

Universidad de Huelva
Escuela Politécnica Superior
PROYECTO FIN DE CARRERA

INGENIERÍA INFORMÁTICA

Departamento de Ingeniería Electrónica, de Sistemas
Informáticos y Automática

Departamento Tecnología de la Información



**Universidad
de Huelva**



WiFiSimExtension

Planificación, optimización y despliegue de redes inalámbricas

HUELVA, DICIEMBRE 2012

Autores:

*Carlos Daniel Parreño Bonaño
Francisco Rodríguez Carrasco*

Tutores:

Francisco A. Márquez Hernández y Tomás de J. Mateo Sanguino

Dedicado a

Nuestros familiares

Y seres queridos

AGRADECIMIENTOS

Llegados a este punto de inflexión entendemos la importancia de un proyecto final de carrera. La tiene porque es el punto y final a una época de nuestras vidas irrepetible que jamás olvidaremos. Ha costado mucho esfuerzo y sacrificio, no solo nuestro, sino de muchas personas más, que nos han dado su ayuda incondicional, y a las cuáles no podemos dejar de agradecérselo.

A las familias, ya que sin sus apoyos y esfuerzos no habríamos gozado de la posibilidad de realizar una carrera universitaria ni de que este día tan especial llegase.

A Miguel Ángel Rodríguez Román, quien ha mostrado su predisposición a ayudarnos en todo momento, siempre con su simpatía y sus bromas.

A Antonio Peregrín Rubio, quien nos ha guiado desde el principio en la consecución de nuestras metas, proponiéndonos soluciones a los diversos problemas encontrados durante el desarrollo de nuestro proyecto.

A César Serrano López, por su amabilidad y ofrecimiento a ayudarnos en todo momento. Pero especialmente por su trabajo, ya que puede considerarse “padre” de nuestro proyecto y sin él, nada de esto hubiera sido posible.

Y sobretodo a Tomás Mateo Sanguino y Francisco Alfredo Márquez Hernández, con quienes nos sentimos en deuda, ya que gracias a su ayuda, entrega, ideas y horas de trabajo (inclusive en vacaciones), hemos podido realizar una proyecto muy completo en un espacio corto de tiempo, y a pesar de que a veces la meticulosidad de ambos nos haya supuesto un quebradero de cabeza, al final siempre ha dado sus frutos.

Todas estas personas han ayudado a que logros como el presente sean una gran satisfacción personal y un buen trabajo académico, pero además han ayudado a hacer del trabajo entre amigos algo posible.

Gracias a todos, de verdad.

RESUMEN

Hace ya algunos años que las comunicaciones inalámbricas están a la orden del día, tanto en el entorno doméstico como en oficinas, campos universitarios o redes a mayor escala. Las redes inalámbricas se han impuesto a las redes convencionales gracias a ventajas tales como el bajo coste, fácil instalación, gran movilidad y alcance de la red. Todo ello se traduce en definitiva en una mayor comodidad de operación para los usuarios de la red.

Por el contrario, entre algunas de las desventajas de las redes inalámbricas se pueden mencionar la facilidad de intrusión entre ciertas configuraciones menos seguras y una mayor tasa de error de bit (BER). Sin embargo, este proyecto se ha querido centrar en aquellos aspectos que disminuyen considerablemente la calidad de la señal o incluso la perdida total de ésta, debido a la distancia y al entorno en el que se encuentra.

A la hora de instalar una red WiFi dentro de un edificio, podemos encontrarnos problemas de cobertura, como la diferencia de calidad de señal en algunas zonas respecto a otras, e incluso zonas en que la señal ni siquiera alcanza a causa de la distancia o del entorno. Estos problemas están estrechamente relacionados, entre otros motivos, con la posición donde se encuentren los puntos de accesos de la red. Siendo este un inconveniente que no siempre se puede resolver mediante el uso de la lógica debido a otros factores incidentes.

Los principales objetivos de nuestro proyecto son la planificación y despliegue de redes inalámbricas, así como la optimización de éstas, mediante el cálculo de la mejor posición de uno o varios puntos de acceso dentro de un edificio. Para ello, se ha implementado una herramienta de diseño que obtiene la solución óptima, mediante algoritmos genéticos, teniendo en cuenta la mayor zona de cobertura y potencia de señal posible.

ABSTRACT

Since some years wireless communications seem to be at the order of the day, both as homes as in offices, college campuses or larger scale networks. Wireless networks have been imposed over conventional networks due to advantages such as low cost, easy installation, high mobility and network reach. This ultimately results in higher operating comfort for network users.

By contrast, among some of wireless networks disadvantages may be mentioned the easy intrusion between certain less secure settings and increased bit error rate (BER). However, this project has sought to focus on those aspects that significantly reduce the quality of the signal or even the total loss of this, because of the distance and the environment in which it is located.

When installing a WiFi network within a building, we can find coverage problems, as the difference in signal quality in some areas compared to others, and even areas where the signal does not even reach. These problems are closely related, among other reasons, to the position where the access points are on the network. Since this is a drawback that cannot always be solved by using logic due to incident factors.

The main goals of our project are the wireless networks planning, deployment, and optimization by working out the best position of one or more access points within a building. To achieve this end, we have implemented a design tool that obtains the optimal solution by using genetic algorithms, taking into account the increased coverage area and the strongest signal possible.

INDICE

1. Introducción.....	11
1.1 Introducción	11
1.2 Objetivos.....	12
1.3 Contenidos.....	15
2. Estado del Arte.....	17
2.1 WiFiSim	18
2.2 WiTuners	21
2.3 Airmagnet Survey.....	24
2.4 Ruckus Wireless Zone Planner.....	25
2.5 Ekahau Site Survey Professional	26
2.6 WiFiSimExtension.....	27
3. Memoria Descriptiva	29
3.1 Proceso de Desarrollo Software	29
3.2 Requisitos del proyecto	31
3.2.1 Diagrama de Casos de Uso	33
3.2.2 Especificación de los Casos de Uso	35
3.3 Arquitectura del Sistema	36
3.3.1 Diagrama de Clases	38
3.4 Implementación	43
3.4.1 Diseño Gráfico de Redes en Edificios	43
3.4.2 Diseño del Sistema <i>Best Solutions</i>	56
3.4.3 Algoritmo Genético	66
3.5 Despliegue	75
4. Experimentación y Resultados	77
4.1 Capa Física	77
4.2 Número APs	80
4.3 APs posiciones	78
4.4 Mutación	84
4.5 Remplazo	88
4.6 Diseño WiFi Real	91

5. Conclusiones y ampliaciones futuras	97
Anexo.....	103
A) Manual de Usuario.....	105
B) Teoría de Algoritmos Genéticos.....	127
C) Especificación Casos de Uso.....	138
D) Diagrama de Gantt.....	149
Referencias.....	151

CAPITULO 1: INTRODUCCIÓN

1.1. INTRODUCCIÓN

La idea de WiFiSimExtension surgió a partir de la aplicación WiFiSim creada por César Serrano López, alumno de la Universidad de Huelva, que tiene como objetivo el diseño y construcción de un simulador de redes inalámbricas basadas en el estándar IEEE 802.11. Dicha herramienta pretende ser utilizada para estudiar el comportamiento de una red en función de parámetros físicos como la distancia, la presencia de obstáculos y las características de los dispositivos de emisión y recepción.

Nuestra aplicación esta implementada modular e independientemente a WiFiSim, aunque está integrada en su misma interfaz, de modo que se pueden comunicar entre ellas y así facilitar la inserción de parámetros a WiFiSim, haciéndolo más dinámico.

Sin embargo, WiFiSimExtension posee su propia interfaz que funciona sin ningún tipo de dependencia externa. Gracias a la modularidad se puede realizar la aplicación completa sin influir en el código interno de la aplicación inicial, y de este modo otras ampliaciones serán posibles fácilmente.

WiFiSimExtension proporciona una interfaz gráfica donde se puede trabajar de forma dinámica sobre planos de edificios. Entre sus principales funciones se encuentra la integración con WiFiSim, lo que permite insertar de una forma sencilla e intuitiva los elementos desde nuestra interfaz para poder llevar a cabo la simulación. Por otra parte destacamos la búsqueda de una solución óptima al problema de colocación de puntos de accesos con el objetivo de conseguir la máxima cobertura posible sobre la red de diseño.

1.2. OBJETIVOS

Los objetivos de este proyecto los podemos dividir en dos grupos principales:

- ❖ Funcionales y académicos.

En este proyecto se pretende entre otras cosas suplir las desventajas de WiFiSim, añadiéndole dinamismo, funcionalidad e interactividad. También se pretende, tras un análisis pormenorizado del negocio de redes inalámbricas en general y del problema de localización de puntos de acceso para conseguir la máxima cobertura en particular, obtener un diseño que proporcione una solución que cumpla todos los requisitos y objetivos marcados, y a la vez atractiva y utilizable de cara al usuario.

Otro de los objetivos que se han marcado es el de darle a nuestra aplicación un enfoque académico, con idea de que pueda ser utilizada más adelante en clases prácticas de *Redes* o de *Inteligencia Artificial*. Ello proporcionará al usuario opciones adicionales para que se adentre en el comportamiento del sistema. Así como acceso al código fuente de la aplicación, el cual estará propiamente comentado y explicado.

- ❖ Profesionales y trabajo en equipo.

En tiempos difíciles como los actuales, conseguir trabajo como ingeniero informático ha llegado a ser una tarea difícil, y más aún conforme el tiempo pasa. Aunque el porcentaje de paro en nuestro campo sea inferior al del resto, la oferta laboral no es realmente buena, sobre todo salarialmente hablando, y conseguir una buena posición como profesionales recién titulados no es algo fácil. Sin embargo, hay algunas ofertas laborales que presentan buenas oportunidades donde, además de la correspondiente educación y conocimientos elevados de inglés, se demanda una alta capacidad de trabajo en equipo por encima incluso de otras cualidades.

Con la realización del proyecto WiFiSimExtension en grupo se ha pretendido adquirir experiencia a la hora de trabajar en equipo. Durante todo el proceso se ha intentado que el entorno de trabajo fuera similar al de una empresa real de software, en la que la planificación era y es un tema vital para la consecución de los éxitos.

Por lo tanto, algunos de los objetivos relacionados con el ámbito profesional y de trabajo en equipo han sido:

- ✓ Realizar una correcta colección de requisitos de los clientes que, en nuestro caso, dicho rol ha sido llevado a cabo por los profesores implicados en el proyecto.
- ✓ *Meetings* semanales entre los “desarrolladores” y los “clientes” con el objetivo de revisar y analizar los requisitos del producto (nuestra aplicación) de una forma iterativa. Ello ha permitido que el cliente pudiera seguir la evolución del proyecto e incluso añadir o cambiar algunos requisitos puntuales.
- ✓ Llevar a cabo una correcta organización y colaboración entre los componentes del grupo, para así conseguir explotar las ventajas que el trabajo en equipo proporciona.
- ✓ Un correcto seguimiento del ciclo de desarrollo de software utilizado para la obtención de experiencia en todas y cada una de sus tareas.
- ✓ Por supuesto uno de los objetivos principales ha sido la satisfacción del cliente y de los futuros usuarios.
- ✓ Una entrega temprana, sin llegar a influir en la consecución de todos los objetivos, que ha sido conseguida fruto de un trabajo en equipo, duro y constante. Esta entrega temprana ha sido uno de los objetivos principales del proyecto porque, como ha sido dicho anteriormente, conseguir empleo se está convirtiendo en el objetivo número uno en la sociedad española.

Así pues, el tiempo total dedicado a la realización de proyecto en la suma de todas sus fases es de aproximadamente 450 horas por alumno.

Diagrama de Gantt

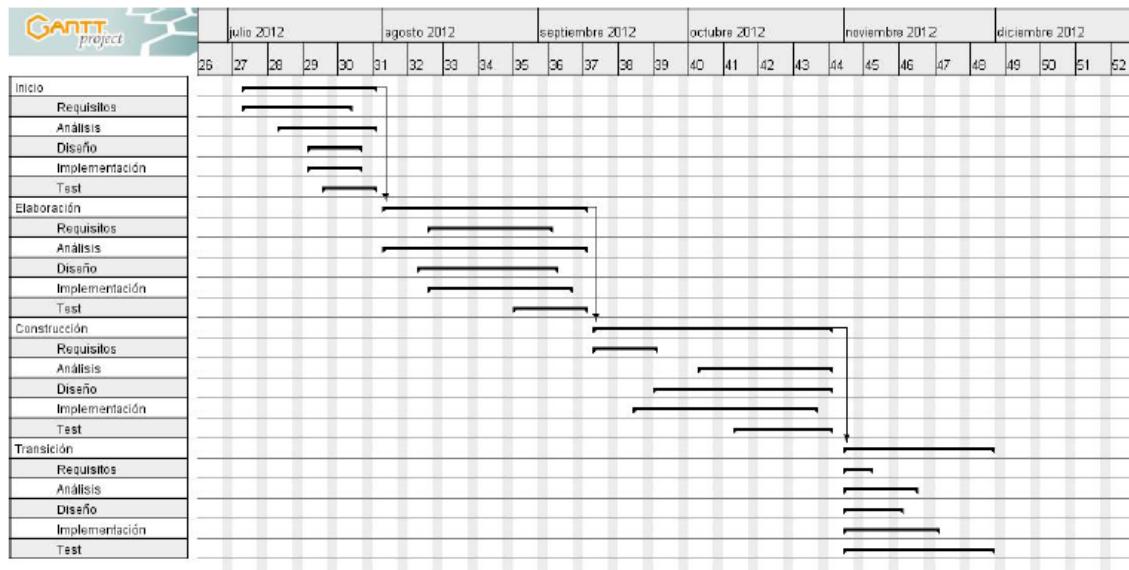


Imagen 1.1. Diagrama de Gantt

En el anexo [D] se adjunta la lista de tareas que componen cada una de las fases mostradas en el diagrama de Gantt.

1.3. CONTENIDOS

La memoria del proyecto contiene aparte del capítulo de introducción, un segundo capítulo llamado *estado del arte* en el que se hace una visión de otras aplicaciones en el mismo ámbito de aplicación que WiFiSimExtension. Se resaltarán tanto ventajas como inconvenientes y algunos otros aspectos que han resultado de interés.

Tras adentrarnos en el entorno sobre el que se desarrolla este proyecto se pasa a dar una visión de la memoria descriptiva, en la cobra especial importancia la gestión software de la aplicación ofreciendo una visión del análisis, diseño e implementación.

El siguiente capítulo estará relacionado con la experimentación del proyecto, con el objetivo de probar así la funcionalidad de éste y en el cual se expondrán los resultados con sus correspondientes interpretaciones.

En el último capítulo de la memoria se presentarán las conclusiones obtenidas y se hace una pequeña reseña a las posibles ampliaciones futuras.

La memoria finaliza con un anexo que contiene algunos puntos como el manual de usuario, teoría de algoritmos genéticos y referencias bibliográficas entre otros.

CAPÍTULO 2: ESTADO DEL ARTE

La tecnología inalámbrica (WiFi), se define como aquel sistema que posibilita conectar redes y ordenadores sin necesidad de cables. WiFi se utiliza como denominación genérica para los productos que incorporan cualquier variante de la tecnología inalámbrica IEEE 802.11, la cuál permite la creación de redes de trabajo inalámbricas (WirelessLAN).

Las redes inalámbricas han adquirido una gran importancia en los últimos años imponiéndose así al resto de tecnologías cableadas de uso doméstico. Además de proporcionar diferentes áreas de aplicación, garantiza la compatibilidad de funcionamiento entre equipos de distintos fabricantes.

Las ventajas son múltiples como por ejemplo, los precios, que se han reducido considerablemente, facilidad de instalación y funcionamiento, movilidad, transparencia (la red funciona como si hubiera cable), y gran escalabilidad, ofreciendo acceso a centenares de usuarios en la propia red.

WiFi tiene muchos puntos a su favor, pero también tiene otros que no lo están. Entre las desventajas de las redes inalámbricas podemos destacar las siguientes:

- ❖ Campo de seguridad: Existen algunos programas capaces de capturar paquetes trabajando con la tarjeta WiFi en modo promiscuo de forma que puedan calcular la contraseña de red y, de esta manera, acceder a ella. Las claves de tipo “WEP” son relativamente fáciles de conseguir con este procedimiento.
- ❖ Menor velocidad: la calidad de la señal WiFi es inferior a la que proporciona una conexión con cable, debido a las interferencias y pérdidas de señal que el ambiente puede acarrear.
- ❖ Ubicación geográfica: Dependiendo de la ubicación geográfica de un dispositivo y de un punto de acceso, la señal puede llegar con menor potencia o incluso no llegar a alcanzar dicha posición.

Existen varias aplicaciones que se dedican al estudio y la estimación del comportamiento WiFi sobre entornos determinados con el objetivo de identificar, reducir o incluso suprimir algunos de los inconvenientes que poseen las redes inalámbricas.

Este proyecto se centra concretamente en el estudio de dos de las desventajas citadas anteriormente (menor velocidad de transmisión de datos debida a la atenuación por obstáculos y el problema de la ubicación geográfica) mediante el análisis de algunas aplicaciones que se dedican a tal estudio.

2.1. WIFISIM - “Aplicación de la Universidad de Huelva”

El objetivo de *WiFiSim* (Wireless Fidelity Simulator) [1] consiste en la simulación del comportamiento de una red inalámbrica construida bajo el estándar IEEE 802.11, mediante la representación gráfica de la interacción de los dispositivos implicados según la configuración realizada por el usuario.

El sistema da soporte a los diferentes tipos de capa física de la actualidad (802.11, 802.11a, 802.11b, 802.11g, 802.11n), permite la configuración independiente de los dispositivos, soporta el mecanismo RTS/CTS, diferentes distribuciones para cargas de tráfico, y el problema del nodo oculto, donde los nodos pueden ser configurados para investigar el problema.

La interfaz de *WiFiSim* se divide en dos subsistemas:

- Subsistema *Configuración*:

El subsistema de configuración lleva a cabo todos los procesos dedicados a recoger los parámetros necesarios para lanzar una simulación, como por ejemplo la capa física, el modo de conexión, e insertar y modificar los datos de los componentes (nodos, puntos de accesos y obstáculos) de la simulación.

WiFiSimExtension

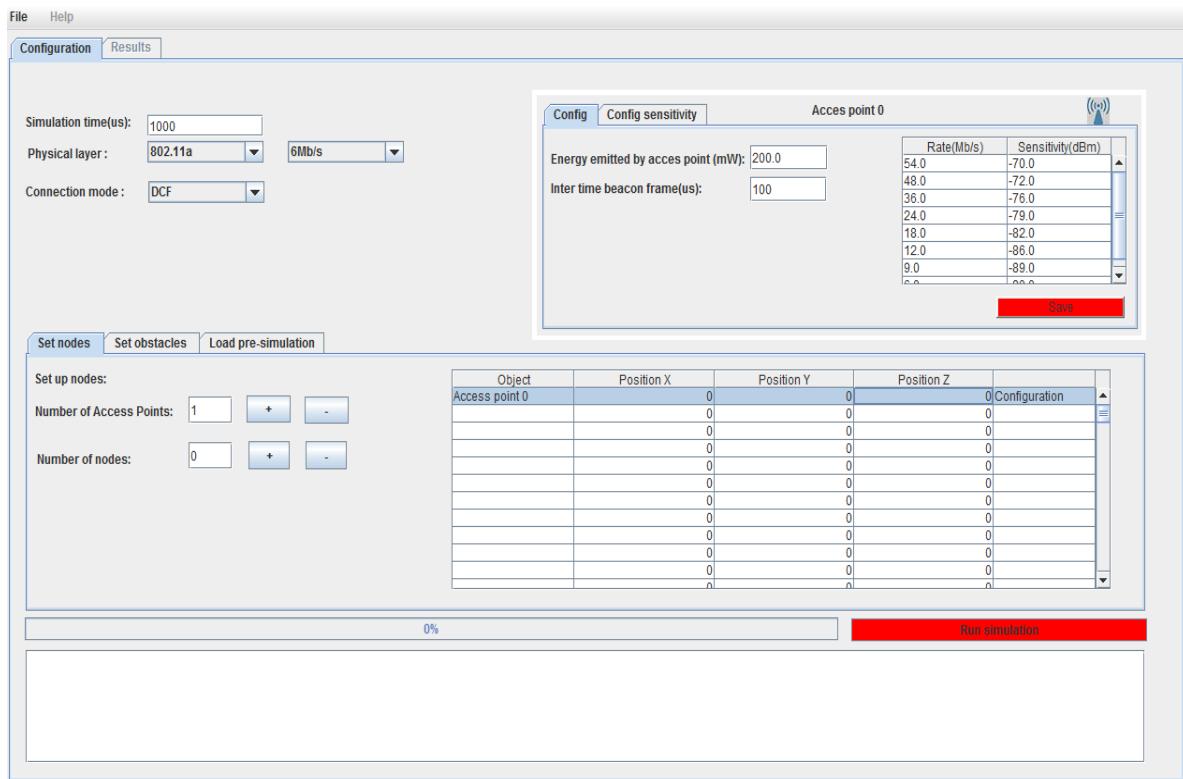


Imagen 2.1. Interfaz de WiFiSim. Pestaña de configuración

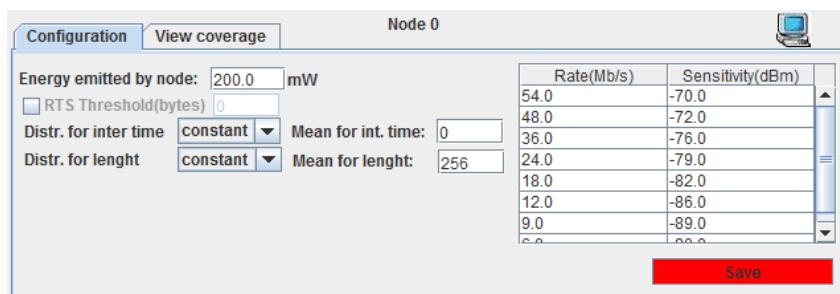


Imagen 2.2. Interfaz de Wifisim. Pestaña de configuración de Nodos

WiFiSimExtension

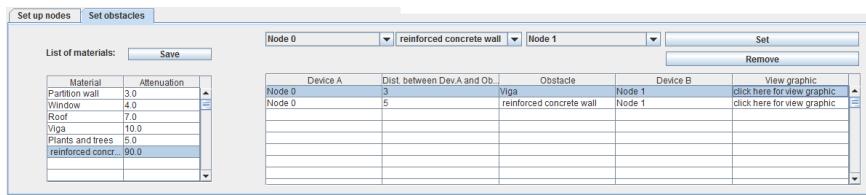


Imagen 2.3. Interfaz de WiFiSim. Pestaña de configuración de Obstáculos

- Subsistema *Results*:

El subsistema de resultados engloba todos los procesos encargados de recoger los resultados de la simulación y mostrarlos al usuario, bajo previa pre-simulación del subsistema de configuración.

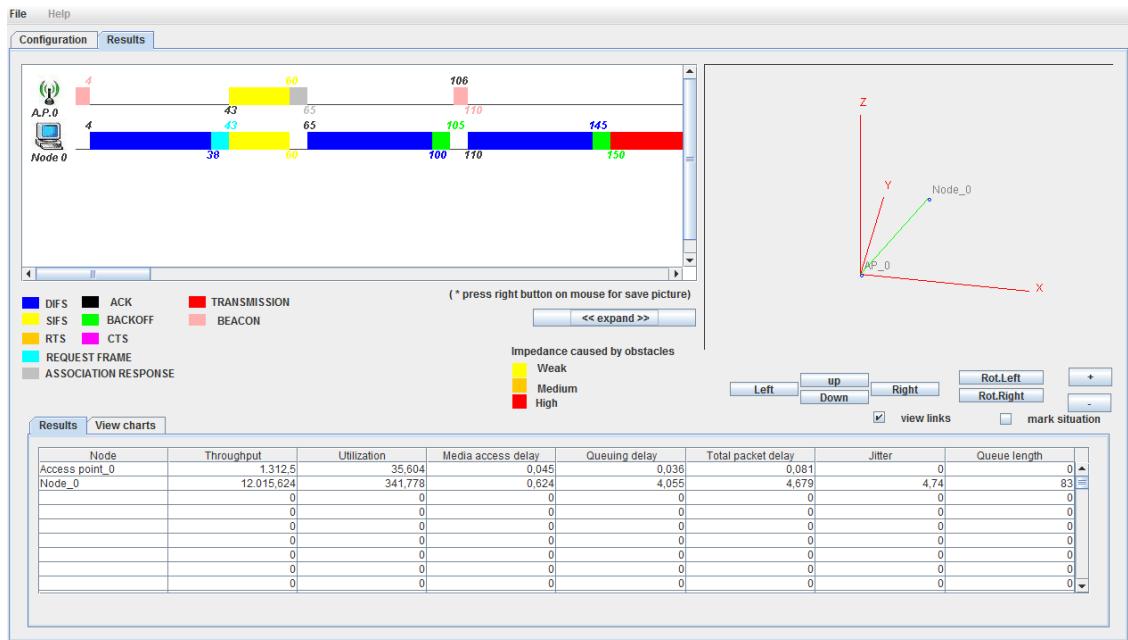


Imagen 2.4. Interfaz de WiFiSim. Pestaña de resultados

WiFiSim proporciona una interfaz que permite tanto el diseño y la construcción de redes inalámbricas basadas en el estándar IEEE 802.11, como el estudio del comportamiento de esta red en función de parámetros físicos como la distancia, la presencia de obstáculos y las características de los dispositivos empleados. También proporciona un estudio del comportamiento del algoritmo CSMA/CA entre dispositivos y del problema de nodo oculto.

Como contrapartida podemos resaltar algunos inconvenientes. Por ejemplo, WiFiSim dispone de una interfaz gráfica la cual no brinda mucha interacción con el usuario. Además podríamos destacar lo tedioso que resulta el proceso destinado a la introducción de datos, sin una idea gráfica de la posición de los elementos, pues cada nodo, AP, obstáculo y todas sus características deben de ser introducidas mediante el uso del teclado, pudiendo resultar una tarea larga y aburrida.

2.2. WiTuners

The logo for WiTuners features the word "WiTuners" in a bold, red, sans-serif font. Above the letter "i", there is a small icon consisting of three curved lines forming a signal or antenna pattern.

WiTuners [2] es un servicio de software online para el despliegue profesional de redes inalámbricas a gran escala. WiTuners incluye un conjunto de servicios WLAN deseados para planificar y mantener redes WiFi en su capacidad máxima, incluyendo también planificación, revisión y optimización WLAN. WiTuners dispone de dos servicios principales:

- ❖ Modulo: Planificación de despliegue avanzado.

Este módulo proporciona una planificación de una red inalámbrica en 3D sobre un edificio multi-planta a gran escala, aunque sin embargo su representación se realizará en 2D. El resultado se genera a través de una optimización del rendimiento, es decir, optimiza la cobertura y la intensidad para hallar la mejor solución.

WiFiSimExtension

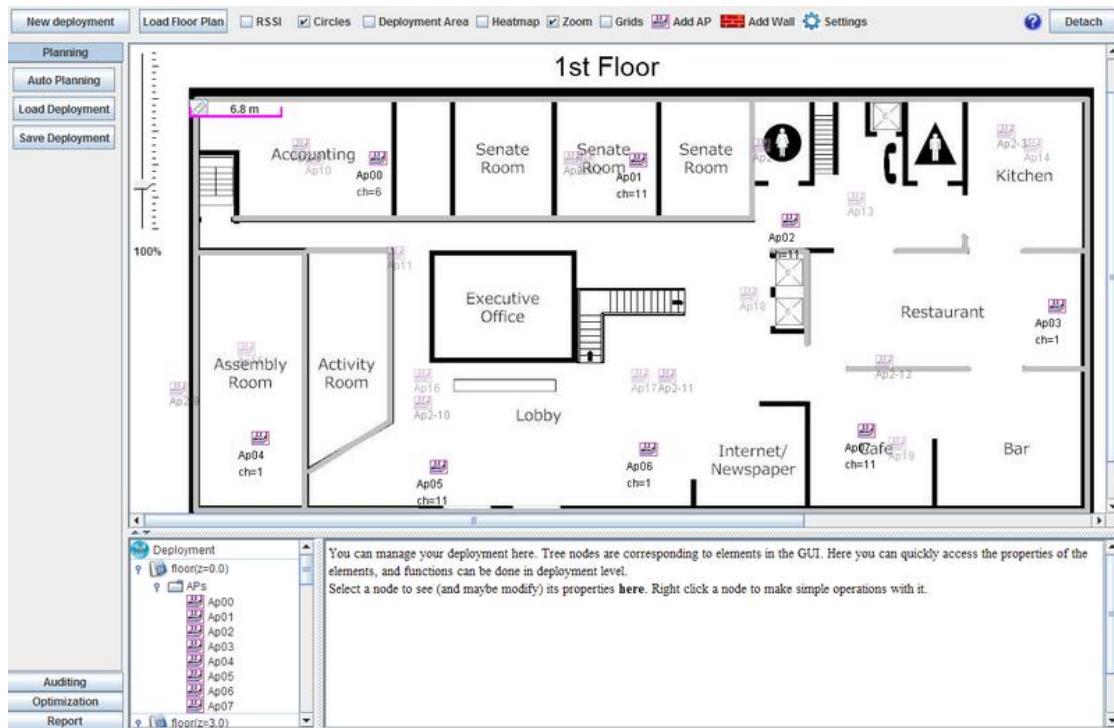


Imagen 2.5. Interfaz de WiTuners.

- ❖ Modulo: Audición y optimización del rendimiento en tiempo real.

Este módulo se encarga de revisar una red inalámbrica en tiempo real, alerta al administrador WLAN cuando detecte que el rendimiento de una WLAN se aproxima a su capacidad, sugiere ajustes óptimos para la condición actual de la WLAN e, incluso, permite una optimización para aumentar la capacidad de WLAN. Los propietarios WLAN se benefician así de un servicio de optimización automatizada con un coste de reducción de la administración.

WiFiSimExtension

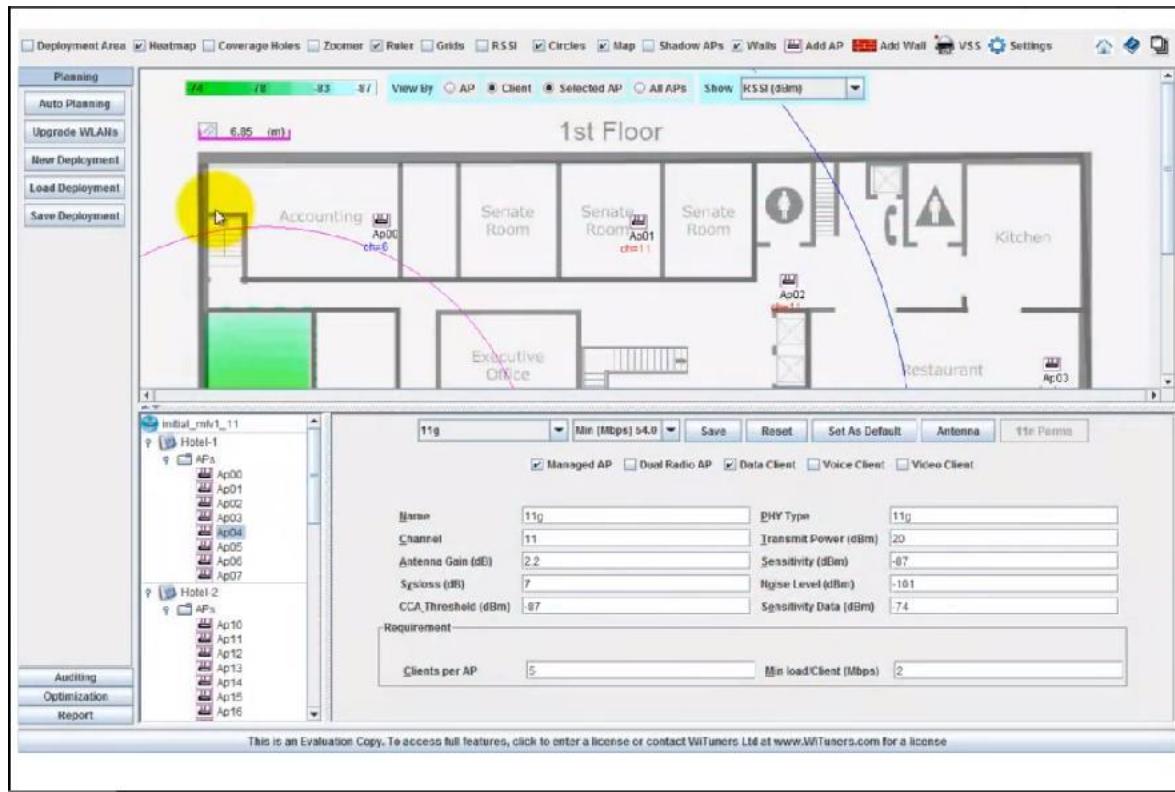


Imagen 2.6. Interfaz de WiTuners II.

WiTuners provee una aplicación útil y bastante atractiva para el usuario proporcionando además del diseño e instalación de la red WLAN, un mantenimiento y gestión de riesgos.

La interfaz de usuario es muy completa y ofrece una alta gama de posibilidades permitiendo de esta forma el diseño de una zona de trabajo WLAN mediante la construcción de edificios de varias plantas, indicación de zonas de cobertura deseadas, objetivos, etc. Sumistrando un informe con la solución del diseño.

Como principal desventaja, WiTuners es una herramienta de pago, de la que empresas que disponen de grandes redes WLAN podrían beneficiarse. Aunque puede que para empresas con pequeñas/medianas redes WLAN montadas en oficinas, la funcionalidad de WiTuners es excesiva y podría no ser rentable.

2.3. AIMAGNET SURVEY

AirMagnet Survey [3] proporciona una solución precisa del sector para la planificación y diseño de redes LAN inalámbricas IEEE 802.11 a/b/g/n para obtener un rendimiento, seguridad y cumplimiento óptimos. Calcula la cantidad, colocación y configuración ideales de puntos de acceso (AP) para lograr una implementación de WLAN correcta.

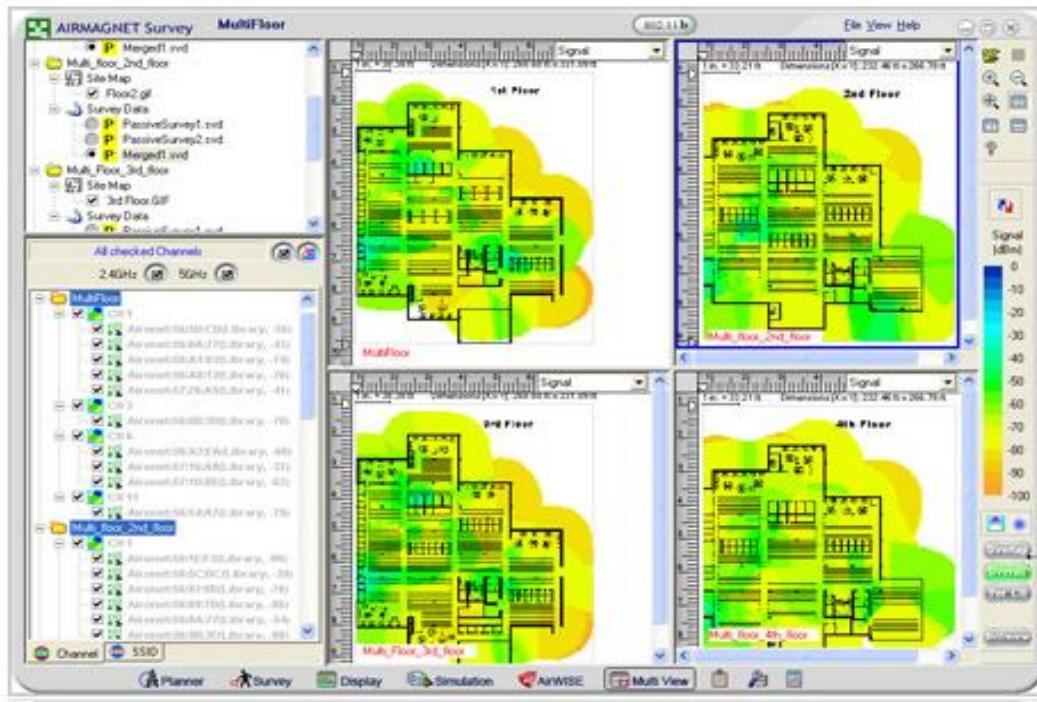


Imagen 2.7. Interfaz de Airmagnet Survey.

AirMagnet Survey provee también algunas características avanzadas que permiten a los usuarios que se integren como analizadores de la necesidad de conseguir WiFi en el edificio en cuestión, modelen situaciones previas, generen informes personalizados, verifiquen requisitos de red y planifiquen con detalle la capacidad del usuario final.

AirMagnet Survey es otra aplicación de pago y disponible en varias versiones dependiendo del usuario. Con una interfaz atractiva, de fácil uso, y con una funcionalidad muy amplia que facilita y mejora el diseño de redes.

2.4. RUCKUS WIRELESS ZONE PLANNER

Esta aplicación es otra destinada al diseño, estimación y despliegue de redes inalámbricas WLAN en entornos cerrados mediante una interfaz gráfica de fácil uso.

ZonePlanner [4] combina su funcionalidad con la herramienta de planificación de *AirMagnet* que es líder del sector.

Como podemos ver en la siguiente imagen, la interfaz es similar a la de otras herramientas de diseño WiFi, donde puedes diseñar la estructura de un edificio, zonas de cobertura, etc.

