
        if Ki <= 0:
            raise ValueError("Ki debe ser positivo")
        Ti = 2 * math.pi * math.sqrt(M / Ki)
        return Ti

    def calcular_Te(Ki, Ke, Ti):
        if Ke <= 0 or Ki <= 0:
            raise ValueError("Ki y Ke deben ser positivos")
        Te = Ti * math.sqrt(Ki / Ke)
        return Te

    Ti = calcular_Ti(W, Ki)
    Te = calcular_Te(Ki, Ke, Ti)
```

```
def obtener_aceleracion_espectral(Te, periods, values, damp_ratio=0.05):
    if not periods or not values:
        raise ValueError("Espectro de diseño vacío")
    if Te < min(periods) or Te > max(periods):
        print(f"Advertencia: Te={Te:.2f} s fuera del rango del espectro ({min(periods)}-{max(periods)} s)")
        Te = max(min(Te, max(periods)), min(periods))
    Sa = np.interp(Te, periods, values) * 9.81
    return Sa

Sa = obtener_aceleracion_espectral(Te, periods, values, damp_ratio)

def calcular_coeficiente_C0(nniv):
    if nniv == 2:
        return 1.2
    elif nniv == 3:
        return 1.3
    elif nniv == 5:
        return 1.4
    else:
        return 1.5

def calcular_R(Sa, Vy, W, C0):
    g = 9.81
    if W <= 0 or Vy <= 0 or Sa <= 0:
        raise ValueError("Sa, Vy, y W deben ser positivos")
    R = ((Sa / g) / (Vy / W)) * (1 / C0)
    if R < 1.0 or R > 8.0:
        print(f"Advertencia: Factor de reducción R={R:.2f} fuera del rango típico (1.0-8.0)")
    return R

def calcular_coeficiente_C1(Te, R, nivel_desempeno):
    # Invertimos los valores de a para que C1 sea menor para OI y mayor para PC
    a = {
        "Ocupación Inmediata": 60,    # Menor denominador, menor C1
        "Seguridad": 90,
        "Prevención al Colapso": 130  # Mayor denominador, mayor C1
    }[nivel_desempeno]
    C1 = 1.0 + (R - 1.0) / (a * Te**2) if Te > 0 else 1.0
    if C1 < 1.0:
        C1 = 1.0
    print(f"C1 para {nivel_desempeno}: {C1:.4f}")
    return C1

def calcular_coeficiente_C2(Te, R, sistema, nivel_desempeno):
    if Te <= 0.2:
        te_range = 1
    elif 0.2 < Te <= 1.0:
        te_range = 2
    else:
```

```
            te_range = 3

        c2_values = {
            (1, 1, 1): 1.0, (1, 1, 2): 1.0, (1, 1, 3): 1.0,  # Tipo 1, Te_range 1
            (1, 2, 1): 1.0, (1, 2, 2): 1.1, (1, 2, 3): 1.3,  # Tipo 1, Te_range 2 (invertido)
            (1, 3, 1): 1.0, (1, 3, 2): 1.2, (1, 3, 3): 1.5,  # Tipo 1, Te_range 3 (invertido)
            (2, 1, 1): 1.0, (2, 1, 2): 1.0, (2, 1, 3): 1.0,  # Tipo 2, Te_range 1
            (2, 2, 1): 1.0, (2, 2, 2): 1.0, (2, 2, 3): 1.0,  # Tipo 2, Te_range 2
            (2, 3, 1): 1.0, (2, 3, 2): 1.0, (2, 3, 3): 1.0   # Tipo 2, Te_range 3
        }
        nivel_map = {"Ocupación Inmediata": 1, "Seguridad": 2, "Prevención al Colapso": 3}
        if sistema not in [1, 2] or nivel_desempeno not in nivel_map:
            raise ValueError("Sistema o nivel de desempeño no válido")

        C2 = c2_values.get((sistema, te_range, nivel_map[nivel_desempeno]), 1.0)
        print(f"C2 para {nivel_desempeno}: {C2:.4f}")
        return C2

    def calcular_coeficiente_C3(Te, R, Ks, Ki):
        alpha = Ks / Ki if Ki > 0 else 0.0
        if alpha > 0.05 or Te > 2.0:
            C3 = 1.0
        else:
            C3 = 1.0 + (abs(alpha) * ((R - 1) ** (3/2)) / Te)
            if C3 < 1.0:
                C3 = 1.0
        print(f"C3: {C3:.4f}")
        return C3

    def calcular_desplazamiento_elastico(Sa, Te):
        if Te <= 0 or Sa <= 0:
            raise ValueError("Te y Sa deben ser positivos")
        delta_e = (Sa * Te**2) / (4 * math.pi**2)
        return delta_e

