



UNIVERSITAT  
POLITÈCNICA  
DE VALÈNCIA

INTEGRACIÓN DE LAS TÉCNICAS BIM EN LA OPTIMIZACIÓN DEL CICLO DE VIDA  
DE ESTRUCTURAS DE HORMIGÓN EN EDIFICACIÓN

Víctor Fernández Mora

Tutor: Víctor Yepes Piqueras

Trabajo Fin de Máster

Máster Universitario en Ingeniería del Hormigón

Departamento Ingeniería de la Construcción y Proyectos de Ingeniería Civil

## RESUMEN

En los últimos tiempos se está llevando a cabo un gran cambio de paradigma en la industria de la Ingeniería, la Arquitectura y la Construcción gracias a la introducción de nuevas tecnologías como el *Building Information Modelling* y técnicas como el *Lean Construction* o la optimización del ciclo de vida de estructuras. Se plantea, en este marco, investigar las ventajas que supondría combinar estas nuevas tendencias y su viabilidad, así como la creación de una aplicación de optimización de estructuras de hormigón integrada completamente en un entorno BIM. Dicha investigación se lleva a cabo utilizando la *Application Programming Interface* de un entorno BIM donde ha sido modelado un elemento tipo viga isostática y mediante un algoritmo genético multicriterio con cuatro criterios de optimización diferentes -coste, emisión de CO<sub>2</sub>, consumo energético y número de barras- se consigue llegar a su diseño óptimo. Este algoritmo genético se ha ajustado utilizando la técnica de diseño de experimentos. La aplicación resultante ha permitido la incorporación de la optimización estructural a los entornos BIM de una manera plena permitiendo tener en cuenta criterios de construcción sostenible desde la fase inicial del proyecto.

## RESUM

Als últims temps s'està duent a terme un gran canvi paradigmàtic a la indústria de l'Enginyeria, l'Arquitectura i la Construcció gràcies a la introducció de noves tecnologies com el *Building Information Modelling* i tècniques com el *Lean Construction* o l'optimització de les estructures. Es planteja, dintre d'aquest marc, investigar els avantatges derivats de la combinació d'aquestes tècniques i estudiar la seua viabilitat, així com la creació d'una aplicació capaç d'optimitzar les estructures de formigó i que funcione integrada completament en un entorn BIM. Aquesta investigació es duu a terme utilitzant la *Application Programming Interface* d'un entorn BIM on ha sigut modelat un element tipus viga isostàtica i mitjançant un algoritme genètic multicriteri amb quatre criteris d'optimització diferents, cost, emissions de CO<sub>2</sub>, consum energètic y nombre de barres; s'aconsegueix arribar al seu disseny òptim. L'algoritme ha sigut ajustat utilitzant la tècnica de disseny d'experiments. L'aplicació resultant ha permès l'incorporació de l'optimització estructural als entorns BIM d'una manera completa permetent tenir en compte criteris de construcció sostenible fins i tot a la fase inicial del projecte.

## ABSTRACT

Lately there has been a huge change in the Architecture, Engineering and Construction industry, due to the introduction of new technologies like Building Information Modeling and techniques like Lean Construction or life cycle structure optimization. In this field we plan to investigate the derived advantages of combining these new trends and its viability, also we create a new application able to optimize reinforced concrete structures that is completely integrated inside a BIM environment. This research is done by utilizing one BIM's Application Programming Interface where an isostatic beam has been modeled and using a multicriteria genetic algorithm with four different optimization criteria –economic cost, CO2 emissions, energetic cost and number of reinforcing bars- we are able to get the optimum design. This genetic algorithm has been adjusted using the design of experiments technique. The resulting application allowed the inclusion of the structure optimization into a BIM environment completely, being able to have in mind the sustainable construction criteria from the beginning of the design stage.

#### **PALABRAS CLAVE**

Ahorro energético, algoritmo genético, *BIM*, construcción sostenible, diseño de experimentos, estructuras de hormigón armado, *Lean Construction*, optimización heurística, optimización multicriterio.

#### **PARAULES CLAU**

Algoritme genètic, *BIM*, construcció sostenible, disseny d'experiments, estalvi d'energia, estructures de formigó armat, *Lean Construction*, optimització heurística, optimització multicriteri.

#### **KEYWORDS**

*BIM, design of experiments, energy saving, genetic algorithm, heuristic optimization, Lean Construction, multicriteria optimization, reinforced concrete structures, sustainable construction.*

#### **LISTADO DE SIGLAS Y ACRÓNIMOS**

| ACRÓNIMO   | DEFINICIÓN   |
|------------|--|
| <b>AEC</b> | Industria de la Arquitectura, Ingeniería y Construcción                  |
| <b>API</b> | <i>Application Programming Interface</i>                                 |
| <b>BIM</b> | <i>Building Information Modelling</i>                                    |
| <b>BM</b>  | <i>Building Model</i>  |
| <b>CAD</b> | <i>Computer Assisted Design</i>  |
| <b>DOE</b> | <i>Design of Experiments</i> (Diseño de Experimentos)                    |
| <b>IPD</b> | <i>Integrated Project Delivery</i><br>(Proyecto Entregado Integralmente) |
| <b>LC</b>  | <i>Lean Construction</i>   |

# ÍNDICE

---

|   |          |
|---|----------|
| <b>1. INTRODUCCIÓN.....</b>                                   | <b>8</b> |
| 1.1 PRESENTACIÓN.....   | 8        |
| 1.2 OBJETIVOS DEL TRABAJO .....                               | 9        |
| 1.2.1 Objetivos .....   | 9        |
| 1.2.2 Alcance de la investigación .....                       | 9        |
| 1.3 BIM.....  | 12       |
| 1.3.1 Definición BIM.....                                     | 12       |
| 1.3.2 BIM vs CAD .....  | 14       |
| 1.3.3 Historia de los BIM .....                               | 17       |
| 1.3.4 Ventajas e inconvenientes de BIM.....                   | 19       |
| 1.3.5 Diferentes plataformas BIM .....                        | 21       |
| 1.3.6 Autodesk Revit como entorno de trabajo.....             | 23       |
| 1.3.7 Revit API y lenguaje de programación C# .....           | 24       |
| 1.4 ALGORITMOS GENÉTICOS .....                                | 26       |
| 1.4.1 Optimización .....                                      | 26       |
| 1.4.2 Optimización aplicada a estructuras de hormigón .....   | 28       |
| 1.4.3 Objetivos de la optimización estructural .....          | 29       |
| 1.4.4 Elección del algoritmo a utilizar .....                 | 31       |
| 1.4.5 Funcionamiento de los genéticos.....                    | 33       |
| 1.5 BIM Y OPTIMIZACIÓN .....                                  | 36       |
| 1.5.1 Tendencias futuras de los BIM y sus posibilidades ..... | 38       |
| 1.5.2 Prácticas de diseño estructural en BIM.....             | 40       |
| 1.5.3 Integración del diseño estructural en BIM.....          | 42       |
| 1.5.4 Optimización estructural en BIM .....                   | 46       |

---

|           |   |            |
|-----------|---|------------|
| <b>2.</b> | <b>DESARROLLO</b>   | <b>49</b>  |
| <hr/>     |   |            |
| 2.1       | PARÁMETROS Y SU CONEXIÓN CON LA API                               | 50         |
| 2.1.1     | Glosario de parámetros y definición de los mismos                 | 50         |
| 2.1.2     | Introducción de los parámetros en Revit                           | 55         |
| 2.1.3     | Captación y volcado de los valores de los parámetros              | 57         |
| 2.2       | VISIÓN GENERAL DE LA APLICACIÓN DE OPTIMIZACIÓN MONOCRITERIO      | 61         |
| 2.2.1     | Funcionamiento del código   | 61         |
| 2.2.2     | Funcionamiento del algoritmo                                      | 64         |
| 2.2.3     | Volcado de los resultados   | 65         |
| 2.2.4     | Objeto: Viga  | 66         |
| 2.2.5     | Objeto: Generación  | 68         |
| 2.2.6     | Objeto: Descendiente  | 70         |
| 2.2.7     | Gestión de Errores  | 73         |
| 2.3       | PLANTEAMIENTO DEL ALGORITMO MULTICRITERIO                         | 74         |
| 2.3.1     | Estudio de los diferentes criterios de optimización y su relación | 76         |
| 2.4       | VISIÓN GENERAL DE LA APLICACIÓN DE OPTIMIZACIÓN MULTICRITERIO     | 85         |
| <b>3.</b> | <b>ESTUDIO PARAMÉTRICO</b>  | <b>88</b>  |
| <hr/>     |   |            |
| 3.1       | DEFINICIÓN DEL CASO DE ESTUDIO                                    | 89         |
| 3.2       | RELACIONES ENTRE LOS DIFERENTES PARÁMETROS DE OPTIMIZACIÓN        | 91         |
| 3.3       | RESULTADOS DEL ESTUDIO PARAMÉTRICO                                | 95         |
| 3.4       | ESTUDIO DE SENSIBILIDAD DE PRECIOS                                | 102        |
| <b>4.</b> | <b>CONCLUSIONES</b>   | <b>113</b> |
| <hr/>     |   |            |
| 4.1       | REVISIÓN DE LOS OBJETIVOS   | 113        |
| 4.2       | CONCLUSIONES  | 116        |

|                                     |   |     |
|-------------------------------------|---|-----|
| 4.3                                 | FUTURAS VÍAS DE INVESTIGACIÓN .....                                   | 117 |
| 4.3.1                               | Diferentes tipologías estructurales .....                             | 117 |
| 4.3.2                               | Diferentes objetivos de optimización y diferentes algoritmos .....    | 118 |
| 4.3.3                               | Combinación con otras tecnologías integradas en los entornos BIM..... | 119 |
| 4.4                                 | AGRADECIMIENTOS .....   | 120 |
| 5.                                  | BIBLIOGRAFÍA.....   | 121 |
| ANEXO 1: CÓDIGO MONOCRITERIO.....   |   | 126 |
| ANEXO 2: CÓDIGO MULTICRITERIO ..... |   | 139 |
| ANEXO 3: GUÍA DE USUARIO .....      |   | 152 |
| <hr/>                               |   |     |
| A.                                  | GUÍA PASO A PASO .....  | 152 |
| B.                                  | EJEMPLO DE AJUSTE DE LOS PARÁMETROS .....                             | 162 |

## 1. INTRODUCCIÓN

---

### 1.1 PRESENTACIÓN

Durante los últimos quince años se ha producido un cambio de paradigma en todos los aspectos del mundo de la construcción debido, principalmente, a la introducción de un nuevo tipo de software conocido como *Building Information Modelling* (BIM). Este tipo de software está basado en la definición de modelos constructivos mediante parámetros de todo tipo. Mucho se ha escrito sobre este nuevo tipo de software y sus características, pero lo cierto es que su funcionamiento implica un cambio de pensamiento radical en lo relativo a la construcción en todos sus aspectos, no solo en la representación.

La introducción de este nuevo software desde mediados de la primera década de los 2000, sobre todo a partir del año 2007, ha significado una mejora en la industria de la construcción. El tratamiento mediante parámetros del modelo ha reducido los errores en la representación y ha abierto un nuevo mundo de posibilidades al poder acceder a los datos del proyecto con mucha más facilidad y en edades muy tempranas del desarrollo.

Esto ha potenciado la creación de diferentes herramientas que utilizan distintos parámetros para evaluar una gran variedad de factores que afectan al modelo, sobre las que hablaremos con más detalle en el apartado 1.5. En este contexto es donde se engloba la herramienta que constituye el elemento central de este trabajo, que pretende utilizar los parámetros estructurales para obtener el dimensionado óptimo en un elemento tipo viga.



## 1.2 OBJETIVOS DEL TRABAJO

### 1.2.1 Objetivos

En los siguientes apartados vamos a mostrar que la tecnología BIM y todo lo relativo a la misma es muy novedoso y está en continuo desarrollo. Hoy en día, se desconocen los límites a los que puede llegar esta tecnología (Azhar, 2011). Para no perder de vista lo que se pretende con esta investigación, se enumerarán todos los puntos que se quieren estudiar, siendo limitados adecuadamente. Una limitación que se hace necesaria, pues de lo contrario se convertiría en una investigación que excedería las características de un trabajo como este, únicamente pretende establecer un primer marco de investigación en el entorno BIM. A continuación se listan los objetivos del trabajo que serán explicados en mayor detalle en los subsiguientes apartados:

- 1) Estudio de viabilidad de la implantación de algoritmos de optimización al entorno BIM en el ámbito de las estructuras de edificación.
- 2) Definición del entorno BIM y estudio de sus posibilidades.
- 3) Acercamiento bibliográfico sobre el entorno BIM.
- 4) Conocimiento del funcionamiento interno de los algoritmos genéticos (GA) en optimización.
- 5) Conocimiento sobre el programa BIM en que se aplicará el algoritmo.
- 6) Conocimiento y profundización en el uso del lenguaje C#.
- 7) Aprendizaje del funcionamiento de la *Application Programming Interface* (API) del programa Revit.
- 8) Desarrollo de una aplicación externa capaz de realizar un cálculo optimizado de una viga de hormigón armado isostática.
- 9) Refinamiento de los diferentes parámetros presentes en los GA mediante diseño de experimentos.
- 10) Establecimiento de relaciones matemáticas entre los distintos parámetros definitorios de la viga a través de estudio estadísticos.

### 1.2.2 Alcance de la investigación

Los objetivos arriba enunciados abarcan un campo muy amplio, por lo que ahora se explicará más detalladamente en qué consiste cada uno de ellos y hasta dónde se pretende llegar con los mismos. Con este fin se dividirán por área temática y de funcionamiento de los diferentes objetivos.

El primer grupo lo conforman los objetivos establecidos en los puntos del uno al cuatro. Este consiste en una revisión bibliográfica que nos permita conocer el marco general que engloba a los BIM, empezando por una definición del mismo y el estudio de sus ventajas. También el estado de desarrollo en que se encuentra y qué aplicaciones se han impulsado en la actualidad para estos entornos. Además, para completar los conocimientos necesarios, necesitaremos profundizar en los GA y fijar el pseudocódigo a utilizar.

La viabilidad del proyecto se estudiará a partir de esta búsqueda, que incluye la localización de las herramientas utilizadas por los distintos desarrolladores BIM y la verificación de la posibilidad de aplicar estas herramientas al ámbito de las estructuras de edificación y a los algoritmos de optimización mediante genéticos.

El siguiente conjunto de objetivos engloba los comprendidos entre los apartados cinco y ocho. Son aquellos relativos al desarrollo de una aplicación funcional, es decir todo aquello necesario para lograr la realización del octavo objetivo. Es en este grupo donde es necesario establecer unos límites, ya que es aquí donde podemos encontrar un mayor número de posibilidades que convirtiera el trabajo en inabarcable. En este apartado se cumple nuestro objetivo principal, el desarrollo de una aplicación que verifique que es posible aplicar la optimización estructural al entorno BIM, lo que se aleja de la creación de una aplicación a pleno funcionamiento de cálculo, diseño y optimización de todo tipo de estructuras.

La aplicación desarrollada será, por tanto, completamente funcional para vigas de hormigón armado de cualquier longitud e isostáticas sometidas a cargas de edificación. No está pensada para otro tipo de barras ni para una barra con otros vínculos externos, pues la diferenciación entre las distintas tipologías estructurales añade una complejidad no deseada. En el apartado 4 se mostrarán guías para superar estos límites e indicar hacia qué dirección puede evolucionar la aplicación. Se ha optado por barras isostáticas para evitar el momento negativo en los extremos que implicaría una doble comprobación de momentos que ralentizaría el algoritmo.

La aplicación de optimización multicriterio generada debe ser capaz de hacerlo según cuatro criterios: coste, emisión de CO<sub>2</sub>, consumo energético y número de barras utilizado. Los valores de los tres primeros se obtienen considerando el valor de los materiales y de su puesta en obra extraídos a partir de una dosificación estándar para cada tipo de hormigón como suma de los valores de cada uno de los componentes, estos datos han sido extraídos de la base de datos del *Institut de Tecnologia de la Construcció de Catalunya* (ITeC).

Para la programación de la aplicación se necesita investigar diferentes alternativas de comunicación con Revit a la hora de extraer y modificar parámetros, así como las diferentes

funciones presentes en la API que puedan sernos útiles a la hora de trabajar con estos parámetros.

El último grupo de objetivos engloba el estudio estadístico de los resultados obtenidos tras la aplicación del algoritmo con la finalidad de conocer e interpretar los que este produce, así como verificar que los mismos tienen fiabilidad. Esto responde a los dos últimos objetivos, números nueve y diez.

En total se van a realizar tres estudios diferentes. El primero determinará las interacciones que existen entre los cuatro criterios de optimización; el segundo estudiará la influencia de cada una de las variables que definen la viga en el resultado final; el tercero de ellos evalúa la sensibilidad de la optimización a la variación en el precio de cada uno de los materiales en diferentes intervalos de subida y de bajada. No se estudia la influencia de la variación de los otros criterios ya que no se prevé que pueda existir esta variación en la realidad. Todos estos estudios quedan explicados en detalle y desarrollados en el apartado 3.

Los anejos proporcionan un ejemplo en el que se utiliza el diseño de experimentos para establecer los valores de los diferentes parámetros que regulan el GA para garantizar la obtención de un óptimo.

Esta técnica estadística se basa en reducir el número de muestras necesarias para analizar las interacciones entre diferentes parámetros estableciendo unas combinaciones estratégicas de valores que permiten conocer cómo influye cada uno de los parámetros de estudio y como se interrelacionan entre ellos.

### 1.3 BIM

El concepto BIM, a pesar de ser conocido y muy utilizado en la actualidad, está lleno de enigmas y no se mundo conoce realmente en que consiste el entorno BIM. En las siguientes páginas esclareceremos sobre el concepto BIM y definiremos los usos que le daremos a lo largo de la investigación.

#### 1.3.1 Definición BIM

Para entender completamente qué son los BIM y sus capacidades, lo primero es encontrar una definición global sobre la que trabajar. En cuanto a definiciones directas, la más extendida y utilizada es la propuesta por la US National BIM Standard, que podemos encontrar en su página web.

*Building Information Modeling (BIM) is a digital representation of physical and functional characteristics of a facility. A BIM is a shared knowledge resource for information about a facility forming a reliable basis for decisions during its life-cycle; defined as existing from earliest conception to demolition.*<sup>1</sup>

*(FAQ About BIM, National BIM Standard - United States, 14/3/2016)*

Esta definición hace clara referencia a BIM como una plataforma de software, una creencia generalizada sobre BIM. Sin embargo, si la observamos atentamente, lo que en realidad está definiendo son dos conceptos distintos para las siglas BIM. Por un lado, la primera frase hace referencia a la representación en un entorno virtual, el *Building Information Modeling* estrictamente hablando. Por otro lado, la segunda oración crea una pequeña confusión, ya que define BIM más que como el software, como el recurso informático creado con ese software, el llamado *Building Information Model* comúnmente abreviado también como BIM.

Esto muestra la verdadera dificultad que presenta definir el concepto BIM. Es un concepto muy amplio que abarca más que el entorno informático con capacidad paramétrica que hemos visto hasta el momento. Por lo tanto, tenemos que buscar una definición más completa, que incluya lo que acabamos de señalar. Chuck Eastman propone la siguiente definición:

---

<sup>1</sup> *Building Information Modeling (BIM)* es una representación digital de las características físicas y funcionales de una construcción. Un BIM es un recurso compartido de información sobre una construcción que supone una base relevante para la toma de decisiones durante el ciclo de vida, desde su concepción hasta su demolición.

*"[...]we define BIM as a modeling technology and associated set of processes to produce, communicate, and analyze building models."*<sup>2</sup> Ch. 1 P.16

*(Eastman et al., 2011)*

Esta definición presenta una mayor exactitud frente a lo que nos encontramos al hablar de BIM, ya que presenta el abanico de posibilidades de esta nueva tecnología. BIM no es únicamente un software capaz de modelar, como nos dice la primera cita, sino que incluye los procesos asociados a producir, comunicar y analizar este modelado. En este aspecto radica el gran éxito del BIM, en que no es solamente una herramienta de representación, sino que lo es también de construcción digital con plena capacidad en todos los aspectos de que ello conlleva. Por tanto, un entorno BIM nos permite, de manera virtual, construir el edificio siguiendo el modelado paso a paso, analizando todos los elementos de esta construcción en cada uno de sus estadios (por ejemplo, extrayendo las partidas de obra en cada fase constructiva) y, además, comunicar el estado de la construcción y el resultado de estos análisis.

El *Building Information Modelling* ha supuesto un cambio de paradigma radical en la industria de la Ingeniería, Arquitectura y Construcción (AEC) (Azhar, 2011) debido a este motivo. Se ha producido un cambio de mentalidad respecto a todo aquello a lo que hemos tenido acceso hasta el momento, ya que dejamos de "representar" para "construir" con todas las ventajas e inconvenientes que ello supone y que iremos viendo más adelante.

El entorno BIM se constituye, por tanto, en el más adecuado para cumplir los requisitos del tipo de contrato en la industria AEC que más se está utilizando en la actualidad (Eastman et al., 2011), el denominado IPD (*Integrated Project Delivery*), que exige un control del proyecto durante todo el ciclo de vida del mismo.

El objeto generado en un entorno BIM se denomina *Building Information Model* o *Building Model* y aunque como hemos dicho anteriormente en la revisión bibliográfica se utilizan las siglas BIM para referirnos a él, en este caso, para evitar confusiones, utilizaremos las siglas BM. Es importante, según la definición anterior, definir las características de un BM. Un *Building Model* se caracteriza por lo siguiente, (Eastman et al., 2011):

-Está formado por diferentes componentes de edificación con representación digital, atributos que les identifican y reglas paramétricas que permiten manipular estos objetos de un modo inteligente.

---

<sup>2</sup> [...] definimos BIM como una tecnología de modelado y el conjunto de procesos asociados a la misma para producir, comunicar y analizar modelos de edificios.

-Estos objetos incorporan datos que explican cómo se comportan frente a procesos necesarios para realizar diferentes análisis (por ejemplo, análisis energético o análisis estructural).

-Toda esta información no es redundante y los cambios sobre la misma se reflejan simultáneamente en todos los casos afectados.

-Coordinación entre las vistas y los elementos del modelo de modo que la representación en todas las vistas se muestra de un modo coordinado y todos los cambios sean efectivos instantáneamente.

En resumen, podemos definir BIM como aquella tecnología capaz de generar *Building Models* y gestionarlos de una manera integral a lo largo de su vida útil, desde su concepción hasta su derrumbe. Esto incluye las capacidades de modificar, analizar y representar este modelo digital en cualquier momento, tanto de su desarrollo como de su ciclo de vida.

### 1.3.2 BIM vs CAD

En el apartado anterior hemos expuesto las capacidades de los BIM y hemos definido qué son para tener una visión más clara sobre qué tipo de herramientas estamos utilizando, pero realmente ¿qué cambios han introducido los BIM en la industria AEC? ¿Cómo afecta esto a nuestro modo de trabajar hasta la fecha con software CAD? ¿Es realmente superior BIM a CAD? Estos son los interrogantes que trataremos de resolver en este punto, enfrentando la nueva tecnología BIM, a la actual CAD, analizando las verdaderas capacidades de cada una.

Para ello empezamos explicando brevemente en qué consiste el CAD. El *software* de *Computer Assisted Design*, más conocido como CAD, consiste según SARCAR et al. (2008) en un *software* basado en la interacción gráfica 2D y 3D con el ordenador para crear, transformar y mostrar datos de manera gráfica mediante símbolos o imágenes. En este, el usuario interactúa con el programa informático mediante comandos para crear estas imágenes. En la mayoría de ocasiones la imagen se construye a partir de elementos geométricos básicos, puntos, líneas y circunferencia. Estas imágenes pueden ser modificadas *a posteriori* por el usuario con las correspondientes subrutinas. Este tipo de *software* goza de gran aceptación entre los profesionales de la industria AEC tras su expansión en la década de los 90. El programa CAD más utilizado en el panorama nacional es AutoCAD, desarrollado por la empresa Autodesk.

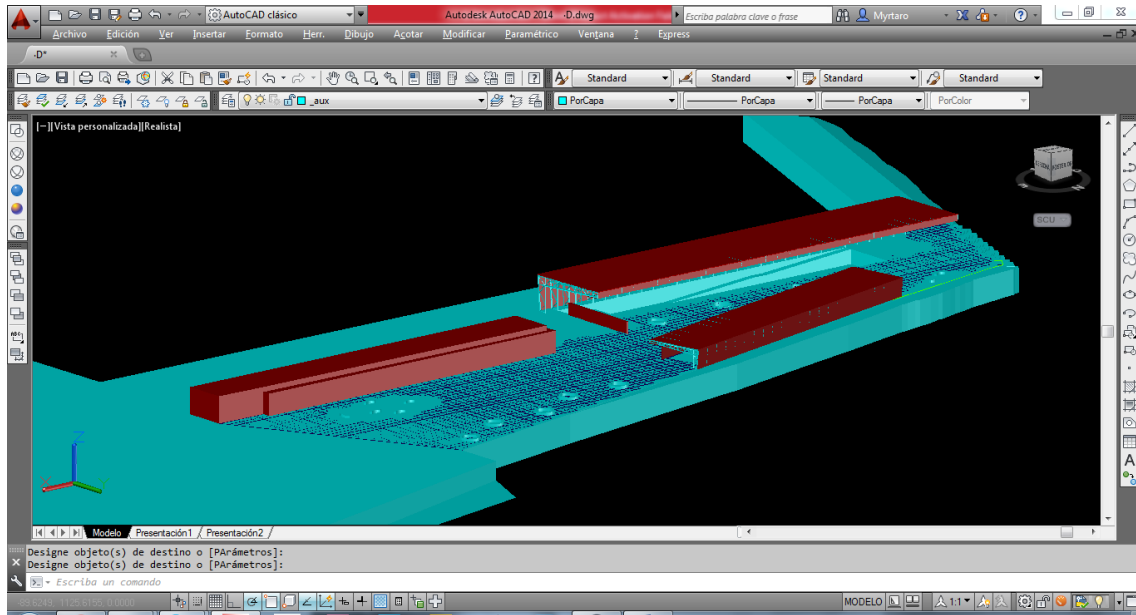


Fig. 1 Modelo 3D realizado con AutoCAD e interfaz del programa

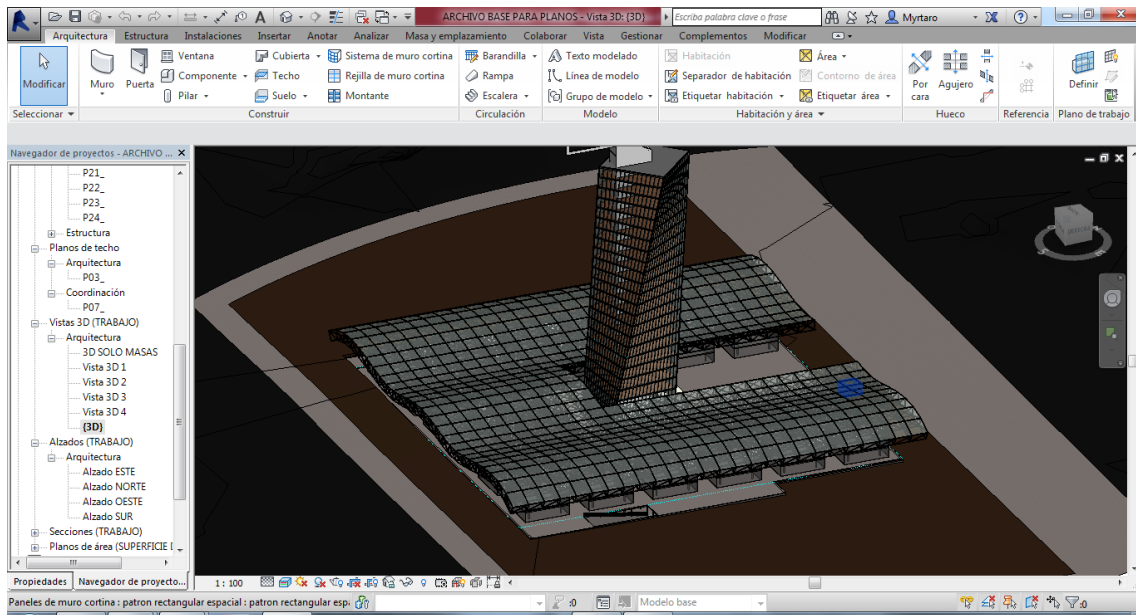


Fig. 2 Building Model realizado con Revit e interfaz del programa

La finalidad de los programas CAD está relacionada con la representación gráfica del objeto, ya sea en un entorno 2D, ya sea 3D, mientras que, como hemos visto, la finalidad de los programas BIM está relacionada con la construcción del edificio a lo largo de su vida útil. Ello demuestra que los programas no son para nada equivalentes y es prácticamente un error el enfrentarlos, ya que, a diferencia de lo que puede parecer a primera vista, los BIM distan de ser una evolución directa de los CAD.

La interacción para los usuarios también es completamente diferente. Por un lado, la utilización de los CAD se basa en la introducción, por parte del usuario, de una serie de comandos que

generan líneas o geometrías simples, cuya interacción (mediante comandos de unión, corte, movimiento o simetría) es la que genera la vista deseada, es decir, un conjunto de líneas ordenadas de tal modo que representan un objeto real. Por otro lado, la utilización de los BIM es más cercana a la experiencia de la construcción, ya que el usuario no crea líneas independientes, sino que estas “líneas” llevan una serie de parámetros asociados que se relacionan con el tipo de objeto que se está insertando en la vista. Este objeto conoce su función y cómo el usuario quiere que se represente a través de estos parámetros.

Es cierto que en los últimos años los objetos de los programas tipo CAD, como por ejemplo AutoCAD, han empezado a incluir parámetros, pero estos son relativos a su representación y caracterización dentro del propio programa, como son el color, grosor de línea o posición dentro del espacio de trabajo. En cambio, los parámetros de un programa BIM trascienden más allá, ya que definen al objeto y le confieren todas sus características, como el material o su fase de construcción. Además, los parámetros en un BIM crean interacción entre los distintos objetos actualizándose los unos a los otros según lo necesiten, mientras que los parámetros de un CAD son totalmente independientes para cada uno de los objetos. Esta diferencia queda patente al comparar las nomenclaturas de los parámetros disponibles en cada uno de los programas, como muestra la siguiente imagen:

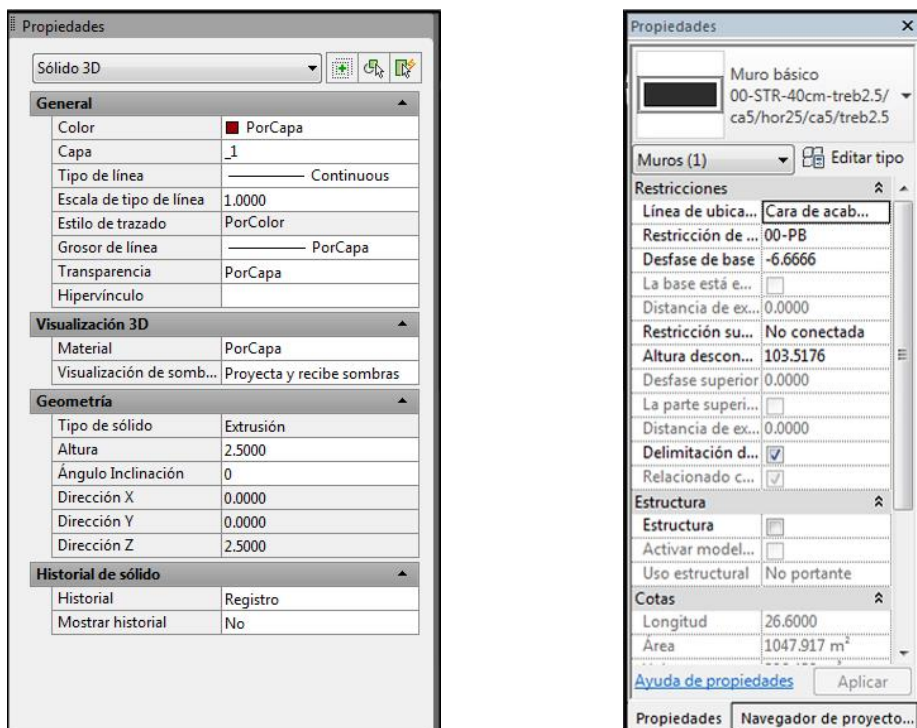


Fig. 3 Izquierda cuadro de propiedades de AutoCAD/Derecha cuadro de propiedades Revit.

A la izquierda podemos ver un panel de propiedades de una línea en el programa AutoCAD (CAD). A la derecha un panel de propiedades en Revit (BIM). Ambos desarrollados por la



compañía Autodesk. No hay que entender los CAD como algo inferior a los BIM, pues como se ha visto, existen diferencias sustanciales entre los objetivos de cada uno de los tipos de *software*. Aun así, es cierto que los CAD son anteriores a los BIM en cuanto a su desarrollo, lo que ha marcado algunas de las limitaciones del primero respecto al segundo.

A pesar de todo esto, en la actualidad y en lo relativo a la industria AEC, existe una clara superioridad en cuanto a productividad para el usuario mediante el uso de los programas BIM frente a los CAD, a parte de otras ventajas propias que estudiaremos en los próximos apartados. Esto es debido, principalmente, a que los programas BIM permiten controlar el modelo del dibujo mientras que los CAD solo una de las vistas del mismo.

En definitiva, podemos concluir que los *software* tipo CAD son el equivalente en un soporte informático al dibujo de planos por separado, mientras que los *software* tipo BIM están más cercanos a la construcción, en un entorno virtual, de un edificio.

### 1.3.3 Historia de los BIM

Muchos de los usuarios de los programas BIM creen que es una “nueva tecnología”, ya que, como nos indica MacDonald (2015) el uso del término BIM ha sufrido un “efecto bola de nieve” en los últimos años, especialmente en la última década. De hecho, existen referencias que afirman que el concepto BIM fue acuñado en el año 2002-2003 gracias a Jerry Laiserin, algo bastante alejado de la realidad, como veremos a continuación.

El primer hito temporal en cuanto al concepto BIM que encontramos en la bibliografía científica se sitúa en el año 1975, hace más de 40 años (Eastman, 1975). Chuck Eastman<sup>3</sup> plantea un prototipo funcional de lo que él denomina *Building Description System* (BDS), consistente en una única base de datos, para análisis visuales y cuantitativos, de diseño paramétrico. Además, ya anunciaba que “*Contractors of large projects may find this representation advantageous for scheduling and materials ordering*”<sup>4</sup>, una de las grandes ventajas y revoluciones que han supuesto los BIM. Este prototipo fue el germen del término BIM actual.

Este concepto BDS, anunciado por C. Eastman, evolucionó de manera diferente en EEUU, donde se denominó *Building Product Models*, y en Europa, donde se denominó *Product Information Model* durante el resto de los años 70 y los 80. La unión de estos términos acuñó, posteriormente, el término BIM. Este desarrollo también originó los *software* CAD, que

---

<sup>3</sup> Chuck Eastman, puede ser reconocido como la mayor autoridad en el mundo respecto a *Building Model*. Ha estado trabajando en este tema desde los años 70 en distintas universidades como UCLA Carnegie-Mellon (Eastman et al., 2011, p. viii)

<sup>4</sup> Contratistas de grandes proyectos pueden encontrar esta representación ventajosa para determinar horarios y pedidos de materiales. (Eastman, 1975)

perdieron parte de las funciones paramétricas anunciadas por C. Eastman debido a las limitaciones de los equipos informáticos del momento, que eran incapaces de manejar una base de datos tan extensa.

No fue hasta el año 1992 cuando se registró por primera vez el término *Building Information Modeling* en un artículo escrito por van Nederveen and Tolman (1992) publicado en Holanda.

En el año 1997 se fundó la compañía *Revit Technology Corporation* con la finalidad de desarrollar una herramienta informática tipo BIM plenamente funcional para el usuario. Esta compañía fue adquirida en 2002 por Autodesk<sup>5</sup>, con la siguiente finalidad según el comunicado de prensa anunciando la compra (Autodesk Inc., n.d.):

*Our combined technologies will be a catalyst for the worldwide building industry to adopt model-based design, construction and management," said David Lemont, Revit Technology CEO. "We can finally fulfill the longstanding desires of building professionals to achieve real process change. Building professionals will have the ability to integrate an intelligent building model into their practice and begin to use architectural design in downstream applications."*<sup>6</sup>

Ese mismo año, Jerry Laiserin <sup>7</sup>publicó un artículo titulado *Comparing Pommés and Naranjas* (Laiserin, 2002) mediante un *Autodesk White Paper* y en su propia página web *The LaiserinLetter*. Hoy, este artículo, se puede consultar en su página web. En él, exponía la necesidad que venía mostrando la industria AEC de ir más allá respecto al software CAD y cuyo primer paso para lograrlo sería otorgarle un nombre, lo que sugería el término BIM para este "nuevo" software y afirmaba que era un término sobre el cual estaba trabajando la compañía Autodesk.

Un año después, el 3 de Abril de 2003, J.Laiserin participó en lo que es conocido como *The Great Debate-BIM* (Laiserin, 2003). Un cara a cara entre Autodesk y Bentley System Inc.<sup>8</sup> dónde

---

<sup>5</sup> Autodesk es un líder mundial en software de diseño 3D para entretenimiento, recursos naturales, fabricación, ingeniería, construcción e infraestructuras civiles. ("Autodesk | Software de diseño, ingeniería y entretenimiento 3D," [www.autodesk.es](http://www.autodesk.es))

<sup>6</sup>"Nuestras tecnologías combinadas serán el catalizador para que la industria mundial de la construcción adopte los diseños basados en modelos, construcción y gestión" afirmó David Lemont, *Revit technologies CEO*. "Podemos finalmente conseguir los deseos de los profesionales de la construcción de conseguir un proceso de cambio real. Los profesionales de la construcción tendrán la habilidad para integrar un modelo de edificio inteligente en su práctica y empezarán a utilizar el diseño arquitectónico en las aplicaciones."

<sup>7</sup> Jerry Laiserin, analista en la industria AEC especializado en tecnologías futuras para el mundo de la construcción y en tecnologías colaborativas para trabajos basados en proyectos. Jerry ayuda diseñadores, constructores y propietarios a realizar decisiones de negocios. En los últimos 20 años ha publicado multitud de artículos que han ayudado a dar forma a la industria AEC actual. ("The LaiserinLetter", <http://www.laiserin.com>)

<sup>8</sup> Bentley System Inc. compañía desarrolladora de software especializada en solucionar las necesidades profesionales de los responsables de crear y gestionar las mayores infraestructuras del mundo. ("Bentley Systems, Incorporated," [www.bentley.com](http://www.bentley.com))

debatieron sobre sus visiones del BIM. Se puede encontrar una grabación de este debate en la plataforma de internet Youtube<sup>9</sup>.

Por estas dos intervenciones se le atribuye a J. Lairserin el título de *“father of BIM”* por muchos autores, aunque como él mismo afirma en su presentación a la primera edición de el *BIM Handbook* (Eastman et al., 2008):

*rather than “Father of BIM” – as a few well-meaning but over-enthusiastic peers have labelled me – I prefer the unattributed epithet “godfather of BIM”, in the sense that a godfather is an adult sponsor of a child not his own. If anyone deserves the title “father of BIM”, surely it is Chuck Eastman<sup>10</sup>*

Conocer la historia de los BIM nos hace darnos cuenta de que no es algo surgido de la noche a la mañana, sino que se trata de una meta conseguida tras casi medio siglo de búsqueda por parte de sus creadores. Hay muchos puntos de la historia que se han dejado fuera de esta pequeña retrospectiva, en la que sólo se han querido mostrar los hitos clave, pero aún así queda patente que la intención inicial de buscar un software que ayude a la industria AEC a reducir errores y a aumentar la colaboración entre los distintos agentes, es la esencia del paradigma BIM.

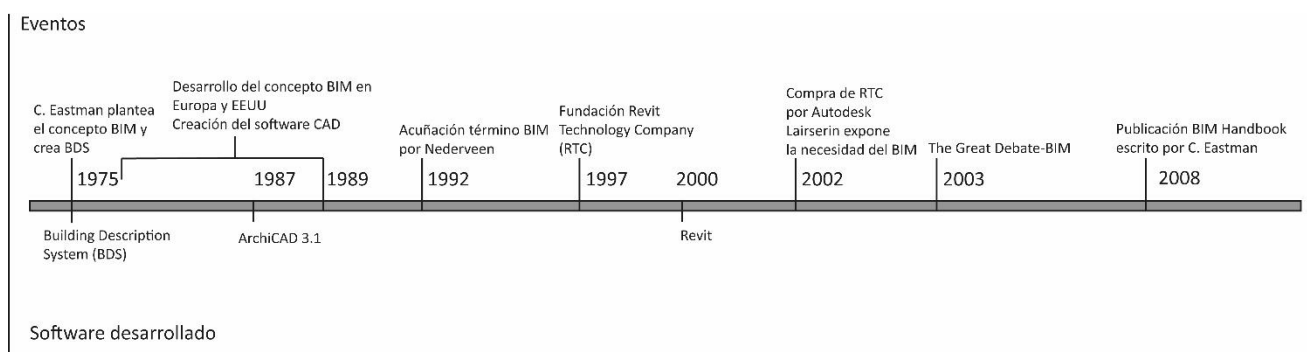


Fig. 4 Línea temporal historia de los BIM

### 1.3.4 Ventajas e inconvenientes de BIM

Es de suponer que la gran aceptación de los BIM por los profesionales de la industria AEC se debe a que posee una serie de ventajas frente a otro tipo de plataformas (por ejemplo los CAD), ventajas que se han ido intuyendo en los apartados anteriores. Pero debemos preguntarnos, ¿cuáles son las capacidades de los BIM?, ¿frente a qué retos y límites se encuentra

<sup>9</sup> The Great Debate- BIM: <https://www.youtube.com/watch?v=ILWHaDZH4d4>

<sup>10</sup> “mas que “Padre del BIM”, como unos pocos bienintencionados pero sobre-entusiastas me han etiquetado, prefiero el epíteto no atribuido de “Padrino del BIM”, en el sentido que el padrino es el patrocinador adulto de un niño que no es suyo. Si alguien se merece el título de “Padre del BIM”, tened por seguro que es Chuck Eastman.

esta tecnología? ¿Es realmente positivo todo lo que nos aporta? BIM representa un nuevo paradigma en la industria y por ello es necesario analizar sus ventajas e inconvenientes detenidamente para conocer sus verdaderas posibilidades.

Ya desde este momento podemos apuntar, según (Azhar, 2011), que la ventaja clave que tienen los BIM es la exactitud geométrica a la hora de la representación de las partes de un edificio en un entorno con datos integrados.. Lo que presenta como consecuencia otras ventajas derivadas de esta y que quedan expuestas a continuación:

- Mayor control en las etapas iniciales del proyecto. Gracias a la facilidad para controlar la cantidad de los materiales y de análisis del edificio en cualquier momento es muy sencillo realizar y controlar la rentabilidad de este.

- Incremento de la calidad del edificio, como en el caso anterior, la facilidad para analizarlo, nos permite controlar en todo momento si este cumple los requisitos exigibles.

- Mayor colaboración entre los distintos agentes del proyecto.

- Reducción de errores, tanto por la capacidad para realizar cualquier comprobación en cualquier momento, como por la corrección automática intrínseca al modelo cuando se han realizado cambios.

- Eliminación de errores en la representación, ya que todo surge de un mismo modelo 3D.

- Mejora de la eficiencia energética y la sostenibilidad del edificio debido al uso de las herramientas de análisis. Además distintos factores locales relacionados con el ciclo de vida pueden ser incorporados al modelo para mayor exactitud.

- Utilización de los modelos diseñados para prefabricación. Ya que los diseños de los elementos se han realizado en 3D, es posible implementarlos en modelos numéricos de producción de elementos de hormigón, vidrio u otros materiales.

- Reducción de costes por varias causas, mejora de la productividad, mejor control de los materiales, reducción de errores, etc.

- Acceso a herramientas de análisis que automaticen ciertos procesos relacionados con el diseño (análisis estructura, energético) e incorporen algoritmos de búsqueda de soluciones a partir de estos análisis aprovechando la accesibilidad a los datos del proyecto.

La mejora de los procesos en cada una de las fases del proyecto gracias a la utilización de los BIM es un factor que los hace verdaderamente atractivo. Sin embargo, el uso de los BIM también conlleva algunos inconvenientes, riesgos y, sobre todo, cambios a la hora de realizar el proyecto.

A continuación, se muestran los inconvenientes y las barreras más inmediatas que se encuentra el BIM a la hora de ser incorporado y aceptado plenamente (Eastman et al., 2011), (Azhar, 2011).

-Necesidad de renovación de equipos informáticos. La carga de los programas BIM sobre los equipos es tan fuerte que en muchas ocasiones será necesario renovar los equipos, además si se desea trabajar colaborativamente se necesitará instalar un servidor.

-Cambios en los contratos, los modelos de contrato actual que se utilizan en la industria AEC presentan algún problema de compatibilidad con la implementación de los BIM.

-Deficiencias a la hora de determinar la autoría del proyecto y de los cambios, ya que el proyecto está siendo realizado por los distintos agentes simultáneamente.

-En relación con el punto anterior surge la dificultad a la hora de cargar responsabilidades tras los errores, las deficiencias de diseño o las inexactitudes del mismo.

-Poca homogeneidad en los formatos de archivos, lo que provoca poca compatibilidad entre dos compañías que utilicen distintas plataformas BIM. Aunque se están realizando esfuerzos (y avanzando) en la eliminación de esta barrera y en la actualidad existe un formato de compatibilidad entre diferentes *softwares*.

-Cambio en el modo de funcionar de la compañía. Dado que los BIM son una tecnología tan distinta a las anteriores, las empresas necesitan una adaptación a la misma más allá de adquirir *software* y *hardware*, y formar a sus empleados. A cada empresa este cambio le afectará de distinta manera y se necesita evaluar independientemente.

Todas estas desventajas se basan en la idea del gran cambio necesario al adoptar BIM. Por ello C. Eastman (Eastman et al., 2011) recomienda que este se realice de manera progresiva para la firma, analizando en todo momento las deficiencias que se perciban en el uso de BIM. Hay que tener en cuenta siempre que la tecnología BIM es altamente sofisticada y que si no se implementa correctamente en la industria y se utiliza por personal sin conocimiento de su funcionamiento puede ralentizar el proceso. No por el mero hecho de utilizar BIM se van a aumentar los beneficios o incluso devolver la inversión (Sacks et al., 2010).

### 1.3.5 Diferentes plataformas BIM

Hasta el momento hemos hablado de los BIM en términos generales, refiriéndonos a ellos como una tecnología y un *software*. Pero no existe un único software denominado BIM, como se habrá deducido hasta este punto, sino que existen una gran diversidad de programas comerciales que hacen uso de este concepto.

Las primeras herramientas de software desarrolladas entre los años 70 y los 80 fueron productos destinados a estaciones de trabajo<sup>11</sup>. Entre estas herramientas encontramos, por ejemplo el *Building Description System (BDS)*, desarrollado por C. Eastman, así como GLIDE, RUCAPS, Sonata y Reflex. Todos estos distintos programas informáticos fueron los padres de los actuales BIM, aunque no llegaron a salir al mercado dado el gran coste computacional que suponía el utilizarlos.

Uno de los mayores problemas que han tenido los BIM a la hora de ser aceptados es la unificación de formatos para trabajar en distintas plataformas con el mismo archivo. En este aspecto, la asociación BuildingSMART<sup>12</sup> ha realizado grandes esfuerzos para la estandarización de un formato abierto para BIM, el conocido como *Industry Foundation Classes (IFC)*.

En la actualidad existe una gran diversidad de programas BIM elaborados por distintas compañías y con distintas finalidades que se usan en mayor o menor medida por los distintos profesionales del sector. Uno de los más utilizados en el ámbito de edificación es Autodesk Revit, propiedad de la empresa Autodesk y en el mercado, con versiones anuales, desde el año 2000 (cuando era propiedad de *Revit Technology Corporation*). En cambio, el programa con más antigüedad hasta la fecha es ArchiCAD de la empresa Graphisoft, que actualmente tiene 19 versiones y está en el mercado desde el 1986 con ArchiCAD 2.0. Aunque existe una gran variedad de ejemplos más, solo se nombrarán algunos de ellos como Navisworks, MicroStation, AllPlan o VectorWorks Architect.

Además, dada la gran aceptación de los BIM en general en la industria AEC algunos países han empezado a tener en cuenta los BIM de distintos modos. El Ministerio de Fomento de España publicó en Julio de 2015 una estrategia nacional para los BIM en la cual se estipula que será requisito indispensable para la construcción de un proyecto público su presentación en formato BIM para el año 2020 (Knutt, 2015). Este mismo requisito ya se exige en países como Hong Kong, Emiratos Árabes y Singapur desde 2015 y se exigirá en Reino Unido (Chi et al., 2014) en el año presente. En Malasia la implementación completa del BIM empieza en 2016 y se exigirá en 2020 ("Construction Industry Transformation Programme, CITP", Ministry of Works Malasya, 2015).

---

<sup>11</sup> En informática una estación de trabajo es un computador de altas prestaciones destinado para trabajo técnico o científico. Las estaciones de trabajo fueron un tipo popular de computadoras para ingeniería, ciencia y gráficos durante las décadas de 1980 y 1990. ("Estación de trabajo" Wikipedia, 2015)

<sup>12</sup> BuildingSMART es una asociación internacional sin ánimo de lucro cuyo principal objetivo es fomentar la eficacia en el sector de la construcción a través del uso de estándares abiertos de interoperabilidad sobre BIM (Building Information Modeling) para alcanzar nuevos niveles en reducción de costes y tiempos de ejecución y aumento de la calidad. ("La Asociación - BuildingSMART Spanish Chapter," 2016)

En Europa, aunque difiere el método según el país, existen asociaciones que abogan por la integración plena de los BIM. La asociación internacional con más importancia en un mayor número de países es BuildingSMART que incluye cada una de sus ramas nacionales (como la española). Estados Unidos es uno de los países donde la integración está más avanzada, ya que las grandes compañías con software BIM son estadounidenses. En la actualidad, están definiendo nuevos tipos de contratos para la industria AEC que suplan los defectos que poseen los contratos actuales respecto al uso de los BIM.

### 1.3.6 Autodesk Revit como entorno de trabajo

Como ya se ha dicho, el objetivo de este trabajo es crear una aplicación integrada dentro de un software BIM y como hemos visto existen diferentes programas comerciales, de entre los cuales es necesario escoger uno, dado que las diferencias entre ellos son demasiado grandes como para desarrollar una aplicación compatible con todos ellos. En nuestro caso, hemos optado por utilizar Autodesk Revit por los motivos que a continuación se explican. A pesar de esta decisión se pretende, en la medida de lo posible, que la aplicación este cerca de ser funcional en cualquier entorno BIM.

Se pretende que la aplicación desarrollada sea funcional, útil y accesible al público, por lo tanto se ha optado por la plataforma más utilizada en la actualidad. Además Autodesk tiene habilitada una AppStore a la que los usuarios pueden añadir sus herramientas externas al programa, lo cual aumenta en gran medida la accesibilidad al *plugin* por parte de otras personas en un futuro.

Otro punto importante ha sido la necesidad, indispensable, de poder establecer una comunicación con el programa, tanto para la obtención inicial de datos como para el volcado final de los mismos. Autodesk Revit se convierte en la plataforma ideal para realizar esto, ya que el usuario tiene acceso a una *Application Programming Interface* (API de ahora en adelante) que le permite interactuar plenamente con el programa.

Autodesk Revit no es el único software BIM con esta característica, ArchiCAD por ejemplo también posee una API, si bien sí es el único con un programa de formación para el uso de la API, " *My First Plug-in Training - My First Revit Plug-in Overview*," (Autodesk, 2015) y una guía para el programador a través del *Revit Software Development Kit* (Revit SDK). Ambas fuentes de información, de libre acceso para cualquier usuario, así como la propia ayuda del programa, que también ofrece información acerca de la API ("Autodesk Revit Help," 2016). Existen otras herramientas para los desarrolladores que se han creado de manera independiente por los propios usuarios de la API, "AEC DevBlog" (2016). Frente a todo lo anterior, la API de ArchiCAD es de acceso restringido, y únicamente aquellos con permisos de programador por parte de la

compañía Graphisoft pueden acceder, lo mismo sucede con VectorWorks de la misma compañía.

Además de estas herramientas, Autodesk ha creado un punto de encuentro para programadores de Autodesk Revit y para sus desarrolladores a través de un foro en su página web<sup>13</sup>. Un punto de contacto para la solución de problemas en la programación y para el informe de errores de la misma.

En último lugar, cabe señalar que Autodesk Revit ha sido, así mismo, escogido por el fácil acceso que se da al programa y a todas sus características, por parte de la Universitat Politècnica de València, dado que existe un acuerdo entre dicha universidad y Autodesk, por el que los estudiantes tienen acceso a licencias gratuitas para estudiantes de todos sus productos sin ningún tipo de restricción.

### 1.3.7 Revit API y lenguaje de programación C#

La interacción de nuestra aplicación con el archivo de Autodesk Revit se realizará mediante las distintas librerías de funciones que conforman la API propia del programa. Es necesario conocer cómo funciona esta API, que los miembros de Autodesk describen como:

*Autodesk Revit provides a rich and powerful .NET API which can be used to automate repetitive tasks, extend the core functionality of Revit in simulation, conceptual design, construction and building management, and much more. Revit .NET API allows you to program with any .NET compliant language including VB.NET, C#, and managed C++.*<sup>14</sup> (“Autodesk - Autodesk Developer Network - Autodesk® Revit, Autodesk® Revit® Architecture, Autodesk® Revit® Structure and Autodesk® Revit® MEP,” 2015)

Es decir, se define como una poderosa herramienta capaz de automatizar tareas y expandir las funcionalidades de Revit. Esta API funciona plenamente en un entorno de programación en lenguajes .NET<sup>15</sup>, de entre los cuales se ha escogido el lenguaje C#, ya que tal y como se observará en la siguiente cita, es un lenguaje de programación orientado a objetos, lo que quiere decir que se basa en la definición de “objetos” que encapsulan información y operaciones dentro de él. Más adelante explicaremos qué objetos utilizaremos y sus operaciones internas.

<sup>13</sup> Autodesk Revit API Forum: <http://forums.autodesk.com/t5/revit-api/bd-p/160>

<sup>14</sup> Autodesk Revit proporciona una rica y poderosa .NET API que puede ser utilizada en procesos de automatización, extender las funcionalidades básicas de Revit en simulación, diseño conceptual, gestión de la construcción y del edificio y mucho más. Revit .NET API te permite programar con cualquier lenguaje que compile .NET, incluyendo VB.NET, C# y managed C++.

<sup>15</sup> .NET Framework es un entorno software desarrollado por Microsoft.



*“C# (pronounced “C sharp”) is the native language for the .NET Common Language Runtime. It has been designed to fit seamlessly into the .NET Common Language Runtime.[...] Because the Common Language Runtime is central to many things in C# [...]”*<sup>16</sup>(Gunnerson, 2001)El funcionamiento general de la API está basado en la llamada a los objetos de programación con los cuales Revit trabaja internamente y las funciones dentro de los mismos que almacenan los datos del *Building Model*. Es necesario profundizar en el conocimiento de las funciones rqueridas para nuestra aplicación con el fin de poder determinar qué funciones vamos a utilizar.

---

<sup>16</sup> C# (pronunciado “C sharp”) es el lenguaje nativo de *.NET Common Language Runtime*. Ha sido diseñado para encajar sin problemas dentro del *.NET Common Language Runtime*[...] Porque el *Common Language Runtime* es central a muchas cosas en C#.

## 1.4 ALGORITMOS GENÉTICOS

Hasta el momento, nos hemos centrado en describir el entorno BIM con todo detalle y en cómo podremos utilizarlo a nuestro favor, pero eso no es todo lo que necesitamos. Para realizar la aplicación explicada en los objetivos es necesario, además, conocer qué es la optimización y cómo aplicarla.

En las siguientes páginas trataremos de abordar este tema, explicando qué es la optimización y sus orígenes, así como el modo de aplicarla. También estableceremos los criterios de optimización que se van a utilizar y cómo se cuantifican.

1.4.1 Optimización

Según el Diccionario de la Lengua Española (ASALE, 2016) optimizar es: “Buscar la mejor manera de realizar una actividad.”(ASALE, 2016)

“La mejor manera” puede referirse a el mayor valor (optimizar los beneficios) o al menor valor (optimizar la ruta para que sea más corta). Hablando en términos matemáticos, esto significaría hallar el máximo o mínimo de una función objetivo, definida en un dominio. Esto depende del problema y sobretodo del criterio de optimización utilizado, ya que según el caso se busca el máximo o el mínimo. Si tenemos una función definida en un dominio podemos definir la

optimización de la misma como: 
$$\begin{cases} \text{minimizar } f(x) \\ \text{sujeto a} \\ g_i(x) \leq 0 \quad i = 1, \dots, m \\ x \in S \subseteq R^n \end{cases}$$

(Yepes, 2015)

La optimización matemática se conoce y se utiliza desde hace muchos años. El método más antiguo conocido de optimización es la optimización lineal, que utilizó Fourier en el siglo XIX para resolver distintas ecuaciones. Aunque el término surge durante la Segunda Guerra Mundial con el fin de reducir los costos (humanos y económicos) del ejército, por George B. Dantzig.

La programación matemática constituye un campo amplio de estudio que se ocupa de la teoría, aplicaciones y métodos computacionales para resolver los problemas de optimización condicionada. En estos modelos se busca el extremo de una función objetivo sometida a un conjunto de restricciones que deben cumplirse necesariamente. Las situaciones que pueden afrontarse con la programación matemática se suelen presentar en ingeniería, empresas comerciales y en ciencias sociales y físicas.(Yepes, 2015)

Los algoritmos de programación lineal están limitados a ecuaciones lineales que no suelen representar fielmente la realidad, sino que la simplifican, además es necesario que para la aplicación de estos algoritmos las variables sean discretas. (Víctor Yepes, 2014)

Esta limitación ha hecho necesaria la creación de distintos algoritmos de optimización por parte de los matemáticos para distintos usos. En la actualidad contamos con una gran variedad de algoritmos, como por ejemplo los algoritmos de *Local Search*, *Simulated Annealing*, *Genetic Algorithms*, por citar algunos de los más conocidos. Algunos de los algoritmos solo pueden ser utilizados con variables discretas, mientras que otros solamente son efectivos con variables continuas. Elegir el algoritmo adecuado para cada función objetivo es muy importante para obtener un resultado refinado. La siguiente imagen muestra un árbol con la clasificación de los distintos tipos de algoritmos:

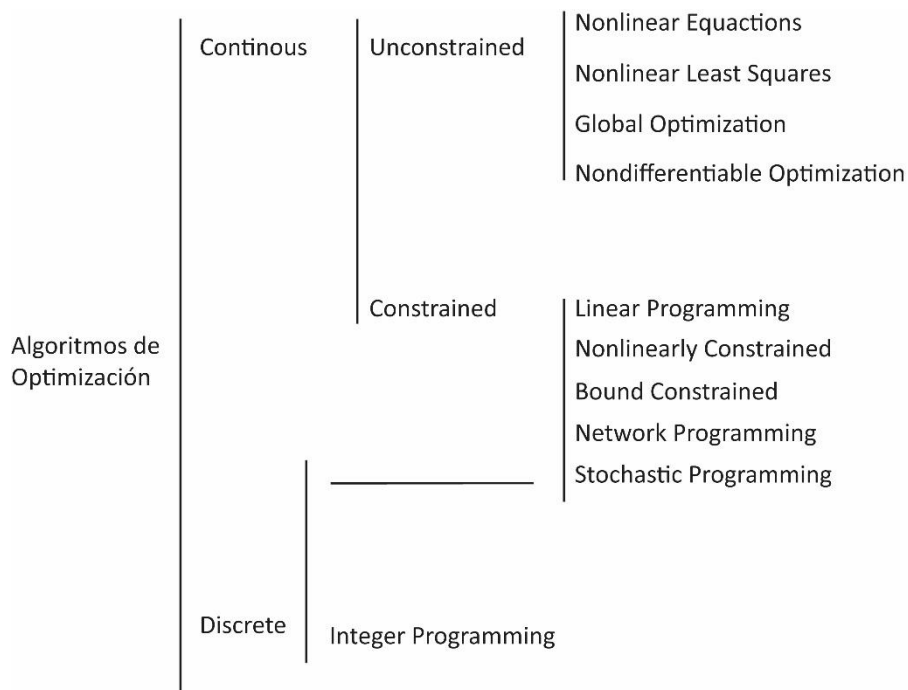


Fig. 5 Clasificación en árbol de diferentes algoritmos de optimización (Elaboración propia a partir de: Yepes, 2012, P.36)

Existe gran variedad de algoritmos de optimización, como los algoritmos basados en el *Simulated Annealing*, explicado en el capítulo 15 del libro “An Efficient Quasi-Human Heuristic Algorithm for Solving the Rectangle-Packing Problem.” (Wenqi Huang and Duanbing Chen, 2008); algoritmos basados en el apareamiento de las abejas (Haddad et al., 2006) o algoritmos basados en el emparejamiento de las libélulas (Miguel et al., 2013), por citar algunos ejemplos puntuales.

### 1.4.2 Optimización aplicada a estructuras de hormigón

La introducción de los algoritmos de optimización en la industria AEC ha supuesto una de las maneras de implantar los criterios del *Lean Construction* a la misma. Estos criterios que buscan como máxima mejorar la eficiencia en los procesos y los diseños de la construcción se pueden ligar íntimamente a los algoritmos, ya que estos permiten encontrar la mejor opción de entre las iversas estudiadas independientemente del número de combinaciones entre los factores posibles.

Uno de los puntos críticos de estos problemas de optimización es el gran número de variables a tener en cuenta, ya que el campo de soluciones posibles afectando directamente tanto a la eficiencia como a la eficacia del algoritmo. Este es uno de los factores decisivos a la hora de elegir el algoritmo a utilizar en concreto. La investigación de Payá et al. (2006) muestra esto al comparar la eficacia de cuatro algoritmos, *Threshold Accepting*, *Simulated Annealing*, Búsqueda Aleatoria y búsqueda por el gradiente, a un mismo problema, la optimización del coste de un pórtico de hormigón armado de cuatro alturas y dos vanos con un total de ochenta y un variables que definen su geometría. Los resultados en este caso muestran una superioridad del *Simulated Annealing* frente a los resultados obtenidos en los otros tres, quedando cerca los resultados obtenidos mediante *Threshold Accepting*. Una de las conclusiones de esta investigación es la ineficiencia del algoritmo de generación aleatoria de encontrar soluciones factibles (capaces de cumplir con los requisitos estructurales exigibles), ya que únicamente el 1% de las soluciones son válidas, debido en gran parte al gran número de combinaciones posibles. En la investigación realizada por Yepes et al. (2008) sobre el diseño óptimo de muros de contención en la que se utilizan un total de veinte variables de diseño no se hace referencia a este problema de ineficiencia, dado que el campo de soluciones es de un tamaño altamente inferior.

