Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERÍAS INDUSTRIALES

Grado en Ingeniería de Organización Industrial

Desarrollo de una aplicación web para la gestión de las actividades docentes

Autor:

Sánchez de Prada, Rocío

Tutor:

Alarcia Estévez, Esperanza

Dpto. Matemática Aplicada

Valladolid, Junio 2018.

Resumen

En este proyecto se busca resolver el problema de asignación horaria y distribución de aulas, para la docencia reglada de cada curso académico existente en la Escuela de Ingenierías Industriales de la Universidad de Valladolid. Para ello se realiza un análisis y síntesis de la situación actual y se desarrolla un algoritmo matemático de tipo Enfriamiento Simulado basado en el comportamiento microestructural de los metales durante el proceso de recocido. La adaptación del problema al algoritmo y su consiguiente programación, donde el método principal de trabajo ha sido la mejora continuada hasta alcanzar soluciones aceptables, se ha llevado a cabo en lenguaje Java y mediante el entorno de trabajo Eclipse. El programa obtenido da resultados válidos y cuyo análisis y validación se presentan también en este proyecto.
Así mismo se establecen las bases para la creación de una aplicación web que integre y permita la utilización de este programa a un mayor número de usuarios y se proponen también diversas mejoras que se podrán llevar a cabo en futuros proyectos.

Palabras clave: Optimización / Algoritmo / Recocido Simulado / Generación de horarios

Abstract

This project seeks to solve subjects timing allocation and optimizing classrooms distribution of Valladolid Industrial Engineering School University. In order to do this, an analysis and synthesis of the current situation is carried out and a Simulated Annealing mathematical algorithm is developed and programmed which is based on the microstructural behavior of the metals during annealing process. The main method of work has been continuous improvement to reach acceptable solutions and algorithm adaptation has been carried out in Java language through Eclipse platform. An analysis and validation of the obtained program results are also presented in this project.
It also proposes several program improvements and establishes the bases for creating a web application that integrates and allows more people to use this program that may be carried out in future projects.

Keywords: Optimization / Algorithm / Simulated Annealing / Time Assignment

Índice de contenidos

Índice de ecuaciones

Índice de figuras

Índice de tablas

Índice de imágenes

Índice de gráficas

CAPÍTULO I. INTRODUCCIÓN

1. Introducción.

En todas las instituciones educativas es necesario realizar una asignación horaria que se adapte a las necesidades existentes y a los recursos disponibles. Este proceso puede llegar a resultar realmente complejo en aquellas instituciones con gran volumen de estudiantes, actividades docentes o gran oferta educativa, ya que no solo hay que garantizar el cumplimiento de las horas lectivas establecidas si no que también se debe asegurar una distribución eficiente de recursos.
Se presenta por tanto un problema de optimización de recursos limitados sujeto a una serie de restricciones definidas para el cual no siempre será posible encontrar una solución exacta, siendo necesaria en muchas ocasiones la aceptación de una solución aproximada.

A día de hoy se pueden encontrar diversas herramientas para llevar a cabo estas tareas, tanto empresas que ofrecen este tipo de servicios, de pago elevándose el precio con la complejidad del problema, como aplicaciones que permiten introducir tus propios parámetros y restricciones y obtener la distribución horaria, sin embargo, este tipo de aplicaciones no suelen ser válidas cuando el problema presenta gran número de restricciones.
Algunas instituciones optan por estas opciones, aunque la gran mayoría suelen llevar a cabo ellos mismos estas laboriosas tareas.

2. Contexto y Justificación.

La Escuela de Ingenierías Industriales de la Universidad de Valladolid realiza los horarios internamente y de manera manual, este proceso requiere gran cantidad de tiempo (suele llevar aproximadamente un mes) y experiencia de la persona implicada ya que tiene que ser conocedora de todos los recursos y restricciones existentes.
Actualmente existen varias sedes en las que se imparten los siguientes Grados, en función de la sede:

- Sede Francisco Mendizábal, donde se imparten los Grados de:
    - Ingeniería Eléctrica.
    - Ingeniería en Diseño Industrial y Desarrollo de Producto.
    - Ingeniería en Electrónica Industrial y Automática.
    - Ingeniería en Tecnologías Industriales.

- Sede Paseo del Cauce, donde se imparten los Grados de:
    - Ingeniería Química.
    - Ingeniería en Organización Industrial.
    - Ingeniería Mecánica.

Está previsto unificar la docencia de todos los grados en una única nueva sede, en el edificio IndUVA, en Prado de la Magdalena, que abrirá sus puertas el próximo curso 2018/2019.

En el presente proyecto se pretende desarrollar una herramienta, en concreto una aplicación web, capaz de realizar una distribución horaria de forma automática de todos los grados impartidos por la Escuela de Ingenierías Industriales en el nuevo aulario IndUVA, optimizando el uso de aulas. Esto supondría agilizar y simplificar todo el proceso de asignación horaria que es necesario llevar a cabo todos los años.

3. Objetivos.

Se definen a continuación los principales objetivos del proyecto:

- Identificar, definir y plantear el problema a resolver.

- Definir y programar un método matemático capaz de obtener una asignación horaria que sea realizable, aplicable y que se ejecute en un tiempo razonable.

- Desarrollar una aplicación web que permita la revisión, modificación y verificación de la información existente y que presente los resultados, asignaciones horarias, derivadas de la ejecución del método matemático. Esta debe ser intuitiva, manejable y funcional.

4. Estructura del documento.

En el capítulo dos se hace una introducción histórica del tipo de técnicas matemáticas que se van a emplear en el desarrollo del proyecto, cómo y por qué surgieron, cuál ha sido su evolución a lo largo de los años y como se emplean en la actualidad. También se desarrollan conceptualmente los algoritmos que se emplean en la realización del presente proyecto.
En el capítulo tres se lleva a cabo la presentación y síntesis del problema a resolver, definiendo toda la información existente.
En el capítulo cuatro se presenta la adaptación del problema a los algoritmos, así como la adecuación de estos y el análisis de los resultados obtenidos.
En el capítulo cinco se proponen nuevas líneas de trabajo derivadas del proyecto y posibles mejoras del mismo.
En el capítulo seis se presentan las conclusiones derivadas de la realización del proyecto y se proponen línea de trabajo para posibles mejoras del mismo.
Y por último la bibliografía.

CAPÍTULO II. INVESTIGACIÓN OPERATIVA Y MÉTODOS DE OPTIMIZACIÓN.

1. Introducción histórica.

Como bien es sabido el desarrollo científico está íntimamente relacionado con los conflictos bélicos, ya que los avances que se producen durante estos periodos influyen notablemente en la evolución posterior de la sociedad debido a las diversas aplicaciones industriales de estos.

Durante la Segunda Guerra Mundial la Luftwaffe, fuerza aérea alemana, atacó continuamente Inglaterra en la conocida Batalla de Inglaterra con el fin de abrir paso a una invasión terrestre que no llegaría gracias en gran parte al trabajo de Robert Alexander Watson-Watt.
Esta fue la primera batalla aérea de la historia que se desarrolló entre julio y octubre de 1940 sobre el Canal de la Mancha y el sureste del territorio británico, y cuyo desenlace sin duda alguna marcó el rumbo de la historia.

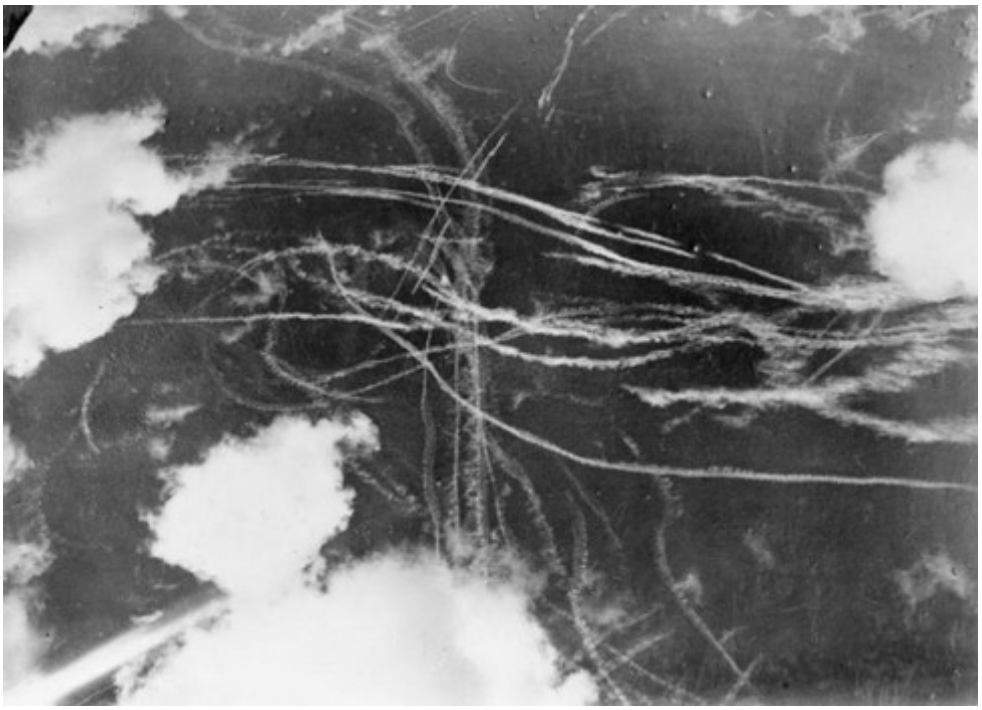


Imagen 1. Estelas de aviones tras una batalla aérea.

A fin de frenar el avance de las tropas alemanas sobre Francia, Inglaterra envió una ingente cantidad de tropas para ayudar al aliado francés, el cual terminó cayendo en manos alemanas. Con la flota aérea inglesa, la RAF (Royal Air Force), menguada tras los desesperados intentos de evitar el avance de las tropas alemanas y proteger al ejército inglés acorralado en Dunkerque, finalmente fue posible llevar a cabo la evacuación de estas justo antes de su caída.

A pesar de estar Francia invadida y controlada por el ejército alemán Inglaterra seguía volando sobre el canal de Mancha y atacando bases militares alemanas situadas en la costa francesa, a lo cual el ejército alemán respondió.

Los alemanes desarrollan entonces un plan, conocido con el nombre de operación León Marino, para continuar con su expansión y hacerse con el territorio británico. Su primer objetivo eran las bases aéreas inglesas cercanas a la costa para hacerse con el control marítimo del canal de la Mancha, para seguidamente proceder a la invasión territorial, para ello eran fundamentales los ataques aéreos. Con gran efectividad los alemanes hicieron retroceder a los ingleses y se hicieron con el control aéreo del canal de la Mancha con lo que ahora los combates aéreos se empezaron a desarrollar sobre el sureste de Inglaterra, su objetivo era hacerse con el control del espacio aéreo para proceder al desembarco.

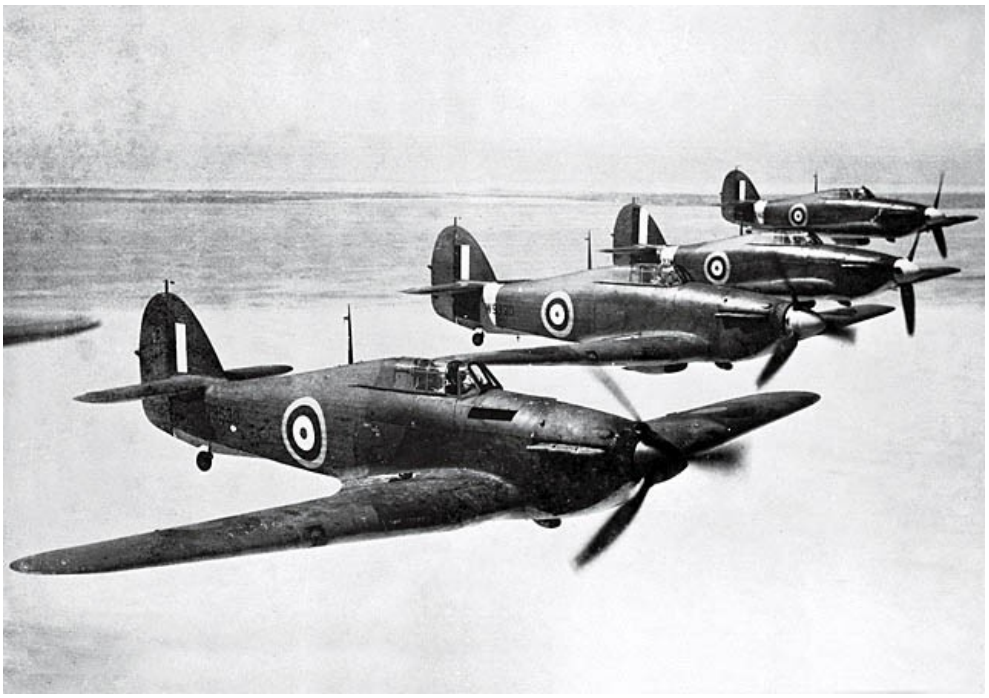


Imagen 2. Formación de Hawkers Hurricane en vuelo.

Con la RAF fuertemente menguada por los constantes ataques alemanes librándose batalles de 300 aviones a 12, la probabilidad de éxito de los ingleses era prácticamente nula, a pesar de disponer de pilotos muy cualificados y de muy buena maquinaria (cazas Hurricane y Spitfire) ya que los recursos eran cada vez menores. Otro de los principales problemas de la RAF, fue que cuando se producían los ataques alemanes y se daba respuesta a los mismos ya era demasiado tarde, alcanzando y derribando muy pocos objetivos, estos ataques al estilo guerras relámpago, táctica de ataque alemana, hacían mucho daño sobre las fuerzas británicas.

Fue entonces cuando el equipo de investigación liderado por Robert Watson Watt desarrolló un sistema de detección y posicionamiento de aviones basado en el radar, el cual empleaba ondas de radio que se reflejaban en los aviones y se recibían de nuevo en la base de rádar capaz de fijar la señal mediante un tubo de rayos catódicos, pudiendo determinar a través de la observación de estos tubos la distancia a la que se encontraba el enemigo en función del tiempo que tardaba en ser devuelta la señal. A su vez los aviones de la RAF llevaban transmisores de señales que les servía para identificarles y guiarles hacia el enemigo. Ante esta situación de forma muy rápida y coordinada y según unas posiciones estratégicas definidas por los científicos ingleses crearon una red de estaciones de radar, la Chain Home, orientables y capaces de fijar la señal de cuyas capacidades los alemanes no eran conscientes y que por lo tanto no era uno de sus principales objetivos. Estas estaciones permitían detectar aviones enemigos a 190 km de distancia, por lo que los ingleses consiguieron la capacidad de reacción necesaria para realizar contraataques altamente efectivos y las bajas enemigas comenzaron a ser mucho más numerosas a pesar de su mayoría numérica.

