UNIVERSIDAD POLITÉCNICA DE MADRID

E.T.S. DE INGENIERÍA DE SISTEMAS INFORMÁTICOS

PROYECTO FIN DE GRADO

GRADO EN INGENIERÍA DEL SOFTWARE

Planificación nutricional mediante algoritmos evolutivos

Autor: Javier Quesada Pajares

Director: Cristian Ramírez Atencia

Madrid, 9 de mayo de 2024

Planificación nutricional mediante algoritmos evolutivos
Proyecto Fin de Grado, 9 de mayo de 2024
Autor: Javier Quesada Pajares
Director: Cristian Ramírez Atencia

E.T.S. de Ingeniería de Sistemas Informáticos
Campus Sur UPM, Carretera de Valencia (A-3), km. 7
28031, Madrid, España

Si deseas citar este trabajo, la entrada completa en BibTeX es la siguiente:

```
@mastersthesis{citekey,
title = {Planificación nutricional mediante algoritmos
evolutivos},
author = { \& }
school = {E.T.S. de Ingeniería de Sistemas Informáticos},
year = {2024},
month = {5},
type = {Proyecto Fin de Grado}
}
```

Agradecimientos

Aquí los agradecimientos que quieras dar. Y si no quieres, borras la entrada `\acknowledgements` de `report.tex` y ya está.

Resumen

Hola

Condensado no quiere decir incompleto. Debe contener la información más destacable. Lo ideal es que ocupe entre media y una cara de un folio A4. Comenzará por el propósito y principales objetivos de la memoria. Luego hablaremos sobre los aspectos más destacables de la metodología empleada, seguido de los resultados obtenidos. Por último se presentarán las conclusiones de forma condensada.

Debe tener un estilo claro y conciso, sin ambigüedades de ningún tipo. Además, al ser un resumen de todo el contenido, ni que decir tiene que deberá ser lo último que elaboraremos, y deberá mantener una absoluta fidelidad con el contenido de la memoria.

Palabras clave: Cuatro o cinco; Expresiones clave; Que resuman; Nuestro proyecto o; Investigación

Abstract

This section must contain the summary that we have written before in Spanish, but in English, as well as the keywords.

Keywords: Four or five; Key Expressions; Summarising; Our Project or; Research

Índice general

Índice de figuras

Índice de tablas

Índice de listados

1. Introducción

1.1. Contexto

En los últimos años, la inteligencia artificial (IA) se ha posicionado como una de las herramientas más útiles e interesantes de las que disponemos. El término inteligencia artificial fue acuñado por primera vez por John McCarthy en la Conferencia de Darmouth de 1956. Años después de que Alan Turing formulase la pregunta sobre si las máquinas podían pensar y plantease el famoso Test de Turing, varios científicos se reunieron con el objetivo de discutir acerca de la posiblidad de un artefacto de comportarse de manera inteligente. Se llegó a la conclusión de que todo aspecto del aprendizaje se puede describir con tanta precisión que resulte factible construir una máquina que los simule.

La IA se enfoca en crear sistemas que puedan realizar tareas que normalmente requerirían de inteligencia humana. Aunque ha tenido un reciente auge debido a los chatbots o los asistentes personales, presenta una gran cantidad de finalidades. Dentro de los usos que se le dan a la IA destacan la creación y el análisis de productos o la automatización de servicios, además de la optimización de procesos. Esta optimización se enfoca en encontrar la mejor solución posible a un problema dado dentro de un conjunto de opciones factibles. Los algoritmos de optimización y de personalización permiten, por ejemplo, crear aplicaciones que permiten organizar tareas de manera inteligente o termostatos que ajustan la calefacción automaticamente.

Este Proyecto Fin de Grado (PFG) se centrará en la resolución de un problema de optimización, la creación de un menú semanal de comidas personalizado que cumpla distintos objetivos, como el número de calorías diarias o la cantidad de macronutrientes ingeridos. Se hará uso de la computación evolutiva que, mediante algoritmos genéticos, diseñará una dieta equilibrada a partir de la selección, cruce y mutación de los distintos alimentos.