La aplicación de la optimización en estructuras no queda limitada únicamente a la búsqueda del menor coste, existen gran variedad de criterios de optimización. Siguiendo la investigación anterior de I. Payá encontramos que tras optimizar el pórtico en base al coste lo hizo en base a la emisión de CO<sub>2</sub>, (Paya-Zaforteza et al., 2009), consiguiendo así una estructura que consiguiera la menor cantidad de emisiones, lo que permite introducir los criterios de sostenibilidad en la fase de diseño de proyecto y que sea un condicionante con un gran peso en la toma de decisiones. Además se aumentó el número de vanos del pórtico, aumentando el número de variables y el problema de la ineficacia a la hora de encontrar diseños. Este problema queda solucionado implementando una generación de valores de los parámetros por horquillas en lugar de una generación totalmente aleatoria como en el caso anterior.

Durante los años se han ido creando y mejorando los algoritmos existentes siendo capaces de manejar un mayor número de variables manteniendo la eficiencia o de aceptar más de un criterio de optimización simultáneamente. De este modo se ha podido evaluar una misma estructura desde dos o más criterios simultáneamente, gracias a esto se han podido crear algoritmos de optimización capaces de encontrar el mwejor diseño para muros de contención (Yepes et al., 2012) y estructuras de hormigón de alta resistecia (Torres-Machi et al., 2013) en base a las emisiones de CO<sub>2</sub> y al coste simultáneamente o la optimización multicriterio de puentes basada en los dos criterios anteriores y en la cogestión de acero dentreo de la sección (Martinez-Martin et al., 2012).

En este último ejemplo hemos hablado del criterio de congestión, algo que hace referencia a la eficiencia estructural del acero, pero que también esta relacionado con la facilidad de construcción del puente. Es decir, los criterios que se pueden introducir en los algoritmos de optimización son muy diversos y solo tienen un requisito, estos han de ser cuantificables. Gracias a esto podemos encontrar una gran variedad de criterios como la energía consumida durante la construcción (Martí et al., 2016), el número de barras utilizado o lña minimización de los despuntes en las barras de acero(Porwal, 2012).

En los ejemplos mostrados en este apartado se han utilizado una gran diversidad de algoritmos diferentes en función, como se ha comentado, del problema, su número de variables y de los criterios de optimización deseados. Estas son las tres claves para lograr realizar con éxito la optimización de cualquier tipología estructural de hormigón armado deseada y corresponde con los puntos que analizaremos en los siguientes apartados.

#### 1.4.3 Objetivos de la optimización estructural

Para seleccionar el algoritmo adecuado es necesario conocer la función o las funciones objetivo que se analiza. En nuestro caso de estudio buscamos optimizar un elemento estructural isostático, una viga, de manera que tenga el menor coste, consumo de CO<sub>2</sub> y energético y número de barras posible y, lógicamente, sea capaz de soportar las cargas a las que está sometida. Por lo que, nuestro elemento estructural tiene que cumplir un total de cinco condicionantes:

1. Que tenga un coste mínimo
2. Que tenga una emisión de dióxido de carbono mínima
3. Que su consumo energético durante su fabricación sea mínimo
4. Que presente el menor número de barras de acero consumidas
5. Que sea una viga capaz de soportar los esfuerzos

Los cuatro primeros requisitos son los criterios de optimización que se utilizan en la optimización multicriterio, mientras que el quinto es una condición que se impone a cada uno de los especímenes para que sean considerados válidos. En (V. Yepes et al., 2015) encontramos las funciones objetivas de dos de los cuatro criterios, coste y emisión de CO<sub>2</sub> planteadas para una optimización multicriterio de estructuras de hormigón armado. Nosotros además añadimos dos criterios extra, de los que es necesario definir sus funciones objetivo. Las funciones objetivo que vamos a utilizar son las siguientes:

$$(1) \quad C(x) = \sum p_i \cdot m_i$$

$$(2) \quad E_{CO_2}(x) = \sum e_i \cdot m_i$$

$$(3) \quad I(x) = \sum i_i \cdot m_i$$

$$(4) \quad B(x) = \sum d_i$$

Donde:

$C(x)$  → Coste total del elemento

$P_i$  → Coste de cada uno de los materiales

$m_i$  → Cantidad de cada uno de los materiales

$E_{CO_2}(x)$  → Emisión de CO<sub>2</sub> total del elemento

$e_i$  → Emisión de CO<sub>2</sub> de cada uno de los materiales

$I(x)$  → Consumo energético total del elemento

$i_i$  → Consumo energético de cada uno de los materiales

$B(x)$  → Número de barras total del elemento

$d_i$  → Número de barras de cada uno de los diámetros

Buscamos, en nuestro algoritmo, hallar el valor mínimo de esta función. Los materiales incluyen el hormigón, el acero, el encofrado y la mano de obra.

No debemos olvidar la restricción ( $g_i$ ) que hemos definido antes, que la viga sea válida. La validez de la viga se verificará mediante las comprobaciones establecidas por la Instrucción Española de Hormigón Estructural-EHE (2008) para los siguientes estados:

-Estados Límites Últimos:

-Flexión (Artículo 42º)

-Cortante (Artículo 44º)

-Estados Límites de Servicio:

-Fisuración (Artículo 49º)

-Deformación (Artículo 50º)

Más adelante explicaremos cómo se han incorporado estas validaciones en el algoritmo utilizado.

#### 1.4.4 Elección del algoritmo a utilizar

Conocidas las funciones objetivo, tenemos un mayor número de datos para seleccionar el algoritmo de optimización más adecuado. Para ello vamos a analizar las posibilidades que nos ofrecen los distintos tipos de algoritmos y escogeremos el que más se ajuste a nuestras necesidades, que son las siguientes:

-Aplicación correcta de la función objetivo.

-Posibilidad de añadir restricciones.

-Velocidad de procesamiento.

Nuestro problema de optimización tiene, entonces, la siguiente estructura:

$$\begin{aligned} & \text{minimizar } C(x) \\ & \text{sujeto a } C(x) \geq 0 \\ & g_i = 1 \text{ }^{17} \end{aligned}$$

Las variables que componen la función objetivo son variables discretas y no continuas, ya que todas toman valores aislados determinados según el algoritmo de generación. Vamos a analizar el funcionamiento de distintos algoritmos del tipo *non-gradient*<sup>18</sup>, dado que son los más utilizados en la optimización estructural debido a sus mejores resultados (Hare et al., 2013). Estos algoritmos los podemos clasificar en cuatro grupos, algoritmos evolucionarios (*Evolutionary algorithms*), algoritmos físicos (*Physical algorithms*), algoritmos en enjambre o meméticos (*Swarm algorithms*) y algoritmos de búsqueda directa (*Direct Search algorithms*).

Los algoritmos evolucionarios utilizan técnicas que imitan la evolución natural. Generalmente siguen cuatro etapas, reproducción, mutación, recombinación y selección y una función de adecuación para determinar qué organismos sobreviven. Estos algoritmos fueron propuestos inicialmente en "*Adaptation in natural and artificial systems*"(1975). Versiones más modernas de estos algoritmos incluyen un paso más, donde varios organismos se emparejan y producen

<sup>17</sup> Siendo  $g_i=1$  si la viga es válida

<sup>18</sup> *Non-gradient algorithms*: algoritmos de optimización no basados en seguir la pendiente de la función objetivo, sino en evaluar directamente la función. Un ejemplo de estos serían los algoritmos genéticos.

una segunda población diferente a la inicial. Son los algoritmos más utilizados debido a sus buenos resultados en cortos periodos de tiempo de computación. Los evolucionarios son algoritmos que se pueden ajustar muy bien según el problema que nos encontremos, aunque para evaluar problemas a pequeña escala (poca variabilidad) pueden resultar, en ocasiones, aparatosos, dado que la muestra de datos necesaria para llegar a un buen resultado es elevada. Son más conocidos como algoritmos genéticos.

Existe una gran variedad de algoritmos físicos y, a diferencia de los evolucionarios, poco tienen en común los unos con los otros. Este tipo de algoritmos está basado en la recreación matemática de un fenómeno físico. Un ejemplo de estos algoritmos sería el *Simulated Annealing* (SA) nombrado anteriormente. Este algoritmo de Cristalización Simulada, emula los procesos de cristalización de distintos materiales en función de su temperatura, dado que los cristales al perder temperatura tienden a organizarse de una manera que requiera una menor energía. Otros ejemplos de este tipo de algoritmos serían: *Harmony Search*, *Ray Optimization*, *Tabu Search*... Estos no tienen ninguna convergencia matemáticamente hablando y tienden a quedarse estancados en mínimos locales. Estos algoritmos son especialmente útiles combinados con otros que encuentren soluciones locales de una manera más eficiente.

De esta idea surge el siguiente grupo de algoritmos, los algoritmos de enjambre o meméticos. Que están basados, generalmente, en una combinación de los dos anteriores que mimetiza el comportamiento de una colonia organizada de seres vivos basado en un modelo biológico. Usualmente en este tipo de algoritmos se lanza un primer algoritmo (algoritmo evolucionario) que encuentra un mínimo local sobre el que se realiza un segundo algoritmo (un algoritmo de búsqueda local o un algoritmo físico). Un ejemplo de estos algoritmos lo tendríamos en el *Honey-Bee Mating Optimization Algorithm* (HBMO) (Haddad et al., 2006), este se empieza combinando un organismo (abeja reina) con distintos organismos, seleccionados mediante un SA (zánganos) a modo de un GA y posteriormente los organismos resultantes (crías) son mutados por otros organismos (abejas obreras) mediante un algoritmo de búsqueda local. Finalmente la mejor de estas crías se convierte en reina y se reinicia el proceso hasta llegar a un resultado que supere la condición de adecuación. Otros algoritmos englobados en esta categoría son: *Ant Colony optimization*, *Particle Swarm optimization* o *Shuffled frog-leaping*.

El último bloque de algoritmos son los denominados algoritmos de búsqueda directa, los únicos algoritmos de la lista que matemáticamente convergen en un óptimo. El funcionamiento de estos está basado en la evaluación de la tendencia de crecimiento o decrecimiento de la función en varios puntos generados aleatoriamente. El método Simplex estaría dentro de estos algoritmos. Aunque estos algoritmos está demostrado que convergen en una solución, tienen



dificultades a la hora de trabajar con funciones objetivos de varias dimensiones (diversas variables), ya que el avance en cada una de las dimensiones hacia el óptimo se produce muy lentamente, lo que supone una gran cantidad de evaluaciones de la función objetivo. Por ello son poco utilizados en la optimización estructural.

En la bibliografía se puede encontrar un estudio más detallado sobre los distintos algoritmos y su uso y resultados en las distintas aplicaciones estructurales (Hare et al., 2013). El algoritmo que más se acopla a nuestros objetivos es el algoritmo genético (GA). El estudio anterior muestra que se obtienen buenos resultados en aplicaciones para vigas y hemos visto que los resultados se obtienen en poco tiempo de computación. Además, funcionan perfectamente con una o varias restricciones a la función objetivo. Por ello son los algoritmos que más se ajustan a nuestros propósitos.

#### 1.4.5 Funcionamiento de los genéticos

En este apartado vamos a explicar cómo funcionan los genéticos y cómo vamos a implementarlos en nuestra aplicación. Como se ha enunciado anteriormente, los algoritmos genéticos datan del año 1975, pero fueron perfeccionados en los años siguientes. Una de estas mejoras fue la iniciación del algoritmo mediante una población válida (Hare et al., 2013).

Esta población inicial es la generación 1 del algoritmo. A partir de este punto el algoritmo selecciona aleatoriamente unos progenitores y recombina sus genes para obtener un hijo, también válido a todas las restricciones. Este proceso de emparejamiento se repite tantas veces como descendientes sea necesario generar. Una vez generados todos, se escogen los

```

start
-Generación población inicial
while (generación != generación_max)
  for (descendiente= 1:max_descendientes)
    -Empareja los progenitores
    -Recombina sus genes hasta generar un descendiente válido
    -Mutación del espécimen generado(possible recuperación de un espécimen
  end
  -Selección de los mejores especímenes
end
end

```

Fig. 6 Pseudocódigo del algoritmo genético

especímenes que más se adecuen a la función objetivo y se desechan el resto para iniciar la siguiente generación con los especímenes supervivientes. El algoritmo se detiene cuando ha llegado a un número determinado de generaciones especificado al inicio del mismo.(Yepes, 2012)

El pseudocódigo expresa el funcionamiento general del algoritmo, pero es necesario introducir algunos matices en el mismo (que estudiaremos con más detalle al desarrollar el código) para garantizar el buen resultado del algoritmo. Uno de estos matices es evitar el denominado incesto, es decir que los dos progenitores de un descendiente no sean el mismo organismo. Esto eliminaría variabilidad en el algoritmo y ralentizaría la llegada a una solución óptima.

Para evitar que el algoritmo se quede estancado en un mínimo local de la función objetivo se ha introducido el factor de “mutación” que introduce la posibilidad de que uno de los genes del descendiente no provenga directamente de los progenitores sino que se genere independientemente. Además existe una posibilidad de que todo el descendiente sea generado independientemente de los progenitores, a este evento le denominaremos “hijo pródigo”(Yepes, 2012).

Cada emparejamiento entre progenitores solo generará un descendiente. Aunque en problemas complicados es preferible generar dos descendientes por cada emparejamiento, dado que se necesitarán (potencialmente) más evaluaciones de una función objetivo que puede necesitar gran tiempo de computación, en este caso la función objetivo no precisa de un gran esfuerzo de cálculo y es preferible conseguir más variabilidad en los distintos organismos.(Yepes, 2012)

El algoritmo de selección a utilizar será un algoritmo de régimen elitista. Solo los especímenes que mejor se adapten entre el conjunto de población más sus descendientes serán seleccionados para formar parte de la nueva generación. No se realizarán distinciones sobre si el espécimen ha sido generado en la última iteración o proviene de una iteración anterior. De este modo se asegura que siempre se queden en el algoritmo los mejores algoritmos y la solución converja hacia el mínimo. Durante la optimización monocriterio este método se aplica de manera directa sobre el criterio de optimización más influyente, punto que necesita ser estudiado. En la aplicación definitiva con optimización multicriterio no tiene sentido dar preferencia a un criterio sobre el resto, por lo que es necesario buscar una manera de adaptar este criterio. Se ha optado por aplicar el criterio elitista utilizando la Frontera de Pareto, conjunto de mejores soluciones combinación de varios factores no directamente proporcionales entre ellos, y la introducción de un criterio de compromiso entre los diferentes criterios de optimización para lograr seleccionar en cada generación los especímenes que más se adapten a él.

Todo el algoritmo lo podremos controlar mediante una serie de parámetros que serán accesibles al usuario. Estos parámetros son el número máximo de generaciones, el número de especímenes de la población inicial, el número de descendientes, la probabilidad de mutación de cada gen

generado y la probabilidad de hijo prodigo. Estos parámetros tendrán que ser ajustados para cada problema en función de las variables de entrada al algoritmo. En este caso las variables de entrada para nuestra función objetivo son, la luz de cálculo, la carga gravitatoria de la barra, la carga variable de la barra y el recubrimiento nominal, además de las variables geométricas que definen las dimensiones y los materiales de la viga. Estas variables son los datos necesarios para que funcione el algoritmo de validación de la viga y que los organismos generados mediante el algoritmo genético sean siempre soluciones con suficiente capacidad resistente.

## 1.5 BIM Y OPTIMIZACIÓN

Los BIM plantean un mundo nuevo de posibilidades en el mundo de la AEC. La mayoría de estas posibilidades están aún por explorar y existe un largo camino por recorrer (Sacks et al., 2010). En las siguientes páginas expondremos algunas de estas aplicaciones posibles de los BIM a las que apuntan distintos autores. Además estudiaremos como plantear la incorporación de algoritmos genéticos para la optimización de estructuras de hormigón armado en un entorno BIM.

En los últimos años, las entradas en las bases bibliográficas que contienen el término “BIM” o “*Building Information Modeling*” se han incrementado de manera progresiva. Analizando el gráfico 1 se observa que el corpus aumenta cada año. Aunque en 2015 el incremento fue menor se superaron los mil artículos relativos al tema. Esto demuestra que es un tema de actualidad con proyección hacia el futuro.

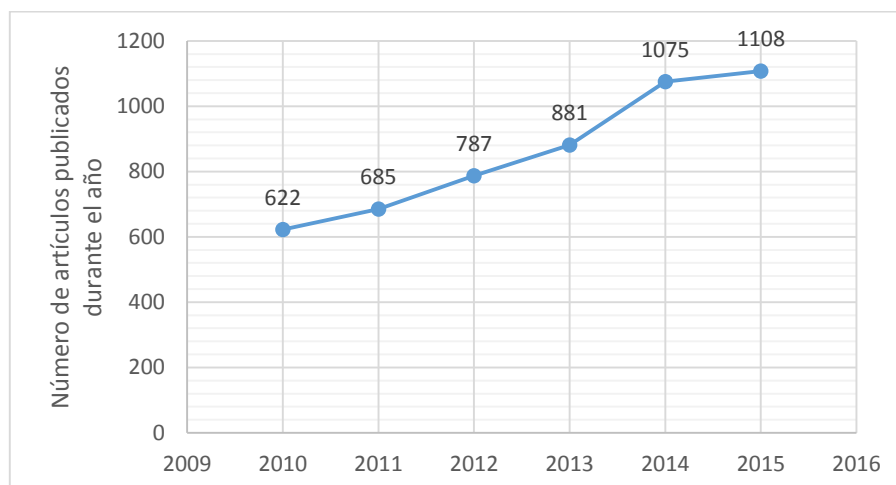


Gráfico 1 Número de entradas relacionadas con BIM en la base bibliográfica Scopus

La búsqueda anterior nos muestra todo lo relativo al campo de los *Building Information Modeling* pero este es un campo tremendamente amplio y exento que engloba nuestro objetivo a la vez que muchos otros. En nuestro caso nos interesan aplicaciones de optimización en un entorno BIM. Restringiendo la búsqueda a estos dos términos (BIM y Optimización) obtenemos el gráfico 2. Lo primero que se observa es el menor número de respuestas de esta búsqueda, ya que esta devuelve un total de sesenta y un artículos en los últimos cinco años, lo que demuestra la diversidad de aplicaciones del paradigma BIM. Tras esto observamos que, como en el caso anterior el número de publicaciones es ascendente, aunque existe un pico en el año 2011. Este incremento repentido es debido a la influencia de la publicación del libro “*BIM handbook : a guide to building information modeling for owners, managers, designers, engineers, and*

*contractors*” de C. Eastman en 2009, la gran referencia sobre el paradigma BIM cuya segunda edición fue lanzada al público en 2011 por su gran demanda en dentro del sector AEC.

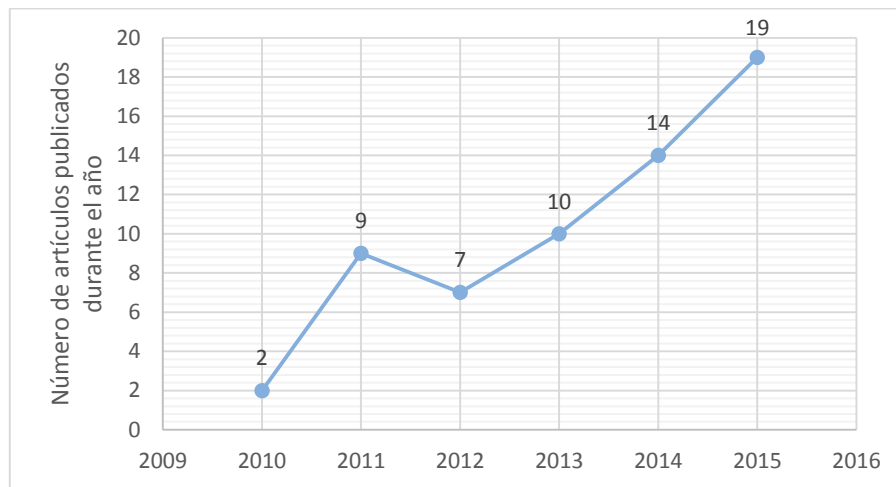


Gráfico 2 Número de entradas relacionadas con BIM y Optimización en la base bibliográfica Scopus

El gráfico 3 nos muestra una búsqueda bibliográfica mucho más restringida que la anterior, en esta hemos añadido el término Estructura a los anteriores, es decir estamos buscando investigaciones que versen sobre optimización estructural aplicada a un entorno BIM. Sorprende en este caso que no existan publicaciones relativas a estos temas anteriores a 2010, aunque se explica gracias al desarrollo de nuevos algoritmos de optimización a partir de la década de los 2000 y a la gran producción de material sobre BIM que hemos visto que se ha producido en los mismos años. Esta línea de investigación la esbozó C. Eastman en su libro en 2009 y es a partir de este momento cuando empezamos a ver una aplicación. No existen una gran cantidad de publicaciones y analizándolas más en detalle observamos que en general únicamente disparan ideas sobre como unir estos tres conceptos sin llegar a una aplicación real. En los siguientes apartados vamos a realizar una revisión de estos y otros artículos relacionados para conseguir la información necesaria para abordar el correcto planteamiento de una aplicación de optimización estructural integrada en un entorno BIM de manera exitosa.

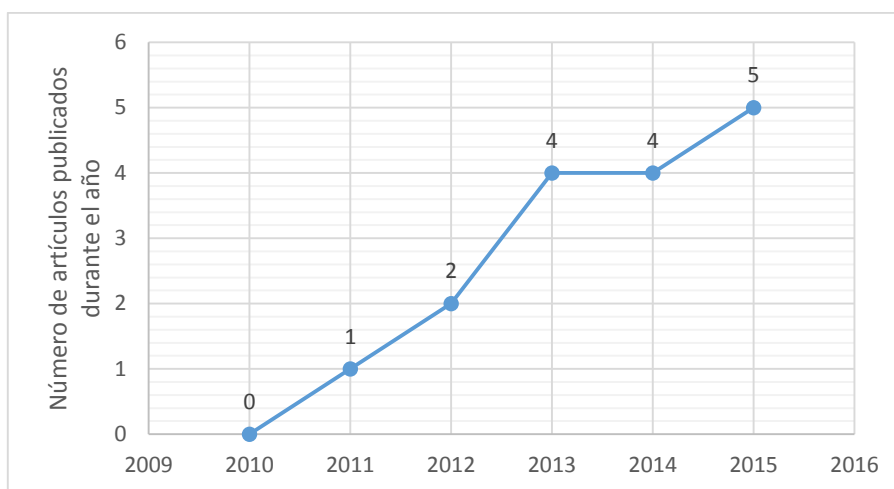


Gráfico 3 Número de entradas relacionadas con BIM y Optimización Estructural en la base bibliográfica Scopus

### 1.5.1 Tendencias futuras de los BIM y sus posibilidades

Llegados a este punto es innegable afirmar que los BIM están cambiando la manera en la que se proyectan los edificios, la manera en la que se construyen y la manera en la que se utilizan. ¿Hasta qué punto afecta este cambio? Es la pregunta que se realizan varios autores sin encontrar una respuesta unívoca. Existe una gran cantidad de posibilidades por explorar en lo relativo a los BIM y varios caminos por los que estos pueden evolucionar, como afirma C. Eastman: “This is an exciting time to be an architect, an engineer, or any other AEC industry professional.”<sup>19</sup>(Eastman et al., 2011, P.352)

Esta afirmación nace justamente de la gran cantidad de cambios que van a producirse en esta época gracias a la incorporación de los BIM y de técnicas de *Lean Construction*<sup>20</sup> a la industria de AEC. (Sacks et al., 2010). Estos dos conceptos revolucionarios van a introducir una gran cantidad de cambios, ya sea afectando a la industria de manera conjunta o por separado.

En cuanto a las posibilidades de los BIM, en el *BIM: Handdbook* de C. Eastman et al. (2011) hay dedicado un capítulo entero a esta cuestión. En este capítulo se muestran interesantes posibilidades de evolución de los BIM a lo largo de los próximos años. Las tendencias indicadas en el libro hasta el año 2015 (apartado “Current Trends”) se han ido cumpliendo en mayor o menor medida, signo inequívoco de la evolución de esta tecnología.

<sup>19</sup> Este es un excitante momento para ser un arquitecto, ingeniero o cualquier profesional de la industria AEC.

<sup>20</sup> Lean Construction: Aplicación y adaptación de los conceptos del *Toyota Production System* a la construcción con la finalidad de reducir el malgasto de material, incrementar el valor del producto y una mejora continuada del sector. (Sacks et al., 2010)

Pero las posibilidades anunciadas por Eastman no acaban en 2015, ya que como se afirma: *“The ready availability of BIM platforms will encourage a new wave of plugins that will emerge during the next five years.”*<sup>21</sup> (Eastman et al., 2011, P. 373)

Es de esperar que las funcionalidades internas del programa aumenten. Por ejemplo, es previsible la creación de distintas herramientas de análisis integradas en el mismo software, ya que la competencia entre las distintas desarrolladoras dará lugar a que aumenten las funcionalidades con este tipo de herramientas. Otra de las ampliaciones de usos posibles es la creación de rutinas automáticas de desarrollo de detalles constructivos en 3D capaz de compatibilizar los parámetros de dos o más elementos que influyen en una intersección.

También existe camino hacia la expansión de los BIM más que hacia una evolución interna. Nos referimos a la mejora de gestión de archivos desde la nube en Internet. Esto mejoraría en gran medida a los BIM: por un lado, se evitaría la dificultad de compartir información a través de los grandes archivos que suponen los BIM y, por otro, podría permitir el trabajo simultáneo de varios agentes. Además se podría coordinar con herramientas de visionado de los modelos que permitieran a agentes externos a la creación del proyecto (como los clientes) observar los cambios y opinar sobre los mismos.

El trabajo sobre la nube supone un gran cambio en el modo de trabajar de la actualidad, por lo que habría que refinar nuevas modalidades contractuales para todos aquellos involucrados en la construcción.

Uno de los desarrollos lógicos hacia los que pueden tender los BIM es la aplicación de códigos normativos. C. Eastman (Eastman et al., 2011) se muestra muy escéptico en este tema, ya que el código computacional para comprobar suele ser muy rígido y no susceptible a cambios rápidos por parte del usuario. Por lo tanto, un desarrollo más natural que incorpore este hecho es la generación de un pseudo-lenguaje de programación altamente intuitivo y personalizable por parte del usuario para que pueda ser adaptado a cualquier código de construcción sin ninguna limitación. Otra posibilidad es que las herramientas de comprobación no se incorporen al programa sino que sean herramientas online actualizadas por los órganos gestores de esa regulación.

Internet no es la única tecnología que amplía las posibilidades de los BIM. La aplicación de algoritmos genéticos con diferentes funciones objetivo sobre los objetos paramétricos permitirá que estos alcancen objetivos de una manera más eficaz. Los entornos BIM también pueden

---

<sup>21</sup> La disponibilidad actual de las plataformas BIM promulgará una oleada de *plugins* durante los próximos cinco años.

tener ventajas de cara a la industria de la prefabricación, coordinando las ventajas de los BIM con las tecnologías de *Additive Manufacturing*<sup>22</sup>, como *Contour Crafting* o *Freeform Construction*, se puede lograr una gran personalización en los diseños sin elevar el coste.

Otras tecnologías que podrían llegar coordinarse con los entornos BIM con un gran éxito son las tecnologías de escaneo láser, realidad virtual e incluso la robótica aplicados a maquinaria de construcción, capaces de realizar acciones de manera automática en función de los parámetros del BM.

Por supuesto, donde existe un mayor potencial es en la unión de las dos grandes fuerzas del cambio de la industria AEC, los BIM y el *Lean Construction*, es decir en la idea de implementar los BIM con la finalidad de reducir costes y mejorar la industria en todos los aspectos (Sacks et al., 2010). Esta tendencia quizás sea debida a la novedad de ambos conceptos frente a lo estático de la industria AEC ante el cambio. Aún así, en el capítulo 9 de Eastman et al. (2011) podemos encontrar diversos ejemplos construidos en los cuales la adopción de los BIM (y de los principios de *Lean Construction*) ha supuesto una ventaja indudable.

En Sacks et al. (2010) encontramos un interesante estudio sobre las diferentes posibilidades de implementación de los principios de *Lean Construction* y BIM conjuntamente analizados en función de su relevancia. Según esta fuente, las funcionalidades BIM con una mayor interacción con el *Lean Construction* observadas están relacionadas con la representación, la visualización “multiusuario” y la comunicación online. Esto compone un total de cuatro funcionalidades desarrolladas de las dieciocho distintas para BIM que nos propone el artículo.

El mismo artículo nos expone una gran matriz con cincuenta y seis propuestas distintas sobre la interacción entre estos dos términos. Algunas de estas se encuentran en desarrollo y otras nacen de la interacción entre ambos conceptos. Aunque hacer una revisión de todas ellas supondría una tarea titánica y se alejaría de la finalidad de este trabajo, todas las propuestas son totalmente diferentes y de un gran interés.

### 1.5.2 Prácticas de diseño estructural en BIM

BIM es un entorno paramétrico, por lo que es muy importante estudiar cómo se introducen los parámetros estructurales en el BM para conocer los vínculos que existen entre los mismos y las relaciones entre ellos. El proceso de diseño estructural se ve afectado por la incorporación de los BIM, por lo que es importante conocer cómo funciona el diseño al trabajar con esta

---

<sup>22</sup> Additive Manufacturing: Proceso consistente en la creación de un objeto a partir de la adición de un material. Comúnmente conocido como Impresión 3D. (Buswell et al., 2007)



herramienta para aprovechar todas las ventajas. Entendemos por diseño estructural lo siguiente:

*Structural design is defined as sequence design actions related to structures, including their appearance, geometrical, mechanical properties, and any other related functionality factors. Architects, structural engineers and system engineers are all involved in such processes at construction design phase<sup>23</sup>(Chi et al., 2014)*

Si separamos el proceso de diseño estructural en los diferentes puntos de vista que están involucrados en él (arquitectónico, estructural y contractual) podemos dibujar el siguiente diagrama que muestra el proceso actual de diseño de estructuras.

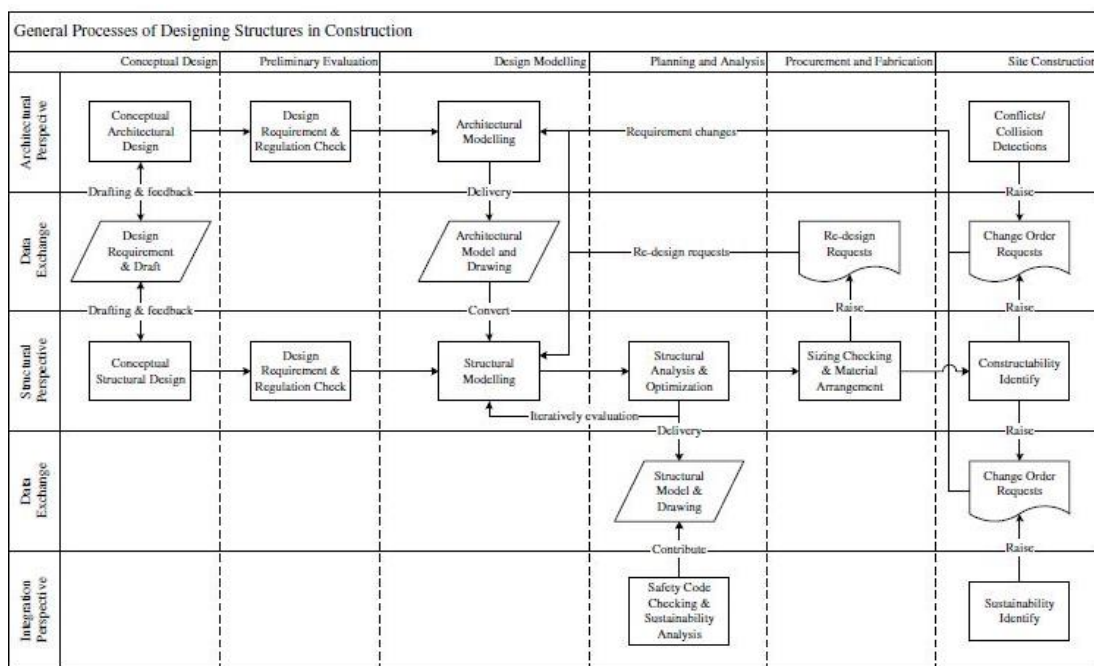


Fig. 7 Proceso del diseño estructural (Chi et al., 2014, P.3)

Como se observa, es un proceso muy fragmentado en fases, en el cual es necesario un constante flujo de información entre los distintos actores, que pueden ser o no la misma persona. El proceso empieza, desde el punto de vista arquitectónico, con un boceto conceptual de lo que será el edificio. Casi simultáneamente nace el primer diseño estructural que se valida, de un modo aproximado habitualmente, para estudiar la viabilidad del proyecto. A partir de este momento se crea un *feedback* constante entre los dos agentes hasta llegar a un modelo estructural, más o menos detallado, que cumpla las exigencias del proyecto y a la vez las

<sup>23</sup> El diseño estructural está definido por una secuencia de acciones de diseño relacionadas con la estructura, incluyendo su apariencia, geometría, propiedades mecánicas y cualquier otro factor funcional. Arquitectos, ingenieros estructurales e ingenieros de sistemas están involucrados en este proceso en las fases de diseño de la construcción.

exigencias normativas. Llegados a este punto, el intercambio pasa a ser más intenso entre el agente estructural y el contractual y se realizan modificaciones (generalmente menores) en el diseño para llegar a un diseño más óptimo, con criterios económicos y sostenibles, que cumpla con los requisitos contractuales. Si se realiza alguna modificación se informa al agente arquitectónico para que actualice su diseño y compruebe la existencia de errores. Por último, todos los agentes verifican por última vez que se cumplen simultáneamente todas las exigencias.

La introducción del software BIM, como es de imaginar, puede, y debe, alterar este proceso por varios motivos. Por un lado, mejora directamente la transmisión de datos entre los distintos agentes, ya que la mayoría de programas BIM permiten la actualización simultánea de un único archivo por varias personas a la vez. Por otro lado, el uso de factores paramétricos presenta varias ventajas intrínsecas, como la creación de un diseño más flexible a los cambios introducidos y la garantía de obtener respuestas eficientes a estos cambios. Además, mediante la manipulación de los parámetros geométricos por el usuario, se pueden explorar de un modo rápido diferentes soluciones y elegir la que mejor satisfaga todos los condicionantes. Esto aumenta la productividad al invertir menos tiempo en realizar y comprobar estas variantes.

Es muy interesante la posibilidad de implementar distintas comprobaciones en el proceso de diseño para agilizarlo y mejorar su resultado. Habitualmente, las únicas comprobaciones que se realizan en las etapas iniciales del proyecto son comprobaciones de índole estructural y, muchas veces con carácter aproximado, lo que deja otras posibles comprobaciones en un segundo plano, cuando son de vital importancia para el proyecto. Estas comprobaciones pueden ser de vías de escape de emergencia (Fenves, 1995), medidas de seguridad, superficies útiles o emisiones de CO<sub>2</sub> del proyecto (Schlueter and Thesseling, 2009). Cuando se realizan, habitualmente, es muy tarde para realizar grandes cambios en el proyecto, lo que lleva a la adopción de soluciones poco óptimas o no deseadas que empeoran el proyecto desde el punto de vista general. Gracias a las características de los BIM se puede alterar el proceso de diseño por completo, de manera que se incluyan desde el primer momento todas estas cuestiones en el diseño (Chi et al., 2014).

### 1.5.3 Integración del diseño estructural en BIM

La introducción de BIM al diseño de la estructura es capaz de reducir costes y de manejar simultáneamente un gran número de información por diferentes *roles* involucrados en este proceso. Además, gracias a la gran base de datos que son los BM se pueden crear comprobaciones de diversas índoles en estadios tempranos del diseño. Como consecuencia de esto, el 82% de los usuarios piensa que BIM tiene impactos positivos en la productividad de su

compañía (Azhar, 2011). Dicho esto, a nivel estructural esta mejora se puede producir desde uno o más puntos de vista diferentes, según la fase de diseño que se esté modificando.

### *1.5.3.1 Parametrización*

La idea principal existente tras los BIM es la captura de toda la información importante mediante parámetros accesibles al usuario de modo que puedan ser editados en cualquier etapa del proyecto. En la actualidad, las variables que afectan al modelo son principalmente geométricas (aunque existen algunas destinadas a la gestión de la obra) que están directamente relacionadas con su representación, pero es de esperar que en un futuro se incorporen a los modelos variables de seguridad, de estructura, durabilidad, medio ambientales, redundancia, etc. Justamente aquí es donde se encuentra el gran potencial de los BIM, en la utilización de estos parámetros para controlar todo lo relativo al proyecto.

Estos parámetros pueden ser utilizados de diversos modos durante el diseño de la estructura. Se podrían utilizar para posicionar los diferentes armados de la estructura y optimizar la longitud de las barras para que se desperdicie el menor material posible (Porwal, 2012), o podríamos utilizar estos parámetros de ubicación de las barras para definir su distribución en losas de manera automática (Cho et al., 2014), e incluso para evaluar ambos a la vez o combinarlos con otras posibilidades. Si utilizamos parámetros que midan el impacto ambiental (emisiones de CO<sub>2</sub>) se puede crear un algoritmo que genere una estructura óptima en base a estos criterios (Diao et al., 2011) y retroalimentar todos estos parámetros de manera cíclica entre para definir un nuevo método de diseño de estructuras en el cual la geometría de los elementos está directamente vinculada con sus materiales, las conexiones estructurales y las cargas (Hu et al., 2008).

Visto esto, las posibilidades son casi infinitas y se están realizando diferentes investigaciones para aprovechar la capacidad que tienen los BIM como grandes bases de datos interconectadas.

### *1.5.3.2 Visualización*

El potencial de los parámetros no se limita al almacenamiento de datos, estos también pueden influir en la manera en la que se visualiza el diseño estructural generado. Las herramientas BIM pueden ayudar a mejorar la visualización y a realizar cambios en tiempo real. La verdadera ventaja radica en la visualización de la información de una manera más intuitiva.

Integrar estas herramientas de visualización con tecnologías nuevas como guantes de control de gestos, (Mistry and Maes, 2009); realidad virtual, (Wang and Dunston, 2011) ; o realidad aumentada, (Wang et al., 2013); ya otorga a los diseñadores una gran capacidad de estudio de sus modelos estructurales y para descubrir los puntos clave o los errores.

Pero en la actualidad se han desarrollado diversas herramientas que implementan herramientas de análisis estructural mediante el método de Elementos Finitos (EF). Estas herramientas permiten vincular el análisis estructural al proyecto y mejorar la visualización a la vez que evitan errores de diseño. En la bibliografía se encuentra una tabla que muestra las diferentes aplicaciones (y objetivos) con las que se han utilizado los elementos finitos integrados en los BIM.

| Name                      | Organisation                       | Type                | Description  | Reference |
|---------------------------|------------------------------------|---------------------|--|-----------|
| Advance Design            | Graitec                            | Application         | The application is developed for the structural analysis of the reinforced concrete and steel structures. It includes a FEM engine and performs structural analysis according to Eurocodes, North American, and Canadian codes. It contains compatibility features that the data can be exchanged through IFC, CIS/2, SDNF, and PSS interfaces with Autodesk Revit Structure | [107]     |
| PKPM                      | China Academy of Building Research | Application Package | The application package is widely-used in China as required tools for architectural and structural design projects. The tools in this package, such as SATWE and FEQ, are capable to be used for high-rise building, shear walls structures analyses. These tools also provide Autodesk Revit plug-in for converting models  | [108]     |
| FEM-Design                | StruSoft                           | Application         | The application is developed based on Eurocodes, and is used for analysing on concrete, steel, and timber structures. It provides add-ins for Revit Structure, ArchiCAD, and Tekla Structure so that the analysis model geometry generated from these applications can be transferred into this application for further analyses   | [109]     |
| 2D/3D FEM                 | SOFiSTiK                           | Application Package | These applications provide finite element analyses that support various kinds of structures, including 2D, or 3D slab, shell, beam, shear walls, and so forth. They can be used for building, bridge, and industrial structure design. The data formats are all compatible with that of AutoCAD and Revit Structure  | [110]     |
| Robot Structural Analysis | Autodesk                           | Application         | The application is an Autodesk product, which combines with Revit Structure  | [111]     |

Fig. 8 Listado de software de EF integrado en BIM (Chi et al., 2014, P.10)

### 1.5.3.3 Intercambio de datos y comprobación del código

Entendiendo BIM como una base de datos de los valores de los parámetros se nos abre otro mundo de posibilidades y aplicaciones referidas tanto al intercambio de datos como a la comprobación del diseño estructural.

El intercambio de estos datos mediante Internet entre los distintos agentes involucrados en el proyecto (e incluso con otros usuarios ajenos al mismo) facilita el análisis, la evaluación y la discusión sobre los mismos. Estos datos se pueden compartir mediante aplicaciones tipo “cloud-BIM”<sup>24</sup> directamente relacionadas con Internet (Singh et al., 2011). Esta facilidad de difusión de los datos ayuda en la toma de decisiones, ya que existe un mayor número de personas que los observan y los valoran.

<sup>24</sup> Aplicaciones Cloud-BIM: Aplicaciones basadas en la computación de datos a través de Internet. Esos datos se obtienen de un modelo generado en un entorno BIM y pueden seguir o no vinculados al mismo.

Es esencial, dada la mayor aceptación que existe de los BIM, que se generen herramientas capaces de realizar análisis de los parámetros introducidos en el modelo. En los últimos años este punto ha sido uno de los más estudiados, aún así, de acuerdo con Chi et al. (2014), el 83% de los usuarios de BIM opinan que es un campo que precisa de una mayor investigación.

Existen herramientas cuya finalidad empieza por captar los datos para poder analizarlos y compartirlos, con la intención de hacer que el proyecto y su proceso de creación sean más transparentes. Estos parámetros, pese a estar compartidos, siguen vinculados al modelo a través de tecnologías Linked Data (König et al., 2013). La herramienta anterior pretende analizar el edificio con criterios de sostenibilidad sin aplicar ningún marco normativo concreto. Un ejemplo de comprobación de código lo encontramos en la herramienta desarrollada por Wu y Chang (2013) en la cual se implementa el análisis de *Ratio of Equivalent Transparency* de la normativa taiwanesa sobre sostenibilidad. La comprobación de código proporciona a los proyectistas herramientas para que puedan proyectar edificios no solo con requisitos de calidad de espacios o buen diseño estructural, sino con criterios de seguridad en la construcción, sostenibilidad o funcionalidad. Las aplicaciones de comprobación de código suelen funcionar siguiendo un mismo esquema:

1- Interpretación de la legislación y estructuración de manera lógica de las normativas para su aplicación.

2- Preparación del *Building Model* mediante la introducción de los parámetros necesarios.

3- Aplicación de la legislación mediante la comprobación

4- Generación de un informe de resultados. (Eastman et al., 2009)

Las herramientas de comprobación son realmente útiles y añaden una gran cantidad de funcionalidades al modelo. En todo proyecto se realizarán estas comprobaciones tarde o temprano. La utilización de estas herramientas adelantará, cronológicamente hablando, la comprobación en la fase de diseño, lo que significa aumentar la importancia de estos condicionantes en la obra. Actualmente, los programas BIM por defecto no incluyen ninguna herramienta de este tipo, pero empiezan a incluir los parámetros necesarios para realizarlas. Autodesk Revit realiza el modelo analítico de la estructura según esta va siendo incorporada al modelo, pero en cambio no permite realizar ningún tipo de comprobación resistente de la misma, si bien permite su importación a otros *softwares* de análisis estructural como Cype o SAP2000.

La clasificación es general y habitualmente cualquier aplicación cuyo objetivo esté relacionado con el diseño estructural en los BIM no está cerrada a uno de estos puntos, sino que su objetivo principal está más cercano a uno de ellos, aunque tocará en mayor o menor medida los otros. Por ejemplo, la aplicación que pretendemos desarrollar estaría clasificada bajo la etiqueta de “Aplicación de código” pero tendría mucho en común con las “Aplicaciones de parametrización”, ya que se pretende evaluar diferentes parámetros para obtener el resultado óptimo.

#### 1.5.4 Optimización estructural en BIM

Una vez realizado todo el estudio anterior podemos entender la complejidad de la aplicación de optimización estructural en un entorno BIM, ya que se está desarrollando una herramienta compleja que tiene que actuar de una manera muy definida para poder obtener buenos resultados.

Hoy en día, encontramos pocos ejemplos de este tipo de aplicaciones. Uno de ellos es la aplicación desarrollada por los japoneses Diao Y. y Kato S. (2011) destinada a generar una herramienta de ayuda que predice el comportamiento del edificio frente a criterios de sostenibilidad a partir de un sistema de optimización basado en una frontera de Pareto. La aplicación en este caso extrae los datos según va evolucionando el modelo y se van añadiendo parámetros al mismo, y muestra al usuario el resultado del análisis. Justamente por este motivo, la herramienta solo está destinada como apoyo y no como herramienta de diseño, ya que el número de datos es creciente y las simulaciones realizadas en las etapas iniciales carecen de gran exactitud. A pesar de esta limitación, se consigue una reducción de hasta el 43% del coste material inicial y una reducción del 8,7% de la carga anual energética de climatización (la primera iteración del algoritmo dio como resultado una reducción del 1,9% y 0,2% respectivamente).

Otro ejemplo podemos encontrarlo en el *plugin* creado por Porwal (2012), cuya finalidad era reducir al mínimo la longitud de barra que se malgasta debido a los despuntes. Para ello, realiza un primer análisis estructural con el dimensionado de las barras y un posterior análisis de las pérdidas, en el que calcula la longitud de cada una de las barras, incluyendo longitudes de anclaje, y los decalajes cuando sean aplicables. El análisis de pérdidas funciona a través de una función unidimensional que agrega las distintas longitudes de barra en distintas combinaciones para reducir los despuntes. En este caso el algoritmo de optimización utilizado es un SA y consigue reducir la pérdida de despunte hasta en un 1,6% de la longitud total de refuerzo del elemento en las mejores condiciones.

En los ejemplos anteriores hemos podido encontrar ciertos puntos en común que nos ayudarán a decidir cómo plantear el problema de optimización estructural. Recordemos que nuestra

aplicación tiene como objetivo una optimización multicriterio cuyos objetos serán evaluados individualmente.

Ya que BIM funciona como una gran base de datos, es lógico incorporar aquellos datos relativos a nuestro algoritmo dentro del propio modelo y que estos estén vinculados al mismo con capacidad de modificarlo según los resultados. Los parámetros se introducirán durante el dibujo del *Building Model* sin necesidad de una posterior modificación del mismo. Más adelante estudiaremos los diferentes modos que tenemos de introducir parámetros a un objeto en Revit. Estos parámetros serán extraídos y clasificados en tres grupos:

-Parámetros de definición del elemento estructural: Aquellos parámetros que definen geométrica y mecánicamente el elemento estructural analizado.

-Parámetros de restricción: Conjunto de parámetros externos al elemento estructural que componen las limitaciones a las que este está sometido.

-Parámetros de algoritmo: Grupo de parámetros cuya misión es controlar el algoritmo de optimización.

En este punto, cada tipo de parámetro asumirá su función y se iniciará el algoritmo genético. Este algoritmo de manera autónoma generará nuevos parámetros de definición cuando los necesite para incrementar la población de objetos evaluados. Todos los objetos deberán cumplir las restricciones de la normativa para pasar al siguiente paso.

Tras la ejecución del algoritmo y la evaluación del coste final del óptimo encontrado se procederá a un volcado de los datos de vuelta al *Building Model*. Lógicamente en este volcado cada parámetro ocupará su correspondiente lugar. Tras el volcado del resultado el dibujo se regenerará y se actualizará automáticamente a los nuevos parámetros. A continuación, una ventana informará al usuario de la finalización del algoritmo y del resultado obtenido. En caso de no estar conforme con el resultado, el usuario podrá repetir el proceso desde el inicio. El proceso queda resumido en el diagrama de flujo adjunto.

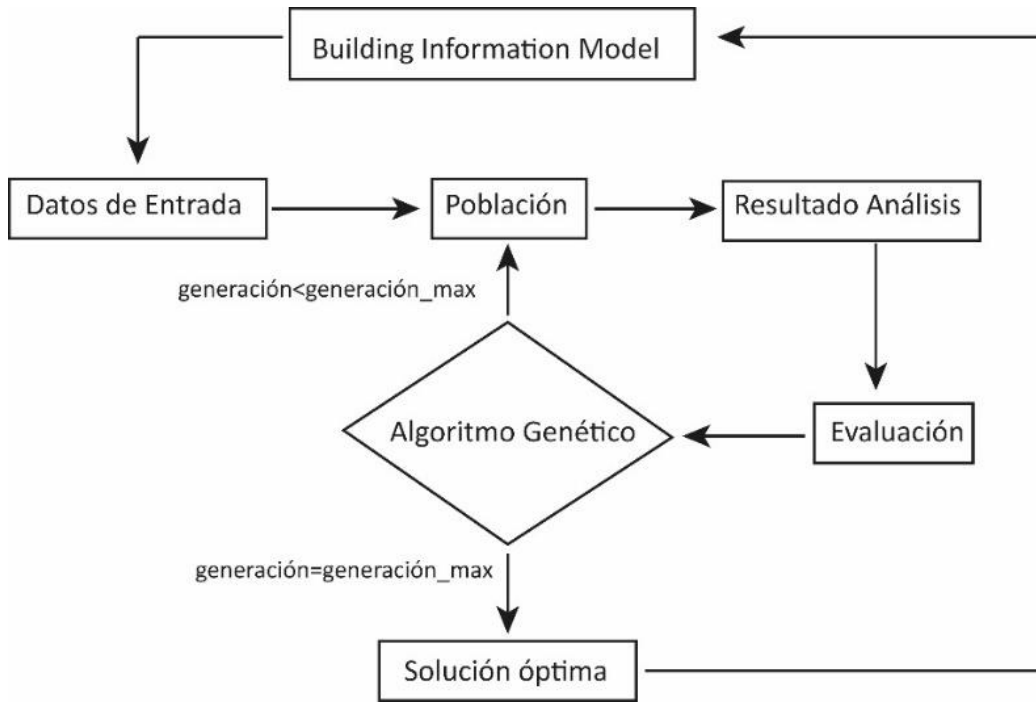


Fig. 9 Gráfico de flujo del diseño óptimo



## 2. DESARROLLO

---

Hasta el momento se han sentado las bases teóricas sobre la aplicación que se busca desarrollar. Además se ha estudiado el entorno de trabajo donde esta va a operar y cómo este funciona y se comporta, así como las técnicas utilizadas por otros investigadores para solucionar problemas semejantes. También se ha analizado el valor que una aplicación como esta podría tener en el desarrollo de los BIM y su futuro.

Se ha estudiado el marco científico donde el *plugin* de optimización que pretendemos desarrollar se va a situar para esclarecer las cualidades que debe poseer y cómo debe comportarse para un correcto funcionamiento. Una vez establecidas estas bases podemos empezar con el desarrollo de la aplicación. Era necesario un estudio como anterior para entender el punto de partida y sus condicionantes, así como para vislumbrar el camino que tenemos que seguir y las diferentes funcionalidades de los BIM y de los algoritmos de optimización que nos pueden resultar útiles.

La creación de la aplicación se explicará detallada y ordenadamente con el fin de clarificar todos y cada uno de los componentes. Primero se tratará la conexión con Revit, el programa BIM elegido, tanto para conocer qué conectar, como para conocer cómo hacerlo. Tras esclarecer este punto, empezaremos a establecer estas conexiones y a crear el algoritmo genético. Este algoritmo será desarrollado en dos fases, para funcionar como algoritmo monocriterio y para hacerlo como algoritmo multicriterio. Aunque existe un gran cambio de perspectiva entre el modo de funcionamiento de ambos tipos de algoritmo, se prevé que este proceso será de utilidad, ya que el algoritmo inicial se utilizará para estudiar mediante herramientas estadísticas las relaciones existentes entre los criterios, que facilitarán la tarea de creación del algoritmo multicriterio en el caso de que este mejore la solución de una manera significativa.

El código se presenta fragmentado y acompañado de comentarios explicativos que clarifiquen su funcionalidad y justifiquen las decisiones tomadas durante la fase de programación. Aunque se intenta mantener la mayor linealidad posible en la fragmentación del código para que el lector tenga una visión global en todo momento. Para garantizar la plena comprensión de la aplicación y su funcionamiento se adjunta una copia completa del código extraída del compilador en el Anexo 1 y en el Anexo 2.

## 2.1 PARÁMETROS Y SU CONEXIÓN CON LA API

Como hemos visto en la sección anterior, los modelos BIM se caracterizan, entre otras cosas, por la existencia de parámetros que restringen y condicionan el modelo. En el caso de la aplicación a desarrollar vamos a utilizar estos parámetros para definir todas las características de la viga y para configurar adecuadamente el algoritmo genético a través de la interfaz de Autodesk Revit.

Definiremos a continuación los parámetros necesarios para nuestro modelo, más adelante quedará justificado su uso y necesidad dentro de la aplicación. Estos deben ser extraídos del BM y asignados de manera correcta a las variables de nuestra aplicación para poder iniciar el algoritmo. Además, tras la evaluación del algoritmo, los parámetros que se hayan modificado deben ser volcados al modelo de modo que actualicen el mismo.

### 2.1.1 Glosario de parámetros y definición de los mismos

Para realizar las tareas descritas anteriormente vamos a necesitar diferentes tipos de parámetros que podemos clasificar en tres grupos. El primer grupo está formado por los denominados parámetros de definición estructural, grupo que incluye los parámetros que definen las características geométricas y mecánicas de la viga. Estos serán los que definan el modelo estructural del elemento a analizar. El segundo grupo está formado por los parámetros de configuración del algoritmo genético, su misión es la de permitir que el usuario sea capaz de configurar el algoritmo genético desde la interfaz del BIM de una manera ágil, cómoda y sin necesitar la modificación del código de programación. Por último el grupo de los parámetros de salida, cuya misión es el volcado de los resultados de coste, emisión de CO<sub>2</sub>, consumo energético y número de barras en el modelo final para que el usuario los conozca y pueda evaluarlos junto a los del resto del edificio para mejorar su sostenibilidad.

#### 2.1.1.1 *Parámetros de definición del elemento estructural*

Para definir el elemento estructural precisaremos de parámetros que definan su geometría, parámetros que definan sus materiales mecánicamente y parámetros que definan su estado de cargas. Lo que supone un total de catorce parámetros destinados a esta finalidad: diez para la definición de la geometría, dos para las características mecánicas y otros dos para su estado de cargas.

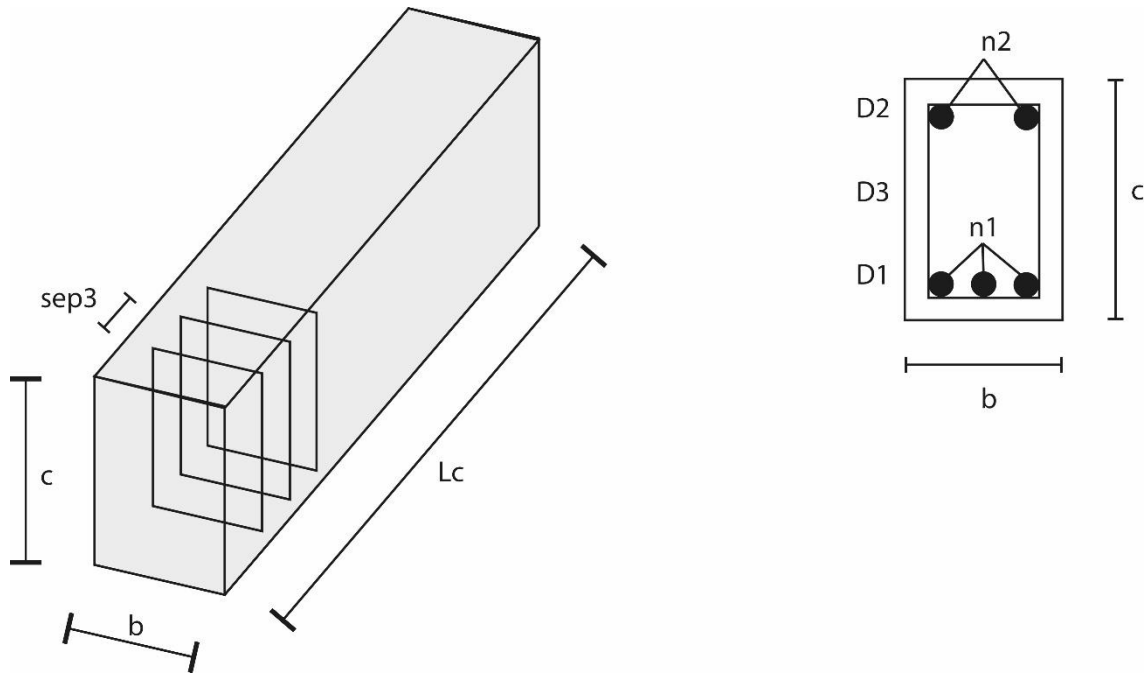


Fig. 10 Parámetros geométricos de la viga

En la Figura 10 se ubican los parámetros de definición geométrica de la viga. Las dimensiones de la sección rectangular de la viga quedan definidas por dos parámetros, el ancho de sección ( $b$ ) correspondiente con la distancia transversal y el canto ( $h$ ) correspondiente con la altura total, medidas ambas en milímetros. La luz de cálculo ( $L_c$ ) es la distancia existente entre los centros de gravedad de los apoyos. Es importante tener en cuenta cómo tiene considera el *software* BIM esta longitud y que no se haya medido desde la cara de los soportes. El recubrimiento geométrico ( $rec$ ) de todo el armado es la distancia entre la cara del elemento estructural hasta la cara más exterior de cualquiera de las armaduras (incluyendo estribos), se recomienda utilizar los valores dados por la EHE en el artículo 37.2.4, aunque siempre se puede elegir utilizar un recubrimiento mayor si así lo desea el usuario. Los diámetros de las armaduras se definen indicando el diámetro de las armaduras, principal ( $D1$ ), secundaria ( $D2$ ) y transversal ( $D3$ ); el número de barras de cada uno de los armados, principal ( $n1$ ) y secundario ( $n2$ ) y el número de barras transversales, expresado mediante la separación entre estribos ( $sep3$ ) que es constante a lo largo de toda la barra. El armado se ha considerado para funcionar con una viga isostática, por ello toda referencia a la armadura principal está dirigida a la armadura inferior de la sección y toda referencia al armado secundario está referida al armado superior, además el armado se mantiene constante a lo largo de todo el elemento estructural, ya que no se pretende calcular luces elevadas donde sí que sería recomendable no prolongar las mismas barras toda la longitud.

Con estos parámetros definiríamos la geometría del elemento estructural, pero aún es necesario determinar los parámetros relativos a los materiales y al estado de cargas. La resistencia característica del hormigón ( $f_{ck}$ ) y el límite elástico del acero ( $f_{yd}$ ) se expresan en  $N/mm^2$  (o el equivalente  $KN/m^2$ ). El segundo no se introduce dentro del algoritmo genético como variable y se fija en un valor de  $500 N/mm^2$  correspondiente con el acero B-500S y B-500SD, los tipos de acero de construcción empleados en la actualidad en España, no obstante, el usuario puede editar este valor sin ningún inconveniente si lo desea, aunque esto puede dar lugar a errores en la estimación final del coste (únicamente en el valor numérico no en el diseño optimizado).

La introducción de cargas se realiza diferenciando las cargas gravitatorias (G), el sumatorio total de los valores nominales de las cargas muertas<sup>25</sup> actuantes sobre el elemento por metro lineal sin incluir el peso propio, y de las cargas variables (Q), valor por unidad de longitud de la carga variable principal aplicada sobre el elemento, ambos expresados en  $N/mm$  o en  $KN/m$ . Con todos estos parámetros tendríamos correctamente definido el sistema estructural actuante sobre la viga.

#### 2.1.1.2 Parámetros de configuración del algoritmo genético

Es previsible que el algoritmo genético no funcione con la misma efectividad en barras de distinta longitud sometidas a distintas cargas. Por ello es preciso que el usuario pueda configurar los parámetros del algoritmo para que estos puedan ajustarse a su caso particular. Así se puede garantizar que el algoritmo converja en un óptimo para cada problema distinto si estos parámetros están ajustados.

Entre los distintos parámetros encontramos el tamaño de la población ( $max\_especimenes$ ) existente en cada una de las diferentes generaciones, uno de los parámetros que más va a influir en el resultado del algoritmo. Relacionado con este está el número de descendientes generados ( $max\_descendiente$ ), el cual determina el número de nuevos especímenes creados en cada generación a partir del cruce, y el número de generaciones, el número de iteraciones completas que atraviesa el algoritmo antes de definir el resultado óptimo. En cada una de estas generaciones se realizan los cruces siguiendo la misma lógica para generar los descendientes, estos descendientes tienen la posibilidad de sufrir mutaciones ( $\%mut$ ) a cada uno de sus genes que les haga diferir del valor original de sus progenitores o pueden provenir de una población exterior ( $\%prodigo$ ) en lugar de ser generados a través de la población actual. Como el objetivo es crear un algoritmo multicriterio se va a forzar que los padres tiendan a ser elegidos de

---

<sup>25</sup> Carga Muerta: Toda carga que cumpla la definición de carga permanente del CTE DB-SE y no sea el peso propio del elemento estructural analizado. Por ejemplo la carga de tabiquería.

miembros que formen parte de la frontera de Pareto (fitness), de este modo el número de especímenes de esta frontera aumenta y el óptimo obtenido al finalizar el proceso será un mejor resultado (más adelante se describirá con más detalle cómo funciona este parámetro).

### 2.1.1.3 Parámetros de salida

Para que el usuario pueda conocer los diferentes costes que se han evaluado en la optimización, se han creado una serie de nuevos parámetros que muestran el resultado final obtenido. Estos parámetros no influyen en el algoritmo, aunque el usuario los edite inicialmente ya que se sobrescriben al final del mismo con los valores obtenidos y se corresponden con cada uno de los cuatro criterios de optimización empleados.