    def calcular_punto_desempeno_fema440(C0, C1, C2, C3, delta_e, nivel_desempeno):
        if any(coef <= 0 for coef in [C0, C1, C2, C3, delta_e]):
            raise ValueError("Todos los coeficientes y delta_e deben ser positivos")
        delta_d = C0 * C1 * C2 * C3 * delta_e
        print(f"Delta_d para {nivel_desempeno}: {delta_d * 100:.2f} cm")
        if delta_d <= 0 or delta_d > 1.0:
            print(f"Advertencia: Desplazamiento objetivo delta_d={delta_d:.2f} m fuera del rango típico (0-1.0 m)")
        return delta_d

    def determinar_sistema(perfiles_vigas, perfiles_columnas):
        try:
            viga_especifica = "Viga_X_1_1_1"
            number_results, Obj, ObjSta, Elm, ElmSta, LoadCase, StepType, StepNum, P, V2, V3, T, M2, M3 = 0, [], [],
[], [], [], [], [], [], [], [], [], [], [], []
```

```
                SapModel.Results.Setup.DeselectAllCasesAndCombosForOutput()
                SapModel.Results.Setup.SetComboSelectedForOutput("ENVOLVENTE")
                ret, number_results, Obj, ObjSta, Elm, ElmSta, LoadCase, StepType, StepNum, P, V2, V3, T, M2, M3 = SapModel.Results.FrameForce(viga_especifica, 0, number_results, Obj, ObjSta, Elm, ElmSta, LoadCase, StepType, StepNum, P, V2, V3, T, M2, M3)
                cortante_actuante = float(max(abs(v) for v in P)) if P and len(P) > 0 else 0.0

                prop_name = create_string_buffer(256)
                s_auto = c_long()
                ret = SapModel.FrameObj.GetSection(viga_especifica, byref(prop_name), byref(s_auto))
                if ret == 0 and prop_name.value:
                    # Convertir el buffer a una cadena de texto
                    seccion_nombre = prop_name.value.decode('utf-8').strip('\x00')  # Eliminar caracteres nulos
                    seccion_data = pd.read_excel(r"C:\PYTHON\WORKSPACE\DATABASE\DB.xlsx")
                    seccion_data = seccion_data[seccion_data['Perfil'] == seccion_nombre]
                    if not seccion_data.empty:
                        area = seccion_data['Area'].values[0] * 0.00064516  # Convertir in² a m²
                        esfuerzo_cortante_permisible = 0.4 * 250e6  # 250 MPa como esfuerzo de corte permisble
                        capacidad_cortante = area * esfuerzo_cortante_permisible / 1000  # Convertir a kN
                    else:
                        capacidad_cortante = 0.0
                else:
                    print(f"Advertencia: No se pudo obtener el nombre de la sección para {viga_especifica}. Retorno: {ret}")
                    capacidad_cortante = 0.0

                ratio = capacidad_cortante / cortante_actuante if cortante_actuante > 0 else float('inf')
                sistema = 1 if ratio > 1.5 else 2
                print(f"Sistema determinado: Tipo {sistema} (Ratio cortante: {ratio:.2f})")
                return sistema
            except Exception as e:
                print(f"Error al determinar el sistema estructural: {e}")
                return 1  # Valor por defecto (Tipo 1) en caso de error

        sistema = determinar_sistema(perfiles_vigas, perfiles_columnas)

        C0 = calcular_coeficiente_C0(nniv)
        R = calcular_R(Sa, Vy, W, C0)
        delta_e = calcular_desplazamiento_elastico(Sa, Te)

        opc2 = calcular_coeficiente_C2(Te, R, sistema, "Ocupación Inmediata")
        sc2 = calcular_coeficiente_C2(Te, R, sistema, "Seguridad")
        pc2 = calcular_coeficiente_C2(Te, R, sistema, "Prevención al Colapso")

        opc1 = calcular_coeficiente_C1(Te, R, "Ocupación Inmediata")
        sc1 = calcular_coeficiente_C1(Te, R, "Seguridad")
        pc1 = calcular_coeficiente_C1(Te, R, "Prevención al Colapso")