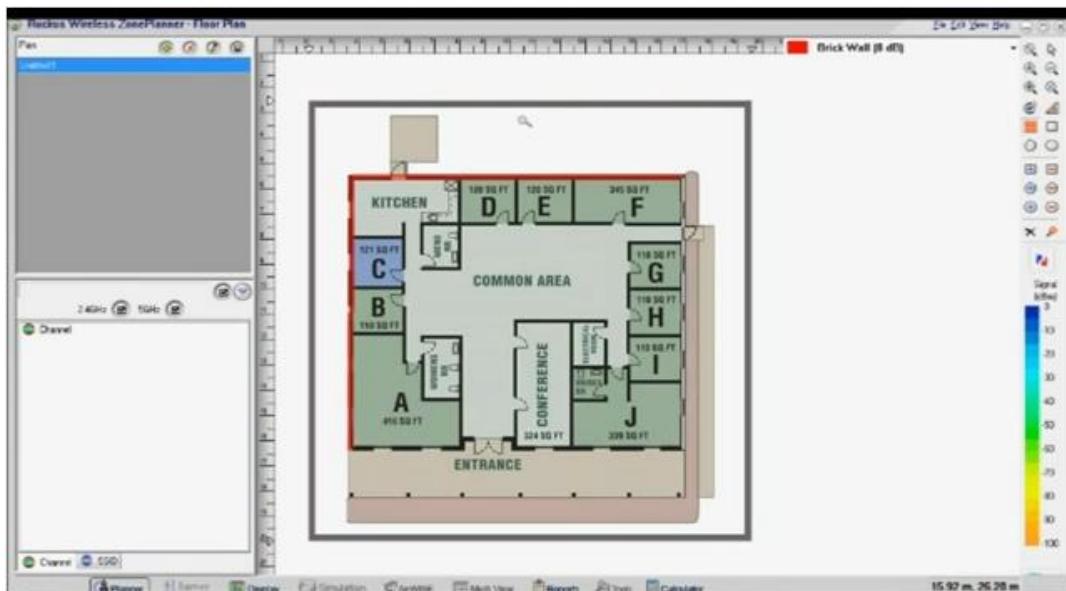


Imagen 2.8. Interfaz de Ruckus Wireless Zone Planner.

La gran diferencia que tiene esta herramienta respecto a otras, es una alta gama de antenas direccionales que provee, ésta enfoca y dirige la señal WiFi a través de los mejores caminos para aumentar el rendimiento de la señal y la capacidad del canal a la vez que mejora significativamente la fiabilidad.

2.5. EKAHAU SITE SURVEY PROFESSIONAL

En Ekahau Site Survey (ESS) [5] encontramos otra herramienta de fácil uso para planificación, estudio y administración de redes inalámbricas. ESS ofrece a los usuarios una visión de la cobertura y del rendimiento de la red en espacios cerrados, lo que les permite crear, mejorar y solucionar problemas de índole WiFi rápida y fácilmente.

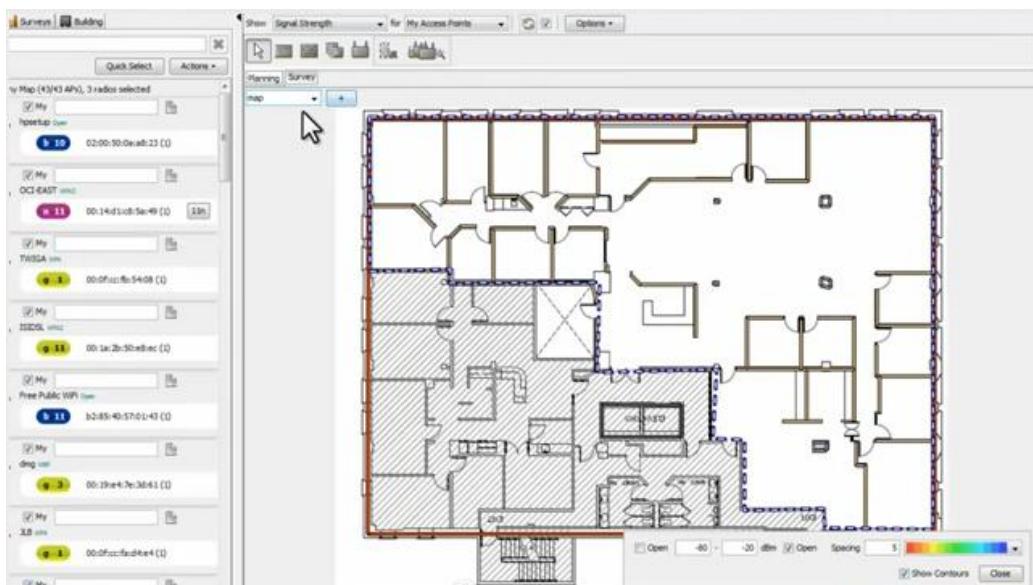


Imagen 2.9. Interfaz de Ekahau Site Survey Professional.

Algunas de sus características son las siguientes:

- Trabaja sobre cualquier red 802.11.
- Proporciona un diseño y planificación WiFi en 3D.
- Proporciona estudios de redes GPS en espacios abiertos.
- Realiza un análisis de espectro que detecta y localiza interferencias que no son tipo WiFi.
- Provee un informe personalizable de la red.
- Completa integración con Cisco Prime NCS / WCS.

2.6. WIFISIMEXTENSION - “Ampliación de WiFiSim”

WiFiSimExtension es una aplicación pensada para mejorar algunas desventajas de WiFiSim. Con esta aplicación se pretende agilizar la tarea de configuración del entorno para realizar la simulación posterior en WiFiSim, para lo cuál se ha diseñado una interfaz gráfica capaz de añadir los elementos WiFiSim (nodos, puntos de acceso y obstáculos) dinámica y gráficamente, haciendo la labor más amena y rápida. Una vez configurado el entorno, sus elementos y sus parámetros son exportables a WiFiSim para su simulación.

Dicha interfaz proporciona toda la gestión de los elementos necesarios. La creación, edición y eliminación de estos elementos sobre el plano de un edificio de manera dinámica hace de WiFiSim una aplicación mucho más completa y útil.

La gran ventaja respecto a WiFiSim es el hecho de que mientras WiFiSim realiza un estudio del entorno estático que el usuario configura, WiFiSimExtension calcula dentro del entorno las posiciones de los puntos de acceso para que la solución obtenida sea la óptima o bien esté lo más cercana posible a esta. Además tiene otras muchas opciones como se verá más adelante que aportan funcionalidad, interacción con el usuario y mayor atraktividad.

En cuanto a las aplicaciones “WiTuners” y “Airmagnet Survey” son dos aplicaciones muy completas que han sido referentes en algunos aspectos. No obstante, WiFiSimExtension está pensada para el ámbito académico proporcionando la posibilidad de configurar parámetros internos y ver el comportamiento del algoritmo genético que usa el sistema.

Por otro lado, WiFiSimExtension calcula una solución con el número de puntos de acceso indicados, no imponiendo de esta manera una condición en el diseño de la instalación real del cliente. Así, WiFiSimExtension permite una solución que se pueda ajustar a la posibilidad económica del cliente.

CAPITULO 3: MEMORIA DESCRIPTIVA

3.1 PROCESO DE DESARROLLO SOFTWARE

Para el desarrollo de nuestro proyecto software nos hemos decantado por seguir la ideología del *Proceso unificado*, más bien conocido como *RUP* [10] y el Lenguaje Unificado de Modelado UML [8].

RUP (Proceso Unificado) es un proceso de desarrollo de software que junto con el *Lenguaje Unificado de Modelado UML* constituye una metodología estándar muy utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos.

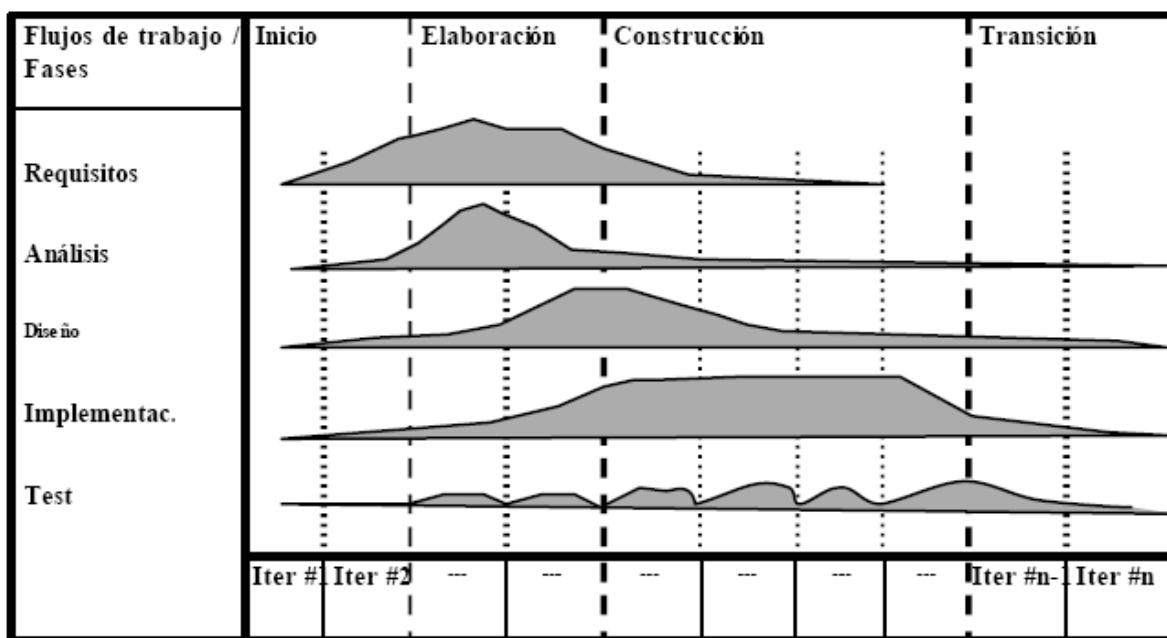


Imagen 3.1. Ciclo de vida RUP

La metodología RUP es más apropiada para proyectos grandes (aunque también pequeños), dado que requiere un equipo de trabajo capaz de administrar un proceso complejo en varias etapas. En proyectos pequeños, es posible que no se puedan cubrir los costos de dedicación del equipo de profesionales necesarios. Por otro lado, en lo que se refiere a la metodología esta comprende tres fases claves: Dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental.

En lo referente a dirigido por los casos de uso, está enfocado hacia el cliente y se utilizan con algunas modificaciones tal vez, hasta la disciplina de pruebas, en la cual, un caso de uso puede a su vez tener uno o más casos de prueba.

RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en las distintas actividades. En la Figura se muestra cómo varía el esfuerzo asociado a las disciplinas según la fase en la que se encuentre el proyecto RUP.

Las primeras iteraciones se enfocan hacia la comprensión del problema y la tecnología, la delimitación del ámbito del proyecto, la eliminación de los riesgos críticos, y al establecimiento de una (Línea Base) de la arquitectura.

Durante la fase de inicio las iteraciones hacen mayor énfasis en actividades de modelado del negocio y de requisitos.

En la fase de elaboración, las iteraciones se orientan al desarrollo de la línea base de la arquitectura, abarcan más los flujos de trabajo de requisitos, modelo de negocios, análisis, diseño y una parte de implementación orientado a la línea base de la arquitectura.

En la fase de construcción, se lleva a cabo la construcción del producto por medio de una serie de iteraciones.

Para cada iteración se seleccionan algunos Casos de Uso, se refinan su análisis y diseño y se procede a su implementación y pruebas. Se realiza una pequeña cascada para cada ciclo. Se realizan iteraciones hasta que se termine la implementación de la nueva versión del producto.

En la fase de transición se pretende garantizar que se tiene un producto preparado para su entrega a la comunidad de usuarios.

Como se puede observar en cada fase participan todas las disciplinas, pero dependiendo de la fase el esfuerzo dedicado a una disciplina varía.

3.2. REQUISITOS DEL PROYECTO

Los requisitos, son principalmente los siguientes:

Requisito-1: El sistema deberá proporcionar una interfaz gráfica en la que se pueda gestionar los elementos de la aplicación WiFiSim: Nodos, puntos de acceso y obstáculos.

Requisito-2: El sistema deberá proporcionar una interfaz gráfica que permita el diseño arquitectónico de edificios a bajo nivel, así como cargar diseños ya establecidos.

Requisito-3: El sistema deberá mantener una comunicación directa con la interfaz de Wifisim. La comunicación será de doble sentido.

Requisito-4: El sistema deberá proporcionar una interfaz atractiva, interactiva y con amplia funcionalidad para el usuario.

Requisito-5: El sistema deberá ser capaz de calcular y devolver una solución convergente al problema de colocación de puntos de accesos en un edificio, de manera que exista la máxima cobertura posible.

Requisito-6: El sistema deberá proporcionar un sistema de ficheros que facilite la gestión de funciones para guardar y cargar proyectos y soluciones.

El proyecto gestiona los siguientes datos:

❖ **Building (Construcción):**

1. **Número de plantas.**
2. **Altura de plantas.**
3. **Escala del plano del edificio.**

❖ **Floor (Planta):**

1. **Imagen planta.**
2. **Número de elementos.**
3. **Número de planta**
4. **Lista de elementos.**

❖ **Node (Nodo (Host)):**

1. **Nombre.**
2. **Posición, coordenadas (x,y,z).**
3. **Potencia.**

❖ **Access Point (AP) (Punto de Acceso):**

1. **Nombre.**
2. **Posición, coordenadas (x,y,z).**
3. **Potencia.**

❖ **Obstacle (Obstáculo):**

1. **Nombre.**
2. **Punto Origen, coordenada (x, y, z).**
3. **Punto Final, coordenada (x, y, z).**
4. **Tipo.**
5. **Atenuación.**
6. **Forma.**
7. **Color.**
8. **Ancho.**

❖ **APSolution (Solución de cada AP):**

1. **Nombre.**
2. **Posición, coordenadas (x,y,z).**
3. **Zona de cobertura.**
4. **Potencia.**

3.2.1. DIAGRAMAS DE CASOS DE USO

Los casos de usos del sistema están divididos en cuatro subsistemas:

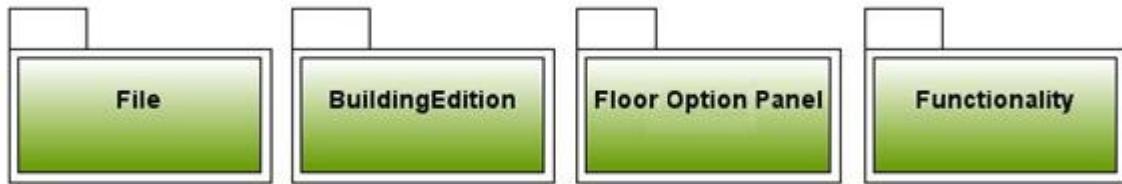


Imagen 3.2. Diagrama de Subsistemas

File:

Este subsistema soporta los casos de uso encargados de la gestión de ficheros de los proyectos de WiFiSimExtension. Contiene casos de usos, para guardar y cargar proyectos y soluciones.

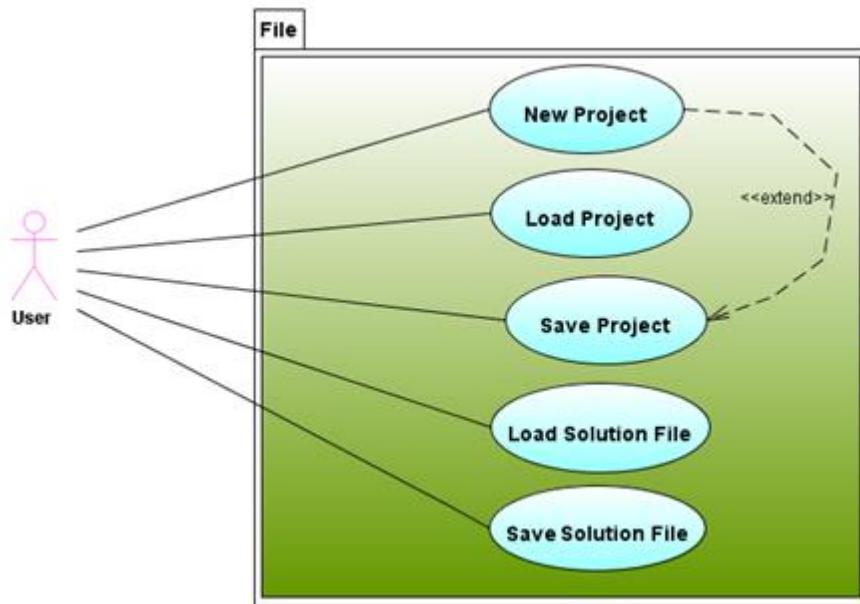


Imagen 3.3. Diagrama Casos de Uso. Subsistema File

BuildingEdition:

Este subsistema contiene todos los casos de usos engargados en la edición del edificio.

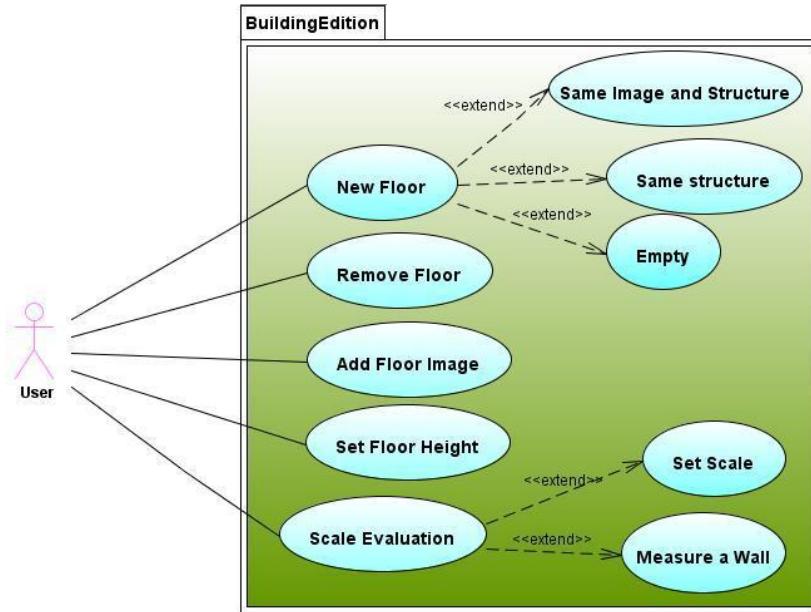


Imagen 3.4. Diagrama Casos de Uso. Subsistema Building Edition

FloorOptionPanel:

Este subsistema contiene los casos de uso relacionados con la parte gráfica de la interfaz.

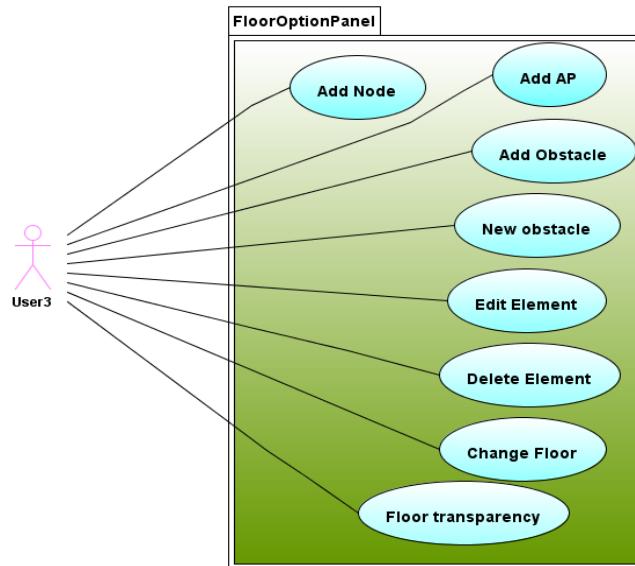


Imagen 3.5. Diagrama Casos de Uso. Subsistema FloorOptionPanel

Functionality:

Es el subsistema de los casos de uso que llevan a cabo la funcionalidad principal del proyecto, como la exportación a WiFiSim y la búsqueda de la mejor solución.

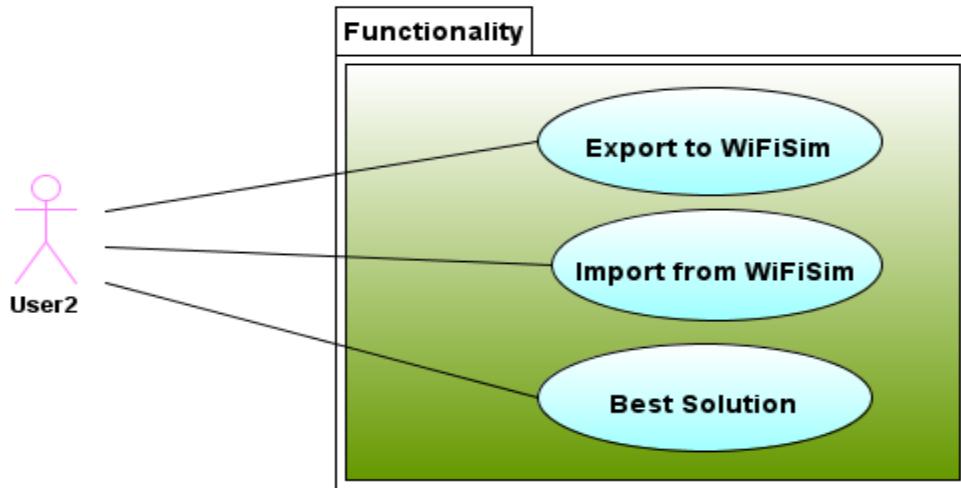


Imagen 3.6. Diagrama Casos de Uso. Subsistema Extension Functionality

3.2.2. ESPECIFICACIÓN DE LOS CASOS DE USO.

Debido a la extensión de este punto se ha optado por desarrollar la especificación de los Casos de Uso se encuentra en el *ANEXO*.

3.3. ARQUITECTURA DEL PROYECTO (Diseño)

La arquitectura del proyecto esta dividida en tres diferentes capas (Presentación, lógica de negocio y preservación de datos), según indica el patrón de diseño software *DAO* (*Data Access Object*) [9]:

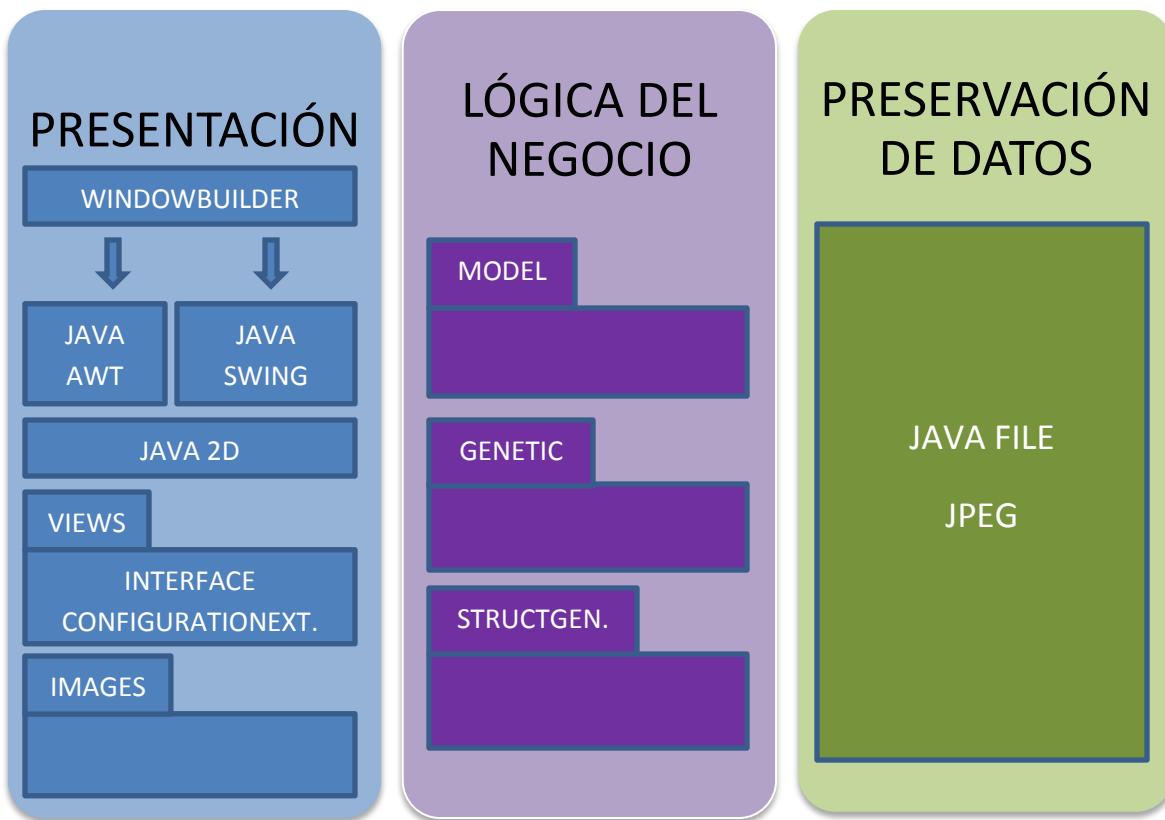


Imagen 3.7. Esquema de la Arquitectura.Patrón DAO

CAPA DE PRESENTACIÓN:

Debido a que este proyecto es una continuación de la plataforma WiFiSim, nos ha condicionado la elección del entorno de desarrollo software utilizado en nuestro proyecto ha sido *Eclipse*, y el lenguaje de programación ha sido Java, por ser un lenguaje multiplataforma y totalmente orientado a objetos, y para el diseño nos hemos basado en el patrón DAO (Data Access Object) para facilitar así la gestión de distintas fuentes de datos.

Para el desarrollo de la interfaz de usuario hemos utilizado el *FrameWork* llamado *WindowBuilder* [6], ya que es una herramienta con muchas posibilidades y de fácil uso,

compuesta por *SWT Designer* y *Swing Designer* y hace muy fácil la tarea de crear “aplicaciones Java GUI” sin perder mucho tiempo escribiendo código.

Para las funciones de diseño gráfico se ha utilizado la librería de JAVA llamada *Java2D* [7].

La capa de presentación también está compuesta por los paquetes *Views* y *Images* que contienen las clases de las interfaces (Interface y ConfigurationExtension) el primero y las imágenes fuentes del proyecto el segundo.

CAPA DE LA LOGICA DEL NEGOCIO:

Capa que contiene todos los paquetes con las clases encargadas de representar a las entidades de nuestro proyecto y realizar toda su funcionalidad. Esta capa se comunica con la capa de presentación, para recibir las solicitudes de las interfaces, y hace uso de los servicios de la capa de datos, para guardar y recuperar datos. Es aquí donde la aplicación reciba toda la funcionalidad que no sea dibujar (donde también interviene la capa de presentación),

CAPA DE PRESERVACIÓN DE DATOS:

Es la encargada de gestionar datos tanto de archivos “.jpeg” (carga de imágenes), como de archivos con la extensión de este programa (o bien “.txt”) para la carga o almacenamiento de planos dibujados anteriormente, o de soluciones halladas.

Se podría indicar la capa de presentación como la que interactúa con el usuario, la de lógica de negocio como la que realiza todas las operaciones y la preservación de datos como la base de datos de nuestro programa.

3.3.1.DIAGRAMA DE CLASES

Nuestro proyecto esta dividido en estos cinco paquetes, de los cuales cuatro de ellos contienen las clases de nuestra aplicación, a excepción del paquete *Images* que solo contiene imágenes.

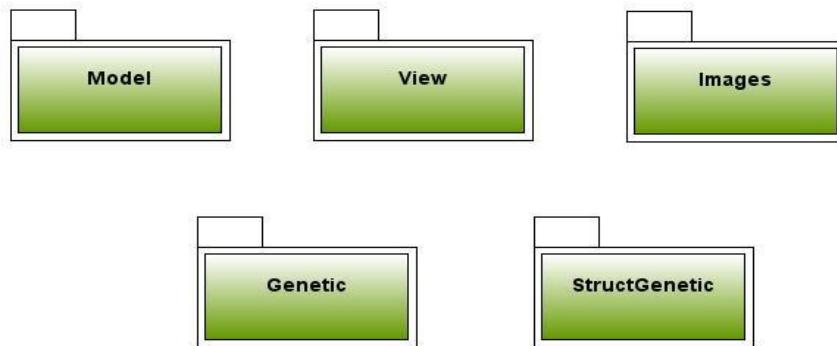


Imagen 3.8. Paquetes del sistema

Diagrama de clases paquete “Model”

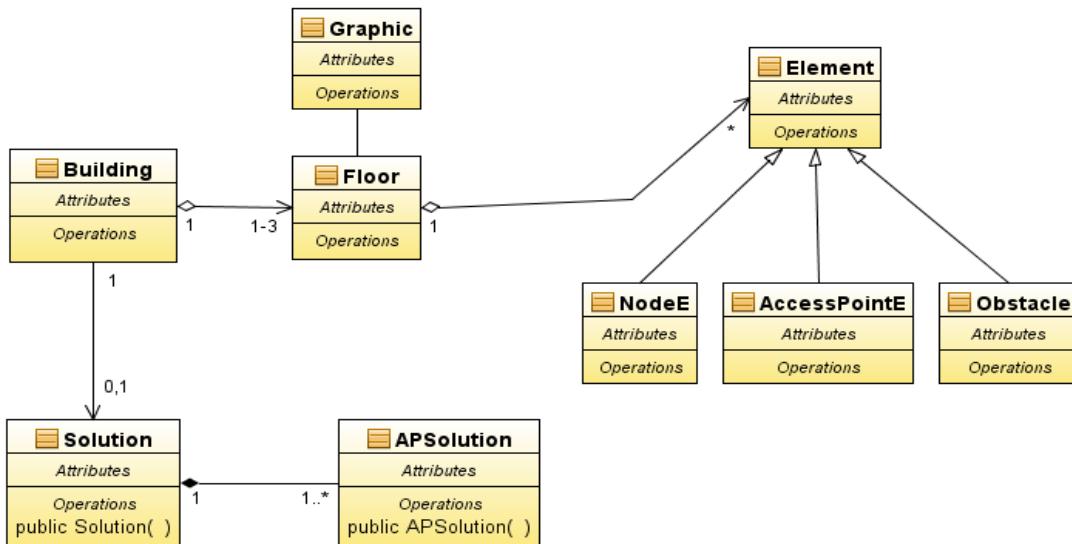


Imagen 3.9. Diagrama de clases “Model“

Descripción de las clases del paquete “Model“

Graphic: Clase encargada de la parte gráfica de aplicación. Contiene la librería Java2D, y es la encargada de llamar a los métodos “pintar“ para mostrar los elementos gráficos de la interfaz.

Building: Clase que representa al “edificio“ donde se va a trabajar con nuestra aplicación. Contiene una lista de objetos “Floor“ uno por cada planta que tenga el edificio. También contiene una lista de objetos Solution, en caso de que se haya calculada alguna solución.

Floor: Clase que representa cada planta de un edificio. Contiene una lista de objetos tipo “Element“ y es el objeto que la clase “Graphic“ se encarga de dibujar sobre el gráfico de la interfaz.

Element: Es la clase padre de los elementos con los que va a interactuar el usuario.

NodeE: Clase que hereda de Element. Es la encargada de representar a los nodos del sistema.

AccessPointE: Clase que hereda de Element. Es la encargada de representar a los puntos de acceso del sistema.

Obstacle: Clase que hereda de Element. Es la encargada de representar a los obstáculos del sistema.

Solution: Clase que representa una solución del sistema. Contiene una lista de objetos tipo “APSolution“ y el porcentaje de celdas que esa lista de objetos cubre.

APSolution: Clase que representa un punto de acceso de una solución encontrada por WiFiSimExtension.

WiFiSimExtension

Diagrama de clases paquete “View”

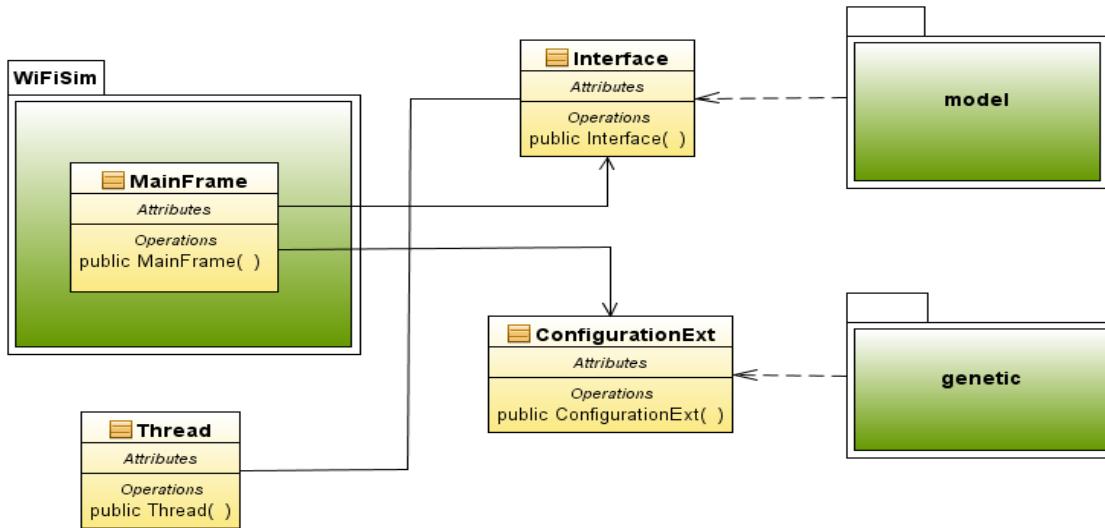


Imagen 3.10. Diagrama de clases “View”

Descripción de clases del paquete “View”

MainFrame: Es la clase que actúa como interfaz principal de WiFiSim. Contiene un objeto tipo “Interface” y otro tipo “ConfigurationExt” para así comunicarse con las interfaces de WiFiSimExtension.

Interface: Es la clase encargada de las funciones de la interfaz de WiFiSimExtension. Se comunica con el paquete “Model” para llevar a cabo toda su funcionalidad.

ConfigurationExt: Es la clase encargada de la configuración del algoritmo genético de WiFiSimExtension.

Thread: La clase Thread es la encargada de llamar a las funciones del algoritmo genético del sistema. Este algoritmo puede ser parado por el usuario por lo que necesitamos dos subprocesos, uno que se encarga de los cálculos del genético y otro para recoger las peticiones del usuario.

Diagrama de clases paquete “StructGenetic“

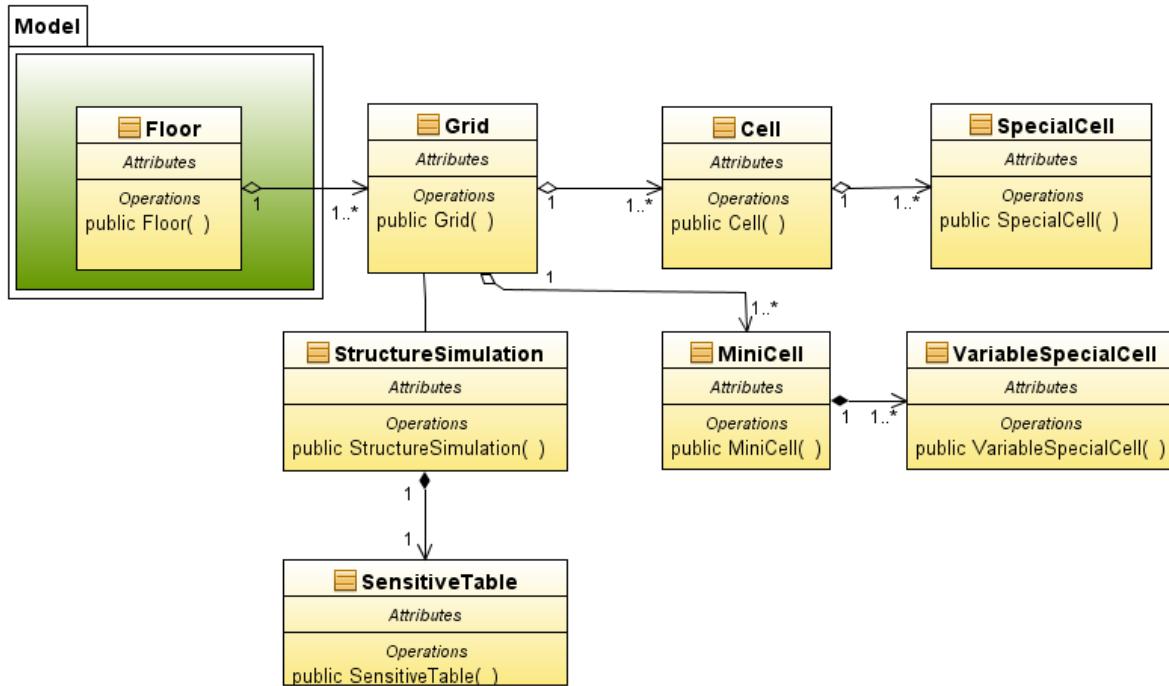


Imagen 3.11. Diagrama de clases “StructGenetic“

Descripción de clases del paquete “StructGenetic“

Grid: Esta es la clase principal del paquete “StructGenetic“. Es la encargada de identificar los límites del plano, y estará compuesto por los diferentes tipos de celdas de la estructura del edificio.

Cell: Clase que representa una celda regular del “grid“, entendiendo como regular una celda de dimensiones que representa un metro cuadrado acorde con la escala del edificio.

MiniCell: Clase que representa una celda irregular del grid, entendiendo como irregular una celda que ha tenido que ser dividida por algún obstáculo.

SpecialCell: Clase que representa una zona especial regular (la cual referencia a una celda), para el cálculo del genético.

VariableSpecialCell: Clase que representa una zona especial irregular (la cual referencia a una minicelda), para el cálculo del genético.

StructureSimulation: Clase encargada de representar a la capa física del sistema.

SensitiveTable: Clase encargada de representar la tabla de sensibilidad de la capa física.