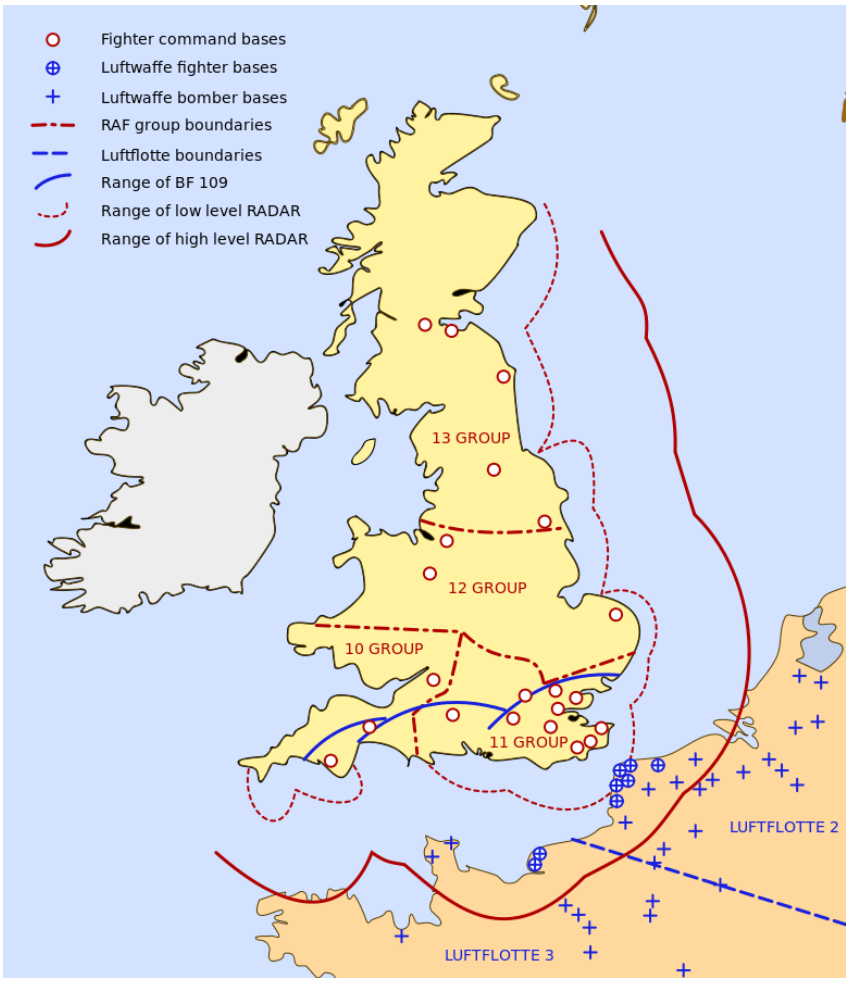


Imagen 3. Rango de alcance de los radáres de la Chain Home.

Un cambio estratégico en el plan de ataque alemán, por el cual se establecía como objetivo principal la población civil y no las bases aéreas, permitió rearmarse al Mando de Caza. Este hecho junto con los descubrimientos anteriores, fruto de mucho trabajo de investigación, fueron decisivos para conseguir la victoria inglesa y la retirada de las tropas alemanas sobre este territorio, lo cual permitiría llevar a cabo posteriormente el desembarco de Normandía.

Es a partir de entonces cuando se empieza a hablar formalmente de Investigación Operativa y se establece como ciencia.

> La Investigación Operativa es la aplicación de métodos científicos a problemas complejos que surgen en la dirección de sistemas de gran tamaño compuestos por hombres, máquinas, materiales, dinero, etc. en la industria, los negocios, el gobierno o la defensa. Su característica distintiva es la de

> construir modelos de los sistemas que incorporen factores tales como oportunidades y riesgos, con los que comparar los resultados de decisiones alternativas, estrategias o métodos de control. El objetivo es ayudar a la dirección de estos sistemas a tomar decisiones y ejecutar acciones de forma científica. [5]

Aunque muchos expertos consideran los comienzos de esta ciencia en las propuestas de Arquímides para la defensa de la ciudad de Siracusa, gracias a las cuales resistió los ataques romanos durante 8 meses durante la II Guerra Púnica, no es hasta después de la Segunda Guerra Mundial cuando se define.

Posteriormente esta forma de trabajo y toma de decisiones se traslada al mundo empresarial y a la población civil, sobre todo durante los años 50 y 60, y va ligado al desarrollo computacional, ya que es la principal herramienta empleada en el análisis, simulación y obtención de resultados. Algunos ejemplos de grandes empresas que han hecho uso de la Investigación Operativa para mejorar sus servicios y rendimientos son:

Empresa	**Aplicación**	**Año**	**Ahorros Anuales**
Electrobras / CEPAL Brasil	Asignación óptima de recursos hidráulicos y térmicos en el sistema nacional de generación de energía.	1986	$43 millones
IBM	Integración de una red nacional de inventario de recambios para mejorar el apoyo al servicio.	1990	$20 millones + $250 millones en menor inventario
American Airlines	Diseño de un sistema de estructura de precios, sobreventas (exceso de reservas) y coordinación de vuelos para mejorar los beneficios.	1992	$500 millones más de ingresos
Procter & Gamble	Rediseño del sistema de producción y distribución norteamericano para reducir costos y mejorar la rapidez de llegada al mercado.	1997	$200 millones

Tabla 1. Aplicaciones reales de la Investigación Operativa.[1]

Dentro de la Investigación Operativa una de las áreas principales es la optimización. Esta rama busca dar herramientas capaces de seleccionar la mejor

alternativa frente a otras también evaluadas en base a unos criterios definidos. Consiste básicamente en encontrar el máximo o mínimo, es decir, la mejor solución posible de una función definida, función objetivo. El hallar esta solución pasa por encontrar los valores de una serie de variables (cumpliendo las restricciones existentes) que harán precisamente que la función sea máxima o mínima.

Se distinguen por lo tanto varios elementos en un problema de optimización:

- **Función objetivo**, que va a representar un sistema determinado el cual se pretende optimizar. Es la medida cuantitativa del modelo representado y a través de la cual se evalúa el mismo.

- **Variables**, son el conjunto de decisiones a tomar dentro del sistema que harán que este tenga un mejor o peor funcionamiento según el valor que tomen, ya que afectan al valor de la función objetivo.

- **Restricciones**, son las relaciones existentes entre las variables que intervienen en el problema y que a su vez lo limitan ya que deben cumplirse para alcanzar el punto de funcionamiento óptimo.

Uno de los primeros problemas de optimización que se planteó fue el Método Simplex. Desarrollado por George Dantzig en 1947, quién desarrolló su trabajo en las Fuerzas Aéreas americanas como consejero matemático y posteriormente continuó como investigador en la Corporación RAND. Se trata de un método matemático para la resolución de problemas de Programación Lineal, cuyos fundamentos matemáticos los estableció John Von Neumann en 1928 en su trabajo “Teoría de Juegos”. Este método es uno de los más eficientes para optimizar las ganancias y los costes y fue programado por primera vez en 1952 para un problema de 42 ecuaciones y 71 variables, su resolución llevó aproximadamente 18 horas.

Una de las aplicaciones iniciales del Método Simplex fue el puente aéreo de Berlín, con el que se estableció un plan de abastecimiento aéreo para la ciudad durante su bloqueo en 1948, durante la guerra fría, consiguiendo así que los costes fuesen mínimos.

Imagen 4. George Bernard Dantzig.

La continuación de los estudios en la mejora de la toma de decisiones por parte de grupos de científicos e investigadores, subvencionados por distintas instituciones tanto públicas como privadas, como la revolución tecnológica, que ha permitido la resolución de cálculos complejos, han sido dos de los factores fundamentales en la evolución, desarrollo y aplicación de la Investigación Operativa.

2. Métodos de optimización.

Existe una gran cantidad de técnicas de optimización en el campo de la Investigación Operativa que hacen referencia a los métodos usados para encontrar la mejor solución posible de un determinado problema.

Es conveniente realizar una clasificación y definición general de las mismas, donde uno de los factores más relevantes a la hora de encontrar una solución va a ser la complejidad del problema que se pretende resolver, el cual puede pertenecer a cualquier ámbito; industrial, científico, militar, etc. Aunque se pueden encontrar muchas divisiones de estos métodos dependiendo del autor o la rama de aplicación, en el presente proyecto se van a distinguir tres tipos:

- Métodos exactos.

- Métodos heurísticos o clásicos.

- Métodos metaheurísticos.

En primer lugar se definen las **técnicas exactas**, estas van a garantizar siempre la obtención de la solución óptima existente. Estos métodos se aplican a problemas relativamente sencillos y son capaces de rastrear todas las alternativas posibles, evaluarlas y dar la mejor solución para el problema planteado.
Otro aspecto a tener en cuenta a la hora de llevar a cabo la resolución mediante un método exacto, si es que este existe, ya que no suele ser lo habitual en este campo, es la eficiencia del método. Hay que ser consciente de que los medios disponibles de cualquier institución que pretenda llevar a cabo la aplicación de este tipo de procedimientos, sea del tipo que sea, van a ser siempre limitados y este es uno de los factores principales en el desarrollo y utilización de los mismos; los tiempos de computación, la capacidad, la memoria, los propios equipos, la tecnología disponible, etc. hacen que muchas veces este tipo de técnicas no se puedan utilizar o se consideren inadecuadas por lo que se descarta su aplicación.

En contraposición a los métodos exactos que como su propio nombre indica se caracterizan por encontrar la solución exacta al problema planteado, se encuentran los **métodos heurísticos**.
La palabra heurística significa método para aumentar el conocimiento, de esta forma aparecen estos métodos que no garantizan la obtención de una solución exacta u óptima pero sí aceptable.
Surgen como respuesta a la necesidad de resolver problemas de mayor complejidad, en los que los tiempos de resolución deben ser razonables y para los cuales es suficiente con la obtención de una solución aproximada, buena o posible y no necesariamente óptima. Una definición de estas técnicas sería:

> Un procedimiento para el que se tiene un alto grado de confianza en que encuentra soluciones de alta calidad con un coste computacional razonable, aunque no se garantice su optimalidad o su factibilidad, e incluso, en algunos casos, no se llegue a establecer lo cerca que se está de dicha situación. [6]

Dentro de las técnicas heurísticas se puede hacer una distinción entre técnicas generales y específicas. Las generales van a ser más sencillas y robustas y podrán aplicarse indistintamente a varios problemas diferentes, mientras que las específicas son las adaptadas a un problema concreto y cuya eficiencia suele ser mucho mayor.

Otra clasificación más interesante y no excluyente, dentro de las muchas que se pueden hacer, es la realizada en función del procedimiento de obtención de la solución, definiéndose por lo tanto los siguientes tipos de métodos [8]:

- **Generación de soluciones aleatorias**. Se generan muchas soluciones posibles de forma aleatoria, se evalúan y se escoge la mejor de ellas.

- **Descomposición del problema**. Consiste en coger un problema complejo y subdividirlo en subproblemas que sean más sencillos de resolver siguiendo un criterio.

- **Métodos inductivos**. Funcionan planteando el mismo problema para un espacio menor, de forma que el método obtenido sirva para problemas más grandes o bien usando un problema que esté relacionado a nivel matemático.

- **Métodos que reducen el espacio de la solución**. Se basa en reducir notablemente el número de soluciones consideradas sin que esto afecte demasiado a la calidad de la solución final. Se puede llevar a cabo eliminando directamente todas aquellas alternativas que no cumplan por ejemplo las restricciones del problema o añadiendo restricciones extra que faciliten un descarte inicial.

- **Métodos de aproximación**. Se basa en el manejo de la representación matemática del problema existente y defiende que en ocasiones dos aproximaciones al problema pueden permitir obtener una mejor solución que una sola aproximación. Para ello, agregar parámetros, modificar o aproximar la función objetivo, emplear distribuciones de probabilidad para aproximar valores de variables o cambiar la naturaleza de las restricciones, hacerlas lineales aunque no lo sean, ignorar restricciones serían alguno de los ejemplos.

- **Métodos constructivos**. Utilizan los datos del problema para ir construyendo poco a poco una solución final. El método Greedy sería un claro ejemplo de esta técnica, el cual a cada paso elige la opción que mayor mejora proporciona a la solución final.

- **Métodos de mejora local**, también conocidos como Hill-climbing. Estos algoritmos parten de una población o solución factible inicial y a continuación la comparan con el vecindario de forma que si alguna de las soluciones es mejor que la anterior la toman como nueva solución posible hasta que ya no haya más mejoras.

Sobre las técnicas heurísticas generales se pueden usar o aplicar diferentes recursos tanto de programación como técnicas inteligentes que van a hacer que evolucionen, es aquí donde se introducen los **métodos metaheurísticos**. El propio sufijo meta significa "más allá", por lo tanto es fácil intuir que estos métodos surgen a partir de la evolución de las técnicas heurísticas.

La palabra metaheurística fue introducida por primera vez por F. Glover en su artículo *Future paths for integer programming and links to artificial intelligence* [9], donde da este nombre a la Búsqueda tabú, definiéndola como una heurística evolucionada.
Otra definición muy conocida es la dada por los profesores I. H. Osman y J. P. Kelly:

> Los procedimientos metaheurísticos son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son ni efectivos ni eficientes. Los metaheurísticos proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y la mecánica estadística. [10]

Se puede decir que son estrategias o procedimientos que sirven para crear algoritmos heurísticos mejorados y que permiten estudiar problemas de gran complejidad con relativa sencillez aportando buenas soluciones. Entre las cualidades deseadas que deben presentar los algoritmos metaheurísticos destacan la eficiencia, simplicidad, robustez y precisión.
La clasificación de las técnicas heurísticas también es aplicable a las metaheurísticas, de esta forma se pueden encontrar también metaheurísticas de relajación, constructivas, de descomposición, de búsqueda, evolutivas…Aunque igualmente se pueden realizar otras muchas clasificaciones.