        C3 = calcular_coeficiente_C3(Te, R, Ks, Ki)
```

```
            delta_d1 = calcular_punto_desempeno_fema440(C0, opc1, opc2, C3, delta_e, "Ocupación Inmediata")
            delta_d2 = calcular_punto_desempeno_fema440(C0, sc1, sc2, C3, delta_e, "Seguridad")
            delta_d3 = calcular_punto_desempeno_fema440(C0, pc1, pc2, C3, delta_e, "Prevención al Colapso")

            plot_path = None
            try:
                plt.figure(figsize=(10, 6))
                plt.plot(desplazamientos, fuerzas, label='Curva Pushover')
                plt.plot([p[0] for p in puntos_bilineal], [p[1] for p in puntos_bilineal], label='Representación Bilineal', linestyle='--
')
                for i, (x, y) in enumerate(puntos_bilineal):
                    plt.scatter(x, y, color='red')
                    plt.text(x, y, f'({x:.2f}, {y:.2f})', fontsize=9, ha='right')
                plt.text(puntos_bilineal[1][0], puntos_bilineal[1][1], f'Ki = {Ki:.2f} Ton/m', fontsize=9, ha='left', va='bottom')
                plt.text(puntos_bilineal[2][0], puntos_bilineal[2][1], f'Ks = {Ks:.2f} Ton/m', fontsize=9, ha='left', va='bottom')
                plt.xlabel('Desplazamiento (m)')
                plt.ylabel('Fuerza (Ton)')
                plt.legend()
                plt.title(f'Curva Pushover y Representación Bilineal - Generación {generacion}')
                plt.grid(True)

                plot_path = f"Pushover_gen_{generacion}.png"
                plt.savefig(plot_path)
                plt.close()
            except Exception as e:
                print(f"Advertencia: No se pudo generar la gráfica Pushover: {e}")

            tabla_desplazamientos = [
                ["Ocupación Inmediata", delta_d1 * 100],
                ["Seguridad", delta_d2 * 100],
                ["Prevención al Colapso", delta_d3 * 100]
            ]

            print(f"El desplazamiento máximo es {max_u1 * 100:.2f} cm")
            print(tabulate(tabla_desplazamientos, headers=["Nivel de Desempeño", "Desplazamiento Objetivo (cm)"],
tablefmt="grid"))

            if max_u1 < delta_d1:
                print("La estructura cumple con Ocupación Inmediata")
            elif delta_d2 > max_u1 > delta_d1:
                print("La estructura cumple con Seguridad")
            elif delta_d3 > max_u1 > delta_d2:
                print("La estructura cumple con Prevención al Colapso")
            else:
                print("La estructura no cumple con ningún nivel de desempeño")

            return max_u1, delta_d1, plot_path, tabla_desplazamientos
```

## 8.7 Herramienta “AG”

```
import random
import math
import pandas as pd
import numpy as np
from tabulate import tabulate
import time
from TESTSG import ejecutar_analisis, calcular_perfiles_iniciales
import matplotlib.pyplot as plt
from openpyxl import Workbook
from openpyxl.drawing.image import Image
import os

# Ruta del archivo Excel
excel_path = r"C:\PYTHON\WORKSPACE\DATABASE\DB.xlsx"

# Leer el archivo Excel
df = pd.read_excel(excel_path)
required_columns = ['Perfil', 'Area', 'd']
if not all(col in df.columns for col in required_columns):
    raise ValueError(f"El archivo Excel debe tener las columnas {required_columns}")

secciones = df['Perfil'].tolist()

# Dimensiones del edificio
nniv = 10
numero_grupos = math.ceil(nniv / 4)
lentrepiso = 4.0
ncrujx = 2
ncrujy = 1
lcruj = 4.0
def obtener_area(perfil):
    seccion_data = df[df['Perfil'] == perfil]
    if not seccion_data.empty:
        return seccion_data['Area'].values[0]
    else:
        raise ValueError(f"El perfil {perfil} no se encontró en la base de datos")

def obtener_altura(perfil):
    seccion_data = df[df['Perfil'] == perfil]
    if not seccion_data.empty:
        return seccion_data['d'].values[0]
    else:
        raise ValueError(f"El perfil {perfil} no se encontró en la base de datos")

def crear_poblacion_inicial(tamano_poblacion):
    poblacion = []
    try:
```

```
        perfiles_vigas_iniciales, perfiles_columnas_iniciales = calcular_perfiles_iniciales(lcruj, lentrepiso, numero_grupos)
        # ... (código existente de validación) ...
    except Exception as e:
        print(f"Error al obtener perfiles iniciales: {e}. Usando respaldo aleatorio.")
        # ... (código existente de respaldo) ...