Diagrama de clases paquete “Genetic“

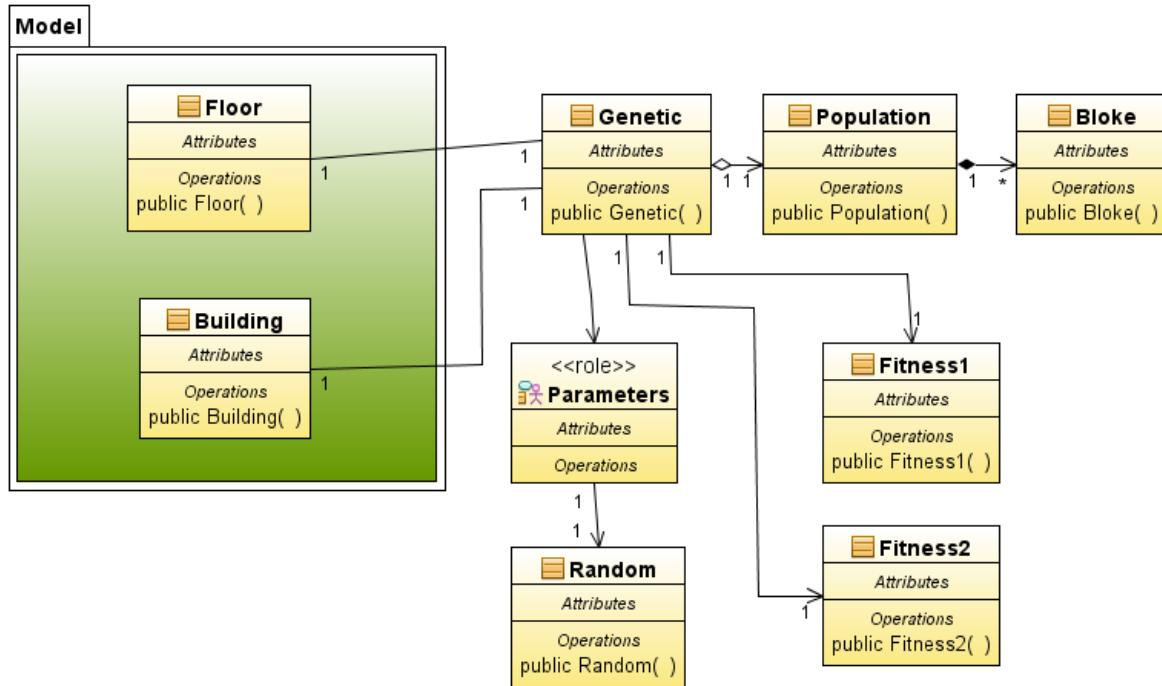


Imagen 3.12. Diagrama de clases “Genetic“

Descripción de clases del paquete “Genetic“

Genetic: Clase principal del paquete “Genetic“ encargada de ejecutar el algoritmo genético.

Population: Esta clase representa a la población de individuos usados para la ejecución del algoritmo genético.

Bloke: Esta clase representa a un individuo de la población.

Parameters: Clase que utiliza el patrón de diseño “singleton“ con todos los parametros necesarios para la ejecución del algoritmo genético.

Fitness1: Esta clase es la encargada de ordenar la población dependiendo del Fitnees 1.

Fitness2: Esta clase es la encargada de ordenar la población dependiendo del Fitnees 2.

Random: Esta clase es la encargada de ejecutar los cálculos probabilísticos.

3.4. IMPLEMENTACION

3.4.1. DISEÑO GRAFICO DE REDES EN EDIFICIOS

Como el nombre de nuestra aplicación bien indica, *WiFiSimExtension*, estamos hablando de la extensión de *WiFiSim*, por lo tanto, nuestra interfaz estará integrada en la interfaz de *WiFiSim*, como podemos apreciar en la siguiente imagen.

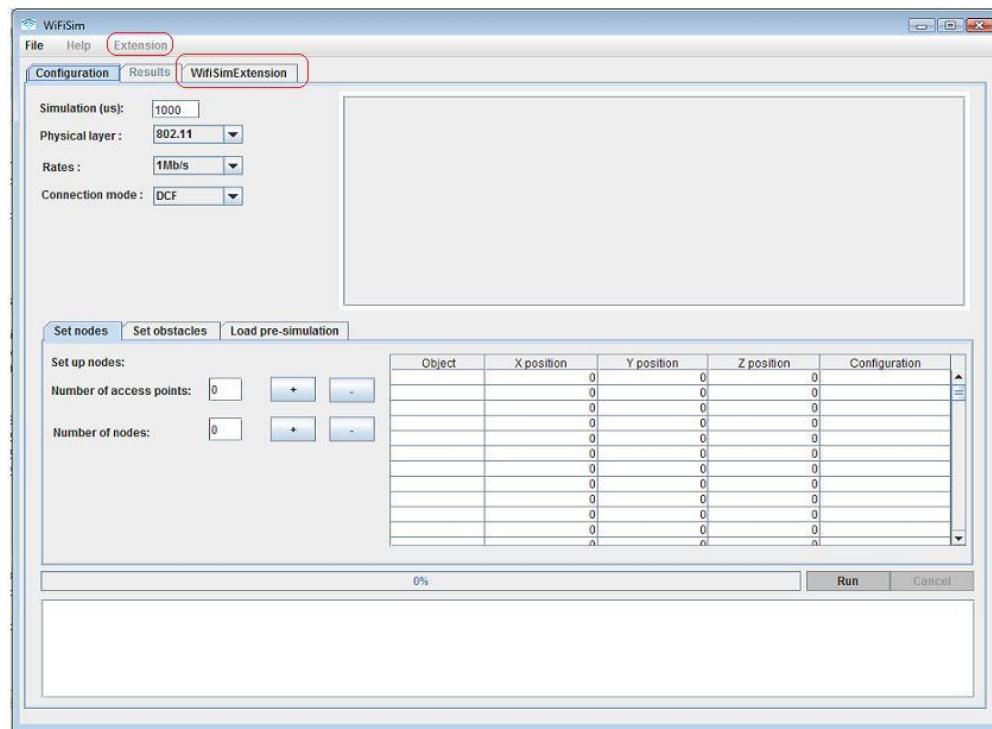


Imagen 3.13. Interfaz de WiFiSim

Los dos círculos rojos que se aprecian en la imagen, son la pestaña y el Menú de WiFiSimExtension, que están correctamente integradas, consiguiendo así una sola interfaz.

La pestaña “Extension“ presenta la siguiente apariencia:

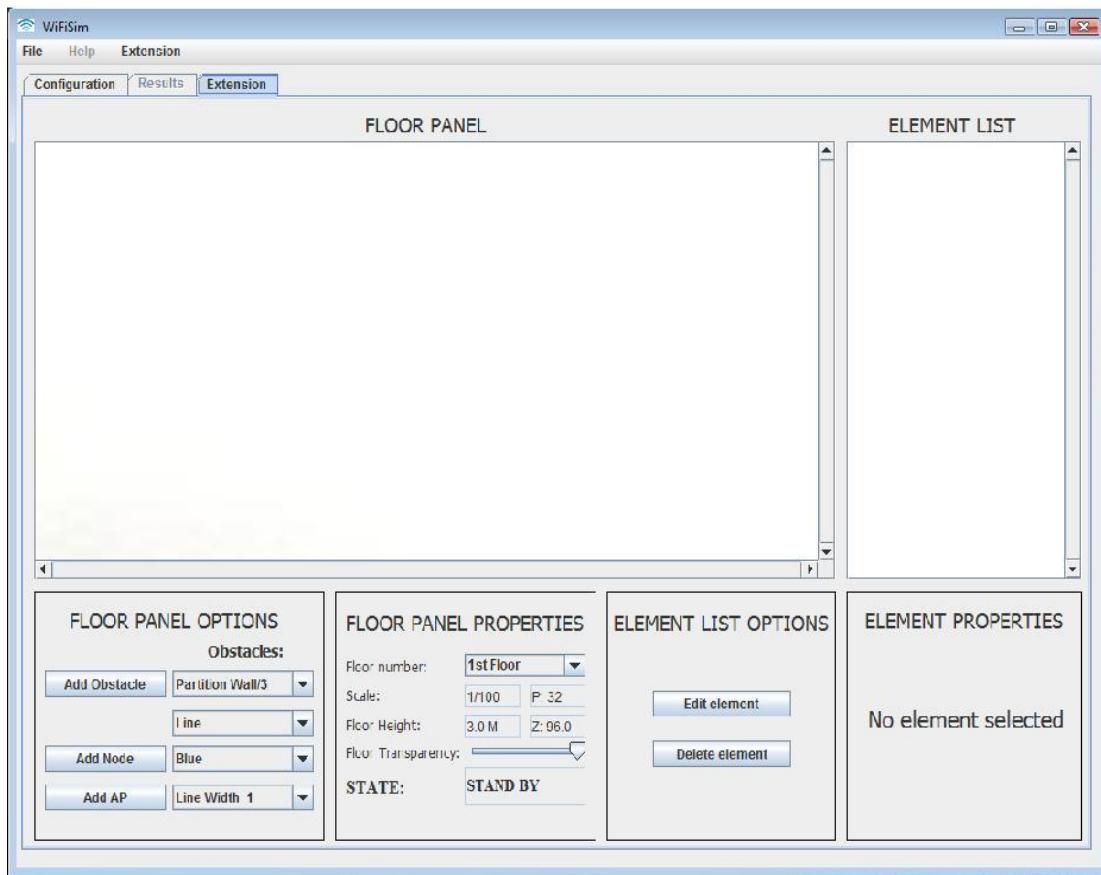


Imagen 3.14. Interfaz de WiFiSimExtension

Para la consecución de los objetivos de este proyecto necesitamos una interfaz capaz de llevar a cabo toda la funcionalidad descrita en los Casos de Uso (Use Case (UC)) del sistema, ahora explicaremos tanto la interfaz como su implementación basandonos en los subsistemas de los casos de uso:

A. Subsistema FILE

UC-New Project

Todos los datos de las clases *Building* y de cada uno de sus elementos *Floor* con sus respectivas listas *Element*, la escala, la altura del edificio, etc. vuelven a sus valores por defecto, como también lo hace la configuración del algoritmo genético.

UC-Save Project

Todos los datos relevantes de un proyecto *WiFiSimExtension* son guardados en un fichero “.txt”. Estos datos pertenecen a las clases *Building* y de los elementos que lo componen.

UC-Load Project

Todos los datos guardados previamente en un fichero *WiFiSimExtension.txt* son cargados por el sistema, restaurando todos los objetos de las clases relevantes con los datos del proyecto.

UC-Save Solution File

Si existe una solución se almacena los datos de esta en un fichero *WiFiSimExtensionSolution.txt*.

UC-Load Solution File

Los datos de una solución guardada de un proyecto es cargada desde un fichero *WiFiSimExtensionSolution.txt*.

Para implementar los casos de usos del subsistema *File* se ha necesitado la ayuda de las funciones de las siguientes librerías JAVA.

- `java.io.File`
- `java.io.FileReader`
- `java.io.FileWriter`
- `java.io.PrintWriter`
- `java.io.IOException`

Que facilitan todas las funciones necesarias para la lectura y la escritura de ficheros.

B. Subsistema BuildingEdition

UC-New Floor

Se pueden crear como máximo tres plantas en un edificio. Cuando añadimos una planta, creamos un nuevo objeto *Floor* que se añade a la lista *Floor* del edificio y a la planta sobre la que hemos añadido este nuevo *Floor* se le añade un elemento tipo *Roof* para que haga la función del techo, con su respectiva atenuación. Se puede copiar la estructura de la planta base si el usuario lo solicita mediante las extensiones de este caso de uso.

El objeto de la clase *Graphic* dibuja el último *Floor* añadido.

UC-Remove Floor

Se elimina el último objeto tipo *Floor* de la lista del edificio y se elimina el elemento tipo *Roof* del *Floor* sobre el que estaba. Si solo tenemos un *Floor*, este caso de uso no se puede llevar a cabo. “Graphic” dibuja la primera planta del edificio.

UC-Add Floor Image

Se carga una imagen tipo “.jpeg” para ser usada de calco de la estructura de un plano arquitectónico, y poder crear la estructura del edificio. El fichero queda almacenado en el atributo *photo* de la clase *Floor*.

UC-Set Floor Height

La altura del edificio es de 3 metros por planta. Con esta función lo que hacemos es cambiar esa altura, y resetear las coordenadas Z, de todos los elementos (nodes, AP, y obstacles) del edificio.

UC-Scale Evaluation

Se cambia la escala del edificio, que por defecto es 1/100, es decir, cada centímetro del dibujo, son 100 centímetros en la realidad. Podemos cambiarla insertando la escala directamente o indicando cuánto mide una obstáculo con forma lineal, a la que se le aplicaría una simple regla de tres para obtener la escala. Al cambiar la escala, todo el proyecto se restaura, para así cambiar los parámetros correspondientes.

Acerca de la clase Graphic

En un edificio de varias plantas solo un *Floor* puede estar activo, y así la clase *Graphic* dibuja los elementos que contiene sobre el *Floor Panel* de la interfaz. *Graphic* utiliza las siguientes librerías JAVA para realizar sus funciones:

- java.awt.Graphics
- java.awt.Graphics

La clase *Graphic* tiene una clase *paint(Graphic g)* que se esta ejecutando constantemente, y cada cambio que se realiza mediante la interfaz se actualiza en la imagen llamando a la función *repaint()* de la clase *Graphic*.

La función *paint* es como se muestra a continuación:

```
@Override  
public void paintComponent(Graphics g) {  
    Graphics2D g2 = (Graphics2D)g;  
    if( this.floor != null){  
        this.floor.paint_floor(g2oculta);  
    }  
    g2.drawImage(Imagen_en_memoria, 0, 0, this);  
    this.repaint();  
}
```

Como vemos en el código de la clase *Graphic*, si existe algún *Floor* llama a su función *paint()*.

Cada vez que hagamos una modificación en el *Floor Panel* y queremos que se actualice *Graphic* usamos la función *repaint()*.

C. Subsistema FloorOptionPane

UC-Add Node

Los nodos son añadidos en el *Floor Panel* indicando su posición con el ratón. Con el evento del ratón que recoje el pixel donde presionamos el ratón, obtenemos ese punto (x,y) del nodo. La coordenada ‘z’ del nodo dependerá de la planta y de la escala, ya que según nuestro diseño vamos a colocar los nodos a un metro desde el suelo. Esto quiere decir que si estamos en la primera planta, y la escala determina que un metro son 32 pixeles, la coordenada ‘z’ de ese nodo será 32.

Aquí podemos ver el código que realiza este caso de uso:

```
g.addMouseListener(new MouseAdapter() {  
  
    public void mousePressed(MouseEvent e) {  
        Point v=new Point(e.getX(),e.getY());  
    }  
    public void mouseRelease(MouseEvent e) {  
        NodeE n= new NodeE();  
        n.setNfloor(f.getNfloor());  
        n.setInicioArrastre2(inicioArrastre);  
        n.setZ(Float.valueOf(String.valueOf((f.getZ())+B.setScale()))));  
        f.addElement(n,numn);  
    }  
}
```

Al presionar el ratón, el evento *mousePressed* recoge el punto (x,y), y al soltar el ratón, el evento *mouseRelease* recoge el momento cuando se suelta el ratón, crea el nodo, le asigna la coordenada ‘z’ y lo añade al *Floor* activo.

UC-Add AP

Los AP son añadidos en el *Floor Panel* indicando su posición con el ratón al igual que los nodos. El comportamiento de la interfaz es el mismo, y el código solo cambia en la creación de un objeto *AccessPointE* en lugar de *NodeE*.

UC-Add Obstacle

Dependiendo del tipo de obstáculo, la forma, el color y el ancho que indiquemos en *Floor Panel Options*, se añadirá un cierto obstáculo. El código de la interfaz que realiza esta función es similar al código para añadir un nodo o un punto de acceso, salvo que necesitamos dos puntos, el de inicio del obstáculo, que se recoge con el evento de presión del botón, y el de final, que se recoge con el evento de soltar el botón.

Hay tres formas posibles para dibujar los obstáculos (línea, rectángulo, elipse). En caso de ser una línea, es sencillo, cada punto indica el inicio y final del obstáculo. En el caso del rectángulo, el inicio será la esquina superior izquierda, y el final la inferior derecha. Y en el caso de la elipse, los puntos corresponden con los del rectángulo, esquina superior izquierda y esquina inferior derecha, capturados con los eventos del ratón, y dentro de ese rectángulo se dibuja la elipse.

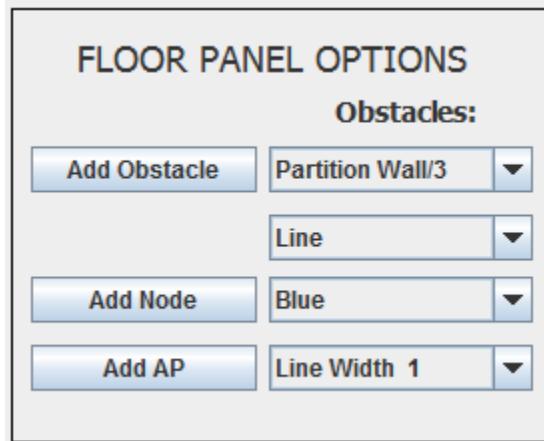


Imagen 3.15. Floor Panel Options

Al soltar el ratón se creará un objeto tipo *Obstacle* con las propiedades indicadas en el *Floor Panel Option* y con la coordenada ‘z’ de la planta donde se encuentre.

UC- Add new Obstacle

Podemos añadir nuevos materiales para usarlos como obstáculos. Los datos que el sistema necesita son el nombre y la atenuación. Tras comprobar que no se repite en la lista de materiales se creará la opción de poder añadir obstáculos de dicho material.

UC-Edit Element

Los elementos tipo *Node* y *AccessPoint* se pueden editar tanto en la posición como en la potencia que transmiten/reciben. Mientras que los obstáculos solo se pueden modificar respecto a su posición actual. Simplemente “seteamos” los nuevos valores o coordenadas capturadas bien mediante la interfaz en *Element Properties* o mediante los eventos del ratón, ya que podemos editar la posición de un elemento directamente en el *Floor Panel* con el ratón. El único elemento que no podemos editar es el techo, ya que se crea por defecto y no se puede modificar.

UC-Delete Element

Esta función elimina el elemento seleccionado de la lista de elementos del *Floor activo*. El único elemento que no podemos eliminar es el techo, al igual que ocurre con la edición. El techo se añade por defecto al crear una planta sobre otra, y se elimina al eliminar dicha planta.

UC-Change Floor

Esta función cambia el *Floor activo*, indicandole al objeto *Graphic* el *Floor* que tiene que dibujar.

UC- Floor transparency

Esta función permite variar el tono de transparencia de la imagen del *Floor activo*. Su funcionalidad es la de facilitar la visión a la hora de pintar la estructura de la planta. El código que realiza esta función es facilitado por las librerías de JAVA para graficos 2D. La función en concreto es la siguiente:

```
AlphaComposite tra=AlphaComposite.getInstance(AlphaComposite.SRC_OVER, 1.0f);  
g2.setComposite(tra);
```

Siendo *g2* el objeto *Graphic* y ‘*tra*’ el objeto con el nivel de transparencia indicado, 1.0f en este caso.

D. Subsistema Functionality

UC- Export to WiFiSim

WiFiSimExtension proporciona la posibilidad a WiFiSim de insertar los elementos necesarios para realizar la simulación dinamicamente mediante nuestra interfaz y luego exportar los datos.

En el *Floor Panel* de nuestra interfaz podemos añadir diferentes tipos de elementos y durante la exportación estos se importan a WiFiSim de la misma manera que los nodos o AP’s dibujados.

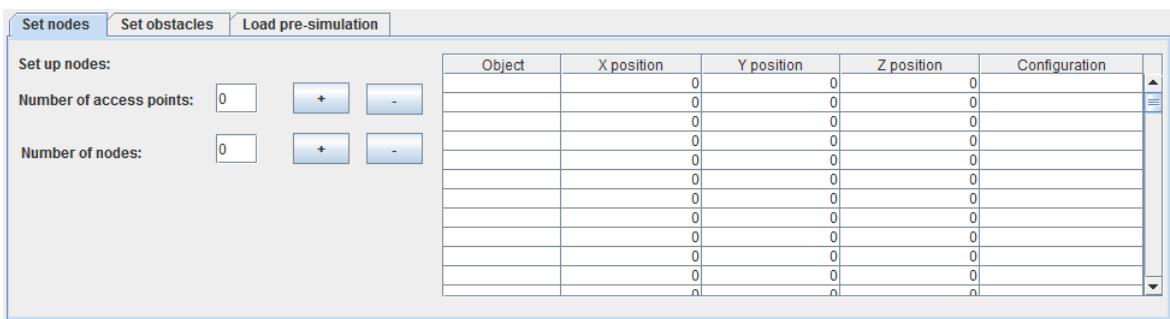


Imagen 3.16. Set nodes Panel de WiFiSim

También tendremos que realizar la detección de obstáculos e insertarlos en la pestaña *Set obstacles* de WiFiSim.

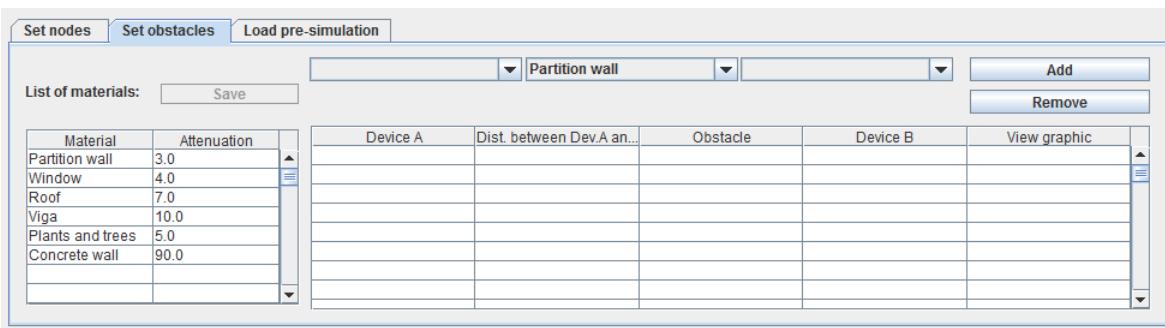


Imagen 3.17. Set obstacles Panel de WiFiSim

Lo primero que hacemos durante la exportación es copiar todos los elementos del edificio en una lista auxiliar en la interfaz principal de WiFiSim. A partir de aquí separaremos los elementos en diferentes listas por tipo (Node, AP, obstacle).

EXPORTACIÓN DE NODES Y APS

Esta es la parte más sencilla de la exportación, simplemente usamos los datos de utilidad de los objetos *Node* y *AP* y los insertamos como tramas *Set nodes* en la interfaz de WiFiSim.

EXPORTACIÓN DE OBSTACULOS

Esta parte es mucho más compleja que la exportación de nodos y puntos de accesos, y se compone de una serie de pasos:

1. Añadir nuevos materiales.

Se añade la lista de materiales de WiFiSimExtension, de manera que podamos crear obstáculos en WiFiSim con los materiales creados en nuestra interfaz.

2. Tratamiento de los diferentes tipos de obstáculos.

En esta parte del proceso, separaremos los obstáculos dependiendo de la forma.

Líneas: Estos obstáculos no necesitan de ningún trato especial, simplemente se dejan tal y como están.

Rectangulos: Un rectangulo esta compuesto por cuatro lineas. Sabiendo que disponemos de los puntos de la esquina superior izquierda, y el de la esquina inferior derecha del rectangulo, tenemos acceso a todos los puntos necesarios para identificar las cuatro lineas y sustituirlas asi por el rectangulo original.

Elipse: La elipse tiene un comportamiento especial en nuestro sistema. Si una elipse se interpone entre dos elementos, deberia cortarse doblemente en dos puntos, como muestra el siguiente ejemplo:

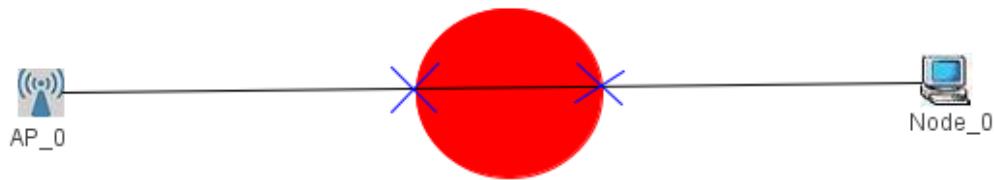
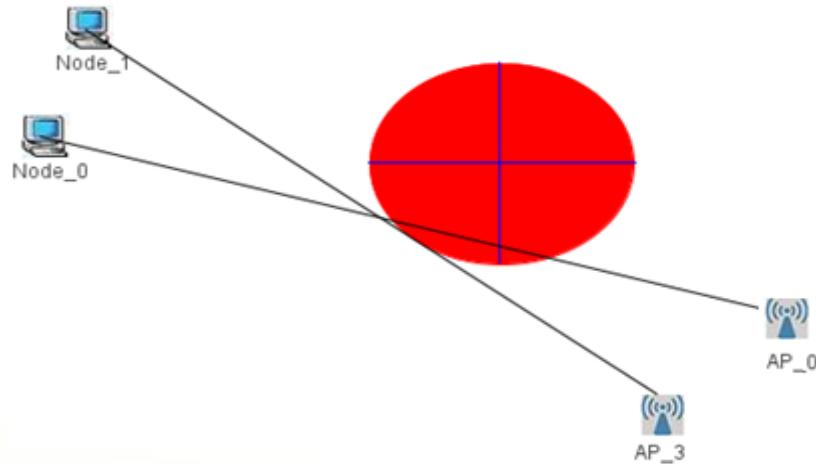


Imagen 3.18. Ejemplo comportamiento elipse.

Sin embargo la elipse sera tomada solo como un elemento compacto que se puede interponer una sola vez entre dos elementos. Por lo que sera tratado de la siguiente forma:



3.19. Ejemplo comportamiento elipse II.

Cada elipse sera sustituida en la lista de obstaculos por dos líneas perpendiculares entre sí pasando por el centro de la ellipse, de manera que si el obstáculo se interpone entre dos elementos, siempre será detectado al cortarse con una de las dos líneas.

En el ejemplo podemos ver como el obstáculo no se interpone entre “Node_1” y “AP_3”, pero sí lo hace entre “Node_0” y “AP_0”. Siempre que se corte con una de las dos líneas, no se tendrá en cuenta si se corta con la otra, para no insertar así dos tramas del mismo obstáculo en *Set obstacles* de WiFiSim.

3. Tratamiento del solapamiento de obstáculos.

A la hora de dibujar la estructura del edificio es posible que algunas líneas se solapen. El sistema tratará de manera especial este solapamiento, para evitar que se deteste el mismo obstáculo dos veces.

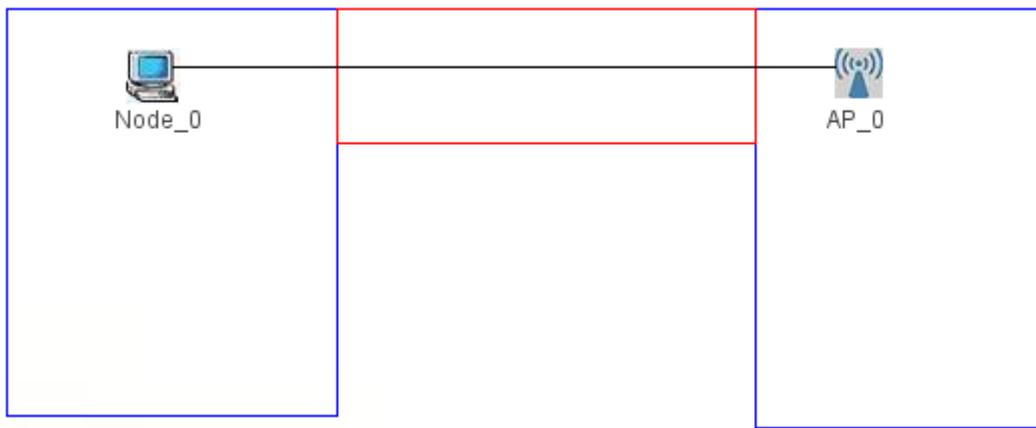


Imagen 3.20. Ejemplo solapamiento.

Como vemos en la imagen, tenemos una estructura la cual el usuario por comodidad ha dibujado usando rectángulos en lugar de líneas, por lo que existe solapamiento entre el rectángulo rojo en la imagen, y los otros dos. Por lo tanto al exportar a WiFiSim, entre el “Node_0” y “AP_0” el sistema detectará cuatro obstáculos, dos del rectángulo rojo y uno por cada rectángulo azul.

Con el tratamiento del solapamiento queremos evitar esta situación. Y cada vez que dos obstáculos se solapen, solo se tendrá en cuenta uno de ellos. En este caso el rectángulo rojo solo se solapa con los azules en las partes de arriba de estos. Lo que el sistema hace es dejar las líneas del rectángulo rojo, y acortar las líneas interiores de los rectángulos azules.

Si dos obstáculos son exactamente iguales, uno encima de otro, borramos uno de los dos.

4. Recorrido por los nodos y Aps y detección de obstáculos.

El sistema tiene que ser capaz de detectar obstáculos entre los puntos de acceso y los nodos, por lo que recorrerá la lista importada desde WiFiSimExtension para realizar la detección de obstáculos.

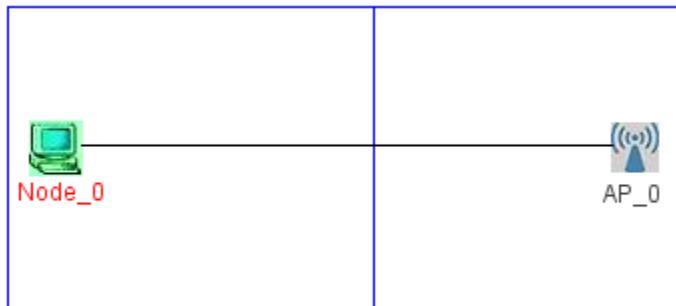


Imagen 3.21. Ejemplo detección de obstáculos entre AP y Node.

Al recorrer la lista de Nodos y la de Aps, cogemos dos elementos activos en las listas y recorremos la lista de obstáculos, que ahora son todos líneas, usando la función `seCortan(línea1, linea2)`, que devuelve la distancia entre el primer elemento al punto de corte si se cortan, o “null” si no se cortan. En este caso, “linea1” es una línea creada entre la posición(x,y,z) del nodo, y la posición(x,y,z) del punto de acceso. Y siendo “linea2” el obstáculo.

Al detectar el obstáculo, añadimos una trama a “Set obstacles“ de WiFiSim, con los dos elementos, la distancia del obstáculo al primer elemento activo y el nombre del material del obstáculo.

5. Recorrido por los nodos y detección de obstáculos.

WiFiSim también realiza la simulación en base al entorno entre nodos, para detectar problemas como el “caso del nodo oculto“. Por lo que realizamos el mismo proceso que hicimos con los nodos y puntos de acceso, pero solo con los nodos.

6. Detección del techo.

Al hacer sendos recorridos de los pasos ‘4‘ y ‘5‘ también se detecta el obstáculo *Roof*, si estan en diferentes plantas, dandose a veces el caso de que existan dos techos entre los elementos, si es que uno esta en la primera planta y el otro en la tercera. Al detectar este obstáculo, se calcula la distancia con el elemento de la planta más baja y se añade a *Set obstacle* de WiFiSim.

Una vez realizada la exportación de elementos WiFiSim podra realizar la simulación normalmente mediante su interfaz.

UC- Import from WiFiSim

Desde WiFiSim se pueden exportar elementos a nuestra interfaz. Se puede pasar tanto nodos, puntos de acceso y obstáculos, como la capa física.

La capa física es necesaria de ser exportada para la realización del algoritmo genético que calcula la mejor solución, ya que dependiendo de la capa física y de su tasa de sensibilidad los puntos de acceso de la solución tendrán mayor o menor alcance.

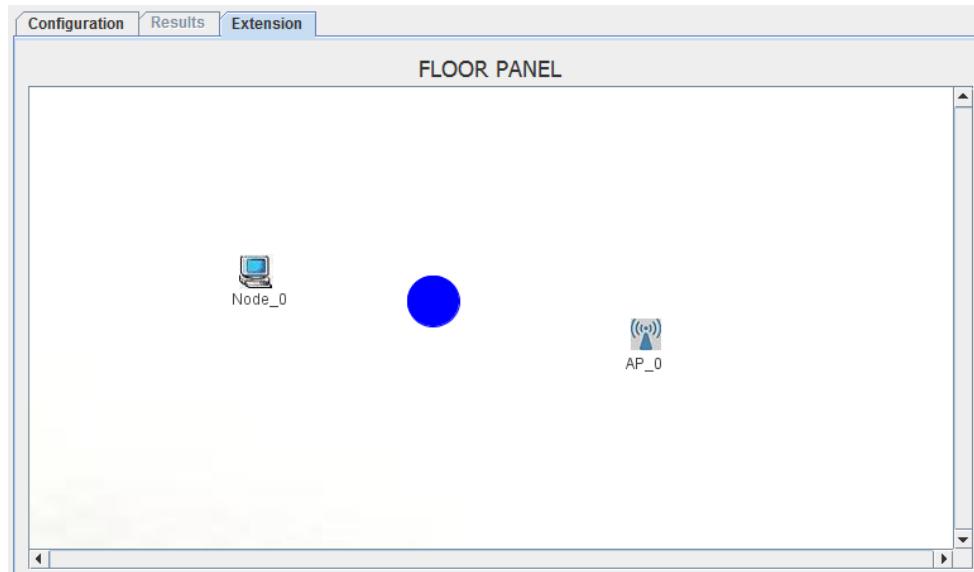


Imagen 3.22. Ejemplo importación desde WiFiSim.

En la imagen podemos ver la importación del “Node_0“, “AP_0“ y de un obstáculo que ha sido insertado desde “Set obstáculos“ de WiFiSim.

3.4.2. DISEÑO DEL SISTEMA *BEST SOLUTION*

Una vez descrita la funcionalidad de nuestra aplicación en la interfaz del usuario (capa de presentación), en el presente punto procederemos a describir la capa lógica de negocio referente a la búsqueda de la mejor solución.

Una vez dibujado el plano deseado, el objetivo principal de nuestra aplicación será calcular las mejores posiciones para instalar los Puntos de Acceso (APs), de manera que exista cobertura para la mayor superficie posible dentro del edificio en cuestión, y dentro de la mayor cobertura, aquella que llegue con mayor intensidad, es decir, si por ejemplo tenemos un plano cuadrado muy pequeño, y se da el caso de que con los APs indicados se tiene el 100% de cobertura, se calculará la mejor posición para que además de que la cobertura sea máxima, no haya zonas donde llegue poca intensidad, sino que incluido el primer criterio de optimicidad (que la cobertura sea máxima), que en cada una de las zonas llegue bastante intensidad y haya el menor número de zonas posibles con señal mínima.

Esto será calculado mediante un algoritmo genético estacionario el cuál será explicado en el siguiente punto.

¿Como dividimos el plano en diferentes zonas?, ¿Cómo colocamos los APs?, ¿Cuántos APs harían falta?, ¿Qué tendremos en cuenta para calcular las posiciones?, ¿De qué manera medimos la cobertura e intensidad?, ¿Dónde colocaremos los APs?, ¿Y los nodos (PCs)?,.....

Todas estas preguntas serán descritas progresivamente a lo largo de este punto.

Una vez cargado o dibujado el edificio, si nuestro objetivo (ya que nuestra aplicación tiene una amplia funcionalidad) es calcular donde colocar los APs para obtener la mejor solución, tendremos que ir a **WiFiSimExtension->BestSolution->OptionPanel**.

WiFiSimExtension

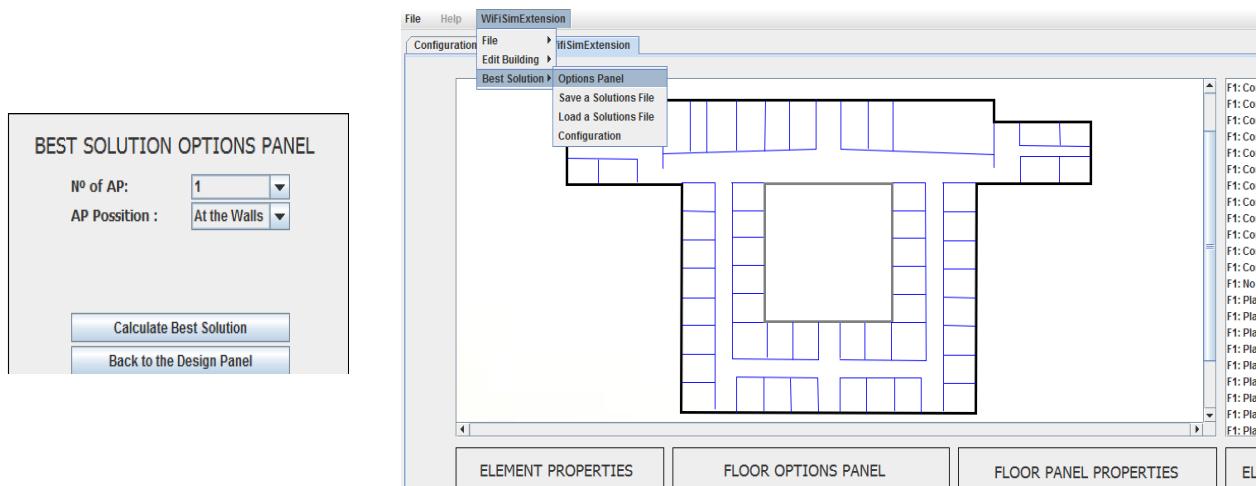


Imagen 3.23. Option Panel BestSolution

El *Floor Options Panel* que será el panel para dibujar (Capa Presentación), será sustituido por el *Best Solution Options Panel*.

En dicho panel, debemos indicarle el número de APs que vamos a tener en el edificio y con este conjunto, tener la mayor cobertura e intensidad posible.

El tipo de AP, es decir, la tecnología de cada AP, será indicada en la pestaña WiFiSim, y aquí se tendrá en cuenta para saber con qué potencia el AP emite la señal.

Por otro lado, podemos indicar la posición donde queremos colocar los APs, bien solo en las paredes o en cualquier posición, en cuyo caso se colocarían en el techo.

Una vez le damos al botón de calcular la mejor solución con los parámetros indicados el sistema trabajará internamente (capa Lógica de Negocio), de la siguiente manera:

OPERACIONES:

1. Lo primero que hará el sistema, por cada planta, será determinar los límites del edificio dibujado. Esto es, nosotros tenemos un edificio pero el sistema no sabe las dimensiones de este. Podríamos decir que tendríamos que realizar una especie de “escaneo” para determinar las dimensiones del edificio.

Para ello se verá dentro de la paleta *Floor Panel* para dibujar cuál es el punto más a la izquierda dibujado, así como el más a la derecha, el más arriba y el más abajo.