1.2. Motivación

La motivación para realizar este proyecto viene dada por el interés creciente en el área de la inteligencia artificial y cómo ha cambiado la manera de plantear los problemas respecto al pasado. En España, el sector TIC (Tecnologías de la Información y la Comunicación) cuenta con más del 40 % de empresas que usan estas herramientas para la automatización de flujos de trabajo, análisis de datos o para la gestión de la cadena de suministro. Buscan mejorar la precisión y la eficiencia a la hora de diseñar soluciones.

No obstante, no solo en sectores tecnológicos se hace uso de la IA. El médico o el alimentario también están incorporándola de manera gradual. Los diagnósticos de imágenes médicas o la agricultura de precisión son cada vez más comunes. También en la nutrición, relacionada con esto ámbitos, estas tecnologías permiten nuevas posibilidades.

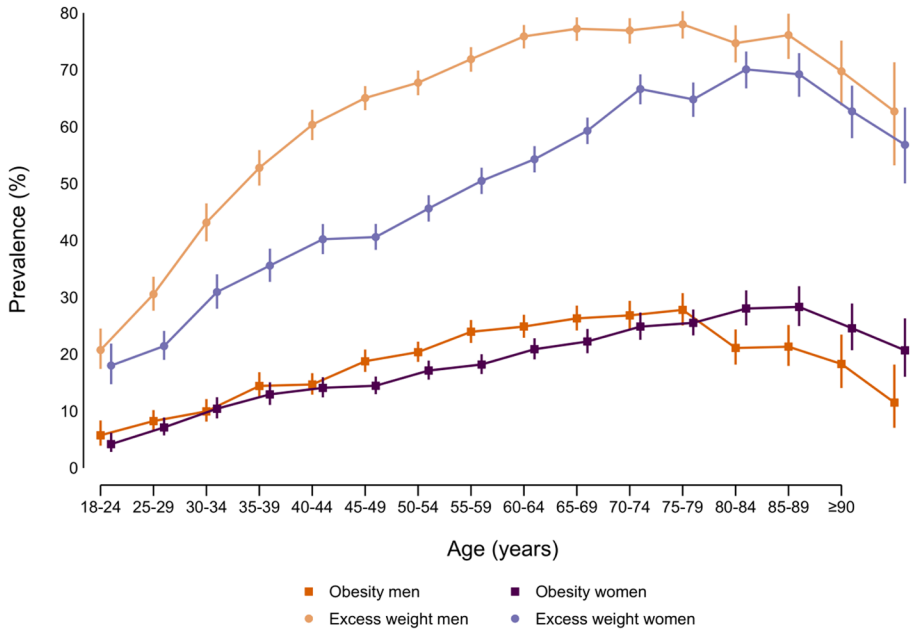


Figura 1.1. Prevalencia (%) de obesidad y exceso de peso por grupos de sexo y edad.

Según un estudio llevado a cabo por la Agencia Española de Seguridad Alimentaria y Nutrición (AESAN) y por el Instituto de Salud Carlos III (ISCIII), un 55,8 % de la población adulta española tiene exceso de peso y un 18,7 % padece obesidad. Esto trae consigo múltiples problemas de salud, como la apa-

rición de enfermedades crónicas o cardiovasculares, complicaciones respiratorias o dificultades al moverse.

Estos datos demuestran la importancia de una buena nutrición y hábitos alimenticios saludables. La planificación nutricional mediante algoritmos genéticos ayuda a comprender cómo se pueden mejorar estos hábitos a través de estas técnicas innovadoras, además de promover una vida sana y bienestar para todos, lo que se relaciona con los Objetivos de Desarrollo Sostenible (ODS), por lo que es un gran aliciente a la hora de realizar este PFG.

1.3. Justificación

En esta sección se deben explicar y argumentar las razones por las cuales se eligió el tema del proyecto, así como su importancia y relevancia. Algunos elementos clave que se pueden abordar en esta sección son:

1. **Relevancia del tema**: ¿Existe alguna necesidad o problema específico que tu proyecto pueda abordar?
2. **Justificación teórica**: Mención sobre qué teorías, enfoques o modelos existentes en la literatura respalden la importancia de abordar este tema.
3. **Brecha en el conocimiento**: ¿Qué aspectos no se han explorado lo suficiente o no han sido abordados en estudios previos? ¿Cómo puede el proyecto contribuir a cerrar esa brecha en el conocimiento?
4. **Contribución práctica**: Aplicaciones del proyecto y cómo pueden beneficiar a la comunidad académica, profesional o a la sociedad en general.