Las características principales que presentan estas técnicas son [11]:

- Son ciegas y aproximativas. Estos métodos no garantizan una solución óptima y por ello va a ser necesario definir un criterio de

parada, ya sea un tiempo de computación, número de iteraciones, un valor numérico, etc.

- Cada solución no tiene por qué ser mejor que la anterior.

- Son relativamente sencillos y generales. Se pueden a aplicar a casi todos los problemas de optimización combinatoria haciendo un modelo adecuado del problema (representación del espacio de búsqueda, definición del punto de partida y del método de exploración), aunque la complejidad de los mismos afectará a la eficacia del programa.

- El criterio de selección depende de cada instante del proceso, aunque dos iteraciones tengan la misma solución la siguiente no tiene por qué ser igual.

Este proyecto se va a centrar en técnicas evolutivas, en concreto en los algoritmos Genéticos, y de búsqueda, en especial en el algoritmo de Recocido Simulado.

Las **metaheurísticas evolutivas** se caracterizan porque las posibles soluciones del espacio de búsqueda interaccionan entre si y evolucionan. En cada iteración va a haber una población que estará formada por un conjunto de individuos, todos ellos serán soluciones posibles, e interaccionarán entre sí para dar lugar a una nueva población mejor que la anterior. Estos métodos suelen ser más lentos y necesitar gran capacidad de memoria, ya que en vez de usar un único individuo emplean una población para escapar de óptimos locales.

Por otro lado, las **metaheurísticas de búsqueda** recorren el espacio de soluciones según diferentes estrategias y van transformando a cada iteración la solución inicial. Se pueden distinguir entre:

- Algoritmos de búsqueda local, son aquellos que buscan las soluciones en el entorno de la solución de partida, pero generalmente este tipo de algoritmos suelen quedar estancados en óptimos locales.

- Algoritmos de búsqueda global, surgen para superar la limitación de los anteriores, tratan de escapar de los óptimos locales de baja calidad; para ello se plantean estrategias como aceptar empeoramientos de la solución que permitan escapar de esos puntos locales (búsquedas no monótonas), reiniciar el algoritmo

desde otro punto de comienzo (arranque múltiple) o modificar el entorno (entorno variable).

Los métodos de búsqueda son muy extensos y se han desarrollado mucho en los últimos años, al igual que las técnicas evolutivas; ambos son quizás las dos técnicas más importantes dentro de este enorme y complejo campo de estudio.

2.1. Algoritmos genéticos.

Surgieron en los años 1970 desarrollados por John Henry Holland. Se trata de una técnica evolutiva de búsqueda ciega y de inspiración biológica basada en el concepto de selección natural definido por el naturalista y geólogo Charles Darwin en su obra “El origen de la especies” (1859).

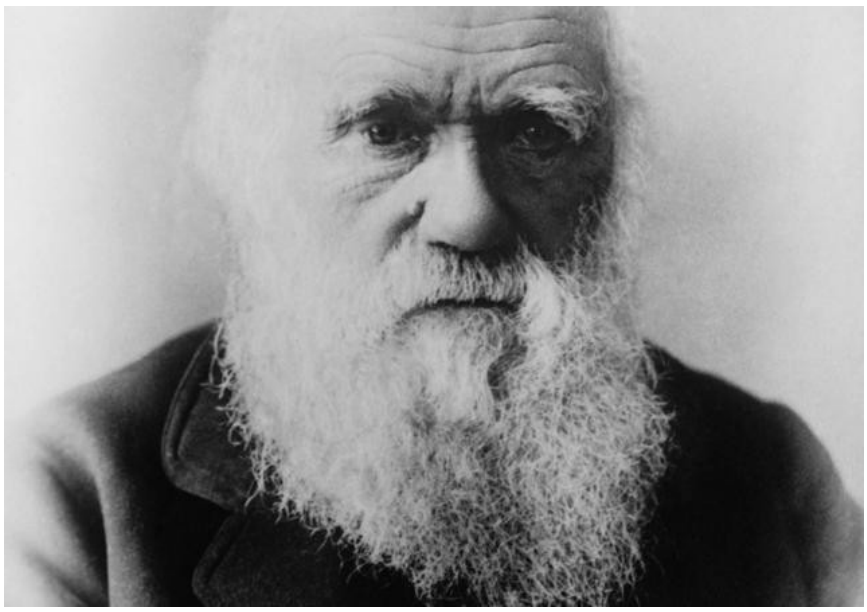


Imagen 5. Charles Robert Darwin.

Esta teoría, basada en el método inductivo, defiende que los seres vivos evolucionan ya que el entorno ofrece recursos limitados y va variando, por lo tantos estos deben luchar por los recursos existentes lo que implica la supervivencia de los más fuertes y esto deriva en una adaptación de estos al medio. Si esta adaptación al medio no tiene lugar, la especie terminará extinguiéndose, y si esto se produce dará lugar a organismos más fuertes.

Los saltos evolutivos los presenta como fruto de las mutaciones aleatorias que tienen lugar, aunque no siempre suponen un beneficio para la especie.

El funcionamiento del algoritmo va a ser análogo: se plantea un espacio de soluciones posibles, serían los individuos, y se evalúan en base a una función objetivo definida, conocida como función de adaptación, de esta forma las soluciones mejores obtendrán una mayor puntuación, son los individuos más fuertes de la población. A continuación se realizan los cruces entre individuos, soluciones, teniendo mayor probabilidad de cruce los de mayor puntuación, dándose lugar a un nuevo espacio de soluciones, una nueva población, que se volverán a evaluar de la misma manera. De esta forma se va consiguiendo "una adaptación" de la solución hacia una solución final mejorada o válida.
Se van a presentar a continuación una serie de conceptos necesarios para el planteamiento del algoritmo.
Los cromosomas contienen la información que caracteriza los seres vivos. Un cromosoma está formado por genes, que son precisamente las características definitorias, el conjunto de estos genes es el genotipo; la manifestación visible de esta información es el fenotipo, es decir, son las propiedades observables del individuo. En la naturaleza dos individuos pueden tener el mismo genotipo pero no igual fenotipo, y puede haber características del genotipo que no se manifiesten, en un algoritmo genético que esto se dé o no depende del tipo de codificación.
En el algoritmo genético, un individuo estará formado por uno o varios cromosomas los cuales contienen un conjunto de genes, genotipo. Estos genes serán la representación de la información, generalmente una cadena de caracteres. A lo largo del algoritmo se llevará a cabo una evaluación de los individuos mediante una función llamada de evaluación o adaptación, la cual representa el entorno. Esta función se encargará de valorar la información contenida en los genes y puntuarla para definir los mejores individuos. Finalmente y para la solución final, se lleva a cabo una descodificación de esta cadena de caracteres, y lo que se obtiene es una posible solución no codificada del problema que se trata resolver, sería el fenotipo.

Operadores del algoritmo:

- **Selección**. Este operador se encarga de seleccionar aquellos individuos que van a participar en la reproducción, pueden ser todos los individuos de la población o solo parte de

ellas. Generalmente se va a dar una mayor probabilidad de selección a los individuos que mayor puntuación de la función de adaptación presenten. Para llevar a cabo este proceso existen distintos mecanismos, algunos son [13]:

- **Selección por ruleta**. Primero se hace una representación de los posibles padres en un gráfico circular, que será la ruleta. La proporción de círculo asignada a cada padre va a ser proporcional a su puntuación en la función de adaptación, con lo que los individuos más fuertes tendrán más posibilidades de reproducirse. La selección de los padres que se cruzarán entre sí se hace haciendo girar la ruleta.

- **Selección por torneo**. En primer lugar se define el número de individuos que van a participar en cada torneo y se escogen aleatoriamente de entre los de la población. A continuación el que tenga mayor valor de la función de adaptación ganará el torneo y pasará a formar parte del grupo de padres que dará lugar a la siguiente población. De esta forma se asegura que el peor individuo no participe.

- **Selección elitista**. Consiste en mantener el mejor individuo en la población siguiente para no perder la información y realizar el resto de la selección mediante otro método.

- **Selección escalada**. Se basa en hacer una selección más exhaustiva y costosa a medida que se van mejorando los valores de la función de adaptación.

- **Selección Jerárquica**. Va a someter a cada generación a varios procesos de selección dónde los primeros serán más rápidos y

permitirán el paso de un mayor número de individuos y los siguientes serán más restrictivos.

- **Cruce.** En este proceso tiene lugar el intercambio de información genética, los individuos seleccionados o padres generarán una descendencia, los hijos, que formarán la nueva población de posibles soluciones y así sucesivamente. Generalmente la población se mantiene a lo largo de la ejecución del algoritmo por lo que del cruce de dos padres surgen dos hijos con información genética heredada. Existen diferentes procedimientos de cruce que dependerán del modo de representación de la información en el genotipo; cruces simples, por puntos de corte, uniforme, combinaciones de las anteriores, etc.

- **Mutación.** Al igual que en la teoría de la evolución van a tener lugar mutaciones, las cuales consisten en realizar algún cambio en alguno de los genes de individuos pudiendo dar lugar a mejoras o empeoramientos de las soluciones. Van a ser importantes ya que ayudan a evitar estancamientos en óptimos locales. Algunos tipos de mutación son por ejemplo: la mutación por intercambio, por inserción o por mezcla de genes.

Por lo tanto para la aplicación de este tipo de algoritmos en primer lugar será necesario identificar los parámetros de los que depende el problema, realizar una codificación de los mismos y definir una función de adaptación adecuada. La mejor de las codificaciones posibles va a ser aquella en la que un único individuo queda representado por una única solución. A continuación se define la población inicial; esta definición se puede realizar de forma aleatoria o bien en función de alguna regla o norma inicial. En cualquier caso el número de individuos que forman la población se mantendrá a lo largo de todo el proceso. Seguidamente esta población inicial se evalúa y se realiza la selección de individuos que darán lugar a la siguiente generación. Tras este proceso se realiza el cruce o intercambio de información entre los padres seleccionados. El tipo de

cruce dependerá de la codificación realizada, existen dos grandes tipos: binaria y no binaria. Y por último puede darse o no una mutación dependiendo de una probabilidad definida, lo cual permitirá introducir nuevos espacios sin explorar dando lugar a posibles mejoras.
Este proceso se repetirá de forma reiterada hasta que se cumpla un criterio de parada establecido.

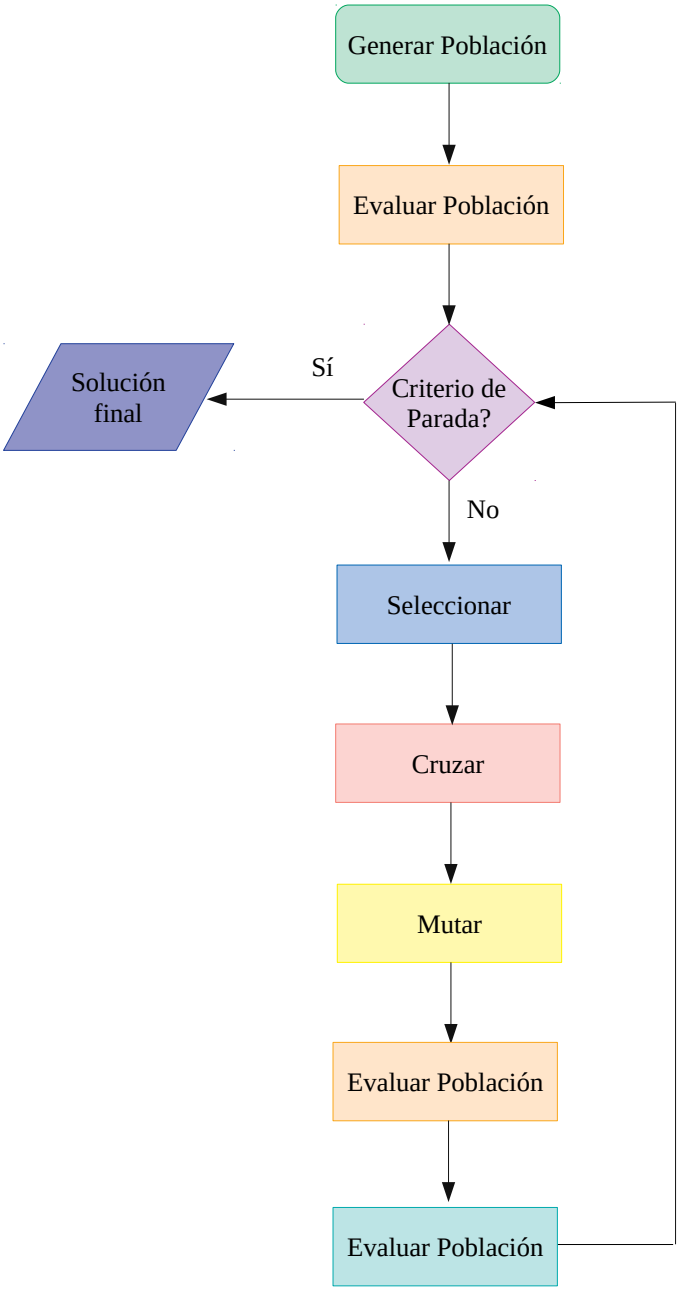


Figura 1. Diagrama general de un algoritmo genético.

2.2. Algoritmo de recocido simulado

Este algoritmo es análogo al proceso de recocido del acero que se emplea para modificar la microestructura de los metales.

El recocido es un tratamiento térmico que se aplica a metales que han sido tratados anteriormente en frío y con el que se pretende: eliminar tensiones, mejorar la tenacidad, la ductilidad, ablandar el material u obtener una microestructura determinada.
Consta de tres etapas diferenciadas:

- **Calentamiento**. Consiste en calentar el material metálico hasta una temperatura determinada, la temperatura de recocido.

- **Mantenimiento.** A continuación se mantiene ese material a dicha temperatura hasta que esta sea uniforme en todo el material.

- **Enfriamiento.** Es la última etapa y se debe hacer lentamente de forma que el material sea capaz de irse estabilizando a medida que esta desciende hasta llegar a la temperatura final, generalmente la temperatura ambiente, con una mejora de una o varias propiedades.

Existen distintos tipos de recocido, en función de lo que se quiera conseguir del material: recocido de eliminación de tensiones, normalización, recocido total, recocido de globulización, completo, incompleto, etc.