Una vez se identifican estos puntos, el sistema unirá dichos puntos de manera que dibujará un rectángulo:

Por ejemplo: Dibujamos esta figura, lo primero que hará nuestro sistema para hallar la mejor solución será identificar los puntos extremos de manera que quedaría así:

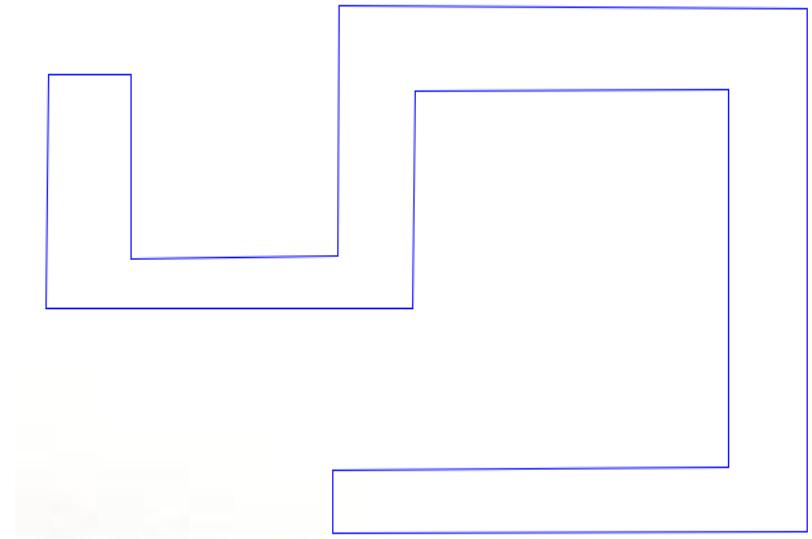


Imagen 3.24. CrearGrid I

Por tanto el rectángulo dibujado en color negro sería lo que se tendría en cuenta en principio, es decir, este sería el *grid*.

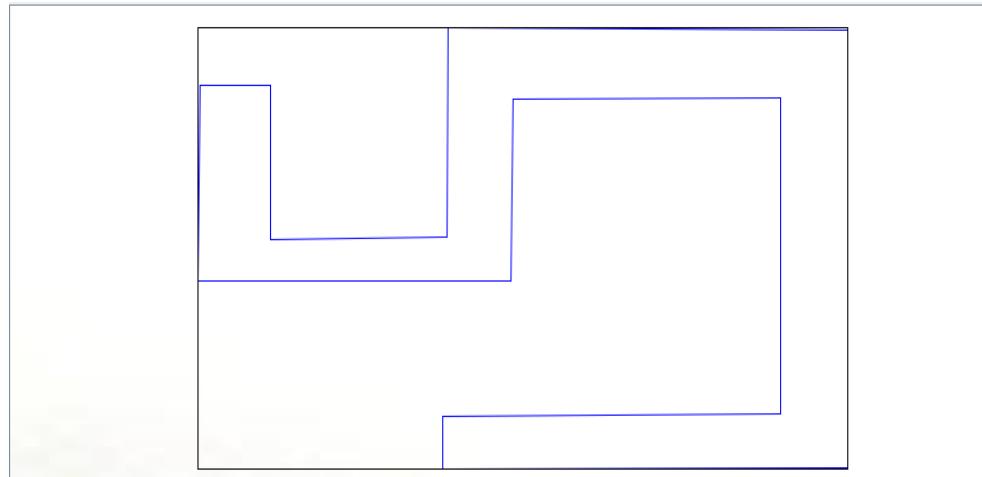


Imagen 3.25. CrearGrid II

Debemos indicar, que en cada planta se realizará un *grid* igual o distinto en función de la figura dibujada en ella, de manera que un edificio tendrá tantos *grid* como plantas posea.

El sistema realiza un *grid* en cada planta, y cada planta tendrá la altura indicada en la aplicación, siendo por defecto de tres metros y pudiéndose modificar si se desea por el usuario.

2. Una vez obtenido el/los *grid*/s, el siguiente punto será determinar cuales son las superficies que se encuentran dentro del edificio y cuales son las superficies que se encuentran fuera.

Una vez hemos dibujado el *grid* el sistema obtiene su propia ventana de trabajo, **ignorando aquellas zonas que estén fuera de este**. Pero es ahora cuando debemos de dividir el *grid* en zonas, y dentro de estas, determinar cuales están fuera del plano y cuales están dentro.

En este punto jugará un papel determinante la escala que estemos utilizando, ya que en función de esta, las zonas serán de este factor cuadrado. Por ejemplo la escala por defecto, indica que un cm en el dibujo son cien en la realidad 1/100. Por tanto las zonas en las que se dividirá el *grid* será en mallas de 1*1 (celdas de aquí en adelante).

De esta forma la división empieza en el punto superior izquierda del *grid*, y terminará una vez se haya recorrido todo y llegue al punto inferior derecha.

Como se indica anteriormente, cada celda tendrá de ancho y de alto la escala, siendo estas totalmente cuadradas. Es entonces cuando encontramos el primer problema; ¿y si no coincide exactamente?

Es decir, imaginemos un edificio que de ancho mide 10 metros y de largo 10 y medio.

De ancho habría 10 celdas, pero de largo habría 10 y media. ¿Cómo solucionamos esto? En caso de que se dé esta circunstancia el *grid* se completará con celdas de tamaño variables (*MiniCeldas* de ahora en adelante) de manera que estas pueden tener distinto largo que ancho no siendo totalmente cuadradas sino rectangulares.

Por ejemplo:

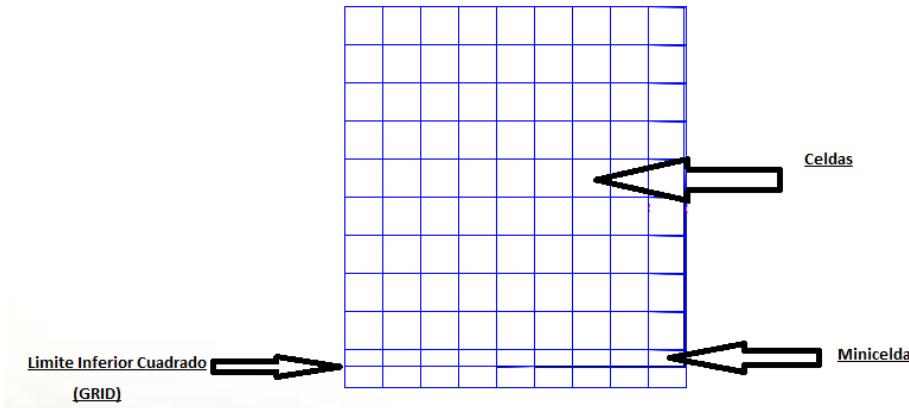


Imagen 3.26. CrearGrid III

Aquí podemos ver gráficamente cuales serían las celdas y miniceldas que quedarían dentro del grid y cuales son las que quedarán fuera.

Por tanto, el sistema deberá de diferenciar ambos conjuntos para saber cuales debe de tener en cuenta (las que quedan dentro de la figura dibujada Y NO DEL GRID) y cual es aquel conjunto que queda fuera del plano dibujado.

Nótese lo siguiente: puede ser que una celda o minicelda esté dentro del *grid*, pero sin embargo esté fuera del plano dibujado, en el citado caso el sistema la considerará fuera.

Cada celda accesible será una zona del plano de un tamaño determinado por la escala (indicada por el usuario o por defecto), y teniendo una altura igual a la planta en cuestión.

3. El sistema reconoce dos conjuntos:

- ➔ Las celdas y miniceldas que quedan fuera (No accesibles), que no las tiene en cuenta
- ➔ Las celdas y miniceldas que quedan dentro del plano (accesibles), en cuyo conjunto se considerará a cada celda una zona del plano, y en estas el sistema considerará que podría tener un nodo a una altura de un metro del suelo.

Por tanto, todas las celdas y miniceldas que estén dentro del plano podrán disponer de un nodo a esa altura, se considerará pues a este conjunto como accesible.

El sistema reconocerá si una celda es accesible o no de la siguiente manera:

- a. Debe de ir desde el centro de la celda hacia arriba hasta el punto más alto del *grid*, y ver que al menos se corta con algún obstáculo.

- b. Debe de ir desde el centro de la celda hacia abajo hasta el punto más bajo del *grid*, y ver que al menos se corta con algún obstáculo.
- c. Debe de ir desde el centro de la celda hacia la izquierda hasta el punto más a la izquierda del *grid*, y ver que al menos se corta con algún obstáculo.
- d. Debe de ir desde el centro de la celda hacia la derecha hasta el punto más a la derecha del *grid*, y ver que al menos se corta con algún obstáculo.
- e. →Si no se dan estos 4 pasos:
La celda o minicelda en cuestión no será accesible.
→Si se dan estos 4 pasos:

Tiene en cuenta la posición de la celda que referenciamos y empieza a contar el número de obstáculos de tipo contorno desde la izquierda del todo del *grid* en esa coordenada Y (la de la celda), hasta que llega al centro de la celda (siempre en la misma coordenada Y la que cambia es la X).

- 5.1 Si el número de obstáculos de tipo contorno contados en este proceso es impar o cero la celda será accesible.
- 5.2 Si el número de obstáculos de tipo contorno contados en este proceso es par la celda no será accesible.

Nótese que: Es **OBLIGATORIO** dibujar el contorno del plano con este tipo de obstáculo.

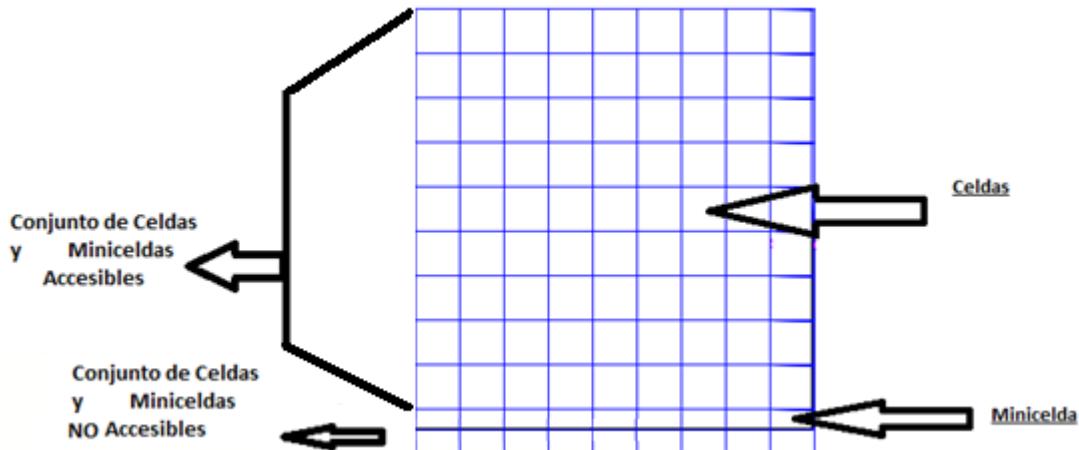


Imagen 3.27. Celdas y miniceldas

- 4. Llegados a este punto el sistema se encuentra con un nuevo problema que debe solventar. Dentro del conjunto accesible, éste debe distinguir entre aquellas celdas y miniceldas en las que pueden colocarse un AP, o si por el contrario a pesar de ser accesible no puede colocarse ninguno.

Esto el sistema lo resuelve de la siguiente manera:

Teniendo ya una lista con aquellas zonas (celdas y/o miniceldas) accesibles, determinará otro conjunto nuevo: *AccesiblesAP*.

En este conjunto irá las celdas y/o miniceldas en las que siendo accesibles, pueda poseer de un AP.

El sistema determinará este subconjunto teniendo en cuenta dos factores:

1. Si el usuario determina que los APs pueden ponerse en el techo, el sistema tendrá en cuenta que toda aquella celda y minicelda que sea accesible, lo sea también por AP, en este caso dichos conjuntos (*Accesible* y *AccesibleAP*) serían el mismo.
2. Si por el contrario el usuario determina que los APs solo pueden ponerse en las paredes, el sistema recorrerá el conjunto de zonas accesibles, y en el caso de que se crucen o estén al lado de una pared (elemento del dibujo del material que desee el usuario), esta zona sea accesible por AP, pudiendo tener así un AP colocado aproximadamente a la altura de la planta en cuestión (unos centímetros menos).

Para mayor comprensión del usuario, pueden verse perfectamente las zonas diferenciadas (*Accesible*, *AccesibleAP* y *NO Accesible*), mediante el color de la celda (verde=Accesible, rojo=AccesibleAP, blanco=NO Accesible).

De este modo no habrá ninguna *celda/minicelda AccesibleAP* que no sea *Accesible*, lo que conlleva a que no podrá existir ninguna zona que pueda poseer un AP pero no pueda poseer un Nodo, sino que estos se podrán ser colocados a distinta altura.

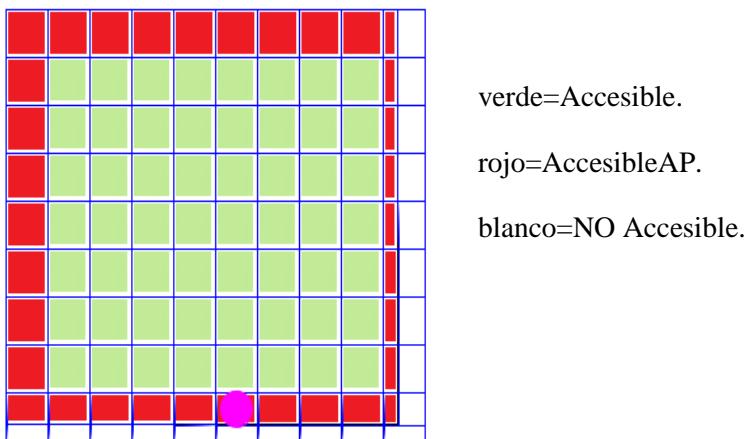


Imagen 3.28. Ejemplo I

Por otra parte, cabe destacar que si la escala utilizada es modificada por el usuario, estas mallas (celdas y miniceldas) cambiarían de tamaño. Esto es, si obtenemos la escala en un determinado plano, y la aplicamos al programa, las celdas en cuestión serán de un metro cuadrado, para valorar así cada zona del plano y no dejarnos ninguna. Esto también podrá darse en función de cada tipo de edificio, puesto que si es un edificio muy grande donde cada cuarto mide 5 metros cuadrados por ejemplo, sería muy raro, que, en el mismo cuarto sin haber obstáculos hubiese zonas que si tuviesen coberturas y otras que no. Por tanto y en estos casos expuestos en concreto, cada celda será del tamaño cuadrado que estimemos sean predominantes en los despachos o cuartos del edificio y sea este la unidad más pequeña sin obstáculos. (Esto se hace sobretodo para mejorar los tiempos de cálculo y respuesta y el comportamiento del algoritmo genético como veremos en el punto más adelante).

Sin embargo y en un edificio donde los cuartos no sean de tipo uniforme, la unidad más pequeña será dispersa, o bien no tenga obstáculos, el tamaño de las celdas y miniceldas se calculará en función de la escala, siendo estas en su defecto de 1 metro cuadrado (las celdas) y representándose gráfica e internamente según la escala utilizada:

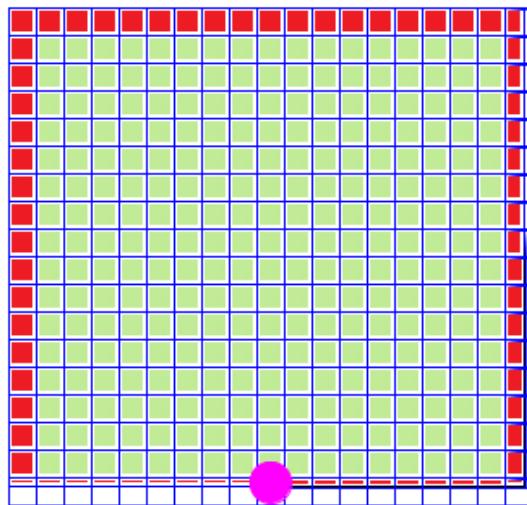


Imagen 3.29. Ejemplo II

Escala 1/200 (1 cm en el dibujo son 2 metros en la realidad)

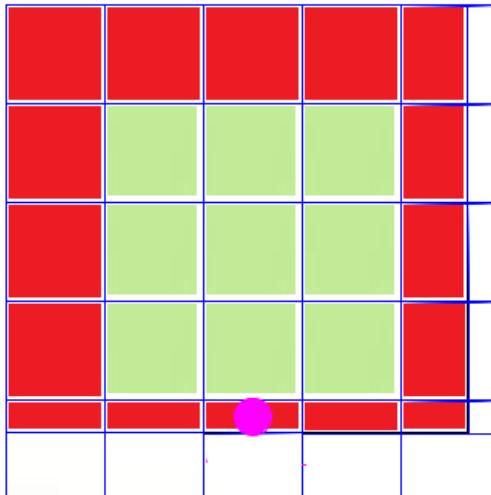


Imagen 3.30. Ejemplo III

Escala 1/50 (1 cm en el dibujo es medio metro en la realidad)

5. Una vez el sistema ha dividido el plano entre las zonas que son No accesibles, accesibles y accesiblesAP como hemos visto en el punto anterior, ahora dichos conjuntos los va a tener que utilizar para hallar la cobertura e intensidad, que serán los elementos con los que el genético trabajará para hallar la mejor solución.

Este paso es estático, solo depende de la figura dibujada pero no cambia a lo largo de la ejecución del genético, solo cambia cuando le cambiamos la escala a la planta o la altura, pero si no cambiamos estos parámetros, cada zona será siempre de ese tamaño indicado y tendrá las alturas descritas, por lo que no cambiará su posición ni su tamaño y conllevando a que de esta manera tampoco variará su cobertura ni su intensidad.

Sirva como ejemplo las losas utilizadas en el suelo de un edificio (celdas):

- Tienen un determinado tamaño elegido a la hora de construir el edificio (que será el lado de la celda al cuadrado, que dependerá de la escala)
- Si sobra un hueco demasiado pequeño donde no quepa una losa, ésta se parte (creando una minicelda) y se coloca.

Se usan unas losas de un determinado tamaño pero una alicatadas, no pueden cambiar ni su tamaño ni su posición.

Por tanto y una vez el sistema calcula los 3 conjuntos mencionados anteriormente, este trabajará con los accesibles y los accesibles AP desechando así los no accesibles.

Este paso aquí descrito se corresponderá con la función putTablas() del genético, que será como calcular las estructuras que vamos a utilizar en el.

El proceso que el sistema seguirá será el siguiente:

- ➔ Para cada una de las zonas (celdas o miniceldas) accesibles por APs:
 - Recorrerá todas las zonas del conjunto de accesibles.
 - Teniendo en cuenta el tipo de tecnología que se emplea (que se indica en WiFiSim) para la zona AP en la que está, y teniendo en cuenta la celda accesible que recorre determinará lo siguiente:
 - Si son distintas: mira si llega cobertura desde la zona accesible AP a la zona accesible, y si llega se añade a la lista de las que llega incrementándose el número de celdas, así como incrementa la intensidad de la zona accesible AP sumándole la intensidad citada.

¿Como halla el sistema este paso anterior, si llega o no cobertura e intensidad?

Cuando pasamos de WiFiSim a WiFiSimExtension, se ha indicado el tipo de tecnología que se va a utilizar, y por tanto la señal mínima por debajo de la cuál no llegará cobertura a esa zona.

Por tanto el sistema determinará esta operación tomando el centro de la zona accesible AP y el centro de la zona accesible, calculando la distancia existente entre ambos puntos (y la pérdida de señal por distancia), así como en dicho camino reconocerá los obstáculos que se cruzan en el (y restará la atenuación de cada uno), de manera que una vez calculado, si está por encima de la cobertura mínima para que exista señal con la tecnología que estemos trabajando existirá cobertura, y añadiremos esta celda en cuestión así como su intensidad (por tanto la intensidad será la suma de todas las celdas a las que llega).

Una vez hallado este paso el genético dispondrá de todas las estructuras para hallar la mejor solución.

3.4.3. ALGORITMO GENÉTICO

Una vez hemos marcado la opción de encontrar la mejor solución, el sistema desencadena la serie de pasos descritos en el apartado anterior, creando así todas las estructuras necesarias para hallar la mejor solución.

Esta mejor solución se determinará por medio de un algoritmo genético, el cuál describiremos a continuación. Es importante indicar que se considerará en primer lugar el número de zonas a las que llega cobertura y en segundo lugar la intensidad media con la que llegará a cada zona en las que existe cobertura.

El algoritmo genético tratado se trata de un genético estacionario y su correspondencia estará en el paquete *Genetics*, y utilizará las estructuras descritas anteriormente en el apartado anterior, las cuales se encuentran en el paquete *StructureGenetics* del código.

El esqueleto del algoritmo genético es básico, por lo que en este punto explicaremos los detalles de diseño tenidos en cuenta en cada punto de éste:

Pseudocódigo del algoritmo:

generarPobInicial()

evaluarPoblacion() <- fitness 1 y 2

hasta mejorSol sea durante 500 it o no le demos a parar hacer

Selección de Padres()

cruce()

mutación()

Reemplazo()

Fin

Codificación del Cromosoma:

Es importante resaltar que antes de empezar, el sistema debe conocer los datos del cromosoma, tales como por ejemplo su tamaño (nº APs que el usuario haya indicado), que es lo que representa (celdas y miniceldas, correspondencia genotipo-fenotipo[Anexo]), y de que manera lo representa, es decir el tipo de codificación que se usará. El sistema pues en generar la población inicial generará tantos cromosomas (Individuos) según una función para ver el tamaño de la población, y estos cromosomas tendrán un tamaño proporcional a los APs que intervengan en la solución. De esta manera, cada posición (gen del cromosoma) tendrá un número que irá desde el cero hasta el tamaño de zonas-1 (celdas+miniceldas-1), siendo este número entero la zona a la que se referencia. Esta generación inicial tendrá lugar en la función ***GenerarPobInicial()***:

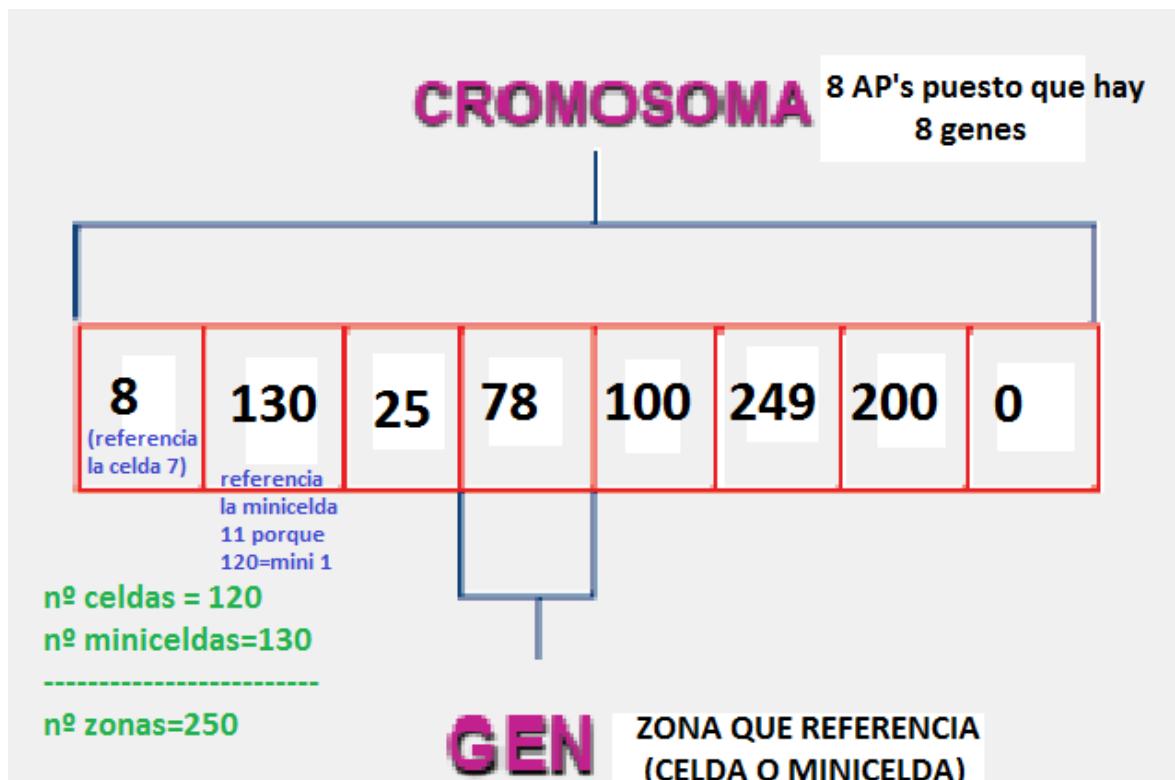


Imagen 3.31. Cromosoma

Generar Población Inicial:

En este punto hemos de considerar antes de nada el número de individuos que tendrá la población. Este paso se hará dinámicamente, ya que no será lo mismo un edificio muy grande que uno muy pequeño. Por tanto la fórmula seguida para hallar el tamaño de individuos que tendrá la población será la siguiente:

Tamaño población: $(nº \text{ casillas accesibles}/\text{tamaño cromosoma}) * k$

Donde:

nº casillas accesibles: número de casillas(celdas y miniceldas) que están dentro del edificio (explicado en el punto anterior).

Tamaño cromosoma: será el número de APs que el usuario indique al sistema para hallar la mejor solución.

K: factor constante=3.

Así pues nos encontramos con un problema en el diseño, y será el siguiente:

En un edificio demasiado grande, el número de celdas accesibles será también muy grande lo que conllevará a que la conversión del genético sea demasiado lenta y el programa resulte pesado debido a su latencia de respuesta.

Por esta razón hemos establecido un límite y a partir de un número demasiado elevado para operar, el mismo disminuirá el tamaño de la población en medida regular, siendo este como máximo de 100 individuos. Este número fue determinado tras varios estudios sobre el tamaño medio de diferentes individuos.

Una vez se ha determinado el tamaño de la población, que será el primer paso de nuestro algoritmo genético, el siguiente paso será el de inicializarla:

Hasta i=tamaño población hacer

Hasta j=tamaño cromosoma hacer

//crear individuos válidos (Sin que se repitan)

Random(1,nºcasillas accesibles)

Fin

Fin

Una vez obtenemos la población inicial, ésta será un número determinado de individuos (el indicado en su tamaño), y cada individuo de ésta será de longitud del tamaño del cromosoma (nº APs), y en cada posición o gen del cromosoma irá el número en cuestión aleatorio dentro de las casillas accesibles.

El siguiente paso será hallar la correspondencia entre lo que tenemos, que solo son números, y a que es lo que referenciamos (celdas y miniceldas).

Esto es en lo que dentro de la rama de algoritmos genéticos se denomina como “*correspondencia entre genotipo y fenotipo*”.

Esta correspondencia el sistema la halla de la siguiente manera:

- ➔ Recordemos que teníamos un array de celdasActivasAP y otro de miniceldasActivas.
- ➔ Si el número en cuestión del gen del individuo es menor o igual que el tamaño del primer array referenciará a la celda en cuestión.
- ➔ Si es mayor referenciará a la minicelda en cuestión quitándole al número la longitud del tamaño del array celdasActivasAP.

evaluarPoblacion()

Una vez obtenemos las celdas a las que referencia dicho genotipo, es decir, una vez hallado el fenotipo el siguiente paso será el de evaluar la población aleatoria inicializada. Este paso constará de lo siguiente:

Fitness1

Fitness2

Quedarse con el mejor

Fitness1():

El fitness1 consiste en la suma de las coberturas a las que llega una determinada zona, sin que las zonas a las que llegan ambas se repitan. Esto se ilustra en el siguiente ejemplo



Imagen 3.32. Fitness 1

Imaginemos que un individuo de dos APs tiene las dos celdas negras.

Estas coinciden en su cobertura en las rojas, es decir por separado cada una llega a 25 celdas, por tanto la suma serían $25+25=50$, pero como coinciden en 5, estas 5 no se suman dos veces por lo que realmente serían 45 el número de celdas con cobertura de esta conjunción.

Otro ejemplo:

→ Si tenemos dos APs y un individuo tiene el 1 y el 8, el fitness uno se considerará de la siguiente manera:

- Si celda 1 llega a las celdas: 1,2,3,4,5,6
- Y celda 8 llega a las celdas: 5,6,7,8,9,10

Entonces el conjunto de las 1 y 8 será la unión: 1,2,3,4,5,6,7,8,9,10.

Fitness2():

Ahora en cada individuo no evaluado, tendremos que poner la suma de la intensidad con la que llega a cada una de sus celdas. Esto análogamente con el fitness 1, se entiende con el ejemplo dispuesto de forma gráfica:

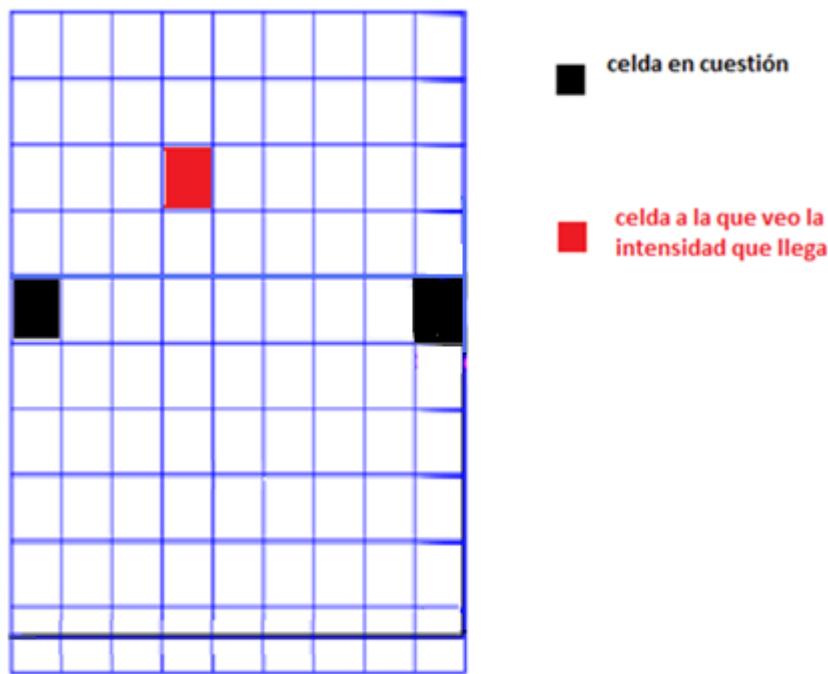


Imagen 3.33. Fitness2

Ahora tengo otra vez el mismo ejemplo de antes con las celdas en negro como cromosoma de algún individuo, el siguiente paso es recorrer las celdas a las cuales llegan (hallado con el fitness1) y determinar la intensidad, sumando esto al parámetro fitness2 del individuo. Pero imaginemos el caso de que a la celda roja llega tanto la celda negra de la izquierda como la de la derecha. En este caso no se suman las dos intensidades sino, aquella intensidad que llegue mejor (en este caso la izquierda porque está más cerca y no existen obstáculos). Esto se hará con todas las celdas en las que haya cobertura según el cromosoma y se irá sumando para determinar el fitness2.

Mejor Individuo():

Una vez hemos determinado el fitness 1 y el fitness2 de todos los individuos no evaluados, el siguiente paso de evaluar población será el de elegir el mejor candidato.

Este paso el sistema lo resolverá de la siguiente manera:

- Aquel individuo con mejor fitness1 será el mejor.
- En caso de empate de más de un individuo con el mismo fitness1 se competirá entre ellos en base a ver cuál es el que mejor fitness2 tiene.

Selección de Padres:

Dentro del bucle del algoritmo, el primer paso que tendremos será el de selección de los padres de la población para posteriormente cruzarlos y mutarlos, y si se da el caso remplazarlos.

El sistema desarrolla este algoritmo de la siguiente manera:

- Escoge aleatoriamente k individuos. (k estará entre 5% y el 15% de la población, y el usuario podrá indicarlo en configurations)
- Dentro de los candidatos a ser padres, seleccionará el mejor de ellos.

Este paso se realizará dos veces, tanto para el padre como para la madre en cada iteración del bucle del algoritmo.

Cruce:

El sistema considerará que **siempre** se crucen sus individuos, o lo que es lo mismo, que la probabilidad de cruce sea 1.

El cruce empleado será el cruce en un punto, de manera que lo hallará así:

- Genera un número aleatorio entre 1 y tamaño cromosoma-1.
- Este será el punto que determine la raíz y la cola de cada individuo.
- Hemos de tener muy en cuenta, que un gen no puede repetirse en un cromosoma.

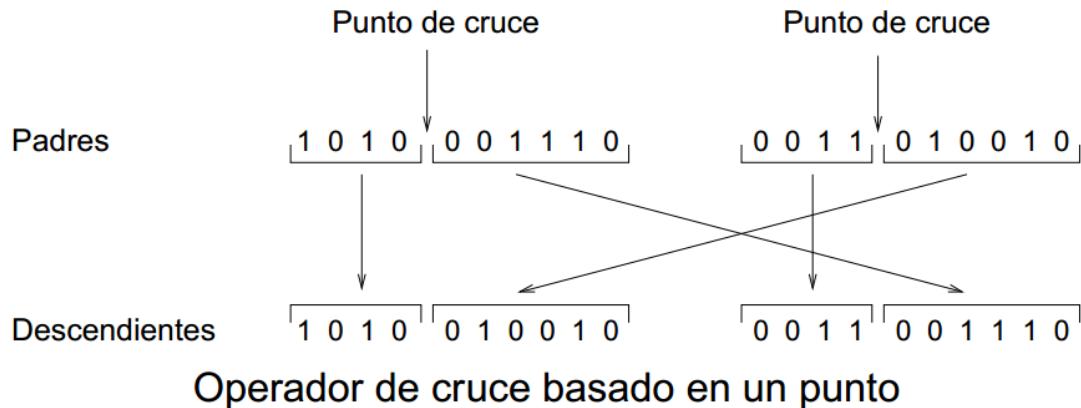


Imagen 3.34. Operador de cruce

Este ejemplo sirve a modo gráfico para ver como sería el proceso pero los genes no serían así sino que cada gen del cromosoma sería el id de la celda AccesibleAP en cuestión, no pudiendo repetirse ésta en un mismo cromosoma. Para ello se ha implementado una función que en dicho caso (en la cabeza de un padre se repite una zona que también es referenciada en la cola del otro), coge el siguiente elemento del cromosoma donde se repite comportándose como una cola circular.

Mutación()

El sistema determinará dos posibles tipos de mutación (indicada por el usuario en configuration):

Disruptiva:

Esta mutación el sistema lo realizará con una probabilidad de un 2% o lo que es lo mismo, con la probabilidad de mutación=0.02.

Si entra en este proceso, se selecciona una posición al azar del cromosoma, que será la que se intercambie, y se intercambiará por una casilla accesible por AP al azar, teniendo en cuenta que no se repita en el cromosoma ni que sea la misma que se sustituye.

No disruptiva:

Esta mutación el sistema lo realizará con una probabilidad de un 10% o lo que es lo mismo, con la probabilidad de mutación=0.1.

Si entra en este proceso, se selecciona una posición al azar del cromosoma, que será la que se intercambie, pero en esta ocasión se intercambiará por una casilla próxima a la sustituida (dentro de un intervalo), de manera que este tipo de mutación es mucho mas suave que la anterior. Todo esto, teniendo en cuenta que no se repita en el cromosoma ni que sea la misma que se sustituye.

Reemplazo():

Antes de entrar aquí el sistema evalúa los individuos hallados con el cruce y la mutación de padres (hijos), es decir le pone a estos el fitness1 y 2 (los evalúa).

El sistema determinará dos posibles tipos de reemplazo (indicada por el usuario en configuration):

- Por Padres: Se evalúan los dos hijos obtenidos y los dos padres y en la población se quedarán los dos mejores.
- Por torneo: Se realizará un torneo con el tamaño establecido, es decir, con el mismo tamaño que en la selección de padres (igual que en la selección de padres pero quedándose con el peor en vez de con el mejor del sorteo), pero en esta ocasión competirán con los hijos obtenidos y el sistema se quedará con los dos mejores.

El proceso por el cuál se determina los dos mejores serán los dos que tengan mejor fitness1 y en caso de empate de 3 o de los 4, se determinará por los mejores fitness2.

Una vez se realiza una iteración del algoritmo, si ha existido un reemplazo, el sistema vuelve a evaluar la población para ver si el mejor individuo cambia (es decir el Individuo que ha reemplazado pasa a ser el mejor) o sigue siendo el mismo. En caso de seguir siendo el mismo durante 500 iteraciones el algoritmo genético se acaba. Otro caso en el que el algoritmo genético acaba es si el usuario pulsa el botón stop de *Best solution options*.

FUNCIÓN DE INTERES: GRADIENTE DE COBERTURA:

Esta función de la interfaz provee mucha más información al usuario, de manera que le indica las zonas que la señal de cada solución alcanza y con qué potencia.

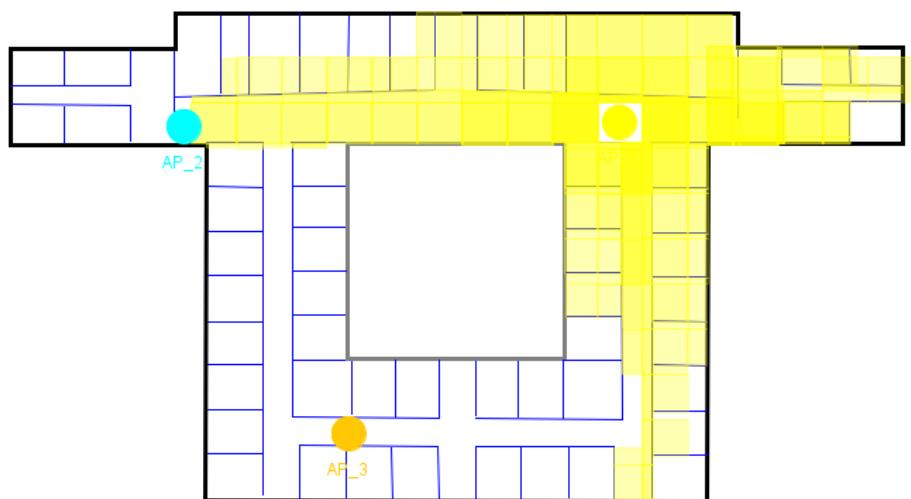


Imagen 3.35. Gradiente de cobertura

3.5. DESPLIEGUE

PAQUETE DE DESPLIEGUE:

Este paquete contiene:

- Carpeta con planos para pruebas
- Manual de Usuario
- Ejecutable WFSE
- Otros archivos
- Sub-paquete con el proyecto java y código fuente.