La sección no tiene por qué ser demasiado extensa, ni tiene por qué incluir (o limitarse) a los puntos anteriores, pero debe ser lo suficientemente clara y convincente para que los lectores comprendan por qué el proyecto es relevante y necesario.

1.4. Objetivos

El objetivo de este proyecto es la creación de un planning semanal de comidas mediante algoritmos genéticos. Experimentando con diferentes algoritmos se busca crear un menú que cumpla con distintos objetivos nutricionales, como la ingesta calórica diaria, a la vez que limitarse según las restricciones dadas. Los objetivos de este PFG son:

- Desarrollar un algoritmo evolutivo capaz de generar menús que cumplan con las restricciones y los objetivos nutricionales establecidos.
- Personalizar el algoritmo para la variación de las comidas según las necesidades específicas de los individuos.
- Experimentar con distintas configuraciones y variantes del algoritmo genético en busca de encontrar la mejor solución posible.
- Evaluar la sensibilidad y eficacia del algoritmo.
- Ejecutar pruebas que validen el algoritmo.
- Documentar los resultados obtenidos.

1.5. Estructura de la memoria

En este subapartado se explicará la estructura del documento.

En el capítulo 2, Estado del arte, se centra en dar un enfoque general al proyecto. Se da contexto sobre las áreas en las que se basa el proyecto, la inteligencia artificial y los algoritmos evolutivos, además de sobre el tema central del PFG, la planificación nutricional mediante algoritmos evolutivos.

En el capítulo 3, Marco teórico, se aporta la base teórica que en la que se desarrolla el proyecto. Se da una explicación general sobre la inteligencia artificial y se indaga en el algoritmo evolutivo y su funcionamiento.

En el capítulo 4, Desarrollo,

2. Estado del arte

2.1. Inteligencia artificial

El campo de la inteligencia artificial en la actualidad se encuentra en un estado de rápida evolución, con una infinidad de usos que se extiende por una gran variedad de sectores económicos y sociales. Su capacidad para analizar y procesar grandes cantidades de datos o para mejorar la eficiencia en distintas industrias han hecho que se trate de una tecnología en auge. El estudio de McKinsey & Company The state of AI in 2022 and a half decade in review estima que el 50 % de las empresas ya usan IA en sus tareas diarias, donde destacan la optimización de servicios, la creación de productos y el análisis del servicio al cliente.

El informe AI Index Report 2024 del Human-Centered AI Institute (HAI) de la Universidad de Stanford muestra que la IA ya supera a las habilidades humanas en ciertas tareas, como en la clasificación de imágenes, con una precisión del 97 % respecto al 95 % de los humanos, o en juegos, como el ajedrez, donde supera consistentemente a los jugadores humanos.

Aunque sin duda, el mayor crecimiento en los dos últimos años corresponde a las herramientas relacionas con la Inteligencia Artificial Generativa (IAG). Esta rama se centra en crear modelos capaces de generar contenido, como conversaciones, imágenes, videos o música. Aprende de datos ya existentes y produce nuevos con características similares. En este ámbito destaca la empresa OpenAI, que cuenta con ChatGPT, capaz de generar texto lógico y actuar como chatbot, o con Dall-E, que puede generar imágenes en base a la descripción que se entregue. The state of AI in 2023: Generative AI's breakout year, también de McKinsey & Company, muestra que el 79 % de los encuestados dice haber tenido almenos alguna exposición a la IAG, lo que indica que es la herramienta de IA que más rápido está captando el interés del público general.

Figura 2.1. Imagen generada con DALL-E 3.

Existe una gran cantidad de proyectos pioneros actualmente relacionados con el campo de la IA. Algunos de los más importantes actualmente son:

- AlphaFold. Este programa de Deepmind utiliza IA para predecir las estructuras de las proteínas. Ha permitido acelerar las investigaciones científicas que desembocan en la creación de nuevos medicamentos.
- Neuralink. Esta empresa trabaja en el desarrollo de un chip que permita comunicar directamente un cerebro humano con una computadora. Posibilitará realizar análisis neurológicos más precisos.
- Vehículos autónomos. Waymo controla una flota de taxis sin conductor que han estado operando por Estados Unidos.
- Medicina personalizada. Análisis de datos genéticos y médicos para diseñar tratatamientos que sean más efectivos para el paciente.