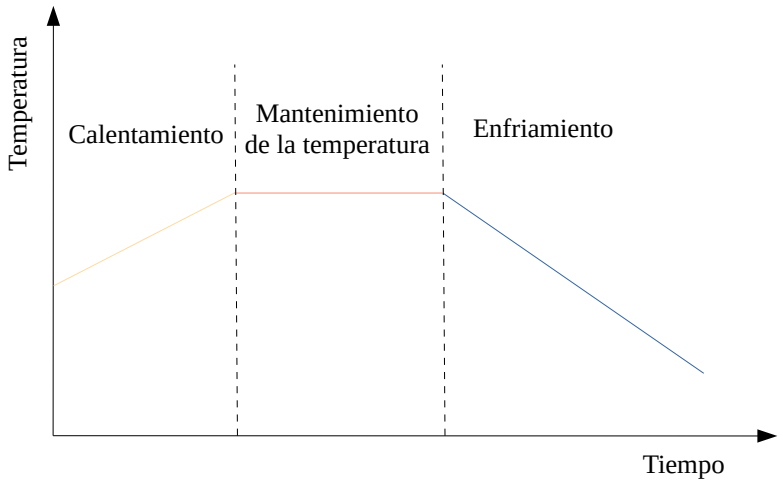


Figura 2. Diagrama de enfriamiento mediante recocido.

En 1953 Metrópolis *et al.* modeló, mediante la termodinámica estadística, el comportamiento microestructural de los metales durante este proceso de enfriamiento; más concretamente los cambios energéticos y movimientos atómicos que sufren las partículas de la microestructura al disminuir su temperatura hasta llegar a un estado estable, el estado de congelación.
En este modelo cada estado correspondiente tiene un desplazamiento atómico aleatorio, lo cual va a producir una variación de energía, δE, este desplazamiento atómico se podrá aceptar o no en función de si la energía del sistema aumenta o decrece. Si la variación de energía es negativa, es decir, la energía del sistema tras el desplazamiento, $E_{f,}$ es menor que la del sistema inicial, E_i , el desplazamiento va a ser aceptado automáticamente, sin embargo, si la variación de energía es positiva, el desplazamiento se podrá aceptar o no en función de una probabilidad de aceptación que viene definida por:

$$P(\delta E) = e^{-\left(\frac{\delta E}{kt}\right)}$$

Ecuación 1. Probabilidad de aceptación de un desplazamiento atómico.

Donde:

δE = $E_f - E_i$ = Variación de energía.
t = Temperatura.
k = Constante de Boltzman.

El hecho de que el enfriamiento se lleve a cabo lentamente permite al sistema estabilizar la microestructura en cada temperatura.

Más tarde Kirkpatrick y Gelatt en 1983, fueron los primeros en aplicarlo en el campo de la optimización combinatoria, introduciendo el concepto de Recocido Simulado, definido como un algoritmo de aceptación de entornos probabilístico que se va adaptando en cada iteración y que está basado en el modelado del enfriamiento de los metales realizado por Metropolis.
En este algoritmo la función de coste queda definida por la energía, la configuración atómica serán los parámetros del problema a optimizar y la temperatura será el parámetro de control. Para cada temperatura se generarán distintas configuraciones, estas serán posibles soluciones (vecindario). La temperatura inicial para la cual de comienzo el algoritmo, al igual que en el tratamiento de recocido será por lo general elevada ya que debe permitir la aceptación de todas las soluciones del entorno, es decir todos los movimientos posibles, este sería el caso ideal, e irá disminuyéndose a medida que avanza el algoritmo mediante un parámetro, α, que es la velocidad de enfriamiento:

$$T_{i+1} = \alpha\, T_i$$

Ecuación 2. Descenso de la temperatura.

El factor de enfriamiento en este caso es geométrico, se pueden definir otros tipos de factores de enfriamiento, aunque este es de los más empleados en la utilización de este tipo de algoritmos.
Para cada iteración se evalúan los vecinos creados y si la función de coste mejora son aceptados automáticamente y pasa a formar parte de los miembros de la siguiente iteración, mientras que si tiene un valor de la función de coste peor que su anterior podrá ser aceptado con una probabilidad determinada que dependerá del parámetro de control, la temperatura. Con este mecanismo el algoritmo consigue escapar de los óptimos locales.
La probabilidad de aceptación de estos estados peores vendrá definida por:

$$P(aceptación) = e^{-\left(\frac{\delta}{t}\right)}$$

Ecuación 3. Probabilidad de aceptación de un estado peor.

Donde:

δ = Incremento de la función de coste.

t = Temperatura.

Esta probabilidad de aceptación irá disminuyendo a medida que avance el algoritmo, ya que el parámetro temperatura es directamente proporcional a la probabilidad de aceptación, por lo tanto a medida que "se enfría" el algoritmo la probabilidad de aceptación va ser menor reduciendo la posibilidad de ir hacia soluciones peores.
Se pueden definir dos fases en base a esta disminución de la probabilidad de aceptación, una primera de exploración, en la que el algoritmo va a evaluar un mayor número de estados peores debido a la elevada temperatura y seguidamente una fase de explotación, en la que el algoritmo ya tiende hacia el óptimo debido a que la probabilidad de aceptar un estado peor será mucho menor.

El tamaño de la población se puede mantener o variar a lo largo del proceso. Habrá que definir también, a parte de la temperatura inicial y de la velocidad de enfriamiento, el estado de congelación que sería el criterio de parada, este puede ser un número determinado de iteraciones, una temperatura final, un valor de la función de evaluación, etc. En el proceso que se ha descrito anteriormente tendría lugar un descenso paulatino de la temperatura lo que iría restringiendo a su vez la aceptación de movimientos hasta llegar a un punto en el que no se aceptara ningún movimiento más, obteniéndose una solución final independiente de la solución inicial. Este proceso equivaldría a un enfriamiento infinitamente lento e ideal, según la teoría de las cadenas de Markov [19] de ser así el proceso el algoritmo convergería al óptimo global con una probabilidad de 1; sin embargo en problemas finitos la convergencia al óptimo no está garantizada.

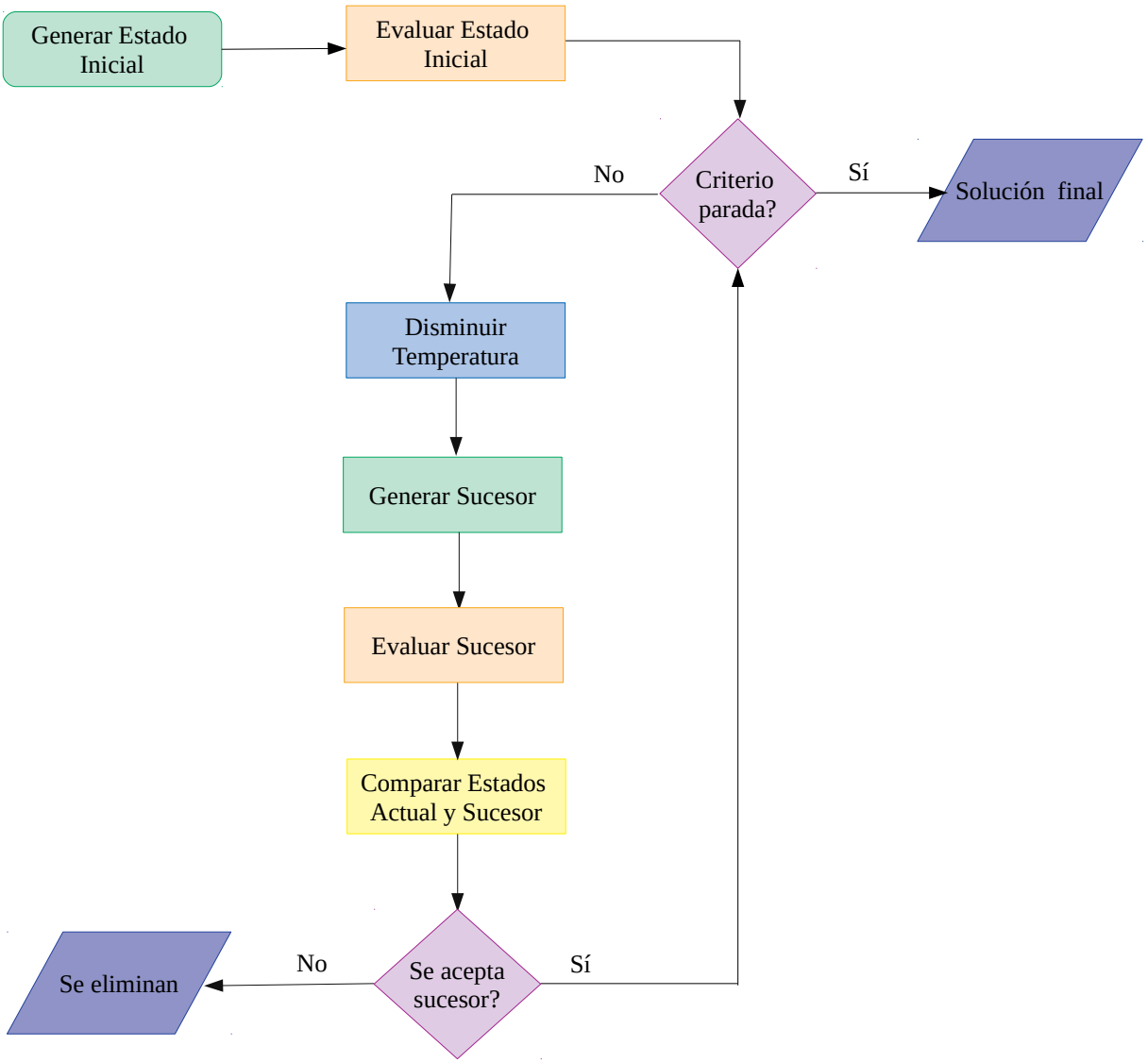


Figura 3. Diagrama general de un algoritmo de recocido simulado.

CAPÍTULO III. DEFINICIÓN DEL PROBLEMA.

1. Descripción del problema.

El problema consiste en realizar la distribución horaria de todos los grados impartidos por la Escuela de Ingenierías Industriales en una nueva y única sede, el Aulario IndUVA (Industriales de la UVA), optimizando el uso de las aulas. Los grados que imparte actualmente la Escuela y para los que se presenta el problema son:

- Grado en Ingeniería Eléctrica.
- Grado en Ingeniería Química.
- Grado en Ingeniería de Organización Industrial.
- Grado en Ingeniería en Diseño Industrial y Desarrollo de Producto.
- Grado en Ingeniería Electrónica Industrial y Automática.
- Grado en Ingeniería Mecánica.
- Grado en Ingeniería en Tecnologías Industriales.

2. Recursos disponibles.

Los recursos van a ser las aulas disponibles del nuevo edificio, va a haber aulas grandes, medianas y pequeñas, así como aulas de dibujo y de ordenadores en las que se impartirán los laboratorios informáticos. Para la docencia el edificio consta de:

- 16 aulas grandes con una capacidad de 80 personas.

- 4 aulas medianas con una capacidad de 60 personas.
- 6 aulas pequeñas con una capacidad de 40 personas.
- 8 laboratorios informáticos con una capacidad de 30 personas.
- 2 aulas de dibujo con una capacidad de 40 personas.

3. Restricciones generales.

A continuación, se va a definir el conjunto de restricciones que debe cumplir la asignación horaria a realizar por la aplicación desarrollada, algunos requisitos de la aplicación quedarán determinados por la definición de las propias restricciones:

- Los alumnos de diferentes grados no pueden tener clases comunes; aunque las materias sí sean comunes, cada alumno pertenecerá a una especialidad.
- La cantidad de estudiantes asistentes a una clase no puede sobrepasar la capacidad del aula:

  Aulas grandes → 80 personas.

  Aulas medianas → 60 personas.

  Aulas pequeñas → 40 personas.

  Laboratorios informáticos → 30 personas.

  Aulas de dibujo → 40 personas.

- En cada materia a impartir se pueden distinguir clases de teoría (horas T), problemas (horas A), seminario (horas S) y laboratorio (horas L). Cada grupo de teoría define un grupo completo de alumnos pertenecientes a un grado, un curso y un cuatrimestre y por lo tanto con un horario común para estos. Este grupo, definido por el grupo de teoría, se subdivide en los subgrupos de clases de problemas, seminario y laboratorio.

Esta información debe ser introducida por el usuario a la hora de realizar una nueva asignación horario ya que la determina el rectorado, aunque las subdivisiones que se usan habitualmente, las cuales dependen del tamaño del grupo, son las siguientes:

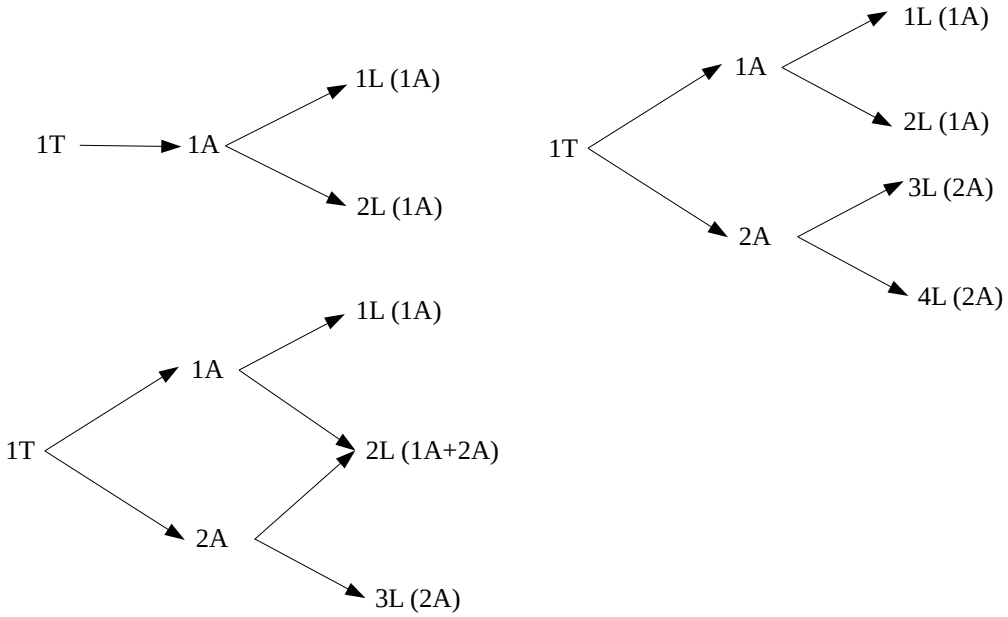


Figura 4. Subdivisiones típicas de los grupos de teoría.