    for _ in range(tamano_poblacion):
        individuo_vigas = []
        individuo_columnas = []
        disponibles_vigas = secciones.copy()
        disponibles_columnas = secciones.copy()
        for i in range(numero_grupos):
            if i == 0:  # Grupo 1
                candidatos_vigas = [s for s in disponibles_vigas if obtener_area(s) <= 30.0]  # Límite para Grupo 1
                candidatos_columnas = [s for s in disponibles_columnas if obtener_area(s) <= 50.0]
            elif i > 0:
                candidatos_vigas = [s for s in disponibles_vigas if obtener_area(s) < obtener_area(individuo_vigas[i-1])]
                candidatos_columnas = [s for s in disponibles_columnas if obtener_area(s) < obtener_area(individuo_columnas[i-
1])]
            if not candidatos_vigas:
                candidatos_vigas = disponibles_vigas
            if not candidatos_columnas:
                candidatos_columnas = disponibles_columnas
            perfil_viga = random.choice(candidatos_vigas)
            perfil_columna = random.choice(candidatos_columnas)
            individuo_vigas.append(perfil_viga)
            individuo_columnas.append(perfil_columna)
            disponibles_vigas.remove(perfil_viga)
            disponibles_columnas.remove(perfil_columna)
        individuo = {"vigas": individuo_vigas, "columnas": individuo_columnas}
        poblacion.append(individuo)

    return poblacion, perfiles_vigas_iniciales, perfiles_columnas_iniciales

def mutar(individuo, tasa_mutacion):
    for i in range(numero_grupos):
        if random.random() < tasa_mutacion:
            if i == 0:  # Grupo 1
                candidatos_vigas = [s for s in secciones if obtener_area(s) <= 30.0]  # Límite para Grupo 1
                candidatos_columnas = [s for s in secciones if obtener_area(s) <= 50.0]
            elif i > 0:
                candidatos_vigas = [s for s in secciones if obtener_area(s) < obtener_area(individuo["vigas"][i-1])]
                candidatos_columnas = [s for s in secciones if obtener_area(s) < obtener_area(individuo["columnas"][i-1])]
            if candidatos_vigas:
                individuo["vigas"][i] = random.choice(candidatos_vigas)
            if candidatos_columnas:
                individuo["columnas"][i] = random.choice(candidatos_columnas)
    return individuo
def evaluar_aptitud(individuo, generacion):
```

```
    perfiles_vigas = individuo["vigas"]
    perfiles_columnas = individuo["columnas"]

    start_time = time.time()
    try:
        max_u1, delta_d1, plot_path, tabla_desplazamientos = ejecutar_analisis(perfiles_vigas, perfiles_columnas, generacion)
        print(f"Individuo evaluado - Vigas: {perfiles_vigas}, Columnas: {perfiles_columnas}, max_u1: {max_u1}")
    except Exception as e:
        print(f"Error en ejecutar_analisis: {e}")
        return 1e10, 0.0, 0.0, 0.0, None, []

    end_time = time.time()
    tiempo_evaluacion = end_time - start_time
    print(f"Tiempo de evaluación: {tiempo_evaluacion:.2f} segundos")

    if not (math.isfinite(max_u1) and math.isfinite(delta_d1)):
        return 1e10, 0.0, 0.0, tiempo_evaluacion, None, []

    max_u1_cm = max_u1 * 100
    delta_d1_cm = delta_d1 * 100

    areas_vigas = [obtener_area(p) for p in perfiles_vigas]
    areas_columnas = [obtener_area(p) for p in perfiles_columnas]
    areas_decrecientes = all(areas_vigas[i] > areas_vigas[i+1] for i in range(len(areas_vigas)-1)) and \
                         all(areas_columnas[i] > areas_columnas[i+1] for i in range(len(areas_columnas)-1))

    # Aptitud base: minimizar max_u1
    aptitud = max_u1_cm
    if max_u1_cm > delta_d1_cm:
        aptitud += (max_u1_cm - delta_d1_cm) * 50  # Penalización por exceder delta_d1

    # Nueva penalización por áreas grandes (umbral de 50 in² como ejemplo)
    umbral_area = 50.0  # Ajusta este valor según lo que consideres razonable
    penalizacion_areas = sum(max(0, area - umbral_area) for area in areas_vigas) + \
                         sum(max(0, area - umbral_area) for area in areas_columnas)
    aptitud += penalizacion_areas * 10  # Factor de penalización ajustable

    if not areas_decrecientes:
        aptitud += 5000  # Penalización por áreas no decrecientes

    return float(aptitud), float(max_u1), float(delta_d1), tiempo_evaluacion, plot_path, tabla_desplazamientos

def seleccionar_padres(poblacion, aptitudes, num_padres, tamano_torneo=3):
    padres = []
    for _ in range(num_padres):
        torneo = random.sample(list(zip(poblacion, aptitudes)), tamano_torneo)
        ganador = min(torneo, key=lambda x: x[1][0])[0]
        padres.append(ganador)
    return padres
```