El primero de ellos es el precio total del elemento analizado una vez terminado en obra, correspondiente con el coste económico del mismo. Se ha considerado tanto el precio del material como la mano de obra y el transporte necesarios para realizar la correcta construcción del elemento in situ. Estos datos de precios se han extraído de la base de datos del Institut de Tecnologia de la Construcció de Catalunya (ITeC) y están expresados en Euros.

El segundo es la masa total del gas CO<sub>2</sub> emitida durante la construcción del elemento medida en kilogramos. Para esta evaluación se consideran las emisiones producidas por la fabricación de los materiales desde la extracción de las materias primas, las emisiones producidas por la energía consumida por la maquinaria de obra utilizada para la construcción. No se consideran las emisiones de CO<sub>2</sub> derivadas de la futura demolición del elemento o del procesamiento de los residuos, así como tampoco se deducen las cantidades de CO<sub>2</sub> captadas por el hormigón debido a la carbonatación durante su vida útil ni las emisiones originadas por el transporte de los diferentes materiales a la obra. Estos tres tipos de emisiones se han descartado debido a la gran incertidumbre que existe en su cálculo (sin estar pensado para una obra en concreto) y a que pueden ser contabilizadas con mayor facilidad utilizando otras herramientas propias de los entornos BIM.

El tercero es el consumo de energía utilizada para la producción del elemento estructural en Kwh. En el conteo del consumo energético se considera la energía consumida por la maquinaria durante todo el proceso de fabricación de los materiales, transporte y puesta en obra. Como en el caso del CO<sub>2</sub> no se ha contabilizado la energía necesaria para el derrumbe, el transporte o la gestión de los recursos generados en la obra debido al número de incógnitas que estas preguntas introducen en el diseño.

El cuarto y último criterio es el número de barras de acero empleadas en el elemento. Este criterio se consigue sumando las diferentes barras que componen la viga, independientemente

de su diámetro o de su función (resistir flexión positiva, negativa o cortante). Lo que se pretende con este criterio es controlar la cantidad de acero utilizada y favorecer la elección de diámetros de gran tamaño. De este modo se consigue de manera indirecta facilitar la puesta en obra del elemento, ya que es necesario ubicar y replantear un menor número de elementos durante la fase de construcción.

Los valores para cuantificar los criterios de optimización, a excepción del número de barras, han sido calculados a partir de la suma de los valores de cada uno de los componentes del material. En el caso del hormigón se ha utilizado una dosificación estándar según la resistencia característica del mismo. Todos los datos necesarios para estas cuantificaciones se han obtenido de la base de datos del ITeC mencionada con anterioridad. En la tabla siguiente se recogen los valores finales introducidos en el algoritmo para los diferentes materiales.

|  | <b>Precio<br/>(€)</b> | <b>Emisión de<br/>CO<sub>2</sub> (Kg)</b> | <b>Consumo de<br/>Energía (Kwh)</b> |
|--|-----------------------|---|-------------------------------------|
| <b>Hormigón (m<sup>3</sup>)</b>                |                       |   |                                     |
| HA-25  | 97,68                 | 335,63                                    | 414,59                              |
| HA-30  | 101,89                | 366,53                                    | 440,74                              |
| HA-35  | 103,51                | 394,6                                     | 458,02                              |
| HA-40  | 106,21                | 417,76                                    | 477,5                               |
| HA-45  | 109,46                | 448,49                                    | 505,73                              |
| HA-50  | 114,45                | 501,01                                    | 551,07                              |
| HA-55  | 116,36                | 561,56                                    | 622,30                              |
| HA-60  | 118,75                | 605,36                                    | 658,58                              |
| HA-70  | 121,86                | 618,97                                    | 670,01                              |
| HA-80  | 128,02                | 630,41                                    | 667,12                              |
| HA-90  | 130,82                | 653,93                                    | 688,14                              |
| HA-100   | 130,85                | 686,87                                    | 723,72                              |
| <b>Acero (Kg)</b>                              | 1,29                  | 3,03                                      | 10,44                               |
| <b>Encofrado de<br/>madera (m<sup>2</sup>)</b> | 31,41                 | 2,97                                      | 10,97                               |

*Tabla 1 Valores de los materiales extraídos de la base de datos del ITeC*

Con esto habríamos terminado la revisión de cada uno de los parámetros necesarios para realizar las diferentes operaciones dentro de nuestro algoritmo de optimización. Estos parámetros cubren todos los requisitos y algunos de ellos han surgido en el transcurso de la investigación y del desarrollo del código según este avanzaba en su programación. Al analizar el código se observará claramente cómo actúa cada uno de los mismos en la obtención de la viga óptima. Optimización en Autodesk Revit

Hasta el momento hemos afirmado que los entornos BIM funcionan a partir de parámetros vinculados entre sí. Aunque esto es cierto, se requiere la explicación de cómo se organizan estos, pues en cada entorno BIM estos funcionan de un modo diferente y se relacionan de una manera única entre ellos y con la API. En Autodesk Revit, el software BIM escogido para ejemplificar la aplicación, los parámetros están almacenados dentro de cada uno de los objetos que el usuario introduce al modelo para crear su BM. Los diferentes objetos del modelo pertenecen a lo que Revit denomina familias. Son estas las que almacenan los datos de los parámetros de cada objeto. El *Building Model* es una suma de diferentes familias y los vínculos existentes entre ellas, así como los valores de los parámetros de cada objeto.

*“A family is a group of elements with a common set of properties, called parameters, and a related graphical representation. Different elements belonging to a family may have different values for some or all of their parameters, but the set of parameters (their names and meanings) is the same. These variations within the family are called family types or types.”*<sup>26</sup>(“Autodesk Revit Help,” 2014)

Como se observa, el funcionamiento de Revit se basa en familias y está vinculado a estas. Existen familias de todo tipo incorporadas al programa, como las “familias estructurales”, y estas pueden ser creadas o modificadas por los usuarios ajustándose así a sus necesidades. Por lo tanto, para la introducción y uso de nuestros parámetros en Revit trabajaremos usando las familias. Nuestro *plugin* funcionará a partir de una familia generada por nosotros que incorporará los parámetros definidos en el apartado anterior. El nombre de esta familia será ConcreteBeamOpt. Las familias pueden contener subfamilias, y en estas subfamilias se pueden introducir diferentes valores por defecto de alguno de los parámetros de la familia. En principio la familia ConcreteBeamOpt no tendrá ninguna subfamilia incluida, aunque se estudiará si sería conveniente crearla.

### 2.1.2 Introducción de los parámetros en Revit

Durante la creación de la familia hay que decidir cómo introducir los parámetros en la misma. Las familias pueden tener tres tipos de parámetros diferentes dependiendo de si afectan a un proyecto, “*Project Parameters*”, a una familia, “*Family Parameters*”, o si se pueden asignar libremente, “*Shared Parameters*”, cada uno de ellos tiene su propio modo de introducción, ventajas y desventajas.

---

<sup>26</sup> Una familia es un grupo de elementos con un conjunto común de propiedades, llamadas parámetros, i una representación gráfica relacionada. Diferentes elementos pertenecientes a una familia pueden tener parámetros con valores diferentes, pero el conjunto (en cuanto a nombres y significado) es el mismo. Estas variaciones se denominan “tipos de familias” o “tipos”.

Los parámetros más habitualmente utilizados para la gestión de la obra son los denominados “*Project Parameters*” o Parámetros de Proyecto. Estos son específicos para un único proyecto y no pueden ser compartidos entre varios archivos o proyectos. Se asignan a elementos a través de categorías o vistas. Dada esta asignación, se utilizan para labores de gestión de la obra (como la planificación de plazos), ordenación de los elementos dentro del archivo o para filtrar algunos de los elementos en ciertas vistas.

Los parámetros más utilizados son los “*Family Parameters*” o Parámetros de Familia. Estos habitualmente definen valores de las variables como la geometría o los materiales. Son específicos para cada familia y pueden ser usados como parámetros de control en una familia anidada dentro de otra asociando el valor de uno de estos parámetros a un parámetro de la familia anidada.

Por último, encontramos los “*Shared Parameters*”, este último tipo no son parámetros en sí mismos, sino que son definiciones de los mismos que están preparadas para ser utilizadas en diferentes familias o proyectos directamente. Una vez generados, estos pueden ser asignados para ser utilizados como parámetros de proyecto o parámetros de familia, que al estar pensados para ser utilizados en diferentes ocasiones, no están almacenados en el archivo. Revit almacena los datos en un archivo de texto externo al cual llama para recuperar cómo funciona dicho parámetro. Esta vinculación con el archivo de texto hace que este tipo de parámetro no pueda ser editado desde el modelo, únicamente se puede modificar su valor manteniendo el nombre, las unidades y la categoría.

Existe un cuarto tipo de parámetro, los “*Global Parameters*” o Parámetros Globales. Estos pueden no pertenecer a una categoría, pero están vinculados a un único fichero. No entraremos en detalle en este tipo de parámetros ya que solo son accesibles para los servicios de mantenimiento de Autodesk.

En otra categoría encontramos los *BuiltInParameters*. Estos son los parámetros existentes por defecto en Autodesk Revit y que llevan incorporados todos los elementos existentes en el archivo. Este tipo de parámetros se encuentran asignados por defecto al objeto y no se pueden eliminar o editar de ningún modo por el usuario. Tienen un nombre específico y se ocupan de una única función. Cada familia puede tener, o no, uno o más *BuiltInParameters*. Muchas veces, estos parámetros están ocultos al usuario y solo pueden visualizarse a través de la herramienta Revit Lookup incluida en el Revit SDK (*Revit Software Development Kit*). Esta herramienta nos muestra, además todas las propiedades (nombre, valor, tipo de variable...) de cada uno de los



parámetros existentes en el objeto. Una lista completa de todos los *BuiltInParameters* y una breve descripción (a veces poco precisa) se puede encontrar en el Revit SDK.

En nuestro caso nos decantamos por la utilización de los *Shared Parameters* utilizando los *BuiltInParameters* en los casos que esto sea ventajoso. Nos hemos decantado por esta opción por tres motivos. Por un lado la mayoría de parámetros serían comunes a otros tipos de estructura si se amplía el campo de actuación de la aplicación, por lo tanto, la existencia de los mismos en un fichero externo que permita asignarlos a componentes de diferente índole nos facilita una posible labor de ampliación de funcionalidades. Por otro lado, al crear los parámetros desde el inicio y tener acceso externo a los mismos tenemos una mayor facilidad de acceso a los datos que nos permitirán identificar estos parámetros desde la API. Además, los *Shared Parameters* son los parámetros que mayor personalización tienen, permitiendo controlar desde la categoría del valor que están midiendo, hasta las unidades en las que está medido, incluyendo el tipo de variable utilizado.

Es cierto que en la API existen funciones capaces de insertar nuevos parámetros contenidos en un fichero de *Shared Parameters* a una familia cualquiera ya existente sin necesidad de generar una familia nueva y obligar al usuario a utilizarla. No obstante esta opción se ha descartado, ya que significaría o bien que el usuario no pudiera editar los valores de los parámetros, dado que estos se asignarían durante la ejecución del código, o bien que existieran dos aplicaciones diferentes, una para la asignación de parámetros y otra para la optimización. Cualquiera de estos dos escenarios no es deseable para el desarrollo de la aplicación, por ello se ha optado por crear la familia *ConcreteBeamOpt* anteriormente explicada.

### 2.1.3 Captación y volcado de los valores de los parámetros

No sólo es importante estudiar los tipos de parámetros que vamos a utilizar, sino también cómo vamos a captar los valores de estos parámetros para introducirlos en nuestro código. Para ello, lo primero que tenemos que decidir es el tipo de variable<sup>27</sup> que tenemos que asignar a cada uno de los parámetros. Se ha decidido que cada parámetro tenga el siguiente tipo de variable asignado:

---

<sup>27</sup> El lenguaje de programación C# tienen diferentes tipos de variable. Según este tipo la variable almacenada puede ser un número entero, decimal, un carácter o una cadena de caracteres. Una lista de los tipos de variable básicos puede consultarse en: <https://msdn.microsoft.com/en-us/library/>

| Nombre del parámetro          | Tipo de variable |
|-------------------------------|------------------|
| <b>b</b>                      | double           |
| <b>h</b>                      | double           |
| <b>Lc</b>                     | double           |
| <b>rec</b>                    | double           |
| <b>D1</b>                     | double           |
| <b>n1</b>                     | double           |
| <b>D2</b>                     | double           |
| <b>n2</b>                     | double           |
| <b>D3</b>                     | double           |
| <b>sep3</b>                   | double           |
| <b>f<sub>ck</sub></b>         | double           |
| <b>f<sub>yd</sub></b>         | double           |
| <b>G</b>                      | double           |
| <b>Q</b>                      | double           |
| <b>max_especimenes</b>        | int              |
| <b>max_descendiente</b>       | int              |
| <b>generación</b>             | int              |
| <b>%mut</b>                   | int              |
| <b>%prodigo</b>               | int              |
| <b>fitness</b>                | int              |
| <b>Coste</b>                  | double           |
| <b>Emisión CO<sub>2</sub></b> | double           |
| <b>Consumo Energético</b>     | double           |

Tabla 2 Tipo de variable asignado a cada parámetro

Estudiando las diferentes funciones existentes en la API de Revit llegamos a la conclusión de que para obtener el valor de los parámetros de un elemento tenemos que utilizar la función que se muestra en la Figura 11. Esta función tiene cuatro sobrecargas diferentes que permiten utilizar cuatro identificadores distintos: `GUID`, `string`, `BuiltInParameter` y `Definition`. Hay que decidir cuál de los cuatro identificadores nos conviene utilizar para captar los valores sin que aparezcan errores ni cruce de datos.

```
Parameter.get_Parameter(identificador)
```

Fig. 11 Función de captación de los valores de un parámetro en la API de Revit

El identificador `GUID` representa el *Global Unique Identifier*, un identificador tipo `string` único que queda asignado a un parámetro y que permite diferenciarlo. Es un componente del .NET Framework donde está englobado el lenguaje C#, que no es exclusivo de Autodesk Revit, pero este los utiliza para nombrar a sus parámetros de una manera única. La sobrecarga con entrada mediante `string` nos pide el nombre exacto del parámetro a localizar. En este caso se puede utilizar el nombre del parámetro que se muestra en el fichero de los *Shared Parameters*. Puede ocasionar problemas si la misma familia tuviese dos o más parámetros con el mismo nombre. La

siguiente sobrecarga únicamente funciona para los `BuiltInParameters` ya que localiza un parámetro a partir del identificador asociado al nombre de este. La última opción que tenemos, funciona a partir del nombre del parámetro distinguiendo si este está definido de manera interna, si existe en el entorno de Revit, o de manera externa, definido por el usuario a partir de un archivo de *Shared Parameter*.

Se ha optado por la utilización del `GUID` para los parámetros definidos a través del fichero externo ya que se prevé que es la opción que produce un menor conflicto al gestionar identificadores únicos para cada parámetro. A ellos se accede manualmente a través del archivo de texto que contiene la información sobre los parámetros. Para los `BuiltInParameters` que vayamos a usar utilizaremos el identificador basado en su nombre, tercera sobrecarga, que obtendremos mediante la Revit Lookup Tool.

Además de obtener los valores y algunos de los parámetros, es necesario volcar el valor de otros obtenidos tras la ejecución completa del código, esto funcionará de una manera similar, aunque con alguna diferencia. Para variar el valor de un parámetro necesitaremos usar la función que se muestra en la Figura 12, la función `LookupParameter`. Esta función únicamente admite identificadores de tipo `string` basados en el nombre exacto del parámetro. Por este motivo primero necesitaremos extraer el nombre de cada uno de ellos y tras esto, volcar el valor de salida del algoritmo. Realizar la operación de este modo nos evita conflictos de nombre solo con realizar un paso intermedio.

```
FamilyInstance.Symbol.LookupParameter(identificador).Set(valor)
```

Fig. 12 Función de volcado de datos en la API de Revit

Tras conocer cómo se va a interactuar con la familia de Revit se crea la misma bajo el nombre de `ConcreteBeamOpt`, a ella se le han asignado todos los parámetros tal y como se ha indicado en los apartados anteriores. Los tres parámetros que controlan las dimensiones geométricas (`b`, `h` y `Lc`) se han asignado a las dimensiones correspondientes de modo que cualquier variación en los valores de los mismos producirá una variación en las dimensiones del objeto dentro del BM. El punto de inserción de la misma se encuentra en el eje de la barra centrado en el centro de gravedad de la sección. En las imágenes siguientes se puede observar la morfología de la familia (Figura 13) y los parámetros que en ella se han asignado a través del fichero de *Shared Parameters* (Figura 14). Los parámetros se consideran parámetros de familia, no de ejemplar, por lo tanto si se quieren optimizar dos o más elementos con diferentes dimensiones es necesario crear una subfamilia para cada uno de ellos.

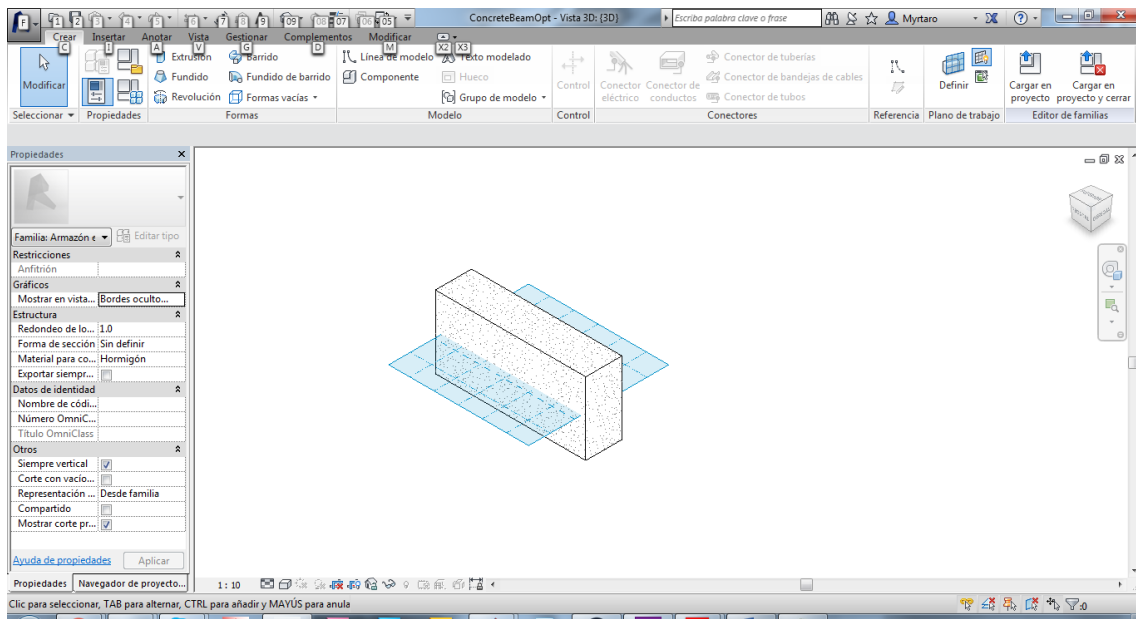


Fig. 13 Familia ConcreteBeamOpt

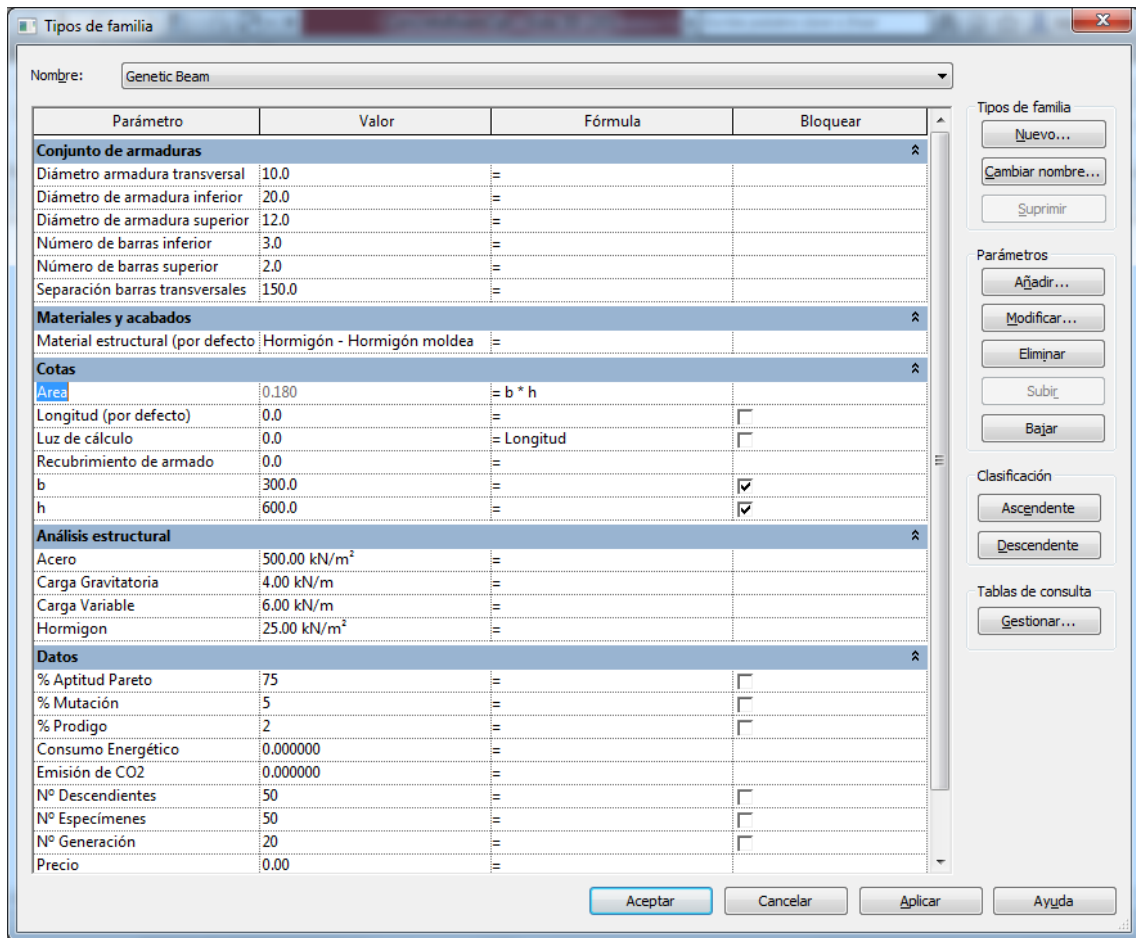


Fig. 14 Parámetros de ConcreteBeamOpt

## 2.2 VISIÓN GENERAL DE LA APLICACIÓN DE OPTIMIZACIÓN MONOCRITERIO

Al tomar todas las decisiones relativas a los parámetros hemos terminado de asentar todo lo necesario para empezar a programar la aplicación en Revit. Nuestro objetivo es realizar una optimización multiobjetivo teniendo en cuenta el coste, la emisión de CO<sub>2</sub> y el consumo energético. Para conseguir esto es necesario conocer cómo interactúan estos factores entre sí. Por ello vamos a empezar realizando una primera aplicación ejemplificativa que optimice con cualquiera de estos tres criterios pero nos muestre el resultado de los otros dos.

Es importante tener claros los objetos que se van a definir en el lenguaje C# para mantener un orden en el código que ayude a su legibilidad. Los objetos son una agrupación de funciones con finalidades similares para poder ser llamadas desde la función `Execute`, es decir, el cuerpo central del código. Para el funcionamiento del código se van a definir tres objetos diferentes. El primero de ellos es el objeto `Viga`, este alberga las funciones relativas a la generación, comprobación y valoración de todas las vigas generadas. El segundo, el objeto `Descendiente`, se encarga de crear un descendiente válido, por ello contiene las funciones relativas al cruce incluyendo la elección de ambos padres y las instrucciones para realizar la combinación de los genes de los mismos. Por último el objeto `Generación` administrar las diferentes generaciones e incluye los criterios de ordenación y de selección de los especímenes aptos para la siguiente generación.

### 2.2.1 Funcionamiento del código

La primera acción que tiene que realizar el código al ejecutarse es la correcta conexión con la API y captar el archivo que en ese momento el programa BIM tiene activo. Las diferentes APIs de los entornos BIM son en realidad un conjunto de librerías extra que necesitan ser cargadas en el código a partir de una serie de comandos. Estas librerías están diseñadas específicamente para cada BIM y no son compatibles entre ellos, en nuestro caso Autodesk Revit tiene varias librerías que definen la API para poder utilizar únicamente las necesarias.

Tras conectar el *plugin* con la API es necesario conectarlo con el archivo activo. Este punto difiere en gran medida en función del software BIM que se este utilizando, en el caso de Revit primero es necesario definir cómo interactúa el algoritmo con el archivo. Este punto se realiza definiendo el modo de transacción, cada una de las interacciones existentes entre el código y el archivo que realicen cambios sobre el archivo, aunque sólo se trate de captar datos, y el modo de regeneración, cómo se realizan las diferentes actualizaciones gráficas. Ambos pueden adquirir el valor de *Manual* o *Automático*. Para el primero escogemos el modo *Automatic*, así otorgamos al código los permisos necesarios para realizar cualquier intercambio de datos entre el *plugin* y

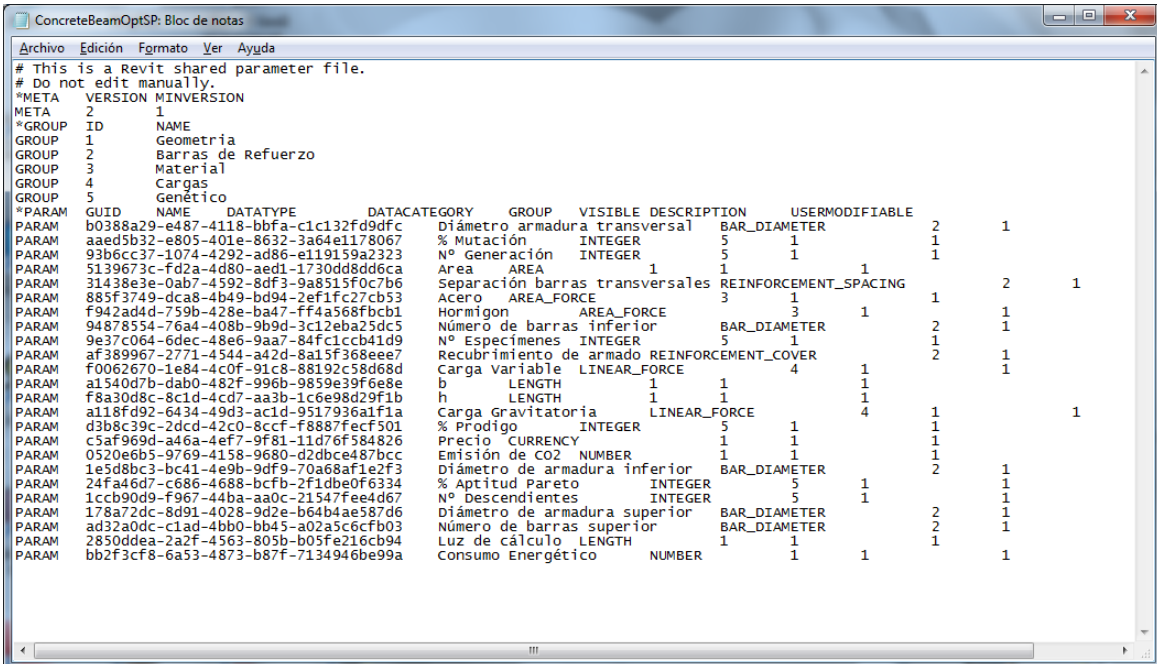
el archivo. La aplicación se va a crear para que únicamente sea necesario una actualización de los valores de los parámetros, y así mejorar su eficiencia, por lo que en realidad es indiferente el tipo de regeneración que asignemos, pero el modo *Manual* es el que evitará ralentizar el código al evitar ciertas comprobaciones durante la ejecución.

Una vez dadas las instrucciones de interacción se capta el archivo que en ese momento tiene el usuario abierto. Este paso es vital para el correcto funcionamiento de todo el *plugin*, ya que si no se realiza una correcta vinculación con el archivo pueden aparecer errores a la hora de intercambiar los diferentes parámetros. En el caso de nuestra aplicación primero tenemos que definir que se trata de un comando externo al propio programa, tras esto se inician los objetos propios de la API que contienen todos los datos del *Building Model* activo, en nuestro caso los objetos que realizan esta función son `UIDocument` y `Document`; además se inician los objetos que cargan las funcionalidades de la API en el archivo, `UIApplication` y `Application`. Estos objetos son únicos en Autodesk Revit. Con esta iniciación de objetos se ha creado una variable en el algoritmo que contiene el archivo.

Durante la ejecución de la aplicación, es posible que aparezca algún error que tenga que ser gestionado. En este caso se mostrará un mensaje informando sobre el error, la mayoría de APIs incorporan una serie de mensajes de errores genéricos que se seleccionan automáticamente cuando se detecta el error. Para gestionar los errores utilizando esta funcionalidad el cuerpo principal del algoritmo está contenido en un bucle de estructura `try...catch` que recoge todas las excepciones que se puedan producir durante la ejecución. Cualquier secuencia de código que difiera de la descrita (por ejemplo si se realiza una selección de más de un objeto simultáneamente) provocará una excepción que cancela la operación y muestra un mensaje con la descripción del error. En el apartado 2.2.7 se explicará con más detalle la gestión de errores.

Una vez iniciada la ejecución del código el usuario tendrá que seleccionar un elemento, que en este caso ha de pertenecer a la familia `ConcreteBeamOpt`. El algoritmo adquirirá los datos de este elemento a través de los parámetros del mismo. Una vez tomada la selección, que no puede contener más de un elemento, se verifica que la entidad seleccionada sea una familia para poder realizar un *cast* del tipo de objeto genérico al tipo de objeto familia que utilizaremos para adquirir los parámetros. Se realiza este cambio de definición del objeto porque la API que estamos utilizando necesita interpretar el objeto como una familia para poder acceder y editar sus parámetros. Durante la ejecución de esta parte de código al usuario le aparecerá un mensaje en la barra de diálogo indicándole que debe realizar una selección. Una vez se haya seleccionado un elemento, una ventana aparecerá frente al usuario confirmándole la selección.

Con el elemento correctamente identificado pasamos a buscar los valores de los parámetros del mismo. Para realizar esto empezamos definiendo los GUIDs de cada uno de los parámetros deseados a partir de la cadena de caracteres que obtenemos del fichero *Shared Parameters*. Con este tipo de parámetros quedandefinidos todos excepto la longitud de cálculo a partir de su GUID. Para lograr captar el valor de la longitud de cálculo utilizamos la definición interna del mismo, `INSTANCE_LENGTH_PARAM`. Con este dato y las definiciones anteriores se pueden obtener los valores de cada uno de los parámetros de la familia. Este proceso dependerá de cómo se hayan definido los parámetros dentro del software BIM utilizado.



```

ConcreteBeamOptSP: Bloc de notas
Archivo Edición Formato Ver Ayuda
# This is a Revit shared parameter file.
# Do not edit manually.
*META VERSION MINVERSION
META 2 1
*GROUP ID NAME
GROUP 1 Geometria
GROUP 2 Barras de Refuerzo
GROUP 3 Material
GROUP 4 Cargas
GROUP 5 Genético
*PARAM GUID NAME DATATYPE DATACATEGORY GROUP VISIBLE DESCRIPTION USERMODIFIABLE
PARAM b0388a29-e487-4118-bbfa-c1c132fd9dfc Diámetro armadura transversal BAR_DIAMETER 2 1
PARAM aaed5b32-e805-401e-8632-3a64e1178067 % Mutación INTEGER 5 1 1
PARAM 93b6cc37-1074-4292-ad86-e119159a2323 Nº Generación INTEGER 5 1 1
PARAM 5139673c-fd2a-4d80-ae11-1730dd8dd6ca Area AREA 1 1 1
PARAM 31438e3e-0ab7-4592-8df3-9a8513f0c7b6 Separación barras transversales REINFORCEMENT_SPACING 2 1
PARAM 885f3749-dca8-4b49-bd94-2ef1fc27cb53 Acero AREA_FORCE 3 1 1
PARAM f942ad4d-759b-428e-ba47-ff4a568fbc1 Hormigon AREA_FORCE 3 1 1
PARAM 94878554-76a4-408b-9b9d-3c12eba25dc5 Número de barras inferior BAR_DIAMETER 2 1
PARAM 9e37c064-6dec-48e6-9aa7-84fc1ccb41d9 Nº Especímenes INTEGER 5 1 1
PARAM af389967-2771-4544-a42d-8a15f368eee7 Recubrimiento de armado REINFORCEMENT_COVER 2 1
PARAM f0062670-1e84-4c0f-91c8-88192c58d68d Carga variable LINEAR_FORCE 4 1 1
PARAM a1540d7b-dab0-482f-996b-9859e39f6e8e b LENGTH 1 1 1
PARAM f8a30d8c-8c1d-4cd7-aa3b-1c6e98d29f1b h LENGTH 1 1 1
PARAM a118fd92-6434-49d3-ac1d-9517936a1f1a Carga Gravitatoria LINEAR_FORCE 4 1 1
PARAM d3b8c39c-2dcd-42c0-8ccf-f8887fecf501 % Prodigio INTEGER 5 1 1
PARAM c5af969d-a46a-4ef7-9f81-11d76f584826 Precio CURRENCY 1 1 1
PARAM 0520e6b5-9769-4158-9680-d2dbce487bcc Emisión de CO2 NUMBER 1 1 1
PARAM 1e5d8bc3-bc41-4e9b-9df9-70a68af1e2f3 Diámetro de armadura inferior BAR_DIAMETER 2 1
PARAM 24fa46d7-c686-4688-bcfb-2f1dbe0f6334 % Aptitud Pareto INTEGER 5 1 1
PARAM 1ccb90d9-f967-44ba-aa0c-21547fee4d67 Nº Descendientes INTEGER 5 1 1
PARAM 178a72dc-8d91-4028-9d2e-b64b4ae587d6 Diámetro de armadura superior BAR_DIAMETER 2 1
PARAM ad32a0dc-clad-4bb0-bb45-a02a3c6cfb03 Número de barras superior BAR_DIAMETER 2 1
PARAM 2850ddea-2a2f-4563-805b-b05fe216cb94 Luz de cálculo LENGTH 1 1 1
PARAM bb2f3cf8-6a53-4873-b87f-7134946be99a Consumo Energético NUMBER 1 1 1

```

Fig. 15 Fichero con los Shared Parameters

Los valores de estos parámetros se asignan a las diferentes variables utilizadas en el algoritmo para la correspondiente optimización. La función `FamilyInstance.Symbol.get_Parameter(GUID)`, utilizada en Revit para esta finalidad obtiene los valores como `string`, por lo que ha sido necesario introducir explícitamente un cambio del tipo de variable para que funcionen de manera adecuada. Además se puede observar en el código del Anexo 1 que algunos de los parámetros han sido ajustados mediante operaciones matemáticas. Esto tiene que ver con las unidades de los mismos y con el modo de trabajo de Revit de modo interno. Tanto el código del *plugin* como la definición de los parámetros se ha realizado pensando en la utilización de unidades del Sistema Internacional de Unidades (SI) como recomienda la EHE y como suele ser habitual en España. En teoría, las unidades del archivo de Revit pueden ser definidas por el usuario, pero esto no completamente cierto. Los valores existentes en la memoria de Revit están expresados en las unidades del Sistema Anglosajón de Unidades (es a estos valores a los que

accedemos con la API), mientras que los valores mostrados son la conversión de estas unidades a las unidades que el usuario desea. Esto queda demostrado si observamos el dato numérico y el dato mostrado a través de la Revit LookUp Tool.

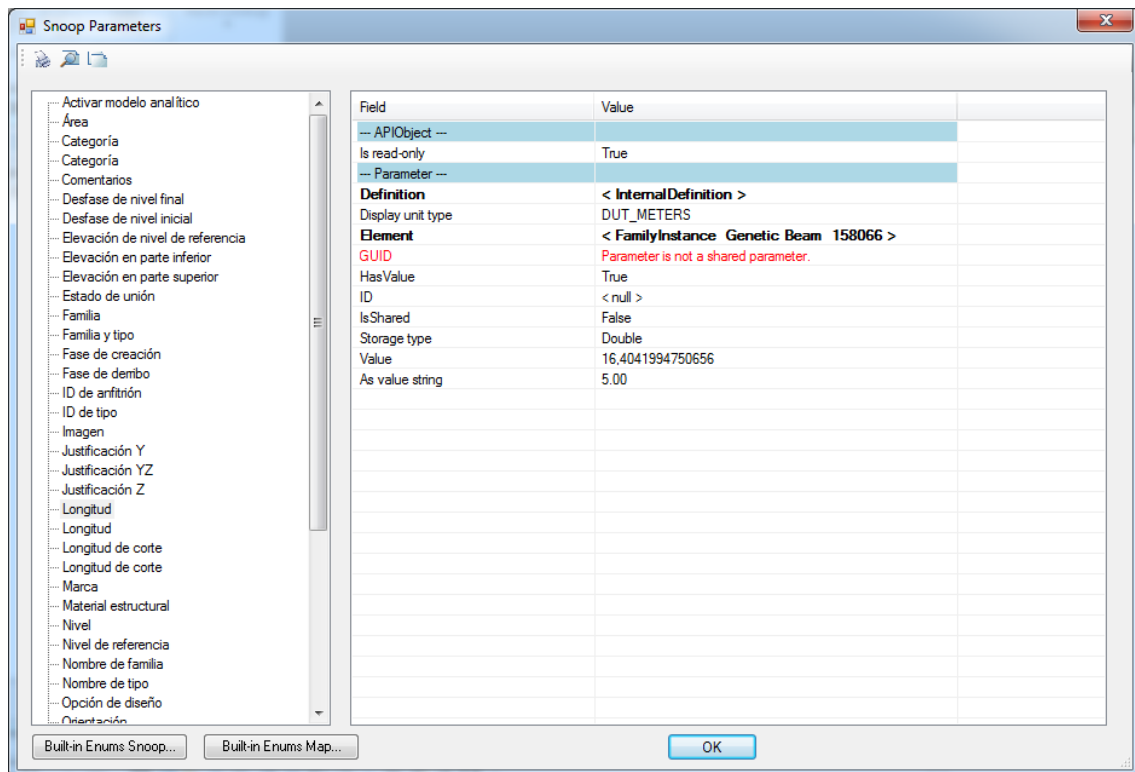


Fig. 16 Imagen Revit LookUp Tool comparando valor interno de Revit (Value) con valor mostrado (As ValueString)

Por lo tanto, para trabajar con unidades del SI y así evitar errores, se han utilizado factores de conversión a la hora de adquirir estos parámetros. A estos factores se ha llegado mediante la creación de rectas de regresión generadas entre los valores introducidos por el usuario (unidades del SI) y los valores obtenidos tras la asignación de datos (unidades anglosajonas). Se ha utilizado este método porque no existe información pública que indique en qué unidades del sistema anglosajón funciona cada una de las categorías paramétricas de Revit a la cual quedan asignados cada uno de los parámetros.

### 2.2.2 Funcionamiento del algoritmo

Tras asignar los valores de los parámetros a las diferentes variables podemos ejecutar el resto del algoritmo para obtener el resultado óptimo, esta parte de la aplicación no depende del entorno BIM que se utilice a diferencia de los pasos explicados hasta el momento. Para ello tras iniciar una serie de variables auxiliares, las cuales nos permiten configurar algunas funcionalidades del algoritmo, cómo el criterio de optimización, de una manera cómoda.



A partir de este punto se inicia un bucle que establece la población inicial de vigas válidas. Los especímenes son generados a partir del estado de cargas y de la longitud de cálculo establecida, de manera aleatoria en la función `Viga.generaviga`. En esta función también están incluidas las labores de validación y valoración de la viga. Tras la generación de cada espécimen este es almacenado en la matriz `poblacion` que recoge los datos de todos los sujetos analizados en cada una de las generaciones.

Una vez se ha generado la primera población se ha de iterar entre las generaciones, produciendo una nueva `poblacion` en cada generación a partir del cruce de dos o más sujetos de la `poblacion` inicial de esa generación utilizando un bucle anidado. El bucle exterior controla el número de generaciones que han sucedido desde que ha comenzado la generación del algoritmo, mientras que el bucle interno controla el número de descendientes creados en cada generación.

Estos descendientes creados a partir de la función `Generación.hijo` son almacenados en la matriz `descendencia`. Una vez se han realizado todos los cruces genéticos y la matriz `descendencia` esta completa, se procede a combinar esta matriz con la matriz `poblacion` y realizar la selección de los mejores especímenes mediante un criterio elitista contenido en la función `Generacion.elitismo`. El *output* de esta función es la nueva matriz `poblacion` utilizada en la próxima generación. El proceso se repite hasta que se cumplan el número de generaciones definido por el usuario. Al tratarse, por el momento, de una optimización monocriterio el espécimen óptimo es el primer sujeto de la matriz `poblacion` una vez esta ha sido ordenada.

### 2.2.3 Volcado de los resultados

Una vez generado y seleccionado el óptimo, nuestro problema de optimización ha terminado. El único proceso que le quedaría por realizar a la aplicación es el volcado del sujeto al entorno BIM utilizado y la actualización del *Building Model* con la introducción de los parámetros correspondientes. Este paso vuelve a necesitar funciones propias de la API del software BIM utilizado, en el caso de ejemplo la función necesaria es la que se muestra en la Figura 17(a). Esta función solo admite como entrada el valor del nombre del parámetro. Para averiguar este nombre utilizaremos la función que muestra en la Figura 17(b) y lo almacenaremos en una variable de texto. De este modo asignaremos los valores obtenidos a los parámetros correspondientes de la familia.

```
FamilyInstance.Symbol.LookupParameter(nombre).Set(valor) (a)
FamilyInstance.Symbol.get_Parameter(GUID).Definition.Name (b)
```

Fig. 17 Funciones para asignar valores a parámetros en la API de Revit

Del mismo modo que al captar los parámetros, al realizar el volcado tenemos que solucionar el problema de las unidades. El procedimiento ha sido análogo al caso anterior, estableciendo rectas de regresión que ajusten el valor de unidades del SI utilizadas en el algoritmo a unidades anglosajonas utilizadas internamente por Revit. Al realizar estas rectas, los coeficientes de correlación siempre han sido de valor igual a la unidad, algo estadísticamente imposible si no existiese de antemano una relación lineal entre las variables (como es lógico al tratarse de un simple cambio de unidades).

Antes de finalizar, el algoritmo muestra al usuario los resultados de coste, emisión de CO<sub>2</sub> y consumo energéticos obtenidos como óptimos a través de una ventana emergente en la que se van mostrando los valores. Con esto finaliza la ejecución del plugin y se actualiza el dibujo con los nuevos valores de las variables, obteniendo así la viga óptima para cualquiera de los tres criterios de optimización definidos.

#### 2.2.4 Objeto: Viga

En la explicación anterior se ha descrito el funcionamiento general del algoritmo de optimización sin entrar en detalle en los objetos que lo componen. En los apartados subsiguientes se va a explicar de manera detallada como funciona cada uno de estos objetos, cuál es su objetivo y qué funciones tiene incluidas. Gracias a esto podremos comprender con más exactitud el comportamiento del *plugin* desarrollado.

El primer objeto que vamos a estudiar es el objeto `Viga`. La finalidad de este objeto es la generación aleatoria de vigas válidas y valoradas. Para cumplir con este objetivo tiene cinco funciones, `Viga.generaviga`, `Viga.comprueba`, `Viga.valoracoste`, `Viga.valoraco2` y `Viga.valoraenergia`.

La función principal es la función `Viga.generaviga` que es la encargada de generar aleatoriamente un sujeto viga capaz de cumplir los requisitos impuestos por las otras funciones. La primera acción que se realiza es la generación de los valores de los distintos parámetros que componen la viga. Esta generación se ha realizado estableciendo unos valores máximos y mínimos a cada valor que provoquen un resultado lógico, habitual en edificación y por encima de los requisitos previsible. Cada valor, tiene también un incremento distinto para disminuir la casuística. Los valores máximos y mínimos quedan recogidos en la Tabla 3.

| Parámetro   | Valor Mínimo         | Valor Máximo         | Intervalo           |
|-------------|----------------------|----------------------|---------------------|
| <b>b</b>    | 25 mm                | 1200 mm              | 5 mm                |
| <b>c</b>    | 25 mm                | 1200 mm              | 5 mm                |
| <b>H</b>    | 25 N/mm <sup>2</sup> | 50 N/mm <sup>2</sup> | 5 N/mm <sup>2</sup> |
| <b>D1</b>   |                      | Especial             |                     |
| <b>D2</b>   |                      | Especial             |                     |
| <b>D3</b>   |                      | Especial             |                     |
| <b>n1</b>   | 1                    | 26                   | 1                   |
| <b>n2</b>   | 1                    | 26                   | 1                   |
| <b>sep3</b> | 50 mm                | c                    | 5 mm                |

Tabla 3 Valores posibles de la viga generada

La única excepción a esto son los valores de los diámetros. Estos valores tienen que estar ceñidos a los diámetros de las barras existentes en el mercado, por lo tanto se ha utilizado una matriz con estos valores y el algoritmo selecciona aleatoriamente uno de ellos en cada caso.

Una vez generada la viga esta se comprueba y únicamente si se trata de un sujeto válido se valora y se da como espécimen apto. En caso contrario se genera una nueva viga siguiendo el mismo procedimiento. Esta labor la realiza la función `Viga.comprueba`, que tiene como tarea dar por válida o descartar una viga, en función de si resiste o no el estado de cargas definido para la misma. La validación se realiza verificando los estados límites últimos y de servicio necesarios (resistencia, equilibrio, deformación y fisuración) siguiendo las directrices de la EHE-08. Un sujeto válido también debe cumplir las prescripciones constructivas de separación de barras establecidas en la misma norma.

Toda viga introducida en el algoritmo de comprobación es válida por defecto y recorre una serie de comprobaciones. Si el resultado de todas estas comprobaciones es válido, la viga se acepta como válida; por el contrario, si falla una comprobación, el algoritmo se detiene y obtiene un resultado de no válido que descarta el elemento. Las comprobaciones no están dispuestas en el orden más lógico si pensamos en la resolución del problema de comprobación de un modo manual. El orden se ha alterado para aumentar la eficiencia del algoritmo. Estas comprobaciones están situadas de menor a mayor complejidad computacional, es decir las operaciones más sencillas se sitúan al inicio y las más complejas al final. Se ha optado por este orden porque al obtener una única comprobación negativa la viga se descarta y de este modo se evita realizar cálculos innecesarios.

A continuación y si la viga generada es válida se empieza con las diferentes valoraciones. Tres de los algoritmos de valoración funcionan de un modo equivalente, por este motivo solo se detallará el de valoración del coste. La valoración de las emisiones de CO<sub>2</sub> y del consumo energético se realiza del mismo modo, variando los valores del criterio estudiado. Los datos necesarios para la valoración se han extraído de la base de datos del ITeC y se introducen

manualmente en el código. No existe manera para el usuario de cambiar estos datos sin acceder y editar el mismo.

El planteamiento consiste en calcular la cantidad de cada material empleada en la viga y posteriormente multiplicarla por el precio del material. Para el coste del acero se han cuantificado los metros lineales de armado de cada uno de los diámetros diferentes y se ha multiplicado por el peso específico del material para obtener los kilogramos de acero utilizados en el elemento. El coste del hormigón se obtiene a partir del volumen bruto de la pieza. Para valorar el encofrado se ha considerado que este es de madera, ya que este tipo de encofrados tienen un mucho menor impacto medioambiental con un incremento de coste inexistente. El precio del mismo se calcula teniendo en cuenta los metros cuadrados de superficie a tapar para realizar el moldeo de la viga. De manera análoga se ha creado el código que cuantifica las emisiones de CO<sub>2</sub> producidas y el consumo energético durante la construcción de la viga. La diferencia radica en los valores que toman los diferentes materiales utilizados.

#### 2.2.5 Objeto: Generación

El siguiente objeto que analizaremos es el objeto `Generacion`, la misión principal de este objeto consiste en gestionar la creación de la siguiente generación. Esto lo realiza estableciendo cómo será el espécimen resultante de la procreación y seleccionando los especímenes que sobrevivirán a esta generación. Para cumplir estos dos objetivos utiliza sus tres funciones `Generacion.hijo`, `Generacion.elitismo` y `Generacion.sort`. El objeto `Generacion` es el objeto principal el algoritmo y de su correcto funcionamiento depende el éxito del algoritmo.

La función `Generacion.hijo` es la encargada de controlar la procreación de los padres seleccionados para crear un sujeto válido que surja de la combinación de estos dos progenitores. Además, al crear el sucesor es cuando se establece si cada uno de los genes del mismo muta y difiere de los valores o no. Además, en caso de que se haya determinado que el espécimen ha de ser un hijo pródigo, espécimen externo a la población y generado aleatoriamente, se generará en este punto.

Esta función puede dividirse en varias partes para entenderla. Primero la función inicia el objeto `Descendiente`, explicado a continuación, y mediante este selecciona ambos progenitores, `padre` y `madre`, y los puntos de corte del código genético de cada uno, `ru11` y `ru12`. Una vez seleccionados estos datos, se procede a la procreación. Para ello se copian los genes de los padres a la matriz `hijo` siguiendo cierto orden. Primero se copian los genes de `padre`, hasta que se llega al gen número `ru11`. En este momento los genes copiados pasan a ser los de `madre`. Por último, cuando se alcanza el gen `ru12` se vuelve a invertir nuevamente la copia y son los genes

de padre los que se asignan a hijo. Además, por cada gen copiado se genera una posibilidad, `mut`, de que el gen mute y adquiera el valor de otro miembro de la población. La imagen muestra gráficamente la creación de un hijo.

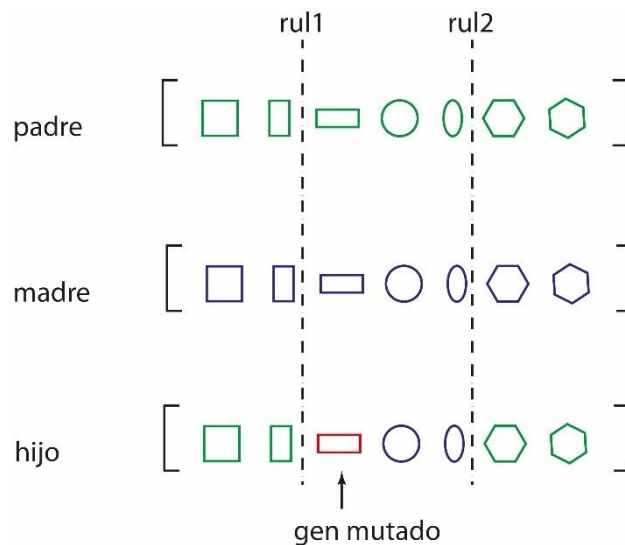


Fig. 18 Combinación genética entre progenitores

Lógicamente, `hijo` tiene que cumplir los requisitos de cualquier miembro de la población para ser considerado válido. El sujeto generado se somete a las comprobaciones realizadas en `Viga.comprueba` y si resulta válido, se valora con las funciones explicadas anteriormente. En caso de no superar la comprobación, el sujeto se descarta y se seleccionan unos nuevos progenitores.

Por último se verifica si el hijo ha de ser un hijo pródigo, espécimen ajeno completamente a la población. De ser así se descarta el sujeto generado hasta el momento y se genera uno nuevo como si de un miembro de la población inicial se tratase, es decir a través de la función `Viga.Generaviga`. Este factor ayuda a mantener la diversidad dentro del algoritmo genético y a que el algoritmo no se estanque en un óptimo local sino que tenga la posibilidad de incorporar características genéticas de otro conjunto de genes. Tras la generación del conjunto de hijos, hemos visto que los resultados de la matriz `poblacion` y `descendencia` se combinan para obtener una matriz `poblacion` nueva que determina la población existente en la próxima generación de sujetos, descartando de este modo las entidades menos adecuadas. Esta selección se realiza con un criterio elitista puro, los especímenes valorados con un mayor valor del criterio seleccionado son eliminados.

El funcionamiento es muy sencillo. Se crea una matriz `generacion` donde se copian todos los datos de `poblacion` y `descendencia`. Tras esta copia masiva, la matriz `generacion` se ordena de menor a mayor en la función `Generacion.sort`. Una vez ordenada se vuelven a copiar los datos a la matriz `poblacion` sobrescribiendo la misma hasta que todos los datos hayan sido sustituidos.

Con esto se crean los nuevos valores de la matriz `poblacion` combinando todos los especímenes generados.

En C# no existe ninguna orden almacenada en una librería que ordene automáticamente de menor a mayor una matriz, aunque sí existen algoritmos creados que sirven a esta finalidad y funciones que la ordenan comparándola con otra matriz base. Por ello ha sido necesario crear la función `Generacion.sort`, basa en una modificación de uno de los algoritmos de ordenación existentes denominado *Bubble-sort method*.

Este algoritmo consiste en comparar una fila de la matriz con la siguiente. Si la segunda fila es menor a la primera, se invierten las posiciones de las mismas, en caso contrario se dejan en la posición actual y se procede a comprobar el siguiente par. La imagen 10 muestra la lógica detrás de la programación.

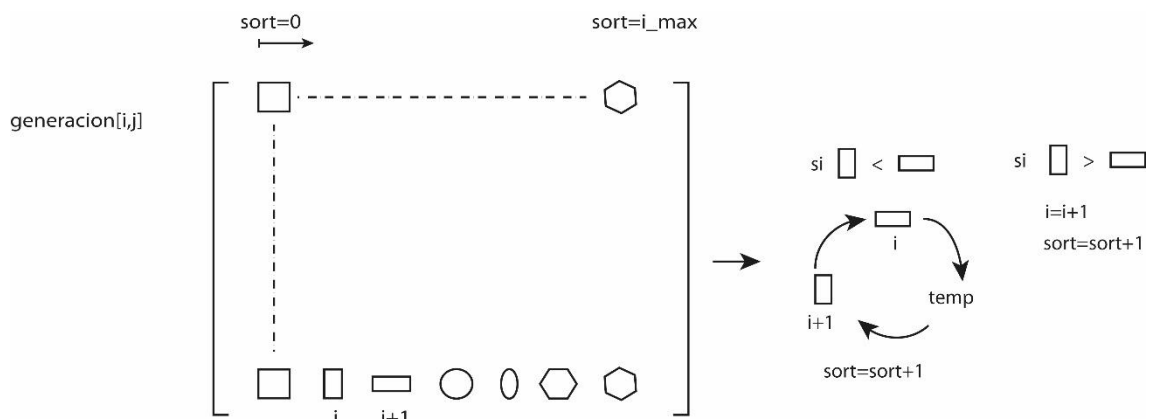


Fig. 19 Funcionamiento del algoritmo Bubble Sort

En nuestro caso queremos ordenar una matriz bidimensional, por ello tenemos que realizar una ligera modificación al algoritmo propuesto. Si la comparación resulta positiva, en lugar de cambiar de posición únicamente un valor de la matriz se cambia toda esa columna. Por ello la matriz auxiliar `temp` tiene que tener la longitud idéntica a la de las columnas de la matriz `generacion` que se quiere ordenar. Introduciendo la modificación obtenemos el código.

### 2.2.6 Objeto: Descendiente

El último objeto, `Descendiente`, es un objeto que establece las pautas que se siguen al generar un hijo mediante la función `Generacion.hijo` estudiada anteriormente. Está compuesto por siete funciones diferentes que determinan la fisionomía del nuevo espécimen creado. Estas, a diferencia de las anteriores, se pueden agrupar, ya que algunas de ellas tienen la misma finalidad y todas se utilizan en la función `Generacion.hijo`. Las funciones que componen este objeto son `Descendiente.elcpadre`, `Descendiente.elcmadre`, `Descendiente.ruleta`,

`Descendiente.ruleta2`, `Descendiente.mutacion`, `Descendiente.seleccmutagen` y `Descendiente.prodigo`.

Las dos funciones `Descendiente.elcpadre` y `Descendiente.elcmadre` tienen un mismo objetivo, seleccionar los sujetos de la matriz `poblacion` que serán utilizados como base para la combinación genética que dará lugar a `hijo`. El valor devuelto como resultado es simplemente un valor numérico entero que determina la fila en la cual queda almacenado el sujeto seleccionado de la matriz `poblacion`.

En principio se podría pensar que si ambas funciones tienen la misma finalidad se podría llamar a la misma función dos veces simultáneamente y obtener dos resultados diferentes. Esto sería una aproximación que se ha evitado voluntariamente. La función `Generacion.elcmadre`, aunque en esencia es igual a `Generacion.elcpadre`, tiene una diferencia vital para el comportamiento del algoritmo, ya que se prohíbe que `madre` sea igual a `padre` y así se evita lo que se conoce como incesto. El incesto en un GA consiste en que el hijo sea generado a partir de un único progenitor que ocupa el lugar de ambos. Evitar esto es primordial, ya que reduce la variabilidad de la población y hace que el algoritmo tienda a estancarse en un punto alejado del óptimo al crear una población más idéntica entre sí.

Una vez se han seleccionado los progenitores es necesario seleccionar cómo se van a combinar los diferentes genes de cada uno de ellos del modo, finalidad de las funciones `Descendiente.ruleta` y `Descendiente.ruleta2`. Esta pareja de funciones se encarga de configurar la combinación genética de los padres para generar a `hijo` explicada en `Generacion.hijo`.

Para este fin, ambas funciones devuelven un valor entero que marca el gen en el que se combinan, según lo explicado en la imagen 9. Este valor representa una fracción de una ruleta virtual a la cual ha sido asignada cada gen, al hacer rodar esta se obtiene una porción de ruleta entre los dos factores la cual es copiada de un progenitor a otro para generar el descendiente. Al funcionar de este modo es imprescindible que el valor devuelto por la función `Descendiente.ruleta2` sea superior a `Descendiente.ruleta`. Además, ambos valores han de estar contenidos dentro del rango de valores de la matriz `hijo` donde se almacenan las características que definen a la misma. Es innecesario copiar las valoraciones de cualquiera de los progenitores a `hijo`, dado que se trata de un espécimen totalmente nuevo.

Como el algoritmo está pensado para solventar problemas sencillos con un espacio de soluciones limitado, se considera innecesaria la idea de generar dos hijos por cada par de progenitores. Esto sería fácilmente implementable si se aplica el mismo criterio a otros

problemas, simplemente sería necesario crear un segundo hijo manteniendo los valores de `Descendiente.ruleta` y `Descendiente.ruleta2` pero invirtiendo los *roles* de `padre` y `madre`, es decir la fracción de genes contenida en la porción de ruleta seleccionada provienen de `padre` y no al revés.

Cuando cada uno de estos genes es copiado al descendiente es posible que mute, es decir que su valor varíe, este hecho viene determinado por el resultado de la función `Descendiente.mutacion`, que determina un número al azar entre 1 y 100. Si este valor es inferior a la probabilidad de mutación establecida como parámetro del algoritmo, el gen mutará y no coincidirá con el de sus progenitores. Si la mutación es efectiva, es necesario crear el valor del nuevo gen mutado, por ello si el gen resulta mutado se utiliza la función `Descendiente.selecmutagen`. Para generar este nuevo gen se genera temporalmente un espécimen extra del cual únicamente se utiliza el gen correspondiente del mismo modo que se crearía un sujeto de la primera generación. De este se extrae el valor del gen correspondiente y se incrusta a la matriz `hijo` en la debida posición. Una vez se ha extraído el valor deseado el resto del espécimen se descarta. Si en un mismo descendiente se generan dos o más mutaciones en sus genes, cosa que no está controlada aunque es improbable, se repite este mismo proceso por todos y cada uno de los genes. Aunque es cierto que esto ralentiza el algoritmo, ya que se genera un mayor número de sujetos e interrupciones en el desarrollo normal del algoritmo, también aumenta en gran medida la eficacia del mismo al aumentar la variabilidad y la combinatoria de las diferentes soluciones, este hecho se demuestra sobretodo en las generaciones finales en las cuales los diferentes sujetos son bastante similares y las mutaciones aportan gran parte de las diferencias entre ellos.

No es recomendable que el porcentaje de mutación sea elevado (superior al 10%), puesto que esto produciría que un gran número de genes mutaran haciendo que el cruce entre los progenitores tuviera una relevancia prácticamente nula al diferir mucho el resultado final de la combinación inicial. Por lo tanto, de escoger un porcentaje elevado no se puede asegurar la convergencia del algoritmo en un óptimo, pues cada descendiente estaría más cercano a un espécimen generado independientemente, que a un espécimen generado a partir de la combinación de las características de los progenitores.

Aunque sí que se mejora la eficacia del algoritmo introduciendo de manera esporádica un nuevo espécimen ajeno a la población existente, lo que se conoce como un hijo prodigo. La función `Descendiente.prodigo` genera un número entre 1 y 100, si este valor es inferior a la probabilidad de que se cree un hijo prodigo establecida por el usuario, el sujeto `hijo` creado en esa iteración no se crea por combinación de los progenitores, sino que se genera de manera



aleatoria e independiente. Esto permite al algoritmo comprobar de manera periódica si existe una región del espacio de soluciones alejada de la zona de estudio actual con un conjunto de soluciones mejores a las actuales y si es así trasladarse a esta nueva región. Si el pródigo generado mejora la solución actual, este se irá combinando con la población, influyendo altamente en futuras generaciones. Si, por el contrario, el pródigo empeora la solución actual, este desaparecerá. Del mismo modo que con la mutación, hay que ser muy precavidos al utilizar este parámetro ya que de ser demasiado influyente invalidaría todo el algoritmo.

### 2.2.7 Gestión de Errores

Todo el código expuesto hasta el momento está ejecutado dentro de un bucle `try...catch`. Este es un bucle inherente a los lenguajes de programación, incluyendo C#, que se utiliza para gestionar los posibles errores de ejecución que puedan aparecer y añadir excepciones que finalicen la ejecución e informen al usuario de dónde está el error. Es necesario establecer una sección `catch` para cada excepción que se quiera gestionar. En nuestro caso se definen dos aprovechando que la propia API de Revit nos proporciona una librería con ordenes adecuadas para las excepciones más habituales que pueden aparecer, la librería `Autodesk.Revit.Exceptions`.