Para lanzar la aplicación debemos de ejecutar el “.jar” ejecutable (WFSE).

CAPÍTULO 4: EXPERIMENTACIÓN Y RESULTADOS

Para cerciorar que todos los detalles de diseño de la aplicación han tenido un estudio y una aplicación adecuados, en el presente punto se detallan los casos de estudios comparando los parámetros indicados y, en función de éstos, el resultado obtenido.

Los parámetros que se tendrán en cuenta para la obtención del resultado serán los siguientes:

- ❖ Tipo de tecnología empleada (cualquiera del estándar IEEE 802.11).
- ❖ Plano del edificio concreto.
- ❖ Parámetros del algoritmo genético:
 - Semilla. (Usadas para los casos aleatorios).
 - Nº APs.
 - Tipo de mutación.
 - Tipo de remplazo.
 - Tamaño del torneo.
 - AP Positions.

Para experimentar los distintos casos, tendremos que cambiar únicamente uno de estos parámetros y entonces comparar los resultados.

4.1. CASO DE ESTUDIO: CAPA FÍSICA

En primer lugar indicaremos al sistema dos simulaciones de un mismo edificio con todos los parámetros iguales salvo el tipo de tecnología (capa física) empleada:

- ➔ *Plano: Torreumbría 1 Planta*
- ➔ *Semilla: Sem1*
- ➔ *Nº Aps: 3*
- ➔ *Tipo Mutación: Disruptiva.*
- ➔ *Tipo Remplazo: Por padres.*
- ➔ *Tamaño del torneo: 5%.*
- ➔ *APs Positions: At the walls.*

Tecnología empleada: 802.11 b

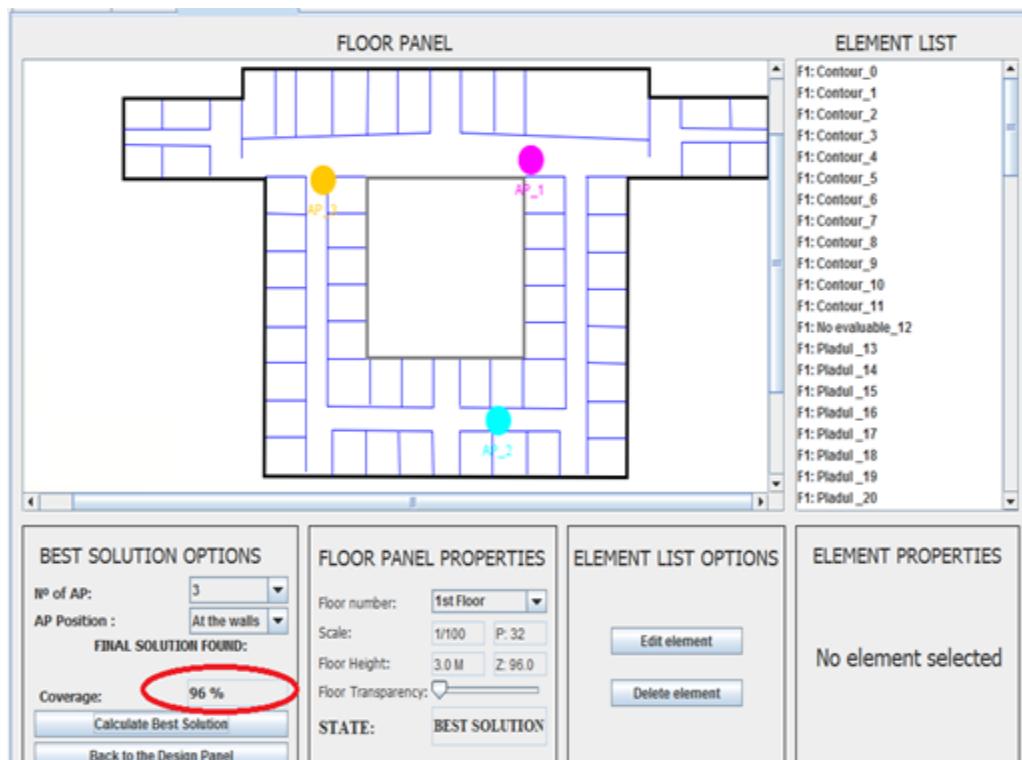


Imagen 4.1. Best Solution. Tecnología 802.11b

Como vemos con este tipo de tecnología (capa física 802.11 b) y los parámetros indicados previamente la cobertura obtenida es de 96%. Es decir, llega a casi todos los puntos del edificio.

Tecnología empleada: 802.11 a

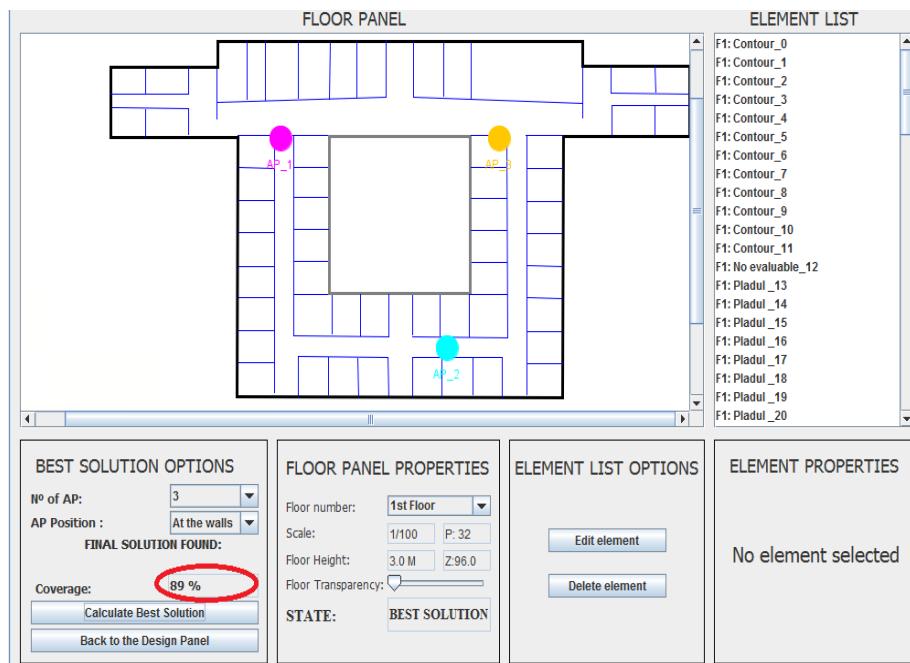


Imagen 4.2. Best Solution. Tecnología 802.11a

Como vemos con esta tecnología (capa física 802.11 a) tendremos alcance en el 89% de las zonas del edificio, menor respecto a la capa física 802.11b que era el 96%.

Por lo tanto, se aprecia claramente que el tipo de tecnología empleada es uno de los parámetros cruciales para determinar el resultado.

4.2. CASO DE ESTUDIO: NÚMERO DE APs

La siguiente experimentación estudia el parámetro referente al número de APs que queremos poner en el edificio. Esta característica es importante, ya que el sistema nos hallará una solución aproximada a la mejor (o la mejor), en función del número de APs que dispongamos.

- ➔ *Plano: Codos Y Muñones*
- ➔ *Semilla: Sem1*
- ➔ **Nº Aps: 2**
- ➔ *Tipo Mutación: Disruptiva.*
- ➔ *Tipo Reemplazo: Por padres.*
- ➔ *Tamaño del torneo: 5%.*
- ➔ *APs Positions: At the walls.*

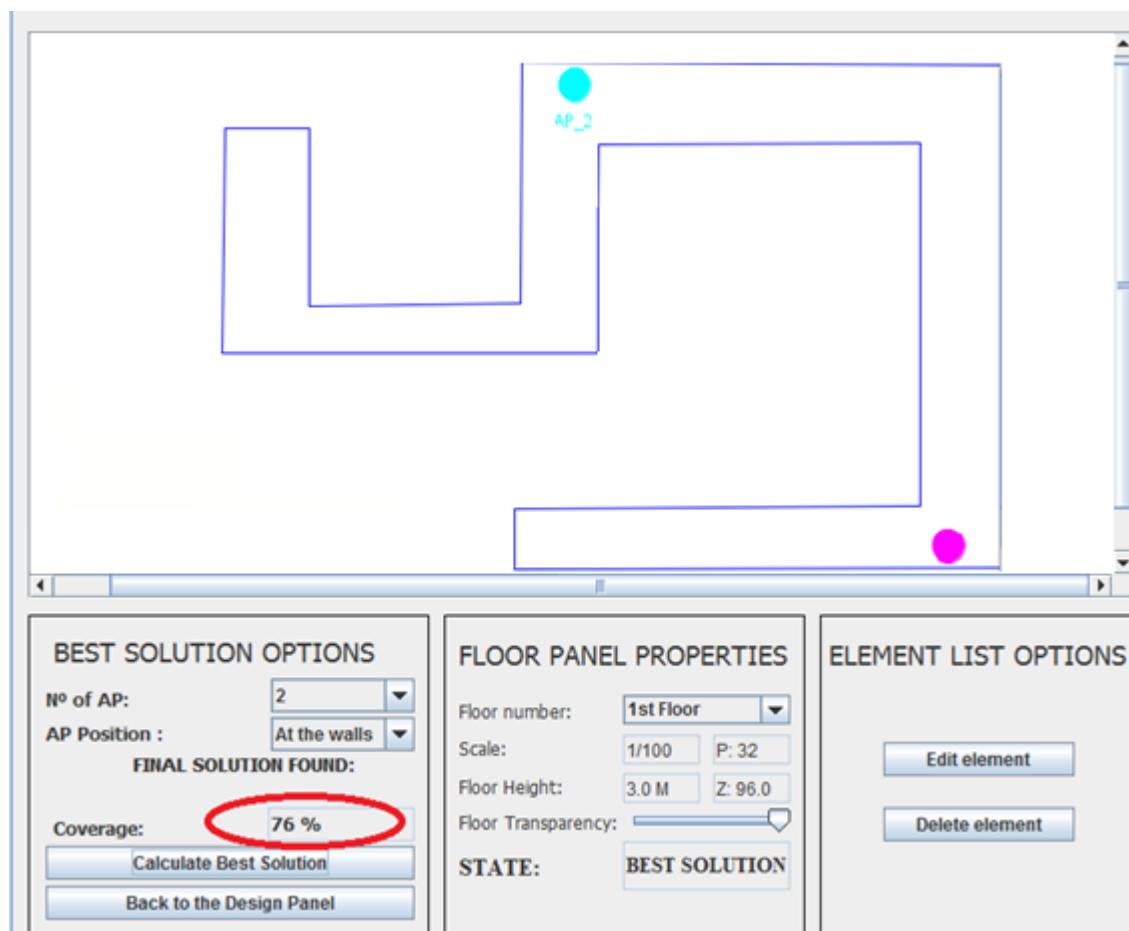


Imagen 4.3. Best Solution. Dos APs

- ➔ *Plano: Codos y muñones*
- ➔ *Semilla: Sem1*
- ➔ **Nº Aps: 3**
- ➔ *Tipo Mutación: Disruptiva.*
- ➔ *Tipo Reemplazo: Por padres.*
- ➔ *Tamaño del torneo: 5%.*
- ➔ *APs Positions: At the walls.*

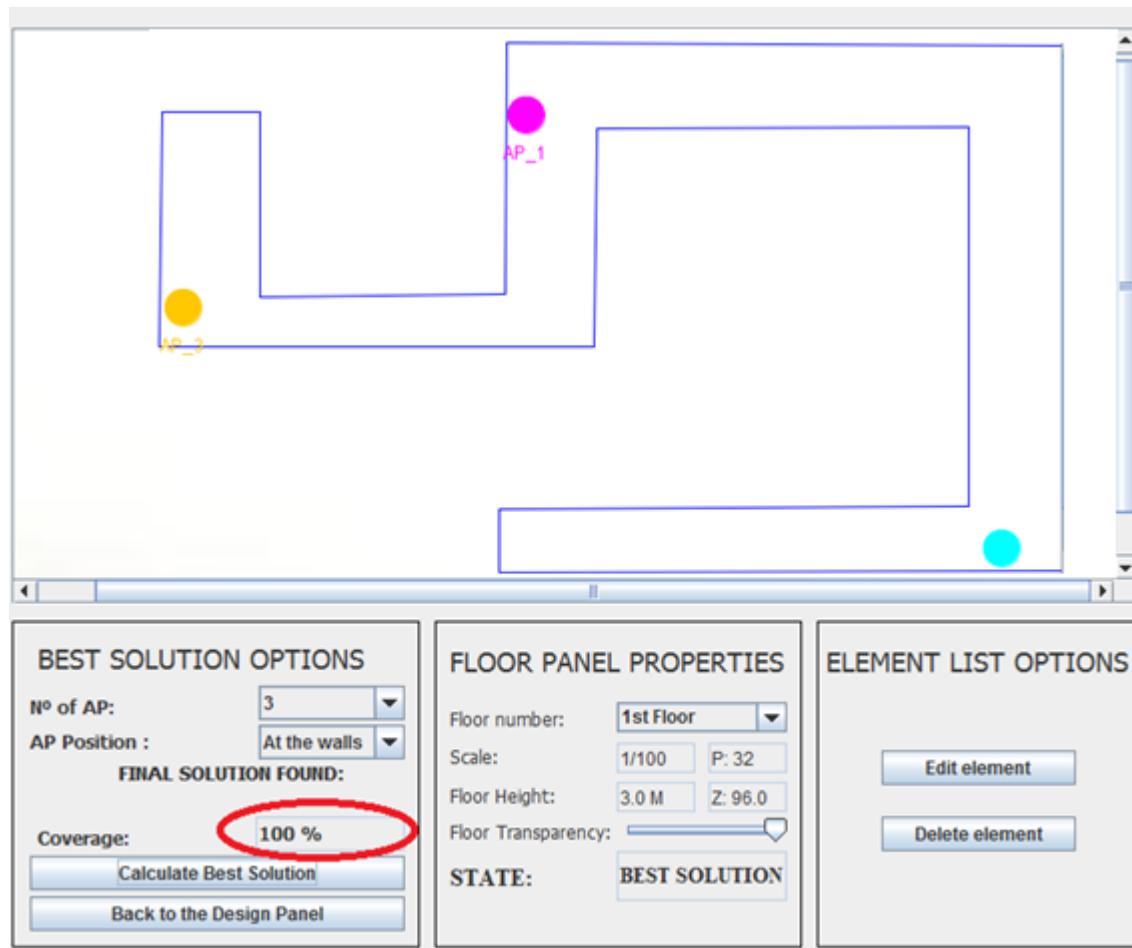


Imagen 4.4. Best Solution. Tres APs

Como vemos el número de APs que utiliza el edificio será primordial para tener más o menos cobertura. Este parámetro sigue una razón directamente proporcional a la cobertura del edificio, teniendo más cobertura cuanto más APs utilicemos.

4.3. CASO DE ESTUDIO: APs POSITION

La siguiente prueba muestra una comparativa en la que los APs sólo puedan ubicarse en las paredes respecto de aquellos que puedan ponerse en cualquier parte del plano.

- ➔ *Plano: Reloj Arena*
- ➔ *Semilla: Sem1*
- ➔ *Nº Aps: 2*
- ➔ *Tipo Mutación: Disruptiva.*
- ➔ *Tipo Reemplazo: Por padres.*
- ➔ *Tamaño del torneo: 5%.*
- ➔ **APs Positions: At the walls.**

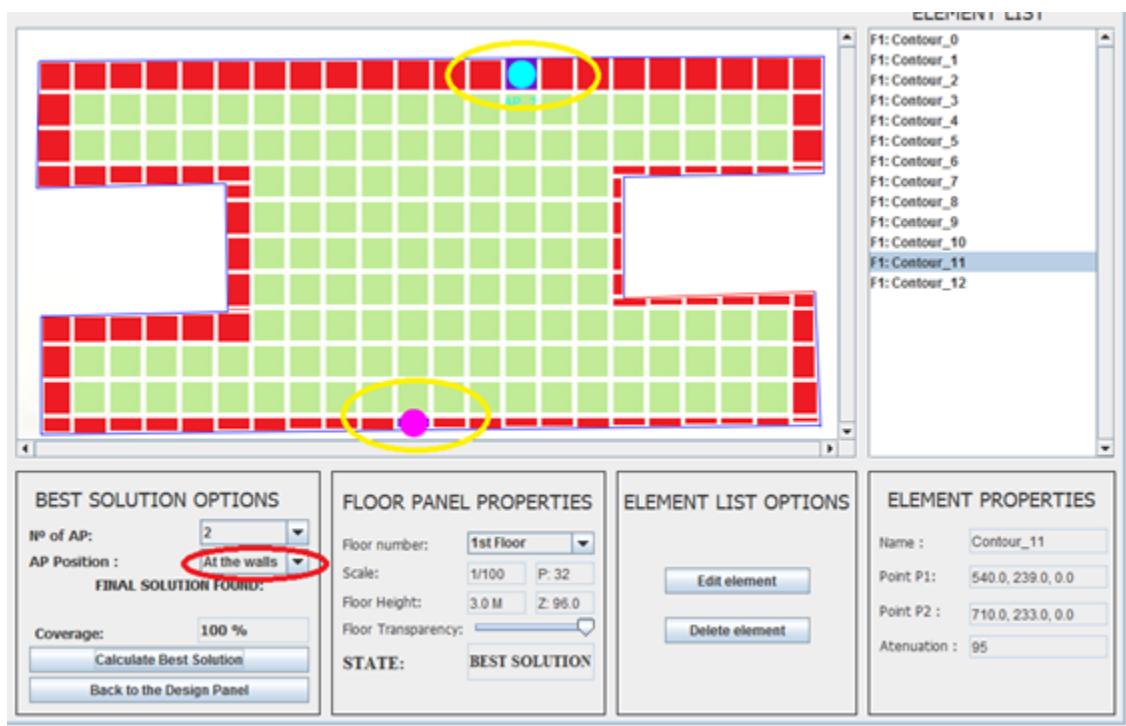


Imagen 4.5. Best Solution. At the walls

- ➔ *Plano: Reloj Arena*
- ➔ *Semilla: Sem1*

- ➔ Nº Aps: 2
- ➔ Tipo Mutación: Disruptiva.
- ➔ Tipo Reemplazo: Por padres.
- ➔ Tamaño del torneo: 5%.
- ➔ APs Positions: At Anywhere.

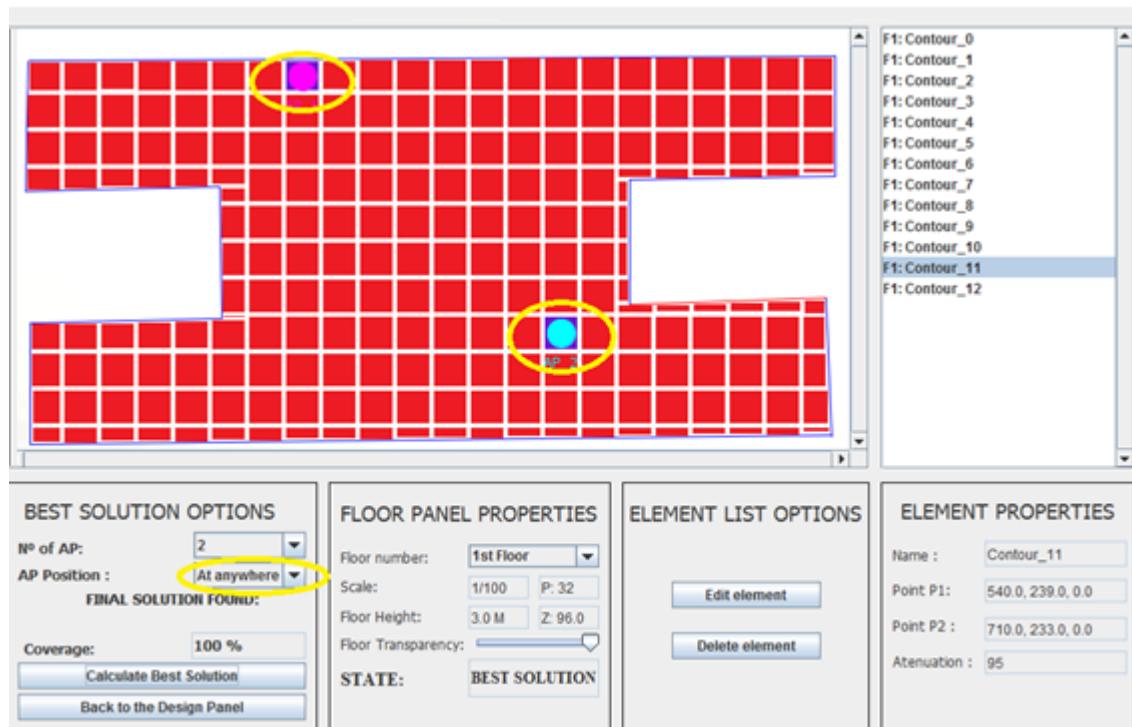


Imagen 4.6. Best Solution. At everywhere

Como podemos comprobar, en la segunda figura existen muchas más opciones para colocar los APs (todas las casillas rojas), por lo que evidencia que habrá una mejor solución más cerca del óptimo que cuando éstas sólo puedan ponerse en las paredes. Como contrapartida, podríamos indicar que el algoritmo genético tardará más en converger puesto que se precisa un campo de espacio de búsqueda mucho mayor.

4.4. CASO DE ESTUDIO: MUTACIÓN

La siguiente prueba se va a centrar en el parámetro tipo de mutación, en el cuál el usuario puede elegir el tipo de mutación que realizará el algoritmo genético implementado por el sistema, bien sea disruptiva o no disruptiva. Es preciso indicar que “jugando” o, mejor dicho, “probando” con los parámetros del genético pueden alcanzarse soluciones que se acerquen o alejen más del óptimo

- ➔ *Plano: Torreumbría una planta*
- ➔ *Semilla: Sem1*
- ➔ *Nº Aps: 3*
- ➔ **Tipo Mutación: Disruptiva.**
- ➔ *Tipo Remplazo: Por padres.*
- ➔ *Tamaño del torneo: 5%.*
- ➔ *AP's Positions: At the walls.*

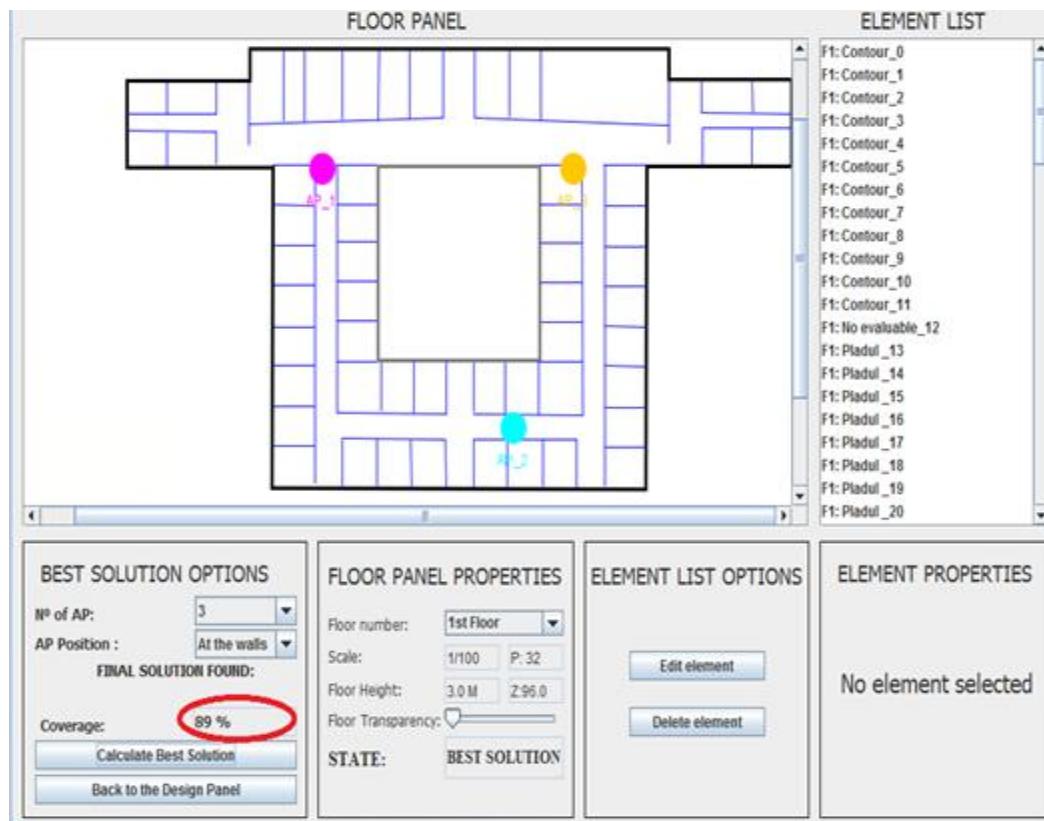


Imagen 4.7. Best Solution. Mutación Disruptiva

- ➔ Plano: Torreumbria una planta
- ➔ Semilla: Sem1
- ➔ Nº Aps: 3
- ➔ **Tipo Mutación: No Disruptiva.**
- ➔ *Tipo Remplazo: Por padres.*
- ➔ *Tamaño del torneo: 5%.*
- ➔ *APs Positions: At the walls.*

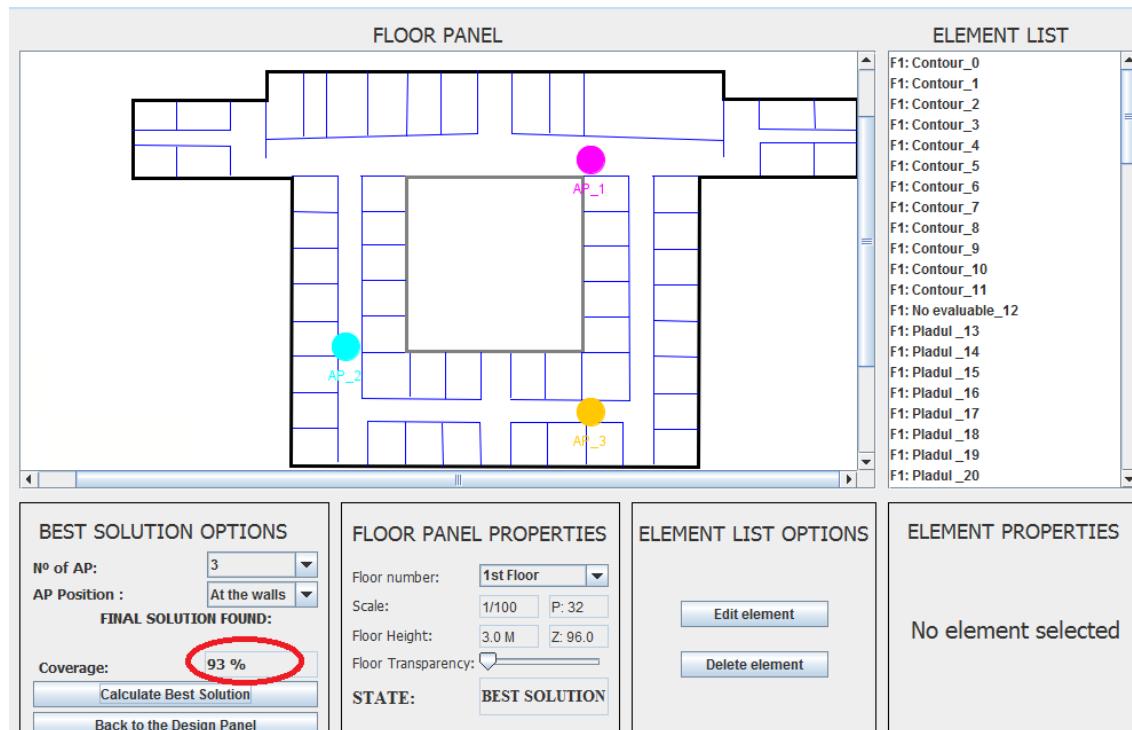


Imagen 4.8. Best Solution. Mutación No Disruptiva

En vista a estos resultados podríamos considerar que la mutación no disruptiva funciona mejor que la disruptiva. Sin embargo, en base a las numerosas pruebas realizadas una y otra mutación dependerá de los demás parámetros de configuración de WiFiSim, ya que si por ejemplo cambiamos el tipo de tecnología a la 802.11b y mantenemos el resto de parámetros intactos los resultados de la mutación disruptiva serán mejor:

MUTACIÓN DISRUPTIVA:

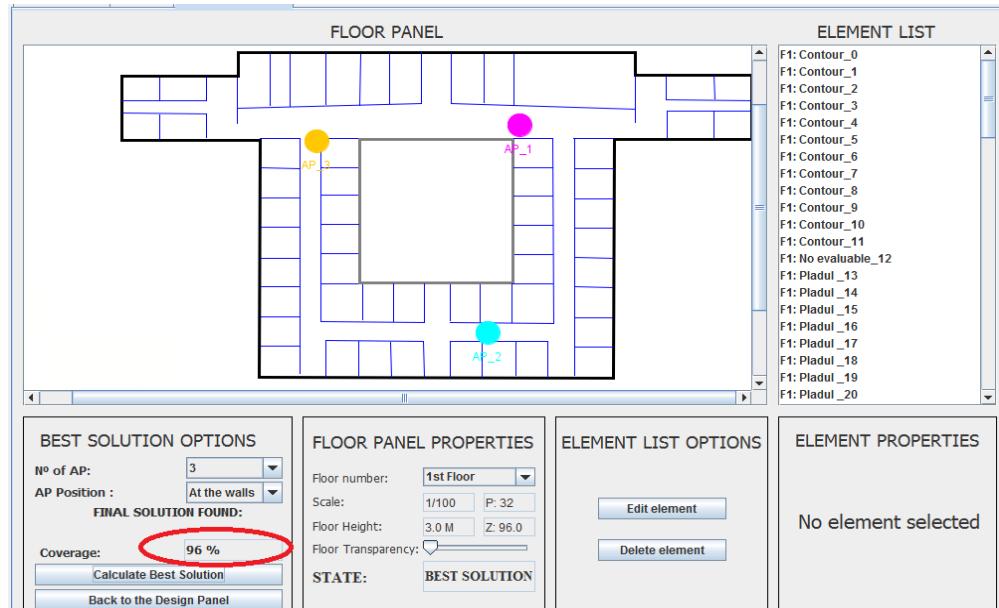


Imagen 4.9. Best Solution. Mutación Disruptiva II

MUTACIÓN NO DISRUPTIVA:

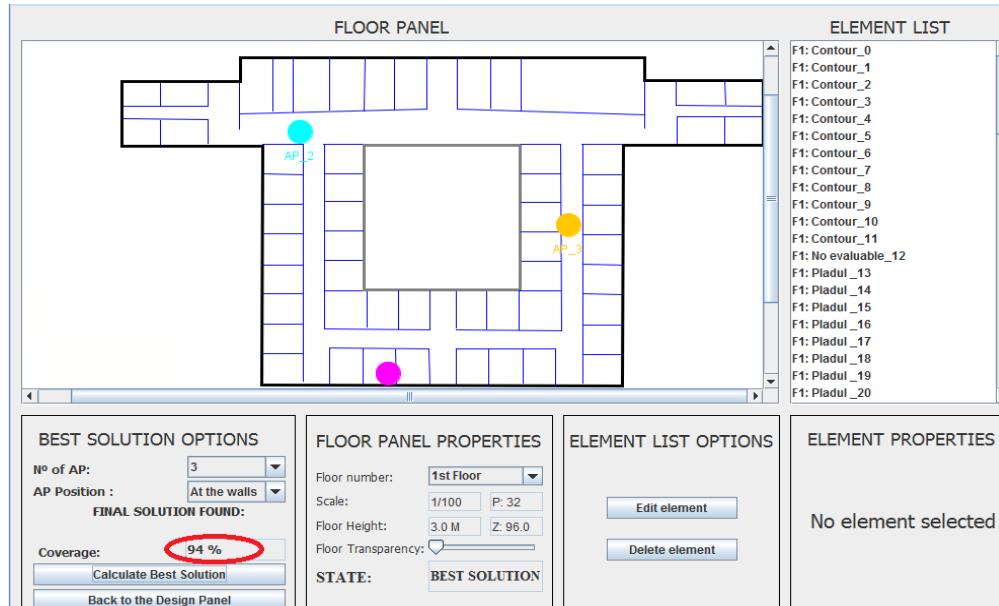


Imagen 4.10. Best Solution. Mutación No Disruptiva II

Como podemos apreciar en el estudio del cambio de la mutación, este parámetro es determinante a la hora de alcanzar una solución. No siempre un tipo de mutación es mejor que la otra, sino que los dos son buenos mecanismos de mutación (la disruptiva con porcentaje 2% y la no disruptiva con porcentaje 10% pero siendo menos brusca). Por tanto, se concluye con que habrá que “probar” con el cambio de este parámetro (así como de otros del algoritmo genético) para hallar la solución más próxima a la óptima.

4.5. CASO DE ESTUDIO: REMPLAZO

Al igual que en la experimentación realizada anteriormente, el estudio de los otros parámetros que el usuario pueda elegir en la ventana de configuración del algoritmo genético arroja diversos resultados como por ejemplo, el tamaño del torneo así como el remplazo por padres o por torneo.

En primer lugar nos centraremos en el tipo de remplazo, que evidentemente dependerá también del tamaño del torneo, siempre para la selección de padres y en el caso de remplazo por torneo así que haremos dichas pruebas conjugando estos dos parámetros:

- ➔ *Plano: Torreumbría 1 Planta*
- ➔ *Tecnología: IEEE 802.11a*
- ➔ *Semilla: Sem1*
- ➔ *Nº Aps: 3*
- ➔ *Tipo Mutación: Disruptiva.*
- ➔ *Tipo Remplazo: Por padres.*
- ➔ *Tamaño del torneo: 5%.*
- ➔ *APs Positions: At the walls.*

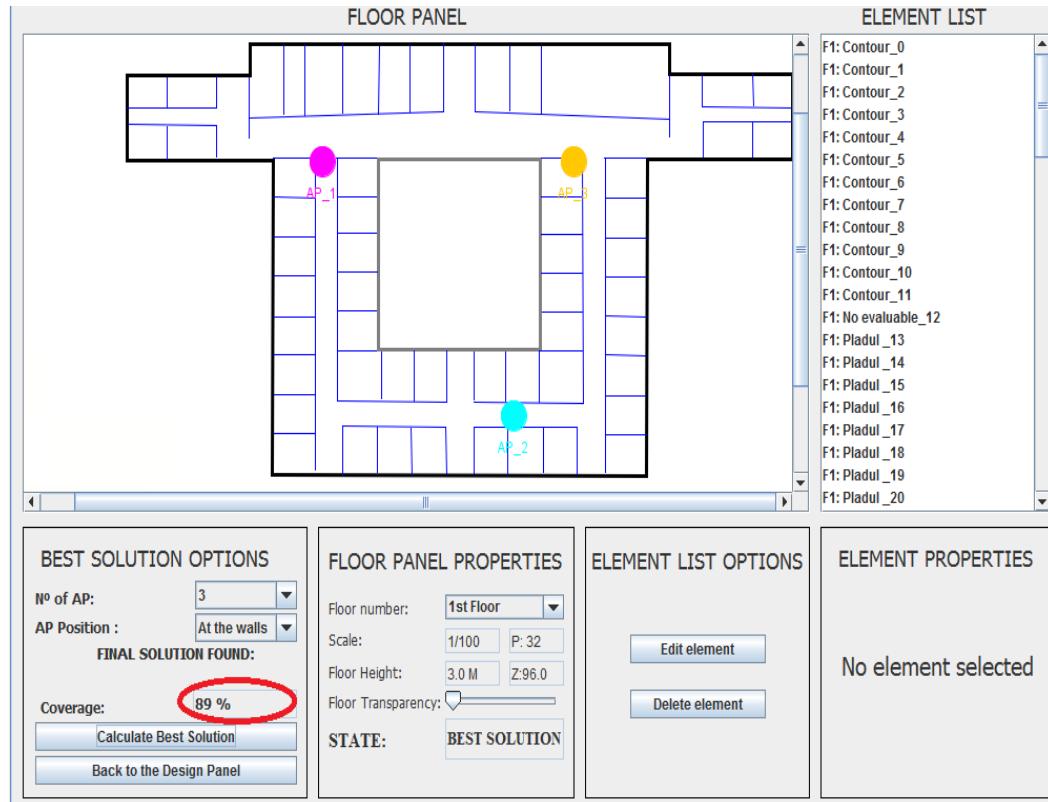


Imagen 4.11. Best Solution. Remplazo por padres

- ➔ *Plano: Torreumbría 1 Planta*
- ➔ *Tecnología: IEEE 802.11a*
- ➔ *Semilla: Sem1*
- ➔ *Nº Aps: 3*
- ➔ *Tipo Mutación: Disruptiva.*
- ➔ *Tipo Reemplazo: Por torneo.*
- ➔ *Tamaño del torneo: 5%.*
- ➔ *APs Positions: At the walls.*

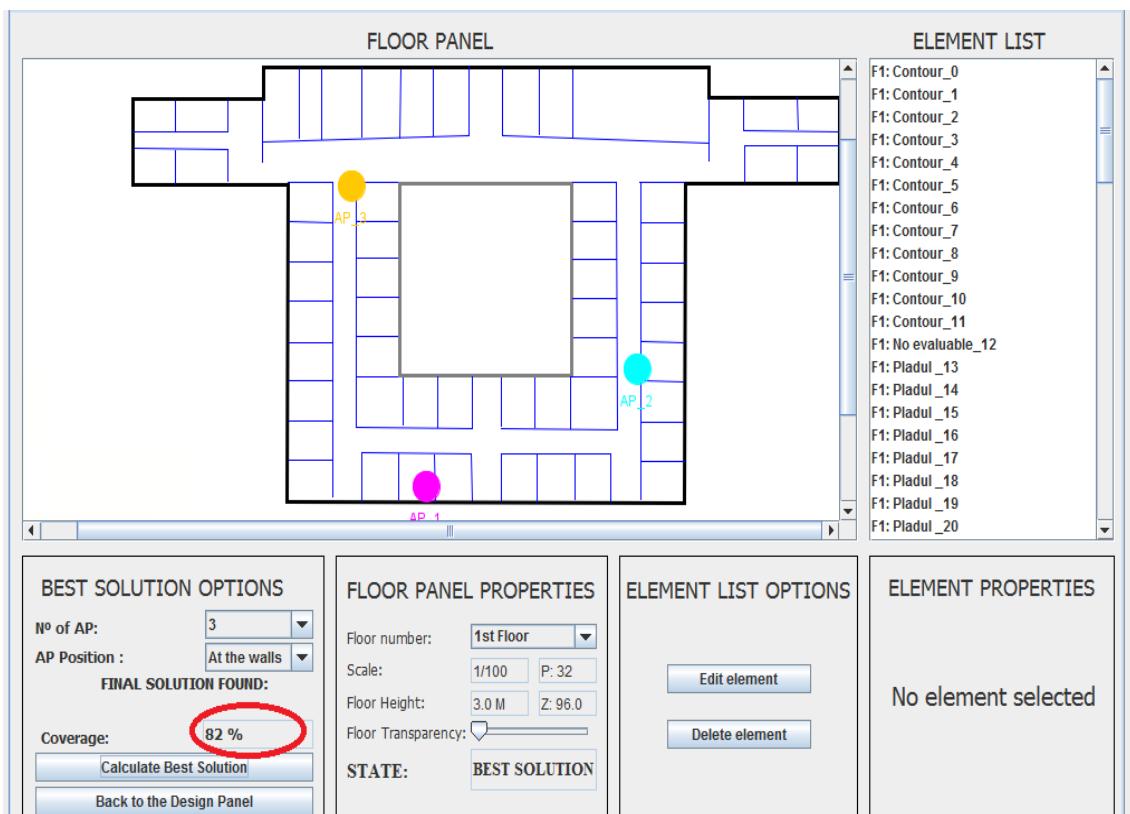


Imagen 4.12. Best Solution. Reemplazo por Torneo

En base a los resultados obtenidos vemos que se comporta mejor el remplazo por los padres, puesto que de esta manera se gana en diversidad y se explora mayor espacio de búsqueda. Podríamos afirmar que el tipo de remplazo dependerá del tipo de plano que se está tratando, de la tecnología empleada así como del número de APs. Por tanto, éste será un parámetro a ajustar para hallar la solución más próxima a la óptima.