- Las clases de teoría y problemas deben repartirse equitativamente a lo largo de todas las semanas lectivas.

- Las clases de laboratorio y seminario se reparten en función de los requisitos docentes fijados por los departamentos, por lo tanto debe ser un parámetro a introducir por el usuario.

- El periodo lectivo puede ser de 14 o 15 semanas, deberá introducirlo el usuario.

- Todas las horas lectivas deben tener un aula asignada.

- Los subgrupos de problemas, seminarios y laboratorios deberán tener el mismo número de horas.

- El máximo de horas de problemas o teoría de una misma asignatura que se pueden dar seguidas es de 2 horas.

- No debe haber horas libres entre horas lectivas.

- No todos los laboratorios requieren de un aula de informática, será necesario definir cuáles.

- Dependiendo del grado se definirán los grupos de tarde o de mañana, actualmente se procede de la siguiente manera:

    - El primer curso va de mañana, salvo que haya más de un grupo, en tal caso uno debe ser de tarde.

    - El segundo curso va de tarde, excepto si hay más de un grupo, en tal caso uno debe ser de mañana.

    - Tercero va de mañana.

    - Cuarto va de tarde.

  Todos los grados se distribuyen así excepto para el Grado de Ingeniería Química que es justo al contrario.

- Siempre que sea posible la impartición de las clases seguirá lo siguiente:

    - Clases de teoría en aulas grandes o medianas.

    - Clases de problemas en aulas pequeñas o medianas.

    - Seminarios en aulas pequeñas.

- La franja horaria en la que se puede llevar a cabo la docencia es:

    - Para los grupos de mañana de 8:00 h a 14:00, siendo preferible no usar la primera hora de forma habitual.

    - Para los grupos de tarde de 16:00 a 22:00 h, siendo preferible no usar la última hora de forma habitual.

- Puede haber laboratorios que se encuentren fuera del periodo, mañana o tarde, asignado para un determinado grupo, habrá que evitarlo cuando sea posible y en caso de haberlos tienen que ser definidos por el usuario.

- Para cada grupo perteneciente a un grado, curso y cuatrimestre, es deseable que se realicen el menor número de cambios de aula posible, para ello se asignará un único aula para las horas de teoría y otros aulas distintas para las clases de seminario, problemas y laboratorios, aunque no habría problema en que alguna de estas se puede dar en el mismo aula de teoría.

4. Datos.

Los datos correspondientes a las horas y asignaturas pertenecientes a cada Grado se incluyen en el Anexo , apartado I. Datos lectivos.

5. Planteamiento web.

En el desarrollo de la aplicación web se van a plantear dos etapas asociadas a la estructura de programación planteada, desarrollo Front-end y desarrollo Back-end.

- En primer lugar se va a desarrollar el Back-end, que corresponde a la parte de la aplicación que se va a encargar de obtener los horarios a partir de los datos introducidos. Va a ser la programación del algoritmo, se podría decir que es el “cerebro” de la aplicación.

- En segundo lugar se procederá al desarrollo del Front-end, que corresponde a la interfaz de usuario, es decir, la página web a través de la cual el usuario se comunica con el “cerebro” y este le devuelve la información correspondiente a las peticiones que realice el usuario.

CAPÍTULO IV. IMPLEMENTACIÓN ALGORÍTMICA.

Como ya se ha comentado en la introducción una de las pretensiones de este proyecto es programar un método matemático que represente el problema planteado y optimizar así el uso de aulas, para ello se han programado las dos técnicas algorítmicas anteriormente introducidas: algoritmo genético y recocido simulado. El que se hayan implementados dos algoritmos es debido a la imposibilidad de obtener resultados satisfactorios con el primero de ellos.

Inicialmente se escogió el algoritmo genético por ser este el más adecuado y empleado por otras instituciones lectivas para resolver este tipo de asignaciones. Es fácil encontrar gran cantidad de trabajos realizados con este algoritmo que llegan a soluciones aceptables, sin embargo en lo que respecta este proyecto no ha sido el caso. Por ello siguiendo el principio KISS (del inglés *Keep it simple, stupid!*: "Mantenlo simple, ¡estúpido!"), se procedió a la búsqueda y programación de un algoritmo más sencillo que pudiera permitir la obtención de soluciones válidas.

Seguidamente se planteó llevar a cabo un algoritmo voraz, que es una de las estrategias de búsqueda más simples que existe, el cual consiste en escoger a cada paso la opción óptima hasta llegar a una solución, sin embargo esto conlleva un rediseño total del planteamiento computacional del problema y por lo tanto de la programación. También se intuye que los tiempo de computación pueden ser mucho mayores debido a la complejidad de las restricciones y al gran número de posibles combinaciones existentes, sobretodo al inicio de la ejecución del problema, las cuales deberían ser todas planteadas y evaluadas para elegir la mejor de ellas. Es por eso que esta opción se descarta.

Finalmente con el fin de no desaprovechar todo el trabajo de programación se procede a la elección de un algoritmo de recocido simulado ya que su estructura es similar a la del algoritmo genético. Estos son algunos de los factores por los cuales se procede a llevar a cabo la implementación de este algoritmo:

- El algoritmo genético parte de una población inicial generada de forma aleatoria y formada por un conjunto de individuos o posibles soluciones, sin embargo, el algoritmo de recocido simulado parte de una única solución inicial, vecino, a partir del cual se generará un vecindario. Ambos tienen una serie de restricciones que marcan

la evolución del algoritmo y son evaluadas mediante una función de evaluación.

- Las restricciones son las mismas para ambos algoritmos, por lo que no es necesario volver a realizar la programación de las mismas.

- Igualmente la función de evaluación se mantiene igual para los dos algoritmos.

- La programación del algoritmo de recocido simulado es mucho más simple que la del algoritmo genético, ya que mientras que en el algoritmo genético los individuos deben intercambiar información cromosómica entre ellos, en el algoritmo de recocido simulado cada vecino que conforma la población mantiene su información y el mecanismo de evolución es la comparación de este con su estado sucesor aceptándose los estados mejores y los peores bajo una determina probabilidad.

- Otro de los puntos más decisivos junto con el anterior es que la estructura de datos es compatible también para el modelo de recocido. La estructura planteada en la programación del algoritmo genético es una estructura cúbica en la que para cada día, hora y aula (división espacio temporal), se asigna un conjunto asignatura, grupo y tipo de grupo, por lo tanto no es necesario replantearla.

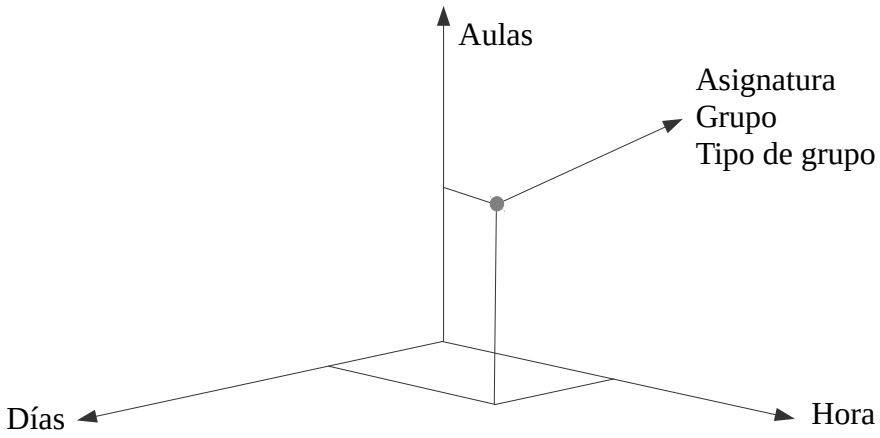


Figura 5. Representación de la estructura de datos.

- Por último el proceso de recocido de los metales había sido objeto de estudio en la asignatura de Ciencia de Materiales impartida en segundo curso por lo que se trataba de un proceso ya conocido y analizado. Al igual que los algoritmos genéticos habían sido estudiados en las asignaturas de Diseño de Sistemas Productivos y Logísticos y Métodos Matemáticos.

1. Algoritmo genético.

1.1. Analogía y representación de datos.

En primer lugar se van a definir una serie de asociaciones que se han empleado en la programación del algoritmo para este problema en concreto.

Conceptualmente cada individuo de la población va a tener un único cromosoma que va a estar formado por tres genes:

- **Gen 1**. Va a ser definido por el usuario será la elección del cuatrimestre (primero o segundo) y el turno (mañana o tarde) para el cual se va a realizar la asignación horario, pudiendo tomar por lo tanto los siguientes valores:
    - Primer cuatrimestre, turno de mañana.
    - Primer cuatrimestre, turno de tarde.
    - Segundo cuatrimestre, turno de mañana.
    - Segundo cuatrimestre, turno de tarde.

Este gen va a ser el mismo para todos los individuos que se creen en la ejecución del algoritmo y se va a mantener constante. Va a servir para seleccionar las asignaturas con las que se debe ejecutar el algoritmo.

- **Gen 2**. Va a ser la estructura de la distribución espacio temporal del horario. Todos los aulas que se emplean tendrán para cada día que forma un cuatrimestre los periodos correspondientes en los que se divide este, que es cuando tiene lugar la impartición/asignación de la clase lectiva / asignatura.

- **Gen 3**. Es la información lectiva y del alumnado, las asignaturas, el grupo y el tipo de grupo.

El conjunto de genes forman el genotipo del algoritmo, que ha seguido una codificación numérica, no binaria, para todas las variables (asignatura, aula, cuatrimestre, turno, día, grado, grupo, hora, tipo de aula y tipo de grupo), generalmente una enumeración, excepto cuando había códigos identificativos dados por la Escuela de Ingenierías, como es el caso de las asignaturas y los grados.
El fenotipo será por lo tanto la interpretación de esta codificación. Cada cromosoma va a dar lugar a la definición de un único individuo, el cual puede ser una posible solución, y el conjunto de estos formará la población sobre la que trabaja el algoritmo genético.

Esta es la forma en la que se han representado los datos, se trata de una estructura cúbica en la que para cada conjunto temporal definido por un aula, un día y una hora se va a realizar una asignación de una asignatura, un grupo y un tipo de grupo.
La población inicial se definirá realizando una asignación aleatoria, a partir de esta se ejecutará el algoritmo que se irá encargando de que se cumplan las restricciones en la medida de lo posible.

1.2. Funcionamiento.

El funcionamiento básico del algoritmo será el descrito en capítulos anteriores.
Centrando el problema que se plantea en este proyecto, un esquema concreto de los pasos que debe seguir la programación computacional del mismo es:

- Iniciar la población. Se hace una asignación de forma aleatoria.

- Evaluar cada individuo que compone la población por medio de la función de evaluación, cuya puntuación representa el grado de cumplimiento de las restricciones.

- Repetir los siguientes pasos hasta que se cumpla la condición de parada:

    - Seleccionar a los padres que van a intercambiar el material genético.

    - Cruzar los padres de dos en dos, los cuales intercambian información cromosómica para lo obtención de los hijos.

    - Mutar parte de la población, según la probabilidad de mutación definida.

    - Evaluar la nueva población generada.

  Se pueden dar dos condiciones de parada, que el algoritmo llegue a la puntuación máxima, esto es posible debido a que todas las restricciones están normalizadas, o que se cumplan el número de iteraciones a realizar por el algoritmo definidas por el usuario.

- Devolver el mejor individuo de la última población evaluada.

Los parámetros que recibe el algoritmo van a ser:

- Los del primer gen, el cuatrimestre y turno, para el cual se realiza el horario.

- El tamaño de la población.

- La probabilidad de mutación.

- La proporción de individuos de la población que participan en el cruce.

1.3. Selección, cruce, mutación y adaptación.

Los mecanismos de selección, cruce y mutación realizados por el algoritmo se llevan a cabo de la siguiente manera:

- **Selección.** Se va a cruzar solamente una proporción de la población, serán los definidos como cromosomas a cruzar. Cuanto mayor sea la proporción de cruce más pesado será el procesamiento y por lo tanto aumentará el tiempo de ejecución, no teniendo por qué darse una mejora de la solución, hay que definir una proporción que de un equilibrio entre tiempos de ejecución y calidad de la solución. La selección se realiza mediante el método de la ruleta, en el cual vendrán representados únicamente individuos que se van a cruzar. Estos serán evaluados y su puntuación (f_i) definirá de manera proporcional la fracción de ruleta que les represente.

$$\%f_i = \frac{f_i}{\sum f_i}$$

Ecuación 4. Porcentaje de aptitud de cada individuo respecto del total.

De esta forma los individuos mejores, aquellos con más puntuación, tendrán más posibilidades de ser elegidos.

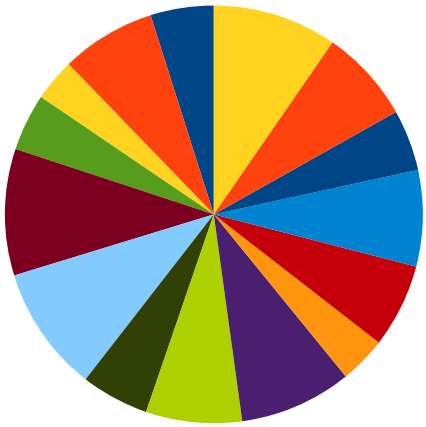


Figura 6. Ejemplo de población representada mediante ruleta.

A consecuencia de esta representación se podrá dar el caso de que un padre se reproduzca varias veces o que no se reproduzca ninguna.

- **Cruce.** El cruce a realizar depende de la representación de la información que se ha llevado a cabo, en este caso la representación es numérica no binaria, pero además está en forma de cubo (figura 5), lo cual introduce una complejidad adicional al problema.
  El cruce que debiera llevarse a cabo es de tipo permutacional. Un ejemplo de este tipo de corte sería el siguiente: conocido el problema del viajante, el cual consiste en dadas un conjunto de ciudades, encontrar el recorrido más corto que las recorra todas y termine en la ciudad de origen pasando una única vez por cada una de ellas.
  Si se realiza un cruce para este problema para dos padres que recorren 4 ciudades (a, b, c y d) según este orden, el padre 1: a-d-c-b, y el padre 2: b-a-d-c, definiendo como punto de corte la segunda ciudad, se generarían dos hijos, el hijo 1: c-d-b-a y el hijo 2: b-a-d-c.