La primera de ellas consiste en cancelar la ejecución del *plugin*, la excepción se activará si el usuario pulsa la letra ESC del teclado en cualquier momento durante la ejecución de la herramienta. Este evento únicamente puede suceder si el usuario pulsa voluntariamente esta tecla, por lo tanto y como es habitual para el resto de comandos de Revit, la ejecución se ve finalizada sin devolver ningún mensaje, pues se considera que el usuario es consciente de que la operación ha sido cancelada. La segunda, se captura utilizando las librerías de C#, esta excepción captura cualquier anomalía producida durante la ejecución del algoritmo. Tras ser capturada, expulsa un mensaje de error al usuario explicando el problema y finaliza la ejecución. El mensaje de error sí que es extraído de la librería de Revit, que contiene una serie de códigos de error que informan de todos los conflictos internos del *Building Model*. Dada la manera de introducir los elementos estructurales en Revit y como se ha modelado la familia *ConcreteBeamOpt* es imposible que aparezcan problemas de colisión geométrica entre diferentes objetos, por lo que no se ha añadido ningún criterio para gestionar los mismos. Es posible que este problema sea susceptible de aparecer en otros software BIM y sea necesario añadir una excepción que priorice una de las dos geometrías, la del elemento del *Building Model* en conflicto o la de la viga optimizada, sobre la otra.

## 2.3 PLANTEAMIENTO DEL ALGORITMO MULTICRITERIO

Hasta el momento se ha descrito como desarrollar un *plugin* para Revit completamente funcional y operativo capaz de optimizar un elemento estructural tipo viga isostática para un único criterio, ya sea este el criterio económico, de emisiones de CO<sub>2</sub> o de consumo energético o minimizar el número de barras. Esto difiere de nuestro objetivo, aunque era un paso necesario, desarrollar una aplicación de optimización multiobjetivo que facilita la consecución del mismo.

La optimización multiobjetivo se va a plantear a partir de la denominada Frontera de Pareto. En la imagen siguiente podemos observar una distribución cualquiera de puntos en función de dos criterios diferentes (por ejemplo coste y emisión de CO<sub>2</sub>). La frontera de Pareto dibujada es la línea que une los puntos situados en una posición más cercana al origen de coordenadas. Se puede afirmar que cualquier punto existente en esta frontera supone un óptimo en esa distribución teniendo en cuenta ambos criterios.

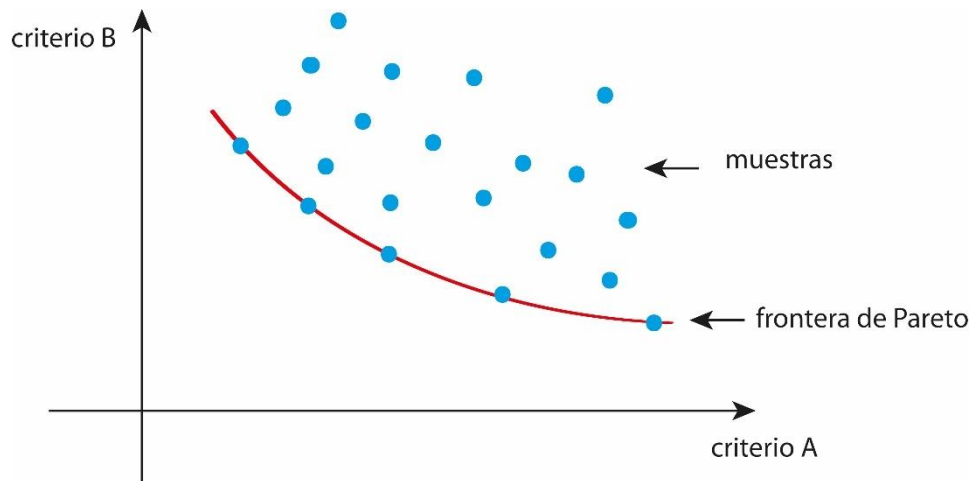


Fig. 20 Diagrama de Pareto

Observando el diagrama se aprecia claramente que los valores pertenecientes a la frontera son inversamente proporcionales entre sí, es decir al aumentar uno, el otro disminuye. Esto marca la estrategia a seguir para programar la creación de la frontera de Pareto. De estos puntos, cualquiera sería “una buena opción”, siendo los más óptimos en función de cada criterio los valores situados en cada extremo y los valores centrales suponiendo una muy buena combinación de ambos parámetros. Nuestro algoritmo tiene que ser capaz de elegir uno de estos puntos, por lo tanto será necesario incorporarle un criterio.

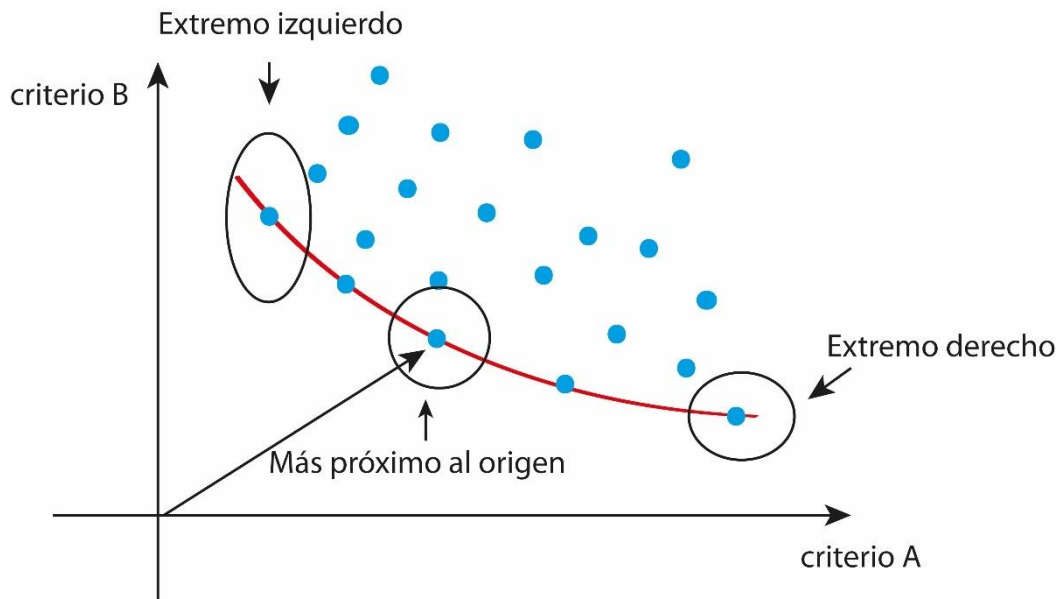


Fig. 21 Ubicación de los óptimos en el diagrama de Pareto

Se puede determinar tres puntos en una frontera de Pareto que, a primera vista, suponen una opción claramente superior a las otras. El extremo izquierdo de la frontera de Pareto supone el menor valor para la variable representada en el eje horizontal. Si esta es la variable principal sería la opción más recomendable a seleccionar. En cambio, si la variable principal es la variable representada en el eje vertical, el mejor punto a escoger queda situado en el extremo derecho. A pesar de esto, el valor más cercano al origen obtenido utilizando diferentes razones trigonométricas es la mejor solución de compromiso entre ambas variables, es decir será el punto con menor valor de la variable del eje vertical sin que se aumente demasiado la variable del eje horizontal. En la explicación hemos utilizado el término “variable principal”, con esta acepción nos referimos a la variable más influyente en el diagrama. Es este último el que deseamos encontrar, ya que es el que menos deteriora una variable respecto a la segunda.

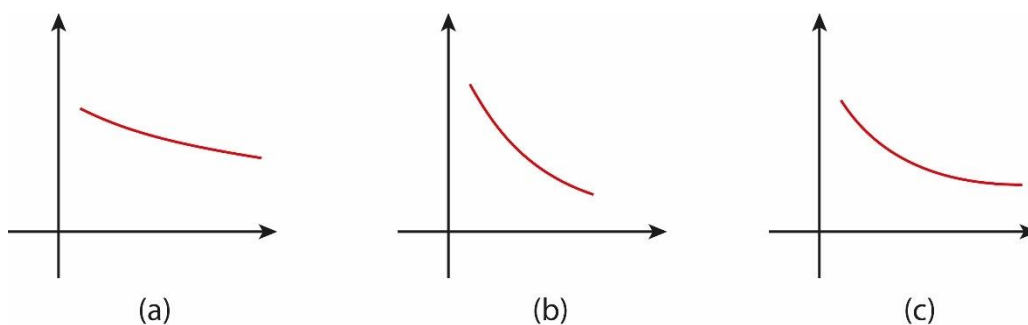


Fig. 22 Diferentes diagramas de Pareto

Qué criterio elegir puede depender del problema al que nos enfrentemos y de las variables que estemos comparando. La imagen 22(a) es un ejemplo en el que es conveniente seleccionar el extremo izquierdo como óptimo, ya que la frontera de Pareto tiene una forma muy tendida, y

escoger este extremo nos proporcionará una mayor ventaja. En cambio, la imagen 22(b) al estar invertida, la gráfica es al contrario, por el mismo motivo sería una mejor opción tomar el extremo derecho. Por último, la imagen 22(c) nos muestra una distribución de Pareto más clásica, en este caso la mejor solución (sin tener en cuenta otros condicionantes) es una solución de compromiso entre ambas variables, el valor más cercano al origen. Ninguno de estos criterios es absoluto y pueden verse condicionados por otros factores inherentes al problema de optimización estudiado en cada caso.

La explicación de la frontera de Pareto se ha realizado utilizando un ejemplo con dos criterios de optimización, mientras que nuestro objetivo es realizarlo con cuatro. El razonamiento es exactamente el mismo, solo que en lugar de comparar dos variables se comparan cuatro y un ejemplar pertenece a la frontera de Pareto si se produce una mejora en una variable sin empeorar el resto.

Independientemente de esto, es necesario estudiar las relaciones entre las variables o criterios de optimización para tomar las decisiones adecuadas. La aplicación monocriterio se ha creado con este objetivo, ya que si se determina que dos o más de los criterios están relacionados entre ellos de manera directamente proporcional, se podría eliminar uno de ellos a la hora de realizar la Frontera de Pareto. Puesto que el hecho que un espécimen pertenezca a la frontera en un criterio asegura la pertenencia a la misma con el segundo y viceversa.

### 2.3.1 Estudio de los diferentes criterios de optimización y su relación

Para el estudio de los criterios de optimización y sus relaciones nos vamos a valer de técnicas estadísticas comparándolos dos a dos para poder observar las relaciones. El proceso consiste en ejecutar veinte veces el código con cada uno de los criterios de optimización diferentes. Tras esta batería de ejecuciones, se representarán en un sistema de ejes cartesianos siendo el eje horizontal la variable optimizada y el eje vertical la variable estudiada. Con esto podemos establecer diferentes fronteras de Pareto y estudiar la relación entre las variables. Por este motivo se ha introducido en el código inicial la opción de cambiar el criterio de optimización de una forma rápida y accesible.

Todas las ejecuciones del código se han realizado con la misma configuración para el algoritmo. Uno de los motivos por los que se ha realizado un número alto de ejecuciones es debido a que el algoritmo no se ha ajustado para cada problema. Por lo tanto, para tener la seguridad de que se obtiene el óptimo, los parámetros deberían ser ajustados de forma estadística. Esto no se ha realizado porque los mejores valores de los parámetros son distintos según el criterio de optimización y se ha optado por unos parámetros con unos valores estándar y un aumento en

el número de ejecuciones. Aún así para dar la muestra por válida se la someterá a una prueba de normalidad para verificar que los resultados no tienen una alta dispersión.

| Parámetros  |     |
|-------------|-----|
| max_espe.   | 100 |
| max_descen. | 100 |
| gen.        | 50  |
| mut.        | 5   |
| prod.       | 2   |

Tabla 4 Parámetros del GA utilizados

| COSTE |         |                 |         |        |
|-------|---------|-----------------|---------|--------|
| N     | Coste   | CO <sub>2</sub> | Energía | Barras |
| 1     | 1057,65 | 2098,66         | 6838,72 | 116,00 |
| 2     | 992,74  | 1927,66         | 6218,91 | 64,00  |
| 3     | 826,84  | 1583,33         | 5127,68 | 100,00 |
| 4     | 1084,68 | 2154,52         | 7017,78 | 66,00  |
| 5     | 980,17  | 1918,10         | 6219,78 | 58,00  |
| 6     | 817,47  | 1543,04         | 4944,28 | 30,00  |
| 7     | 832,47  | 1610,43         | 5268,95 | 87,00  |
| 8     | 774,15  | 1477,53         | 4822,16 | 77,00  |
| 9     | 971,25  | 1932,32         | 6366,86 | 38,00  |
| 10    | 956,14  | 1898,98         | 6250,65 | 192,00 |
| 11    | 855,44  | 1640,10         | 5327,63 | 112,00 |
| 12    | 1000,89 | 1963,55         | 6372,23 | 49,00  |
| 13    | 920,24  | 1791,50         | 5841,99 | 112,00 |
| 14    | 740,47  | 1400,80         | 4579,78 | 62,00  |
| 15    | 767,46  | 1459,53         | 4749,88 | 39,00  |
| 16    | 927,78  | 1821,59         | 5953,53 | 72,00  |
| 17    | 998,99  | 1960,48         | 6364,77 | 197,00 |
| 18    | 829,33  | 1587,94         | 5139,02 | 64,00  |
| 19    | 967,97  | 1909,47         | 6235,95 | 53,00  |
| 20    | 883,56  | 1706,60         | 5529,08 | 79,00  |

Tabla 5 Valores de la optimización en base al coste

El primer criterio estudiado es el coste, el resultado de las veinte optimizaciones se muestra en la Tabla 5. La prueba de normalidad de Kolmogorov-Smirnov nos confirma que se trata de una población normal en las cuatro distribuciones. Por lo tanto podemos aceptar esta muestra y estudiar su distribución. Representando los valores obtenemos los gráficos X y Y.

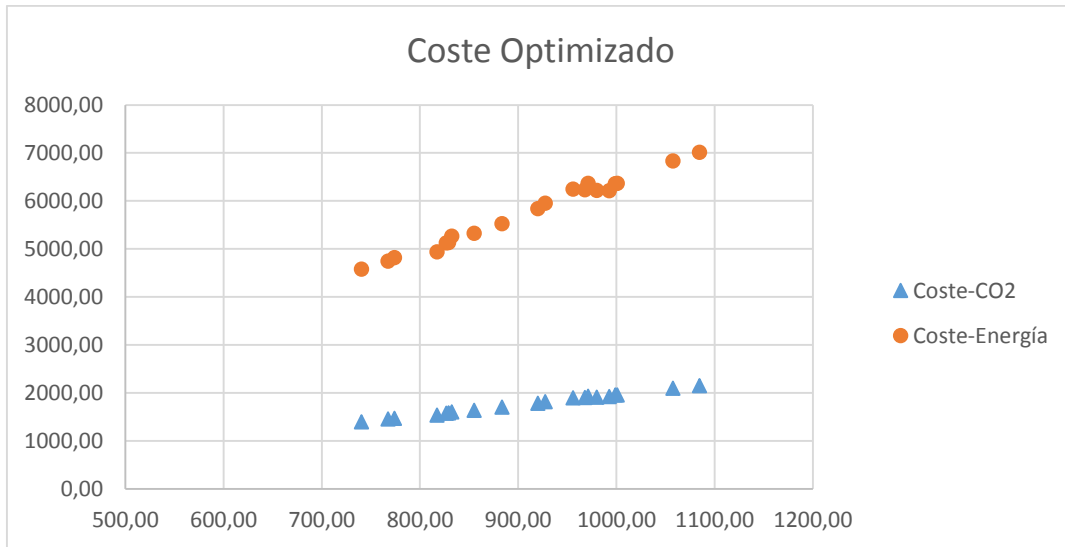


Gráfico 4 Relación del coste con emisión de CO2 y consumo energético

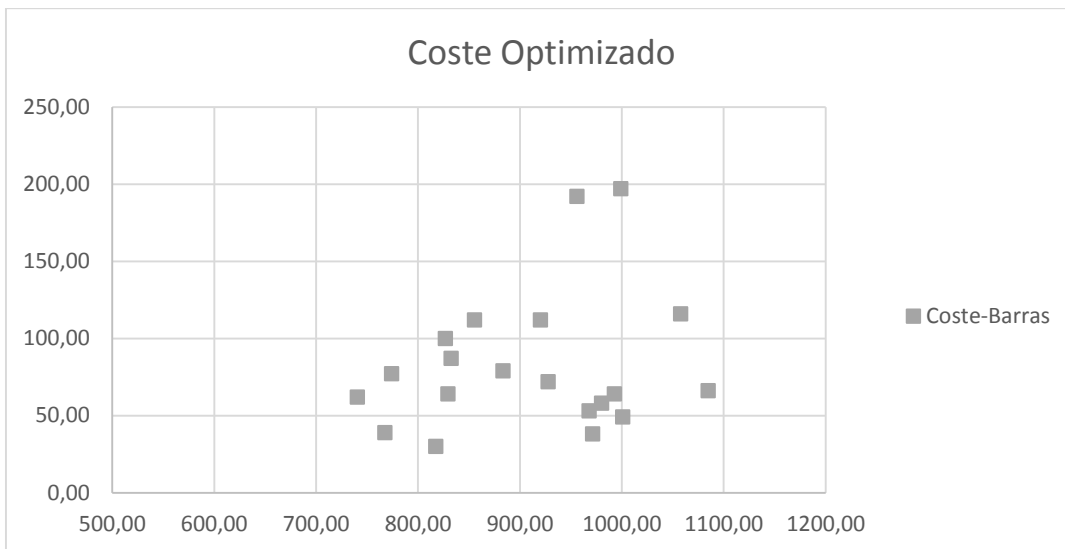


Gráfico 5 Relación del coste con número de barras

En el gráfico X se muestran las relaciones Coste-CO<sub>2</sub> y Coste-Consumo energético los resultados obtenidos difieren de la relación inversamente proporcional entre los parámetros que deberían tener los valores de la frontera de Pareto, aunque están en correspondencia con los resultados que se muestran en (Víctor Yepes et al., 2015). Contrariamente, al disminuir el coste se disminuye tanto la emisión de CO<sub>2</sub> como el consumo energético. Si todas las relaciones fueran idénticas no sería necesario realizar una optimización multiobjetivo ya que optimizando un único criterio tendríamos el resto. En cambio en el gráfico Y vemos que esta relación se rompe, de hecho no existe ninguna relación claramente visible. La distribución de los datos no sigue ninguna tendencia y existen unos claros valores mejores que otros, los situados en la zona inferior izquierda. Entre estos dos parámetros tiene sentido establecer una Frontera de Pareto y seleccionar el óptimo de entre ellos, ya que una variable si que puede degenerar a la segunda.

Cabría esperar que las mismas relaciones se repitieran independientemente del criterio de optimización utilizado, es decir que se mantuviera una relación directamente proporcional entre el coste, la emisión de CO<sub>2</sub> y el consumo energético y que no existiera ninguna relación de estos criterios con el número de barras mínimo empleado.

Para verificar esto, antes de seguir, vamos a realizar estudio análogos utilizando como criterio de optimización cada uno de los otros 3 criterios. El siguiente criterio de optimización que se estudiará son las emisiones de CO<sub>2</sub>. Los resultados de este estudio se muestran en la Tabla X. Se ha comprobado, cómo en cada caso, que los diferentes resultados obtenidos constituyen distribuciones normales. También se han representado los resultados en los Gráficos X e Y. CO<sub>2</sub>

| <b>CO<sub>2</sub></b> |         |                 |         |        |
|-----------------------|---------|-----------------|---------|--------|
| <b>N</b>              | Coste   | CO <sub>2</sub> | Energía | Barras |
| <b>1</b>              | 710,10  | 1318,30         | 4280,00 | 70,00  |
| <b>2</b>              | 830,83  | 1599,69         | 5201,42 | 67,00  |
| <b>3</b>              | 915,92  | 1800,36         | 5900,06 | 86,00  |
| <b>4</b>              | 1006,29 | 2001,59         | 6564,66 | 219,00 |
| <b>5</b>              | 983,50  | 1949,06         | 6134,08 | 123,00 |
| <b>6</b>              | 921,70  | 1802,20         | 5869,71 | 95,00  |
| <b>7</b>              | 1002,20 | 1987,07         | 6496,33 | 96,00  |
| <b>8</b>              | 878,58  | 1679,67         | 5397,09 | 115,00 |
| <b>9</b>              | 908,46  | 1779,45         | 5812,53 | 72,00  |
| <b>10</b>             | 735,02  | 1386,47         | 4515,37 | 93,00  |
| <b>11</b>             | 815,41  | 1529,11         | 4879,91 | 52,00  |
| <b>12</b>             | 1027,96 | 2019,56         | 6551,46 | 61,00  |
| <b>13</b>             | 938,01  | 1829,14         | 5935,94 | 45,00  |
| <b>14</b>             | 846,78  | 1602,06         | 5210,34 | 92,00  |
| <b>15</b>             | 1218,92 | 2443,67         | 7970,18 | 58,00  |
| <b>16</b>             | 821,68  | 1573,48         | 5103,97 | 61,00  |
| <b>17</b>             | 969,42  | 1901,39         | 6182,18 | 120,00 |
| <b>18</b>             | 937,43  | 1839,22         | 5996,79 | 100,00 |
| <b>19</b>             | 1127,37 | 2254,51         | 7364,06 | 46,00  |
| <b>20</b>             | 1055,89 | 2073,04         | 6713,84 | 181,00 |

*Tabla 6 Valores de la optimización en base a la emisión de CO<sub>2</sub>*

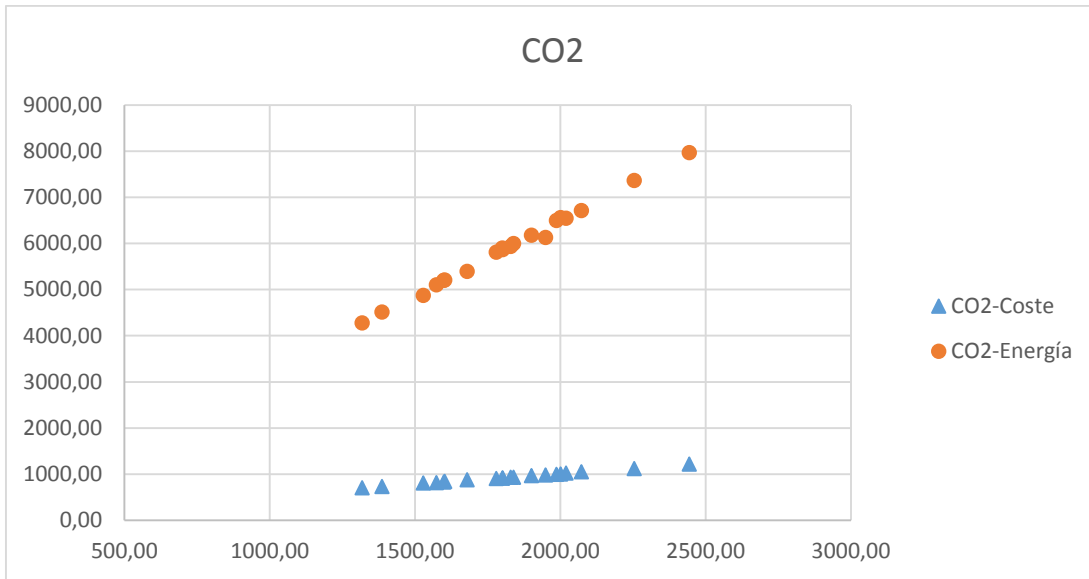


Gráfico 6 Relación de la emisión de CO2 con el coste y el consumo energético

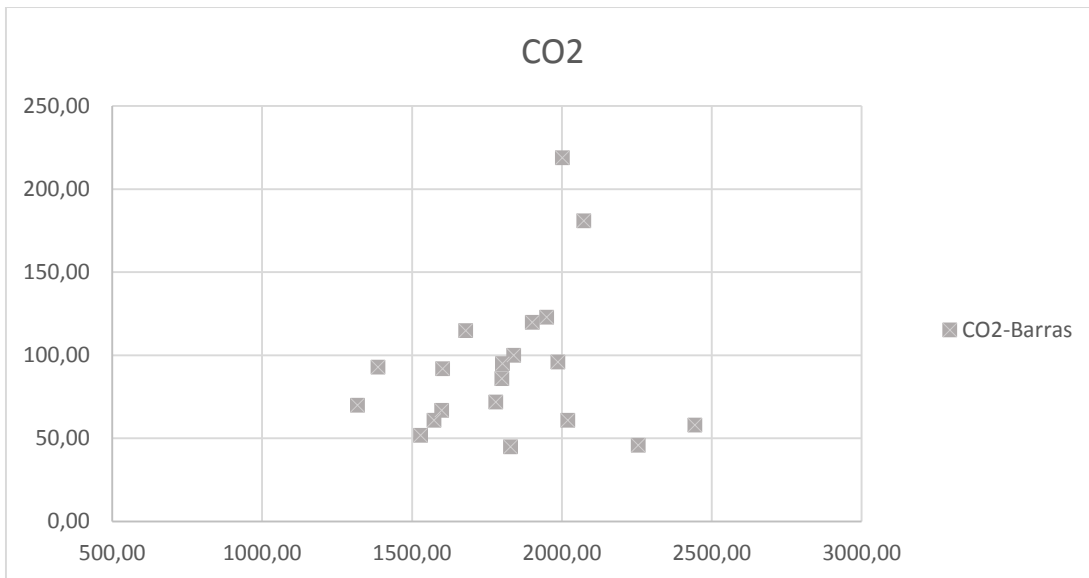


Gráfico 7 Relación de la emisión de CO2 con el número de barras

Las mismas relaciones se repiten en este caso, por lo que cabría esperar que en los casos restantes se comportase de manera similar. A pesar de esto, y a la vista de los resultados, si se tuviera que elegir sólo uno de los tres criterios con una relación proporcional entre ellos, sería preferible utilizar el criterio del coste, ya que se consiguen, de media, unos mejores resultados.

Una vez verificados los dos anteriores no se espera ningún cambio en las tendencias de las relaciones entre los diferentes criterios de optimización, por ello se van a exponer las relaciones extraídas tras el estudio de los dos criterios restantes simultáneamente. Primero se muestran los resultados y los gráficos referentes al consumo energético como criterio único y tras esto se muestran los obtenidos al utilizar el criterio de mínimo número de barras.



**ENERGÍA**

| N  | Coste   | CO <sub>2</sub> | Energía | Barras |
|----|---------|-----------------|---------|--------|
| 1  | 859,34  | 1635,89         | 5249,59 | 168,00 |
| 2  | 1035,21 | 2022,56         | 6539,75 | 56,00  |
| 3  | 1052,53 | 2090,65         | 6822,63 | 80,00  |
| 4  | 930,22  | 1793,97         | 5773,52 | 32,00  |
| 5  | 702,50  | 1271,28         | 4093,25 | 106,00 |
| 6  | 1010,78 | 1962,22         | 6301,16 | 292,00 |
| 7  | 803,30  | 1539,61         | 5014,84 | 119,00 |
| 8  | 1012,64 | 1961,16         | 6282,43 | 124,00 |
| 9  | 719,18  | 1349,77         | 4435,27 | 67,00  |
| 10 | 1418,22 | 2845,87         | 9267,99 | 118,00 |
| 11 | 841,59  | 1597,02         | 5122,51 | 84,00  |
| 12 | 609,37  | 1074,14         | 3486,99 | 77,00  |
| 13 | 855,34  | 1640,59         | 5299,92 | 67,00  |
| 14 | 810,35  | 1535,51         | 4971,31 | 56,00  |
| 15 | 851,74  | 1642,99         | 5338,18 | 62,00  |
| 16 | 1085,28 | 2155,66         | 7023,46 | 37,00  |
| 17 | 896,25  | 1757,94         | 5767,11 | 43,00  |
| 18 | 1086,56 | 2138,79         | 6931,44 | 74,00  |
| 19 | 757,53  | 1438,79         | 4699,13 | 24,00  |
| 20 | 948,10  | 1844,38         | 5967,94 | 112,00 |

Tabla 7 Valores de la optimización en base al consumo energético

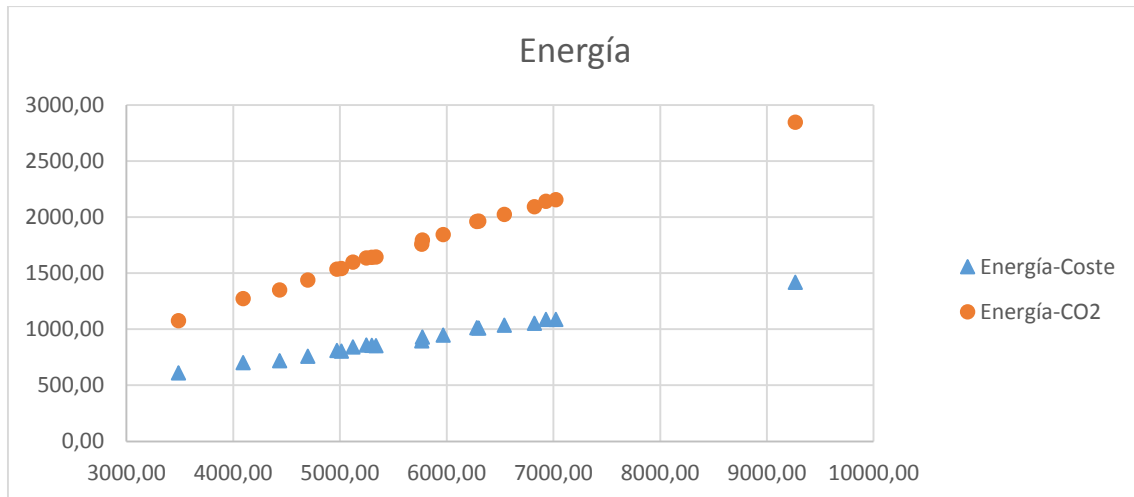


Gráfico 8 Relación del consumo energético con el coste y la emisión de CO2

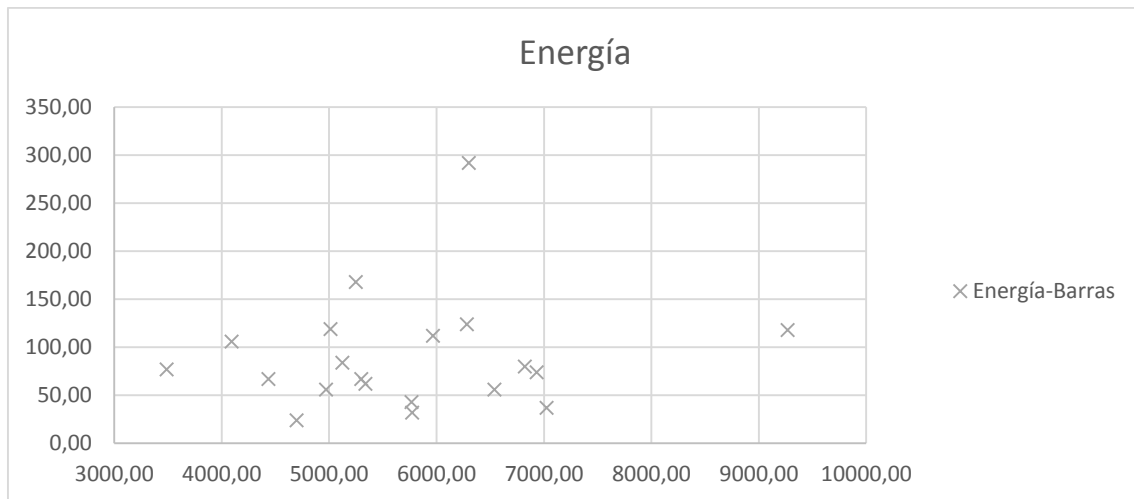


Gráfico 9 Relación del consumo energético con el número de barras

| <b>BARRAS</b> |              |                       |                |               |
|---------------|--------------|-----------------------|----------------|---------------|
| <b>N</b>      | <b>Coste</b> | <b>CO<sub>2</sub></b> | <b>Energía</b> | <b>Barras</b> |
| 1             | 12725,12     | 27598,66              | 91657,15       | 49,00         |
| 2             | 9289,55      | 21312,83              | 71341,00       | 72,00         |
| 3             | 6815,58      | 15498,55              | 50125,27       | 82,00         |
| 4             | 5218,25      | 11250,34              | 36677,81       | 46,00         |
| 5             | 21017,33     | 48533,23              | 164195,88      | 107,00        |
| 6             | 10610,36     | 23594,66              | 76856,35       | 64,00         |
| 7             | 5742,33      | 13164,89              | 40769,00       | 41,00         |
| 8             | 9414,95      | 21763,95              | 70951,96       | 64,00         |
| 9             | 7512,86      | 17255,55              | 53060,48       | 51,00         |
| 10            | 10998,63     | 25695,18              | 75975,22       | 43,00         |
| 11            | 6602,35      | 15005,42              | 47986,94       | 45,00         |
| 12            | 9681,39      | 21681,58              | 70533,27       | 69,00         |
| 13            | 2525,24      | 5456,93               | 17934,90       | 42,00         |
| 14            | 17837,24     | 41220,93              | 140653,33      | 124,00        |
| 15            | 8027,72      | 18413,33              | 61766,31       | 97,00         |
| 16            | 20804,15     | 46382,40              | 156680,80      | 98,00         |
| 17            | 5509,20      | 12459,87              | 41903,43       | 80,00         |
| 18            | 7907,97      | 17983,51              | 60291,39       | 64,00         |
| 19            | 14384,25     | 32638,73              | 111013,93      | 115,00        |
| 20            | 9610,20      | 21710,62              | 68571,49       | 50,00         |

Tabla 8 Valores de la optimización en base al número de barras

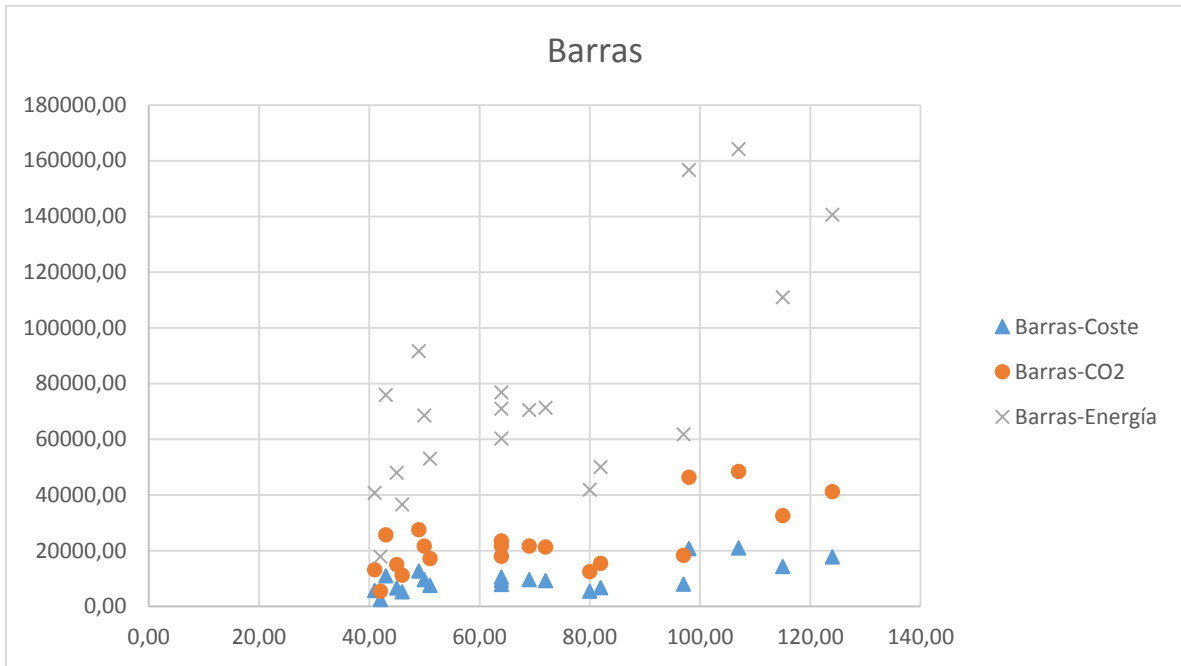


Gráfico 10 Relación del número de barras con el resto de criterios

Como se esperaba, no ha aparecido ninguna nueva tendencia entre los criterios de optimización. Esto significa que existen tres criterios con una relación directamente proporcional entre ellos, el coste, la emisión de CO<sub>2</sub> y el consumo energético, además de un cuarto criterio que no depende de los anteriores. Por lo tanto, a la hora de realizar la optimización multicriterio únicamente tendremos que optimizar comparando dos de los criterios, lo que simplifica altamente el problema.

Se puede deducir que uno de los criterios que son obligatorios seleccionar en la optimización multicriterio, es el número de barras, ya que funciona de un modo totalmente independiente al resto. Pero la elección del segundo no es tan sencilla, se ha de seleccionar el criterio que mejor resultado garantice. El mejor criterio será aquel que nos proporcione una media menor para cada uno de los criterios de optimización, la Tabla X muestra los valores medios de cada criterio a partir de las simulaciones anteriores.

|                 | Criterio seleccionado |                 |         |          |
|-----------------|-----------------------|-----------------|---------|----------|
|                 | Coste                 | CO <sub>2</sub> | Energía | Barras   |
| Coste           | 909,28                | 932,57          | 914,30  | 10111,73 |
| CO <sub>2</sub> | 1769,31               | 1818,45         | 1764,94 | 22931,06 |
| Energía         | 5758,48               | 5904,00         | 5719,42 | 75447,30 |
| Barras          | 83,35                 | 92,60           | 89,90   | 70,15    |

Tabla 9 Valores medios de las anteriores optimizaciones

Observando las medias aritméticas se verifica por un lado que en todos los casos, exceptuando la emisión de CO<sub>2</sub>, la menor media de cada criterio se produce cuando se optimiza en base a ese

criterio, como se espera. Obviando este hecho, en general el menor valor al comparar todos los criterios se produce al seleccionar el criterio del coste, este nos da el mejor resultado en cuanto a coste y número de barras y el segundo mejor resultado, con poca diferencia, con el resto de criterios. Ningún otro criterio de los cuatro estudiados tiene a un óptimo mejor. Por ello la optimización multicriterio se realizará utilizando el coste y el número de barras como criterios de optimización.

## 2.4 VISIÓN GENERAL DE LA APLICACIÓN DE OPTIMIZACIÓN MULTICRITERIO

Una vez estudiadas estas relaciones hemos definido las existentes entre los criterios y simplificado la idea de aplicar la frontera de Pareto. Para que la optimización sea multicriterio, vamos a utilizar la optimización monocriterio creada anteriormente como punto base para añadir modificaciones que nos permitan establecer la correcta frontera y elegir el sujeto adecuado..

Cuando se implemente la frontera de Pareto al algoritmo genético, esta influirá en la elección de progenitores a la hora de crear un sucesor porque no se descartan automáticamente los sujetos no pertenecientes a esta frontera con el fin de no disminuir la diversidad en la población, ya que puede que exista una combinación genética que no pertenezca a la frontera, pero su combinación con otro gen suponga una mejora de la misma. Aunque esta combinación difícilmente se producirá, ya que se va a utilizar el parámetro `fitness` para favorecer la elección de progenitores que pertenezcan a la frontera (este parámetro está definido en el apartado 2.1.1.2). Esta técnica se describe en (Deb et al., 2002) cómo una de las técnicas para crear un algoritmo genético multicriterio.

Para saber si un sujeto pertenece o no a la frontera de Pareto se realizará una comprobación tras ordenar la matriz `población` en función de uno de los criterios. Se compara el valor del segundo criterio de ese sujeto con el inmediatamente anterior, si el valor es inferior el sujeto pertenece a la frontera de Pareto. Inicialmente todos los sujetos se consideran en la frontera de Pareto.

### 2.4.1.1 Funcionamiento de la aplicación

Para implementar esta funcionalidad es necesario llevar a cabo unos cambios en el código que afectan a la mayoría de las funciones. A continuación describiremos esos cambios, diremos dónde se encuentran y por qué. Vamos a empezar explicando los cambios introducidos en la función principal del algoritmo.

A partir de este momento se utiliza un nuevo parámetro, denominado `fitness`, que como el resto de parámetros ha sido añadido a la familia `ConcreteBeamOpt` a través del fichero `Shared Parameters`. Por este motivo es necesario captar el valor de este parámetro mediante la API como se ha hecho con el resto. El parámetro `fitness` fuerza al algoritmo a elegir progenitores que formen parte de la Frontera de Pareto al ser seleccionados para crear un nuevo espécimen. Además de iniciar este parámetro tenemos que crear un identificador que señale los especímenes pertenecientes a la Frontera de Pareto para que el algoritmo sea capaz de diferenciarlos.

La selección de los sujetos más aptos continúa funcionando con el criterio elitista. Cuando se seleccionen los especímenes más aptos a través de este criterio sólo se ordenan en base a uno de los criterios, en este caso se ha seleccionado el coste, ya que se obtienen mejores resultados. Una vez se han ordenado los parámetros en función de este criterio se comprueba, asumiendo que el primer sujeto pertenece a la frontera de Pareto, si los sujetos sucesivos mejoran la solución del segundo criterio, los que cumplan esta condición se marcan como pertenecientes a la frontera. Este tipo de selección se define en Deb et al. (2002).

La frontera de Pareto no sólo tiene influencia a la hora de realizar la combinación genética y la selección de los progenitores. El óptimo tiene que provenir de ella, pues contiene las mejores soluciones de compromiso entre los diferentes criterios de optimización.

Se ha anunciado con anterioridad que cualquiera de las soluciones pertenecientes a esta frontera sería una buena solución, pero en la aplicación se ha implementado un criterio concreto: el algoritmo selecciona como óptimo el sujeto más cercano al *Pareto Knee*.

La región conocida como *Pareto Knee* es la región donde se minimiza la distancia de la frontera al origen, tal y como se muestra en la Figura 23. La frontera de Pareto no tiene por qué asemejarse a una curva, como hemos ido mostrando en las imágenes, de hecho, rara vez esto se produce, en nuestro caso tiene una forma escalonada y no existe ninguna variable que sea claramente más influyente que otra. El gráfico 2 muestra la representación de puntos de la muestra de Coste-Barras que se ha estudiado en el apartado anterior con la frontera de Pareto situada. Se observa que, en este conjunto de simulaciones la región Knee está situada alrededor de una de las soluciones en concreto.

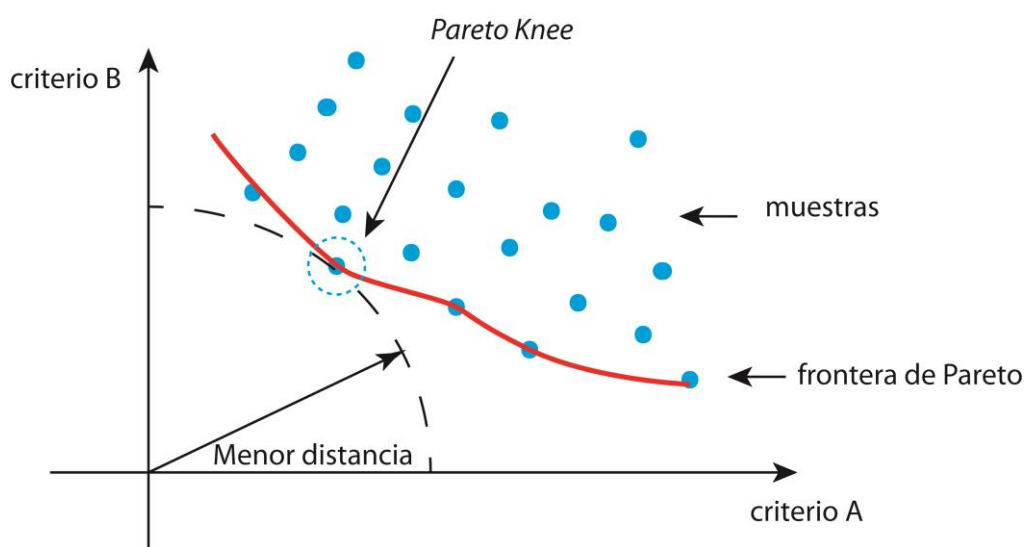


Fig. 23 Pareto Knee y óptimo seleccionado

Anteriormente hemos establecido que tres de los criterios estaban directamente relacionados, por lo que nuestra aplicación multicriterio ha quedado simplificada a la única comparación entre dos criterios. Con estos condicionantes es sencillo establecer el criterio para seleccionar el óptimo de Pareto. Nuestra aplicación buscará al sujeto que menor distancia Euclídea hasta el punto de origen. Esto se realiza comprobando paulatinamente las distancias: cuando se encuentra el primer espécimen con una distancia superior al anterior se selecciona el sujeto inmediatamente anterior y este es extraído como el “óptimo solución”.

Introduciendo estas funcionalidades adicionales, el *plugin* pasa a estar plenamente operativo para realizar una optimización multicriterio. Para conseguir realizar esta aplicación y estudiar el comportamiento de cada uno de los parámetros, se ha utilizado una aplicación monocriterio creada para esta finalidad. Además, durante el desarrollo se han expuesto los puntos en los cuales se han tenido que utilizar las características de la API de Revit y que difieren de un entorno Bim a otro. En este apartado se ha evitado la presencia del código de computación para facilitar la lectura, una copia completa e íntegra de los códigos utilizados puede ser consultada en los Anexos.

### 3. ESTUDIO PARAMÉTRICO

---

Una vez creada la aplicación de optimización multicriterio se ha cumplido el objetivo de conectar el software de optimización con un entorno BIM. Este hecho nos abre un mundo de posibilidades a la hora de aplicar técnicas de *Lean Construction* y de mejorar en general el proyecto como anunciábamos en el apartado 1.

En este apartado se van a utilizar diferentes técnicas estadísticas que por un lado demuestren el correcto funcionamiento del algoritmo, pese a funcionar a través de un software externo, y por otro, nos ayudarán a establecer una serie de conclusiones y reglas inherentes a la tipología estructural estudiada: la viga isostática.

Se van a realizar un total de tres estudios diferentes utilizando, siempre que se pueda, un mismo problema como base. Estos análisis están realizados en orden de mayor a menor complejidad y se han ejecutado con la finalidad de demostrar paso a paso el correcto funcionamiento de la aplicación. En todo caso las diferentes muestras optimizadas han sido ejecutadas a través del entorno BIM siguiendo los pasos que se detallan en la Guía de Usuario adjunta en los anejos a este mismo documento.

El primer estudio que se va a realizar es un estudio en que se relacionan los diferentes parámetros de optimización establecidos y se comparan los resultados con aquellos obtenidos previamente al realizar la optimización monocriterio. De este modo se cuantifica qué mejoría nos supone el uso de un algoritmo multicriterio, con mayor complejidad, frente a uno monocriterio. Como conclusión de este estudio, se obtendrán unas reglas aproximadas que nos permitan evaluar de un modo intuitivo el valor de los diferentes criterios de optimización a partir de uno de ellos.

El segundo estudio se centrará en la influencia de la luz en los diferentes parámetros de definición del elemento y del valor de los criterios. Con ello se pretende analizar la eficiencia estructural del elemento estudiado y establecer unas reglas generales de diseño a partir de los datos obtenidos. Servirá también para establecer conclusiones sobre el comportamiento de la aplicación.

Por último, el tercer estudio analizará el comportamiento de la optimización cuando varían los precios de cada uno de los materiales individualmente. De este modo se conoce cuál es el material que más influye sobre los diferentes costes del elemento y se verifica cómo de bueno será el resultado que proporciona el algoritmo en un futuro cercano teniendo en cuenta las



fluctuaciones de precio en el mercado, ya que los precios de los materiales no pueden ser editados por el usuario.

Cada uno de los estudios precisa de una metodología diferente basada en las características propias de cada análisis y las conclusiones buscadas. En cada uno de los apartados se describe en qué consiste esta metodología y su modo de aplicación, así como los diferentes motivos por los que se llega a dichas conclusiones.

### 3.1 DEFINICIÓN DEL CASO DE ESTUDIO

A continuación, se van a definir los diferentes valores que tomarán los parámetros de entrada al algoritmo para la realización de los diferentes estudios paramétricos llevados a cabo en este apartado. Se mantienen constantes los valores de los parámetros del algoritmo genético, mientras que se definen diferentes casos de luces y estados de carga. Una de las combinaciones de estado de carga y luz, la utilizada por defecto, se define como caso estándar. En todos los casos estamos utilizando la aplicación desde el entorno BIM directamente, como si se tratara de un uso real.

Los valores de los parámetros genéticos han sido establecidos mediante un estudio estadístico aplicando la técnica del diseño de experimentos, cuyos valores están contenidos en la Tabla 10. En caso de utilización del *plugin* en un caso real, estos parámetros deberían ser ajustados para el caso particular utilizando la misma técnica. Un ejemplo de la aplicación del diseño de experimentos para ajustar los parámetros utilizados se muestra en el Anexo 3. Guía de Usuario. El ejemplo utilizado en este anejo corresponde con el denominado caso estándar.

| Parámetros         |     |
|--------------------|-----|
| <b>max_espe.</b>   | 250 |
| <b>max_descen.</b> | 250 |
| <b>gen.</b>        | 100 |
| <b>mut.</b>        | 5   |
| <b>prod.</b>       | 2   |
| <b>Fitness</b>     | 75  |

Tabla 10 Valores de los parámetros genéticos

Estos valores son más elevados de lo estrictamente necesario para obtener el espécimen óptimo y el resultado que se obtiene no sufre mejoras a lo largo de varias de las últimas generaciones. Generalmente, este resultado no sería deseable, ya que se aumenta el tiempo de computación sin obtener una mejora, pero en este caso los tiempos de computación son ínfimos, dada la sencillez del problema, de modo que se han despreciado. Además, como se van a realizar modificaciones en las demandas a la viga y estas afectarán a la dificultad de encontrar el óptimo, se busca evitar que los resultados obtenidos se vayan alejando periódicamente de la mejor

solución posible. Es por todo lo anterior que hemos escogido unos valores elevados para los parámetros que controlan el algoritmo genético.

La aplicación de optimización basada en un algoritmo genético multicriterio está pensada para ser utilizada en problemas de edificación. Es decir, en estructuras de hormigón de luces relativamente cortas y ambientes sin una gran agresión a la durabilidad. Los diferentes valores se han seleccionado siguiendo estos criterios. La Tabla 11 muestra los valores que toman las cargas y las luces, valores que corresponden con el problema estándar objeto de estudio de varios de los análisis como se ha señalado previamente. Se utilizan por ser los valores más habituales en esta tipología estructural, luces cortas con cargas medias y ambiente considerado IIb, lo que determina un recubrimiento de treinta y cinco milímetros siguiendo la EHE-08.

| Parámetros      |      |
|-----------------|------|
| <b>L (mm)</b>   | 5000 |
| <b>Rec (mm)</b> | 35   |
| <b>G (KN/m)</b> | 6    |
| <b>Q (KN/m)</b> | 4    |

Tabla 11 Características del problema estándar

Todas las pruebas estadísticas se han realizado con el mismo ordenador para garantizar la homogeneidad en los tiempos de computación, aunque como se ha explicado anteriormente estos no se han tenido en cuenta. Para este fin se ha utilizado un Acer Aspire 5750G con un procesador Intel(R) Core (TM) i5-2410M con 2,30 Ghz de procesamiento, el equipo tiene 4 Gb de memoria RAM y el sistema operativo utilizado es Windows 7 Home Premium. El entorno BIM utilizado en todos los casos es Autodesk Revit 2016 y el *plugin* ha sido compilado mediante Microsoft Visual Studio 2012, ya que las versiones más recientes de este software ocasionaban problemas de compatibilidad e inestabilidades entre el software BIM y el sistema operativo.

## 3.2 RELACIONES ENTRE LOS DIFERENTES PARÁMETROS DE OPTIMIZACIÓN

Con este estudio se pretende establecer una serie de reglas estimadas capaces de relacionar los tres criterios de optimización que en el apartado 2.4.1 se ha demostrado que son proporcionales. Al realizar esto en primer lugar conseguimos dos cosas: demostrar el correcto funcionamiento del algoritmo de optimización y establecer una serie de normas sencillas que nos permitan, de manera intuitiva, conocer los valores de los diferentes criterios.

Estas reglas únicamente tienen sentido como una opción más veloz, que mantenga el orden de magnitud de los valores por parte del usuario sin la necesidad de consultar periódicamente los valores reales. A su vez, no se prevén para realizar cálculos que contabilicen los diferentes valores de los criterios, ya que los valores reales son volcados al *Building Model* y se puede acceder a ellos desde el mismo.

En la creación de estas reglas se van a utilizar una serie de simulaciones con parámetros idénticos a los del apartado 2.4.1 aunque empleando el algoritmo multicriterio, es decir una luz de cálculo de 5,5 m., un recubrimiento nominal de 35 mm. y unas cargas gravitatorias y variables de 6 y 4 KN/m respectivamente. A partir de los resultados, recogidos en la Tabla 12, se elabora un conjunto de rectas de regresión entre los tres criterios de optimización.

| <b>Multicriterio (€-Barras)</b> |              |                       |                |               |
|---------------------------------|--------------|-----------------------|----------------|---------------|
| <b>N</b>                        | <b>Coste</b> | <b>CO<sub>2</sub></b> | <b>Energía</b> | <b>Barras</b> |
| <b>1</b>                        | 942,08       | 1821,98               | 5854,57        | 23,00         |
| <b>2</b>                        | 793,88       | 1473,84               | 4656,10        | 34,00         |
| <b>3</b>                        | 736,35       | 1370,35               | 4441,09        | 82,00         |
| <b>4</b>                        | 731,39       | 1355,65               | 4373,87        | 43,00         |
| <b>5</b>                        | 749,39       | 1392,86               | 4486,56        | 55,00         |
| <b>6</b>                        | 687,54       | 1253,50               | 4005,42        | 44,00         |
| <b>7</b>                        | 800,63       | 1506,68               | 4817,16        | 83,00         |
| <b>8</b>                        | 622,36       | 1086,30               | 3476,60        | 89,00         |
| <b>9</b>                        | 723,71       | 1329,29               | 4217,81        | 47,00         |
| <b>10</b>                       | 1003,62      | 1929,88               | 6144,25        | 63,00         |
| <b>11</b>                       | 831,99       | 1556,46               | 4923,46        | 49,00         |
| <b>12</b>                       | 903,07       | 1737,12               | 5582,25        | 21,00         |
| <b>13</b>                       | 886,29       | 1709,44               | 5523,09        | 56,00         |
| <b>14</b>                       | 846,75       | 1603,17               | 5114,72        | 46,00         |
| <b>15</b>                       | 750,17       | 1399,17               | 4497,56        | 86,00         |
| <b>16</b>                       | 919,73       | 1764,33               | 5642,65        | 40,00         |
| <b>17</b>                       | 744,02       | 1381,55               | 4422,79        | 87,00         |
| <b>18</b>                       | 816,44       | 1531,92               | 4870,82        | 44,00         |
| <b>19</b>                       | 918,03       | 1782,28               | 5769,50        | 55,00         |
| <b>20</b>                       | 916,69       | 1769,12               | 5692,49        | 55,00         |
| <b>Media</b>                    | 816,21       | 1537,74               | 4925,64        | 55,10         |

Tabla 12 Valores de la optimización multicriterio

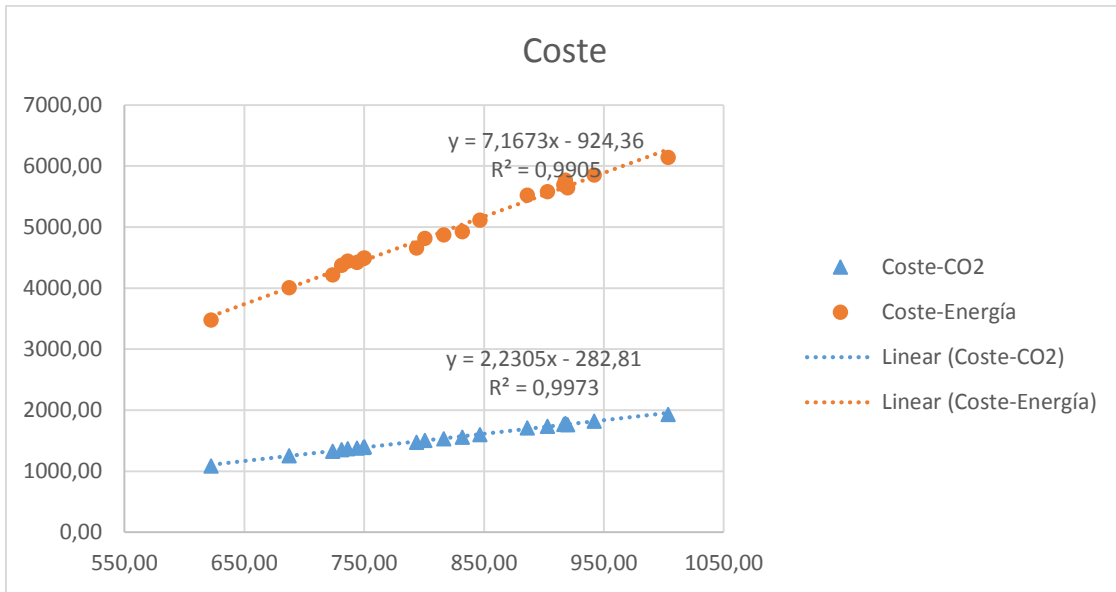


Gráfico 11 Relación del Coste con el CO<sub>2</sub> y el Consumo energético

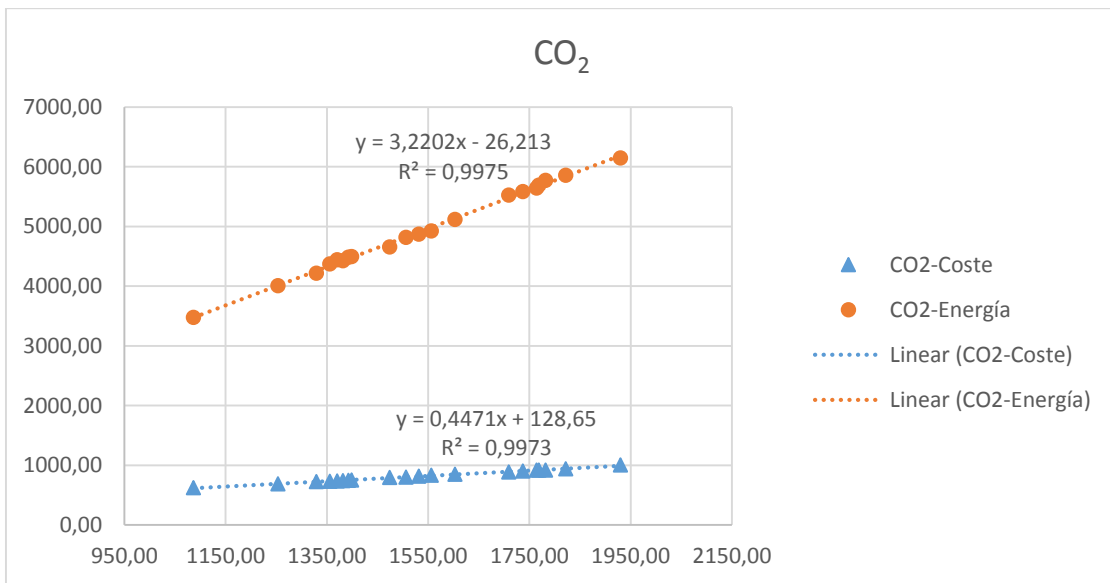


Gráfico 12 Relación del CO<sub>2</sub> con el Coste y el Consumo energético

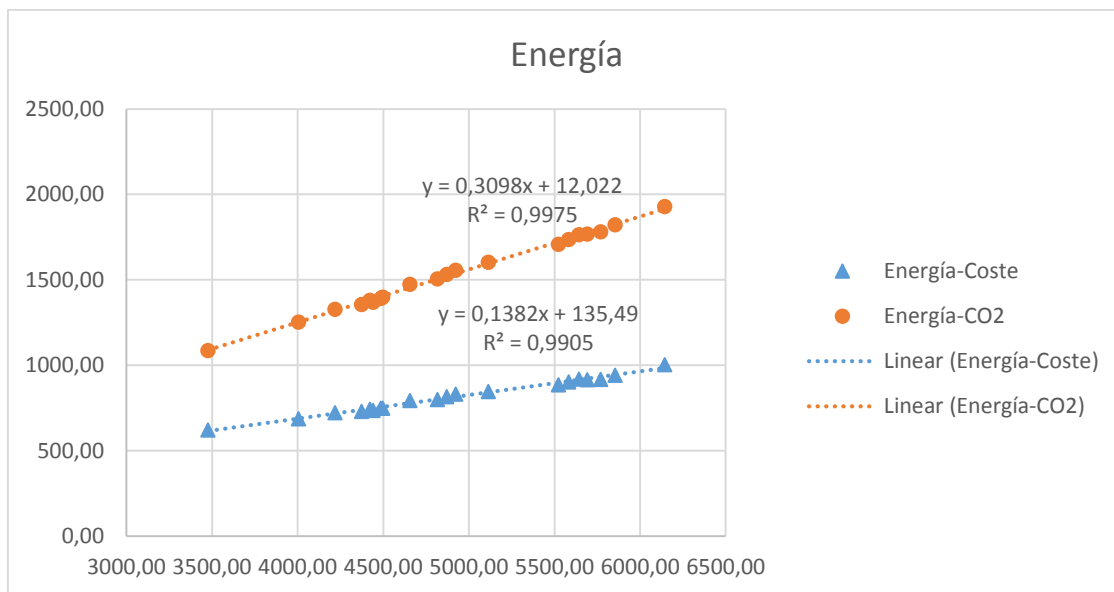


Gráfico 13 Relación del Consumo energético con el Coste y el CO2

Cada uno de los gráficos, 8, 9 y 10, toman como variable principal uno de los criterios de optimización y lo comparan con los otros dos. Se ha utilizado esta nueva muestra y no la extraída anteriormente, en el apartado 2.4.1, porque en esta se ha utilizado el algoritmo multicriterio. En los gráficos se pueden observar las diferentes ecuaciones de las rectas de regresión, establecidas mediante el método de los mínimos cuadrados, y los coeficientes de correlación de Pearson. Todos los coeficientes de correlación son superiores a 0,99, lo que las valida al ser capaces de predecir los resultados con un 99% de precisión.