En base a las pruebas realizadas anteriormente, cuyas conclusiones las trataremos en el capítulo siguiente, podríamos dar una visión general acerca de los parámetros a ajustar y los parámetros decisivos para la mejora:

Parámetros decisivos:

De forma resumida, se podrían definir a continuación aquellos parámetros que favorecen encontrar una solución óptima manteniendo siempre la misma configuración. Éstos son:

- ➔ Tipo de tecnología: siempre tendrá más cobertura aquella a la que más lejos llegue, independientemente de los demás parámetros.
- ➔ Posición de los APs: siempre se obtendrá una solución mejor o igual si los APs pueden colocarse en cualquier lugar del plano.
- ➔ Nº de APs: siempre se tendrá una solución mejor con mayor número de APs.
- ➔ Plano: siempre se tendrá en cuenta los mismos obstáculos y distancias tratadas.

Parámetros a ajustar:

A continuación se definen a aquellos parámetros que encontrarán una solución mejor o peor en función de la figura, la semilla, el tipo de tecnología, etc., y la conjunción entre ellos. Por tanto son parámetros con los que el usuario podrá interactuar para estudiar la mejor solución. Éstos son:

- ➔ Tipo de mutación: habrá casos para los que funcione mejor un tipo de mutación y habrá casos para los que funcione mejor la otra.
- ➔ Tipo de remplazo: habrá casos para los que funcione mejor un tipo de remplazo y habrá casos para los que funcione mejor el otro.
- ➔ Tamaño torneo: habrá casos para los que funcione mejor un tamaño y habrá casos para los que funcione mejor otros.

4.6. CASO DE ESTUDIO: DISEÑO WiFi REAL

Para demostrar la optimización de diseños WiFi que WiFiSimExtension proporciona, en este apartado se realiza un estudio de un diseño WiFi real, y se compara con el diseño que el sistema obtiene.

El caso de estudio se desarrollará sobre el edificio donde se encuentran los despachos del personal de los departamentos DIESIA y DTI, el edificio *Torreumbría*, localizado en la E.P.S. La Rábida, de la Universidad de Huelva.

Según la siguiente imagen el citado edificio tiene tres APs visibles (existen dos APs más ocultos aunque se desconoce si están o no operativos). Por tanto, el caso de estudio se centrará en el diseño del edificio *Torreumbría* con tres APs. Estos están distribuidos de la siguiente forma:

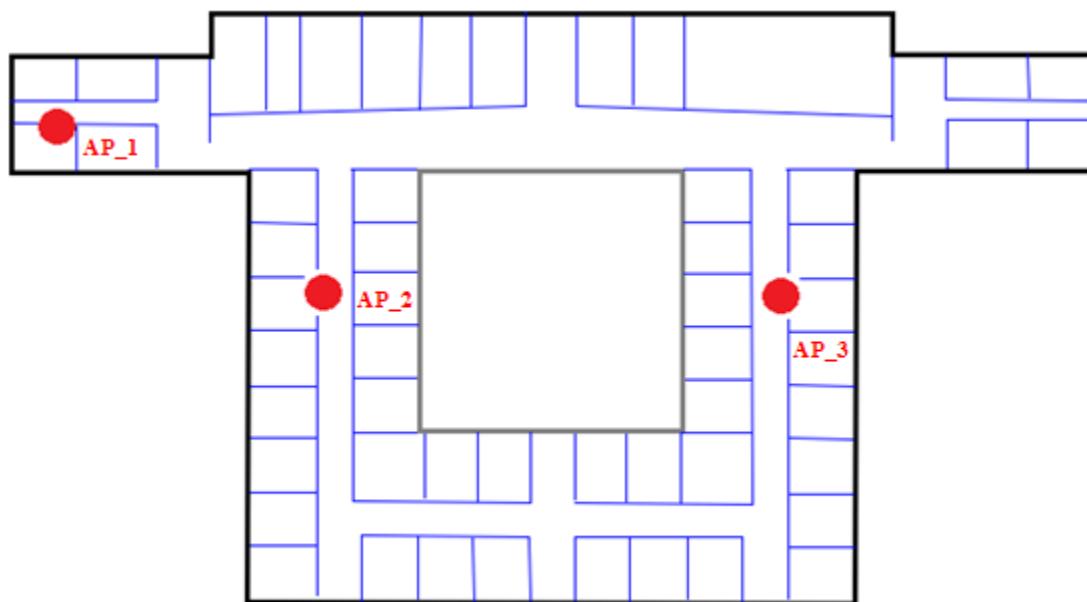


Imagen 4.13. APs Torreumbría

Según el plano, la distribución de los APs, y el tipo de obstáculos que intervienen (considerando las paredes negras como contorno con atenuación (95%) y las azules como pladul (8%)), podríamos predecir un gráfico de cobertura aproximado como se muestra en la imagen:



Imagen 4.14. Cobertura Torreumbría

Si observamos detenidamente el gráfico, podemos observar que existen muchas zonas sin cobertura, así como zonas donde la potencia es muy baja. Evidentemente se llega a la conclusión de que los APs no están colocados de la mejor manera posible.

Mediante el uso de la aplicación WiFi Analyzer (referencia), que detecta la cobertura y potencia de redes inalámbricas en una localización concreta, se ha podido corroborar que hay zonas sin cobertura y con baja potencia. Los resultados obtenidos han sido los siguientes:

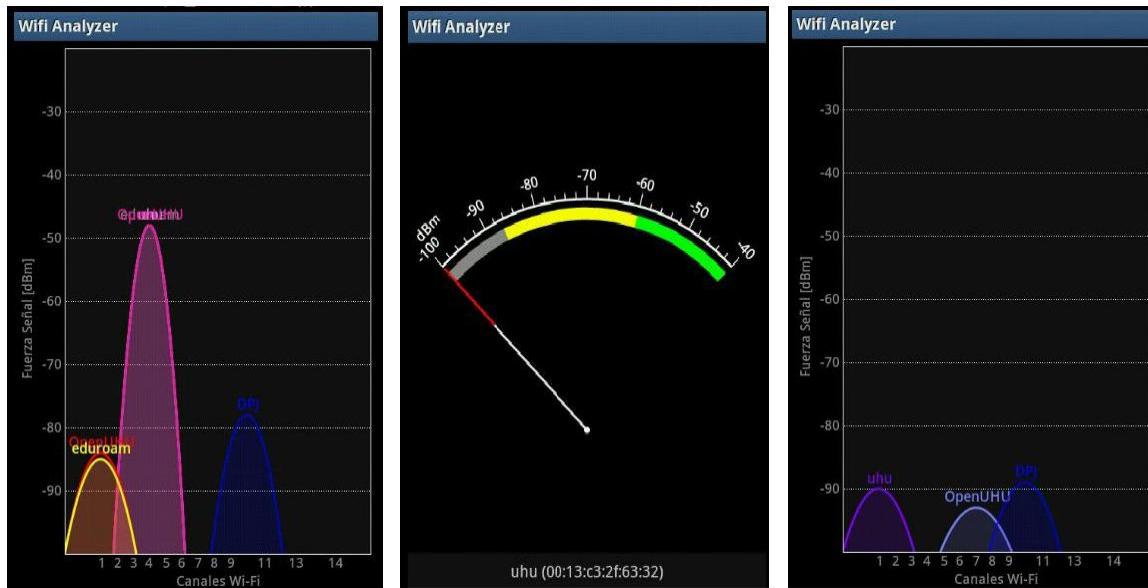


Imagen 4.15. Cobertura Torreumbría WiFi Analyzer

Con WiFi Analyzer se han obtenido resultados de diferente tipo. Desde zonas con cobertura donde la potencia llega con una potencia alta (imagen de la izquierda), hasta zonas donde no hay nada de cobertura (imagen central). Sin embargo, otras zonas tienen cobertura aunque la señal llega con una potencia media/baja (imagen de la derecha).

Por lo tanto nos encontramos ante un diseño WiFi con una solución no óptima. A continuación se mostrará el comportamiento de WiFiSimExtension ante este problema de localización de APs.

¿Cual es la mejor forma de distribuir los APs disponibles, tres en nuestro caso, para que haya cobertura en el máximo número de zonas posibles y que además estas zonas lleguen con la mejor intensidad posible?

Mediante el uso de la función *BestSolution* de WiFiSimExtension creamos la estructura del edificio y le indicamos al sistema el número de APs que deseamos. Tras los correspondientes cálculos WiFiSimExtension devuelve la siguiente distribución de APs:

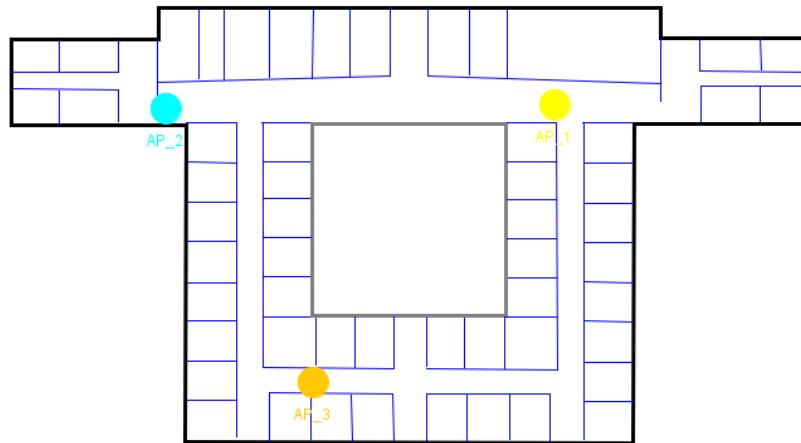


Imagen 4.16. Solución Torreumbría 3 APs

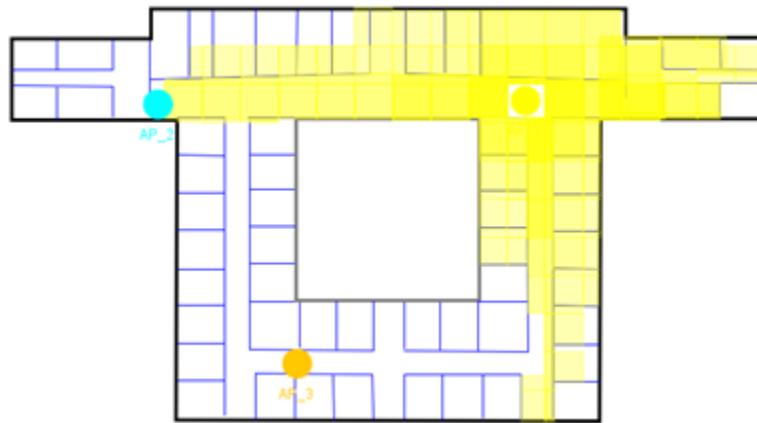


Imagen 4.17. Solución Torreumbría 3 APs. Gradiente AP_1

Como observamos en la imagen, se ha obtenido una solución con esta distribución la cual llegaría a más zonas que con la distribución actual, aunque tampoco sería suficiente para llegar a todas las zonas del edificio, puesto que el edificio es demasiado grande y además tiene dos plantas. Con la attenuación de los obstáculos que se han tenido en cuenta durante la simulación solo se ha obtenido un porcentaje de cobertura igual a 80%.

Si por el contrario queremos saber cuantos APs serían necesarias para que hubiera cobertura en todo el edificio, tendríamos que usar un número mayor de APs y realizar varias simulaciones (ya que según la teoría de algoritmos genéticos (véase Anexo), no se puede sacar conclusiones de una única ejecución), así como interactuar con los parámetros del genético.

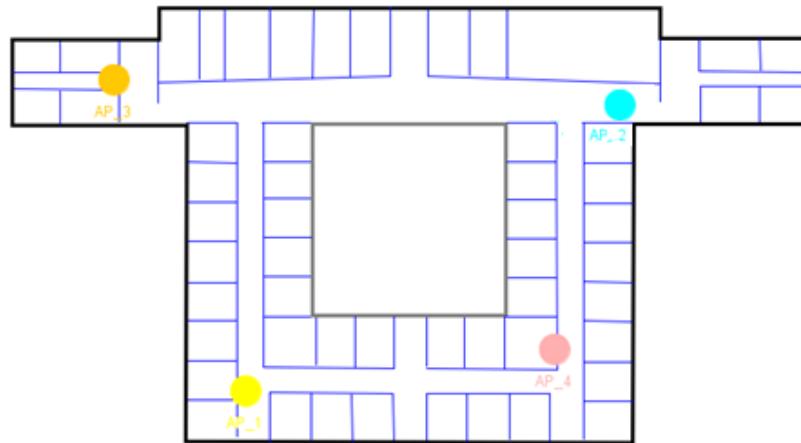


Imagen 4.18. Solución Torreumbría 4 APs.

Esta solución tendría 4 APs y todo el edificio tendría cobertura (100%), sin embargo si volviésemos a calcular una solución con 4 APs el algoritmo genético con otros parámetros obtendríamos otra solución diferente:

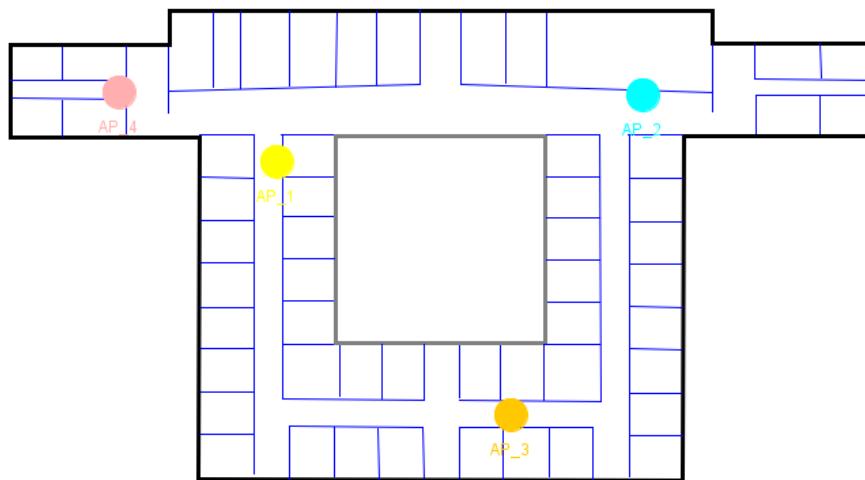


Imagen 4.19. Solución Torreumbría 4 APs II.

Esta solución tiene un porcentaje de cobertura igual a 98%, no cubre todas las zonas pero es una solución muy próxima a la óptima.

En consecuencia a los resultados obtenidos en el caso de estudio realizado sobre el edificio *Torreumbría*, podemos obtener las siguientes conclusiones:

- El diseño del edificio no muestra una solución óptima, pues se ha demostrado mediante dos aplicaciones diferentes, WiFiSimExtension y WiFi Analyzer que existen muchas zonas sin cobertura y con una mala señal.
- WiFiSimExtension ha obtenido una solución óptima con 4 APs y una solución mejor que la original con 3 APs.
- Si los dos APs ocultos del edificio estuviesen activos, podríamos decir que es un gasto innecesario instalar 5 APs, puesto que con 4 nuestro sistema es capaz encontrar una solución óptima que cubriría el 100% de las zonas del edificio.
- El sistema no siempre haya la mejor solución, sin embargo, la solución obtenida será o estará muy cerca de la óptima.

Realmente, existen zonas en el edificio *Torreumbría* donde no existe cobertura WiFi con el diseño actual, por tanto, **invitamos a los responsables de los departamentos de DIESIA y DTI a cambiar la distribución de estos según la solución que WiFiSimExtension proporciona.**

CAPÍTULO 5:

CONCLUSIONES Y AMPLIACIONES FUTURAS

Las comunicaciones inalámbricas se han impuesto a las redes convencionales gracias a ventajas tales como el bajo coste, fácil instalación, gran movilidad y alcance de la red. Sin embargo, las redes WiFi tienen también desventajas tales como la considerable disminución de la calidad de la señal o incluso la pérdida total de ésta. Esto puede deberse al entorno donde se encuentra desplegada la red WiFi, la distancia y los obstáculos pueden llegar a ser factores determinantes.

Este proyecto se ha centrado en el estudio de estos factores, como también, en la planificación, optimización y el despliegue de redes inalámbricas, proporcionando así un diseño WiFi adecuado y con garantías. Para ello, se ha implementado una herramienta de diseño que obtiene la solución óptima, mediante el uso de algoritmos genéticos, teniendo en cuenta la mayor zona de cobertura y potencia de señal posible.

Las raíces de este proyecto se encuentran en WiFiSim la aplicación creada por César Serrano López, que tiene como objetivo el diseño y construcción de un simulador de redes inalámbricas basadas en el estándar IEEE 802.11. Nuestra aplicación esta implementada modular e independientemente a WiFiSim, aunque está integrada en su misma interfaz. Sin embargo, WiFiSimExtension posee su propia interfaz que funciona sin ningún tipo de dependencia externa. Gracias a la modularidad se puede realizar la aplicación completa sin influir en el código interno de la aplicación inicial, y de este modo otras ampliaciones serán posibles fácilmente.

Entre los objetivos funcionales del proyecto se encontraban algunos como suplir las desventajas de WiFiSim y obtener un diseño que proporcione una solución óptima de redes inalámbricas. Objetivos que se han obtenidos con creces, proporcionando un sistema dinámico, funcional e interactivo, que además resulta bastante atractivo y utilizable de cara al usuario.

Los objetivos profesionales y de trabajo en equipo también han resultado altamente satisfactorios:

- ❖ Ha sido realizada una correcta colección de requisitos de los clientes, e incluso se han añadidos muchos más de parte de los desarrolladores, con la mirada puesta en la consecución de un proyecto mejor.
- ❖ Se ha llevado a cabo una revisión semanal con los tutores del proyecto, donde en conjunto, se han analizado varios asuntos, se ha valorado el diseño y el comportamiento del sistema, han surgido nuevos requisitos y se han propuesto nuevas ideas y mejoras con un resultado al gusto de todas las partes implicadas.
- ❖ La colaboración y planificación de los componentes del grupo ha resultado un éxito rotundo, superando las expectativas. Se ha trabajado con esfuerzo, seriedad, respeto y compañerismo, explotando las ventajas del trabajo en equipo y consiguiendo así además de la construcción del proyecto, forjar una gran amistad.
- ❖ Se ha seguido correctamente el ciclo de vida de desarrollo software elegido, obteniendo gracias a éste, experiencia en una y cada una de sus tareas.
- ❖ Otro de los éxitos de éste proyecto ha sido la satisfacción del cliente, ya que hemos conseguido crear una aplicación que será de gran utilidad y probablemente de ayuda en futuros proyectos.
- ❖ Uno de los objetivos del proyecto era una entrega temprana, que ha sido conseguida fruto de una gran dedicación, sin influir en la consecución de los objetivos principales.

Con todo lo anteriormente mencionado podemos llegar a la conclusión que la realización del proyecto WiFiSimExtension en grupo nos ha dotado de experiencia a la hora de trabajar en equipo y se ha hecho del entorno de trabajo, algo similar al de una empresa real de software.

Respecto a la mejora de las desventajas de WiFiSim, con nuestra aplicación se ha conseguido agilizar la tarea de configuración del entorno para realizar la simulación en WiFiSim. Se ha llevado a cabo el diseño de una interfaz gráfica capaz de añadir elementos WiFiSim (nodos, puntos de acceso y obstáculos) dinámica y gráficamente, haciendo la labor visual, más amena y rápida. La interfaz de WiFiSimExtension también proporciona la creación de diseños arquitectónicos para la creación de redes inalámbricas en espacios cerrados, así como su optimización y despliegue de éstas.

Además WiFiSimExtension está pensada para el ámbito académico proporcionando la posibilidad de configurar parámetros internos y ver el comportamiento del algoritmo genético que usa el sistema.

La construcción de la aplicación WiFiSimExtension ha supuesto desde el primer día, un verdadero reto para nosotros. Desde el inicio, allá en el mes de Julio, el tiempo total dedicado a la realización de nuestro proyecto en la suma de todas sus fases ha sido aproximadamente 450 horas por alumno, aprendiendo a utilizar nuevas disciplinas (por ejemplo Java2D, WindowBuilder, algoritmos genéticos, etc.), de las que hemos obtenido un amplio conocimiento en el desarrollo software. Todos los casos de uso han sido implementados satisfactoriamente y la arquitectura software ha quedado lista para que cualquier extensión del proyecto pueda ser realizada fácil y modularmente.

Después del minucioso estudio seguido en el edificio *Torreumbría*, hemos podido corroborar que nuestra herramienta puede resultar de gran utilidad no sólo en los ámbitos de redes y algoritmos genéticos, sino también en ámbitos cotidianos o bien profesionales, para tener en cuenta de qué manera distribuir los APs y dotar de buena cobertura a todo el edificio. Además se ha podido probar el realismo que tiene la aplicación poniendo en entredicho diseños reales de redes inalámbricas.

Resumiendo, tras la finalización del proyecto estamos muy contentos del trabajo realizado. Cuando pensamos en WiFiSimExtension no podemos evitar verla como un “diamante en bruto”, de la cual nos sentimos muy orgullosos, y el hecho de que vaya a ser utilizada por futuros compañeros en clases impartidas por profesores implicados, así como de su publicación como una herramienta **eficiente y experimental**, nos llena de satisfacción.

Finalizamos resaltando que WiFiSimExtension ha abierto una línea de investigación dentro del Departamento de Ingeniería electrónica, sistemas informáticos y automática de la Universidad de Huelva, y eso, a nuestro favor, es mucho que decir.

AMPLIACIONES FUTURAS:

❖ VISION 3 PLANTAS

Una visión de todas las plantas del edificio sería lo ideal, ya que proporcionaría una visión total del diseño de la red y obtendríamos una interfaz mucho más atractiva.

❖ AÑADIR NUEVAS TECNOLOGIAS

Con el paso de los años las tecnologías usadas en el proyecto que siguen el protocolo 802.11 irán evolucionando, teniendo así nuevas tablas de sensibilidad entre otras características. Resultaría muy interesante darla a la aplicación la posibilidad de ir actualizando ésta lista de tecnologías disponibles.

❖ INFORME

La realización de un informe de un proyecto WiFiSimExtension en “PDF” o cualquier otro formato sería una característica idónea que haría de nuestra aplicación una herramienta mucho más profesional.

❖ ALGORITMO MULTI-OBJETIVO

Se podría implementar un algoritmo multi-objetivo, el cuál no solo tuviese un único factor en cuenta, es decir no solo se persiguiera el hecho de tener mayor cobertura e intensidad, sino además otros objetivos adicionales.

❖ WiFi ANALYZER

Sería una buena idea, realizar una ampliación para sincronizar WiFiSimExtension con WiFi Analyzer, (utilizado en el capítulo de experimentación).

❖ EDITAR PATRÓN DE RADIACIÓN DE ANTENA

El patrón de radiación de antena que usa WiFiSimExtension es bidireccional. Esta ampliación le daría aún más realismo a nuestra aplicación.

❖ EVALUACIÓN DE DISEÑO WiFi INSERTADO MANUALEMENTE

Se podría implementar una funcionalidad más, en el caso de que dentro de un edificio indicásemos los APs gráficamente y el programa te determinara como de buena es esta solución (esto lo hace WiFiSim y por tanto al haber hecho la sincronización completa entre ambas aplicaciones no hemos querido desarrollarla, pero se tratará en futuras versiones)

❖ SUBPROCESO GRID

Una de las tareas en las que WiFiSimExtension dedica más tiempo, es a la hora de crear el *grid* para dividir la estructura en celdas y mini-celdas. La aplicación se podría agilizar mediante la creación de un subproceso que fuera creando el *grid* en tiempo de ejecución, es decir, a la vez que se va creando la estructura. Las ventajas serían múltiples, ya que el algoritmo genético se ahorraría este paso y además la ampliación anteriormente descrita sería viable.

ANEXO

A) MANUAL DE USUARIO

❖ Ejecución:

Para lanzar la aplicación debemos de ejecutar el ‘jar’ ejecutable *WlanSimulator2.0*.

❖ Configuración:

Lo primero que nos encontramos al ejecutar nuestra aplicación será la ventana de WiFiSim, la cuál se dividirá en distintos paneles:

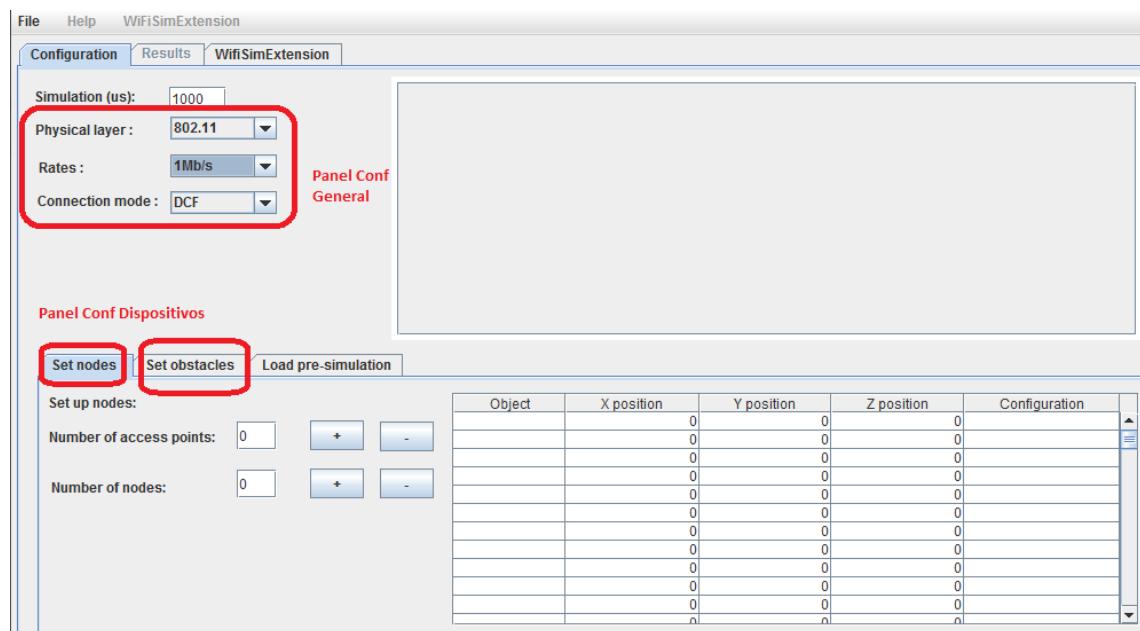


Imagen A.1. WiFiSim Panel de Configuración

Panel de Configuración General:

En el panel de configuración podemos seleccionar el tipo de tecnología que se usará para la simulación, cada tecnología tiene distintas velocidades dependiendo de su tabla de sensibilidad. Dicha tecnología puede ser seleccionada en la parte de la interfaz que se indica a continuación.

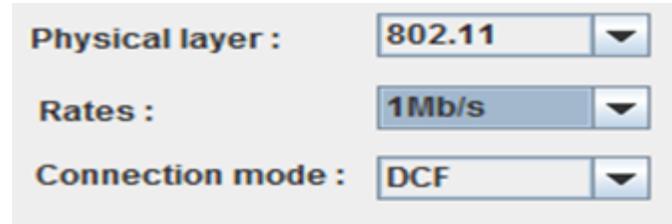


Imagen A.2. WiFiSim Panel de Tecnología 802.11

Panel Configuración Dispositivos:

En este panel se pueden configurar el número de nodos y APs, así como sus posiciones en el espacio (tres dimensiones).

The image shows a configuration panel for setting nodes. On the left, there are two sets of controls: "Number of access points" with a value of 0 and buttons for increasing (+) or decreasing (-); and "Number of nodes" with a value of 0 and similar buttons. To the right is a table with columns for "Object", "X position", "Y position", "Z position", and "Configuration". The table has 12 rows, each with values 0, 0, 0, and an empty configuration field.

Object	X position	Y position	Z position	Configuration
	0	0	0	
	0	0	0	
	0	0	0	
	0	0	0	
	0	0	0	
	0	0	0	
	0	0	0	
	0	0	0	
	0	0	0	
	0	0	0	
	0	0	0	

Imagen A.3. WiFiSim Panel. Set Nodes

Panel Configuración Obstáculos:

Desde esta pestaña podremos modificar y añadir obstáculos entre los dispositivos previamente insertados en la simulación.

The screenshot shows the 'Set obstacles' tab in the WiFiSim application. At the top, there is a 'List of materials:' table with columns 'Material' and 'Attenuation'. The table includes rows for 'Partition wall' (3.0), 'Window' (4.0), 'Roof' (7.0), 'Viga' (10.0), 'Plants and trees' (5.0), and 'Concrete wall' (90.0). To the right of this table is a dropdown menu set to 'Partition wall'. Below these are two buttons: 'Add' and 'Remove'. A large table below these buttons has columns for 'Device A', 'Dist. between Dev.A an...', 'Obstacle', 'Device B', and 'View graphic'. The first row of this table is currently empty.

Material	Attenuation
Partition wall	3.0
Window	4.0
Roof	7.0
Viga	10.0
Plants and trees	5.0
Concrete wall	90.0

Imagen A.4. WiFiSim Panel. Set obstacles

Results

La interfaz de Results, es en la que se muestran los resultados de la simulación de WiFiSim. La segunda pestaña de WiFiSim da acceso a Results aunque estará inactiva al lanzar la aplicación. Esta se activa al realizar la simulación.

WiFiSimExtension

Cuando seleccionamos la tercera pestaña de la aplicación (Extension), se obtiene acceso a la interfaz de WiFiSimExtension que tendrá la siguiente apariencia:

WiFiSimExtension

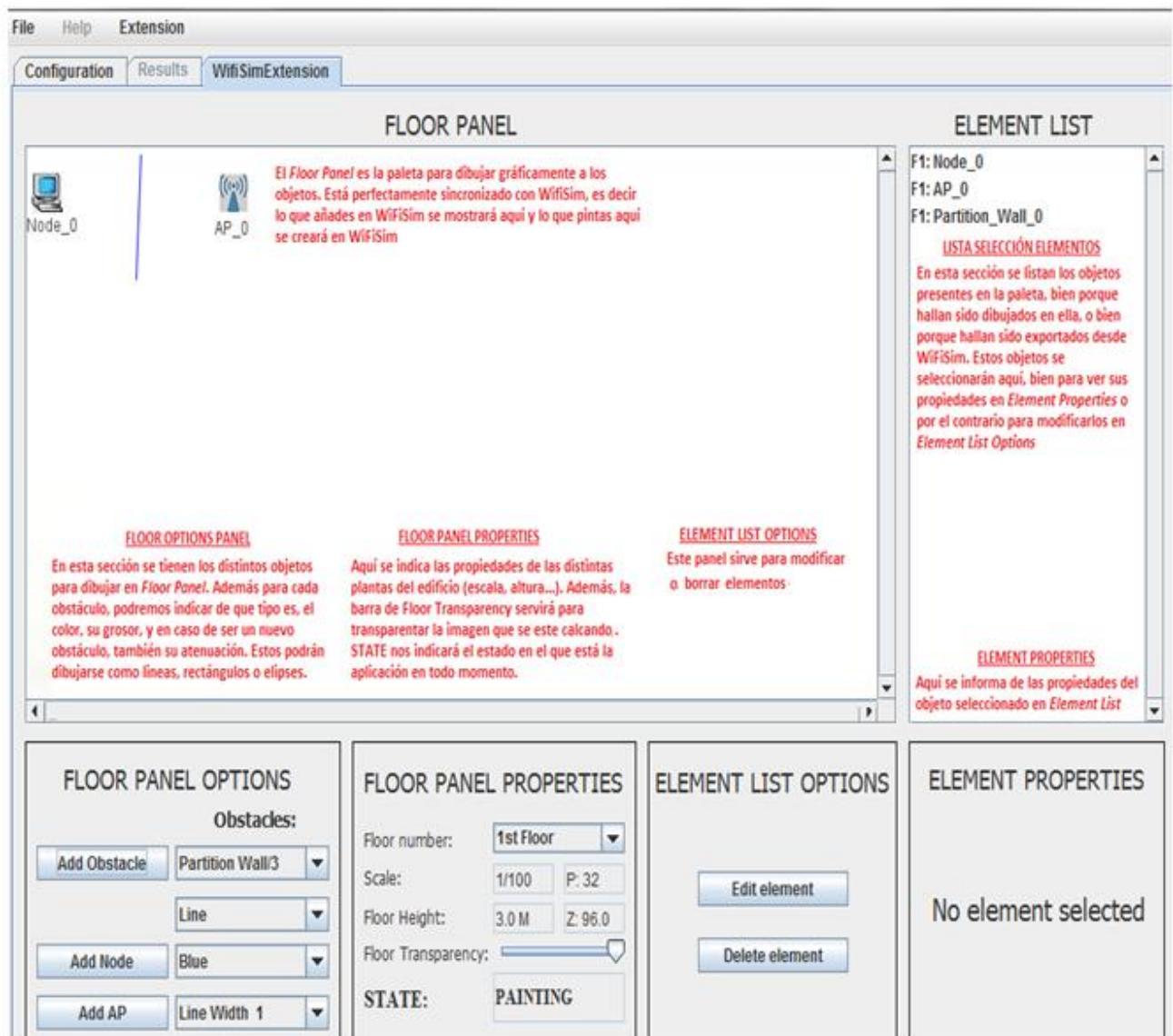


Imagen A.5. Interfaz WiFiSimExtension

Además de las opciones descritas en la imagen anterior, debido a la múltiple funcionalidad de WiFiSimExtension también podrás realizar distintas acciones que se encuentran en el menú de WiFiSimExtension.

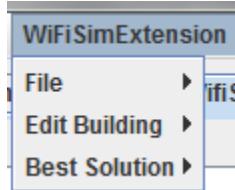


Imagen A.6.a. Menú WiFiSimExtension

Este menú solo estará disponible si nos encontramos en la interfaz WiFiSimExtension.

FILE:

New Project: Se crea un proyecto desde cero, con todos los parámetros por defecto. Se da la opción de salvar el proyecto actual al usuario.

Load Project: Se puede cargar un proyecto WiFiSimExtension guardado.

Save Project: Se puede guardar el proyecto actual como un fichero WiFiSimExtension.

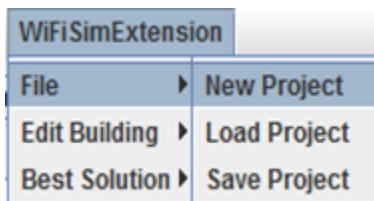


Imagen A.6.b. Sub-Menú File

EDIT BUILDING:

El sub-menú *Edit Building* tiene la siguiente apariencia. Tiene varias opciones, desde añadir nuevas plantas hasta cambiar la escala del edificio.

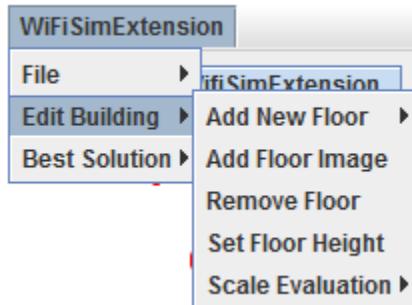


Imagen A.7. Sub-menú Edit Building

Add New Floor:

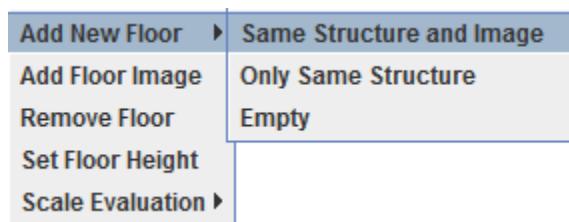


Imagen A.8. Sub-menú Add new Floor

Se puede añadir sobre el edificio una nueva planta de diferentes formas. Con la misma estructura e imagen, solo con la misma estructura, o vacío.

Add Floor Image:

Opción para cargar una imagen. Facilitará la tarea de realizar el diseño de la estructura, ya que se podrá usar la imagen de la estructura de calco. La barra *floor transparency* se utiliza para transparentar en mayor o menor medida la imagen cargada.

Remove Floor:

Eliminará la planta más alta del edificio. La primera planta no se puede eliminar.

Set Floor Height:

Establece la altura de la planta de un edificio. Por defecto esta altura será de tres metros.