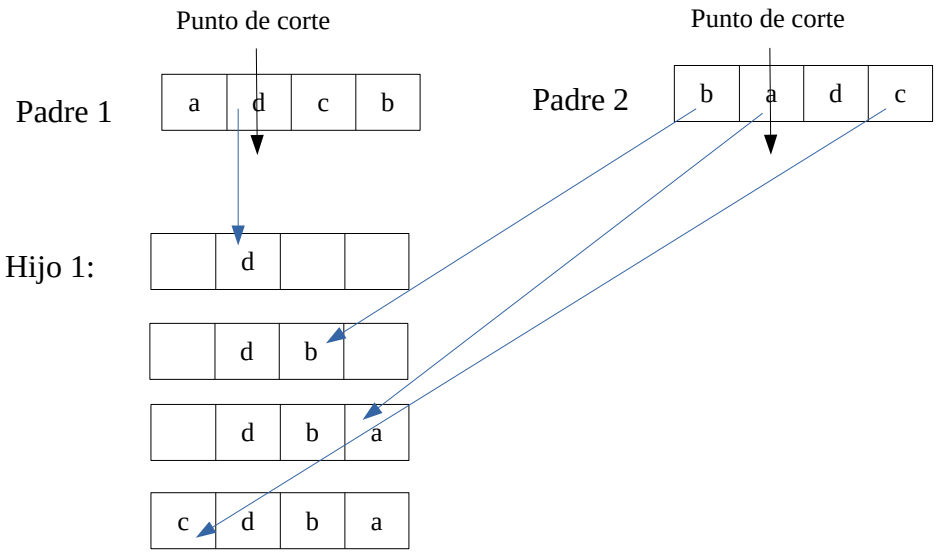


Figura 7. Obtención del hijo 1 por permutación.

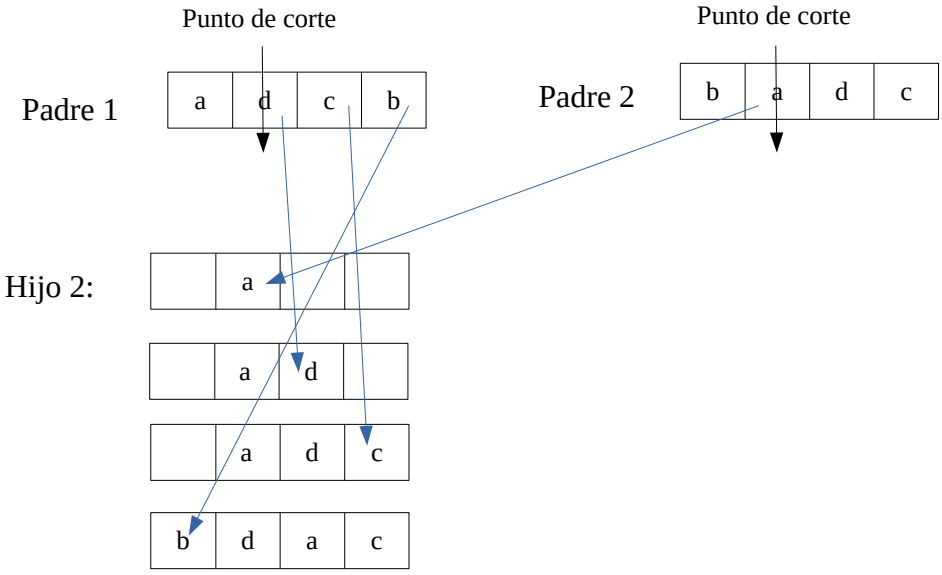


Figura 8. Obtención del hijo 2 por permutación.

Como se observa la segunda ciudad se va a mantener en su posición para cada uno de los descendientes, ya que es el punto dónde se realiza el corte, y a continuación con la

información del otro padre se van a llenar las posiciones sucesoras siguiendo el orden en el que se presentan y sin repetir la información.
De esta manera se van a obtener combinaciones válidas de ciudades, ya que ninguna ciudad se repite y en caso de que hubiera algún trayecto que no se pudiera llevar a cabo lo que se haría es dar una gran penalización al algoritmo para que no tenga menos posibilidades frente a otras soluciones candidatas. En este tipo de problemas tan sencillos generalmente se puede asegurar una convergencia.

Para el problema que se presenta habría que establecer un punto de corte y dada la representación cúbica de datos es realmente difícil su definición y programación. Además sería necesario tener en cuenta demasiada información, lo hijos deben producir poblaciones mejoradas genéticamente, para ello en primer lugar habría que asegurar que las asignaturas intercambiadas pertenecieran al mismo grado y curso, y si esto no diera lugar a una evolución satisfactoria es probable que hubiera que considerar a mayores que la subdivisión grupal y el tipo de grupos que intervienen.
Todo esto llevaría a un algoritmo con un elevado coste temporal y sin garantía de obtención de un resultado válido.

- **Mutación.** Se realiza en función de la proporción de mutación que se define antes de ejecutar el algoritmo genético. La van a experimentar tanto los cromosomas a cruzar, como los cromosomas no cruzados, y supondrá la generación aleatoria de un nuevo individuo.

- **Adaptación.** La función de adaptación va a ser definida como la suma de la puntuación obtenida de la evaluación de cada una de las restricciones (R_1, R_2,…,R_n) para un individuo determinado. El peso, p_i, va a ser el mismo, e igual a la unidad, para todas las restricciones:

$$f(adaptación) = \sum_{i=1}^{n} R_i \times p_i$$

Ecuación 4. Función de adaptación.

Cada una de las restricciones se han normalizado para poder estimar la calidad de la solución obtenida ya que así habrá una puntuación máxima que implicaría ser un a solución perfecta, entendiendo por tal aquella que cumple todos los requisitos impuestos.
De esta forma se establece también un nuevo criterio de parada de ejecución del algoritmo, la obtención de la máxima puntuación. Se van a definir 7 restricciones:

- **Restricción 0**. Por cada día de la semana no pueden haber horas vacías entre clase y clase, es preferible que los alumnos de un mismo grupo, cuatrimestre y curso tengan todas las asignaturas seguidas.

- **Restricción 1.** No puede haber más de dos horas seguida de la misma materia, siempre que no sean clases de tipo laboratorio o seminario.

- **Restricción 2.** Se penalizarán las franjas horarias primera y última de los turnos de tanto de mañana como de tarde con el fin de que se asignen lo menos posible.

- **Restricción 3.** Deben haber suficientes aulas pequeñas para asignar las clases de problemas.

- **Restricción 4.** Deben haber suficientes aulas de informática para asignar los laboratorios que requieran dicho aula.

- **Restricción 5.** Es necesario que haya suficientes aulas de dibujo para impartir las asignaturas que requieren este tipo de aula.

- **Restricción 6.** Es necesario que haya suficientes aulas pequeñas o medianas para asignar las horas de seminario.

1.3. Conclusiones.

No ha sido posible realizar una programación completa del método debido a la complejidad de los cruces para esa representación de datos. No se ha obtenido ningún tipo de solución válida, pero si se obtienen resultados procedentes de la asignación inicial y de las mutaciones, las cuales si están programadas. Como es lógico estas no obtienen buenos resultados.

El llevar a cabo esta programación, tal y como se plantea el problema, es muy probable que aumentara notablemente los tiempos de ejecución ya que la cantidad de información a tratar es muy elevada, y debido a eso y a la complejidad de las restricciones que el horario debe cumplir la convergencia hacia una solución perfecta no iba a tener cabida en un tiempo razonable. También pudiera ser que se diera un estancamiento para una puntuación determinada o que tan si quiera se alcanzara una solución válida o aceptable. Para comprobarlo sería necesario realizar la programación del cruce.

2. Recocido simulado.

Ante la imposibilidad de completar la programación del problema representado mediante un algoritmo genético se procede a programar un algoritmo de recocido simulado, el cual es, a nivel de programación, más sencillo que el anterior.
El tratamiento que hace este algoritmo de los datos es más simple con lo que tiene probablemente más probabilidades de éxito tanto en el desarrollo de la programación como en la obtención de soluciones válidas para el problema de optimización de aulas al que se enfrenta.

2.1. Herencia genética y mejoras introducidas.

Se pretende aprovechar al máximo la programación ya realizada, por lo cual la estructura ternaria de representación de los datos (figura 5), el gen 1, que contiene la información del turno y cuatrimestre para el cual se realiza el horario, y la función de evaluación junto con las restricciones

van a ser programaciones heredadas y mantenidas para el algoritmo de recocido simulado.

También se van a introducir algunas mejoras respecto del algoritmo genético. En la asignación inicial del algoritmo de recocido se crean las estructuras que van a contener los datos, una por cada tipo de aula existente, y se procede a una asignación inicial aleatoria, con ello se consigue que los tipos de aulas asignados a las clases (teoría, problemas, laboratorio, dibujo) sean siempre correctas.

La principal diferencia entre los dos algoritmos es que esta asignación inicial se va a realizar sólo con las horas de teoría, por lo que el algoritmo de recocido simulado va a trabajar únicamente con las clases de teoría y serán las restricciones las que se encarguen de que haya suficientes aulas para los demás tipos de clases.
Tras ejecutar el algoritmo de recocido se realiza una asignación final y ordenada de las clases de problemas, laboratorios, seminarios y dibujo en las estructuras de datos creadas en durante la asignación inicial.

Habrá también una serie de asignaturas con laboratorios que no necesiten aulas de informática, en ese caso como sigue siendo necesario hacer una asignación temporal para la completa definición del horario se les asignará un aula virtual denominada DEPARTAMENTO, ya que serán los departamentos los que se encarguen de definir estos espacios.

2.2. Funcionamiento.

El funcionamiento genérico de este tipo de algoritmos es el descrito por la figura 3. Centrando el problema de optimización que se plantea en este proyecto, un esquema de los pasos que debe seguir la programación computacional son:

- Generar un estado inicial.

- Evaluar el estado inicial a partir de la función de evaluación.

- Repetir los siguientes pasos hasta que se cumpla una de las condiciones de parada, que son; que el algoritmo alcance la puntuación máxima o que se cumplan el número de iteraciones definidas por el usuario o número de enfriamientos, es decir, que llegue a la temperatura de congelación:

    - Para cada temperatura se repiten los siguientes pasos hasta que se llegue al número total de vecinos (individuos) que definen el vecindario (población):

        - Generar un sucesor para cada vecino.

        - Calcular la diferencia energética entre ambos, que será la diferencia entre las puntuaciones obtenidas en la función de evaluación.

        - Aceptar o rechazar el sucesor generado. Si la puntuación de la función de evaluación es mayor se acepta el sucesor automáticamente, mientras que si es menor, se aceptará si la probabilidad generada aleatoriamente es menor que la probabilidad de aceptación definida por la ecuación 1.1. Si se rechaza el sucesor, se mantiene el estado actual, que será el que pase al siguiente vecindario junto con los demás vecinos.

        - Actualizar el valor del mejor individuo de la población.

- Reducir la temperatura.

- Devolver el mejor vecino de la última población evaluada, que será la solución algorítmica propuesta.

2.3. Parametrización.

Los parámetros que recibe el algoritmo van a ser el cuatrimestre y turno para el cual se realiza el cuatrimestre (gen 1), la temperatura inicial, el factor de descenso, el número de enfriamientos y el número de iteraciones que deben realizarse para cada temperatura.

- **Parámetro de control inicial o temperatura inicial, T_0.**
  Para garantizar un buen funcionamiento del algoritmo debe ser lo suficientemente alta como para permitir que todos los cambios sean aceptados, es decir, que la probabilidad de pasar de un estado a otro sea elevada, lo cual implica que el sistema tiene un alto grado de libertad y garantiza una buena exploración del espacio de soluciones.

- **Número de vecinos o transiciones por enfriamiento, M.**
  Es el criterio de cambio de temperatura, por lo tanto se trata del número de iteraciones que se hacen en cada etapa para permitir que el sistema alcance el equilibrio. Es frecuente que este número varíe a lo largo del algoritmo, definiendo un número de aceptaciones que se permiten hacer en cada etapa, con lo que a medida que avanza el algoritmo, el alcanzar el equilibrio es más difícil, ya que se producen menos aceptaciones con lo que el número de vecinos aumenta.
  Para este problema y por simplicidad, se va a mantener constante durante toda la ejecución del algoritmo.

- **Velocidad de enfriamiento o factor de descenso, α.**
  Definido en la ecuación 2, va a ser de tipo geométrico, da la relación entre una temperatura y la siguiente.

- **Temperatura final, T_f, o número de enfriamientos, C.** Es el parámetro de control de final, sería la temperatura de congelación que define el estado estable al que converge el sistema al finalizar el enfriamiento, dónde se alcanzan las propiedades deseadas del metal.
  En este caso el criterio de parada va a ser un número definido de enfriamientos o iteraciones.

A pesar de ser un algoritmo muy sencillo, la dificultad que presenta se encuentra en llevar a cabo una buena definición de los parámetros citados ya que son los que condicionan la calidad de la solución final obtenida.

Se va a realizar en primer lugar una definición fundamentada de los parámetros [21]. A continuación se observarán las soluciones obtenidas y se procederá a la modificación de estos (en caso de ser necesario), hasta llegar a unos valores que proporcionen una solución equilibrada tanto en tiempos de computación como en calidad.

Para la temperatura inicial, la probabilidad de pasar al estado siguiente viene dada por:

$$p_{0\to 1} = e^{\frac{-\Delta f}{T_0}}$$

Ecuación 5. Probabilidad de aceptación de un estado siguiente.

Si $p_{0\to 1} = 0{,}99$ y $\Delta f \approx f_0$, entonces $T_o \approx 100\ f_0$., de esta forma queda definida la temperatura inicial, ya que el valor de la población inicial es conocido. Si se pretende obtener una solución final f_f con un error inferior a εf_0, con $\varepsilon < 1$, la temperatura final de la última iteración se estima a partir de:

$$p_{f\to h} = e^{\frac{-\varepsilon f_0}{T_f}} < \frac{1}{M}$$

Ecuación 6. Probabilidad de aceptación de un estado final.

Las transiciones por iteración viene dada por $M = \delta g^{\beta}$, donde δ y β son constantes y g es el número de grados de libertad del sistema.

Con lo que la temperatura final viene dada por:

$$T_f = \frac{-\varepsilon f_0}{\log(\frac{1}{M})} = \frac{\varepsilon f_0}{\beta \log(\delta g)}$$

Ecuación 7. Estimación de la temperatura final.