La relación entre los tres criterios de optimización sigue la misma tendencia que la que demostraba el algoritmo de optimización monocriterio y la bibliografía consultada. También se han realizado pruebas de normalidad sobre las muestras y en todos los casos se han obtenido poblaciones normales. Estos dos hechos demuestran que el algoritmo multicriterio funciona adecuadamente. En el gráfico 11 se muestran los resultados de las medias de los diferentes conjuntos de simulaciones realizados hasta el momento, los cuatro correspondientes a apartados anteriores y el realizado en este mismo.

El algoritmo multicriterio mejora los resultados obtenidos, entre un 10,24% y un 21,45%, con respecto al algoritmo monocriterio utilizando cada uno de los criterios de optimización<sup>28</sup>. Por estos dos motivos podemos asumir que el algoritmo genético multiobjetivo funciona correctamente.

<sup>28</sup> Esta comparación se realiza en base a el valor medio de cada uno de los cuatro criterios del algoritmo multicriterio con el valor medio obtenido en el algoritmo monocriterio para cada criterio de optimización.

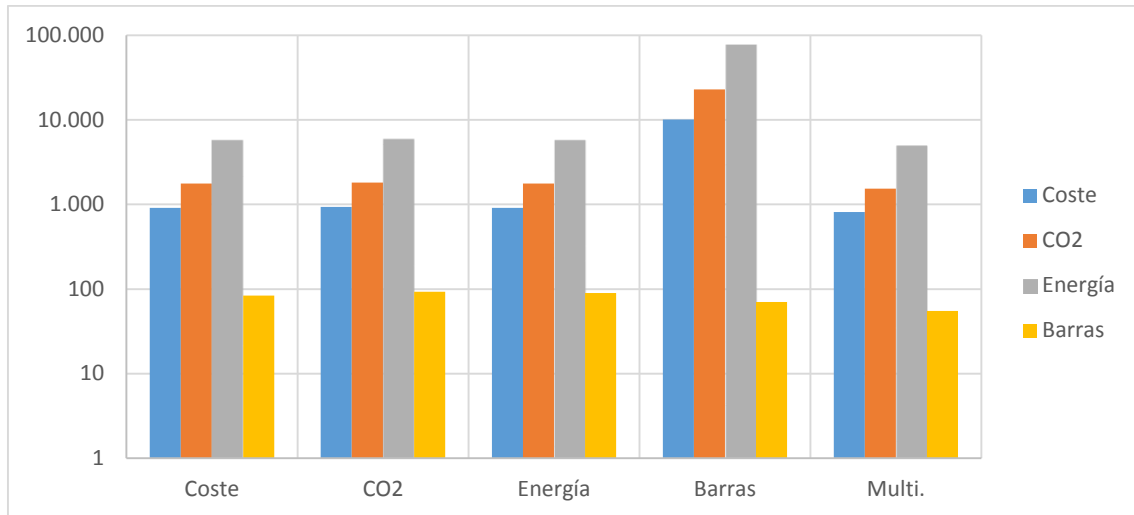


Gráfico 14 Comparación de las medias (escala logarítmica)

| Criterio seleccionado |        |         |        |
|-----------------------|--------|---------|--------|
| Coste                 | CO2    | Energía | Barras |
| 10,24%                | 15,44% | 13,88%  | 21,45% |

Tabla 13 Porcentaje de mejora del resultado según el criterio

Las rectas de regresión de los gráficos 8, 9 y 10 aunque corresponden con las relaciones más o menos exactas entre los parámetros, están lejos de ser fácilmente memorizables. La intención es encontrar unos parámetros fáciles de recordar que puedan indicar de un modo aproximado los valores de todos los criterios de optimización. Estos valores son los contenidos en la Tabla 14 y se consiguen aproximando las pendientes de las rectas de regresión a unos valores simplificados. El hecho de aproximar los parámetros disminuye su correlación, pero no estamos buscando un valor exacto sino un valor que proporciona un orden de magnitud. Estos parámetros se pueden utilizar para estimar, a falta de mejores herramientas, las emisiones de CO<sub>2</sub> y el consumo energético de otros elementos de hormigón pertenecientes al *Building Model* que no se hayan optimizado y así controlar los niveles de emisiones en el conjunto del proyecto desde edades tempranas.

|                 | Coste | CO <sub>2</sub> | Energía |
|-----------------|-------|-----------------|---------|
| Coste           | 1,00  | 0,45            | 0,15    |
| CO <sub>2</sub> | 2,20  | 1,00            | 0,30    |
| Energía         | 7,15  | 3,20            | 1,00    |

Tabla 14 Parámetros de relación entre los criterios de optimización

## 3.3 RESULTADOS DEL ESTUDIO PARAMÉTRICO

Esta sección compara los mejores diseños del elemento considerando todos los criterios de optimización teniendo en cuenta seis luces diferentes, cuya medida varía desde cinco hasta diez metros. Las luces seleccionadas corresponden a las más habituales en edificación en esta tipología estructural, es decir la mayor gama de estructuras objetivo a las que está dirigido este *plugin*.

Para cada uno de los casos estudiados se han realizado diez muestreos y se analizan los resultados en base a la media de los mismos. Analizando las características de diseño e identificando las diferencias se han encontrado una serie de normas de diseño que mejoran la sostenibilidad de las vigas de hormigón. Todas las conclusiones obtenidas en este estudio son únicamente fiables dentro del rango de luces estudiadas y fuera de este rango el comportamiento podría tener una gran diferencia, aunque se intuya que es similar.

Al realizar este estudio una de las cosas con mayor interés que podemos establecer son las relaciones entre los diferentes criterios de optimización tenidos en cuenta y la luz, gráficos 15, 16, 17 y 18. Los cuatro criterios aumentan en valor al aumentar la luz, aunque en distinta proporción, siendo el consumo energético el que presenta un mayor aumento. Mediante las funciones obtenidas se pueden saber, *a priori*, los diferentes valores de los criterios en la viga. Todas las funciones de regresión presentan un coeficiente de correlación entre noventa y siete y noventa y nueve, por lo que pueden calcularse con mucha exactitud estos valores mediante las expresiones.

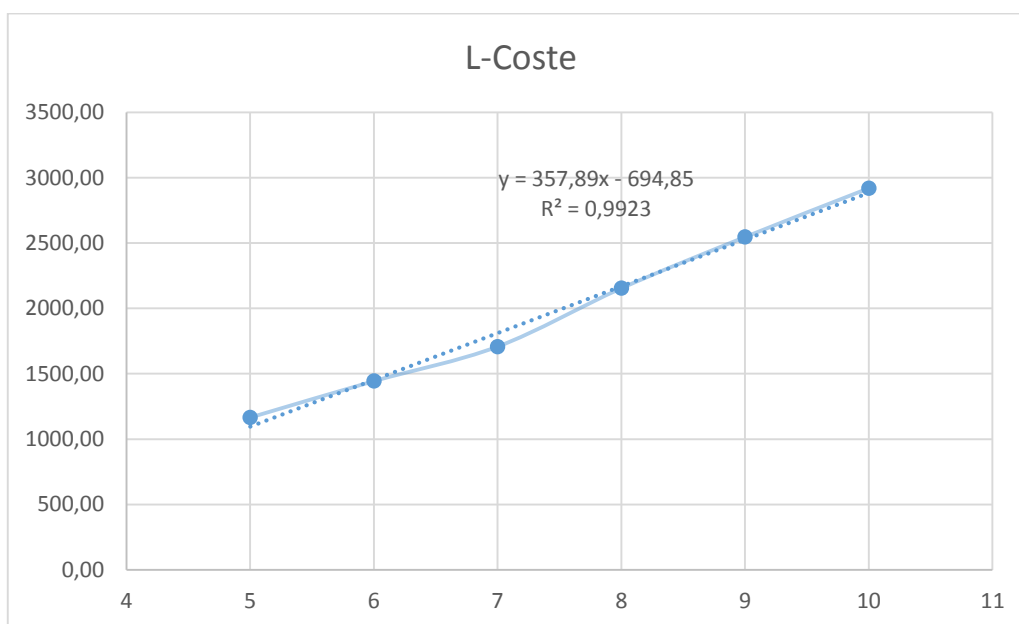


Gráfico 15 Relación Luz-Coste

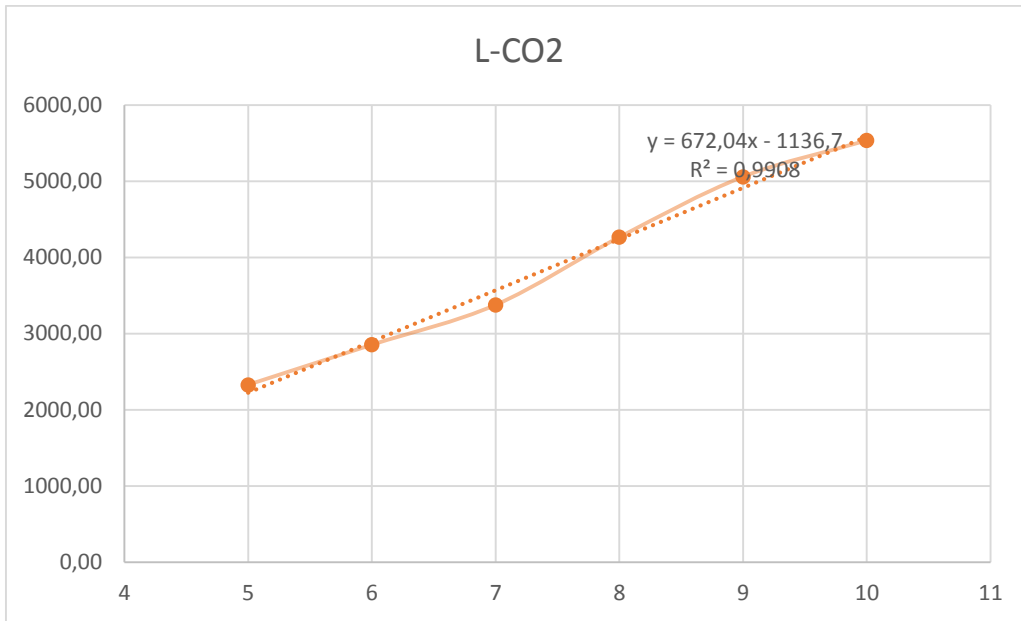


Gráfico 16 Relación Luz-CO<sub>2</sub>

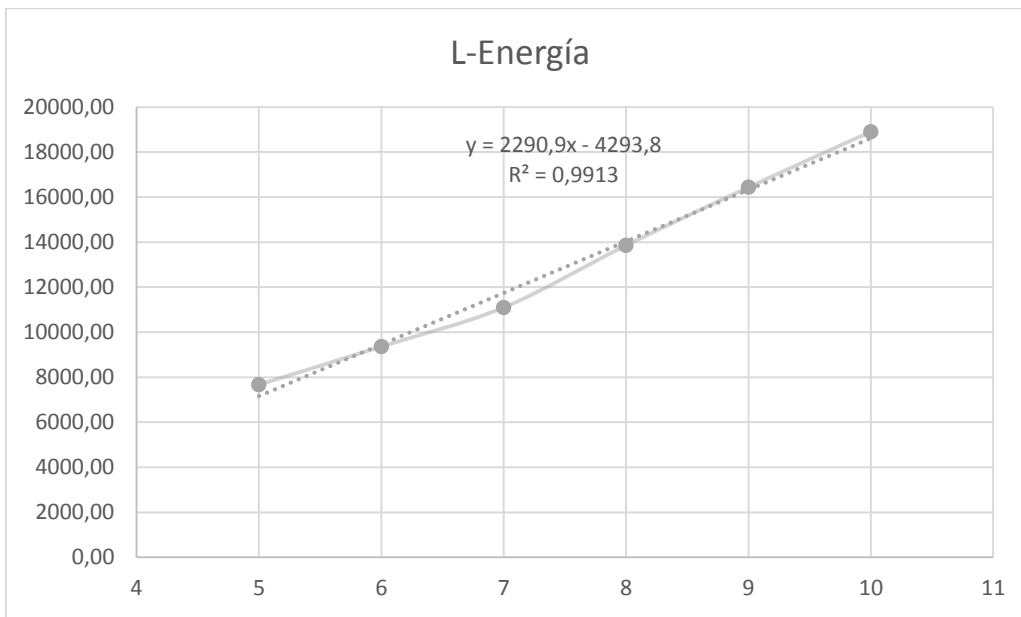
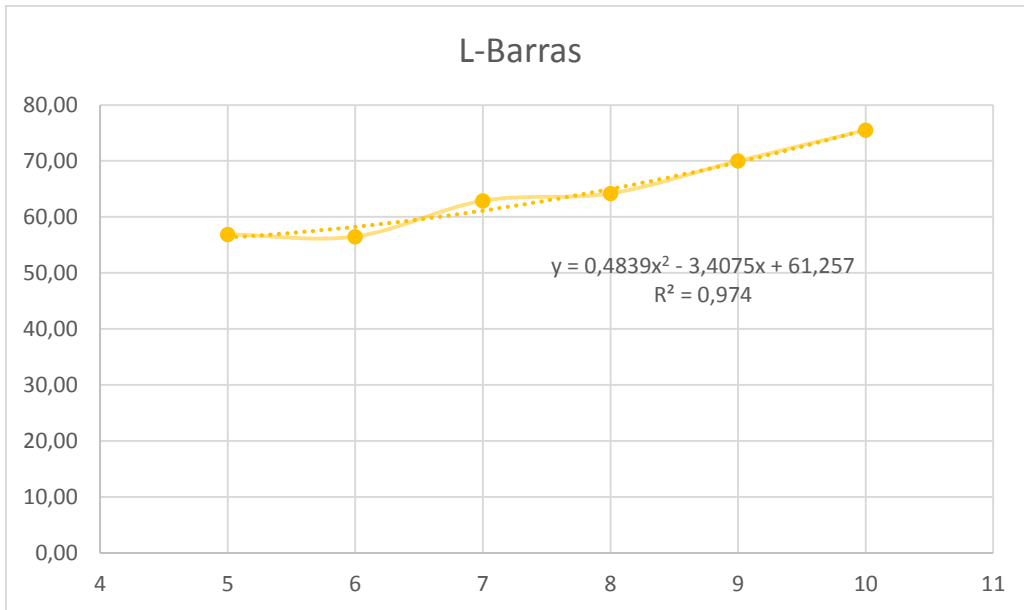


Gráfico 17 Relación Luz-Consumo Energético





*Gráfico 18 Relación Luz-Número de Barras*

Aunque se ha demostrado anteriormente que existe una relación prácticamente lineal entre los valores de los diferentes criterios utilizados, -coste, emisión de CO<sub>2</sub> y consumo energético- en el Gráfico 12, 13 y 14 se muestra que esta relación no es tan sencilla como pudiera parecer, sino que varía en función de la luz y al aumentar esta, aumenta la diferencia entre estos factores. Estos gráficos representan el incremento del valor de cada uno de los criterios al acrecentar en un centímetro la luz de cálculo y se observa que la pendiente de las rectas es diferente, por lo que cada uno aumenta con un incremento diferente y una relación cambiante entre los mismos. Esto implica que los elementos con una mayor luz tengan una mayor emisión de CO<sub>2</sub> y un mayor consumo energético en relación con su coste por lo que son menos sostenibles.

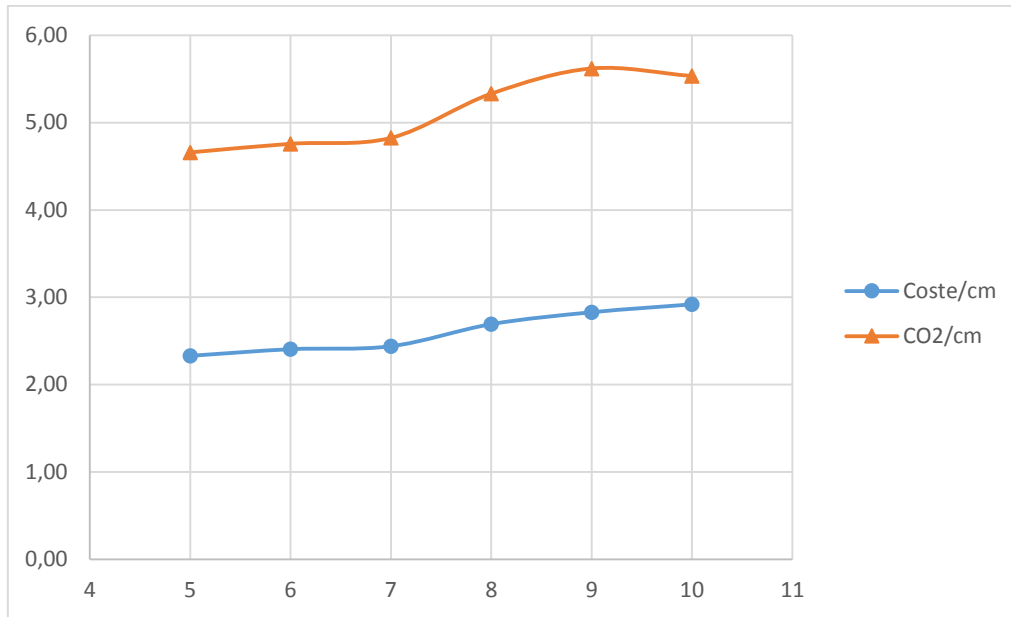


Gráfico 19 Aumento del Coste y de las emisiones de CO<sub>2</sub> con el aumento de Luz

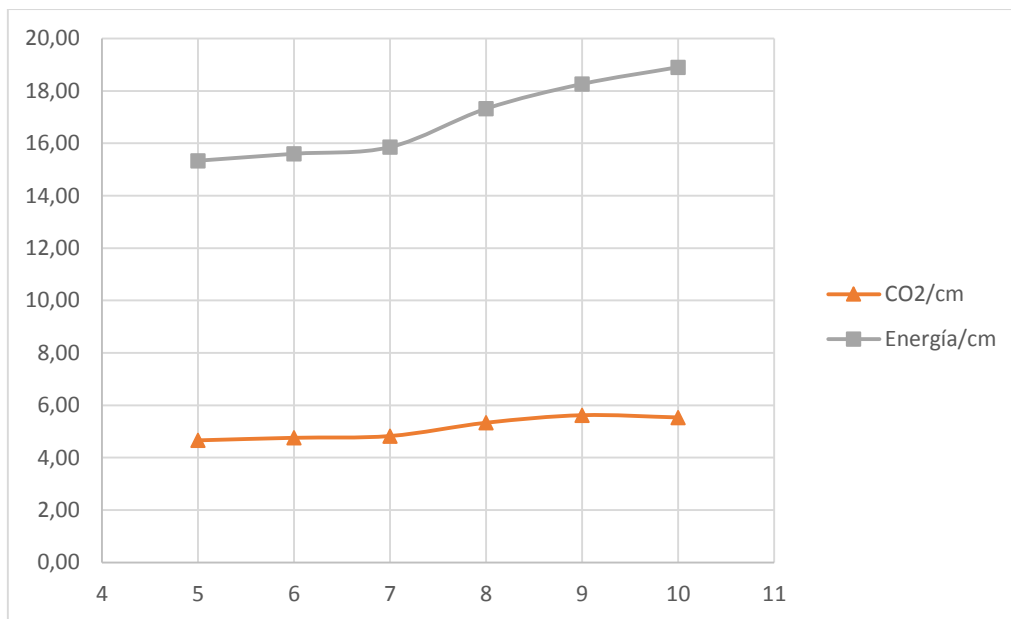


Gráfico 20 Aumento del CO<sub>2</sub> y del Consumo energético con el aumento de Luz

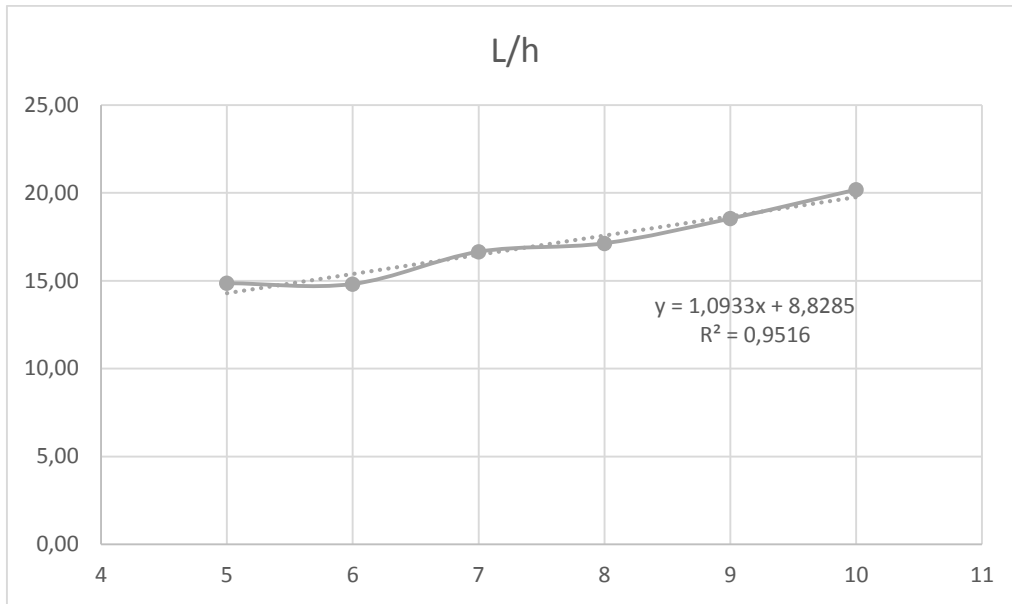


Gráfico 21 Relación Canto-Luz óptima

El gráfico anterior muestra la relación entre el canto del elemento y su luz. La razón h/L varía aproximadamente entre 1/15 y 1/20 de una luz menor a una mayor respectivamente. La recta de regresión del gráfico muestra que hay que incrementar esta relación en uno por cada metro extra de longitud, lo que significa que la relación disminuye conforme aumenta la luz y su capacidad estructural es utilizada de un modo más eficiente. Lógicamente, el canto siempre se agranda, pero esto es cada vez menor, como muestra esta variación en la relación y el gráfico 19. En este último gráfico también se ha representado la variación del ancho de la sección, que como se puede observar es prácticamente inexistente si lo comparamos, pues a nivel estructural es más eficiente aumentar el canto que el ancho.

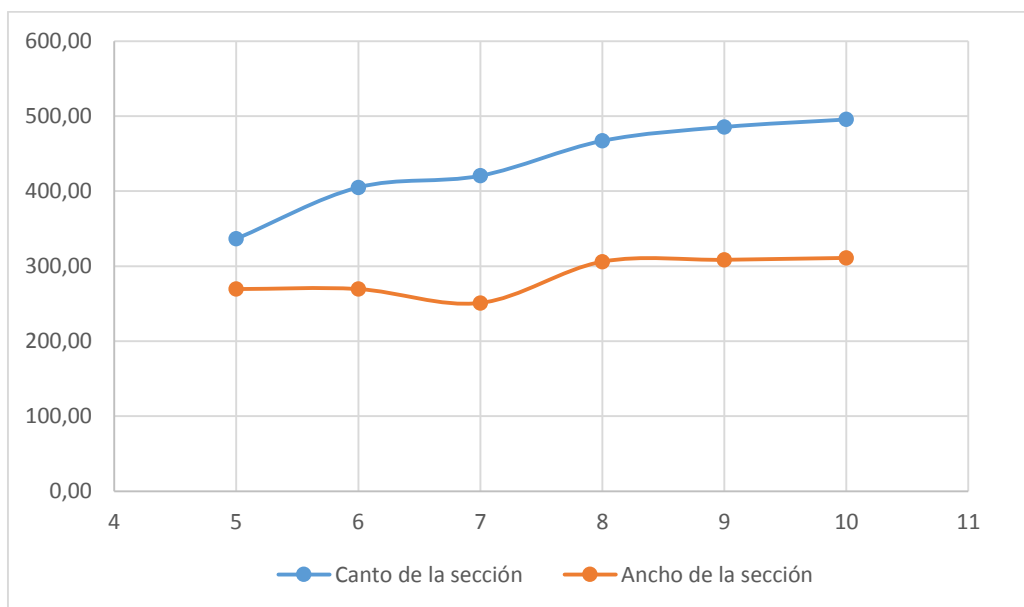


Gráfico 22 Variación del ancho y el canto con la Luz

En los diferentes datos mostrados hasta el momento se observa que existe un quiebro en las tendencias en el valor de luz de siete metros. En algunos casos, esta inflexión supone un hito en el comportamiento de esa tendencia, como en el incremento de coste por centímetro de longitud. La explicación a este salto radica en la eficiencia estructural: a partir de los siete metros de longitud esta tipología estructural sufre un aumento en la demanda de las prestaciones. Aunque a partir de esta luz no se aumenta en gran medida el número de barras de la armadura principal, sí se produce un aumento destacable en el diámetro que denota el salto en la demanda. Este cambio en las prestaciones se produce concretamente en esta longitud por las cargas actuantes, de modo que es previsible que un aumento en las cargas reduzca la luz necesaria. Además es a partir de este punto cuando la viga trabaja más eficientemente y se aprovechan mejor sus capacidades de resistencia.

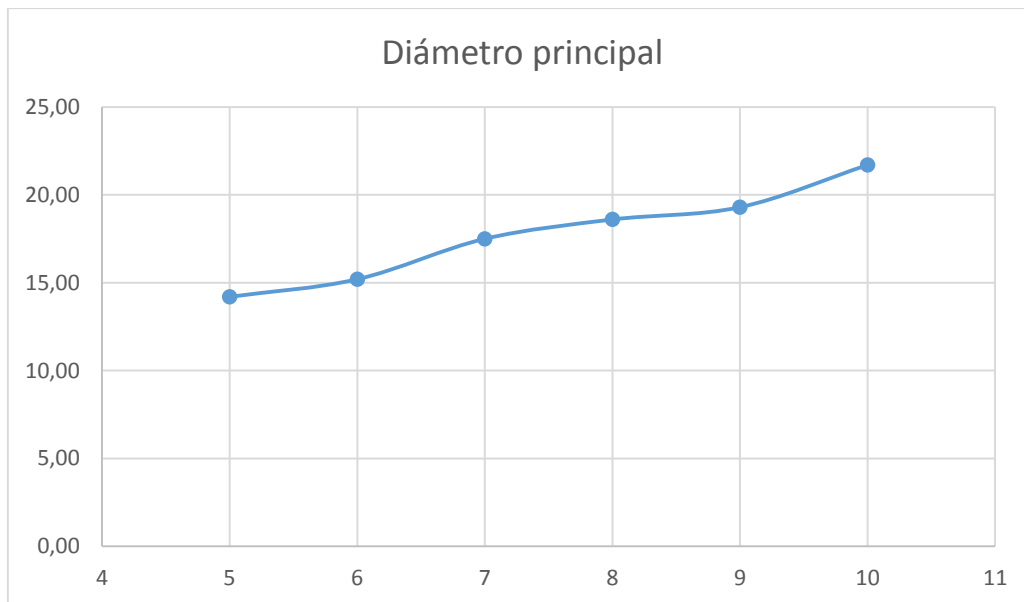


Gráfico 23 Variación del diámetro de la armadura principal con la Luz

En la siguiente tabla se exponen los resultados medios obtenidos tras la optimización de todos los casos ordenados según la luz de cálculo utilizada. Una de las primeras conclusiones que podemos extraer es que las vigas de canto siempre son más óptimas que las vigas planas de acuerdo a la tradición constructiva, ya que en todos los casos la media de los cantos es siempre superior a la de los anchos. Resulta destacable que el hormigón seleccionado sea siempre un hormigón HA-25, aunque la gama posible de hormigones sea hasta el HA-100 para las luces estudiadas, el aumento de resistencia no supone un aumento en las características resistentes de la estructura que compense el aumento de precio con la disminución de sección. Esta resistencia únicamente varía en casos aislados para la mayor luz y en estos casos se obtiene como resultado hormigones de alta resistencia, lo cual significa que los hormigones de altas

prestaciones suponen una ventaja en grandes luces. Según se incrementan las luces, el número de barras utilizado en la armadura principal no aumenta en grandes proporciones, aunque sí lo hace el diámetro utilizado como consecuencia de la intención de minimizar el número de barras empleado, que siempre favorecerá la utilización de diámetros mayores frente a otros más pequeños. Esta tendencia no se reproduce para la armadura secundaria que se mantiene aproximadamente constante y cercana a las cuantías mínimas exigidas normativamente, pues estas barras no tienen influencia en el comportamiento estructural de la viga. La relación entre el canto y la separación transversal ( $sep_3$ ) es en todos los casos estudiados del treinta por ciento, la separación máxima exigida por la normativa con un cortante elevado para la resistencia de la viga, lo que demuestra que el elemento estructural está sujeto a un elevado esfuerzo de corte.

|                       | 5       | 6       | 7        | 8        | 9        | 10       |
|-----------------------|---------|---------|----------|----------|----------|----------|
| <b>b</b>              | 269,50  | 269,50  | 251,00   | 306,00   | 308,50   | 311,00   |
| <b>h</b>              | 336,50  | 405,00  | 420,50   | 467,00   | 485,50   | 495,50   |
| <b>f<sub>ck</sub></b> | 25,00   | 25,00   | 25,00    | 25,00    | 25,00    | 28,00    |
| <b>D1</b>             | 14,20   | 15,20   | 17,50    | 18,60    | 19,30    | 21,70    |
| <b>D2</b>             | 8,60    | 9,00    | 8,20     | 9,00     | 8,80     | 9,20     |
| <b>D3</b>             | 8,00    | 8,00    | 8,00     | 8,00     | 8,00     | 8,00     |
| <b>n1</b>             | 3,10    | 3,10    | 3,30     | 3,20     | 3,40     | 3,50     |
| <b>n2</b>             | 2,00    | 2,30    | 2,10     | 2,30     | 2,60     | 2,30     |
| <b>sep3</b>           | 100,50  | 121,30  | 125,90   | 137,60   | 145,10   | 148,00   |
| <b>Coste</b>          | 1164,97 | 1443,59 | 1707,82  | 2154,07  | 2546,19  | 2919,40  |
| <b>CO<sub>2</sub></b> | 2329,80 | 2854,94 | 3378,05  | 4265,87  | 5058,69  | 5534,30  |
| <b>Energía</b>        | 7667,61 | 9359,92 | 11098,86 | 13856,74 | 16437,90 | 18905,28 |
| <b>Barras</b>         | 56,90   | 56,50   | 62,90    | 64,20    | 70,00    | 75,50    |

Tabla 15 Valores óptimos medios en función de la Luz

### 3.4 ESTUDIO DE SENSIBILIDAD DE PRECIOS

A continuación vamos a estudiar cuanto influye el precio de cada material en el proceso de optimización y cómo la variación en el mismo puede afectar al coste del elemento. De este modo podemos saber cómo afectan las diferentes fluctuaciones de los precios de los materiales en el mercado al coste total y a la creación de una viga óptima de este tipo.

Únicamente se van a estudiar las variaciones en el coste y no en los otros criterios de optimización puesto que es el único criterio que puede presentar previsible variaciones en sus valores iniciales. Los valores de dos de los tres criterios restantes, las emisiones de CO<sub>2</sub> y el consumo energético, dependen del proceso de fabricación de los materiales y no se prevé una gran mejora en ellos en un futuro cercano, mientras que el resultado del criterio de optimización restante depende completamente del diseño del elemento, que es realizado de manera aleatoria durante la ejecución del algoritmo, por lo que no consideramos la realización del análisis de la variabilidad sobre alguno de estos criterios.

En este examen se van a utilizar siete intervalos de precios diferentes, siendo el central el coste actual de los diferentes materiales. A partir de este punto central se disponen tres intervalos incrementales y tres decrementales, cada uno con una diferencia del cinco por ciento con el anterior. Es decir, en intervalos de disminución del quince, diez y cinco por ciento e intervalos de aumento del coste base de las mismas magnitudes. Con ellos se cubren las variaciones en el precio que se pueden llegar a producir en un futuro próximo.

Para analizar estas variaciones se van a utilizar los parámetros descritos anteriormente en el caso de estudio estándar y se realizan un total de veinte simulaciones para cada escenario de estudio. En total, se estudian diecinueve escenarios diferentes, ya que solo se modifica el precio de uno de los materiales cada vez, lo que provoca que se necesiten trescientos ochenta sujetos óptimos distintos para poder llevarlo a cabo.

Dada la gran cantidad de muestras, y para facilitar la lectura, no se van a mostrar todos los resultados obtenidos, sino que se aporta una pequeña muestra de estos a modo de guía. La tabla 16 contiene el resultado de un sujeto para cada uno de los escenarios. Esto, aunque no sea muy representativo, sirve para obtener un orden de magnitud en los resultados que en la mayoría de casos no presentan una gran dispersión.

|           |      | Coste   | CO <sub>2</sub> | Energía | Barras |
|-----------|------|---------|-----------------|---------|--------|
| Hormigón  | -15% | 1087,77 | 2254,52         | 7371,37 | 51,00  |
|           | -10% | 1201,99 | 2441,72         | 7917,21 | 39,00  |
|           | -5%  | 1003,92 | 2083,96         | 6916,62 | 58,00  |
|           | 0%   | 1026,66 | 2084,76         | 6825,42 | 47,00  |
|           | 5%   | 1104,11 | 2333,20         | 7804,32 | 65,00  |
|           | 10%  | 1064,90 | 2162,17         | 7084,93 | 48,00  |
|           | 15%  | 1152,88 | 2428,39         | 8096,03 | 64,00  |
| Acero     | -15% | 1056,01 | 2616,43         | 8774,72 | 77,00  |
|           | -10% | 995,03  | 2218,79         | 7273,29 | 50,00  |
|           | -5%  | 1185,76 | 2536,95         | 8280,54 | 45,00  |
|           | 0%   | 1026,66 | 2084,76         | 6825,42 | 47,00  |
|           | 5%   | 1171,26 | 2404,95         | 8079,91 | 74,00  |
|           | 10%  | 1246,11 | 2433,27         | 8117,53 | 64,00  |
|           | 15%  | 1231,06 | 2203,25         | 2082,20 | 38,00  |
| Encofrado | -15% | 1133,32 | 2421,68         | 8023,17 | 57,00  |
|           | -10% | 1099,49 | 2239,76         | 7200,37 | 39,00  |
|           | -5%  | 1216,02 | 2551,78         | 8317,15 | 53,00  |
|           | 0%   | 1026,66 | 2084,76         | 6825,42 | 47,00  |
|           | 5%   | 1133,88 | 2271,04         | 7334,39 | 42,00  |
|           | 10%  | 1030,46 | 2087,84         | 6890,19 | 53,00  |
|           | 15%  | 1103,22 | 2223,63         | 7326,90 | 50,00  |

Tabla 16 Ejemplo de resultados

En las subsecuentes gráficas se han representado todos los valores obtenidos tras las simulaciones y sus respectivos valores medios. Estas muestras están agrupadas para cada intervalo de precios estudiado y se ha tendido una línea entre los valores medios. Esta recta marca la variabilidad de cada criterio de optimización en función del coste de los diferentes materiales.

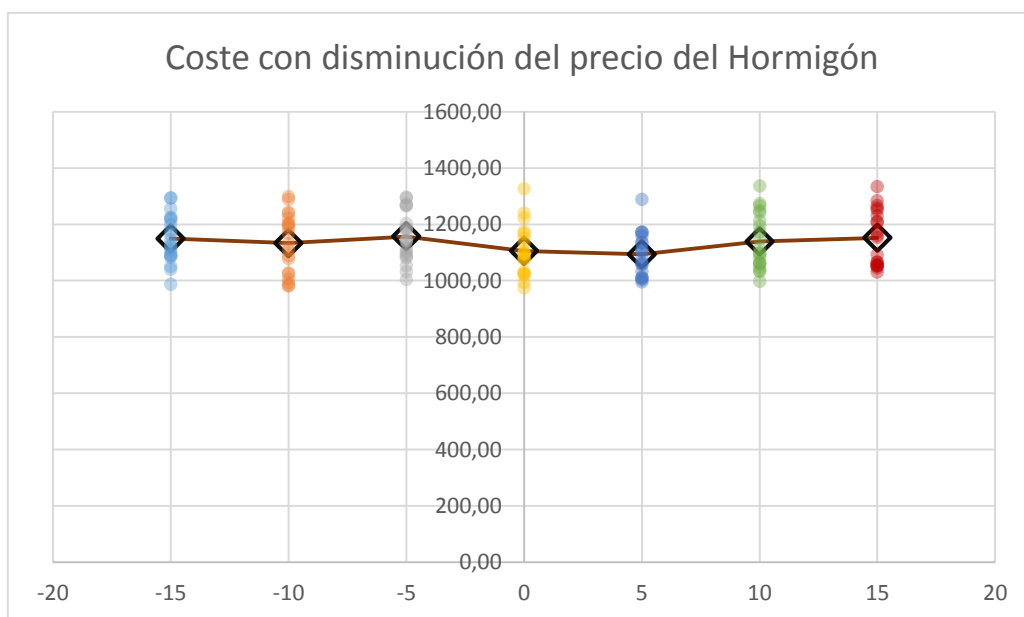


Gráfico 24 Variación del Coste con la variación del precio del Hormigón

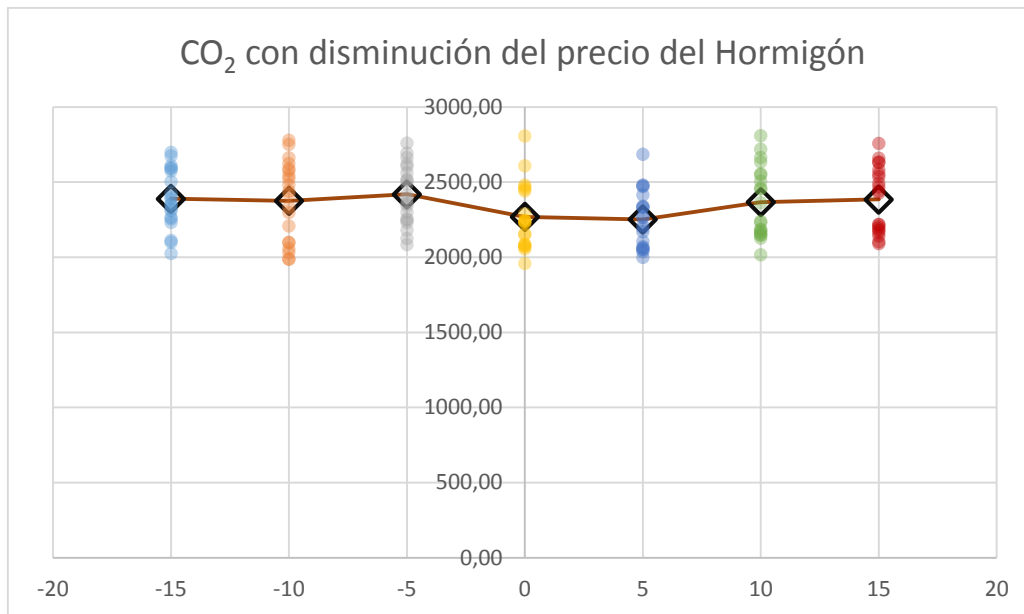


Gráfico 25 Variación del CO<sub>2</sub> con la variación del precio del Hormigón

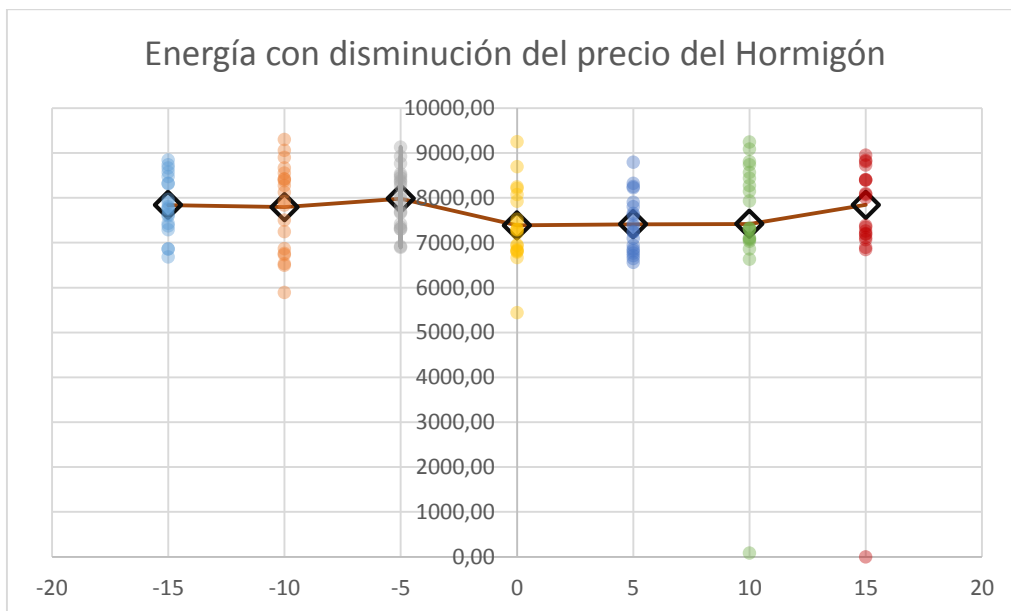


Gráfico 26 Variación del Consumo energético con la variación del precio del Hormigón



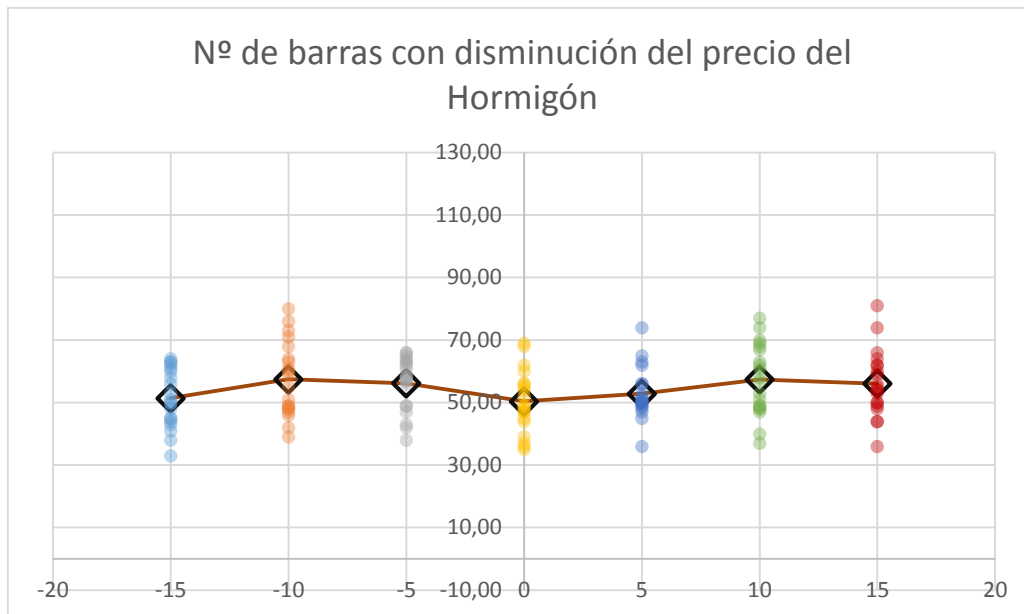


Gráfico 27 Variación del Número de Barras con la variación del precio del Hormigón

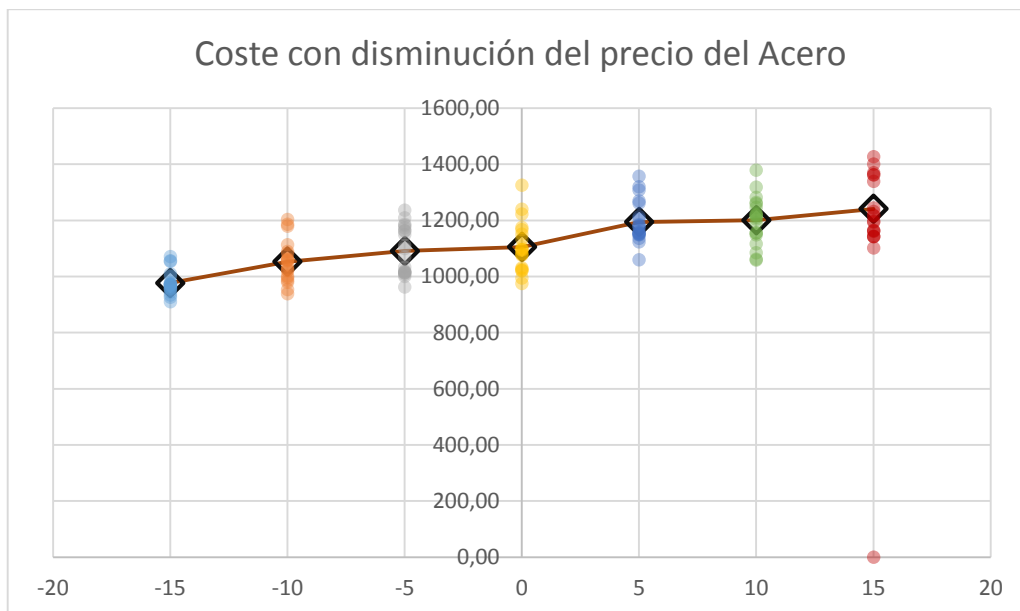


Gráfico 28 Variación del Coste con la variación del precio del Acero

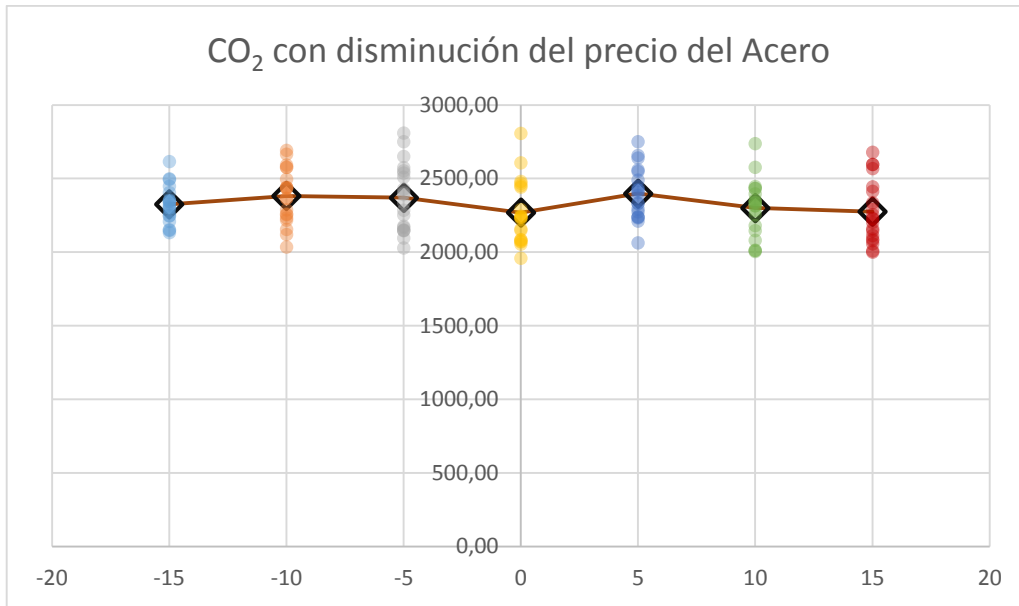


Gráfico 29 Variación del CO<sub>2</sub> con la variación del precio del Acero

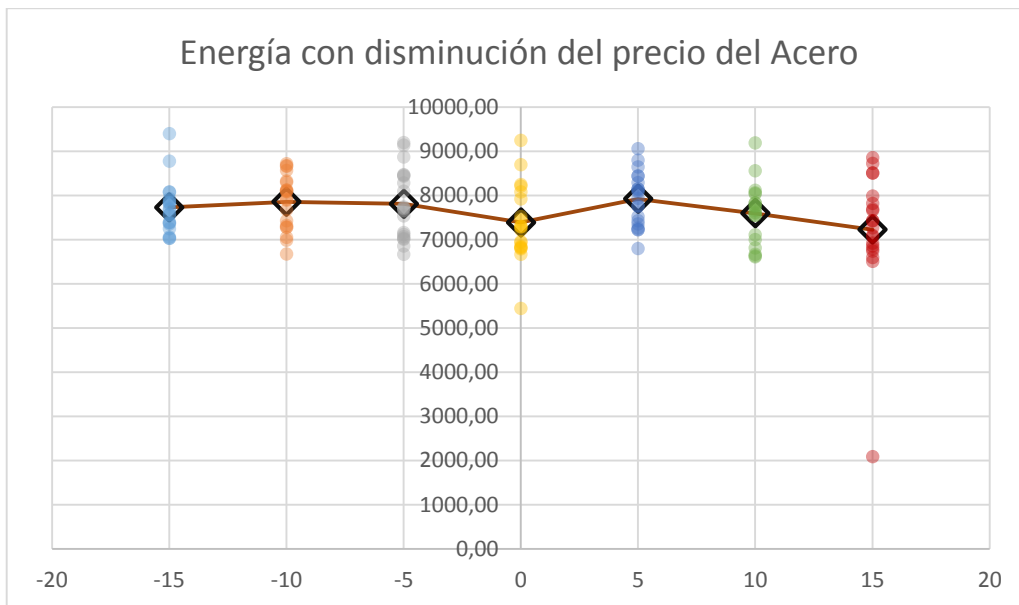


Gráfico 30 Variación del Consumo energético con la variación del precio del Acero

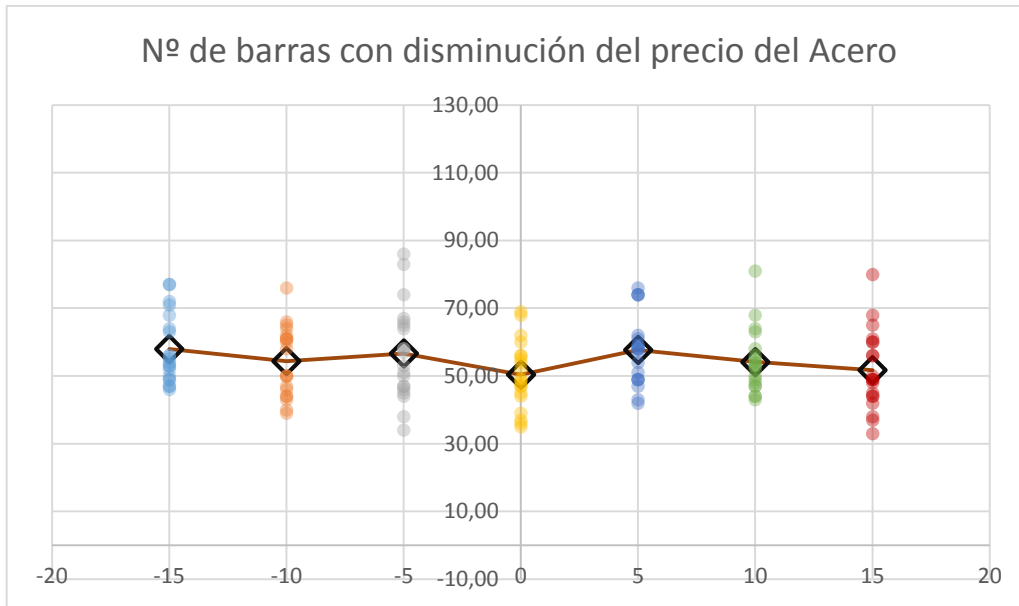


Gráfico 31 Variación del Número de Barras con la variación del precio del Acero

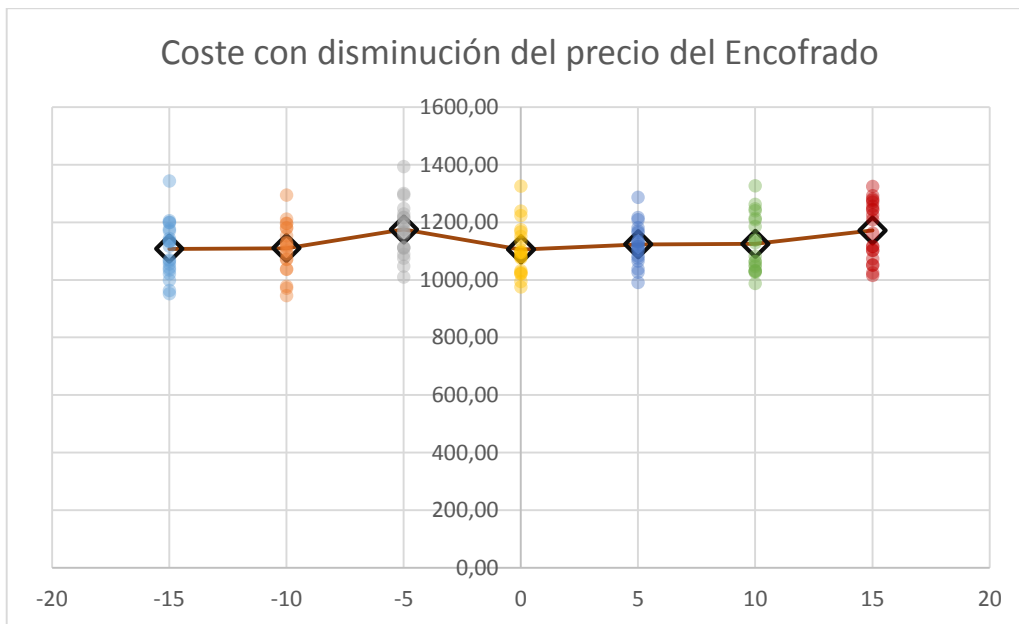


Gráfico 32 Variación del Coste con la variación del precio del Encofrado

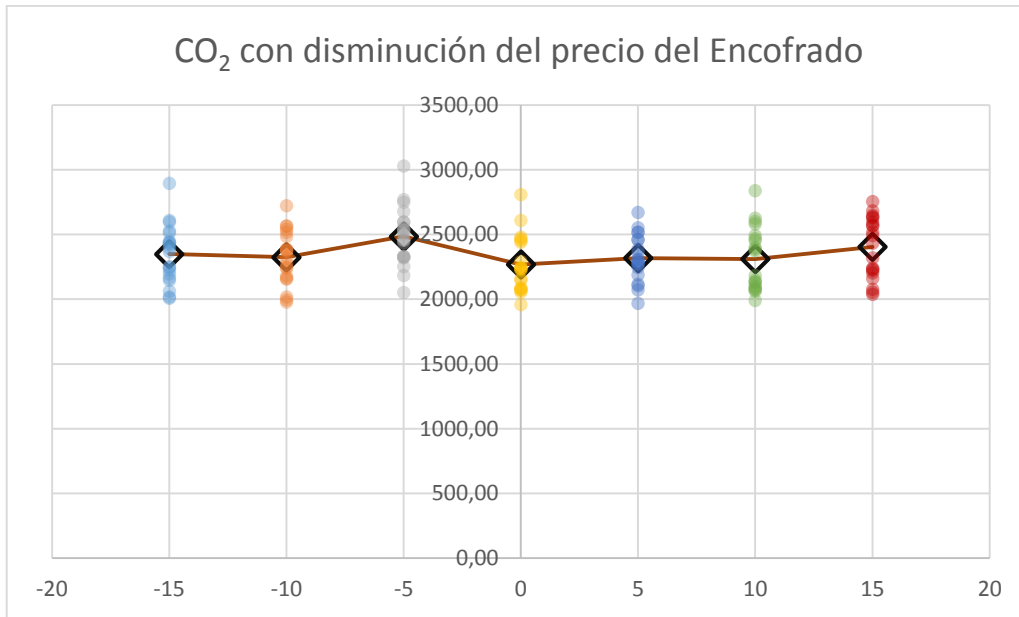


Gráfico 33 Variación del CO<sub>2</sub> con la variación del precio del Encofrado

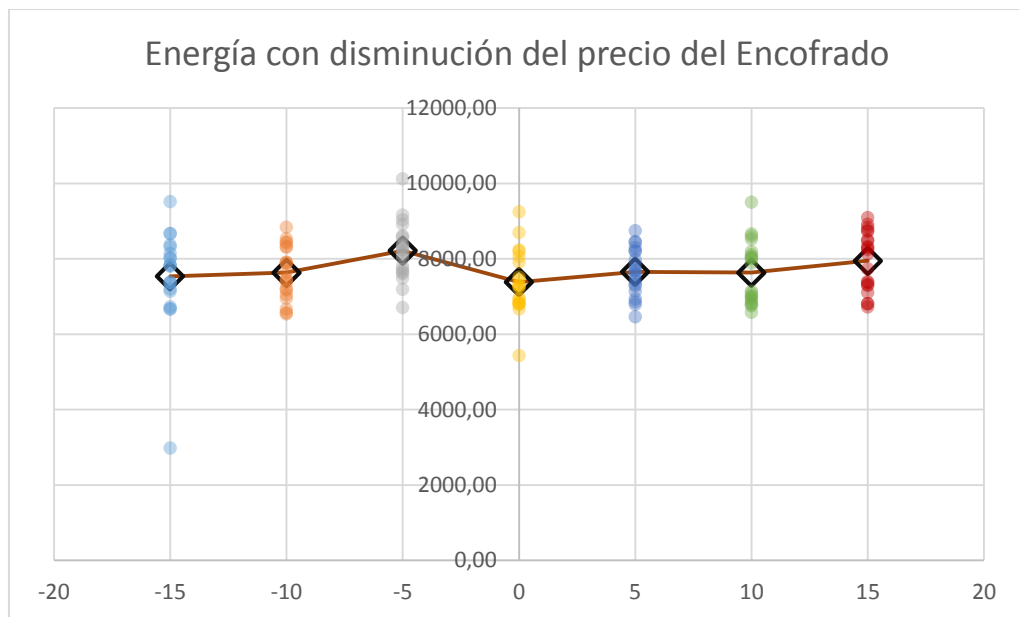


Gráfico 34 Variación del Consumo energético con la variación del precio del Encofrado

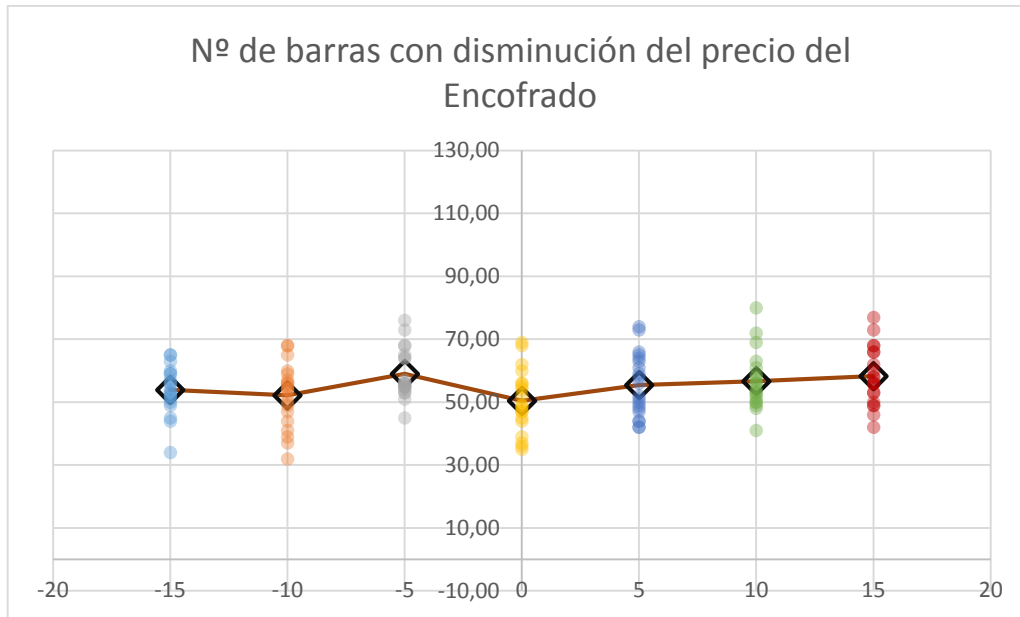


Gráfico 35 Variación del Número de barras con la variación del precio del Encofrado

En la mayoría de casos, las muestras no presentan una gran dispersión, pero existen algunas excepciones, sobre todo en lo relativo al número de barras. La mayor dispersión en este criterio de optimización se debe a la gran diferencia existente entre sus valores, pero, aún así, gran parte de los valores se agrupan en torno a la media. Los valores medios de todos los escenarios estudiados están contenidos en la tabla 17.

|                  |      | Coste   | CO <sub>2</sub> | Energía | Barras |
|------------------|------|---------|-----------------|---------|--------|
| <b>Hormigón</b>  | -15% | 1148,84 | 2389,43         | 7840,64 | 51,40  |
|                  | -10% | 1132,98 | 2374,99         | 7792,70 | 57,45  |
|                  | -5%  | 1156,71 | 2418,51         | 7982,77 | 56,15  |
|                  | 0%   | 1105,23 | 2268,57         | 7386,45 | 50,40  |
|                  | 5%   | 1093,33 | 2251,44         | 7410,49 | 52,85  |
|                  | 10%  | 1138,71 | 2366,46         | 7416,51 | 57,40  |
|                  | 15%  | 1151,81 | 2384,90         | 7841,14 | 56,05  |
| <b>Acero</b>     | -15% | 976,81  | 2325,47         | 7729,66 | 57,95  |
|                  | -10% | 1052,58 | 2382,46         | 7855,76 | 54,35  |
|                  | -5%  | 1090,82 | 2370,02         | 7808,44 | 56,65  |
|                  | 0%   | 1105,23 | 2268,57         | 7386,45 | 50,40  |
|                  | 5%   | 1194,02 | 2397,76         | 7918,89 | 57,70  |
|                  | 10%  | 1200,48 | 2300,64         | 7594,97 | 54,15  |
|                  | 15%  | 1240,75 | 2274,87         | 7224,97 | 51,70  |
| <b>Encofrado</b> | -15% | 1107,66 | 2348,36         | 7538,96 | 53,85  |
|                  | -10% | 1110,08 | 2324,54         | 7637,03 | 52,15  |
|                  | -5%  | 1175,13 | 2485,01         | 8217,19 | 58,95  |
|                  | 0%   | 1105,23 | 2268,57         | 7386,45 | 50,40  |
|                  | 5%   | 1122,97 | 2318,33         | 7655,46 | 55,40  |
|                  | 10%  | 1124,91 | 2310,41         | 7634,46 | 56,60  |
|                  | 15%  | 1171,57 | 2403,88         | 7948,22 | 58,30  |

Tabla 17 Valores medios obtenidos con la variación de precio

No existe ninguna tendencia común en cuanto a la variabilidad de los precios y los resultados. En muchos casos las tendencias observadas aparentan ser contraintuitivas, llegando incluso a incrementar el coste total medio cuando se disminuye el precio de los materiales (Gráfica X,Y). Hasta el momento se desconoce si existen diferencias entre las medias significativas que verificarían estas tendencias. Si estas no fueran significativas se podría considerar que las medias son iguales en todos los casos y que por lo tanto la variación del coste no afecta al resultado del criterio de optimización.