Scale Evaluation:

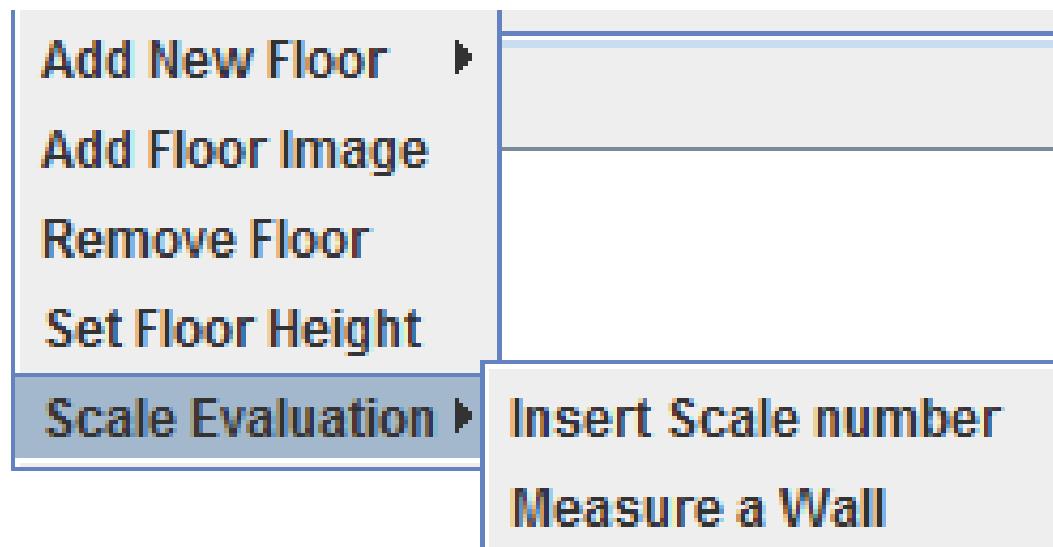


Imagen A.9. Sub-menú Add new Floor

Establecerá la escala del edificio. Por defecto será $1/100$, es decir cada centímetro en el dibujo serán 100 en realidad (un metro).

Insert Scale Number:

Se cambia la escala del edificio indicándola.

Measure a Wall:

Mediante esta opción cambiaremos la escala indicando la longitud de una pared. Tras seleccionar una pared tendremos acceso a esta opción.

BEST SOLUTION:

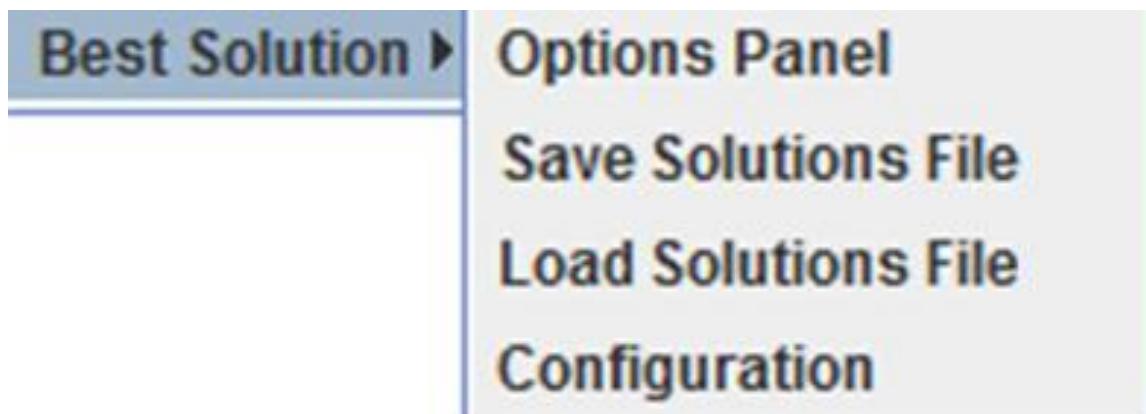


Imagen A.10. Sub-menú Best Solution

Options Panels:

Si seleccionamos esta opción, se mostrará la interfaz en el lugar de FLOOR OPTIONS PANEL que cambiará y tendrá la siguiente apariencia:

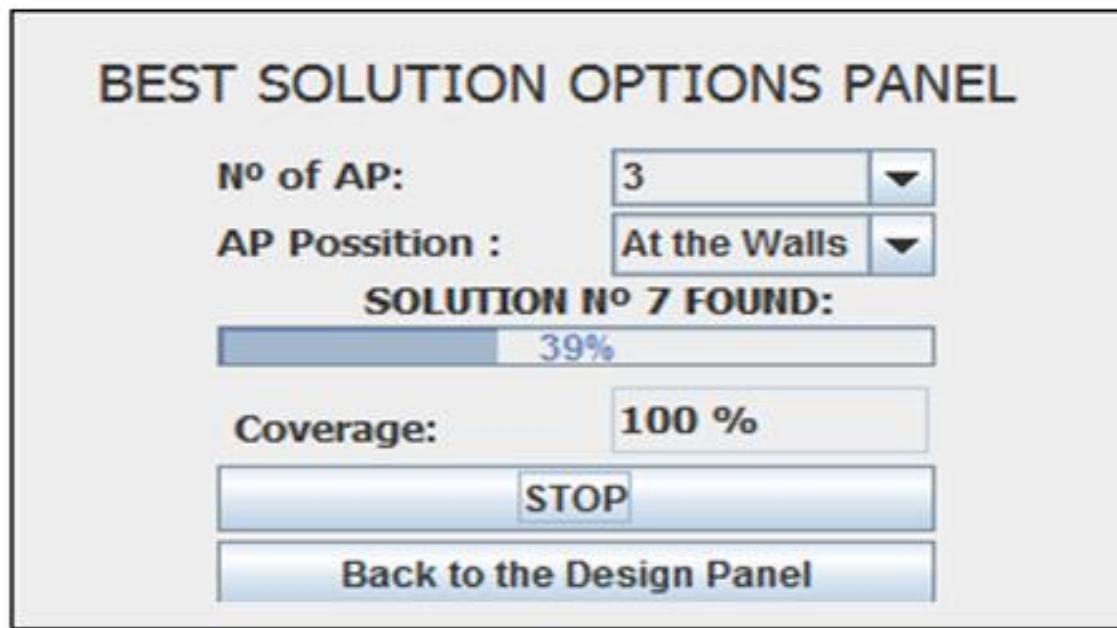


Imagen A.11. Best Solution Option Panel

Aquí indicando el número de APs que se desea poner en el edificio previamente dibujado, nos calculará mediante el algoritmo genético, la mejor posición teniendo en cuenta principalmente la cobertura del edificio y después la intensidad con la que llega a cada uno de dichos puntos. Esto se calculará teniendo en cuenta la distancia y la atenuación por obstáculos.

Esta solución podrá hacerse con distintos números de Aps e indicando si dichos Aps solo pueden ir colgados en paredes (At the Walls), los cuales irían a tres metros o bien en todas las posiciones (At Anywhere), en cuyo caso estarían colocados en el techo.

Una vez seleccionados estos parámetros, podremos ejecutar la función *Best Solution* presionando el botón *Calculate Best Solution*, y debido a que un algoritmo genético tarda un tiempo considerable en converger completamente, cada cierto tiempo que encuentre una solución mejor que la anterior se indicará por pantalla por si el usuario desea esa solución, en cuyo caso pulsando el botón *STOP* el sistema parará la búsqueda de una solución mejor y terminará quedándose con la última solución encontrada. Si por el contrario el usuario quiere encontrar una mejor, el genético sigue su curso, indicándose el porcentaje de cobertura actual y la barra de estado que indica cuando acabará la búsqueda.

Resaltar que el tiempo del genético tardará más o menos en función del tamaño del edificio.

El botón *Back to Design Panel* se utiliza para volver al panel *FLOOR PANEL OPTIONS*.

Save Solution File:

Seleccionando esta acción WiFiSimExtension permite guardar un fichero con las soluciones obtenidas en la ejecución del genético. En el fichero se guardan las diez últimas iteraciones.

Load Solution File:

Seleccionando esta acción WiFiSimExtension permite cargar un fichero de soluciones anteriormente guardado, en cuyo caso, se activará una barra desde la que podremos visualizar la solución iteración a iteración mientras el sistema la muestra gráficamente.

WiFiSimExtension

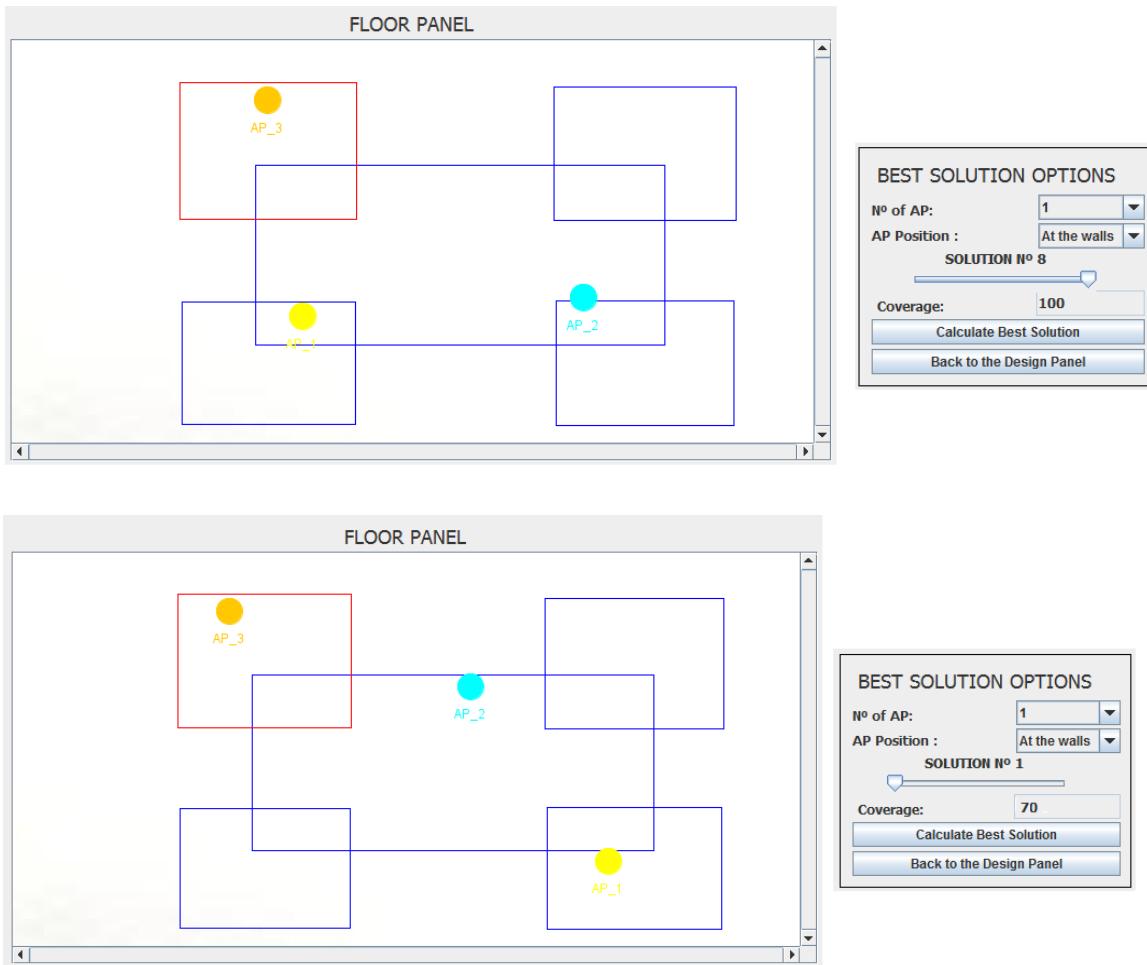


Imagen A.12. Load Solution File

Como vemos en la imagen anterior mediante la barra que se aprecia podemos ir cambiando la solución cargada iteración a iteración.

Configuration:

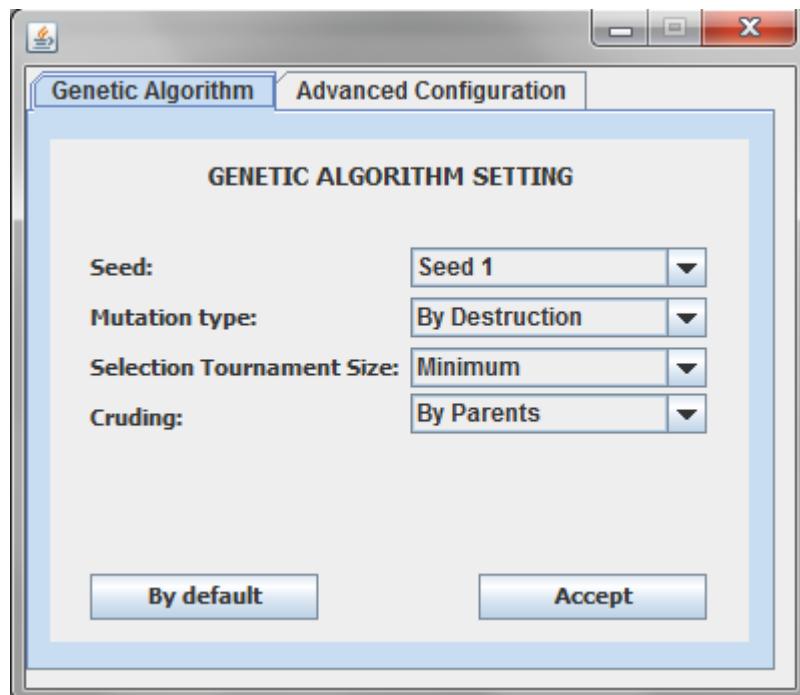


Imagen A.13. Configuración algoritmo genético

Tendrás dos ventanas de configuración:

- La primera servirá para “jugar” con los parámetros del algoritmo genético. De esta forma se puede seguir el comportamiento del algoritmo.

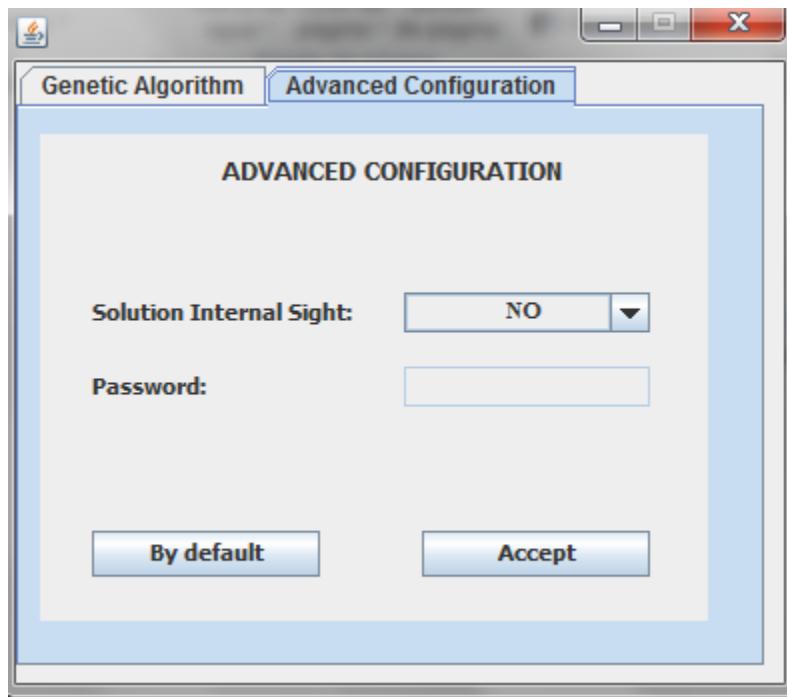


Imagen A.14. Configuración avanzada

→ La segunda pestaña es la pestaña de configuración avanzada, que se utiliza para cambiar el modo de visualizar la solución del algoritmo genético, de manera que por defecto estará en vista usuario y solo mostrará las soluciones obtenidas dibujando los Aps en las posiciones correspondientes. La solución interna muestra las celdas internas del edificio.

La contraseña para poder cambiar el modo de visualización es “wfse2012”.

FUNCIONALIDAD EXTRA A TENER EN CUENTA:

Mejor solución con nodos:

Si se da la situación en la que el usuario indique donde van a estar colocados los nodos. El proceso por el cuál el algoritmo genético calcula la mejor solución, **tendría únicamente en cuenta las posiciones en la que estén colocados estos nodos**. Con esto se consigue que el usuario pueda indicar al sistema dónde estarán los nodos, y el sistema de prioridad a dichas zonas, de manera que si el usuario desencadena la acción de calcular la mejor solución y existe algún nodo dibujado en el edificio, la acción del programa sería distinta.

Ver gradiente de cobertura de un AP:

Cuando el programa ha hallado la mejor solución de un determinado edificio, (o bien el usuario a pulsado el botón *STOP*), se podrá visualizar la zona de cobertura que tiene el AP seleccionado. Para ello habría que dejar pulsado el ratón sobre el AP deseado de la solución.

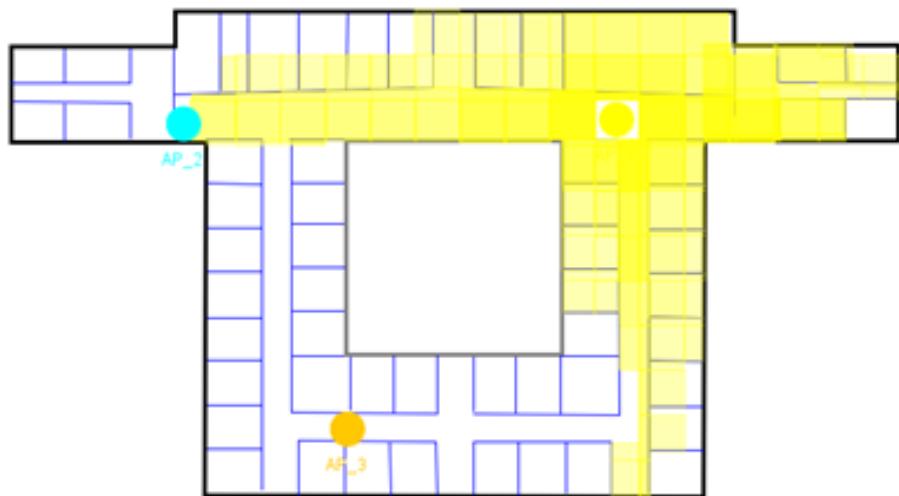


Imagen A.15. Gradiente de cobertura

Exportación de datos a WiFiSim:

La exportación de datos se realiza al cambiar de pestaña desde WiFiSimExtension a WiFiSim. En tal situación todos los datos relevantes serán exportados.

Mediante una serie de ejemplos vamos a ver el comportamiento de WiFiSimExtension a la hora de realizar la exportación a WiFiSim.

EJEMPLO 1:

- *Scale: 1/100 (32 pixel/1metro)*
- *Height: 3 metros*
- *Nº plantas: 1*
- *Nº nodes: 3*
- *Nº APs: 1*

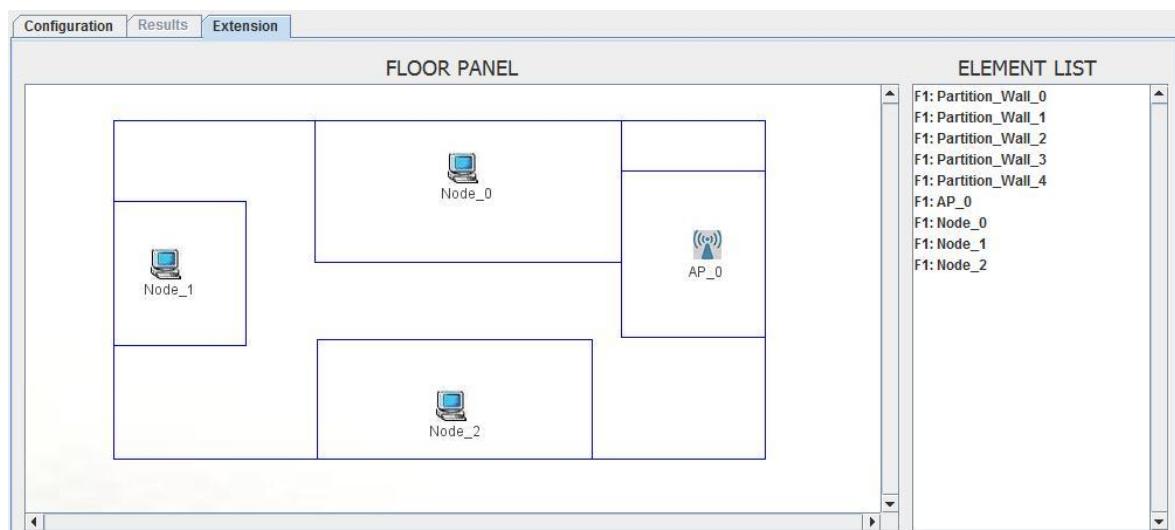


Imagen A.16. Exportación a WiFiSim. Ejemplo 1. Floor Panel

En este ejemplo tenemos una estructura simple formada por cuatro rectángulos dentro de una superficie grande. Podemos apreciar que dos líneas se solapan entre el *Node_0* y el *AP_0*. Los obstáculos tienen que ser detectados entre *APs* y *Nodes*, y entre *Nodes*. Por lo tanto, deberían detectarse los siguientes obstáculos:

- *AP_0 – Node_0*: Sólo uno, el sistema debe detectar el solapamiento y tomar el obstáculo como uno sólo: (1)
- *AP_0 – Node_2*: Dos obstáculos, uno por rectángulo implicado: (2)
- *AP_0 – Node_1*: Tres obstáculos, el primero con el rectángulo en el que se encuentra el nodo y otros dos con el rectángulo superior central en el que el sistema debe detectar de nuevo el solapamiento: (3)
- Entre los Nodos: Dos obstáculos por cada par: (2) + (2) + (2)

En total el sistema debería de detectar: (1) + (2) + (3) + (2) + (2) + (2) = 12 obstáculos.

Object	X position	Y position	Z position	Configuration
Access point 0	571	124	94	Click here
Node 0	361	57	32	Click here
Node 1	107	139	32	Click here
Node 2	351	260	32	Click here
	0	0	0	

Imagen A.17. Exportación a WiFiSim. Ejemplo 1. Set nodes de WiFiSim

En esta imagen podemos observar cómo la exportación de *Nodes* y *APs* se ha realizado correctamente. Podemos apreciar cómo los nodos tienen sus coordenadas *z* a un metro del suelo, es decir 32 pixeles, que según la escala por defecto es un metro. Y el *AP* con una coordenada *z* igual a 94, indicando que se encuentra en el techo.

WiFiSimExtension

Device A	Dist. between Dev.A an...	Obstacle	Device B	View graphic
Node 0	4.954	Ext:Partition wall	Access point 0	click here for view grap...
Node 1	4.712	Ext:Partition wall	Node 0	click here for view grap...
Node 1	2.849	Ext:Partition wall	Node 0	click here for view grap...
Node 2	1.676	Ext:Partition wall	Node 0	click here for view grap...
Node 2	3.553	Ext:Partition wall	Node 0	click here for view grap...
Node 1	12.609	Ext:Partition wall	Access point 0	click here for view grap...
Node 1	4.490	Ext:Partition wall	Access point 0	click here for view grap...
Node 1	2.723	Ext:Partition wall	Access point 0	click here for view grap...
Node 1	5.568	Ext:Partition wall	Node 2	click here for view grap...
Node 1	2.994	Ext:Partition wall	Node 2	click here for view grap...
Node 2	2.705	Ext:Partition wall	Access point 0	click here for view grap...
Node 2	5.911	Ext:Partition wall	Access point 0	click here for view grap...

Imagen A.18. Exportación a WiFiSim. Ejemplo 1. Set obstacles de WiFiSim

Finalmente obtenemos los resultados esperados, doce filas en total, con sus respectivas distancias entre *Device A* y el obstáculo de cada fila.

EJEMPLO 2:

- ➔ *Scale: 1/100 (32 pixel/1metro)*
- ➔ *Height: 3 metros*
- ➔ **Nº plantas: 3**
- ➔ *Nº nodes: 1*
- ➔ *Nº APs: 3*

En este ejemplo vamos a disponer de tres plantas, para ver el comportamiento que tiene el sistema respecto a la detección del techo.

WiFiSimExtension

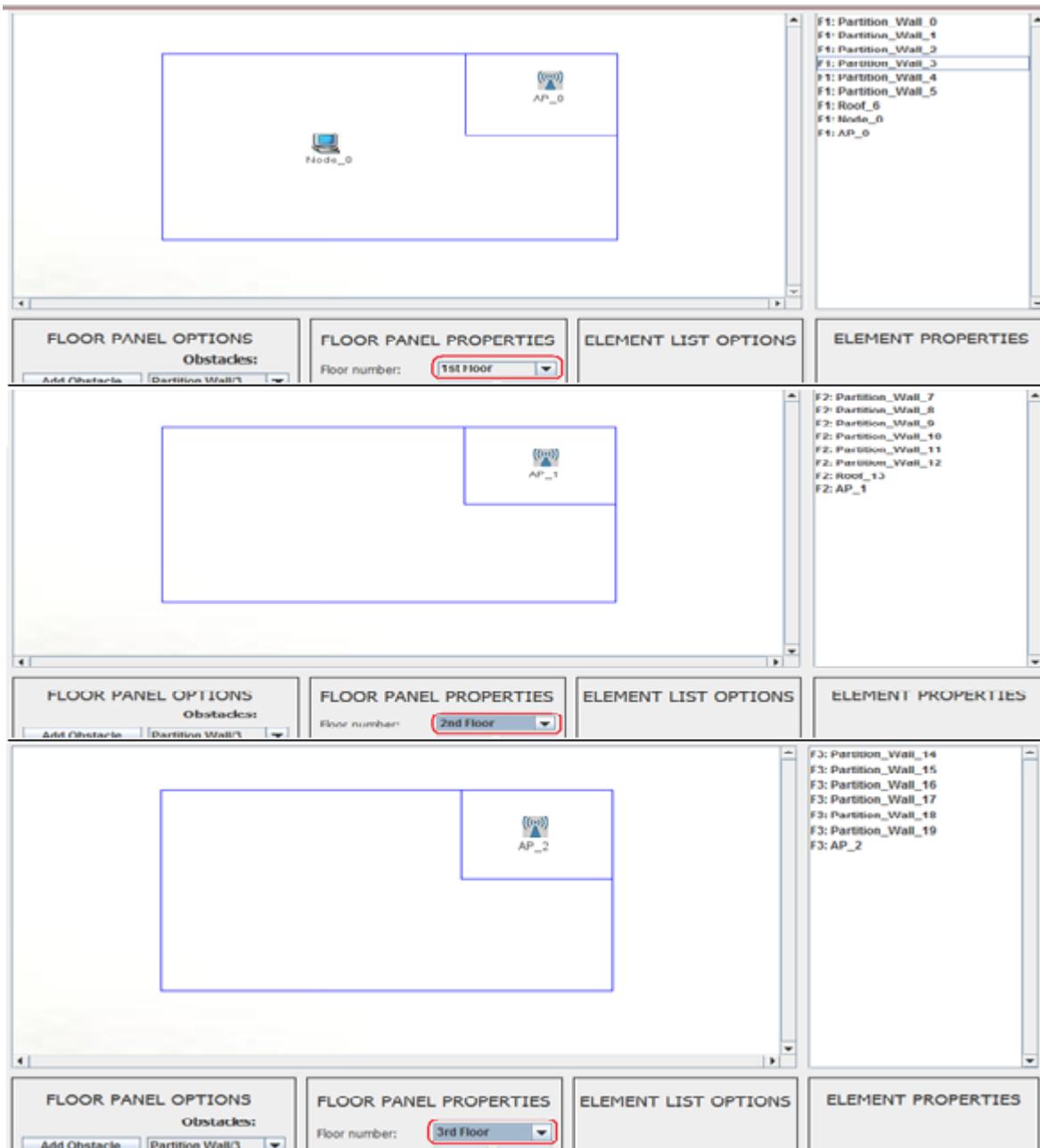


Imagen A.19. Exportación a WiFiSim. Ejemplo 2. Floor Panel

Tenemos aquí una estructura sencilla con tres plantas, con un *Node* en la primera planta y un *AP* en cada planta. Por lo tanto deberían de detectarse los siguientes obstáculos:

- *AP_0 – Node_0*: Un sólo obstáculo y sin techo, ya que se encuentran en la misma planta: (1).
- *AP_1 – Node_0*: Dos obstáculos, el del *Partition_Wall* que existe en la habitación cada planta y el del techo entre la primera y la segunda planta: (2).
- *AP_2 – Node_0*: Tres obstáculos, el del *Partition_Wall* al igual que antes, y el de los techos, de la primera y segunda planta: (3)

En total el sistema debería de detectar: (1) + (2) + (3) = 6 obstáculos.

Object	X position	Y position	Z position	Configuration
Node 0	282	152	32	Click here
Access point 0	496	73	94	Click here
Access point 1	493	80	190	Click here
Access point 2	486	83	286	Click here
	0	0	0	
	0	0	0	

Imagen A.20. Exportación a WiFiSim. Ejemplo 2. Set nodes de WiFiSim

En esta imagen podemos observar cómo la exportación de *Nodes* y *APs* se ha realizado correctamente. Podemos apreciar cómo los *APs* tienen sus coordenadas *z* en los techos de cada planta.

Device A	Dist. between Dev.A an...	Obstacle	Device B	View graphic
Node 0	4.937	Ext:Partition wall	Access point 0	click here for view graph...
Node 0	4.896	Ext:Partition wall	Access point 1	click here for view graph...
Node 0	3.742	Ext:Roof	Access point 1	click here for view graph...
Node 0	4.936	Ext:Partition wall	Access point 2	click here for view graph...
Node 0	3.759	Ext:Roof	Access point 2	click here for view graph...
Node 0	5.895	Ext:Roof	Access point 2	click here for view graph...

Imagen A.21. Exportación a WiFiSim. Ejemplo 2. Set obstacles de WiFiSim

Finalmente vemos cómo obtenemos los seis obstáculos, tres de ellos de tipo “Roof” y otros tres “Partition_Wall”. Si nos fijamos en las distancias, vemos como las dos distancias entre el “Nodo_0” respecto al techo de la primera planta están alrededor de “3.700” metros. Algo lógico teniendo en cuenta que el nodo está a un metro, el techo a tres, y éstos se encuentran en diagonal el uno respecto del otro. Mientras que la distancia con el techo de la segunda planta es alrededor de 6 metros, también razonable, ya que el techo se encuentra a 6 metros.

EJEMPLO 3:

- Scale: 1/200 (16 pixel/1metro)
- Height: 4 metros
- N^o plantas: 1
- N^o nodes: 1
- N^o APs: 1

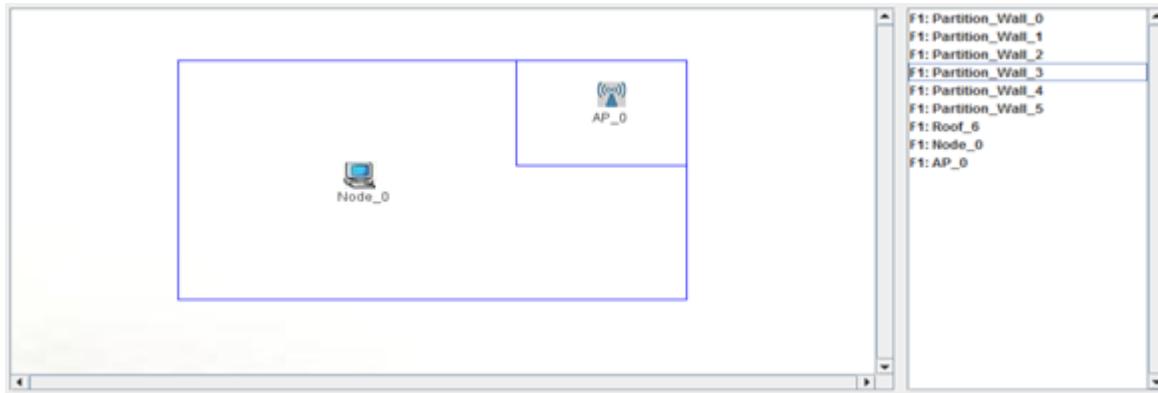


Imagen A.22. Exportación a WiFiSim. Ejemplo 3. Floor Panel

En este ejemplo vamos a usar una escala y altura del techo diferentes para ver cómo WiFiSimExtension se comporta ante esta situación.

Object	X position	Y position	Z position	Configuration
Node 0	329	105	16	Click here
Access point 0	478	90	62	Click here
	0	0	0	

Imagen A.23. Exportación a WiFiSim. Ejemplo 3. Set nodes de WiFiSim

Vemos cómo la colocación de las coordenadas z en el *Nodo 0* es 16 coincidiendo con el número de píxeles que componen un metro. También puede apreciarse como en el *Access point 0* es 62, ya que al ser la altura de la planta igual a 4 metros concuerda perfectamente con el resultado obtenido $(16*4) - 2 = 62$. Restamos 2 píxeles porque el techo siempre está colocado dos píxeles debajo del inicio de la siguiente planta, que en este caso es 64.

Importación desde WiFiSim:

Ahora vamos a ver el comportamiento de *WiFiSimExtension* a la hora de realizar la exportación pero en el otro sentido, es decir importaremos los datos desde *WiFiSim* a *WiFiSimExtension*.

Object	X position	Y position	Z position	Configuration
Access point 0	20	20	0	Click here
Node 0	685	315	0	Click here
	0	0	0	

Imagen A.24. Importación desde WiFiSim. Set nodes de WiFiSim

Se inserta un *Node* y un *AP* en *Set nodes* de *WiFiSim*, con coordenadas z igual a 0, es decir, en la primera planta. *WiFiSimExtension* las recolocará directamente a un metro y a la altura del techo respectivamente.

WiFiSimExtension

Device A	Dist. between Dev.A an...	Obstacle	Device B	View graphic
Access point 0	3	Partition wall	Node 0	click here for view grap...
Access point 0	6	Partition wall	Node 0	click here for view grap...
Access point 0	9	Partition wall	Node 0	click here for view grap...
Access point 0	12	Partition wall	Node 0	click here for view grap...
Access point 0	15	Partition wall	Node 0	click here for view grap...
Access point 0	18	Partition wall	Node 0	click here for view grap...

Imagen A.25. Importación desde WiFiSim. Set obstáculos de “WiFiSim”

Ahora se insertan seis obstáculos desde *Access point 0* a *Node 0* con diferentes distancias entre ellos.

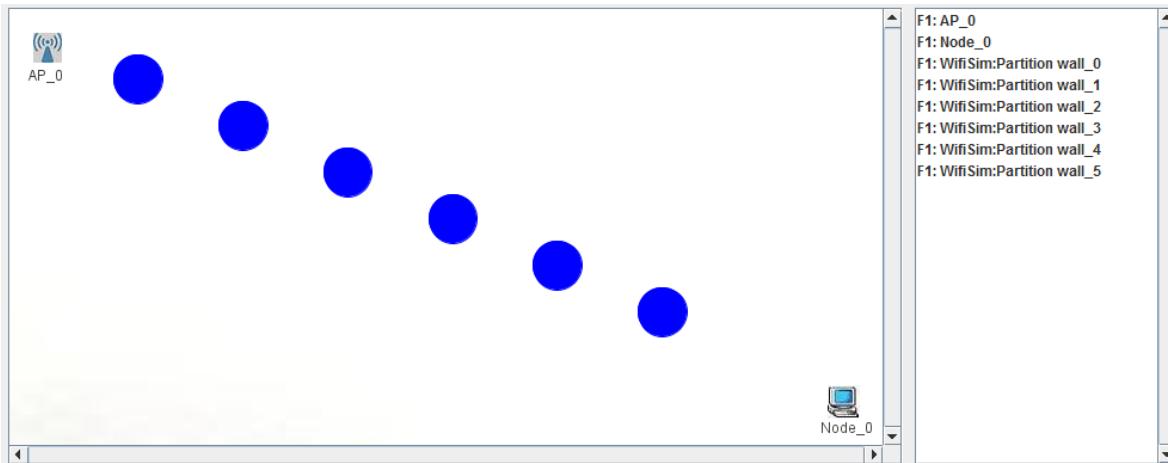


Imagen A.26. Importación desde WiFiSim. Floor Panel

Como se aprecia en la solución la importación desde *WiFiSim* se ha realizado correctamente con los dos elementos y los seis obstáculos.

B) ESPECIFICACIÓN DE CASOS DE USO

Subsistema File

Use-Case: <New Project >

Descripción

El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicita la opción “New Project”.

Pre-condiciones

No existen.

Secuencia normal

1. El actor desencadena el CU “New Project”.
2. El sistema preguntará si se desea guardar el proyecto, “Save Project”.
3. El actor responde NO.
4. El sistema vuelve a su estado inicial reseteando todos los datos del proyecto.

Secuencia alternativa

- 3b. El actor responde SI.
- 4b. Se realiza el caso de uso “Save Project”.
5. El sistema vuelve a su estado inicial reseteando todos los datos del proyecto.

Post-condiciones

El proyecto actual esta reseteado con la configuración inicial por defecto.

Use-Case: <Load Project >

Descripción

El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicita la opción “Load Project”.

Pre-condiciones

No existen.

Secuencia normal

1. El actor desencadena el CU “Load Project”.
2. El sistema abrirá la ventana de diálogo.
3. El actor selecciona el fichero donde se encuentra el proyecto.
4. El sistema restaura el proyecto y todos sus datos.

Secuencia alternativa

- 4b. El fichero no es válido o erróneo.
- 5b. El sistema lo comunica y el Caso de Uso queda sin efecto.

Post-condiciones

El proyecto actual esta restaurado con la configuración del fichero cargado.

WiFiSimExtension

Use-Case: <Save Project>

Descripción

El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicita la opción “Save Project”.

Pre-condiciones

No existen.

Secuencia normal

1. El actor desencadena el CU “Save Project”.
2. El sistema abrirá la ventana de diálogo.
3. El actor indica el directorio y nombre del fichero donde se guardará el proyecto.
4. El sistema guarda el proyecto y todos sus datos en el fichero indicado.

Secuencia alternativa

- 4b. El fichero no es válido o erróneo.
- 5b. El sistema lo comunica y el Caso de Uso queda sin efecto.

Post-condiciones

No existen.

Use-Case: <Load Solution File >

Descripción

El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicita la opción “Load Solution File”.