```
def cruzar(padre1, padre2):
    hijo_vigas = []
    hijo_columnas = []
    for i in range(numero_grupos):
        if random.random() < 0.7:  # Aumentar probabilidad de cruce
            nuevo_perfil_viga = padre1["vigas"][i]
        else:
            nuevo_perfil_viga = padre2["vigas"][i]
        if random.random() < 0.7:
            nuevo_perfil_columna = padre1["columnas"][i]
        else:
            nuevo_perfil_columna = padre2["columnas"][i]

        if i > 0:
            candidatos_vigas = [s for s in secciones if obtener_area(s) < obtener_area(hijo_vigas[i-1])]
            candidatos_columnas = [s for s in secciones if obtener_area(s) < obtener_area(hijo_columnas[i-1])]
            if candidatos_vigas and obtener_area(nuevo_perfil_viga) >= obtener_area(hijo_vigas[i-1]):
                if candidatos_vigas:
                    nuevo_perfil_viga = random.choice(candidatos_vigas)
                else:
                    nuevo_perfil_viga = hijo_vigas[i-1]
            if candidatos_columnas and obtener_area(nuevo_perfil_columna) >= obtener_area(hijo_columnas[i-1]):
                if candidatos_columnas:
                    nuevo_perfil_columna = random.choice(candidatos_columnas)
                else:
                    nuevo_perfil_columna = hijo_columnas[i-1]

        hijo_vigas.append(nuevo_perfil_viga)
        hijo_columnas.append(nuevo_perfil_columna)

    areas_vigas = [obtener_area(p) for p in hijo_vigas]
    areas_columnas = [obtener_area(p) for p in hijo_columnas]
    if not (all(areas_vigas[i] > areas_vigas[i+1] for i in range(len(areas_vigas)-1)) or
            all(areas_columnas[i] > areas_columnas[i+1] for i in range(len(areas_columnas)-1))):
        if not all(areas_vigas[i] > areas_vigas[i+1] for i in range(len(areas_vigas)-1)):
            hijo_vigas = [p for _, p in sorted(zip(areas_vigas, hijo_vigas), reverse=True)]
        if not all(areas_columnas[i] > areas_columnas[i+1] for i in range(len(areas_columnas)-1)):
            hijo_columnas = [p for _, p in sorted(zip(areas_columnas, hijo_columnas), reverse=True)]

    hijo = {"vigas": hijo_vigas, "columnas": hijo_columnas}
    return hijo

def mutar(individuo, tasa_mutacion):
    for i in range(numero_grupos):
        if random.random() < tasa_mutacion:  # Aumentada probabilidad de mutación
            if i > 0:
                candidatos_vigas = [s for s in secciones if obtener_area(s) < obtener_area(individuo["vigas"][i-1])]
                candidatos_columnas = [s for s in secciones if obtener_area(s) < obtener_area(individuo["columnas"][i-1])]
```

```
            else:
                candidatos_vigas = secciones
                candidatos_columnas = secciones
            if candidatos_vigas:
                individuo["vigas"][i] = random.choice(candidatos_vigas)
            if candidatos_columnas:
                individuo["columnas"][i] = random.choice(candidatos_columnas)
    return individuo

def suavizar_datos(datos, ventana=3):
    return np.convolve(datos, np.ones(ventana)/ventana, mode='valid')
def algoritmo_genetico(tamano_poblacion=30, num_generaciones=30, tasa_mutacion=0.2):
    poblacion, perfiles_vigas_iniciales, perfiles_columnas_iniciales = crear_poblacion_inicial(tamano_poblacion)
    total_evaluaciones = tamano_poblacion + tamano_poblacion * num_generaciones + tamano_poblacion

    generaciones = []
    max_u1_por_generacion = []
    promedio_max_u1_por_generacion = []
    promedio_areas_vigas_por_generacion = []
    promedio_areas_columnas_por_generacion = []
    excel_output_path = r"C:\PYTHON\WORKSPACE\max_u1_evolution.xlsx"
    wb = Workbook()
    image_paths = []
    tiempos_evaluacion = []

    print("Evaluando población inicial...")
    resultados = []
    for individuo in poblacion:
        diferencia, max_u1, delta_d1, tiempo, plot_path, tabla_desplazamientos = evaluar_aptitud(individuo, 0)
        resultados.append((diferencia, max_u1, delta_d1, tiempo, plot_path, tabla_desplazamientos))
        tiempos_evaluacion.append(tiempo)