Para llegar a dicha temperatura final el sistema tiene que pasar por un número de iteraciones, C, que depende de la tasa de variación de la temperatura, α, que se puede estimar a partir de:

$$\alpha^C \cdot T_0 = T_f$$

Ecuación 8. Relación de temperaturas inicial y final.

Obteniendo:

$$C = \frac{\log \varepsilon - \log(100 \beta \log(\delta g))}{\log \alpha} < \frac{\log \varepsilon}{\log \alpha}$$

Ecuación 9. Número de iteraciones en función de los parámetros de recocido.

Si se realizan los cálculos para un valor del error $\varepsilon = 10^{-6}$ se obtiene:

α	0,9	0,946	0,974
C	130	240	524

Tabla 2. Número de iteraciones en función de la velocidad de enfriamiento.

Queda por lo tanto acotado el número de iteraciones independientemente del problema planteado y definidos los parámetros de funcionamiento del algoritmo.

Cambiando el valor de las transiciones por enfriamiento, M, se conseguirá una mayor o menor velocidad de ejecución del algoritmo, siendo conveniente siempre que el recocido se lleve a cabo lentamente, generalmente para valores de g=1 y δ=10 se puede obtener generalmente una lentitud suficiente que permita recorrer las diferentes temperaturas.

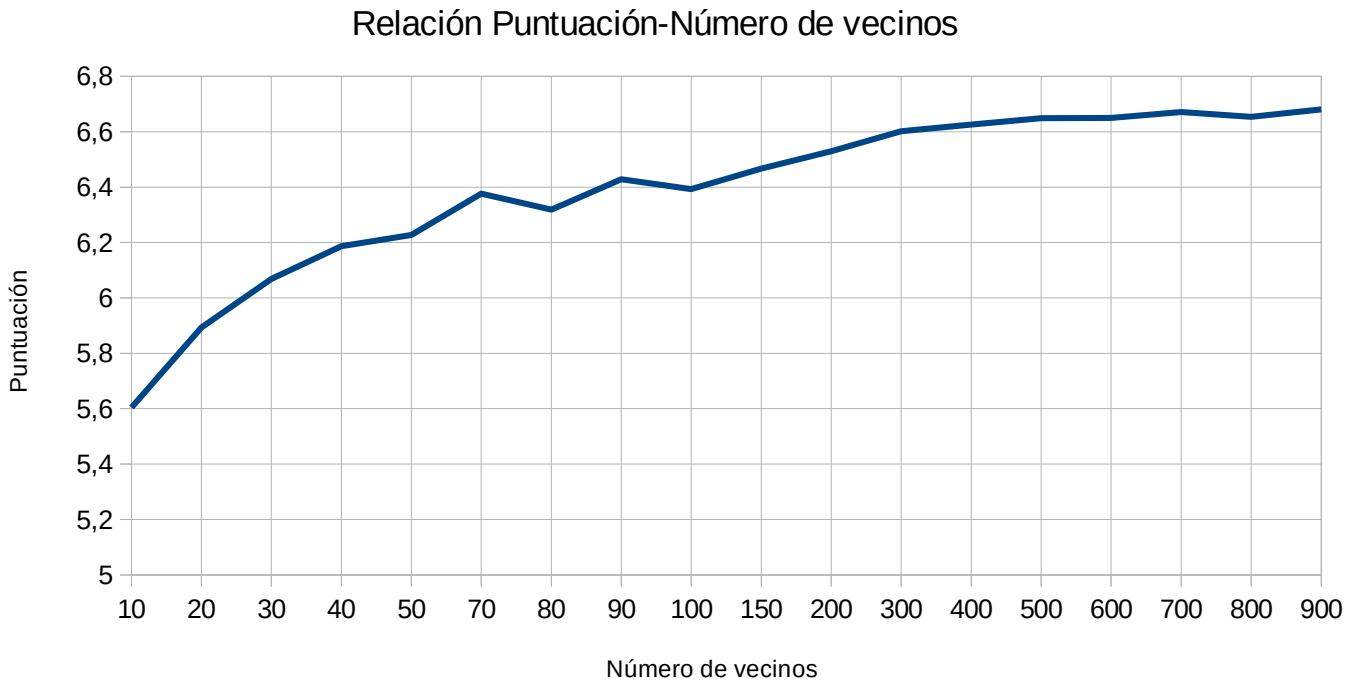
A partir de las ecuaciones anteriores se obtiene la definición inicial de los parámetros:

Parámetro	Valor
Temperatura inicial, T_o	504,67
Error, ε	10^{-6}
Velocidad de enfriamiento, α	0,974
Número de enfriamientos, C	524
Temperatura final, T_f	$5{,}104x10^{-4}$
Número de vecinos, M	10

Tabla 3. Valores iniciales de los parámetros del algoritmo.

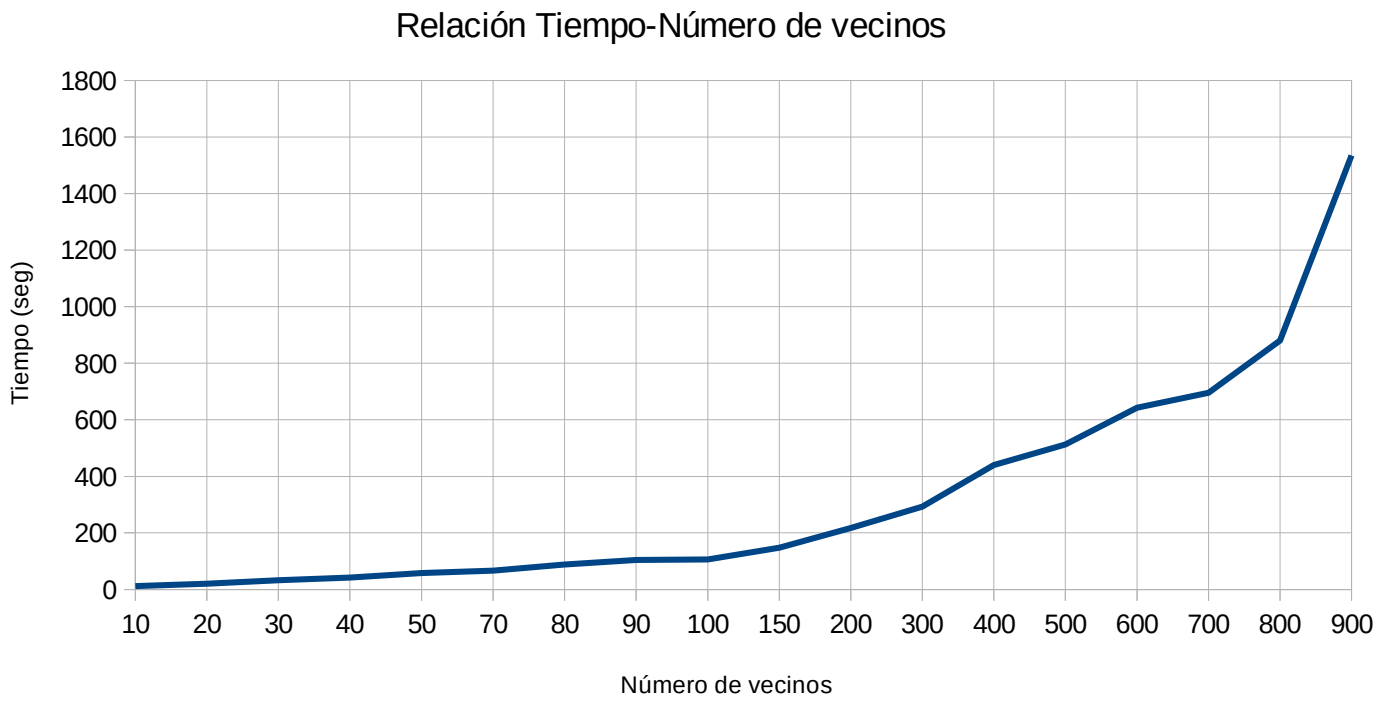
Si se ejecuta el algoritmo para estos valores, se obtiene un valor de la función de adaptación de 5,604. Dado que la puntuación máxima es de 7, va a ser necesario realizar un ajuste de los parámetros.

Para ello se va a empezar por variar el número de transiciones por iteración, M, y ver como afecta a la solución final del algoritmo.
Si se hace una representación del valor de la puntuación obtenida por el algoritmo en función del número de transiciones:



Gráfica 1. Relación puntuación – Número de vecinos.

Se observa como el aumento del número de vecinos implica un aumento de la puntuación obtenida, ya que para cada temperatura se hace una exploración más grande, sin embargo también aumenta el tiempo.
Si se representa gráficamente:

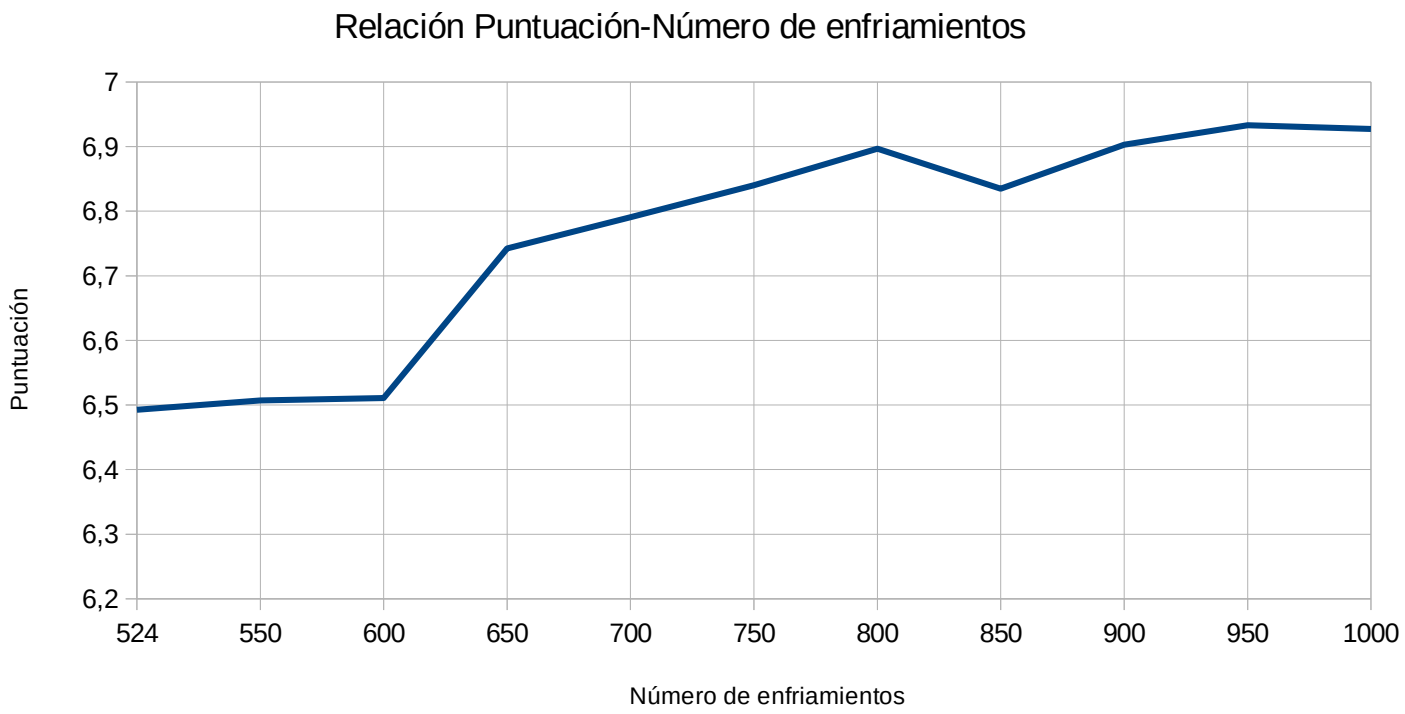


Gráfica 2. Relación tiempo – Número de vecinos.

A partir de ciertos valores el aumento del tiempo va a ser mucho más significativo que la mejora de la solución. Por ello se va a seleccionar un valor de M = 100. Para este valor las puntuaciones obtenidas se encuentran en torno a 6,4 puntos, y los valores de las restricciones:

- Restricción 0: 0.883
- Restricción 1: 0.763
- Restricción 2: 0.777
- Restricción 3: 1.000
- Restricción 4: 0.996
- Restricción 5: 0.973
- Restricción 6: 1.000

Dos de las restricciones han alcanzado su óptimo, pero no todas las que implican la asignación total de horas lectivas (restricciones 3, 4, 5 y 6) han alcanzado el óptimo, con lo que se va a proceder a la modificación del número total de enfriamientos a realizar para ver si un aumento de este supone una mejora en los resultados para un menor coste temporal.
La representación gráfica es la siguiente:



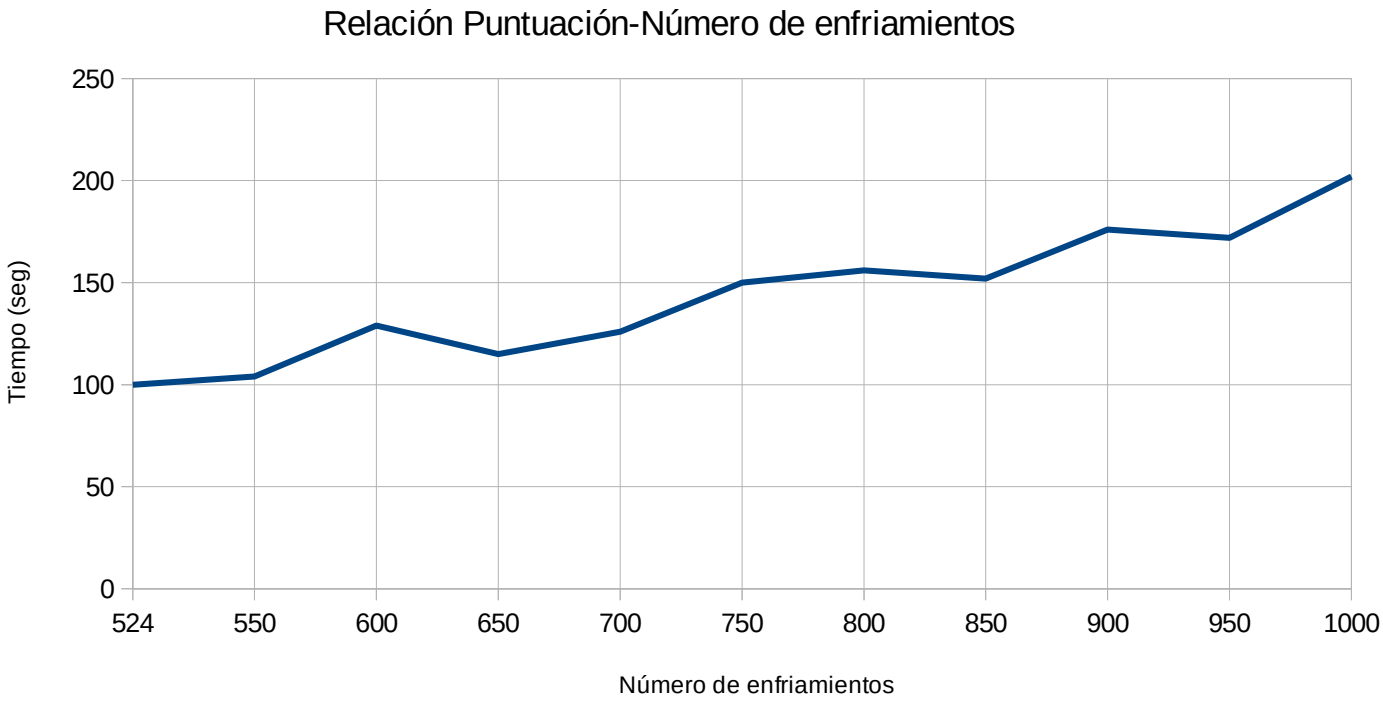
Gráfica 3. Relación puntuación – Número de enfriamientos.