Pre-condiciones

No existen.

Secuencia normal

1. El actor desencadena el CU “Load Solution File”.
2. El sistema abrirá la ventana de diálogo.
3. El actor selecciona el fichero donde se encuentra el fichero de solución.
4. El sistema carga la solución y puede ser vista iteración a iteración.

Secuencia alternativa

- 4b. El fichero no es válido o erróneo.
- 5b. El sistema lo comunica y el Caso de Uso queda sin efecto.

Post-condiciones

El proyecto actual esta restaurado con la solución del fichero cargado.

Use-Case: <Save Solution File>

Descripción

El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicita la opción “Save Solution File”.

Pre-condiciones

Tiene que existir una solución.

Secuencia normal

1. El actor desencadena el CU “Save Solution File”.
2. El sistema abrirá la ventana de diálogo.
3. El actor indica el directorio y nombre del fichero donde se guardará el fichero.
4. El sistema guarda la solución en el fichero indicado.

Secuencia alternativa

- 4b. El fichero no es válido o erróneo.
- 5b. El sistema lo comunica y el Caso de Uso queda sin efecto.

Post-condiciones

No existen.

Subsistema Building Edition

Use-Case: <New Floor>

Descripción

El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicita la opción “New Floor”.

Pre-condiciones

El número de plantas del edificio tiene que ser menor que 3.

Secuencia normal

1. El actor desencadena el CU “New Floor”.
2. El sistema solicitará el modo de creación.
Si el usuario indica “Same Image and Structure”:
 3. Se realiza el caso de uso “Same Image and Structure”.

Si el usuario indica “Only Same Structure”:

3. Se realiza el caso de uso “Same Structure”.

Si el usuario indica “Same Image and Structure”:

3. Se realiza el caso de uso “Empty”.

Secuencia alternativa

No existe.

Post-condiciones

No existen.

Use-Case: <Same Image and Structure>

Descripción

El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicita la opción “Same Image and Structure”.

Pre-condiciones

No existen.

Secuencia normal

1. El actor desencadena el CU “Same Image and Structure”.
2. El sistema creará una nueva planta con la misma imagen y estructura que la base.

Secuencia alternativa

No existe.

Post-condiciones

El edificio incrementará su número de plantas.

Use-Case: <Same Structure>

Descripción

El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicita la opción “Only Same Structure”.

Pre-condiciones

No existen.

Secuencia normal

1. El actor desencadena el CU “Only Same Structure”.
2. El sistema creará una nueva planta con la misma estructura que la base.

Secuencia alternativa

No existe.

Post-condiciones

El edificio incrementará su número de plantas.

WiFiSimExtension

Use-Case: <Empty>

Descripción

El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicita la opción “Empty”.

Pre-condiciones

No existen.

Secuencia normal

1. El actor desencadena el CU “Empty”.
2. El sistema creará una nueva planta completamente vacía.

Secuencia alternativa

No existe.

Post-condiciones

El edificio incrementará su número de plantas.

Use-Case: <Remove Floor>

Descripción

El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicita la opción “Remove Floor”.

Pre-condiciones

No existen.

Secuencia normal

1. El actor desencadena el CU “Remove Floor”.
2. El sistema eliminará la planta más alta.

Secuencia alternativa

No existe.

Post-condiciones

El edificio decrementará su número de plantas.

Use-Case: <Add Floor Image>

Descripción

El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicita la opción “Add Floor Image”.

Pre-condiciones

No existen.

Secuencia normal

1. El actor desencadena el CU “Add Floor Image”.
2. El sistema abrirá el cuadro de diálogo.
3. El actor seleccionará el archivo con la imagen deseada.
4. El sistema añadirá la imagen a la planta.

Secuencia alternativa

- 4b. El fichero no es valido o erróneo.
- 5b. El sistema lo comunica y el Caso de Uso queda sin efecto.

Post-condiciones

La planta tendrá una imagen asociada.

Use-Case: <Set Floor Height>

Descripción

El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicita la opción “Set Floor Height”.

Pre-condiciones

No existen.

Secuencia normal

1. El actor desencadena el CU “Set Floor Height”.

WiFiSimExtension

-
- 2. El sistema solicitará la altura de planta.
 - 3. El actor introducirá la altura.
 - 4. El sistema cambiará la antigua altura por la nueva.

Secuencia alternativa

- 4b. El dato introducido es incorrecto.
- 5b. El sistema lo comunica y el Caso de Uso queda sin efecto.

Post-condiciones

La planta tendrá una imagen asociada.

Use-Case: <Scale Evaluation>

Descripción

El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicita la opción “Scale Evaluation”.

Pre-condiciones

No existen.

Secuencia normal

- 1. El actor desencadena el CU “Scale Evaluation”.
- 2. El sistema solicitará el modo de evaluación.

Si el usuario indica “Set Scale”:

- 3. Se realiza el Caso de Uso “Set Scale”.

Si el usuario indica “Measure a Wall”:

- 3. Se realiza el caso de uso “Measure a Wall”.

Secuencia alternativa

No existe.

Post-condiciones

No existen.

Use-Case: <Set Scale>

Descripción

El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicita la opción “Set Scale”.

Pre-condiciones

No existen.

Secuencia normal

- 1. El actor desencadena el CU “Set Scale”.
- 2. El sistema solicitará la escala del edificio.
- 3. El actor introducirá la escala.
- 4. El sistema cambiará la antigua escala por la nueva.

Secuencia alternativa

4b. El dato introducido es incorrecto.

5b. El sistema lo comunica y el Caso de Uso queda sin efecto.

Post-condiciones

El edificio tendrá la escala introducida.

Use-Case: <Measure a Wall>

Descripción

El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicita la opción “Measure a Wall”.

Pre-condiciones

Una pared de la estructura del edificio tiene que estar seleccionada.

Secuencia normal

- 1. El actor desencadena el CU “Measure a Wall”.
- 2. El sistema solicitará la distancia de la pared.
- 3. El actor introducirá la distancia.

4. El sistema cambiará la antigua escala por la nueva.

Secuencia alternativa

4b. El dato introducido es incorrecto.

5b. El sistema lo comunica y el Caso de Uso queda sin efecto.

Post-condiciones

El edificio tendrá la escala introducida.

Subsistema Floor Panel Option

Use-Case: <Add Node>

Descripción

El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicita la opción “Add Node”.

Pre-condiciones

No existen.

Secuencia normal

1. El actor desencadena el CU “Add Node”.
2. El sistema está en estado: “Painting”.
3. El actor selecciona la posición con el ratón en la pantalla de diseño.
4. El sistema añade un Nodo en la posición seleccionada.

Secuencia alternativa

No existe.

Post-condiciones

El edificio añade un elemento tipo “Node” a su lista de elementos.

Use-Case: <Add Node>

Descripción

El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicita la opción “Add AP”.

Pre-condiciones

No existen.

Secuencia normal

1. El actor desencadena el CU “Add AP”.
2. El sistema está en estado: “Painting”.
3. El actor selecciona la posición con el ratón en la pantalla de diseño.
4. El sistema añade un “AP” en la posición seleccionada.

Secuencia alternativa

No existe.

Post-condiciones

El edificio añade un elemento tipo Punto de Acceso a su lista de elementos.

Use-Case: <Add Obstacle>

Descripción

El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicita la opción “Add Obstacle”.

Pre-condiciones

No existen.

Secuencia normal

1. El actor selecciona el tipo, forma, color, ancho y desencadena el CU “Add Obstacle”.
2. El sistema está en estado: “Painting”.
3. El actor presiona el ratón en la posición de origen del obstáculo en la pantalla de diseño, mueve el ratón y lo suelta en la posición final.
4. El sistema añade un “Obstacle” en la posición seleccionada, del tipo, forma, color y ancho seleccionado.

Secuencia alternativa

No existe.

Post-condiciones

El edificio añade un elemento tipo Obstacle a su lista de elementos.

Use-Case: <New Obstacle>

Descripción

El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicita la opción “New obstacle”.

Pre-condiciones

No existen.

Secuencia normal

1. El actor selecciona el tipo, forma, color, ancho y desencadena el CU “New Obstacle”.
2. El sistema solicitará la información necesaria para crear un nuevo obstáculo.
3. El actor introduce los datos solicitados.
4. El sistema añade un nuevo tipo de Obstáculo y ejecuta el Use-Case “Add Obstacle”.

Secuencia alternativa

No existe.

Post-condiciones

El edificio añade un elemento tipo Obstacle a su lista de elementos.

Use-Case: <Edit Element>

Descripción

El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicita la opción “Edit Element”.

Pre-condiciones

Un elemento del edificio tiene que estar seleccionado.

Secuencia normal

1. El actor desencadena el CU “Edit Element”.
2. El sistema está en estado editar, y habilita los campos de edición tanto como la edición de la posición en la pantalla de diseño.
3. El actor introducirá los nuevos datos de edición y pulsará guardar cambios, o cambiará la posición con el ratón directamente en la pantalla de diseño.
4. El sistema actualizará el elemento con los nuevos datos introducidos.

Secuencia alternativa

4b. Los datos introducidos son incorrectos.

5b. El sistema lo comunica y el Caso de Uso queda sin efecto.

Post-condiciones

No existen.

WiFiSimExtension

Use-Case: <Delete Element>

Descripción

El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicita la opción “Delete Element”.

Pre-condiciones

Un elemento del edificio tiene que estar seleccionado.

Secuencia normal

1. El actor desencadena el CU “Delete Element”.
2. El sistema elimina el elemento seleccionado de la lista de elementos.

Secuencia alternativa

1. No hay ningún elemento seleccionado.
2. El sistema lo comunica y el Caso de Uso queda sin efecto

Post-condiciones

No existen.

Use-Case: <Floor Transparency>

Descripción

El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicita la opción “Floor Transparency”.

Pre-condiciones

La planta seleccionada tiene que tener imagen.

Secuencia normal

1. El actor desencadena el CU “Floor Transparency” colocando la barra de transparencia en la posición deseada.
2. El sistema dará el nivel de trasparencia introducido a la imagen de la planta.

Secuencia alternativa

No existe.

Post-condiciones

No existen.

Use-Case: <Change Floor>

Descripción

El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicita la opción “Change Floor”.

Pre-condiciones

No existe.

Secuencia normal

1. El actor desencadena el CU “Change Floor” indicando el número de planta deseada.
2. El sistema visualiza la planta seleccionada.

Secuencia alternativa

No existe.

Post-condiciones

No existen.

WiFiSimExtension

Use-Case: <Change APpositions>

Descripción

El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicita la opción “Change APpositions”.

Pre-condiciones

La planta seleccionada tiene que tener imagen.

Secuencia normal

1. El actor desencadena el CU “Change APpositions” indicando donde quiere que se puedan colocar los puntos de acceso en el edificio (en las paredes o en cualquier lugar).
2. El sistema guardará la selección.

Secuencia alternativa

No existe.

Post-condiciones

No existen.

Subsistema WiFiSimExtension functionality

Use-Case: <Export to WiFiSim>

Descripción

El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicita la opción “Export to WiFiSim”.

Pre-condiciones

No existen

Secuencia normal

1. El actor desencadena el CU “Export to WiFiSim”.
- Si la interfaz actual de trabajo es WiFiSimExtension:
 2. El sistema importará los datos desde WiFiSimExtension a la interfaz de WiFiSim.

Secuencia alternativa

No existe.

Post-condiciones

Ambas interfaces tendrán la misma lista de elementos.

Use-Case: <Import from WiFiSim>

Descripción

El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicita la opción “Import from WiFiSim”.

Pre-condiciones

No existen

Secuencia normal

1. El actor desencadena el CU “Import from WiFiSim”.
- Si la interfaz actual de trabajo es WiFiSim:
 2. El sistema importará los datos desde WiFiSim a la interfaz de WiFiSimExtension.

Secuencia alternativa

No existe.

Post-condiciones

Ambas interfaces tendrán la misma lista de elementos.

Use-Case: <Best Solution>

Descripción

El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicita la opción “Best Solution”.

Pre-condiciones

No existen

Secuencia normal

1. El actor desencadena el CU “Best Solution”, indicando el número de “AP” y “APposition”
2. El sistema devuelve la mejor solución encontrada.

Secuencia alternativa

No existe.

Post-condiciones

No existe.

C) TEORÍA DE ALGORITMOS GENÉTICOS

1. Introducción.

Los algoritmos genéticos son algoritmos de optimización, búsqueda y aprendizaje inspirados en los procesos de la evolución natural y la evolución genética.

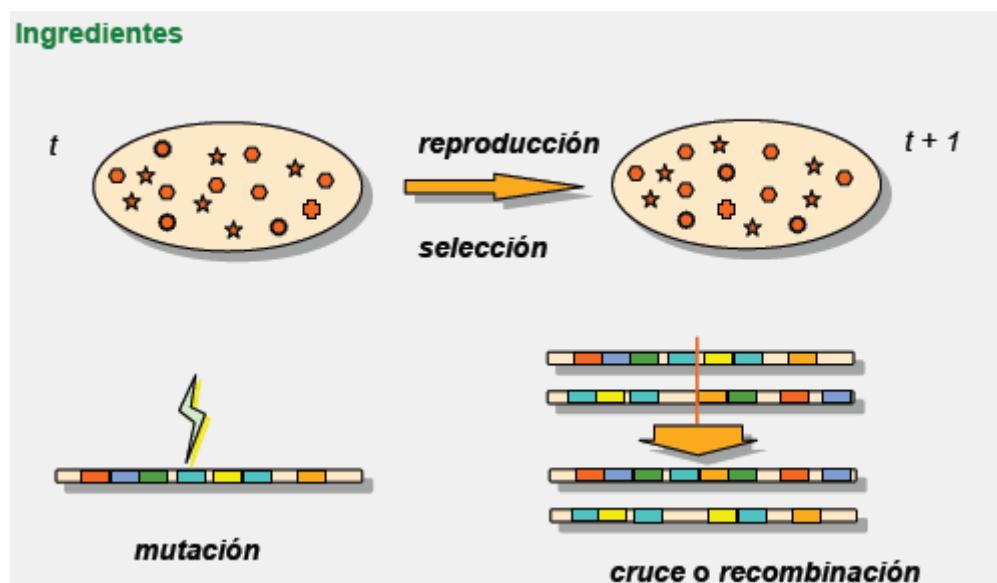


Imagen C.1. Ingredientes



Imagen C.2. Ciclo de evolución

Procedimiento Algoritmo Genético

```
Inicio (1)
    t = 0;
    Inicializar P(t);
    Evaluar P(t);
    Mientras (no se cumpla la condición de parada) hacer
        Inicio(2)
            t = t + 1
            Seleccionar P(t) desde P(t-1)
            Recombinar P(t)
            Mutar P(t)
            Evaluar P(t)
        Final(2)
    Final(1)
```

Imagen C.3. Procedimiento Algoritmo Genético

1.1 Algoritmo genético estacional vs estacionario

Modelo Generacional: Durante cada iteración se crea una población completa con nuevos individuos. La nueva población remplaza directamente a la antigua.

Modelo Estacionario: Durante cada iteración se escogen dos padres de la población (diferentes mecanismos de muestreo) y se les aplican los operadores genéticos. El/Los descendiente/s remplazan a uno/dos cromosoma/s de la población inicial.

1.2 Proceso de construcción

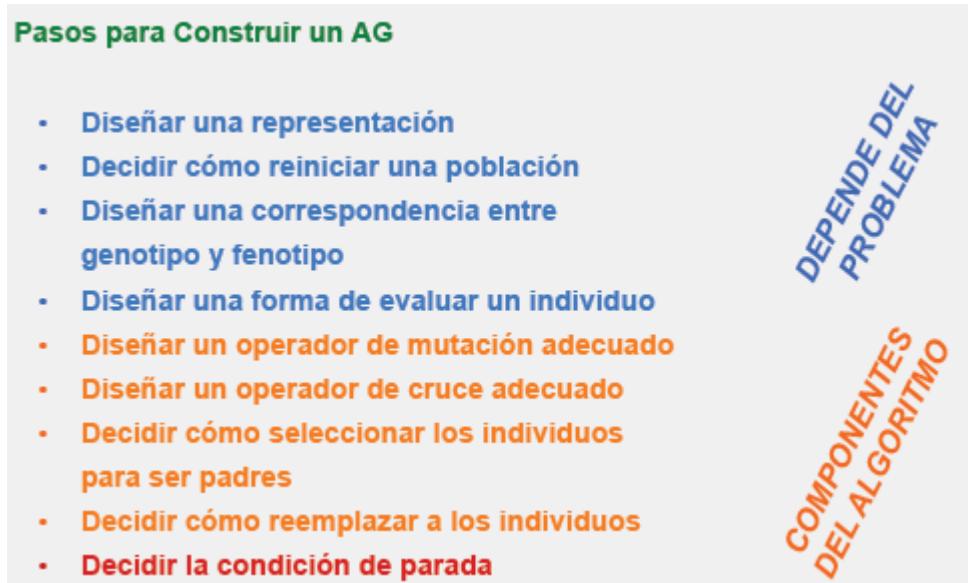


Imagen C.4. Procedimiento de construcción

1.3.1 Diseñar una representación:

- Debemos disponer de un mecanismo para codificar un individuo como un genotipo (conjunto de genes).
- Existen muchas maneras de hacer esto y se ha de elegir la más relevante para el problema en cuestión.
- Una vez elegida una representación, hemos de tener en mente como los genotipos (codificación) serán evaluados y qué operadores genéticos hay que utilizar.
- La representación de un individuo puede hacerse mediante una codificación discreta, como por ejemplo binaria.

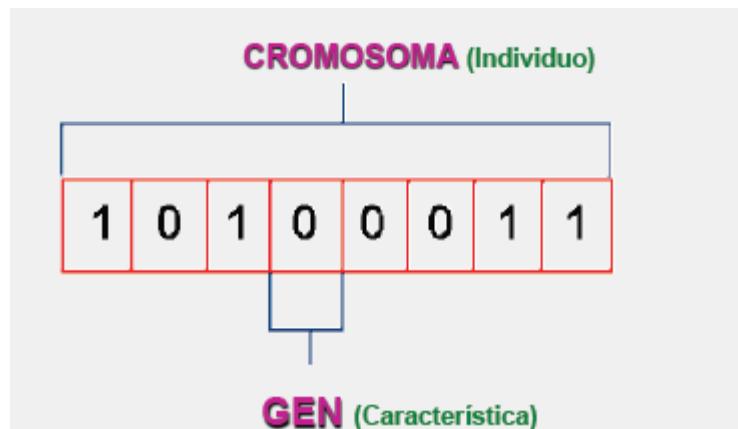


Imagen C.5. Cromosoma-Gen

- Tendremos que obtener el fenotipo:

Genotipo: Mecanismo para codificar un individuo, ejemplo: usar 8 bits.

Fenotipo: Lo que se puede codificar en un conjunto de genes, ejemplo: un entero, un real, etc...

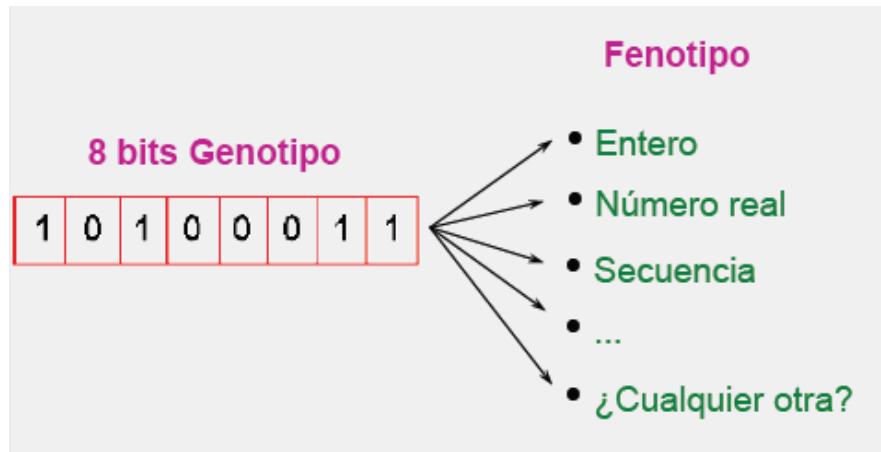


Imagen C.6. Genotipo-Fenotipo

Por ejemplo:

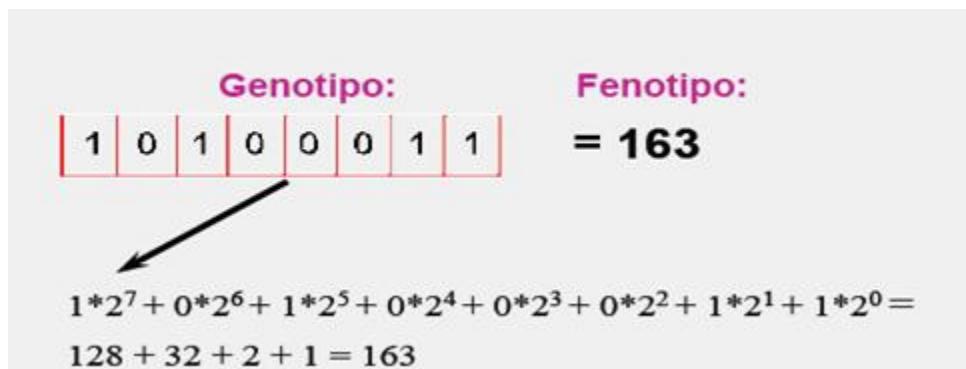


Imagen C.7. Genotipo-Fenotipo II

1.3.2. Decidir cómo reiniciar (inicializar) una población. A ser posible. Elegida de modo uniforme sobre el espacio de búsqueda.

Ejemplos:

Representación Binaria: 0 o 1 con probabilidad 0.5.

Representación Real: uniforme sobre un intervalo dado.

1.3.3. Diseñar una correspondencia entre genotipo y fenotipo

- En ocasiones el genotipo puede ser un conjunto de parámetros para algún algoritmo, el cuál trabaja sobre los datos de un problema para obtener un fenotipo.

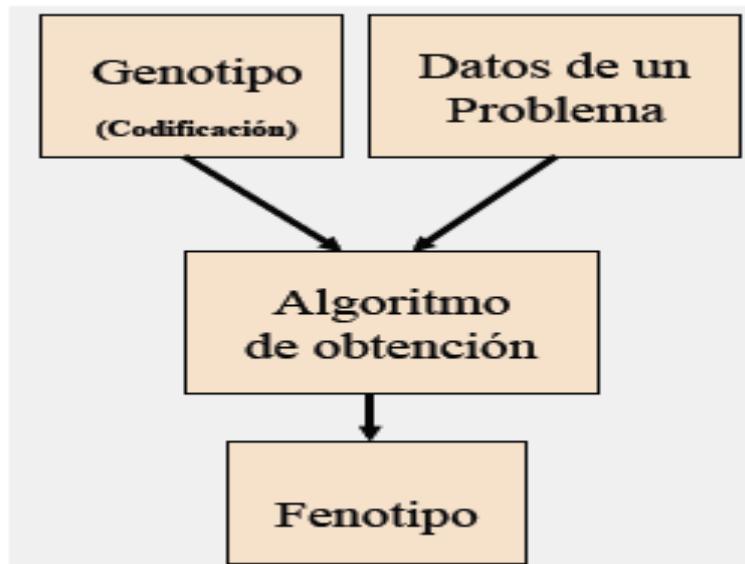


Imagen C.8. Genotipo-Fenotipo III

1.3.4 Diseñar una forma de evaluar un individuo

- Este es el paso más costoso (computacionalmente) para una aplicación real.
- Cuando hay restricciones, éstas se pueden introducir en el costo comopenalización.
- Si hay múltiples objetivos se busca una solución de compromiso.
- Este proceso se conoce como Fitness (cuanto mejor es una solución).

1.3.5 Decidir cómo seleccionar los individuos para ser padres

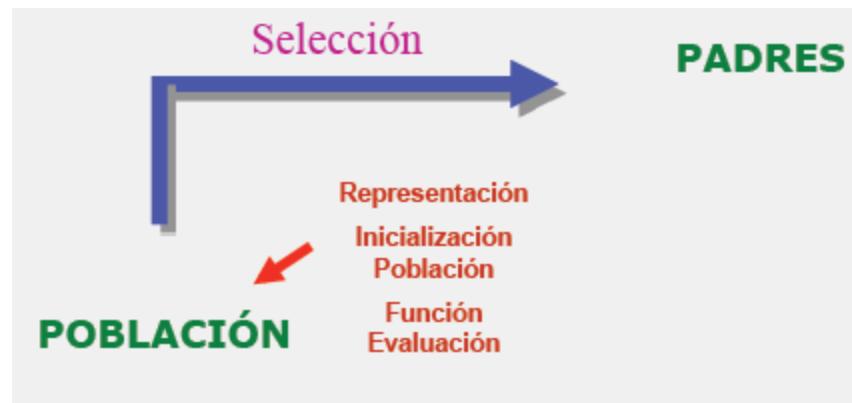


Imagen C.9. Población

- Debemos garantizar que los mejores individuos tienen una mayor posibilidad de ser padres (reproducirse) frente a los individuos menos buenos.
- Debemos ser cuidadosos para dar una oportunidad de reproducirse a los individuos menos buenos. Éstos pueden incluir material genético útil en el proceso de reproducción.
- Esta idea nos define la presión selectiva que determina en qué grado la reproducción está dirigida por los mejores individuos.
- Selección por Torneo: Método para regular la presión selectiva:
 - Escoger aleatoriamente k individuos, (con reemplazo, generalmente) de una población.
 - Seleccionar el mejor de ellos (o los L mejores de ellos).
 - Repitiendo el proceso, se genera la población. (k se denomina tamaño del torneo. A mayor k , mayor presión selectiva y viceversa; usualmente $k=2,3,\dots$).

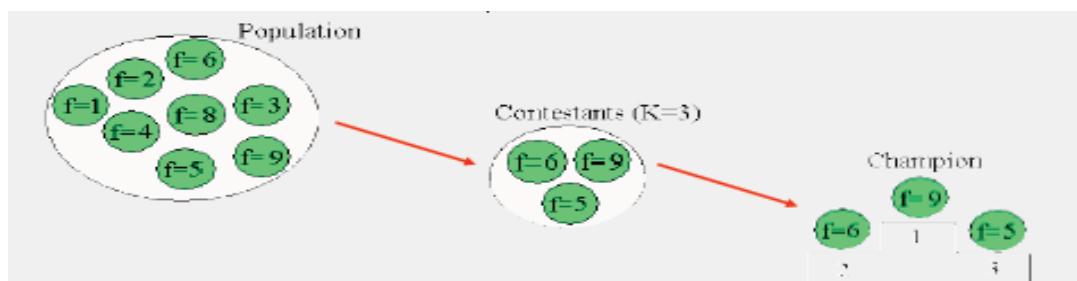


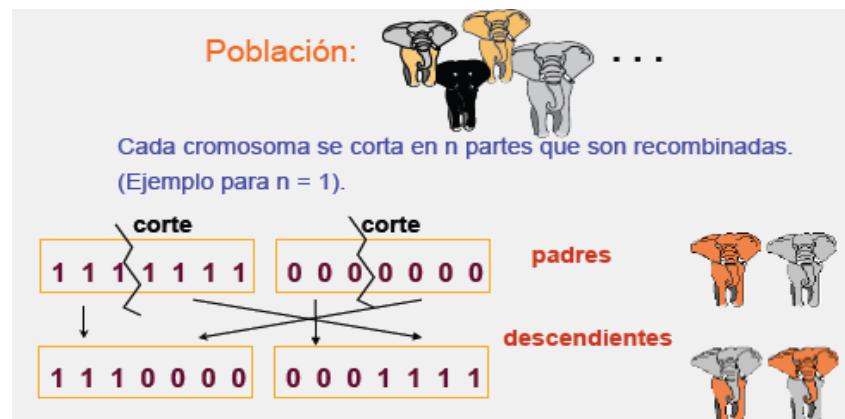
Imagen C.9. Población II

Diseñar un operador de cruce adecuado



Imagen C.10. Población III

- Podríamos tener uno o más operadores de cruce para nuestra representación.
- Algunos aspectos importantes a tener en cuenta son:
 - Los hijos deberían heredar algunas características de cada parente. Si éste no es el caso, entonces estamos ante un operador de mutación.
 - Se debe diseñar de acuerdo a la representación.
 - La recombinación debe producir cromosomas válidos.
 - Se utiliza con una probabilidad alta de actuación sobre cada pareja de padres a cruzar (P_c entre 0.6 y 0.9), si no actúa, los padres son los descendientes del proceso de recombinación de la pareja.
- Ejemplo:



Diseñar un operador de mutación adecuado.

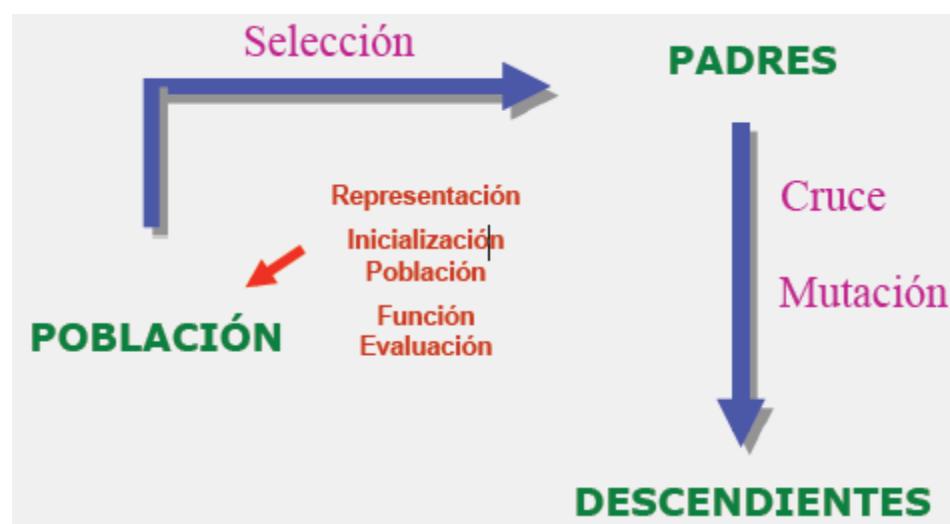


Imagen C.12. Población V

- Podemos tener uno o más operadores de mutación para nuestra representación.
- Algunos aspectos importantes a tener en cuenta son:
 - Debe permitir alcanzar cualquier parte del espacio de búsqueda.
 - El tamaño de la mutación debe ser controlado.
 - Debe producir cromosomas válidos.
 - Se aplica con una probabilidad muy baja de actuación sobre cada descendiente obtenido tras aplicar el operador de cruce (incluidos los descendientes que coinciden con los padres porque el operador de cruce no actúa).

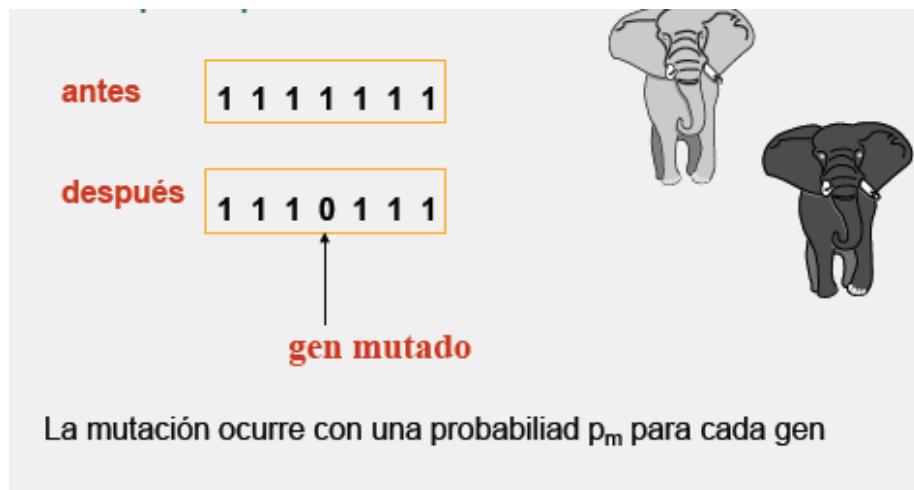


Imagen C.13. Población VI

1.3.6. Decidir como remplazar individuos.



Imagen C.14. Población VII

- La presión selectiva se ve también afectada por la forma en que los cromosomas de la población son reemplazados por los nuevos descendientes.
- Podemos utilizar métodos de reemplazamiento aleatorios, o determinísticos.
- Podemos decidir no reemplazar al mejor cromosoma de la población: Elitismo (que consiste en mantener a los mejores).

Algunas Estrategias de Reemplazo para AGs Estacionarios:

- Reemplazar el peor de la población (RW). Genera Alta presión selectiva.
- Torneo Restringido (RTS): Reemplaza el más parecido de entre w (w=3). Objetivo: Mantener una cierta diversidad.
- Peor entre semejantes (WAMS): Busca equilibrio entre diversidad y presión selectiva.
- Algoritmo de Crowding Determinístico (DC): Reemplaza a su padre más parecido. Mantiene diversidad.

1.3.7. Decidir la condición de parada

- Ideal: Cuando se alcanza el óptimo.
- Recursos limitados de CPU:
 - ➔ Fijar el máximo número de evaluaciones
- Límite sobre la paciencia del usuario: Después de algunas iteraciones sin mejora.

1.4 Utilización de AGs

- Nunca se deben sacar conclusiones de una única ejecución utilizar medidas estadísticas (medias, medianas, ...) con un número suficiente de ejecuciones independientes.
- No se debe ajustar/comprobar la actuación de un algoritmo sobre ejemplos simples si se desea trabajar con casos reales.
- Existe una comentario genérico en el uso de los Algoritmos no determinísticos:

“Se puede obtener lo que se desea en una experimentación de acuerdo a la dificultad de los casos utilizados.”(Se encuentran propuestas en las que basta encontrar un caso adecuado para un algoritmo para afirmar que es muy bueno, pero esta afirmación no significa que sea extensible a otros casos: es el error en el que incurren algunos autores).

D) DIAGRAMA DE GANTT DETALLADO

Lista de tareas			
Nombre	Recursos		
	Fecha de inicio	Fecha de fin	
Inicio	4/07/12	31/07/12	
Requisitos	4/07/12	26/07/12	
Reunión 1	4/07/12	5/07/12	
Reunión 2	11/07/12	12/07/12	
Reunión 3	18/07/12	19/07/12	
Reunión 4	25/07/12	26/07/12	
Análisis	11/07/12	31/07/12	
Estudio herramientas	11/07/12	17/07/12	
Estudio Wifisim	18/07/12	28/07/12	
WFSExtension	30/07/12	31/07/12	
Diseño	17/07/12	28/07/12	
Interfaz	17/07/12	28/07/12	
Implementación	17/07/12	28/07/12	
Gestión Obstáculos	17/07/12	25/07/12	
Gestión Nodes y AP	25/07/12	28/07/12	
Test	20/07/12	31/07/12	
Test iniciales	20/07/12	31/07/12	
Elaboración	1/08/12	11/09/12	
Requisitos	10/08/12	4/09/12	
Reunión 5	10/08/12	11/08/12	
Reunión 6	15/08/12	16/08/12	
Reunión 7	23/08/12	24/08/12	

WiFiSimExtension

	Reunión 8	3/09/12	4/09/12
	Análisis	1/08/12	11/09/12
	Integración en WiFiSim	1/08/12	4/08/12
	Grid	16/08/12	18/08/12
	Escala, edificio, plantas	29/08/12	11/09/12
	Diseño	8/08/12	5/09/12
	Integración WiFiSim	8/08/12	18/08/12
	Grid	16/08/12	5/09/12
	Implementación	10/08/12	8/09/12
	Integración WiFiSim	10/08/12	31/08/12
	Grid, celdas	16/08/12	8/09/12
	Test	27/08/12	11/09/12
	Test específicos	27/08/12	11/09/12
	Construcción	12/09/12	30/10/12
	Requisitos	12/09/12	25/09/12
	Reunión 9	12/09/12	13/09/12
	Reunión 10	19/09/12	20/09/12
	Reunión 11	24/09/12	25/09/12
	Análisis	3/10/12	30/10/12
	Reunión 12	3/10/12	4/10/12
	Reunión 13	29/10/12	30/10/12
	Diseño	24/09/12	30/10/12
	Interfaz, escala, edificio, exportación,etc.	24/09/12	6/10/12
	Genético	8/10/12	30/10/12
	Implementación	20/09/12	27/10/12
	Genético	20/09/12	10/10/12
	Cargar, Guardar proyectos	11/10/12	12/10/12
	Detección obstáculos	15/10/12	20/10/12
	Funcionalidad interfaz	22/10/12	25/10/12
	Interfaz conf. genético	25/10/12	27/10/12
	Test	10/10/12	30/10/12
	Test específicos	10/10/12	20/10/12
	Test genético	15/10/12	30/10/12
	Transición	1/11/12	1/12/12
	Requisitos	1/11/12	7/11/12
	Últimos requisitos	1/11/12	7/11/12
	Análisis	1/11/12	16/11/12
	Genético	1/11/12	10/11/12
	Últimas reuniones	15/11/12	16/11/12
	Diseño	1/11/12	13/11/12
	Interfaz Final	1/11/12	13/11/12
	Implementación	1/11/12	20/11/12
	Últimos detalles y requisitos	1/11/12	20/11/12
	Test	1/11/12	1/12/12
	Test Generales	1/11/12	1/12/12

REFERENCIAS

- [1] César Serrano López, Simulador del Protocolo 802.11, Proyecto fin de carrera de la Universidad de Huelva.
- [2] WiTuners, <http://www.wituners.com/>
- [3] AirgMagnet, <http://es.flukenetworks.com/enterprise-network/wireless-network/AirMagnet-Survey>
- [4] Ruckus Wireless Zone Planner, www.ruckuswireless.com/products/zoneplanner
- [5] Ekahau Survey Professional, <http://www.ekahau.com/products/ekahau-site-survey/overview.html>
- [6] WindowBuilder, <http://www.eclipse.org/windowbuilder/>
- [7] JAVA2D, <http://www.apl.jhu.edu/~hall/java/Java2D-Tutorial.html>
- [8] UML, http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado
- [9] DAO, http://es.wikipedia.org/wiki/Data_Access_Object
- [10] RUP, http://es.wikipedia.org/wiki/Proceso_Unificado_de_Rational