Para obtener unas conclusiones adecuadas sobre este hecho es necesario realizar un análisis sobre la varianza a través de la prueba ANOVA. Previo a este se han eliminado de los resultados aquellos valores que se muestran como valores atípicos en un diagrama de cajas y bigotes y se ha verificado la normalidad de las muestras tras dicha supresión. Los valores atípicos se producen debido a la aleatoriedad a la hora de elegir los valores del elemento estructural dentro del algoritmo, lo que provoca que ocasionalmente una de las poblaciones quede estancada en un óptimo local alejado del óptimo global y produzca una viga poco optimizada. Por este motivo se han dispuesto métodos como la mutación en el algoritmo para reducir estos estancamientos. De entre todos los óptimos solo cuatro han sido marcados como valores atípicos, lo cual supone un uno por ciento del total de elementos generados.

La prueba ANOVA es una prueba estadística basada en los métodos de regresión lineal y permite determinar si distintos valores presentan una diferencia significativa entre las medias. Se utiliza esta prueba ya que permite superar las limitaciones de otras pruebas similares para hacer constantes bilaterales por parejas que restringe el conjunto de variables comparable a dos en lugar de las siete que tenemos en nuestro caso. La hipótesis nula que se verifica con esta prueba es la inexistencia de diferencias significativas entre las medias de las diferentes variables estudiadas y se comprueba con un intervalo de confianza del noventa y cinco por ciento.

|              | Coste | CO <sub>2</sub> | Energía | Barras |
|--------------|-------|-----------------|---------|--------|
| <b>Horm.</b> | 0,218 | 0,115           | 0,284   | 0,124  |
| <b>Acero</b> | 0,000 | 0,165           | 0,109   | 0,154  |
| <b>Enc.</b>  | 0,053 | 0,054           | 0,144   | 0,059  |

Tabla 18 p-valores resultantes tras la prueba ANOVA

La tabla anterior muestra los resultados de las diferentes pruebas ANOVA para cada una de las variaciones de coste estudiadas. El valor que ha sido destacado es el único inferior a 0,05, valor umbral mínimo para la aceptación de la hipótesis nula del estudio, por lo que es el único caso en el cual esta hipótesis es descartada y en que se han encontrado diferencias significativas

entre las medias de las muestras. En el resto de escenarios estudiados no existe estas diferencias.

A pesar de que el único caso donde el coste de los materiales afecta al coste final del elemento optimizado es cuando se varía el coste del acero, las variaciones del resto de costes no producen una diferencia significativa en el resultado final. Esto es así porque el coste del elemento viene determinado principalmente por el acero. Como se demostrará a continuación, el resto de costes influye en el precio final, aunque no de manera significativa.

La variación del coste con la variación del precio del acero es menor que la variación del precio del material mismo. Al observar los valores de las medias, se observa que cuando el valor del material disminuye en un quince por ciento, el del elemento únicamente lo hace en un once por ciento; mientras que si el coste del material se aumenta en la misma cantidad, este aumento es de un doce por ciento. La variación no coincide en ningún caso con la del precio del acero, lo que significa que, aunque el acero sea el material más influyente en la determinación del precio final del óptimo, este no es el único factor influyente, y que también existe una influencia por parte del coste de los otros materiales, si bien no de un modo suficientemente relevante como para afectar al coste total por ellos mismos.

|               | <b>Coste</b> | <b>Porc. Variación</b> |
|---------------|--------------|------------------------|
| <b>-15,00</b> | 976,81       | 11,62%                 |
| <b>-10,00</b> | 1052,58      | 4,76%                  |
| <b>-5,00</b>  | 1090,82      | 1,30%                  |
| <b>0,00</b>   | 1105,23      | 0,00%                  |
| <b>5,00</b>   | 1194,02      | 8,03%                  |
| <b>10,00</b>  | 1200,48      | 8,62%                  |
| <b>15,00</b>  | 1240,75      | 12,26%                 |

*Tabla 19 Porcentaje de variación del coste medio al variar el precio del acero*

Las emisiones de CO<sub>2</sub> y el consumo energético no están influidas por el coste del material utilizado. De esto se deduce que las cantidades de los materiales en las soluciones finales son similares, ya que estos dos criterios dependen completamente de las cantidades necesarias para fabricar la viga estudiada. Gracias a esto podemos afirmar que el algoritmo está refinado para encontrar el óptimo, sino la variación del precio produciría anomalías en las cantidades de CO<sub>2</sub> y Kwh que supondrían sujetos con una mayor cantidad de material. Además los tres criterios que hemos visto anteriormente que son linealmente proporcionales están relacionados entre sí a través de esta cantidad de material y por lo tanto la proporción existente entre sus valores puede variar si cambia su valor por unidad de material.

Esta tendencia a mantener la cantidad total de los materiales de manera constante a lo largo de las diferentes optimizaciones es acorde con la no influencia de la variación sobre el número de

barras. Se podría discutir que este criterio debería estar influido por ella debido a que este criterio está relacionado directamente con la cantidad de uno de los materiales y por lo tanto con el coste, por lo que no sería extraño el pensar *a priori* que en una reducción en el coste del acero, el óptimo tendría una mayor cantidad de este. Algo que no se produce debido a que el número de barras es minimizado durante el algoritmo y que el número necesario de barras viene marcado por unos requisitos estructurales que, llegado a un punto, no se pueden disminuir ni compensar añadiendo más hormigón sin comprometer el precio del óptimo.

A lo largo de este apartado se ha verificado el correcto funcionamiento de la aplicación de optimización en un entorno BIM creadatanto por la ejecución (al poder realizar todas estas comprobaciones sin errores), como por los resultados obtenidos y su coherencia. Además se han establecido diferentes reglas de diseño y de cálculo extraídas extrapolando los resultados que son útiles para conocer *a priori* las características del elemento. También se ha verificado que el algoritmo responderá de una manera adecuada frente a las diferentes subidas y bajadas en el precio de mercado de los materiales, lo que garantiza su uso futuro, junto con la adaptación de los resultados obtenidos al precio aproximado de mercado del momento.

Con todo esto, podemos concluir que la creación de una aplicación integrada en un entorno BIM capaz de realizar una optimización multicriterio ha sido exitosa.



## 4. CONCLUSIONES

---

En este último apartado se pretende recopilar todo aquello que se ha realizado durante el trabajo con el fin de exponer los logros alcanzados, las limitaciones con las que nos hemos encontrado y las conclusiones obtenidas tras la realización de los diferentes estudios.

Además, dado el carácter novedoso de la investigación, recordemos que pese a que el número de investigaciones relacionadas con los BIM son numerosas al estar estos en auge, no existe una gran cantidad de ellas que se centren en la optimización, y mucho menos en aplicaciones estructurales de las mismas, por lo que se van a exponer tres vías diferentes mediante las cuales la investigación y el desarrollo de la aplicación se puede ampliar en un futuro. En principio las futuras aplicaciones tienen posibilidad de realización inmediata, aunque también existe alguna perteneciente a un futuro más lejano.

### 4.1 REVISIÓN DE LOS OBJETIVOS

Una vez terminada la investigación se va a realizar una revisión de los objetivos planteado en el apartado 1.2, para conocer cuáles han sido cumplidos y hasta qué punto, con la finalidad de conocer cuán exitosa ha sido esta investigación. Si recordamos, se establecieron un total de diez objetivos agrupados en tres secciones diferentes.

El primer conjunto de objetivos eran aquellos relativos al estudio de la viabilidad del proyecto y a la adquisición de la información necesaria. El segundo conjunto era el referente a la puesta en funcionamiento de la aplicación y el tercero de ellos se centraba en un conjunto de estudios paramétricos que corroboraran el correcto funcionamiento de la aplicación.

Como era de esperar, el resultado del estudio de viabilidad verifica la posibilidad de creación de una aplicación de optimización de estructuras de hormigón integrada en un entorno BIM. De hecho, la investigación ha ido un paso más allá y además de demostrar dicha viabilidad, ha demostrado que una de las líneas de investigación más prolíficas en la actualidad en lo relativo a la industria AEC es el estudio de los BIM y sus posibilidades y cómo a pesar de que existe alguna investigación sobre optimización y BIM, no existe un gran número que se dediquen exclusivamente a la optimización estructural. Por este motivo se ha determinado una carencia en este ámbito de investigación de los BIM en la actualidad y la determinación, por tanto, de un nicho de investigación y de evolución de este nuevo paradigma. La búsqueda de información sobre los algoritmos de optimización nos ha permitido seleccionar un algoritmo adecuado y efectivo, además de establecer las funciones objetivo deseadas, aunque en este aspecto se

debería profundizar más, sobre otros posibles algoritmos de optimización multiobjetivo para asegurarse de que el escogido es el que produce un mejor resultado.

Durante el desarrollo de la aplicación se ha aumentado de un modo eficaz y conveniente los conocimientos del programa BIM que se ha decidido utilizar, en este caso Autodesk Revit, tanto en lo relativo al uso del programa como usuario, como al uso del programa desde el punto de vista del desarrollador. Para completar esta formación y poder llevar a cabo el *plugin*, se ha mejorado el dominio del lenguaje de programación C#, tanto en funciones propias como en las exclusivas de la API del entorno BIM.

Tras este estudio inicial que sienta las bases, se ha desarrollado la aplicación, momento del proceso en que han aparecido obstáculos inesperados. Inicialmente se planteaba este *plugin* como una aplicación que realizaba una optimización monocriterio basada en uno de los cuatro criterios establecidos, pero una vez desarrollada nos hemos dado cuenta de que los resultados presentaban una gran mejoría si la optimización se realizaba desde un punto de vista multicriterio. Esto ha supuesto un cambio de perspectiva a la hora de aproximarse al problema y una nueva necesidad, la de conocer el funcionamiento de este tipo de algoritmos, previamente descartados en la etapa anterior. Tras esta investigación adicional, se ha llevado a cabo un leve estudio sobre los criterios de optimización y sus relaciones, lo que ha facilitado la utilización de un algoritmo multicriterio y ha garantizado el mejor funcionamiento posible del algoritmo.

La aplicación se ha programado siempre integrada con el entorno BIM Revit a través de la API correspondiente, lo que ha supuesto un constante uso de las funciones internas. Esta conexión no ha resultado problemática en general, si bien ha surgido algún conflicto, relacionado con las unidades, ya que Revit trabaja internamente con unidades imperiales y esto no puede ser modificado, por lo que para el correcto funcionamiento del *plugin* se ha tenido que implementar un cambio de unidades implícito en el código. En el apartado 2.3 se explica cómo ha surgido el problema y su solución.

Tras completar la programación del *plugin* y verificar su correcto funcionamiento se han realizado una serie de estudios estadísticos de los cuales se han extraído una serie de normas que ayudan a explicar el comportamiento de la viga. Estos estudios se muestran en el apartado 3 y a partir de ellos se establecen las relaciones que existen entre los diferentes criterios de optimización y entre cada uno de los mismos con la luz de cálculo, así como conclusiones sobre el funcionamiento estructural del elemento y de la influencia de cada uno de sus parámetros.

Para terminar, y con el fin de completar el documento, se adjuntan tres anejos. Dos de ellos presentan los códigos utilizados en la aplicación, con optimización monocriterio y multicriterio

respectivamente. El tercer anejo es una guía de usuario en la que se desarrolla paso a paso un ejemplo práctico de uso para solventar las dudas que pudieran surgir en el usuario, además de un ejemplo sobre cómo realizar el ajuste de los valores de los parámetros del algoritmo genético mediante el diseño de experimentos.

## 4.2 CONCLUSIONES

Se ha desarrollado una aplicación capaz de realizar la optimización multicriterio de un elemento estructural tipo viga integrada completamente dentro del entorno BIM Revit, con lo que se ha cumplido el objetivo principal del trabajo. Para alcanzar dicho objetivo se han seguido una serie de pasos que han supuesto la consecución de otros objetivos secundarios, lo que ha llevado a establecer una serie de conclusiones importantes.

- La incorporación de algoritmos de optimización a los entornos BIM proporciona una serie de oportunidades únicas a este tipo de software en expansión en la industria AEC, ya que se unen las ventajas de los BIM con la posibilidad de encontrar la mejor solución a un problema estructural de modo que mejora la aplicación de concepto de *Lean construction* a los proyectos.
- No se ha encontrado ningún método para cuantificar cuanta mejoría supone la utilización de algoritmos de optimización estructural integrados en un entorno BIM, ya que se trata de una mejora funcional difícilmente cuantificable. Aún así se demuestra que los algoritmos multicriterio consiguen reducir el coste de los elementos en un veinte por ciento respecto a algoritmos monocriterio y un mínimo de un treinta por ciento respecto a la no utilización de algoritmos. Además se han mostrado ejemplos de proyectos reales en los cuales el uso de los BIM ha logrado reducir el coste total hasta en un veinticinco respecto al coste inicial sin la utilización de este tipo de *software*.
- Se ha demostrado que los algoritmos genéticos multicriterio mejoran el resultado que se obtiene aplicando un algoritmo de optimización monocriterio con el coste de tiempo computacional, siempre que entre los criterios de optimización no exista una relación de linealidad.
- Existe una relación de linealidad entre el coste, la emisión de CO<sub>2</sub> y el consumo energético, siendo el segundo aproximadamente 2,20 veces el primero y el último 7,15 con el primero. El resto de relaciones también han sido cuantificadas.
- La geometría más eficiente a nivel estructural es una cuya relación canto-ancho sea superior a la unidad, lo que corresponde con la tipología conocida como viga de canto.
- La relación canto-luz con mayor eficiencia disminuye según aumenta la luz y oscila entre valores de L/15 y L/20 para luces entre cinco y diez metros.
- El factor más influyente en el coste es el precio del acero, siendo los precios del encofrado y del hormigón menos relevantes en el total.

### 4.3 FUTURAS VÍAS DE INVESTIGACIÓN

La investigación ha cumplido con sus objetivos iniciales y por ello se ha considerado terminada y exitosa, pero no por ello podemos considerar que se trate de una labor finalizada. La aplicación ha sido desarrollada como una primera aproximación de la integración de los algoritmos de optimización en los entornos BIM, y en este aspecto ha sido un éxito en su desarrollo y sus resultados, pero existen un gran número de posibilidades de ampliación y de mejora. En este apartado se exponen estas diferentes rutas que podría tomar la investigación en el futuro para mejorar las funcionalidades del entorno BIM.

Se establecen tres líneas de investigación futuras, la ampliación de tipologías de estructuras optimizadas dentro del entorno BIM, la introducción de nuevos y variados objetivos de optimización y la combinación de estas técnicas con diferentes tecnologías que se están incorporando en este momento a las plataformas BIM. Aunque a ciencia cierta no se conoce cómo va a evolucionar esta tecnología, preferentemente se deberían estudiar estas nuevas líneas en el mismo orden en que se exponen, ya que de este modo el campo de investigación se va ampliando paulatinamente.

#### 4.3.1 Diferentes tipologías estructurales

La aplicación desarrollada permite únicamente optimizar vigas de hormigón armado isostáticas destinadas a edificación, una tipología estructural muy concreta, limitada y poco habitual, ya que lo habitual serían barras con comportamiento hiperestático. Anteriormente se ha justificado la elección de esta tipología, siendo el motivo principal el desconocimiento del proceso de creación de dicha aplicación, pero una vez demostrado que es viable incorporar la optimización a un entorno BIM, ha llegado el momento de ampliar las tipologías estructurales.

Existen multitud de tipologías estructurales que pueden ser optimizadas, tanto en el conjunto destinado a edificación como en la ingeniería civil. En la actualidad hay una gran variedad de investigaciones en las que se han optimizado diferentes tipologías estructurales destinadas tanto a edificación, pórticos, forjados reticulares, forjados mixtos, muros de contención y de sótano, etc.; como en tipologías destinadas a la ingeniería civil, estribos de puentes, artesas, diferentes tipos de losas pretensas y postesas, puentes en voladizo, etc. Aún existe un mayor número de tipologías si ampliamos el campo de estudio a otros materiales como las estructuras de acero o las estructuras de madera, pues también existen investigaciones sobre su optimización.

Todas estas tipologías podrían ser incorporadas a diferentes entornos BIM. Para su incorporación será necesario estudiar las particularidades de cada una de ellas y los datos

necesarios para definir las, así como el entorno BIM más adecuado. Este, puede ser el más adecuado por diferentes motivos, incluyendo la adecuación del software BIM a la finalidad, por ejemplo Revit está destinado a edificación y por esto ha sido el elegido en nuestro caso, mientras que si estuviéramos optimizando un puente en artesía sería más adecuado utilizar un software BIM destinado a la ingeniería civil como AllPlan.

Al incorporar nuevas tipologías estructurales también será necesario incorporar un mayor catálogo de hipótesis de cargas. Las hipótesis incorporadas deberían estar relacionadas con la tipología estructural estudiada, ya que carece de sentido aumentar la complejidad del algoritmo incorporando una hipótesis de carga que no es aplicable al caso estudiado, por ello no se considera una línea de investigación separada. Una excepción se produciría si se quisieran incorporar las cargas sísmicas o de fuego a la optimización, cuya compleja evaluación y diferentes modos de cálculo en función de la naturaleza de la estructura supondrían la apertura de un nuevo foco de investigación.

#### 4.3.2 Diferentes objetivos de optimización y diferentes algoritmos

En la optimización se han utilizado un total de cuatro objetivos diferentes en un algoritmo genético multicriterio. Aunque esta elección como ya ha quedado bien justificada es motivada, existen diferentes posibilidades tanto en el uso de criterios de optimización como en la elección de algoritmos e optimización. El hecho de que se haya realizado una optimización multicriterio ya supone un paso importante en cuanto a técnicas de optimización respecto a la optimización monocriterio, como se ha demostrado en apartados anteriores, pero no cubre toda la casuística.

Una gran variedad de criterios de optimización que se pueden incorporar con diferentes finalidades. Estos criterios tienen que ser seleccionados para la tipología estructural adecuada y con la finalidad deseada, ya que no tendría sentido evaluar la durabilidad de una estructura como un factor decisivo en el diseño si se pretende realizar una construcción temporal que apenas vaya a tener unos meses de vida. Por este motivo, los criterios tienen que ser seleccionados con cuidado y adaptados a la finalidad del estudio.

Algunos de los criterios que puedan presentar un mayor interés para las estructuras de edificación podrían ser la búsqueda del mayor coeficiente de seguridad de cargas, la evaluación de la durabilidad de la estructura, la minimización de despuntes de las barras o la incorporación de diferentes criterios de carácter social. El número de diferentes criterios que se puede encontrar es numeroso y la selección de un mayor número de criterios de optimización puede ser contraproducente, ya que aumenta el tiempo de computación sin realmente mejorar la solución aunque se evalúe desde un mayor número de puntos de vista.

Además de estos criterios también existen una gran variedad de algoritmos de optimización en la actualidad y se han realizado investigaciones sobre los resultados entre los mismos y cual es el más adecuado para cada uso, este punto se ha comentado anteriormente en el apartado 1. En función de la tipología empleada se debe utilizar un algoritmo u otro para obtener unos mejores resultados, lo que significa unos diferentes parámetros para ajustar estos algoritmos.

De cara a una posible aplicación de uso público comercial por parte de un usuario no documentado en el funcionamiento interno de los algoritmos de optimización, se puede estudiar la viabilidad de crear una aplicación que cumpla con diferentes demandas y que sea el propio usuario quién defina en cada caso los criterios deseados.

#### 4.3.3 Combinación con otras tecnologías integradas en los entornos BIM

La última línea de investigación que podemos constatar a partir del desarrollo de esta aplicación no está directamente relacionada con las estructuras o con la optimización como las anteriores, sino que se basa en ampliar sus funcionalidades de cara a los BIM en combinación con otras tecnologías e investigaciones que se realizan en la actualidad en este ámbito. Esta es la línea más abstracta de todas, ya que realmente en la actualidad se desconoce hasta donde pueden llegar los BIM y sus capacidades reales, pero sí que existen grandes posibilidades de ampliación para compaginar varias de las nuevas utilidades, la mayoría de ellas relacionadas con la filosofía *Lean Construction*.

Uno de los mayores campos de estudio en la actualidad en los BIM es el trabajo en la nube, que ya se ha apuntado en algunos momentos de esta investigación. De hecho, existen entornos BIM que ya incorporan la posibilidad de la edición de archivos por varios usuarios simultáneamente a través de un servicio online. Si incorporamos herramientas de optimización estructural a este ámbito obtenemos mecanismos eficaces para intercambiar datos reales, veraces y acertados entre los diferentes agentes que están realizando el BIM, en este caso un intercambio de datos entre la figura del calculista y del diseñador, quienes pueden estudiar desde edades muy tempranas la eficacia de cierta tipología estructural en ese diseño. Además, esta eficacia está basada en ser la mejor posible desde el inicio, lo que evita problemas generados a partir de la revisión de los datos al ser ajustados durante el proceso.

Existen otros campos de estudio de los BIM, donde la optimización supone una gran ventaja y mejora. El *Additive Manufacturing* tiene un gran serie de facilidades si es tratado desde BIM, ya que existe una correspondencia muy directa entre el modelo creado por ordenador y su realización física, si además incorporamos la optimización, nos permite, de modo sistemático, crear una pieza con un menor consumo de material que cumpla con las demandas deseadas,

que evitaría problemas derivados de las inexactitudes de la construcción. En una relación menos directa podemos combinar las tecnologías de la Realidad Virtual o la robótica.

Es innegable que los BIM suponen un cambio radical en la industria AEC a todo los niveles y queda todavía mucho terreno para que sus usos se amplíen y mejoren. En este documento se ha conseguido contribuir a esta expansión a través de la introducción de la optimización estructural al conjunto de herramientas disponibles en este entorno, pero lo que se ha ido exponiendo es solo un pequeño fragmento del gran marco BIM. Se desconocen los límites de este nuevo paradigma a quién todos los autores auguran un largo futuro, lo que es seguro es que la industria AEC está cambiando radicalmente a todos los niveles y cómo afirma Chuck Eastman:

*“This is an exciting time to be an architect, an engineer, or any other AEC industry professional.”(Eastman et al., 2011, P.352).*

#### 4.4 AGRADECIMIENTOS

La realización de este trabajo hubiese sido imposible sin la colaboración de muchas personas. Muchas gracias a todos aquellos quienes en alguna ocasión han mostrado interés y me han prestado ayuda. Especial atención a mi tutor Víctor Yepes Piqueras, por su inestimable labor como guía; a Gemma Burgos Segarra por su gran apoyo, sus ánimos y sus labores de corrección; a Adrián Calero Sevilla por su ayuda en la programación y a Ignacio Barrios Oltra y Manuel Sifón Miralles. También agradecer a la Universitat Politècnica de València, por proporcionarme acceso a todos los programas, artículos y herramientas necesarias para la realización de este trabajo.

Muchas gracias.



## 5. BIBLIOGRAFÍA

---

1. About Us - Bentley Systems, Incorporated [WWW Documento], URL <https://www.bentley.com/en/about-us> (accedido 17-3-16).
2. AEC DevBlog [WWW Documento], URL <http://adndevblog.typepad.com/aec/> (accedido 21-3-16).
3. ASALE, R., Diccionario de la lengua española - Edición del Tricentenario [WWW Documento]. Dicc. Leng. Esp. URL <http://dle.rae.es/?id=R7YxPPp> (accedido 22-3-16).
4. Autodesk | Software de diseño, ingeniería y entretenimiento 3D [WWW Documento], URL <http://www.autodesk.es/> (accedido 17-3-16).
5. Autodesk - Autodesk Developer Network - Autodesk® Revit, Autodesk® Revit® Architecture, Autodesk® Revit® Structure and Autodesk® Revit® MEP [WWW Documento], n.d. URL <http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=2484975> (accedido 21-3-16).
6. Autodesk - My First Plug-in Training - My First Revit Plug-in Overview [WWW Documento], URL <http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=16777469> (accedido 5-3-16).
7. Autodesk Inc., n.d. Autodesk - Investor Relations - News Release [WWW Documento]. URL <http://investors.autodesk.com/phoenix.zhtml?c=117861&p=irol-newsArticle&ID=261618> (accedido 17-3-16).
8. Autodesk Revit Help [WWW Documento], URL <http://help.autodesk.com/view/RVT/2014/ENU/> (accedido 5-3-16).
9. Azhar, S., 2011. Building Information Modeling (BIM): Trends, Benefits, Risks, and Challenges for the AEC Industry. *Leadersh. Manag. Eng.* 11, 241–252. doi:10.1061/(ASCE)LM.1943-5630.0000127
10. Buswell, R.A., Soar, R.C., Gibb, A.G.F., Thorpe, A., 2007. Freeform Construction: Mega-scale Rapid Manufacturing for construction. *Autom. Constr.* 16, 224–231. doi:10.1016/j.autcon.2006.05.002
11. Chi, H.-L., Wang, X., Jiao, Y., 2014. BIM-Enabled Structural Design: Impacts and Future Developments in Structural Modelling, Analysis and Optimisation Processes. *Arch. Comput. Methods Eng.* 22, 135–151. doi:10.1007/s11831-014-9127-7

12. Cho, Y.S., Lee, S.I., Bae, J.S., 2014. Reinforcement Placement in a Concrete Slab Object Using Structural Building Information Modeling. *Comput.-Aided Civ. Infrastruct. Eng.* 29, 47–59. doi:10.1111/j.1467-8667.2012.00794.x
13. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* 6, 182–197. doi:10.1109/4235.996017
14. Diao, Y., Kato, S., Hiyama, K., 2011. Development of an optimal design aid system based on building information modeling. *Build. Simul.* 4, 315–320. doi:10.1007/s12273-011-0054-3
15. Eastman, C., Lee, J., Jeong, Y., Lee, J., 2009. Automatic rule-based checking of building designs. *Autom. Constr.* 18, 1011–1033. doi:10.1016/j.autcon.2009.07.002
16. Eastman, C.M., 1975. The use of computers instead of drawings in building design. *AIA J.* 63, 46–50.
17. Eastman, C.M., Teicholz, P., Sacks, R., Liston, K., 2011. *BIM handbook : a guide to building information modeling for owners, managers, designers, engineers, and contractors*, 2nd ed. ed. John Wiley & Sons, cop2011, Hoboken, NJ.
18. Eastman, C.M., Teicholz, P., Sacks, R., Liston, K., 2008. *BIM handbook : a guide to building information modeling for owners, managers, designers, engineers, and contractors*, 1st. ed. ed. John Wiley & Sons, cop2011, Hoboken, NJ.
19. Estación de trabajo, 2015. . Wikipedia Encicl. Libre. [WWW Documento]. URL [https://es.wikipedia.org/wiki/Estaci%C3%B3n\\_de\\_trabajo](https://es.wikipedia.org/wiki/Estaci%C3%B3n_de_trabajo) (accedido 17-3-16).
20. Fennes, S., 1995. Computer representations of design standards and building codes: U.S. perspective. *Int. J. Constr. Inf Technol.* 3, 13–34.
21. Frequently Asked Questions About the National BIM Standard-United States™ | National BIM Standard - United States [WWW Documento], URL <https://www.nationalbimstandard.org/faqs> (accedido 14-3-16).
22. Gunnerson, E., 2001. *A Programmer's Introduction to C#, 2nd ed.* ed, Books for Professionals bt Professionals. Apress, Berkeley.
23. Haddad, O., Afshar, A., Mariño, M.A., 2006. Honey-Bees Mating Optimization (HBMO) Algorithm: A New Heuristic Approach for Water Resources Optimization. *Water Resour. Manag.* 20, 20. doi:10.1007/s11269-005-9001-3
24. Hare, W., Nutini, J., Tesfamariam, S., 2013. A survey of non-gradient optimization methods in structural engineering. *Adv. Eng. Softw.* 59, 19–28. doi:10.1016/j.advengsoft.2013.03.001

25. Hu, Z., Zhang, J., Deng, Z., 2008. Construction Process Simulation and Safety Analysis Based on Building Information Model and 4D Technology. *Tsinghua Sci. Technol.* 13, Supplement 1, 266–272. doi:10.1016/S1007-0214(08)70160-3
26. Knutt, E., 2015. BIM+ - Spain launches BIM strategy with pencilled-in 2018 mandate [WWW Documento]. URL <http://www.bimplus.co.uk/news/spain-launches-bim-strategy-pencilled-2018-mandate/> (accedido 21-3-16).
27. König, M., Dirnbek, J., Stankovski, V., 2013. Architecture of an open knowledge base for sustainable buildings based on Linked Data technologies. *Autom. Constr.* 35, 542–550. doi:10.1016/j.autcon.2013.07.002
28. La Asociación - BuildingSMART Spanish Chapter [WWW Documento], n.d. URL <http://www.buildingsmart.es/bssch/la-asociaci%C3%B3n/> (accedido 21-3-16).
29. Laiserin, J., n.d. Comparing Pommes and Naranjas [WWW Documento]. URL <http://www.laiserin.com/features/issue15/feature01.php> (accedido 17-3-16).
30. Laiserin, J., 2003. The Great Debate BIM [WWW Document]. *Laiserin Lett.* URL <http://www.laiserin.com/features/bim/index.php> (accedido 17-3-16).
31. MacDonald, J., 2015. History of Building Information Modelling | CODEBIM: Collaborative Design Education using BIM.
32. Martí, J.V., García-Segura, T., Yepes, V., 2016. Structural design of precast-prestressed concrete U-beam road bridges based on embodied energy. *J. Clean. Prod.* 120, 231–240. doi:10.1016/j.jclepro.2016.02.024
33. Martínez-Martin, F.J., González-Vidoso, F., Hospitaler, A., Yepes, V., 2012. Multi-objective optimization design of bridge piers with hybrid heuristic algorithms. *J. Zhejiang Univ. Sci. A* 13, 420–432. doi:10.1631/jzus.A1100304
34. Miguel, L.F.F., Lopez, R.H., Miguel, L.F.F., 2013. Multimodal size, shape, and topology optimisation of truss structures using the Firefly algorithm. *Adv. Eng. Softw.* 56, 23–37. doi:10.1016/j.advengsoft.2012.11.006
35. Ministerio de Fomento, 2008. Instrucción de Hormigón Estructural, EHE-08.
36. Ministry of Works Malaysia, 2015. Construction Industry Transformation Programme, CITP.
37. Mistry, P., Maes, P., 2009. SixthSense: A Wearable Gestural Interface, in: *ACM SIGGRAPH ASIA 2009 Sketches, SIGGRAPH ASIA '09*. ACM, New York, NY, USA, p. 11:1–11:1. doi:10.1145/1667146.1667160
38. Payá, I., Yepes, V., Clemente, J.J., González, F., 2006. Optimización heurística de pórticos de edificación de hormigón armado.

39. Paya-Zaforteza, I., Yepes, V., Hospitaler, A., González-Vidosa, F., 2009. CO<sub>2</sub>-optimization of reinforced concrete frames by simulated annealing. *Eng. Struct.* 31, 1501–1508. doi:10.1016/j.engstruct.2009.02.034
40. Porwal, A., 2012. Building Information Modeling–Based Analysis to Minimize Waste Rate of Structural Reinforcement. *J. Constr. Eng. Manag.* 138, 943–954. doi:10.1061/(ASCE)CO.1943-7862.0000508
41. Sacks, R., Koskela, L., Dave, B., Owen, R., 2010. Interaction of Lean and Building Information Modeling in Construction. *J. Constr. Eng. Manag.* 136, 968–980. doi:10.1061/(ASCE)CO.1943-7862.0000203
42. SARCAR, M.M.M., RAO, K.M., NARAYAN, K.L., 2008. *Computer Aided Design and Manufacturing*. PHI Learning Pvt. Ltd.
43. Schlueter, A., Thesseling, F., 2009. Building information model based energy/exergy performance assessment in early design stages. *Autom. Constr.* 18, 153–163. doi:10.1016/j.autcon.2008.07.003
44. Singh, V., Gu, N., Wang, X., 2011. A theoretical framework of a BIM-based multi-disciplinary collaboration platform. *Autom. Constr., Building Information Modeling and Changing Construction Practices* 20, 134–144. doi:10.1016/j.autcon.2010.09.011
45. The LaiserinLetter (tm) | About [WWW Documento], n.d. URL <http://www.laiserin.com/about/index.php> (accedido 17-3-16).
46. Torres-Machi, C., Yepes, V., Alcalá, J., Pellicer, E., 2013. Optimization of high-performance concrete structures by variable neighborhood search. *Int. J. Civ. Eng.* 11, 90–99.
47. van Nederveen, G.A., Tolman, F.P., 1992. Modelling multiple views on buildings. *Autom. Constr.* 1, 215–224. doi:10.1016/0926-5805(92)90014-B
48. Víctor Yepes, 2014. Programación lineal. Optimización y programación matemática #Optimización de estructuras de hormigón. [WWW Document]. <http://victoryepes.blogs.upv.es/> (accedido 18-3-16).
49. Wang, X., Dunston, P.S., 2011. Comparative Effectiveness of Mixed Reality-Based Virtual Environments in Collaborative Design. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* 41, 284–296. doi:10.1109/TSMCC.2010.2093573
50. Wang, X., Kim, M.J., Love, P.E.D., Kang, S.-C., 2013. Augmented Reality in built environment: Classification and implications for future research. *Autom. Constr.* 32, 1–13. doi:10.1016/j.autcon.2012.11.021
51. Wenqi Huang, Duanbing Chen, 2008. *An Efficient Quasi-Human Heuristic Algorithm for Solving the Rectangle-Packing Problem*. INTECH Open Access Publisher.

52. Wu, I.-C., Chang, S., 2013. Visual Req calculation tool for green building evaluation in Taiwan. *Autom. Constr.* 35, 608–617. doi:10.1016/j.autcon.2013.01.006
53. Yepes, V., 2015. Optimización y programación matemática #Optimización de estructuras de hormigón. [WWW Document]. <http://victoryepes.blogs.upv.es/> (accedido 17-3-16).
54. Yepes, V., 2012. Apuntes de optimización heurística.
55. Yepes, V., Alcalá, J., Perea, C., González-Vidosa, F., 2008. A parametric study of optimum earth-retaining walls by simulated annealing. *Eng. Struct.* 30, 821–830. doi:10.1016/j.engstruct.2007.05.023
56. Yepes, V., García-Segura, T., Moreno-Jiménez, J.M., 2015. A cognitive approach for the multi-objective optimization of RC structural problems. *Arch. Civ. Mech. Eng.* 15, 1024–1036. doi:10.1016/j.acme.2015.05.001
57. Yepes, V., Gonzalez-Vidosa, F., Alcalá, J., Villalba, P., 2012. CO2-Optimization Design of Reinforced Concrete Retaining Walls Based on a VNS-Threshold Acceptance Strategy. *J. Comput. Civ. Eng.* 26, 378–386. doi:10.1061/(ASCE)CP.1943-5487.0000140
58. Yepes, V., Martí, J.V., García-Segura, T., 2015. Cost and CO2 emission optimization of precast–prestressed concrete U-beam road bridges by a hybrid glowworm swarm algorithm. *Autom. Constr.* 49, 123–134. doi:10.1016/j.autcon.2014.10.013

## ANEXO 1: CÓDIGO MONOCRITERIO

Este anexo contiene el código que se ha desarrollado siguiendo las directrices mostradas en el apartado 2.2. Con ello se consigue una aplicación plenamente funcional capaz de optimizar en función de uno de los criterios estudiados pero evaluando los cuatro simultáneamente mediante un algoritmo genético.

Esta aplicación se ejecuta directamente desde el entorno BIM Autodesk Revit sin necesidad de ninguna herramienta externa. Al tratarse de un paso intermedio en la realización del objetivo final no se le permiten realizar modificaciones sobre el criterio a optimizar desde dentro de la plataforma BIM, sino que estas deben ser realizadas exclusivamente desde el código. Esto supone una limitación menor, ya que durante la fase de investigación se puede acceder al código libremente y la aplicación final funcionará de modo multicriterio.

En el código se pueden apreciar comentarios explicativos sobre las funcionalidades del conjunto de líneas, de este modo se puede seguir en todo momento en que punto del algoritmo se encuentra el código y entender completamente su funcionamiento.

```
using Sytem;
using System.Collections.Generic;
using System.Linq;
using System.Diagnostics;
using Autodesk.Revit.DB;
using Autodesk.Revit.DB.Architecture;
using Autodesk.Revit.UI;
using Autodesk.Revit.UI.Selection;
using Autodesk.Revit.ApplicationServices;
using Autodesk.Revit.Attributes;

[TransactionAttribute(TransactionMode.Automatic)]
[RegenerationAttribute(RegenerationOption.Manual)]
public class Plugin2 : IExternalCommand
{
    public Result Execute(ExternalCommandData commandData, ref string message,
        ElementSet elements)
    {
        //Connexión con la API y el documento abierto
        UIApplication uiApp = commandData.Application;
        UIDocument uidoc = uiApp.ActiveUIDocument;
        Application app = uiApp.Application;
        Document doc = uiApp.ActiveUIDocument.Document;
        //El try-catch se utiliza para manejar errores
        try
        {
            //Selección de elemento a optimizar
            //Añadir filtro de vigas!
            Reference pickedRef = null;
            Selection sel = uidoc.Selection;
            pickedRef = sel.PickObject(ObjectType.Element, "Selecciona un
elemento");
            if (pickedRef != null)
```

```

    { }
    TaskDialog.Show("Revit", "Se ha añadido un elemento a la
seleccion");
    Element ele = doc.GetElement(pickedRef);
    FamilyInstance famInst = ele as FamilyInstance;
    //Control del tiempo
    Stopwatch time = Stopwatch.StartNew();
    time.Start();
    //Parametros extraidos y editados
    //Shared Parameters del archivo "ConcreteBeamOptSP"
    Guid D1guid = new Guid("1e5d8bc3-bc41-4e9b-9df9-70a68af1e2f3");
    Guid D2guid = new Guid("178a72dc-8d91-4028-9d2e-b64b4ae587d6");
    Guid D3guid = new Guid("b0388a29-e487-4118-bbfa-c1c132fd9dfc");
    Guid n1guid = new Guid("94878554-76a4-408b-9b9d-3c12eba25dc5");
    Guid n2guid = new Guid("ad32a0dc-c1ad-4bb0-bb45-a02a5c6cfb03");
    Guid sep3guid = new Guid("31438e3e-0ab7-4592-8df3-9a8515f0c7b6");
    Guid Hguid = new Guid("f942ad4d-759b-428e-ba47-ff4a568fbc1");
    Guid bguid = new Guid("a1540d7b-dab0-482f-996b-9859e39f6e8e");
    Guid cguid = new Guid("f8a30d8c-8c1d-4cd7-aa3b-1c6e98d29f1b");
    Guid costeguid = new Guid("c5af969d-a46a-4ef7-9f81-11d76f584826");
    Guid co2guid = new Guid("0520e6b5-9769-4158-9680-d2dbce487bcc");
    Guid energiaguid = new Guid("bb2f3cf8-6a53-4873-b87f-7134946be99a");
    Guid Gguid = new Guid("a118fd92-6434-49d3-ac1d-9517936a1f1a");
    Guid Qguid = new Guid("f0062670-1e84-4c0f-91c8-88192c58d68d");
    Guid recguid = new Guid("af389967-2771-4544-a42d-8a15f368eee7");
    Guid espguid = new Guid("9e37c064-6dec-48e6-9aa7-84fc1ccb41d9");
    Guid descenguid = new Guid("1ccb90d9-f967-44ba-aa0c-21547fee4d67");
    Guid genguid = new Guid("93b6cc37-1074-4292-ad86-e119159a2323");
    Guid mutguid = new Guid("aaed5b32-e805-401e-8632-3a64e1178067");
    Guid prodguid = new Guid("d3b8c39c-2dcd-42c0-8ccf-f8887fecf501");
    Guid fitnessguid = new Guid("24fa46d7-c686-4688-bcfb-2f1dbe0f6334");

    //Parámetros asignados desde Revit
    double Lc = 1000 *
(Math.Round((famInst.get_Parameter(BuiltInParameter.INSTANCE_LENGTH_PARAM).AsDouble() / 3.2808, 2)); //Luz de calculo en mm
    double rec =
Math.Round((((famInst.Symbol.get_Parameter(recguid).AsDouble())) / 0.0033),
0); //Recubrimiento nominal en mm
    double G = (famInst.Symbol.get_Parameter(Gguid).AsDouble()) /
1000; //Carga gravitatoria total en N/mm
    double Q = (famInst.Symbol.get_Parameter(Qguid).AsDouble()) /
1000; //Carga Variable en N/mm
    int max_especimenes =
(famInst.Symbol.get_Parameter(espguid).AsInteger()); //Numero maximo de
especimenes conservados en cada generacion
    int generacion_max =
famInst.Symbol.get_Parameter(genguid).AsInteger(); //Numero de generaciones
    int max_descen =
famInst.Symbol.get_Parameter(descenguid).AsInteger(); //Numero maximo de
descendientes generados
    int prob_mut = famInst.Symbol.get_Parameter(mutguid).AsInteger();
//Probabilidad que se produzca una mutacion en %
    int prob_prodigio =
famInst.Symbol.get_Parameter(prodguid).AsInteger(); //Probabilidad de generacion
de un hio prodigo en %
    int fitness =
famInst.Symbol.get_Parameter(fitnessguid).AsInteger(); //Probabilidad de que los
progenitores sean soluciones de Pareto

```

```

int num_especimenes = 0;
int generacion = 0;
int num_descen = 0;
char crit = 'p'; //Criterio de ordenación p-precio/c-co2/e-coste
energético
int critopt = 0;
int v = 0;
double[] viga = new double[13];
double[,] poblacion = new double[max_especimenes, viga.Length];
double[,] descendencia = new double[max_descen, viga.Length];
Random rnd = new Random();
Generacion descen = new Generacion();
int config = descen.config();
if (crit == 'p') { critopt = 9; }
if (crit == 'c') { critopt = 10; }
if (crit == 'e') { critopt = 11; }
if (crit == 'b') { critopt = 12; }
do
{
    /*Orden del codigo de la viga:
    b(ancho), c(canto), H(tipo de hormigon), D1(diametro inferior),
    D2(diametro superior), D3(diametro estribo), n1(nº barras D1), n2(nº barras D2),
    sep3(separacion entre estribos)*/
    //Generación especímenes iniciales
    Viga pionero = new Viga();
    v = pionero.generaviga(ref viga, pionero, rnd, rec, Lc, G, Q);
    for (int i = 0; i < viga.GetLength(0); i++)
    {
        poblacion[num_especimenes, i] = viga[i];
        Console.WriteLine(viga[i]);
    }

    num_especimenes++;
}
while (num_especimenes < max_especimenes);
bool[] nopareto = new bool[max_especimenes];
do
{
    //Generaciones posteriores
    double[] hijo = new double[viga.Length];

    num_descen = 0;
    while (num_descen < max_descen)
    {
        Console.WriteLine(max_descen);
        Console.WriteLine(num_descen);
        //Generacion descendientes
        int a = descen.hijo(max_especimenes, prob_mut, prob_prodigo,
poblacion, nopareto, fitness, rec, Lc, G, Q, ref hijo, rnd);
        for (int i = 0; i < max_especimenes; i++) { nopareto[i] =
false; }

        for (int i = 0; i <= viga.Length - 1; i++)
        {
            descendencia[num_descen, i] = hijo[i];
        }
        //Comprobacion
        Console.WriteLine(hijo[critopt]);
        Console.WriteLine(descendencia[num_descen, critopt]);
        num_descen = num_descen + 1;
    }
}

```



```

        descen.elitismo(max_especimenes, max_descen, ref poblacion,
descendencia, ref nopareto, critopt);
        for (int i = 0; i < max_especimenes; i++) {
Console.WriteLine(poblacion[i, critopt]); }
        //Generacion de descendientes
        //Seleccion
        generacion++;
        Console.WriteLine("Generacion: " + generacion);
    }
    while (generacion < generacion_max);
    //Distancia al origen de coordenadas
    double[] modulo = new double[max_especimenes];
    int optimo = 0;
    for (int i = 0; i < max_especimenes - 1; i++)
    {
        modulo[i] = Math.Sqrt((Math.Pow(poblacion[i, critopt], 2) +
Math.Pow(poblacion[i, 12], 2)));
    }
    //seleccion del Pareto Knee
    for (int i = 0; i < max_especimenes - 2; i++)
    {
        if (modulo[i + 1] < modulo[i]) { optimo = i + 1; }
    }

    //Editar Parametros de Familia OUTPUT
    string strbName =
famInst.Symbol.get_Parameter(bguid).Definition.Name;
    famInst.Symbol.LookupParameter(strbName).Set((poblacion[optimo, 0] +
0.0035) / (1000 * 0.3048));

    string strcName =
famInst.Symbol.get_Parameter(cguid).Definition.Name;
    famInst.Symbol.LookupParameter(strcName).Set((poblacion[optimo, 1] +
0.0035) / (1000 * 0.3048));

    string strHName =
famInst.Symbol.get_Parameter(Hguid).Definition.Name;
    famInst.Symbol.LookupParameter(strHName).Set((100 * poblacion[optimo,
2]) / 0.328);

    string strD1Name =
famInst.Symbol.get_Parameter(D1guid).Definition.Name;
    famInst.Symbol.LookupParameter(strD1Name).Set((poblacion[optimo, 3] +
0.0035) / (1000 * 0.3048));

    string strD2Name =
famInst.Symbol.get_Parameter(D2guid).Definition.Name;
    famInst.Symbol.LookupParameter(strD2Name).Set((poblacion[optimo, 4] +
0.0035) / (1000 * 0.3048));

    string strD3Name =
famInst.Symbol.get_Parameter(D3guid).Definition.Name;
    famInst.Symbol.LookupParameter(strD3Name).Set((poblacion[optimo, 5] +
0.0035) / (1000 * 0.3048));

    string strn1Name =
famInst.Symbol.get_Parameter(n1guid).Definition.Name;
    famInst.Symbol.LookupParameter(strn1Name).Set((poblacion[optimo, 6] +
0.0035) / (1000 * 0.3048));

```

```

        string strn2Name =
famInst.Symbol.get_Parameter(n2guid).Definition.Name;
        famInst.Symbol.LookupParameter(strn2Name).Set((poblacion[optimo, 7] +
0.0035) / (1000 * 0.3048));

        string strsep3Name =
famInst.Symbol.get_Parameter(sep3guid).Definition.Name;
        famInst.Symbol.LookupParameter(strsep3Name).Set((poblacion[optimo, 8]
+ 0.0035) / (1000 * 0.3048));

        string strcosteName =
famInst.Symbol.get_Parameter(costeguid).Definition.Name;
        famInst.Symbol.LookupParameter(strcosteName).Set(poblacion[optimo,
9]);

        string strco2Name =
famInst.Symbol.get_Parameter(co2guid).Definition.Name;

famInst.Symbol.LookupParameter(strco2Name).Set(Math.Round(poblacion[optimo, 10],
2));

        string strenergiaName =
famInst.Symbol.get_Parameter(energiaguid).Definition.Name;

famInst.Symbol.LookupParameter(strenergiaName).Set(Math.Round(poblacion[optimo,
11], 2));

//Control del tiempo
time.Stop();

TaskDialog.Show("Coste Optimizado", "El coste de la viga es: " +
famInst.Symbol.get_Parameter(costeguid).AsString() + " €");
TaskDialog.Show("CO2 Optimizado", "La emisión de CO2 de la viga es: "
+ famInst.Symbol.get_Parameter(co2guid).AsString() + " Kg");
TaskDialog.Show("Coste Energético Optimizado", "El coste energético
de la viga es: " + famInst.Symbol.get_Parameter(energiaguid).AsString() + "
Kwh");
TaskDialog.Show("Nº de barras Optimizado", "El número total de barras
de refuerzo es: " + poblacion[optimo, 12]);
TaskDialog.Show("Tiempo", time.Elapsed.ToString());
    }
    catch (Autodesk.Revit.Exceptions.OperationCanceledException)
    {
        return Result.Cancelled;
    }
    catch (Exception ex)
    {
        message = ex.Message;
        return Result.Failed;
    }
    return Result.Succeeded;
}
}

//Objeto que genera, comprueba y valora las vigas
class Viga
{
    public int generaviga(ref double[] viga, Viga pionero, Random rnd, double
rec, double Lc, double G, double Q)
    {

```

```

    int[] diametro = new int[] { 8, 10, 12, 16, 20, 25 };
    double[] thorm = new double[] { 25, 30, 35, 40, 45, 50, 55, 60, 70, 80,
90, 100 };
    int contador = 0;
    for ( ; ; )
    {
        viga[0] = 5 * (rnd.Next(41, 2410));
        viga[1] = 5 * (rnd.Next(6, 1410));
        viga[2] = thorm[rnd.Next(0, 12)];
        viga[3] = diametro[rnd.Next(0, 6)];
        viga[4] = diametro[rnd.Next(0, 6)];
        viga[5] = diametro[rnd.Next(0, 6)];
        viga[6] = rnd.Next(2, 27);
        viga[7] = rnd.Next(2, 27);
        viga[8] = (rnd.Next(30, (int)viga[1]));
        contador++;
        int p = pionero.comprueba(viga[0], viga[1], viga[2], viga[3],
viga[4], viga[5], viga[6], viga[7], viga[8], rec, G, Q, Lc);
        if (p == 1)
        {
            viga[9] = pionero.valoracoste(viga[0], viga[1], viga[2], viga[3],
viga[4], viga[5], viga[6], viga[7], viga[8], rec, Lc);
            viga[10] = pionero.valoraco2(viga[0], viga[1], viga[2], viga[3],
viga[4], viga[5], viga[6], viga[7], viga[8], rec, Lc);
            viga[11] = pionero.valorenergia(viga[0], viga[1], viga[2],
viga[3], viga[4], viga[5], viga[6], viga[7], viga[8], rec, Lc);
            viga[12] = pionero.valorabarra(viga[3], viga[6], viga[4],
viga[7], viga[5], viga[8], rec, Lc);

                break;
            };
        }
        return contador;
    }
    public int comprueba(double b, double c, double H, double D1, double D2,
double D3, double n1, double n2, double sep3, double rec, double G, double Q,
double Lc)
    {
        //Comprobación
        int check = 1;
        // Comprobaciones geometricas

        //Separación barras inferior
        double linf;
        linf = (b - 2 * rec - 2 * D3 - n1 * D1) / (n1 - 1);
        if (linf <= 20.00 || linf <= D1) { check = 0; return check; }
        // Separación barras superior
        double lsup;
        lsup = (b - 2 * rec - 2 * D3 - n2 * D2) / (n2 - 1);
        if (lsup <= 20 || lsup <= D2) { check = 0; return check; }

        //Cuantías geometricas de armado

        //Armadura de tracción
        double As1, rho1;
        As1 = n1 * ((Math.PI * (Math.Pow(D1, 2))) / 4);
        rho1 = As1 / (b * c);
        if (rho1 < (2.8 / 1000)) { check = 0; return check; }
        //Armadura de compresión
    }

```

```

double As2, rho2;
As2 = n2 * ((Math.PI * (Math.Pow(D2, 2))) / 4);
rho2 = As2 / (b * c);
if (rho2 < (0.3 * (2.8 / 1000))) { check = 0; return check; }

//Cuantías mecánicas de armado

//Cuantía de Flexión
if (As1 < (0.04 * b * c * (1.15 * H / (500 * 1.5)))) { check = 0; return
check; }
//Cuantía de cortante
double A90, fctm;
A90 = (2 * ((Math.PI * (Math.Pow(D3, 2))) / 4)) / sep3;
fctm = 0.3 * (Math.Pow(H, (2.0 / 3.0)));
if (A90 < (fctm * b / (7.5 * 400))) { check = 0; return check; }

//Comprobaciones ELU
//ELU Flexión Simple
double Md;
double lambda = 0.8, epscu = 0.0035;
int eta = 1;
Md = ((1.35 * (b * c * (25 / 1000000) + G) + 1.5 * Q) * (Math.Pow(Lc,
2))) / 8;
if (H > 50) { eta = (1 - (((int)H) - 50) / 200); lambda = 0.8 - (H - 50)
/ 400; epscu = 0.0026 + 0.01455 * (Math.Pow((100 - H / 100), 4)); }
double d = c - rec - D3 - (D1 / 2);
double d1 = rec + D3 + D2 / 2;
double X = 0.00;
double X1, X2;
double ax, bx, cx, square;
ax = b * lambda * eta * (H / 1.5);
bx=(As2*epscu*210000)-(As1*(500/1.15));
cx = -1*(As2 * 210000 * d1);
square = (bx * bx) - (4 * ax * cx);
if(square>0)
{
X1 = (-1*bx + Math.Sqrt((Math.Pow(bx,2))-(4*ax*cx))) / (2 * ax);
X2 = (-1*bx - Math.Sqrt((Math.Pow(bx, 2)) - (4 * ax * cx))) / (2 * ax);
double Xlim = 0.617 * d;
if (X1 <= Xlim && X1 > 0) { X = X1; }
if (X2 <= Xlim && X2 > 0 && X1<X2) { X = X2; }
}
else
{X1 = 0.00; X2 = 0.00;}
double Mu = (b * lambda * X * eta * (H / 1.5) * (d - (lambda * X / 2))) +
(As2 * epscu * 210000 * ((X - d1) / X)*(d-d1));
if (Mu < Md) { check = 0; return check; }
//ELU cortante compresión del hormigón
double Vd1 = ((1.35 * (((b * c * 25) / 1000000) + G)) + (1.5 * Q)) * Lc /
2;
double f1cd = 0.6 * H / 1.5;
if (H > 60) { f1cd = ((0.9 - H / 200) * H / 1.5); if (f1cd > (0.5 * H /
1.5)) { f1cd = 0.5 * H / 1.5; } }
double Vu = f1cd * b * d / 2;
if (Vu < Vd1) { check = 0; return check; }
//Separación barras transversales
double ltrans;
if (Vd1 <= 0.2 * Vu) { ltrans = 0.75 * c; if (sep3 > ltrans || sep3 >
600) { check = 0; return check; } }
if (Vd1 > 0.2 * Vu && Vd1 <= (2 / 3) * Vu) { ltrans = 0.6 * c; if (sep3 >
ltrans || sep3 > 450) { check = 0; return check; } }

```

```

    if (Vd1 > (2 / 3) * Vu) { ltrans = 0.3 * c; if (sep3 > ltrans || sep3 >
300) { check = 0; return check; } }
    //ELU cortante tracción alma--Revisada
    double Vd2 = ((1.35 * (((b * c * 25) / 1000000) + G)) + (1.5 * Q)) * ((Lc
/ 2) - d);
    Vu = 0.1 * (1 + (Math.Pow((200 / d), (1.0 / 2.0)))) * (Math.Pow((100 *
rho1 * H), (1.0 / 3.0))) * b * d;
    Vu = Vu + (0.9 * d * A90 * 400); //OK
    if (Vu < Vd2) { check = 0; return check; }

    //ELS deformacion
    double Ec = 8500 * (Math.Pow(H + 8, (1.0 / 3.0)));
    double Ib = (b * Math.Pow(c, 3) / 12);
    double f = 5 * Q * Math.Pow(Lc, 4) / (384 * Ec * Ib);
    if (0.8 * f > Lc / 1000) { check = 0; return check; }

    //ELS Fisuracion
    double X = 0.00;
    double Mf = (b * c * (25 / 1000000) + G + 0.2 * Q) * Math.Pow(Lc, 2) / 8;
    double sm = 2 * (rec + D3 + D1 / 2) + 0.2 * linf + 0.05 * D1 * b *
Math.Max(rec + D3 + 8 * D1, c / 2) / As1;
    rho1 = As1 / (b * d);
    rho2 = As2 / (b * d);
    double nc = 200000 / Ec;
    X = 1 + (2 * (1 + rho2 * d1 / rho1 * d) / (nc * rho1 * Math.Pow((1 + rho2
/ rho1), 2)));
    X = nc * rho1 * (1 + rho2 / rho1) * (-1 * +Math.Pow(X, (1 / 2)));
    X = d * X;
    double If = nc * As1 * (d - X) * (d - X / 3) + nc * As2 * (X - d1) * (X /
3 - d1);
    double sigma_s = nc * Mf * (d - X) / If;
    double fctmfl = Math.Max((1.6 - (c / 1000) * fctm), fctm);
    double Ih = Ib + nc * (As1 * Math.Pow((d - c / 2), 2)) + As2 *
Math.Pow((c / 2 - d1), 2);
    double Mfis = fctmfl * Ih / (c / 2);
    double sigma_sr = nc * Mfis * (d - X) / If;
    double wk = 0;
    if (Mfis < Mf)
    {
        double eps_sm = Math.Max(sigma_s / 200000 * (1 - Math.Pow((sigma_sr /
sigma_s), 2)), 0.4 * sigma_s / 200000);
        wk = 1.7 * sm * eps_sm;
    }
    if (wk >= 0.3) { check = 0; return check; }

    return check;
}
public double valoracoste(double b, double c, double H, double D1, double D2,
double D3, double n1, double n2, double sep3, double rec, double Lc)
{
    //double var = 1.05;
    //Coste Económico
    double[] PRECIOSH = { 97.68, 101.89, 103.51, 106.21, 109.46, 114.45,
116.36, 118.75, 121.86, 128.02, 130.82, 130.85 };
    double PRECIOSA = 1.29;
    double PRECIOSENC = 31.41;

```

```

        //Precio
        //Coste del acero-->area*n barras*lon*coste
        double costea;
        costea = (D1 * n1 * (Lc - 2 * rec - D1 + 300) / 1000); //Armadura inferior
        costea = costea + (D2 * n2 * (Lc - 2 * rec - D2 + 300) /
1000); //Armadura superior
        costea = costea + (D3 * ((Math.Round((Lc - 2 * rec) / sep3, 0)) + 1) *
((2 * (b + c) - 8 * rec + 100)) / 1000); //Armadura cortante
        costea = costea * PRECIOSA; //Precio Total
        //Coste hormigon-->Volumen*coste
        int H1;
        if (H <= 55) { H1 = ((int)H / 5) - 5; } else { H1 = ((int)H / 10) + 1; }
        double costeh;
        costeh = (Lc * b * c / (1000000000)) * PRECIOSH[H1];
        //Coste encofrado-->area enc*coste
        double costenc;
        costenc = ((2 * (Lc * c + b * c) + Lc * b) / 1000000) * PRECIOSENC;
        //Coste total
        double coste = costea + costeh + costenc;

        return coste;
    }
    public double valoraco2(double b, double c, double H, double D1, double D2,
double D3, double n1, double n2, double sep3, double rec, double Lc)
    {
        //Emisión C02
        double[] C02H = { 335.63, 366.53, 394.6, 417.75, 448.49, 501.01, 561.56,
605.36, 618.97, 630.41, 653.93, 686.87 };
        double C02A = 3.03;
        double C02ENC = 2.97;

        //C02
        //C02 del acero-->area*n barras*lon*co2
        double co2a;
        co2a = (D1 * n1 * (Lc - 2 * rec - D1 + 300) / 1000); //Armadura inferior
        co2a = co2a + (D2 * n2 * (Lc - 2 * rec - D2 + 300) / 1000); //Armadura
superior
        co2a = co2a + (D3 * ((Math.Round((Lc - 2 * rec) / sep3, 0)) + 1) * ((2 *
(b + c) - 8 * rec + 100)) / 1000); //Armadura cortante
        co2a = co2a * C02A; //Precio Total
        //C02 hormigon-->Volumen*co2
        int H1;
        if (H <= 55) { H1 = ((int)H / 5) - 5; } else { H1 = ((int)H / 10) + 1; }
        double co2h;
        co2h = (Lc * b * c / (1000000000)) * C02H[H1];
        //C02 encofrado-->area enc*co2
        double co2enc;
        co2enc = ((2 * (Lc * c + b * c) + Lc * b) / 1000000) * C02ENC;
        //C02 total
        double co2 = co2a + co2h + co2enc;
        return co2;
    }
    public double valoraenergia(double b, double c, double H, double D1, double
D2, double D3, double n1, double n2, double sep3, double rec, double Lc)
    {
        //Coste Energético
        double[] EH = { 414.59, 440.74, 458.02, 477.5, 505.73, 551.07, 622.30,
658.58, 670.01, 667.12, 688.14, 723.72 };
        double EA = 10.44;
        double EENC = 10.97;
        //Energía

```

```

    //Coste energético del acero-->area*n barras*lon*Kwh
    double ea;
    ea = (D1 * n1 * (Lc - 2 * rec - D1 + 300) / 1000); //Armadura inferior
    ea = ea + (D2 * n2 * (Lc - 2 * rec - D2 + 300) / 1000); //Armadura
superior
    ea = ea + (D3 * ((Math.Round((Lc - 2 * rec) / sep3, 0)) + 1) * ((2 * (b +
c) - 8 * rec + 100)) / 1000); //Armadura cortante
    ea = ea * EA; //Precio Total
    //Coste energético hormigon-->Volumen*Kwh
    int H1;
    if (H <= 55) { H1 = ((int)H / 5) - 5; } else { H1 = ((int)H / 10) + 1; }
    double eh;
    eh = (Lc * b * c / (100000000)) * EH[H1];
    //Coste energético encofrado-->area enc*Kwh
    double eenc;
    eenc = ((2 * (Lc * c + b * c) + Lc * b) / 1000000) * EENC;
    //Coste total
    double e = ea + eh + eenc;
    return e;
}
public double valorabarra(double D1, double n1, double D2, double n2, double
D3, double sep3, double rec, double Lc)
{
    //Suma de la cantidad de barras
    double b = (n1) + (n2) + (((Math.Round((Lc - 2 * rec) / sep3, 0)) + 1));
    return b;
}
}

//Selección de progenitores, genes, hijos prodigos y mutación
class Descendiente
{
    public int elcpadre(int max_especimenes, Random rnd)
    {
        int padre = 0;
        padre = rnd.Next(0, max_especimenes - 1);
        return padre;
    }

    public int elecmadre(int max_especimenes, int padre, Random rnd)
    {
        int madre = 0;
        madre = rnd.Next(0, max_especimenes - 1);
        //Prohibición incesto
        while (madre == padre) { madre = rnd.Next(0, max_especimenes - 1); };
        return madre;
    }

    public int ruleta(Random rnd)
    {
        int ruleta = 1 + rnd.Next(0, 6);
        return ruleta;
    }

    public int ruleta2(int rul1, Random rnd)
    {
        int ruleta2 = 1 + rnd.Next(0, 7);
        while (ruleta2 <= rul1) { ruleta2 = 1 + rnd.Next(0, 7); }
        return ruleta2;
    }

    public int mutacion(Random rnd)
    {

```

```

        int mut = rnd.Next(0, 99);
        return mut;
    }
    public double selecmutagen(int i, Random rnd, double rec, double Lc, double
G, double Q)
    {
        Viga mutado = new Viga();
        double[] mutante = new double[13];
        mutado.generaviga(ref mutante, mutado, rnd, rec, Lc, G, Q);
        return mutante[i];
    }

    public int prodigo(Random rnd)
    {
        int prodigo = rnd.Next(0, 99);
        return prodigo;
    }
}
//Generación de descendiente y selección de los especimenes
class Generacion
{
    public int config()
    {
        return 0;
    }

    public int hijo(int max_especimenes, int prob_mut, int prob_prodigo,
double[,] poblacion, double rec, int Lc, int G, int Q, ref double[] hijo, Random
rnd)
    {
        int check = 0;
        int numhijos = 0;
        Descendiente padres = new Descendiente();
        Viga viguita = new Viga();
        int prodigo = padres.prodigo(rnd);
        for (; ; )
        {
            int rul1 = padres.ruleta(rnd);
            int rul2 = padres.ruleta2(rul1, rnd);
            int padre = padres.elcpadre(max_especimenes, rnd);
            int madre = padres.elecmadre(max_especimenes, padre, rnd);
            Console.WriteLine("Padre: " + padre + " Madre: " + madre);
            Console.WriteLine("Rul1: " + rul1 + " Rul2: " + rul2);
            for (int i = 0; i <= 8; i++)
            {
                if (i < rul1) { hijo[i] = poblacion[padre, i];}
                if (i < rul2 && i >= rul1) { hijo[i] = poblacion[madre, i];}
                if (i <= 8 && i >= rul2 && rul2 != 8) { hijo[i] =
poblacion[padre, i]; }
                //Mutacion de uno de los genes
                int mut = padres.mutacion(rnd);
                if (mut <= prob_mut) { hijo[i] = padres.selecmutagen2(i, rnd,
rec, Lc, G, Q);}
            }
            check = viguita.comprueba(hijo[0], hijo[1], hijo[2], hijo[3],
hijo[4], hijo[5], hijo[6], hijo[7], hijo[8], rec, G, Q, Lc);
            numhijos++;
            Console.WriteLine("Numero de hijos: " + numhijos);
            Console.WriteLine("Check: " + check);
            if (check == 1)
            {

```



```

        hijo[9] = viguita.valoracoste(hijo[0], hijo[1], hijo[2],
hijo[3], hijo[4], hijo[5], hijo[6], hijo[7], hijo[8], rec, Lc);
        hijo[10] = viguita.valoraco2(hijo[0], hijo[1], hijo[2],
hijo[3], hijo[4], hijo[5], hijo[6], hijo[7], hijo[8], rec, Lc);
        hijo[11] = viguita.valoraenergia(hijo[0], hijo[1], hijo[2],
hijo[3], hijo[4], hijo[5], hijo[6], hijo[7], hijo[8], rec, Lc);
        hijo[12] = viguita.valorabarra(hijo[3], hijo[6], hijo[4],
hijo[7], hijo[5], hijo[8], rec, Lc);
        break;
    };
}
check = 0;
if (prodigo < probab_prodigo)
{
    for (; ; )
    {
        viguita.generaviga(ref hijo, viguita, rnd, rec, Lc, G, Q);
hijo[4], hijo[5], hijo[6], hijo[7], hijo[8], rec, G, Q, Lc);
        check = viguita.comprueba(hijo[0], hijo[1], hijo[2], hijo[3],
        if (check == 1)
        {
            hijo[9] = viguita.valoracoste(hijo[0], hijo[1], hijo[2],
hijo[3], hijo[4], hijo[5], hijo[6], hijo[7], hijo[8], rec, Lc);
            hijo[10] = viguita.valoraco2(hijo[0], hijo[1], hijo[2],
hijo[3], hijo[4], hijo[5], hijo[6], hijo[7], hijo[8], rec, Lc);
            hijo[11] = viguita.valoraenergia(hijo[0], hijo[1],
hijo[2], hijo[3], hijo[4], hijo[5], hijo[6], hijo[7], hijo[8], rec, Lc);
            hijo[12] = viguita.valorabarra(hijo[3], hijo[6], hijo[4],
hijo[7], hijo[5], hijo[8], rec, Lc);
            break;
        }
    }
}
return 1;
}

public int sort(ref double[,] generacion, int critord)
{
    //Bubble method en referencia al coste de menor a mayor
    double[] temp = new double[generacion.GetLength(0)];
    for (int i = 0; i < generacion.GetLength(0); i++)
    {
        for (int sort = 0; sort < generacion.GetLength(0) - 1; sort++)
        {
            if (generacion[sort, critord] > generacion[sort + 1, critord])
            {
                for (int t = 0; t <= 12; t++) { temp[t] = generacion[sort +
1, t]; };
                for (int s = 0; s <= 12; s++) { generacion[sort + 1, s] =
generacion[sort, s]; };
                for (int t = 0; t <= 12; t++) { generacion[sort, t] =
temp[t]; };
            }
        }
    }
    return 0;
}

public int elitismo(int max_especimenes, int max_descen, ref double[,]
poblacion, double[,] descendencia, ref bool[] pareto, int critord)
{

```

```
        double[,] generacion = new double[max_especimenes + max_descen,
poblacion.GetLength(1)];
        Generacion descendencias = new Generacion();
        for (int i = 0; i < max_especimenes; i++)
        {
            for (int j = 0; j <= 12; j++)
            {
                generacion[i, j] = poblacion[i, j];
            }
        }
        for (int i = max_especimenes; i < poblacion.GetLength(0) +
descendencia.GetLength(0); i++)
        {
            for (int j = 0; j <= 12; j++)
            {
                generacion[i, j] = descendencia[i - max_especimenes, j];
            }
        }
        descendencias.sort(ref generacion, critord);
        for (int i = 0; i < max_especimenes; i++)
        {
            for (int j = 0; j <= 12; j++)
            {
                poblacion[i, j] = generacion[i, j];
            }
        }

        for (int i = 0; i < poblacion.GetLength(0) - 1; i++)
        {
            if (poblacion[i + 1, 12] > poblacion[i, 12]) { pareto[i] = true; }
        }

        return 0;
    }
}
```

## ANEXO 2: CÓDIGO MULTICRITERIO

Tras introducir al código del Anexo 1 las modificaciones explicadas en el apartado 2.3 se consigue desarrollar el objetivo final del documento, la generación de un plugin de optimización estructural multicriterio integrado en un entorno BIM utilizando un algoritmo genético. Este anexo contiene este código creado a partir de los planteamientos desarrollados en el apartado 2.