    mejor_indice = resultados.index(min(resultados, key=lambda x: x[0]))
    mejor_individuo = poblacion[mejor_indice]
    generaciones.append(0)
    max_u1_por_generacion.append(resultados[mejor_indice][1])
    max_u1_values = [r[1] for r in resultados if np.isfinite(r[1])]
    promedio_max_u1_por_generacion.append(np.mean(max_u1_values) if max_u1_values else np.nan)

    areas_vigas = [obtener_area(p) for ind in poblacion for p in ind["vigas"]]
    areas_columnas = [obtener_area(p) for ind in poblacion for p in ind["columnas"]]
    promedio_areas_vigas_por_generacion.append(np.mean(areas_vigas))
    promedio_areas_columnas_por_generacion.append(np.mean(areas_columnas))

    ws = wb.create_sheet("Generación 0")
    if resultados[mejor_indice][4]:
        try:
            img = Image(resultados[mejor_indice][4])
            ws.add_image(img, 'A1')
```

```
            image_paths.append(resultados[mejor_indice][4])
        except Exception as e:
            print(f"Advertencia: No se pudo agregar la imagen para la generación 0: {e}")

    for gen in range(num_generaciones):
        print(f"\nGeneración {gen + 1}/{num_generaciones}")
        aptitudes = [r[0] for r in resultados]
        padres = seleccionar_padres(poblacion, resultados, tamano_poblacion // 2)
        nueva_generacion = []

        nueva_generacion.append(mejor_individuo)

        for i in range(0, len(padres)-1, 2):
            padre1 = padres[i]
            padre2 = padres[i+1]
            hijo1 = cruzar(padre1, padre2)
            hijo2 = cruzar(padre1, padre2)
            nueva_generacion.append(mutar(hijo1, tasa_mutacion))
            nueva_generacion.append(mutar(hijo2, tasa_mutacion))

        while len(nueva_generacion) < tamano_poblacion:
            nueva_generacion.append(random.choice(poblacion))
        poblacion = nueva_generacion[:tamano_poblacion]

        resultados = []
        for individuo in poblacion:
            diferencia, max_u1, delta_d1, tiempo, plot_path, tabla_desplazamientos = evaluar_aptitud(individuo, gen + 1)
            resultados.append((diferencia, max_u1, delta_d1, tiempo, plot_path, tabla_desplazamientos))
            tiempos_evaluacion.append(tiempo)

        mejor_indice = resultados.index(min(resultados, key=lambda x: x[0]))
        mejor_individuo = poblacion[mejor_indice]
        generaciones.append(gen + 1)
        max_u1_por_generacion.append(resultados[mejor_indice][1])
        max_u1_values = [r[1] for r in resultados if np.isfinite(r[1])]
        promedio_max_u1_por_generacion.append(np.mean(max_u1_values) if max_u1_values else np.nan)

        areas_vigas = [obtener_area(p) for ind in poblacion for p in ind["vigas"]]
        areas_columnas = [obtener_area(p) for ind in poblacion for p in ind["columnas"]]
        promedio_areas_vigas_por_generacion.append(np.mean(areas_vigas))
        promedio_areas_columnas_por_generacion.append(np.mean(areas_columnas))

        ws = wb.create_sheet(f"Generación {gen + 1}")
        if resultados[mejor_indice][4]:
            try:
                img = Image(resultados[mejor_indice][4])
                ws.add_image(img, 'A1')
                image_paths.append(resultados[mejor_indice][4])
            except Exception as e:
```

```
            print(f"Advertencia: No se pudo agregar la imagen para la generación {gen + 1}: {e}")

        promedio_tiempo = sum(tiempos_evaluacion) / len(tiempos_evaluacion)
        evaluaciones_restantes = total_evaluaciones - len(tiempos_evaluacion)
        tiempo_restante = promedio_tiempo * evaluaciones_restantes / 60
        print(f"Tiempo promedio por evaluación: {promedio_tiempo:.2f} segundos")
        print(f"Evaluaciones restantes: {evaluaciones_restantes}")
        print(f"Tiempo estimado restante: {tiempo_restante:.2f} minutos")

    print("\nEvaluando población final...")
    resultados = []
    for individuo in poblacion:
        diferencia, max_u1, delta_d1, tiempo, plot_path, tabla_desplazamientos = evaluar_aptitud(individuo,
num_generaciones)
        resultados.append((diferencia, max_u1, delta_d1, tiempo, plot_path, tabla_desplazamientos))
        tiempos_evaluacion.append(tiempo)

    aptitudes = [round(r[0], 2) for r in resultados]
    print("\n\nAptitudes finales:", aptitudes)