Se puede apreciar como al aumentar el numero de iteraciones mejora la puntuación de las restricciones mucho más rápido que si se aumentan las transiciones por ciclo o número de vecinos.

Los tiempos de ejecución a pesar del aumento de enfriamientos son considerablemente menores respecto al aumento del número de transiciones por enfriamiento, por lo que el parámetro anterior se mantendrá con el nuevo valor dado, y para mejorar la puntuación final se modificará el número de enfriamientos.

A partir de los 700 enfriamientos se consigue que las restricciones 3, 4, 5 y 6 alcancen siempre la puntuación máxima, este será el valor mínimo que se le dará a este parámetro.

Si se siguen aumentando los enfriamientos tanto la puntuación como el tiempo subirán, por lo que pasará a ser el tiempo el factor que defina el número de enfriamientos y no la puntuación, ya que la mejora de esta una vez se han cumplido las restricciones anteriores no va a ser significativa. El número mínimo de enfriamientos va a definirse en C = 950 enfriamientos que corresponde a aproximadamente unos 3 minutos de ejecución.



Gráfica 4. Relación tiempo – Número de enfriamientos.

Por lo tanto el valor final de los parámetros que se recomienda utilizar para realizar las distintas asignaciones horarias son:

Parámetro	Valor
Temperatura inicial, T_o	504,67
Error, ε	10^{-6}
Velocidad de enfriamiento, α	0,974
Número de enfriamientos, C	950
Temperatura final, T_f	$1{,}808x10^{-8}$
Número de vecinos, M	100

Tabla 3. Valores finales de los parámetros del algoritmo.

Con estos valores de los parámetros ya definidos se procede al análisis y validación final de las soluciones obtenidas.

2.3. Análisis y validación de las soluciones.

Se van a obtener por medio de la ejecución del algoritmo soluciones para todos los horarios a realizar en un curso académico, es decir, primer y segundo cuatrimestre, grupos de mañana y de tarde.
Los resultados obtenidos por el programa se incluyen en el anexo, apartado II. Resultados obtenidos.

Es importante destacar que el cumplimiento de las restricciones número 3, 4, 5 y 6, las cuales hacen referencia a que haya suficiente número de aulas para dar las clases de problemas, seminarios, laboratorios con aula de informática y dibujo implica una asignación completa de todas las horas lectivas, por lo que siempre se deberán seleccionar soluciones cuya puntuación para estas restricciones sea de 1, que es su valor máximo.

Las soluciones obtenidas cumplen el requisito anterior, y se han llevado a cabo realizando una sobrecarga lectiva para asegurar el funcionamiento del algoritmo en cualquiera de los casos. La sobrecarga consiste en que para cada grupo de teoría de cada Ingeniería se ha subdividido en dos grupos de problemas, dos de seminario y tres de laboratorio, cuando generalmente no es lo habitual, por lo que la obtención de buenos resultados para estos casos asegura su funcionamiento para los que se llevan a cabo en la realidad. Para conseguir esto se ha aumentado el numero de iteraciones a

realizar por el algoritmo, ya que debido a la sobrecarga grupal para las iteraciones antes definidas no se obtenían asignaciones completas.

3. Conclusiones.

Este algoritmo si se ha podido programar manteniendo la estructura de datos y ha permitido obtener resultados válidos, que era uno de los objetivos del proyecto.
Además presenta muy buenos resultados a pesar de la complejidad del problema, ya que las asignaciones de las horas lectivas a aulas generalmente siempre son completas (restricciones 3, 4, 5 y 6 con valor 1 implican un asignación total).

En ocasiones, y debido a las distribuciones grupales, puede ser que las restricciones de asignación total no lleguen a su máxima puntuación en tal caso el parámetro que se debe variar es el número de enfriamientos a realizar por el algoritmo, generalmente un aumento de este da lugar a el cumplimiento de estas restricciones y por lo tanto a la obtención de una asignación horaria válida.
También puede suceder que el algoritmo no sea capaz de realizar una asignación inicial, aunque no es lo habitual (suele ser debido a fallos en los datos), en tal caso será necesario aumentar el número de días para los que se ejecuta el algoritmo y proceder después a una reasignación manual de estas horas en las franjas horarias penalizadas.

Otro de los puntos más importantes a destacar de esta herramienta es la rapidez del algoritmo en la resolución del problema, ya que en menos de media hora se tiene una asignación válida y aplicable, cuando este proceso actualmente se lleva a cabo de forma manual requiriendo tiempos de trabajo muy elevados, lo cual supone un ahorro de recursos muy importante.

CAPÍTULO V. MEJORAS Y LÍNEAS DE DESARROLLO.

Actualmente el algoritmo está pensado para realizar una distribución para todos y cada uno de los días que componen un cuatrimestre. Sin embargo se puede plantear también llevar a cabo una asignación semanal de horas que se repita a lo largo del cuatrimestre hasta cumplir con todas las horas, y haciendo una reserva de horas semanales para seminarios, laboratorios y clases que no se llevan a cabo todas las semanas. Para esto sería suficiente con cambiar los datos de entrada y poner las horas semanales, y variar ligeramente la programación. En primer lugar entraría el algoritmo para hacer la distribución semanal y posteriormente habría que programar la comprobación del cumplimiento de las horas totales, lo cual es relativamente sencillo.

Otra de las mejoras que se puede llevar a cabo, es la inclusión de una nueva restricción que actúe sobre los grupos de problemas ya que en la solución actual para dos grupos de problemas distintos se les asigna en la misma hora la misma materia, lo cual solo es posible hacer si hay profesorado suficiente y no es lo habitual. Por lo tanto otra línea de mejora sería el tener en cuenta las limitaciones docentes.

Quizás la línea de desarrollo más importante viene dada por el desarrollo de la aplicación, el planteamiento de esta va a ser fundamental en la utilidad del algoritmo, ya que por ejemplo se pueden llevar a cabo varias bases de datos (semanales y cuatrimestrales), se puede permitir la modificación e inclusión de estos datos de entrada, permitir la modificación de los parámetros del algoritmo para que en caso de distribuciones grupales complejas se pueda obtener una asignación total, hacer un tratamiento de salida de datos que sea más visual para el usuario y que pueda entregarse directamente al alumno, etc.
Sin duda, el trabajo más importante que definirá la capacidad del algoritmo programado será el correcto planteamiento de la funcionalidad de la aplicación web para su posterior realización y utilización.

Otra de la grandes posibilidades que se debe tener en cuenta es sin duda alguna, el ser consciente de que se dispone de una base de datos con todos los horarios pudiendo llevar a cabo otra aplicación en la que el alumno obtenga, introduciendo las asignaturas de las que se ha matriculado, su horario particular. Con ello pueden consultar qué clases tiene y en qué aulas, ya que en ingeniería es bastante común que los alumnos tengan asignaturas de varios cursos, y recibir avisos de las semanas que tienen laboratorios que se imparten en distintas sedes y aulas.

CAPÍTULO VI. CONCLUSIONES.

La falta de conocimiento ha sido una de las grandes constantes en este proyecto, sin duda la base de programación dada en el primer curso del Grado ha sido fundamental para poder llevar a cabo este proyecto junto con el desarrollo de la capacidad resolutiva desarrollada y adquirida durante estos años de estudio.

En primer lugar se intentó llevar a cabo la programación del algoritmo mediante el lenguaje Php, algo totalmente ineficiente e inadecuado para un problema de estas características. Seguidamente se procedió a buscar otro lenguaje de programación más adecuado, Java [20], que fue con el que se realizó finalmente el proyecto, ya que en cuanto a desarrollo web y multidispositivo es de los lenguajes más empleados y además es muy versátil. Para ello ha sido necesario emplear la plataforma de programación Eclipse (existen otras como Netbeans, pero se desecharon simplemente porque Eclipse resultaba más intuitiva).
Además Java presenta grandes similitudes con la programación en C++, lo cual facilitó ligeramente el desarrollo, sin embargo fue necesario estudiar una nueva estructuración de programación muy empleada en el campo de la informática que es la programación orientada a objetos, en la que está basada todo el proyecto.
Tras todo este gran proceso de aprendizaje, y con mucho esfuerzo, finalmente se ha conseguido terminar este proyecto y obtener soluciones más que satisfactorias.

En cuanto a los objetivos del proyecto se ha definido y programado el problema de optimización planteado (parte correspondiente al Back-end). Sin embargo el objetivo total del proyecto que era desarrollar una aplicación web no se ha alcanzado, faltaría la parte correspondiente a la interfaz de usuario, el Front-end (y la conexión entre ambos).

BIBLIOGRAFÍA

[1] D. Izquierdo y J. J. Ruiz , “PHPSimplex”, Versión 0.8 2016. [En línea]. Disponible en: http://www.phpsimplex.com/index.htm. [Accedido: 06-fer-2018]

[2] R. Bradly, *Blood and Steel. The Battle of Britain*, [Documental] Cromwell Productions Ltd., 1998.

[3] Geraargentina, “La Batalla de Inglaterra y el Blitz”, 23-febrero-2013. [Documental]. Disponible en: https://www.youtube.com/watch?v=W39Mtta2_LE. [Accedido: 10-feb-2018]

[4] R. H. Silva, “Aplicación de la teoría de investigación operativa a las operaciones militares en el nivel estratégico operacional”,trabajo final integrador, Escuela Superior de Guerra Conjunta de las Fuerzas Armadas, 2015 [En línea]. Disponible en: https://bit.ly/2xbtZPe. [Accedido: 11-feb-2018]

[5] Operational Research Society de Gran Bretaña, ORS., 1962.

[6] B. Melián, J. A. Moreno y J. M. Moreno, “Metaheurísticas: una visión global”, *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial*, ISSN: 1137-3601, vol. 7, no. 019, 2003 [En línea]. Disponible en: http://www.redalyc.org/pdf/925/92571901.pdf. [Accedido: 09-Sep-2016]

[7] A. Ramos, P. Sánchez, J. M. Ferrer, J. Barquín y P. Linares, “Modelos matemáticos de Optimización”, Escuela Técnica Superior de Ingeniería Departamento de Organización Industrial, 2010 [En línea]. Disponible en: https://bit.ly/2pAN8mK. [Accedido: 09-Sep-2016]

[8] E. A. Silver, et al, “A tutorial on Heuristic Methods”, *European Journal of Operational Reseach*, vol.5, Issue 3, pp.153-162, Septiembre 1980.

[9] F. Glover, “Future paths for integer programming links to artificial intelligence”, *Computers and Operationes Research*, vol. 13, Issue 5, pp.533-549, 1986.

[10] I. H. Osman y J. P. Kelly, *Meta-Heuristics: Theory and Applications*, Kluwer Academic Publishers, pp.1-21 Boston 1996.

[11] S. M. Sadiq, y Y. Habib, *Iterative Computer Algorithms With Applications in Engineering: Solving Combinatorial Optimization Problems*. Wiley-IEEE Computer Society Pr, 1999.

[12] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learnings.* Addison-Wesley, 1989 [En línea]. Disponible en: https://bit.ly/2siPM1R.[Accedido: 12-Sep-2016]

[13] C.R. Reeeves. Genetic Algorithms. Cap.3 en F. Glover y G. Kochenberger (eds.) Handbook on MetaHeuristics, 2003.

[14] J. Arranz y A. Parra, "Algoritmos Genéticos", Universidad Carlos III [En línea]. Disponible en: http://www.it.uc3m.es/jvillena/irc/practicas/06-07/05.pdf. [Accedido: 12-Sept-2016]

[15] K. A. Dowsland y B. A. Díaz, "Diseño de Heurísticas y Fundamentos del Recocido Simulado", *Revista Iberoamericana de Inteligencia Artificial*, vol. 7, no. 019, pp 93-102, 2003.

[16] W. D. Callister, Jr., *Introducción a la Ciencia e Ingeniería de los Materiales,* Ed. Reverté, S.A., 1995

[17] N. Metropolis, A. Rosenbluth, M. Rosenbluth. A. Teller y E. Teller, *"Equation of State Calculations by Fast Computing Machines"*, J.Chem. Phys., 21, 6, p1087-1092, 1953.

[18] Optimization by Simulated Annealing. S. Kirkpatrick; C. D. Gelatt; M. P. Vecchi Science, New Series, Vol. 220, No. 4598. (May 13, 1983), pp. 671-680.

[19] M. V. Espí, "Recocido simulado: nuevo algoritmo para la optimición de estructuras", Departamento de Estructuras de Edificación de la Escuela Técnica Superior de Arquitectura de Madrid, pp. 26-56, 1994.

[20] Pildorasinformaticas, "Curso Java", 5-may-2014. [Curso]. Disponible en: https://www.youtube.com/watch?v=coK4jM5wvko. [Accedido: 02-Mar-2017]