Durante el código se han realizado anotaciones explicativas que explican de un modo resumido y condensado el objetivo de las siguientes líneas de programación, así como algunos de los planteamientos. Estas anotaciones se han realizado siguiendo las normas del lenguaje de programación C#, por lo que no suponen ninguna traba en la ejecución del algoritmo.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Diagnostics;
using Autodesk.Revit.DB;
using Autodesk.Revit.DB.Architecture;
using Autodesk.Revit.UI;
using Autodesk.Revit.UI.Selection;
using Autodesk.Revit.ApplicationServices;
using Autodesk.Revit.Attributes;

[TransactionAttribute(TransactionMode.Automatic)]
[RegenerationAttribute(RegenerationOption.Manual)]
public class Plugin2 : IExternalCommand
{
    public Result Execute(ExternalCommandData commandData, ref string message,
        ElementSet elements)
    {
        //Connexión con la API y el documento abierto
        UIApplication uiApp = commandData.Application;
        UIDocument uidoc = uiApp.ActiveUIDocument;
        Application app = uiApp.Application;
        Document doc = uiApp.ActiveUIDocument.Document;
        //El try-catch se utiliza para manejar errores
        try
        {
            //Selección de elemento a optimizar
            Reference pickedRef = null;
            Selection sel = uidoc.Selection;
            pickedRef = sel.PickObject(ObjectType.Element, "Selecciona un
elemento");
            if (pickedRef != null)
            {}
            TaskDialog.Show("Revit", "Se ha añadido un elemento a la
seleccion");
            Element ele = doc.GetElement(pickedRef);
            FamilyInstance famInst = ele as FamilyInstance;
            //Control del tiempo
            Stopwatch time = Stopwatch.StartNew();
            time.Start();
            //Parametros extraídos y editados
            //Shared Parameters del archivo "ConcreteBeamOptSP"
```

```

Guid D1guid = new Guid("1e5d8bc3-bc41-4e9b-9df9-70a68af1e2f3");
Guid D2guid = new Guid("178a72dc-8d91-4028-9d2e-b64b4ae587d6");
Guid D3guid = new Guid("b0388a29-e487-4118-bbfa-c1c132fd9dfc");
Guid n1guid = new Guid("94878554-76a4-408b-9b9d-3c12eba25dc5");
Guid n2guid = new Guid("ad32a0dc-c1ad-4bb0-bb45-a02a5c6c7b03");
Guid sep3guid = new Guid("31438e3e-0ab7-4592-8df3-9a8515f0c7b6");
Guid Hguid = new Guid("f942ad4d-759b-428e-ba47-ffa568fbc1");
Guid bguid = new Guid("a1540d7b-dab0-482f-996b-9859e39f6e8e");
Guid cguid = new Guid("f8a30d8c-8c1d-4cd7-aa3b-1c6e98d29f1b");
Guid costeguid = new Guid("c5af969d-a46a-4ef7-9f81-11d76f584826");
Guid co2guid = new Guid("0520e6b5-9769-4158-9680-d2dbce487bcc");
Guid energiaguid = new Guid("bb2f3cf8-6a53-4873-b87f-7134946be99a");
Guid Gguid = new Guid("a118fd92-6434-49d3-ac1d-9517936a1f1a");
Guid Qguid = new Guid("f0062670-1e84-4c0f-91c8-88192c58d68d");
Guid recguid = new Guid("af389967-2771-4544-a42d-8a15f368eee7");
Guid espguid = new Guid("9e37c064-6dec-48e6-9aa7-84fc1ccb41d9");
Guid descenguid = new Guid("1ccb90d9-f967-44ba-aa0c-21547fee4d67");
Guid genguid = new Guid("93b6cc37-1074-4292-ad86-e119159a2323");
Guid mutguid = new Guid("aaed5b32-e805-401e-8632-3a64e1178067");
Guid prodguid = new Guid("d3b8c39c-2dcd-42c0-8ccf-f8887fecf501");
Guid fitnessguid = new Guid("24fa46d7-c686-4688-bc7b-2f1d8e0f6334");

//Parámetros asignados desde Revit
double Lc = 1000 *
(Math.Round((famInst.get_Parameter(BuiltInParameter.INSTANCE_LENGTH_PARAM).AsDouble() / 3.2808, 2)); //Luz de calculo en mm
double rec =
Math.Round((((famInst.Symbol.get_Parameter(recguid).AsDouble()) / 0.0033)),
0); //Recubrimiento nominal en mm
double G = (famInst.Symbol.get_Parameter(Gguid).AsDouble()) /
1000; //Carga gravitatoria total en N/mm
double Q = (famInst.Symbol.get_Parameter(Qguid).AsDouble()) /
1000; //Carga Variable en N/mm
int max_especimenes =
(famInst.Symbol.get_Parameter(espguid).AsInteger()); //Numero maximo de
especimenes conservados en cada generacion
int generacion_max =
famInst.Symbol.get_Parameter(genguid).AsInteger(); //Numero de generaciones
int max_descen =
famInst.Symbol.get_Parameter(descenguid).AsInteger(); //Numero maximo de
descendientes generados
int prob_mut = famInst.Symbol.get_Parameter(mutguid).AsInteger();
//Probabilidad que se produzca una mutacion en %
int prob_prodigio =
famInst.Symbol.get_Parameter(prodguid).AsInteger(); //Probabilidad de generacion
de un hio prodigo en %
int fitness =
famInst.Symbol.get_Parameter(fitnessguid).AsInteger(); //Probabilidad de que los
progenitores sean soluciones de Pareto

int num_especimenes = 0;
int generacion = 0;
int num_descen = 0;
char crit = 'p'; //Criterio de ordenación p-precio/c-co2/e-coste
energético
int critopt = 0;
int v = 0;
double[] viga = new double[13];
double[,] poblacion = new double[max_especimenes, viga.Length];
double[,] descendencia = new double[max_descen, viga.Length];
Random rnd = new Random();

```

```

Generacion descen = new Generacion();
int config = descen.config();
if (crit == 'p') { critopt = 9; }
if (crit == 'c') { critopt = 10; }
if (crit == 'e') { critopt = 11; }
if (crit == 'b') { critopt = 12; }
do
{
    /*Orden del codigo de la viga:
    b(ancho), c(canto), H(tipo de hormigon), D1(diametro inferior),
D2(diametro superior), D3(diametro estribo), n1(nº barras D1), n2(nº barras D2),
sep3(separacion entre estribos)*/
    //Generación especímenes iniciales
    Viga pionero = new Viga();
    v = pionero.generaviga(ref viga, pionero, rnd, rec, Lc, G, Q);
    for (int i = 0; i < viga.GetLength(0); i++)
    {
        poblacion[num_especimenes, i] = viga[i];
        Console.WriteLine(viga[i]);
    }

    num_especimenes++;
}
while (num_especimenes < max_especimenes);
bool[] nopareto = new bool[max_especimenes];
do
{
    //Generaciones posteriores
    double[] hijo = new double[viga.Length];

    num_descen = 0;
    while (num_descen < max_descen)
    {
        Console.WriteLine(max_descen);
        Console.WriteLine(num_descen);
        //Generacion descendientes
        int a = descen.hijo(max_especimenes, prob_mut, prob_prodigo,
poblacion, nopareto, fitness, rec, Lc, G, Q, ref hijo, rnd);
        for (int i = 0; i < max_especimenes; i++) { nopareto[i] =
false; }

        for (int i = 0; i <= viga.Length - 1; i++)
        {
            descendencia[num_descen, i] = hijo[i];
        }
        //Comprobacion
        Console.WriteLine(hijo[critopt]);
        Console.WriteLine(descendencia[num_descen, critopt]);
        num_descen = num_descen + 1;
    }
    descen.elitismo(max_especimenes, max_descen, ref poblacion,
descendencia, ref nopareto, critopt);
    for (int i = 0; i < max_especimenes; i++) {
Console.WriteLine(poblacion[i, critopt]); }
    //Generacion de descendientes
    //Selección
    generacion++;
    Console.WriteLine("Generacion: " + generacion);
}
while (generacion < generacion_max);
//Distancia al origen de coordenadas
double[] modulo = new double[max_especimenes];

```

```

        int optimo = 0;
        for (int i = 0; i < max_especimenes - 1; i++)
        {
            modulo[i] = Math.Sqrt((Math.Pow(poblacion[i, critopt], 2) +
Math.Pow(poblacion[i, 12], 2)));
        }
        //seleccion del Pareto Knee
        for (int i = 0; i < max_especimenes - 2; i++)
        {
            if (modulo[i + 1] < modulo[i]) { optimo = i + 1; }
        }

        //Editar Parametros de Familia OUTPUT
        string strbName =
famInst.Symbol.get_Parameter(bguid).Definition.Name;
        famInst.Symbol.LookupParameter(strbName).Set((poblacion[optimo, 0] +
0.0035) / (1000 * 0.3048));

        string strcName =
famInst.Symbol.get_Parameter(cguid).Definition.Name;
        famInst.Symbol.LookupParameter(strcName).Set((poblacion[optimo, 1] +
0.0035) / (1000 * 0.3048));

        string strHName =
famInst.Symbol.get_Parameter(Hguid).Definition.Name;
        famInst.Symbol.LookupParameter(strHName).Set((100 * poblacion[optimo,
2]) / 0.328);

        string strD1Name =
famInst.Symbol.get_Parameter(D1guid).Definition.Name;
        famInst.Symbol.LookupParameter(strD1Name).Set((poblacion[optimo, 3] +
0.0035) / (1000 * 0.3048));

        string strD2Name =
famInst.Symbol.get_Parameter(D2guid).Definition.Name;
        famInst.Symbol.LookupParameter(strD2Name).Set((poblacion[optimo, 4] +
0.0035) / (1000 * 0.3048));

        string strD3Name =
famInst.Symbol.get_Parameter(D3guid).Definition.Name;
        famInst.Symbol.LookupParameter(strD3Name).Set((poblacion[optimo, 5] +
0.0035) / (1000 * 0.3048));

        string strn1Name =
famInst.Symbol.get_Parameter(n1guid).Definition.Name;
        famInst.Symbol.LookupParameter(strn1Name).Set((poblacion[optimo, 6] +
0.0035) / (1000 * 0.3048));

        string strn2Name =
famInst.Symbol.get_Parameter(n2guid).Definition.Name;
        famInst.Symbol.LookupParameter(strn2Name).Set((poblacion[optimo, 7] +
0.0035) / (1000 * 0.3048));

        string strsep3Name =
famInst.Symbol.get_Parameter(sep3guid).Definition.Name;
        famInst.Symbol.LookupParameter(strsep3Name).Set((poblacion[optimo, 8]
+ 0.0035) / (1000 * 0.3048));

        string strcosteName =
famInst.Symbol.get_Parameter(costeguid).Definition.Name;
        famInst.Symbol.LookupParameter(strcosteName).Set(poblacion[optimo,
9]);

```

```

        string strco2Name =
famInst.Symbol.get_Parameter(co2guid).Definition.Name;

famInst.Symbol.LookupParameter(strco2Name).Set(Math.Round(poblacion[optimo, 10],
2));

        string strenergiaName =
famInst.Symbol.get_Parameter(energiaguid).Definition.Name;

famInst.Symbol.LookupParameter(strenergiaName).Set(Math.Round(poblacion[optimo,
11], 2));

        //Control del tiempo
        time.Stop();

        TaskDialog.Show("Coste Optimizado", "El coste de la viga es: " +
famInst.Symbol.get_Parameter(costeguid).AsValueString() + " €");
        TaskDialog.Show("CO2 Optimizado", "La emisión de CO2 de la viga es: "
+ famInst.Symbol.get_Parameter(co2guid).AsValueString() + " Kg");
        TaskDialog.Show("Coste Energético Optimizado", "El coste energético
de la viga es: " + famInst.Symbol.get_Parameter(energiaguid).AsValueString() + "
Kwh");
        TaskDialog.Show("Nº de barras Optimizado", "El número total de barras
de refuerzo es: " + poblacion[optimo, 12]);
        TaskDialog.Show("Tiempo", time.Elapsed.ToString());
    }
    catch (Autodesk.Revit.Exceptions.OperationCanceledException)
    {
        return Result.Cancelled;
    }
    catch (Exception ex)
    {
        message = ex.Message;
        return Result.Failed;
    }
    return Result.Succeeded;
}
}

//Objeto que genera, comprueba y valora las vigas
class Viga
{
    public int generaviga(ref double[] viga, Viga pionero, Random rnd, double
rec, double Lc, double G, double Q)
    {
        int[] diametro = new int[] { 8, 10, 12, 16, 20, 25 };
        double[] thorm = new double[] { 25, 30, 35, 40, 45, 50, 55, 60, 70, 80,
90, 100 };
        int contador = 0;
        for ( ; ; )
        {
            viga[0] = 5 * (rnd.Next(41, 2410));
            viga[1] = 5 * (rnd.Next(6, 1410));
            viga[2] = thorm[rnd.Next(0, 12)];
            viga[3] = diametro[rnd.Next(0, 6)];
            viga[4] = diametro[rnd.Next(0, 6)];
            viga[5] = diametro[rnd.Next(0, 6)];
            viga[6] = rnd.Next(2, 27);
            viga[7] = rnd.Next(2, 27);
            viga[8] = (rnd.Next(30, (int)viga[1]));

```

```

        contador++;
        int p = pionero.comprueba(viga[0], viga[1], viga[2], viga[3],
viga[4], viga[5], viga[6], viga[7], viga[8], rec, G, Q, Lc);
        if (p == 1)
        {
            viga[9] = pionero.valoracoste(viga[0], viga[1], viga[2], viga[3],
viga[4], viga[5], viga[6], viga[7], viga[8], rec, Lc);
            viga[10] = pionero.valoraco2(viga[0], viga[1], viga[2], viga[3],
viga[4], viga[5], viga[6], viga[7], viga[8], rec, Lc);
            viga[11] = pionero.valorenergia(viga[0], viga[1], viga[2],
viga[3], viga[4], viga[5], viga[6], viga[7], viga[8], rec, Lc);
            viga[12] = pionero.valorabarra(viga[3], viga[6], viga[4],
viga[7], viga[5], viga[8], rec, Lc);

            break;
        }
    }
    return contador;
}
public int comprueba(double b, double c, double H, double D1, double D2,
double D3, double n1, double n2, double sep3, double rec, double G, double Q,
double Lc)
{
    //Comprobación
    int check = 1;
    // Comprobaciones geometricas

    //Separación barras inferior
    double linf;
    linf = (b - 2 * rec - 2 * D3 - n1 * D1) / (n1 - 1);
    if (linf <= 20.00 || linf <= D1) { check = 0; return check; }
    // Separación barras superior
    double lsup;
    lsup = (b - 2 * rec - 2 * D3 - n2 * D2) / (n2 - 1);
    if (lsup <= 20 || lsup <= D2) { check = 0; return check; }

    //Cuantías geometricas de armado

    //Armadura de tracción
    double As1, rho1;
    As1 = n1 * ((Math.PI * (Math.Pow(D1, 2))) / 4);
    rho1 = As1 / (b * c);
    if (rho1 < (2.8 / 1000)) { check = 0; return check; }
    //Armadura de compresión
    double As2, rho2;
    As2 = n2 * ((Math.PI * (Math.Pow(D2, 2))) / 4);
    rho2 = As2 / (b * c);
    if (rho2 < (0.3 * (2.8 / 1000))) { check = 0; return check; }

    //Cuantias mecanicas de armado

    //Cuantia de Flexión
    if (As1 < (0.04 * b * c * (1.15 * H / (500 * 1.5)))) { check = 0; return
check; }
    //Cuantia de cortante
    double A90, fctm;
    A90 = (2 * ((Math.PI * (Math.Pow(D3, 2))) / 4)) / sep3;
    fctm = 0.3 * (Math.Pow(H, (2.0 / 3.0)));
    if (A90 < (fctm * b / (7.5 * 400))) { check = 0; return check; }
}

```



```

//Comprobaciones ELU
//ELU Flexión Simple
double Md;
double lambda = 0.8, epscu = 0.0035;
int eta = 1;
Md = ((1.35 * (b * c * (25 / 1000000) + G) + 1.5 * Q) * (Math.Pow(Lc,
2))) / 8;
if (H > 50) { eta = (1 - (((int)H) - 50) / 200); lambda = 0.8 - (H - 50)
/ 400; epscu = 0.0026 + 0.01455 * (Math.Pow((100 - H / 100), 4)); }
double d = c - rec - D3 - (D1 / 2);
double d1 = rec + D3 + D2 / 2;
double X = 0.00;
double X1, X2;
double ax, bx, cx, square;
ax = b * lambda * eta * (H / 1.5);
bx=(As2*epscu*210000)-(As1*(500/1.15));
cx = -1*(As2 * 210000 * d1);
square = (bx * bx) - (4 * ax * cx);
if(square>0)
{
X1 = (-1*bx + Math.Sqrt((Math.Pow(bx,2))-(4*ax*cx))) / (2 * ax);
X2 = (-1*bx - Math.Sqrt((Math.Pow(bx, 2)) - (4 * ax * cx))) / (2 * ax);
double Xlim = 0.617 * d;
if (X1 <= Xlim && X1 > 0) { X = X1; }
if (X2 <= Xlim && X2 > 0 && X1<X2) { X = X2; }
}
else
{X1 = 0.00; X2 = 0.00;}
double Mu = (b * lambda * X * eta * (H / 1.5) * (d - (lambda * X / 2))) +
(As2 * epscu * 210000 * ((X - d1) / X)*(d-d1));
if (Mu < Md) { check = 0; return check; }
//ELU cortante compresión del hormigón
double Vd1 = ((1.35 * ((b * c * 25) / 1000000) + G)) + (1.5 * Q)) * Lc /
2;
double f1cd = 0.6 * H / 1.5;
if (H > 60) { f1cd = ((0.9 - H / 200) * H / 1.5); if (f1cd > (0.5 * H /
1.5)) { f1cd = 0.5 * H / 1.5; } }
double Vu = f1cd * b * d / 2;
if (Vu < Vd1) { check = 0; return check; }
//Separación barras transversales
double ltrans;
if (Vd1 <= 0.2 * Vu) { ltrans = 0.75 * c; if (sep3 > ltrans || sep3 >
600) { check = 0; return check; } }
if (Vd1 > 0.2 * Vu && Vd1 <= (2 / 3) * Vu) { ltrans = 0.6 * c; if (sep3 >
ltrans || sep3 > 450) { check = 0; return check; } }
if (Vd1 > (2 / 3) * Vu) { ltrans = 0.3 * c; if (sep3 > ltrans || sep3 >
300) { check = 0; return check; } }

//ELU cortante tracción alma
double Vd2 = ((1.35 * (((b * c * 25) / 1000000) + G)) + (1.5 * Q)) * ((Lc
/ 2) - d);
Vu = 0.1 * (1 + (Math.Pow((200 / d), (1.0 / 2.0)))) * (Math.Pow((100 *
rho1 * H), (1.0 / 3.0))) * b * d;
Vu = Vu + (0.9 * d * A90 * 400);
if (Vu < Vd2) { check = 0; return check; }

```

```

//ELS deformacion
double Ec = 8500 * (Math.Pow(H + 8, (1.0 / 3.0)));
double Ib = (b * Math.Pow(c, 3) / 12);
double f = 5 * Q * Math.Pow(Lc, 4) / (384 * Ec * Ib);
if (0.8 * f > Lc / 1000) { check = 0; return check; }

//ELS Fisuracion
double X = 0.00;
double Mf = (b * c * (25 / 1000000) + G + 0.2 * Q) * Math.Pow(Lc, 2) / 8;
double sm = 2 * (rec + D3 + D1 / 2) + 0.2 * linf + 0.05 * D1 * b *
Math.Max(rec + D3 + 8 * D1, c / 2) / As1;
rho1 = As1 / (b * d);
rho2 = As2 / (b * d);
double nc = 200000 / Ec;
X = 1 + (2 * (1 + rho2 * d1 / rho1 * d) / (nc * rho1 * Math.Pow((1 + rho2
/ rho1), 2)));
X = nc * rho1 * (1 + rho2 / rho1) * (-1 * Math.Pow(X, (1 / 2)));
X = d * X;
double If = nc * As1 * (d - X) * (d - X / 3) + nc * As2 * (X - d1) * (X /
3 - d1);
double sigma_s = nc * Mf * (d - X) / If;
double fctmfl = Math.Max((1.6 - (c / 1000) * fctm), fctm);
double Ih = Ib + nc * (As1 * Math.Pow((d - c / 2), 2) + As2 *
Math.Pow((c / 2 - d1), 2));
double Mfis = fctmfl * Ih / (c / 2);
double sigma_sr = nc * Mfis * (d - X) / If;
double wk = 0;
if (Mfis < Mf)
{
    double eps_sm = Math.Max(sigma_s / 200000 * (1 - Math.Pow((sigma_sr /
sigma_s), 2)), 0.4 * sigma_s / 200000);
    wk = 1.7 * sm * eps_sm;
}
if (wk >= 0.3) { check = 0; return check; }

return check;
}
public double valoracoste(double b, double c, double H, double D1, double D2,
double D3, double n1, double n2, double sep3, double rec, double Lc)
{
    //double var = 1.05;
    //Coste Económico
    double[] PRECIOSH = { 97.68, 101.89, 103.51, 106.21, 109.46, 114.45,
116.36, 118.75, 121.86, 128.02, 130.82, 130.85 };
    double PRECIOSA = 1.29;
    double PRECIOSENC = 31.41;

    //Precio
    //Coste del acero-->area*n barras*lon*coste
    double costea;
    costea = (D1 * n1 * (Lc - 2 * rec - D1 + 300) / 1000); //Armadura inferior
    costea = costea + (D2 * n2 * (Lc - 2 * rec - D2 + 300) /
1000); //Armadura superior
    costea = costea + (D3 * ((Math.Round((Lc - 2 * rec) / sep3, 0)) + 1) *
((2 * (b + c) - 8 * rec + 100) / 1000)); //Armadura cortante

```

```

    costea = costea * PRECIOSA; //Precio Total
    //Coste hormigon-->Volumen*coste
    int H1;
    if (H <= 55) { H1 = ((int)H / 5) - 5; } else { H1 = ((int)H / 10) + 1; }
    double costeh;
    costeh = (Lc * b * c / (1000000000)) * PRECIOSH[H1];
    //Coste encofrado-->area enc*coste
    double costenc;
    costenc = ((2 * (Lc * c + b * c) + Lc * b) / 1000000) * PRECIOSENC;
    //Coste total
    double coste = costea + costeh + costenc;

    return coste;
}
public double valoraco2(double b, double c, double H, double D1, double D2,
double D3, double n1, double n2, double sep3, double rec, double Lc)
{
    //Emisión C02
    double[] CO2H = { 335.63, 366.53, 394.6, 417.75, 448.49, 501.01, 561.56,
605.36, 618.97, 630.41, 653.93, 686.87 };
    double CO2A = 3.03;
    double CO2ENC = 2.97;

    //C02
    //C02 del acero-->area*n barras*lon*co2
    double co2a;
    co2a = (D1 * n1 * (Lc - 2 * rec - D1 + 300) / 1000); //Armadura inferior
    co2a = co2a + (D2 * n2 * (Lc - 2 * rec - D2 + 300) / 1000); //Armadura
superior
    co2a = co2a + (D3 * ((Math.Round((Lc - 2 * rec) / sep3, 0)) + 1) * ((2 *
(b + c) - 8 * rec + 100)) / 1000); //Armadura cortante
    co2a = co2a * CO2A; //Precio Total
    //C02 hormigon-->Volumen*co2
    int H1;
    if (H <= 55) { H1 = ((int)H / 5) - 5; } else { H1 = ((int)H / 10) + 1; }
    double co2h;
    co2h = (Lc * b * c / (1000000000)) * CO2H[H1];
    //C02 encofrado-->area enc*co2
    double co2enc;
    co2enc = ((2 * (Lc * c + b * c) + Lc * b) / 1000000) * CO2ENC;
    //C02 total
    double co2 = co2a + co2h + co2enc;

    //Console.WriteLine("co2: " + co2);
    return co2;
}
public double valoraenergia(double b, double c, double H, double D1, double
D2, double D3, double n1, double n2, double sep3, double rec, double Lc)
{
    //Coste Energético
    double[] EH = { 414.59, 440.74, 458.02, 477.5, 505.73, 551.07, 622.30,
658.58, 670.01, 667.12, 688.14, 723.72 };
    double EA = 10.44;
    double EENC = 10.97;
    //Energía
    //Coste energético del acero-->area*n barras*lon*Kwh
    double ea;
    ea = (D1 * n1 * (Lc - 2 * rec - D1 + 300) / 1000); //Armadura inferior
    ea = ea + (D2 * n2 * (Lc - 2 * rec - D2 + 300) / 1000); //Armadura
superior
    ea = ea + (D3 * ((Math.Round((Lc - 2 * rec) / sep3, 0)) + 1) * ((2 * (b +
c) - 8 * rec + 100)) / 1000); //Armadura cortante

```

```

    ea = ea * EA; //Precio Total
    //Coste energético hormigon-->Volumen*Kwh
    int H1;
    if (H <= 55) { H1 = ((int)H / 5) - 5; } else { H1 = ((int)H / 10) + 1; }
    double eh;
    eh = (Lc * b * c / (100000000)) * EH[H1];
    //Coste energético encofrado-->area enc*Kwh
    double eenc;
    eenc = ((2 * (Lc * c + b * c) + Lc * b) / 1000000) * EENC;
    //Coste total
    double e = ea + eh + eenc;

    return e;
}
public double valorabarra(double D1, double n1, double D2, double n2, double
D3, double sep3, double rec, double Lc)
{
    //Suma de la cantidad de barras
    double b = (n1) + (n2) + (((Math.Round((Lc - 2 * rec) / sep3, 0)) + 1));
    return b;
}
}
//Seleccion de progenitores, genes, hijos prodigos y mutacion
class Descendiente
{
    public int elcpadre(int max_especimenes, bool[] nopareto, int fitness, Random
rnd)
    {
        int fitpar = rnd.Next(0, 100);
        int padre = 0;
        for (int i = 0; i < 10000000; i++)
        {
            padre = rnd.Next(0, max_especimenes - 1);
            if (nopareto[padre] == false && fitpar <= fitness) { break; }
            if (fitpar > fitness) { break; }
        }
        return padre;
    }
    public int elecmadre(int max_especimenes, bool[] nopareto, int fitness, int
padre, Random rnd)
    {
        int fitpar = rnd.Next(0, 100);
        int madre = 0;
        for (int i = 0; i < 10000000; i++)
        {
            madre = rnd.Next(0, max_especimenes - 1);
            //Prohibicion incesto
            while (madre == padre) { madre = rnd.Next(0, max_especimenes - 1); };
            if (nopareto[madre] == false && fitpar <= fitness) { break; }
            if (fitpar > fitness) { break; }
        }
        return madre;
    }
    public int ruleta(Random rnd)
    {
        int ruleta = 1 + rnd.Next(0, 6);
        return ruleta;
    }
    public int ruleta2(int rul1, Random rnd)
    {
        int ruleta2 = 1 + rnd.Next(0, 7);
        while (ruleta2 <= rul1) { ruleta2 = 1 + rnd.Next(0, 7); }
    }
}

```

```

        return ruleta2;
    }
    public int mutacion(Random rnd)
    {
        int mut = rnd.Next(0, 99);
        return mut;
    }
    public double selecmutagen(int i, Random rnd, double rec, double Lc, double
G, double Q)
    {
        Viga mutado = new Viga();
        double[] mutante = new double[13];
        mutado.generaviga(ref mutante, mutado, rnd, rec, Lc, G, Q);
        return mutante[i];
    }

    public int prodigo(Random rnd)
    {
        int prodigo = rnd.Next(0, 99);
        return prodigo;
    }
}
//Generación de descendiente y selección de los especimenes
class Generacion
{
    public int config()
    {
        return 0;
    }
    public int hijo(int max_especimenes, int prob_mut, int prob_prodigo,
double[,] poblacion, bool[] pareto, int fitness, double rec, double Lc, double G,
double Q, ref double[] hijo, Random rnd)
    {
        int check = 0;
        int numhijos = 0;
        Descendiente padres = new Descendiente();
        Viga viguita = new Viga();
        int prodigo = padres.prodigo(rnd);
        for (; ; )
        {
            int rul1 = padres.ruleta(rnd);
            int rul2 = padres.ruleta2(rul1, rnd);
            int padre = padres.elcpadre(max_especimenes, pareto, fitness, rnd);
            int madre = padres.elecmadre(max_especimenes, pareto, fitness, padre,
rnd);
            Console.WriteLine("Padre: " + padre + " Madre: " + madre);
            Console.WriteLine("Rul1: " + rul1 + " Rul2: " + rul2);
            for (int i = 0; i <= 8; i++)
            {
                //Combinacion de los genes del padre i de la madre
                if (i < rul1) { hijo[i] = poblacion[padre, i];;}
                if (i < rul2 && i >= rul1) { hijo[i] = poblacion[madre, i];}
                if (i <= 8 && i >= rul2 && rul2 != 8) { hijo[i] =
poblacion[padre, i];;}
                //Mutacion de uno de los genes
                int mut = padres.mutacion(rnd);
                if (mut <= prob_mut) { hijo[i] = padres.selecmutagen(i, rnd, rec,
Lc, G, Q);}
            }
            check = viguita.comprueba(hijo[0], hijo[1], hijo[2], hijo[3],
hijo[4], hijo[5], hijo[6], hijo[7], hijo[8], rec, G, Q, Lc);
            numhijos++;
        }
    }
}

```

```

        Console.WriteLine("Numero de hijos: " + numhijos);
        Console.WriteLine("Check: " + check);
        if (check == 1)
        {
            hijo[9] = viguita.valoracoste(hijo[0], hijo[1], hijo[2], hijo[3],
hijo[4], hijo[5], hijo[6], hijo[7], hijo[8], rec, Lc);
            hijo[10] = viguita.valoraco2(hijo[0], hijo[1], hijo[2], hijo[3],
hijo[4], hijo[5], hijo[6], hijo[7], hijo[8], rec, Lc);
            hijo[11] = viguita.valoraenergia(hijo[0], hijo[1], hijo[2],
hijo[3], hijo[4], hijo[5], hijo[6], hijo[7], hijo[8], rec, Lc);
            hijo[12] = viguita.valorabarra(hijo[3], hijo[6], hijo[4],
hijo[7], hijo[5], hijo[8], rec, Lc);
                break;
            };
        }
        check = 0;
        if (prodigo < probab_prodigos)
        {
            for (; ; )
            {
                viguita.generaviga(ref hijo, viguita, rnd, rec, Lc, G, Q);
                check = viguita.comprueba(hijo[0], hijo[1], hijo[2], hijo[3],
hijo[4], hijo[5], hijo[6], hijo[7], hijo[8], rec, G, Q, Lc);
                if (check == 1)
                {
                    hijo[9] = viguita.valoracoste(hijo[0], hijo[1], hijo[2],
hijo[3], hijo[4], hijo[5], hijo[6], hijo[7], hijo[8], rec, Lc);
                    hijo[10] = viguita.valoraco2(hijo[0], hijo[1], hijo[2],
hijo[3], hijo[4], hijo[5], hijo[6], hijo[7], hijo[8], rec, Lc);
                    hijo[11] = viguita.valoraenergia(hijo[0], hijo[1], hijo[2],
hijo[3], hijo[4], hijo[5], hijo[6], hijo[7], hijo[8], rec, Lc);
                    hijo[12] = viguita.valorabarra(hijo[3], hijo[6], hijo[4],
hijo[7], hijo[5], hijo[8], rec, Lc);
                        break;
                    }
                }
            }
        }
        return 1;
    }
    public int sort(ref double[,] generacion, int critord)
    {
        //Bubble method en referencia al coste de menor a mayor
        double[] temp = new double[generacion.GetLength(0)];
        for (int i = 0; i < generacion.GetLength(0); i++)
        {
            for (int sort = 0; sort < generacion.GetLength(0) - 1; sort++)
            {
                if (generacion[sort, critord] > generacion[sort + 1, critord])
                {
                    for (int t = 0; t <= 12; t++) { temp[t] = generacion[sort +
1, t]; };
                    for (int s = 0; s <= 12; s++) { generacion[sort + 1, s] =
generacion[sort, s]; };
                    for (int t = 0; t <= 12; t++) { generacion[sort, t] =
temp[t]; };
                }
            }
        }
        return 0;
    }
    public int elitismo(int max_especimenes, int max_descen, ref double[,]
poblacion, double[,] descendencia, ref bool[] pareto, int critord)

```

```
{
    double[,] generacion = new double[max_especimenes + max_descen,
poblacion.GetLength(1)];
    Generacion descendencias = new Generacion();
    for (int i = 0; i < max_especimenes; i++)
    {
        for (int j = 0; j <= 12; j++)
        {
            generacion[i, j] = poblacion[i, j];
        }
    }
    for (int i = max_especimenes; i < poblacion.GetLength(0) +
descendencia.GetLength(0); i++)
    {
        for (int j = 0; j <= 12; j++)
        {
            generacion[i, j] = descendencia[i - max_especimenes, j];
        }
    }
    descendencias.sort(ref generacion, critord);
    for (int i = 0; i < max_especimenes; i++)
    {
        for (int j = 0; j <= 12; j++)
        {
            poblacion[i, j] = generacion[i, j];
        }
    }

    for (int i = 0; i < poblacion.GetLength(0) - 1; i++)
    {
        if (poblacion[i + 1, 12] > poblacion[i, 12]) { pareto[i] = true; }
    }