    mejor_individuo = None
    mejor_aptitud = float('inf')
    mejor_tabla_desplazamientos = []
    nivel_desempeno = ""
    for i, (aptitud, max_u1, delta_d1, _, plot_path, tabla_desplazamientos) in enumerate(resultados):
        areas_vigas = [obtener_area(p) for p in poblacion[i]["vigas"]]
        areas_columnas = [obtener_area(p) for p in poblacion[i]["columnas"]]
        if all(areas_vigas[j] > areas_vigas[j+1] for j in range(len(areas_vigas)-1)) and \
           all(areas_columnas[j] > areas_columnas[j+1] for j in range(len(areas_columnas)-1)):
            if aptitud < mejor_aptitud:
                mejor_aptitud = aptitud
                mejor_individuo = poblacion[i]
                mejor_max_u1 = max_u1
                mejor_delta_d1 = delta_d1
                mejor_tabla_desplazamientos = tabla_desplazamientos

                # Validar tabla_desplazamientos antes de acceder a sus elementos
                print(f"Debug: tabla_desplazamientos = {tabla_desplazamientos}")
                if isinstance(tabla_desplazamientos, list) and len(tabla_desplazamientos) >= 3 and all(isinstance(row, list) and
len(row) >= 2 for row in tabla_desplazamientos):
                    delta_d2 = tabla_desplazamientos[1][1] / 100
                    delta_d3 = tabla_desplazamientos[2][1] / 100
                    if mejor_max_u1 < mejor_delta_d1:
                        nivel_desempeno = "Ocupación Inmediata"
                    elif delta_d2 > mejor_max_u1 > mejor_delta_d1:
                        nivel_desempeno = "Seguridad"
                    elif delta_d3 > mejor_max_u1 > delta_d2:
                        nivel_desempeno = "Prevención al colapso"
                    else:
```

```
                nivel_desempeno = "Ningún nivel de desempeño"
            else:
                print(f"Advertencia: tabla_desplazamientos no tiene el formato esperado. Usando 'Ningún nivel de desempeño'.")
                nivel_desempeno = "Ningún nivel de desempeño"

            ws = wb.create_sheet(f"Generación {num_generaciones}")
            if plot_path:
                try:
                    img = Image(plot_path)
                    ws.add_image(img, 'A1')
                    image_paths.append(plot_path)
                except Exception as e:
                    print(f"Advertencia: No se pudo agregar la imagen para la generación final: {e}")

    if mejor_individuo is None:
        print("\nError: No se encontró ningún individuo con áreas decrecientes en la población final")
        return None

    if mejor_max_u1 <= mejor_delta_d1:
        print(f"\nÉxito: el desplazamiento de la estructura ({mejor_max_u1*100:.2f} cm) es menor que el desplazamiento de clasificación ({mejor_delta_d1*100:.2f} cm)")
    else:
        print(f"\nAdvertencia: el desplazamiento de la estructura ({mejor_max_u1*100:.2f} cm) es mayor que el desplazamiento de clasificación ({mejor_delta_d1*100:.2f} cm)")

    tabla_iniciales = []
    for i in range(numero_grupos):
        grupo = f"Grupo {i+1}"
        perfil_viga = perfiles_vigas_iniciales[i]
        area_viga = obtener_area(perfil_viga)
        d_viga = obtener_altura(perfil_viga)
        perfil_columna = perfiles_columnas_iniciales[i]
        area_columna = obtener_area(perfil_columna)
        d_columna = obtener_altura(perfil_columna)
        tabla_iniciales.append([grupo, perfil_viga, area_viga, d_viga, perfil_columna, area_columna, d_columna])

    headers_iniciales = ["Grupo", "Perfil de viga", "Área (in²)", "d (in)", "Perfil de columna", "Área (in²)", "d (in)"]
    print("\nPerfiles iniciales calculados:")
    print(tabulate(tabla_iniciales, headers=headers_iniciales, tablefmt="grid"))

    tabla_mejor = []
    for i in range(numero_grupos):
        grupo = f"Grupo {i+1} (Niveles {i*4}-{(i+1)*4-1 if (i+1)*4-1 < nniv else nniv-1})"
        perfil_viga = mejor_individuo["vigas"][i]
        perfil_columna = mejor_individuo["columnas"][i]
        tabla_mejor.append([grupo, perfil_viga, perfil_columna, round(mejor_max_u1 * 100, 2), round(mejor_delta_d1 * 100, 2)])
```

```
    headers_mejor = ["Grupo", "Perfil de viga", "Perfil de columna", "Desplazamiento máximo (cm)", "Desplazamiento para
Ocupación Inmediata (cm)"]
    print("\nResultados del mejor diseño:")
    print(tabulate(tabla_mejor, headers=headers_mejor, tablefmt="grid"))

    areas_vigas = [obtener_area(p) for p in mejor_individuo["vigas"]]
    areas_columnas = [obtener_area(p) for p in mejor_individuo["columnas"]]
    if all(areas_vigas[i] > areas_vigas[i+1] for i in range(len(areas_vigas)-1)) and \
       all(areas_columnas[i] > areas_columnas[i+1] for i in range(len(areas_columnas)-1)):
        print("\nÉxito: Las áreas son estrictamente decrecientes en la mejor solución")
    else:
        print("\nError: Las áreas no son estrictamente decrecientes en la mejor solución")
        print(f"Áreas de vigas: {areas_vigas}")
        print(f"Áreas de columnas: {areas_columnas}")

    tiempo_total = sum(tiempos_evaluacion) / 60
    print(f"\nTiempo total ejecutado: {tiempo_total:.2f} minutos")

    ws = wb.create_sheet("Evolución max_u1")
    plt.figure(figsize=(10, 6))
    max_u1_cm = [m * 100 for m in max_u1_por_generacion]
    promedio_max_u1_cm = [m * 100 for m in promedio_max_u1_por_generacion]
    plt.plot(generaciones, max_u1_cm, color='blue', marker='o', alpha=0.5, label='Mejor max_u1')
    plt.plot(generaciones, promedio_max_u1_cm, color='green', linestyle='--', label='Promedio max_u1')
    if len(max_u1_cm) > 3:
        max_u1_suavizado = suavizar_datos(max_u1_cm, ventana=3)
        generaciones_suavizado = generaciones[len(generaciones) - len(max_u1_suavizado):]
        plt.plot(generaciones_suavizado, max_u1_suavizado, color='red', linestyle='-', label='Mejor max_u1 suavizado')
    plt.xlabel('Generación')
    plt.ylabel('max_u1 (cm)')
    plt.title('Evolución de max_u1 por Generación')
    plt.grid(True)
    plt.legend()
    max_u1_plot_path = "max_u1_evolution.png"
    plt.savefig(max_u1_plot_path)
    plt.close()
    img = Image(max_u1_plot_path)
    ws.add_image(img, 'A1')
    image_paths.append(max_u1_plot_path)

    ws_areas = wb.create_sheet("Evolución Áreas Promedio")
    plt.figure(figsize=(10, 6))
    plt.plot(generaciones, promedio_areas_vigas_por_generacion, color='blue', marker='o', label='Promedio áreas vigas')
    plt.plot(generaciones, promedio_areas_columnas_por_generacion, color='orange', marker='s', label='Promedio áreas
columnas')
    plt.xlabel('Generación')
    plt.ylabel('Área promedio (in²)')
    plt.title('Evolución del Promedio de Áreas por Generación')
    plt.grid(True)
```

```
plt.legend()
areas_plot_path = "areas_evolution.png"
plt.savefig(areas_plot_path)
plt.close()
img_areas = Image(areas_plot_path)
ws_areas.add_image(img_areas, 'A1')
image_paths.append(areas_plot_path)

ws = wb.create_sheet("Resumen")
ws.append(["Cantidad de evaluaciones totales", total_evaluaciones])
perfiles_evaluados = set()
for gen_resultados in [resultados] + [resultados for gen in range(num_generaciones)]:
    for individuo in poblacion:
        perfiles_evaluados.update(individuo["vigas"])
        perfiles_evaluados.update(individuo["columnas"])
ws.append(["Cantidad de perfiles evaluados", len(perfiles_evaluados)])
ws.append([])

ws.append(["Perfiles iniciales calculados"])
ws.append(headers_iniciales)
for row in tabla_iniciales:
    ws.append(row)
ws.append([])

ws.append(["Resultados del mejor diseño"])
ws.append(headers_mejor)
for row in tabla_mejor:
    ws.append(row)
ws.append([])

ws.append(["Nivel de desempeño cumplido", nivel_desempeno])
ws.append([])

ws.append(["Desplazamientos del mejor diseño"])
headers_desplazamientos = ["Nivel de Desempeño", "Desplazamiento Objetivo (cm)"]
ws.append(headers_desplazamientos)
for row in mejor_tabla_desplazamientos:
    ws.append(row)
ws.append(["Desplazamiento máximo (max_u1)", round(mejor_max_u1 * 100, 2)])

wb.save(excel_output_path)
print(f"\nGráfica y datos guardados en: {excel_output_path}")

for path in image_paths:
    if os.path.exists(path):
        os.remove(path)

return mejor_individuo
```

```
# Ejecutar el algoritmo con los parámetros que usaste
mejor_solucion = algoritmo_genetico(tamano_poblacion=20, num_generaciones=20, tasa_mutacion=0.2)
```