    return 0;
}
}
```

## ANEXO 3: GUÍA DE USUARIO

### A. GUÍA PASO A PASO

Se va a desarrollar un ejemplo explicado paso a paso del funcionamiento del plugin generado. En este ejemplo se han utilizado los valores de los parámetros que se han definido en el apartado 3.1 como problema estándar.

#### Paso 1: Crear el archivo

Para iniciar la aplicación desarrollada lo primero que tenemos que generar es el plugin. Al abrir el software Autodesk Revit este nos muestra la siguiente pantalla:

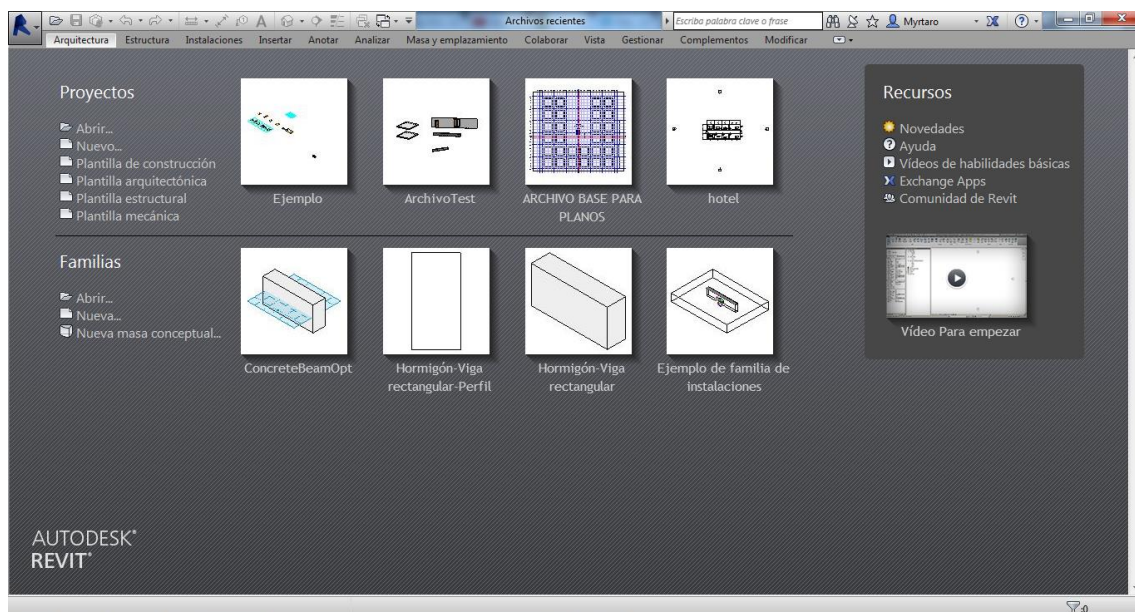


Fig. 24 Ventana de inicio Autodesk Revit

En esta pantalla se nos muestran varias opciones de creación de archivo, en el menú izquierdo. Por un lado tenemos las opciones relativas a la generación de diferentes plantillas para *Building models* y por otro diferentes opciones para generar una familia nueva. En nuestro caso queremos generar un *Building Model* que contenga nuestra familia ConcreteBeamOpt, que ya está creada, por lo que tenemos que seleccionar una de las opciones bajo la categoría de Proyectos.

En esta sección encontramos cuatro opciones, Plantilla de Construcción, Plantilla Arquitectónica, Plantilla Estructural y Plantilla mecánica. La realidad es que las únicas diferencias entre estas plantillas son algunas de las opciones de visualización y representación de objetos activas por defecto, por el resto las plantillas funcionan exactamente del mismo modo y todos estos factores pueden ser editados posteriormente. Nuestra intención al crear esta aplicación



es que se pueda utilizar directamente desde cualquier archivo, por este motivo se va a seleccionar la plantilla de uso más habitual, la plantilla arquitectónica.

Con el archivo abierto procedemos a modelar el edificio que vamos a utilizar de ejemplo. Se trata de una vivienda tipo extraída de un edificio de viviendas en corredor. La vivienda de 55 m<sup>2</sup> tiene el siguiente programa: un salón-comedor, cocina, dos cuartos dobles y un cuarto de baño.

El edificio modelado es el siguiente.

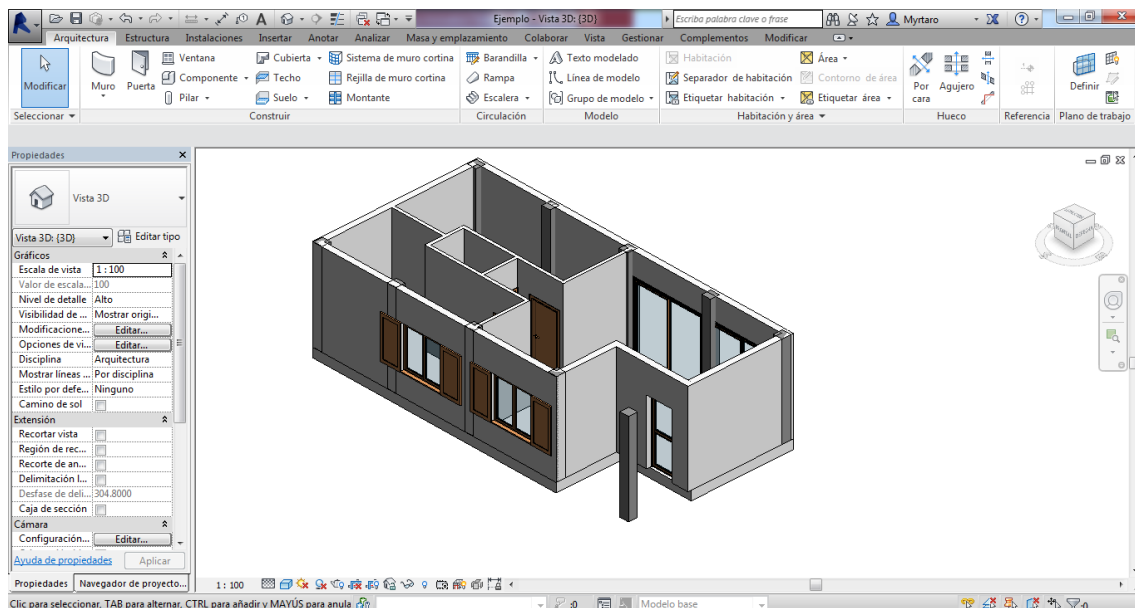


Fig. 25 Modelo utilizado en el ejemplo

## Paso 2: Introducción de la familia

En este momento tenemos todos los datos del *Building Model* en nuestro archivo excepto las vigas que queremos optimizar. Para introducir una viga nos situamos en la pestaña Estructura de la barra de Autodesk Revit y pulsamos en el botón Viga. Esto inicia el comando para la introducción de familias que Revit reconoce como vigas, y por tanto forman parte del modelo analítico.

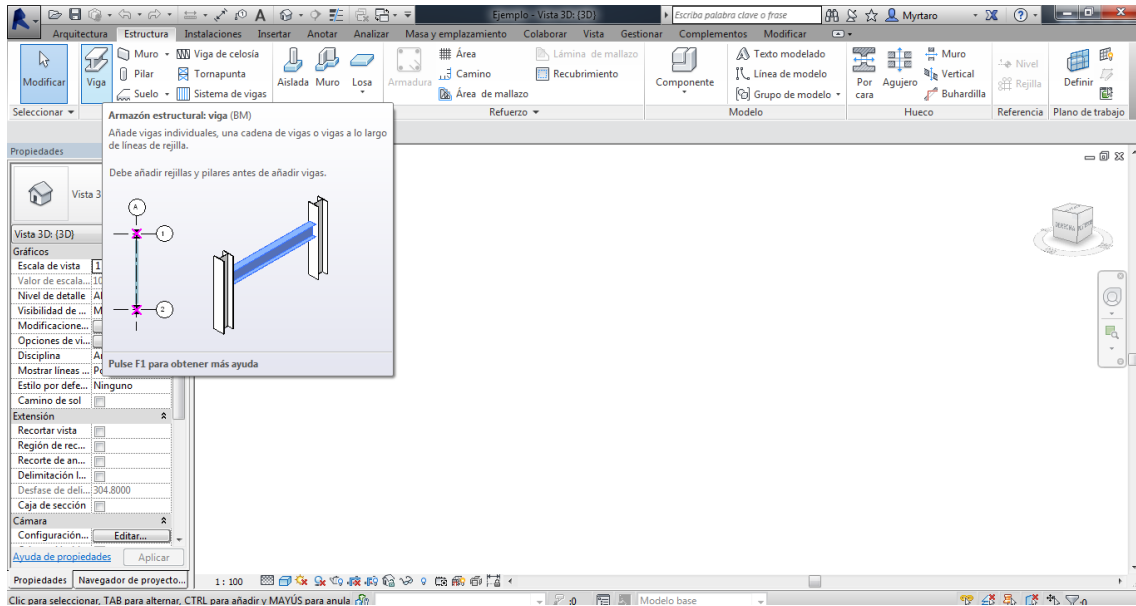


Fig. 26 Herramienta de creación de vigas

La familia que se selecciona por defecto al activar este comando es la familia que Revit tiene por defecto para las vigas. Este objeto posee los elementos analíticos de una viga, pero no posee todos los parámetros que hemos definido para el correcto funcionamiento de nuestro algoritmo. Por lo tanto es una familia que no podemos utilizar.

La familia que necesitamos utilizar es la que creada *ex profeso* para nuestro algoritmo, la familia *ConcreteBeamOpt*. Esta no es una de las familias que se cargan por defecto al generar el fichero, por lo que hemos de cargar la familia. Para cargar una familia, con el comando de Viga activo, hemos de pulsar el botón Cargar Familia que nos aparece.

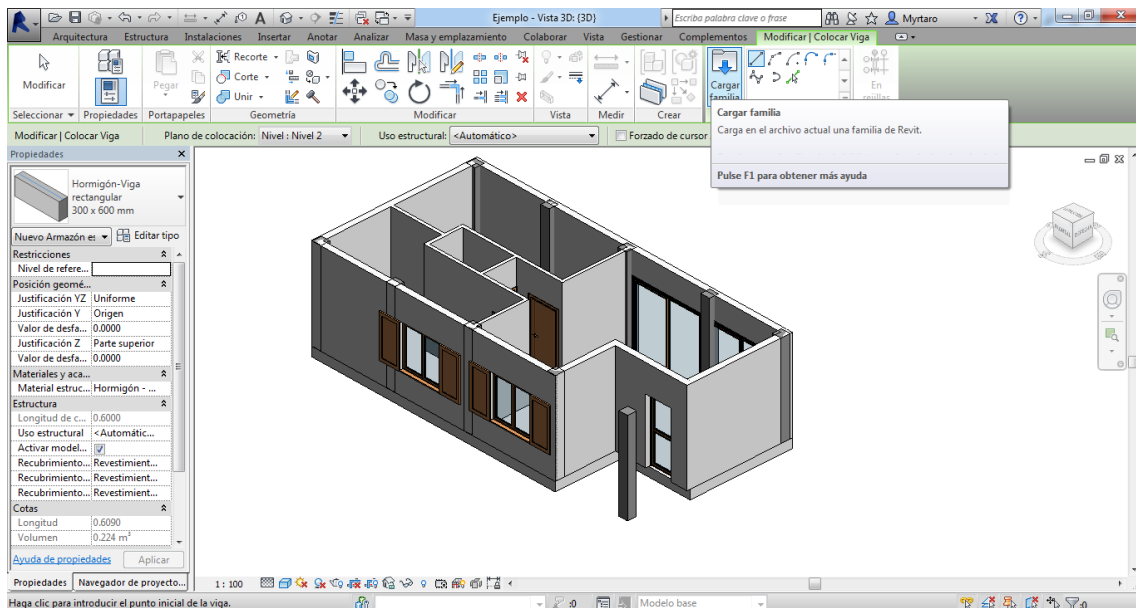


Fig. 27 Carga de la familia

Esto nos abre un explorador de archivos que nos permite seleccionar entre los diferentes ficheros de familias que tengamos en el disco duro. Estos archivos tienen la extensión *.rfa*. Utilizando el explorador seleccionamos la familia *ConcreteBeamOpt*. En el ejemplo esta familia esta alojada en la misma ubicación que el resto de familias gestionadas por Revit, esto no es necesario para el funcionamiento del plugin.

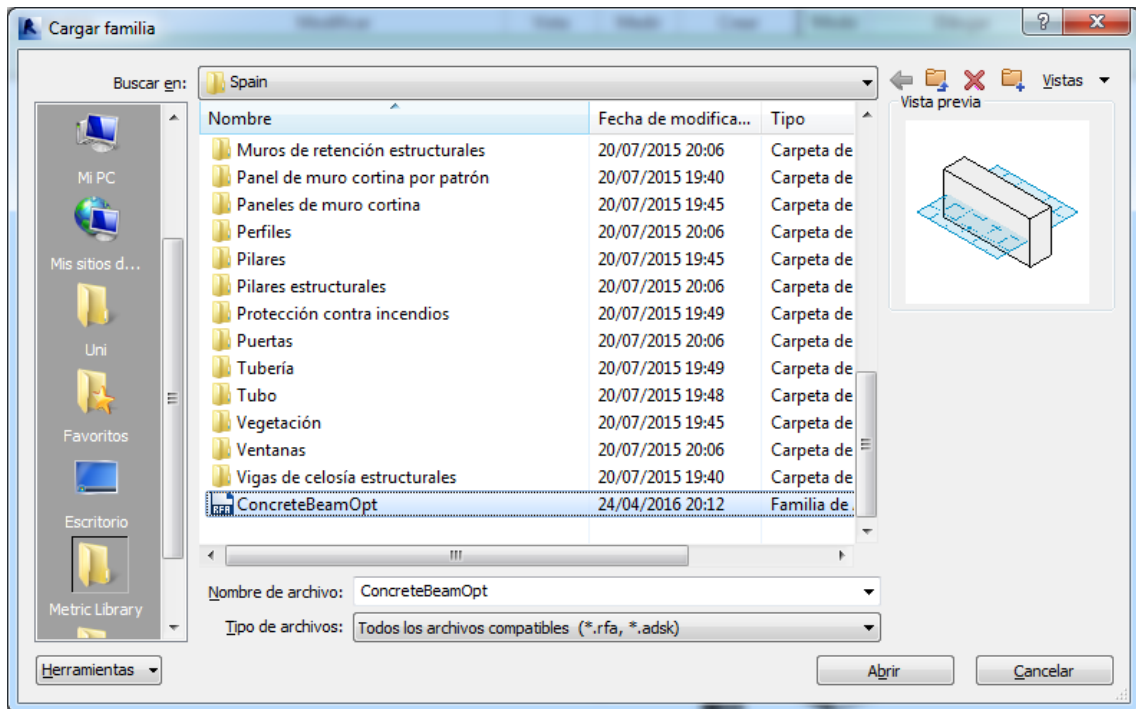


Fig. 28 Selección de la familia *ConcreteBeaOpt*

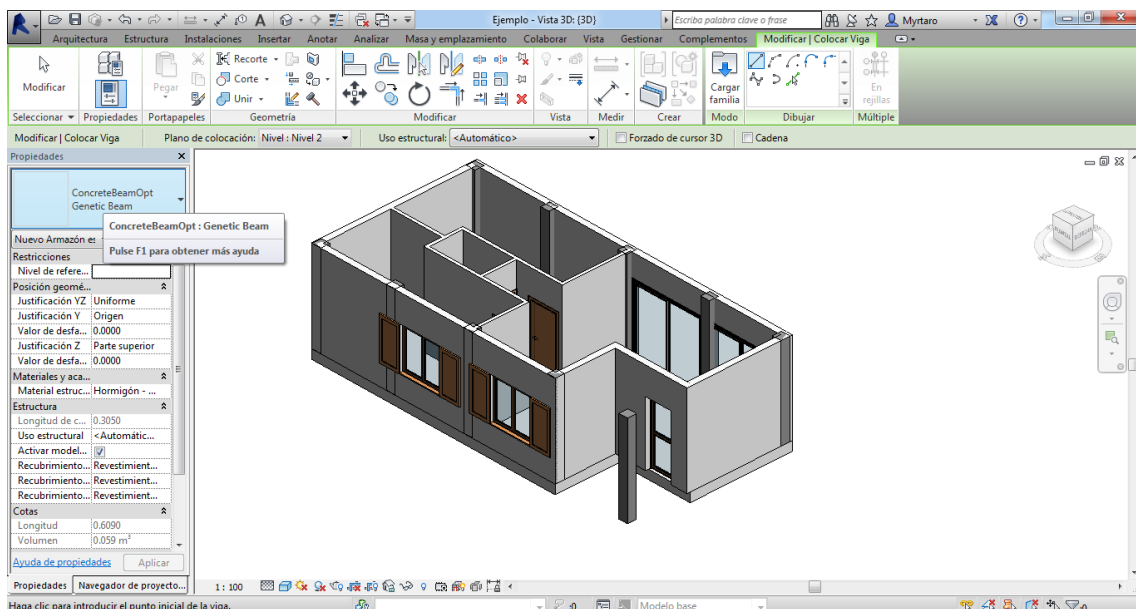


Fig. 29 Familia carga y lista para su introducción

Una vez cargada la familia esta nos aparece seleccionada en la ventana de Propiedades. Si esto no sucede, es necesario pulsar la familia actual y la familia *ConcreteBeamOpt* estara disponible

en el menú desplegable. Con la familia seleccionada se introduce en el modelo en su ubicación real.

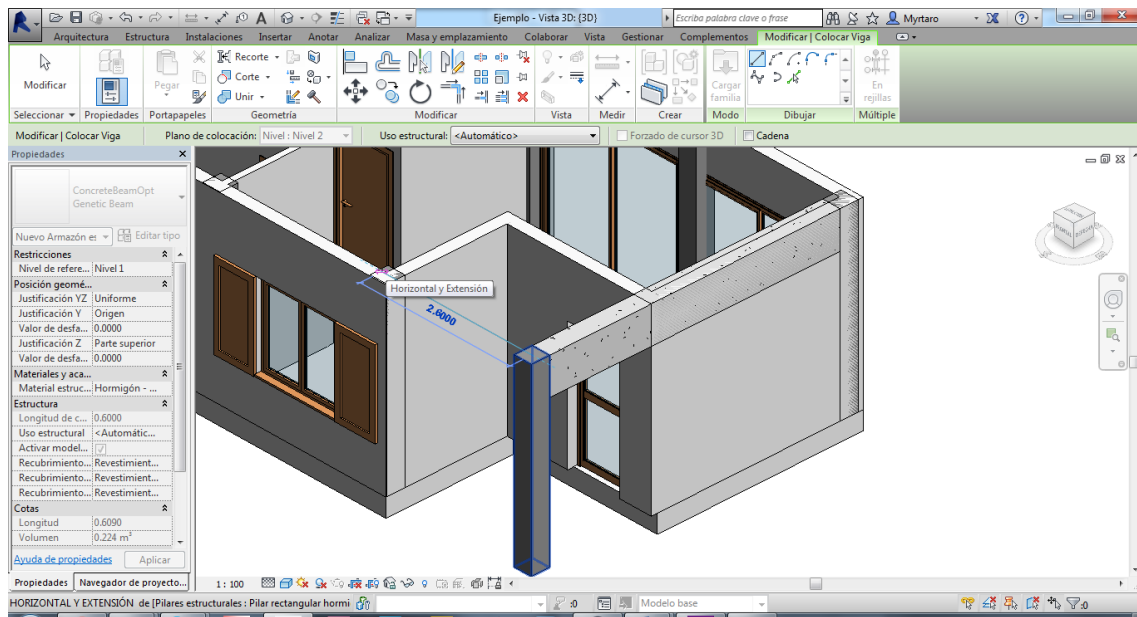


Fig. 30 Intrucción de la familia en el modelo

Paso 3: Configuración de los parámetros.

Antes de activar la aplicación hemos de realizar un último paso. Configurar los diferentes parámetros a la viga. Para ello, con el elemento seleccionado pulsamos en el botón Editar Tipo de la barra de Propiedades para que nos muestre un desplegable con los parámetros del ejemplar.

Los parámetros relativos a la geometría de la pieza (incluyendo los relativos a las barras de armado) no es necesario cambiarlos ya que serán sobrescritos por los valores óptimos tras la ejecución del algoritmo. Tampoco es necesario introducir la resistencia del hormigón por el mismo motivo. Si se quiere realizar la optimización de más de un elemento y se prevee la obtención de resultado diferentes durante la ejecución del algoritmo sí que se debe duplicar el tipo de la familia para que la variación de los parámetros afecte únicamente a los ejemplares deseados.

El proyecto es un edificio de viviendas en corredor. Por lo que una estimación de cargas habitual en este tipo de edificación es es la siguiente:

**Cargas**

|                                  | Carga Superficial   | Carga Lineal |
|----------------------------------|---------------------|--------------|
| <b>Carga Permanente: Forjado</b> | 3 KN/m <sup>2</sup> | 9 KN/m       |
| <b>Carga Variable: Uso</b>       | 2 KN/m <sup>2</sup> | 6 KN/m       |

Tabla 20 Estimación de Cargas

El ámbito de carga de cada una de las barras es de 3 m, que consideramos entero aunque estemos en el vano limítrofe debido a que existe una vivienda adosada. Conocido este dato introducimos los valores de las cargas en la familia.

Por último tenemos que editar los parámetros que configuran al algoritmo genético. Estos parámetros deben ser ajustados mediante métodos estadísticos para obtener el mejor resultado posible con la menor variabilidad. Como esta es la primera vez que se ejecuta el algoritmo con esta estructura es imposible conocer de antemano cuales son estos valores, por lo que utilizaremos los valores estándar que la familia trae por defecto. En el apartado A3.2 de este mismo anexo se muestra el modo de utilizar la técnica de diseño de experimentos para ajustar estos parámetros, los resultados de este estudio definen los parámetros por defecto de la familia. Los valores que utilizaremos son los siguientes.

| Parámetros         |     |
|--------------------|-----|
| <b>max_espe.</b>   | 250 |
| <b>max_descen.</b> | 250 |
| <b>gen.</b>        | 100 |
| <b>mut.</b>        | 5   |
| <b>prod.</b>       | 2   |
| <b>Fitness</b>     | 75  |

Tabla 21 Parámetros del algoritmo genético

Una vez introducidos estos parámetros tenemos la familia configurada para iniciar la aplicación. La lista de parámetros de ejemplar de la familia tiene los siguientes valores.

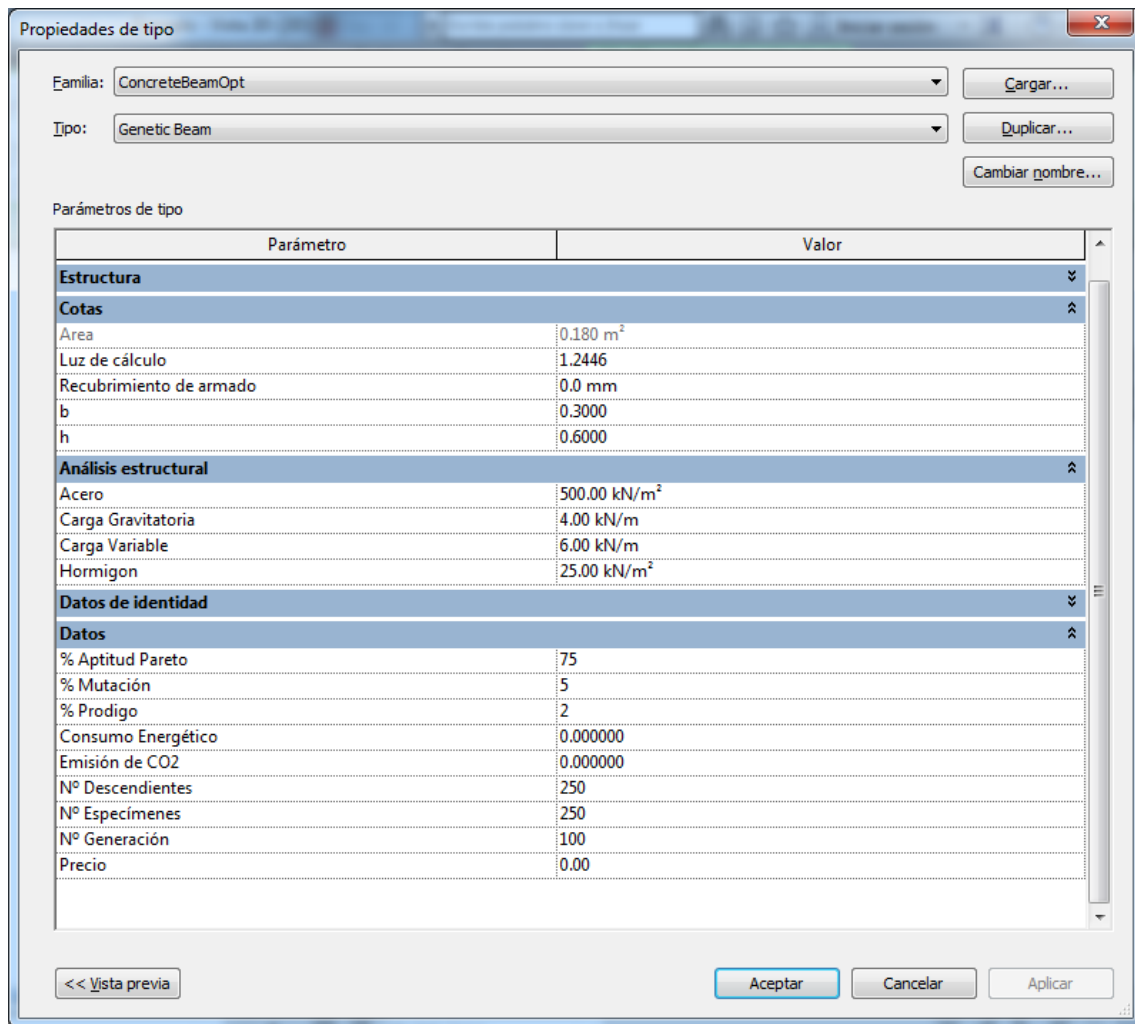


Fig. 31 Parámetros de optimización de la familia

#### Paso 4: Optimización de la familia

Con la introducción de los parámetros hemos finalizado la configuración de la barra para su optimización. Únicamente nos queda iniciar la aplicación, para esto nos situamos en la pestaña Complementos y abrir el desplegable de Herramientas Externas. En este desplegable se nos muestra todos los *plugins* que tenemos disponibles en Revit. Activamos nuestra aplicación seleccionándola en el desplegable.

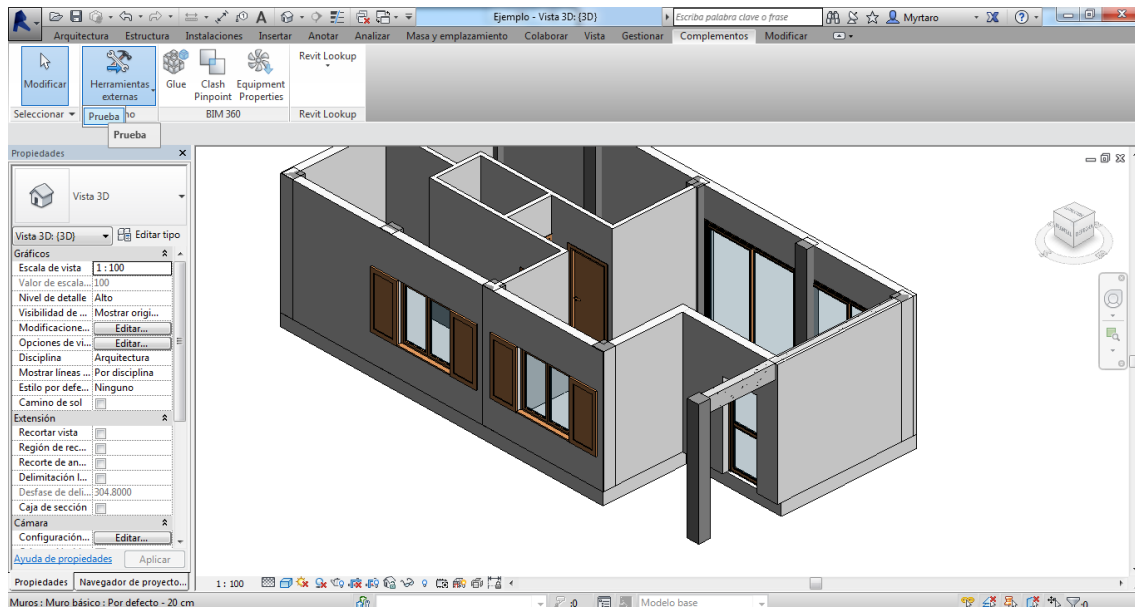


Fig. 32 Inicio del plugin

Una vez seleccionada la aplicación se comunica con el usuario pidiéndole que seleccione un objeto mediante un mensaje en la esquina inferior izquierda.

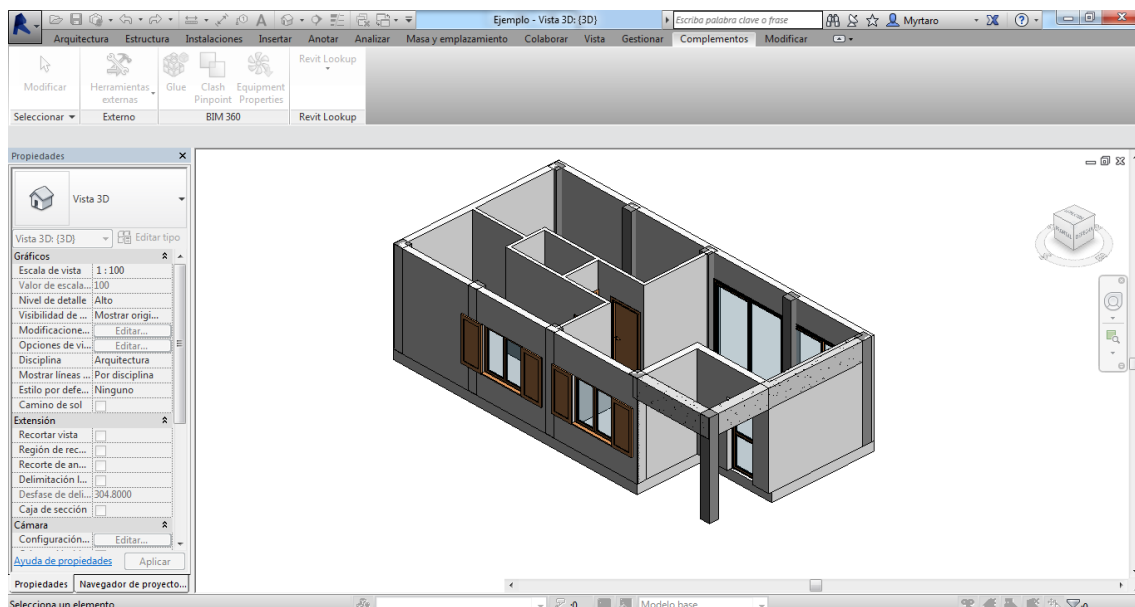


Fig. 33 Selección del elemento a optimizar

Haciendo clic sobre nuestra viga el algoritmo nos proporciona un mensaje informando que la selección ha sido aceptada. Tras esto procede de modo automático con la optimización. Al terminar nos muestra una serie de mensajes informativos sobre el coste económico de la viga, la cantidad total de emisiones de CO<sub>2</sub> producidas y el consumo energético que conlleva la fabricación de la misma. Todas estas ventanas emergentes se muestran en la imagen siguiente nombradas de (a) a (d) respectivamente.

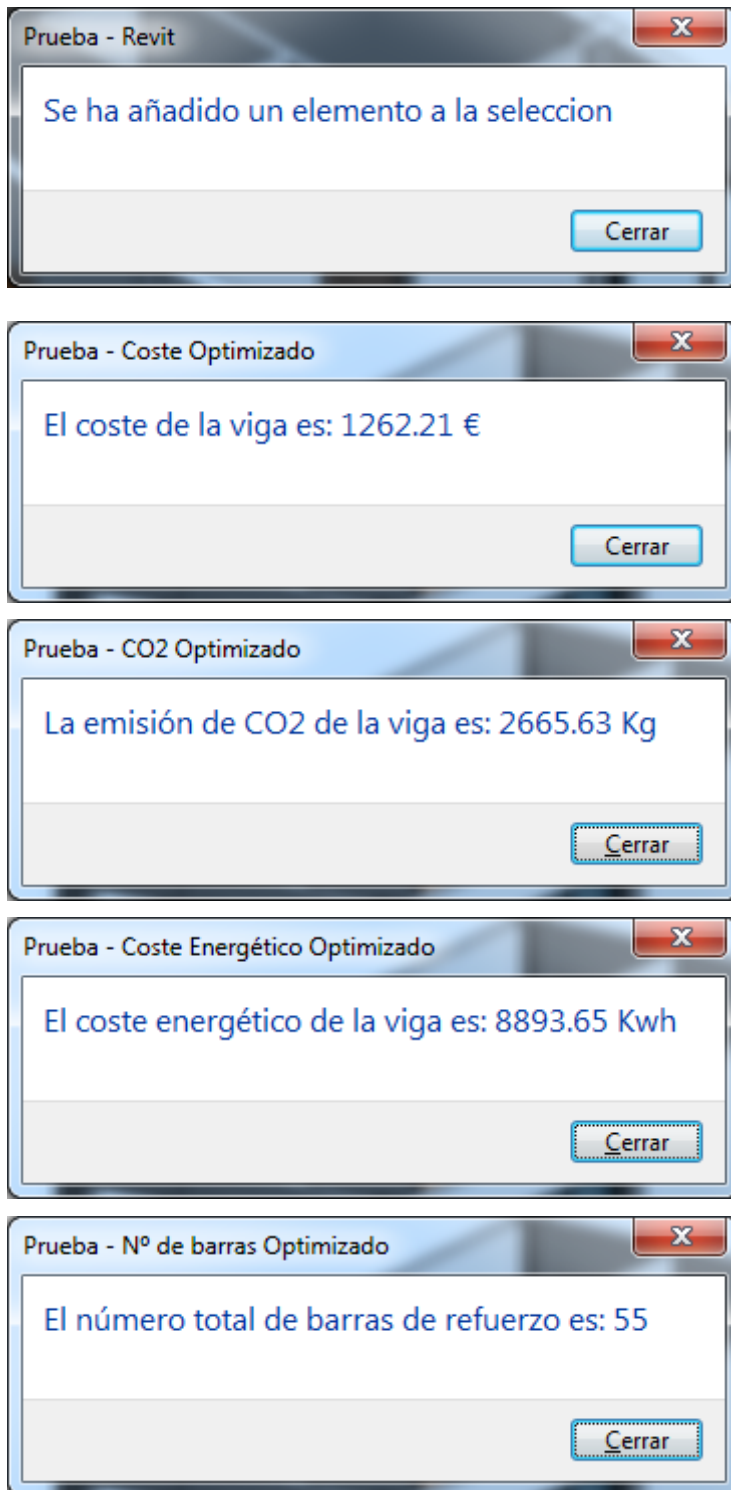


Fig. 34 (a-e) Resultados de la optimización

Una vez se han mostrado las ventanas, se finaliza la transacción mediante la cual Revit recibe los datos generados por el algoritmo y se actualiza. Los cambios existentes en la geometría de la barra tras la actualización son las nuevas dimensiones del elemento una vez optimizado. El valor de estos y del resto de parámetros se puede consultar seleccionando el elemento y entrando en la ventana Editar Tipo.



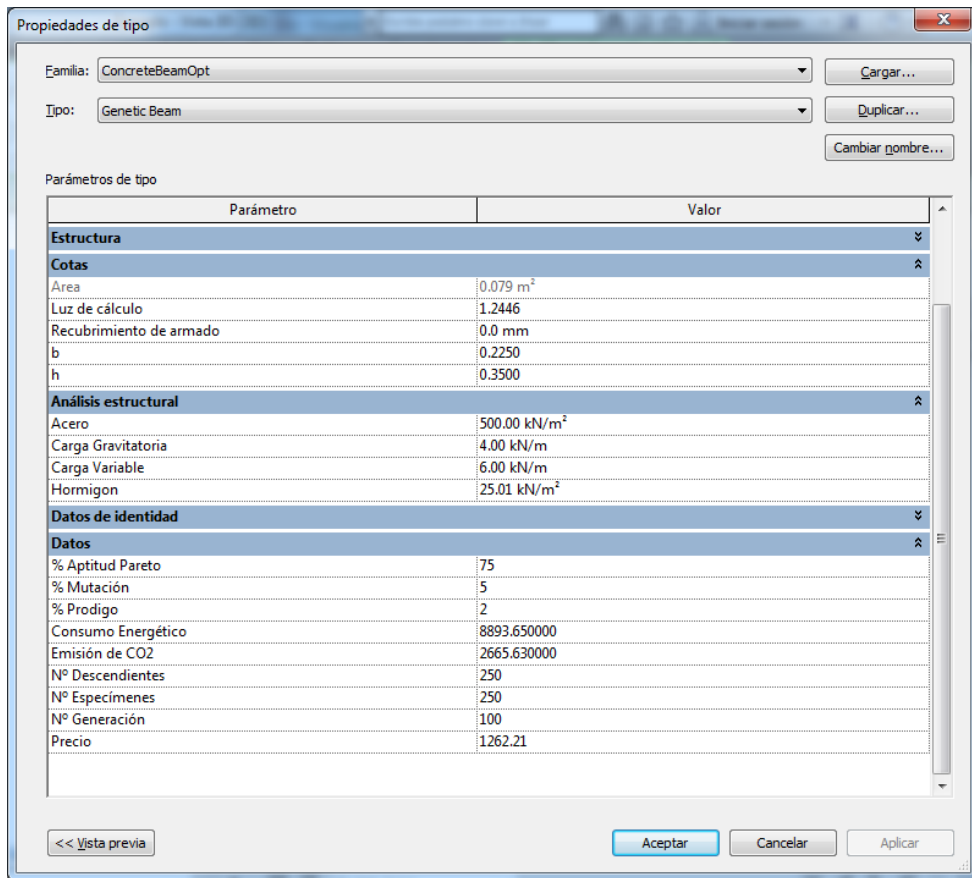
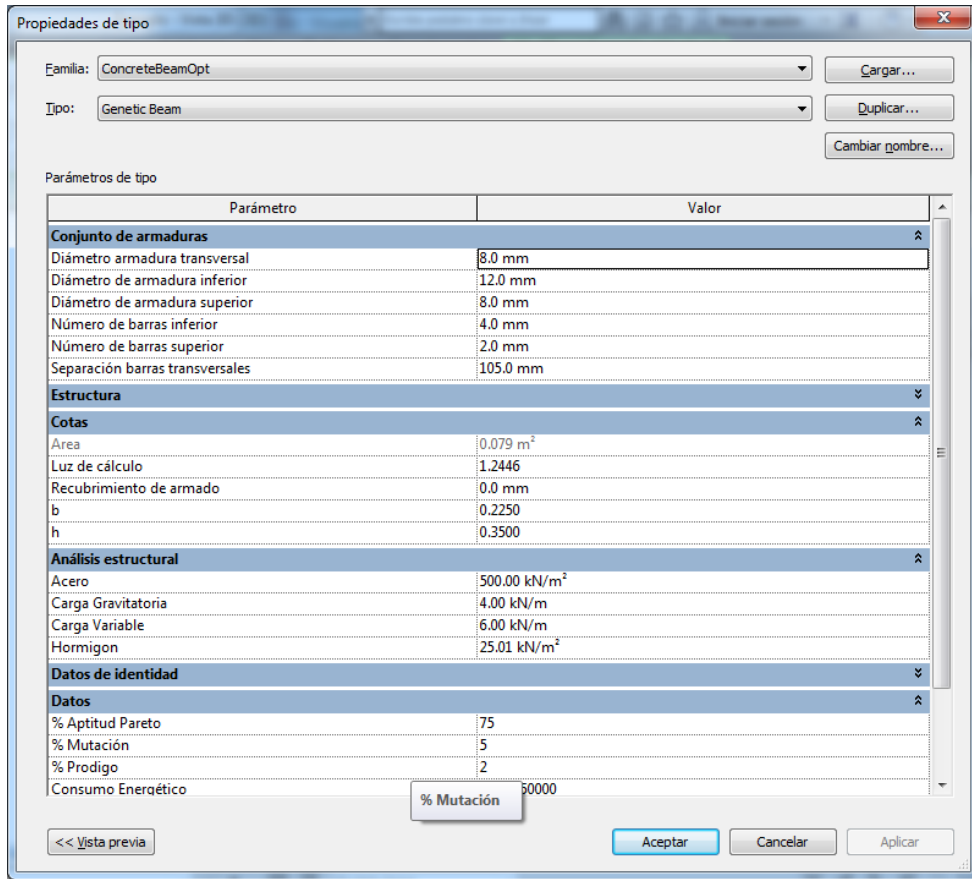


Fig. 35 Parámetros de familia tras la optimización

Con esto se habría terminado la ejecución del algoritmo y se tendría la viga optimizada. Algunos de los valores de la geometría pueden no tener sentido a nivel constructivo, dado que los incrementos en las dimensiones son de 5 mm. La herramienta está refinada y preparada para fines de investigación, no para construcción, por lo que se utilizan incrementos de dimensiones poco prácticos en la realidad para hallar el óptimo. Para el uso en construcción del elemento se recomienda refinar estas medidas y normalizarlas a lo largo de toda la obra. Esta es una de las características de mejora que se expondrán en el apartado 4.4.

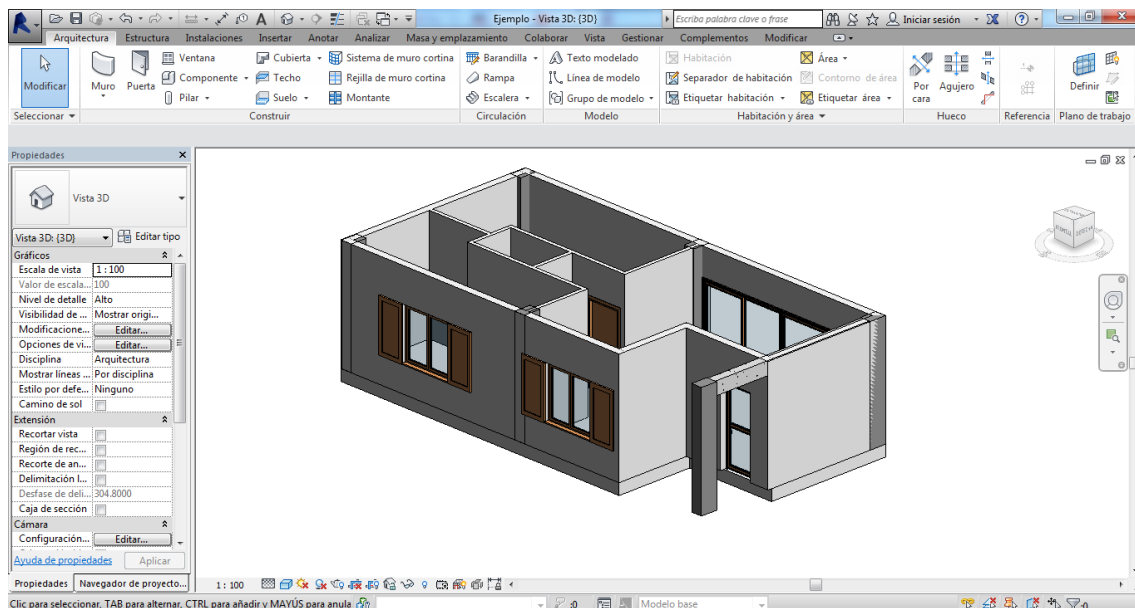


Fig. 36 Modelo tras realizar la optimización de la viga

## B. EJEMPLO DE AJUSTE DE LOS PARÁMETROS

### a. Diseño de experimentos

El diseño de experimentos (DOE) es un conjunto de técnicas que manipulan el proceso para inducirlo a proporcionar la información requerida para su mejora. Estas técnicas estadísticas permiten lograr la máxima cantidad de información con el menor número de procesos, o experimentos, realizados. De este modo se consigue reducir el coste y el esfuerzo a la hora de realizar los experimentos.

Para estudiar como afectan diferentes factores a un mismo proceso, tradicionalmente sería necesario realizar todas las combinaciones entre los factores. Por ejemplo si tuvieramos tres factores con dos valores cada uno sería necesario realizar un total de ocho combinaciones para cubrir toda la casuística y aplicar el método científico. Algunas de estas observaciones realizadas no aportarían gran cantidad de información mientras que otras serían las más relevantes.

El diseño de experimentos nos permite, *a priori*, estudiar cuáles son las combinaciones de factores más relevantes, con la finalidad de disminuir el número de observaciones necesarias para obtener unos resultados significativos. Siguiendo el ejemplo anterior y aplicando el diseño de experimentos conseguiríamos obtener unas conclusiones semejantes a las del ejemplo anterior realizando únicamente dos o cuatro combinaciones.

A un mismo estudio se le puede aplicar diferentes diseños de experimentos que requerirán un número diferentes de observaciones. Al disminuir el número de observaciones es posible que se confundan varios de los efectos significativos producidos en el experimento, por lo que es necesario estudiar cómo aplicar esta técnica para obtener un buen resultado.

El objetivo de todo este proceso es determinar qué valores de los seis parámetros que regulan el algoritmo genético producen un mejor óptimo sin aumentar en exceso el tiempo de ejecución del algoritmo, además de garantizar cierta estabilidad en los resultados.

#### b. Tabla de diseño

Vamos a aplicar esta técnica para regular el valor de los parámetros del algoritmo genético para obtener un mejor resultado. Este proceso sería necesario realizarlo cada vez que varían los condicionantes estructurales de la viga, ya que es posible que el algoritmo responda de manera diferente. Se toma el modelo estructural utilizado en el ejemplo anterior.

En total tenemos que ajustar seis valores (ver tabla 3) que toman dos valores diferentes cada uno. Si realizáramos una observación completa necesitaríamos realizar un total de sesenta y cuatro emulaciones diferentes. Además se ha observado que el algoritmo, con los mismos parámetros, puede obtener resultados con gran variabilidad (diferencias de coste superiores al 30%) por lo que resultaría interesante realizar más de una emulación por cada caso. Esto dispara el número de observaciones a realizar.

Por todo esto nos hemos decidido a utilizar un diseño de experimentos. Existen diferentes posibilidades dentro del diseño de experimentos, siendo el más adecuado para este caso el diseño factorial fraccionado a dos niveles debido al gran número de variables a analizar. Además se quiere realizar más de una observación por cada combinación, por lo que resulta necesario mantener un número bajo, menor a 20, de observaciones.

Por todo esto nos decantamos por un diseño factorial  $2^{6-2}$  de resolución IV con dos réplicas para cada combinación, es decir un total de treinta y dos pruebas. Para conseguir este diseño y su posterior análisis se ha utilizado el programa estadístico Minitab. De este programa extraemos la siguiente tabla de diseño:

|    | A | B | C | D | AB | AC | AD | BC | BD | CD | ABC | ABD | BCD | ABCD |
|----|---|---|---|---|----|----|----|----|----|----|-----|-----|-----|------|
| 1  | + | + | + | + | +  | +  | +  | +  | +  | +  | +   | +   | +   | +    |
| 2  | - | + | + | + | -  | -  | -  | +  | +  | +  | -   | -   | +   | -    |
| 3  | + | - | + | + | -  | +  | +  | -  | -  | +  | -   | -   | -   | -    |
| 4  | - | - | + | + | +  | -  | -  | -  | -  | +  | +   | +   | -   | +    |
| 5  | + | + | - | + | +  | -  | +  | -  | +  | -  | -   | +   | -   | -    |
| 6  | - | + | - | + | -  | +  | -  | -  | +  | -  | +   | -   | -   | +    |
| 7  | + | - | - | + | -  | -  | +  | +  | -  | -  | +   | -   | +   | +    |
| 8  | - | - | - | + | +  | +  | -  | +  | -  | -  | -   | +   | +   | -    |
| 9  | + | + | + | - | +  | +  | -  | +  | -  | -  | +   | -   | -   | -    |
| 10 | - | + | + | - | -  | -  | +  | +  | -  | -  | -   | +   | -   | +    |
| 11 | + | - | + | - | -  | +  | -  | -  | +  | -  | -   | +   | +   | +    |
| 12 | - | - | + | - | +  | -  | +  | -  | +  | -  | +   | -   | +   | -    |
| 13 | + | + | - | - | +  | -  | -  | -  | -  | +  | -   | -   | +   | +    |
| 14 | - | + | - | - | -  | +  | +  | -  | -  | +  | +   | +   | +   | -    |
| 15 | + | - | - | - | -  | -  | -  | +  | +  | +  | +   | +   | -   | -    |
| 16 | - | - | - | - | +  | +  | +  | +  | +  | +  | -   | -   | -   | +    |

Tabla 22 Tabla de diseño del DOE

Nombramos los factores E y F a las siguientes interacciones:

$$E=ABC$$

$$F=BCD$$

Con ello obtenemos la siguiente ecuación generatriz o estructura ALIAS:

$$I=ABCE=ADEF=BCDF$$

Conocida esta se obtienen las interacciones existentes entre los diferentes factores debidas a este diseño:

- A = BCE = DEF = ABCDF
- B = ACE = CDF = ABDEF
- C = ABE = BDF = ACDEF
- D = AEF = BCF = ABCDE
- E = ABC = ADF = BCDEF
- F = ADE = BCD = ABCEF
- AB = CE = ACDF = BDEF
- AC = BE = ABDF = CDEF
- AD = EF = ABCF = BCDE
- AE = BC = DF = ABCDEF
- AF = DE = ABCD = BCEF
- BD = CF = ABEF = ACDE
- BF = CD = ABDE = ACEF
- ABD = ACF = BEF = CDE
- ABF = ACD = BDE = CEF

Este diseño tiene la ventaja de que los efectos principales únicamente se confunden con efectos de tercer orden, que podemos afirmar que son poco significativos y por lo tanto despreciar.

Es necesario establecer un valor máximo y mínimo para cada uno de los parámetros estudiados y asignar estos a uno de los factores del diseño. Como es la primera aproximación se va a considerar un rango alto de variación entre los factores que permita ser refinado con un segundo diseño.

| Factor | Parámetro   | Valor bajo | Valor alto |
|--------|-------------|------------|------------|
| A      | max_espe.   | 50         | 500        |
| B      | max_descen. | 50         | 500        |
| C      | gen.        | 50         | 150        |
| D      | mut.        | 2          | 10         |
| E      | prod.       | 2          | 10         |
| F      | Fitness     | 50         | 99         |

Tabla 23 Valores de diseño

Tras establecer los valores de los factores se realizan los diferentes experimentos y se almacenan los cuatro factores estudiados, coste, emisión de CO<sub>2</sub>, energía consumida y tiempo de ejecución. La siguiente tabla muestra los resultados obtenidos así como los valores tomados para cada uno de los casos estudiados.

|    | max_espe. | max_desc. | gen. | mut. | prodigo. | fitness | Coste   | CO <sub>2</sub> | Energía  | N. barras | Tiempo |
|----|-----------|-----------|------|------|----------|---------|---------|-----------------|----------|-----------|--------|
| 1  | 50        | 50        | 50   | 2    | 2        | 50      | 1599,45 | 3444,79         | 11405,76 | 63        | 0,53   |
| 2  | 50        | 50        | 150  | 10   | 10       | 50      | 1378,89 | 2936,33         | 9747,64  | 58        | 3,11   |
| 3  | 50        | 500       | 150  | 2    | 2        | 50      | 1149,67 | 2395,22         | 7942,68  | 51        | 9,06   |
| 4  | 50        | 500       | 150  | 10   | 2        | 99      | 1109,93 | 2272,79         | 7511,7   | 45        | 35,64  |
| 5  | 50        | 50        | 50   | 10   | 2        | 99      | 1434,4  | 3084,26         | 10275,47 | 64        | 1,08   |
| 6  | 50        | 50        | 150  | 10   | 10       | 50      | 1224,78 | 2477,6          | 8111,68  | 40        | 2,9    |
| 7  | 500       | 50        | 50   | 2    | 10       | 50      | 3523,74 | 8125,85         | 26794,55 | 84        | 1,06   |
| 8  | 50        | 500       | 50   | 2    | 10       | 99      | 1110,97 | 2304,46         | 7662,39  | 50        | 4,22   |
| 9  | 500       | 50        | 150  | 10   | 2        | 50      | 1670,43 | 3674,1          | 11582,13 | 48        | 5,25   |
| 10 | 500       | 500       | 50   | 2    | 2        | 99      | 1151,81 | 2379,56         | 7883,76  | 48        | 7,53   |
| 11 | 50        | 500       | 50   | 10   | 10       | 50      | 1114,64 | 2276,64         | 7509,39  | 44        | 13,94  |
| 12 | 500       | 500       | 150  | 10   | 10       | 99      | 1152,98 | 2343,09         | 7693,69  | 42        | 44,08  |
| 13 | 500       | 50        | 150  | 2    | 2        | 99      | 1442,58 | 3156,36         | 10660,2  | 79        | 2,53   |
| 14 | 500       | 50        | 50   | 10   | 10       | 99      | 4326,91 | 9727,98         | 29436,82 | 29        | 1,64   |
| 15 | 50        | 500       | 150  | 2    | 2        | 50      | 1194,65 | 2427,71         | 7959,84  | 42        | 9,12   |
| 16 | 500       | 50        | 50   | 10   | 10       | 99      | 4219,87 | 9741,53         | 32485,97 | 109       | 1,92   |
| 17 | 500       | 500       | 50   | 2    | 2        | 99      | 1139,16 | 2351,83         | 7803     | 48        | 8,02   |
| 18 | 500       | 500       | 150  | 2    | 10       | 50      | 1162,03 | 2391,3          | 7889,32  | 46        | 19,23  |
| 19 | 50        | 50        | 150  | 2    | 10       | 99      | 1541,64 | 3326,79         | 11082,32 | 64        | 1,49   |
| 20 | 50        | 500       | 50   | 2    | 10       | 99      | 1348,63 | 2728,46         | 8835,6   | 37        | 4,64   |
| 21 | 50        | 50        | 150  | 2    | 10       | 99      | 1466,68 | 3194,86         | 10729,53 | 77        | 1,13   |
| 22 | 50        | 500       | 150  | 10   | 2        | 99      | 1162,03 | 2391,3          | 7889,32  | 46        | 39,83  |
| 23 | 500       | 500       | 50   | 10   | 2        | 50      | 1179,12 | 2331,82         | 7538,5   | 34        | 17,03  |
| 24 | 500       | 500       | 150  | 10   | 10       | 99      | 1111,63 | 2299,73         | 7638,54  | 49        | 65,01  |
| 25 | 500       | 50        | 150  | 2    | 2        | 99      | 1230,44 | 2540,85         | 8292,52  | 44        | 3,15   |
| 26 | 500       | 50        | 50   | 2    | 10       | 50      | 1108,57 | 2214            | 7249,08  | 38        | 12,97  |
| 27 | 500       | 500       | 50   | 10   | 2        | 50      | 1210,03 | 2435,32         | 7904,12  | 38        | 16,77  |
| 28 | 50        | 500       | 50   | 10   | 10       | 50      | 1124,08 | 2333,14         | 7747,52  | 50        | 13,17  |
| 29 | 50        | 50        | 50   | 2    | 2        | 50      | 1898,44 | 3893,3          | 12202,11 | 32        | 0,38   |
| 30 | 50        | 50        | 50   | 10   | 2        | 99      | 1299,71 | 2703,99         | 8900,34  | 47        | 1,12   |
| 31 | 500       | 50        | 150  | 10   | 2        | 50      | 1436,75 | 3076,91         | 10225,91 | 54        | 5,29   |
| 32 | 500       | 500       | 150  | 2    | 10       | 50      | 1165,97 | 2408,52         | 7990,8   | 48        | 18,46  |

Tabla 24 Combinaciones y resultados obtenidos en el DOE

A la vista de los resultados podemos observar que los tiempos de ejecución del algoritmo son generalmente bajos y se ven influenciados por los valores que utilizemos para los parámetros. De hecho, el mayor tiempo es de 65,01 segundos y el menor de 0,38 segundos una diferencia producida por el gran rango de estudio entre los parámetros. La primera conclusión que podemos extraer de este hecho es que al aumentar el valor de los parámetros aumenta exponencialmente el tiempo de ejecución.

Antes de continuar con el análisis tenemos que verificar que las muestras obtenidas cumplen una distribución normal para aceptar los resultados. Los resultados de las pruebas de normalidad son los que se muestran en la siguiente figura. Para todas las distribuciones los p-valor es inferior a 0,05 del intervalo de confianza por lo que se consideran muestras normalmente distribuidas y aceptamos los resultados obtenidos.

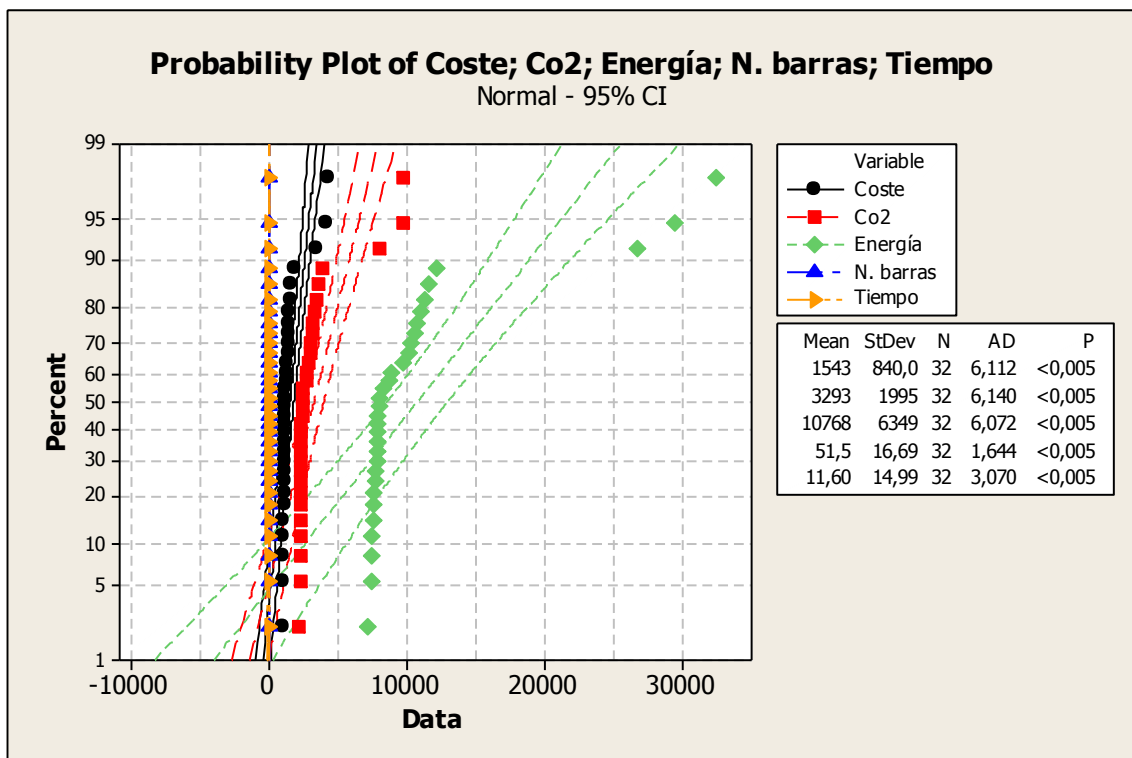


Fig. 37 Pruebas de normalidad Kolmogorov-Smirnov sobre los resultados

De este diseño de experimentos podríamos extraer unas rectas de regresión capaces de aproximar el valor que se va a obtener en cada uno de los criterios de optimización en función de los valores de los parámetros, pero en este caso no se va a realizar este estudio por dos motivos. Por un lado, al existir cuatro criterios diferentes de optimización existen cuatro rectas de regresión diferentes y cada una tiene que ser alimentada con los valores de los seis parámetros de entrada, aunque esto no es de alta dificultad si que supone una mayor inversión de tiempo que simplemente ejecutar el algoritmo dados los bajos tiempos de ejecución. Por

otro lado a través de cada una de las rectas únicamente obtendríamos un valor aproximado que tendría una variabilidad diferente en función del criterio utilizado.

A continuación vamos a estudiar, mediante diagramas de Pareto, la influencia que tiene cada uno de los parámetros en cada uno de los criterios de optimización. Este dato es interesante ya que nos permite en cierta manera priorizar la mejora de alguno de los criterios frente a otro sin incrementar mucho los tiempos de ejecución.

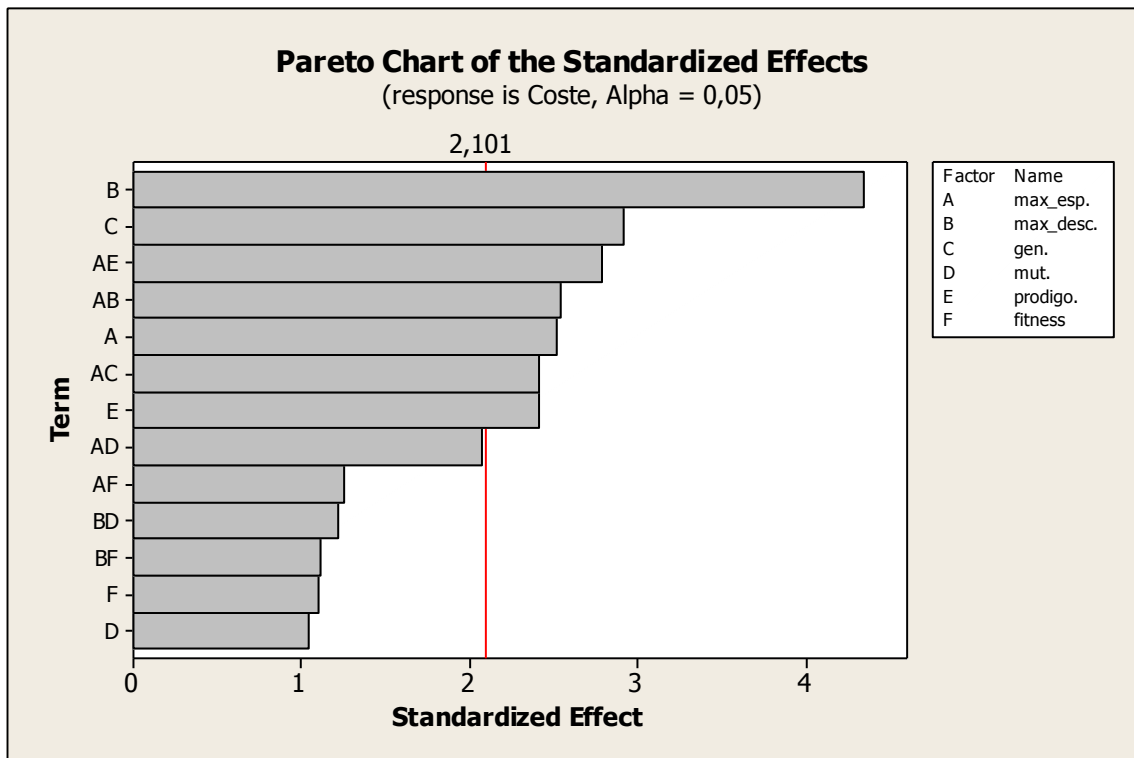


Gráfico 36 Frontera de Pareto de los efectos estandarizados sobre el Coste

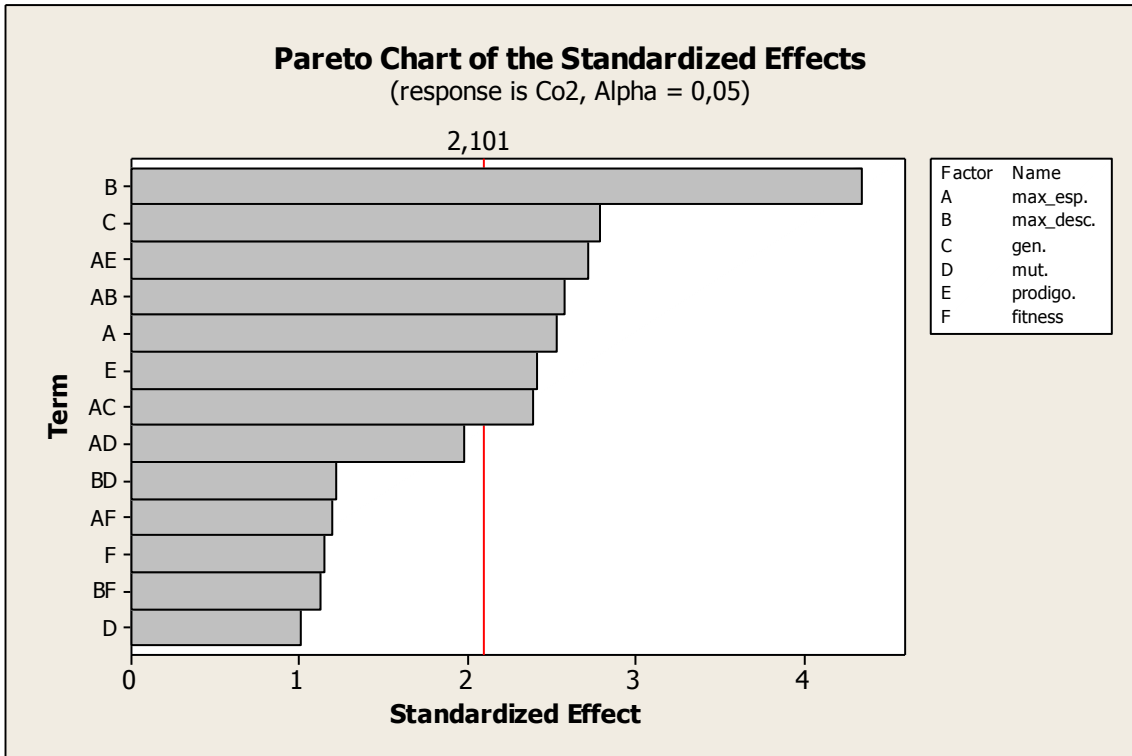


Gráfico 37 Frontera de Pareto de los efectos estandarizados sobre el CO<sub>2</sub>

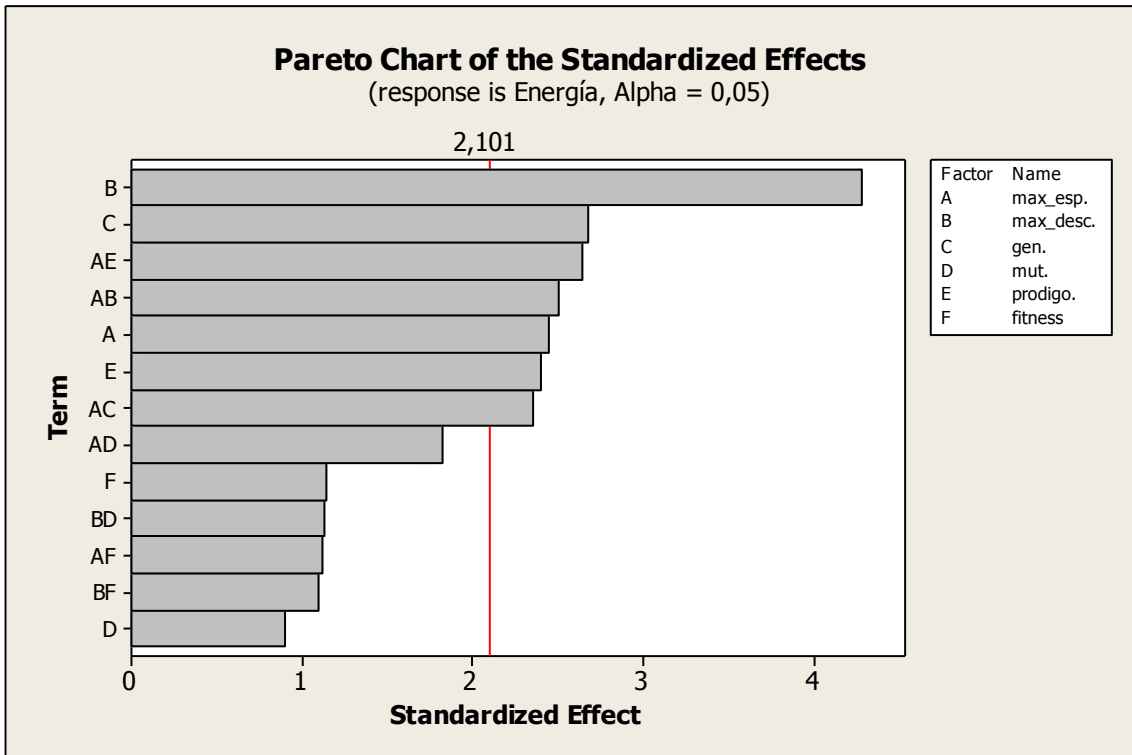


Gráfico 38 Frontera de Pareto de los efectos estandarizados sobre el Consumo Energético



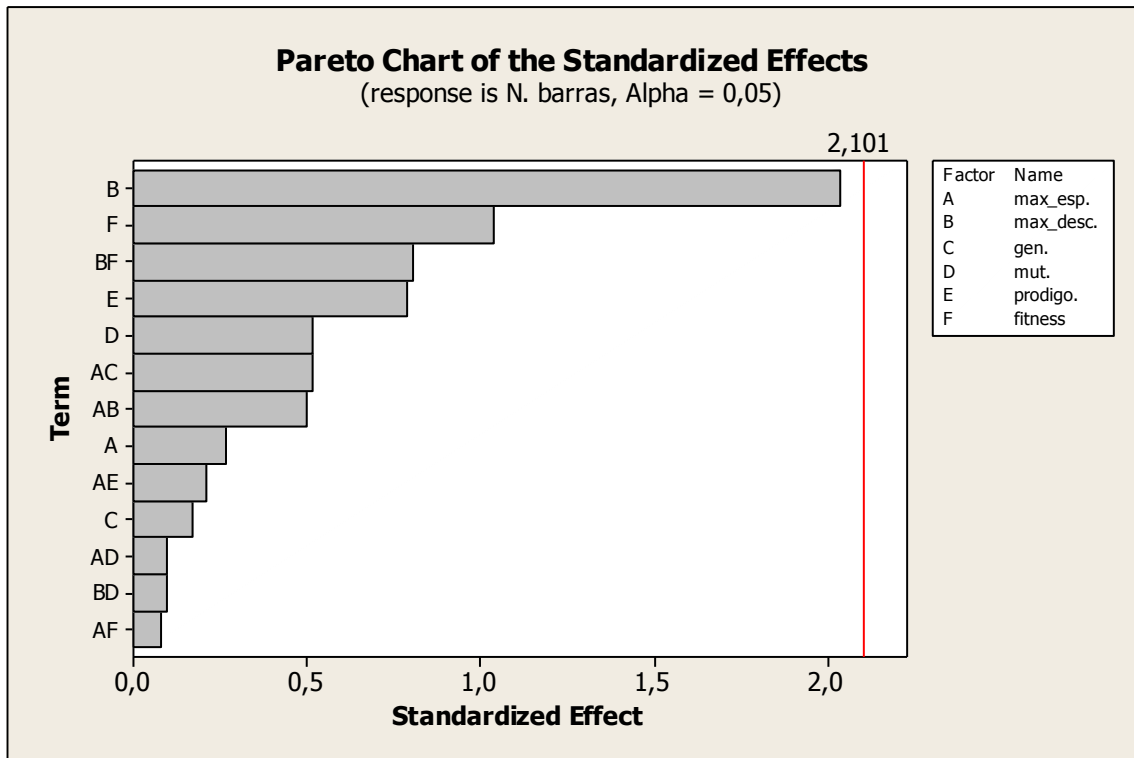


Gráfico 39 Frontera de Pareto de los efectos estandarizados sobre el Número de Barras

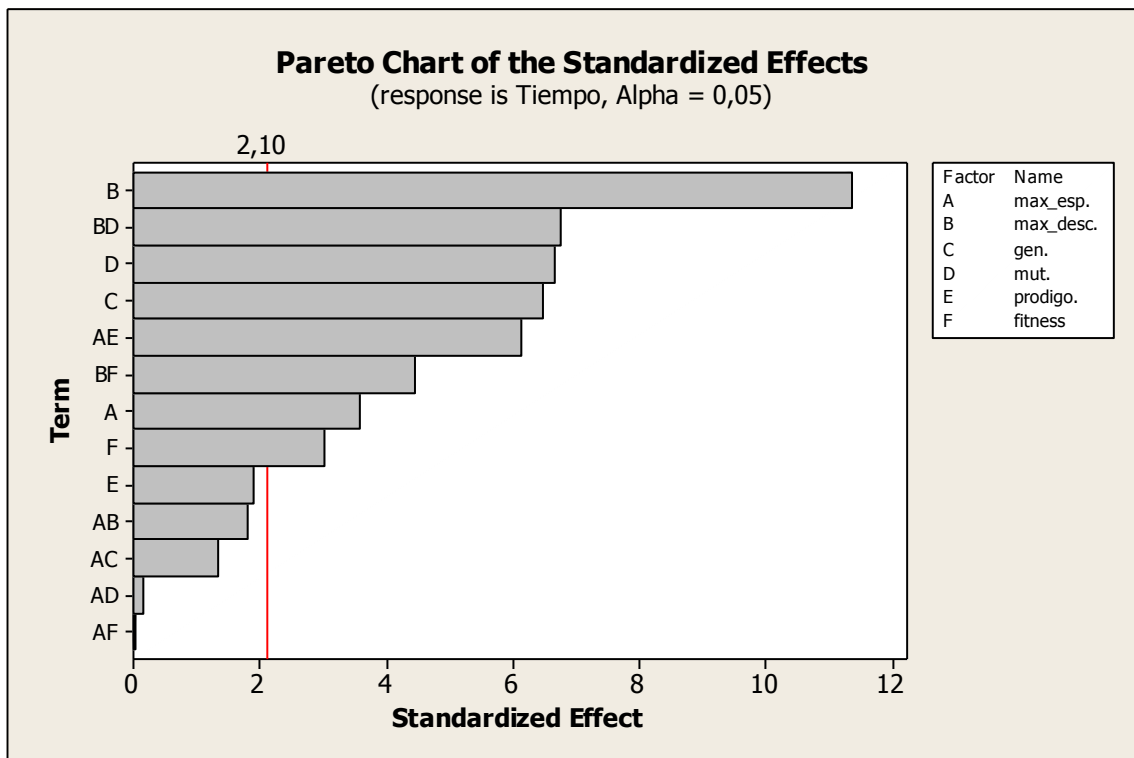


Gráfico 40 Frontera de Pareto de los efectos estandarizados sobre el Tiempo

En estos diagramas también se pueden observar los efectos secundarios, aquellos combinación de dos parámetros, que influyen. Se consideran influyentes todos aquellos efectos que superen la línea roja en los gráficos que determina la frontera de Pareto. En particular para nuestro

objetivo de determinar los valores de los parámetros nos preocupa el tiempo de ejecución, situado en el gráfico 40. De este frontera obtenemos que el factor más influyente en el tiempo es el número de descendientes, aunque todos excepto el porcentaje de hijo pródigo tienen una influencia significativa. Además también es relevante la interacción entre el número de descendientes y el porcentaje de mutación, debido a la creación de nuevos especímenes cuando se produce una mutación, la interacción entre el número de especímenes y el porcentaje de hijo pródigo, ya que es necesario generar un mayor número de especímenes, y la interacción entre el número de descendientes y la adecuación al umbral de Pareto, dada la dificultad de encontrar progenitores pertenecientes a este umbral con adecuaciones altas. En general son relevantes todas las interacciones que produzcan la utilización prolongada de los bucles anidados existentes en el algoritmo.

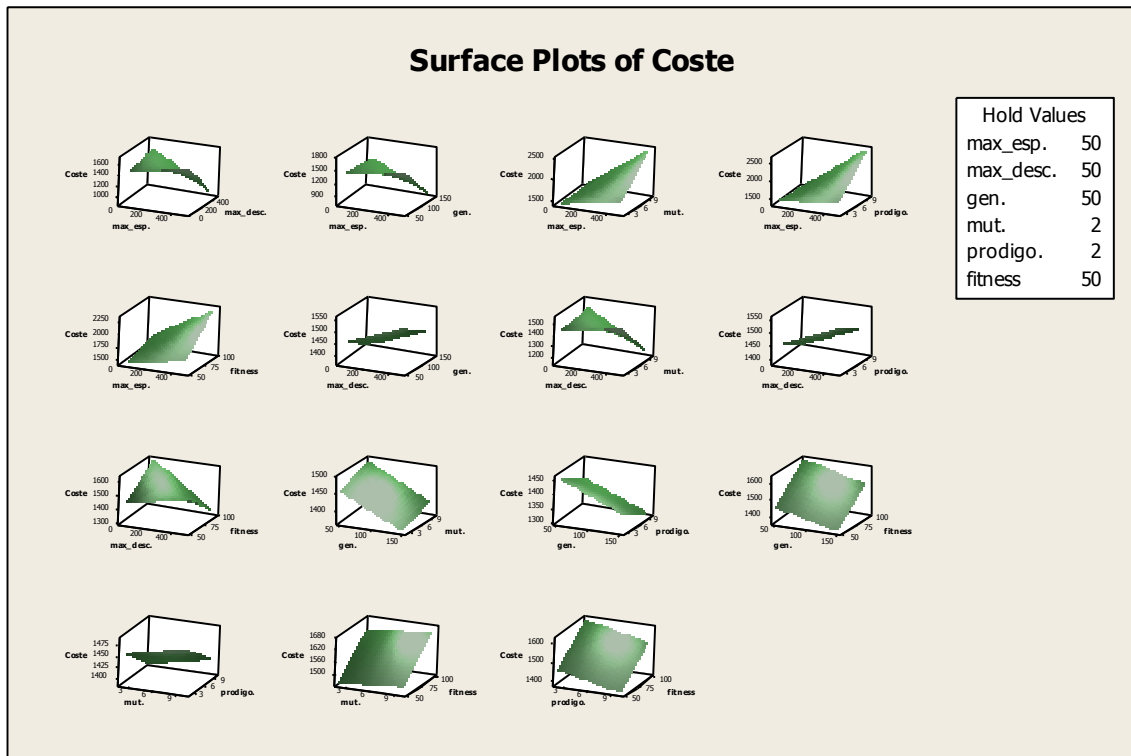


Gráfico 41 Superficies de respuesta de los parámetros sobre el Coste

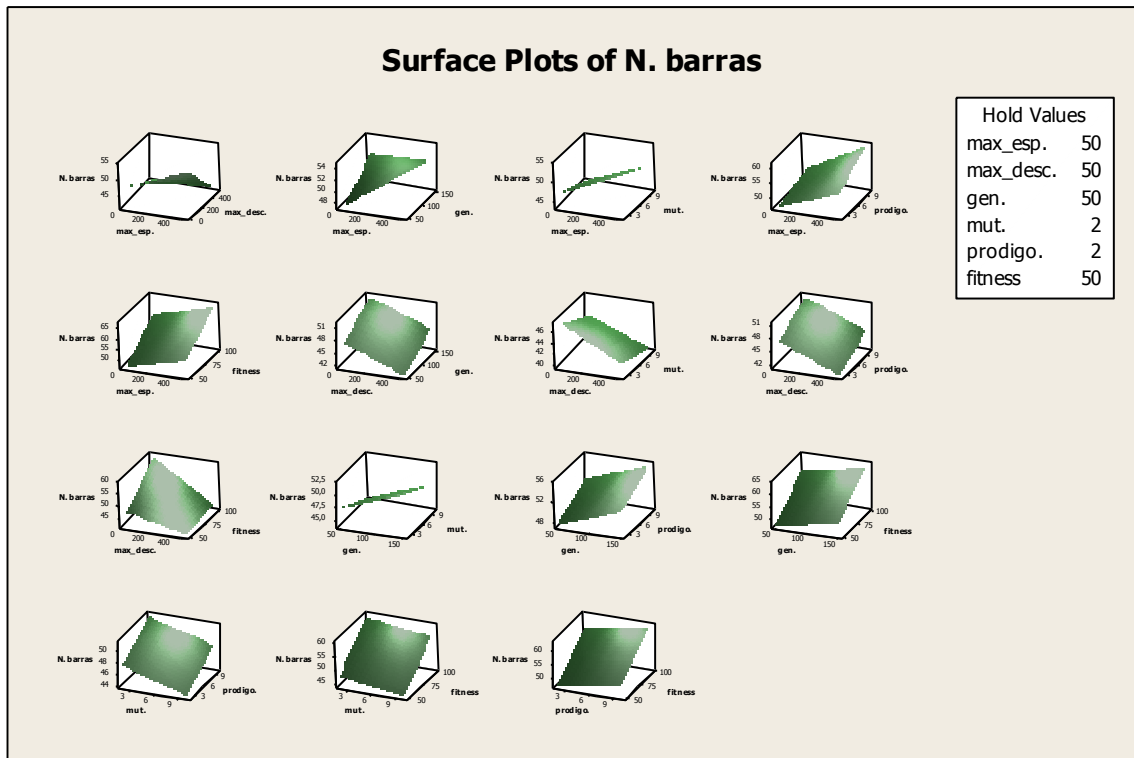


Gráfico 42 Superficies de respuesta de los parámetros sobre el Número de Barras

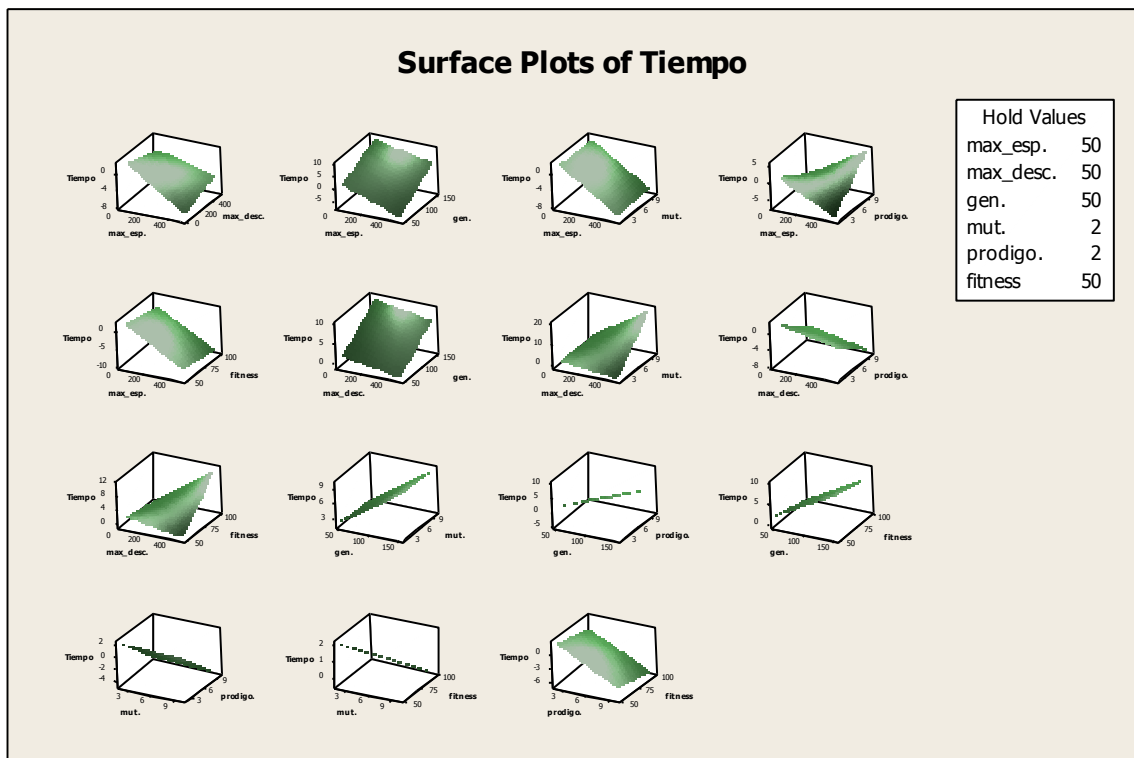


Gráfico 43 Superficies de respuesta de los parámetros sobre el Tiempo

Los gráficos anteriores muestran las superficies de respuesta de los diferentes criterios frente a los diferentes parámetros, se han obviado las superficies correspondientes a las emisiones de CO<sub>2</sub> y al consumo energético, ya que como se demuestra en el apartado 3 están directamente

relacionadas con el coste y no existe ninguna diferencia entre los resultados. Las diferentes curvaturas de estas superficies muestran en que punto o secuencia de puntos aumentar el valor del parámetro no produce una mejora significativa en el resultado final. Las superficies relativas al tiempo son generalmente planas ya que aumentar los valores produce un aumento en el tiempo de ejecución, por lo que es necesario analizar estas curvaturas detenidamente para establecer los puntos óptimos que nos proporcionen el mejor valor sin aumentar el tiempo de cálculo en exceso. Los valores seleccionados son los contenidos en la tabla 10 del apartado 3.1.