2.2. Algoritmos evolutivos

Los algoritmos evolutivos han tenido varios proyectos que han marcado el desarrollo de esta tecnología. Los primeros acercamientos tuvieron lugar en la década de 1960. Por un lado, Lawrence Fogel comenzó a explorar la programación evolutiva, centrándose en la evolución de autómatas finitos. Uno de los primeros intentos de aplicar principios evolutivos a la informática. Por otro lado, Rechenberg y Shwefel desarrollaron estrategias de evolución para problemas de optimización en ingeniería. Centradas en la selección y en la mutación, resultaron ser muy útiles para aplicarlas en la realidad.

Tras estos primeros pasos, John Holland fue fundamental para el desarrollo de los algoritmos genéticos. Su libro Adaptation in Natural and Artificial Systems del año 1975 es básico para entender el funcionamiento de los mismos. Su trabajo se centró en la idea de que la evolución biológica podía ser simulada y utilizada para resolver problemas complejos en computación. Hacen uso de los operadores como la selección, el cruce o la mutación para evolucionar una población candidata de individuos hacia una mejor solución. Cada solución es evaluada seguún una función de fitness, que mide qué tan buena es la solución al problema en cuestión.

Listado 2.1. Algoritmo Genético

```
INICIAR
    INICIALIZAR población
    EVALUAR fitness
    REPETIR HASTA CUMPLIR condición de parada:
        SELECCIONAR individuos
        CRUZAR padres
        MUTAR hijos
        EVALUAR individuos
        FORMAR nueva generación
    RETORNAR solución
FINALIZAR
```

Después de que Holland sentara las bases de los algoritmos genéticos, en

los años siguientes fueron surgiendo estudios que hicieron evolucionar la rama. Varios de los más importantes fueron de John Koza, quien introdujo la programación genética, técnica usada para desarrollar automáticamente programas que realicen una tarea definida por el usuario. Se optimiza una población de individuos (programas) respecto a una función de aptitud. Koza probó su viabilidad para resolver problemas de robótica o de optimización.

Tras él han seguido surgiendo nuevas técnicas dentro de los algoritmos evolutivos. Entre ellas se puede destacar el desarrollo de los Algoritmos genéticos híbridos (AGH), que combinan los algoritmos genéticos con otras técnicas de optimización para mejorar las soluciones a los problemas complejos. Otro avance importante es el de la Programación genética cartesiana (CGP), que sustituye los árboles de busca usados en la programación genética tradicional por grafos dirigidos, lo que es muy útil, por ejemplo, en el diseño de circuitos electrónicos.

A lo largo de los años se han desarrollado distintos proyectos pioneros que han hecho uso de los algoritmos genéticos. Algunos son:

- Space Technology 5 (ST5). Se usó algoritmos genéticos para la creación de una antena ultracompacta para la misión ST5 de la NASA, superando las expectativas de rendimiento.
- The EvoTanks Project. Se estudió el uso de algoritmos genéticos para desarrollar estrategias de combate para tanques autónomos en simulaciones.
- Sector financiero. Se puede utilizar para predecir movimientos del mercado. Además, a nievel doméstico, hay apps que lo usan para optimizar el método de compartir gastos entre distintos usuarios.
- Bioingeniería. Ayudan a modelar secuencias genéticas, que acelera los avances en medocina personalizada.

2.3. Planificación nutricional mediante algoritmos evolutivos

Los primeros acercamientos para intentar resolver problemas de optimización nutricional se dan en la década de 1940. George Stigler, en su artículo The Cost of subsistence, planteó el problema de encontrar la dieta de menor coste que cumpliese con unos objetivos nutricionales. Al no poseer ordenadores, utilizó técnicas manuales.

El problema fue formalmente resuelto dos años después por Jack Laderman usando programación lineal. Fue capaz de calcular la combinación de alimentos que cumpliese con los requisitos económicos y nutricionales. Con esto se demostró la utilidad de técnicas computacionales en tareas de optimización.

Tras las bases que sentó Holland, empezaron a aparecer distintos trabajos que hacían uso de los algoritmos genéticos para resolver problemas de optimización, incluyendo los de planificación nutricional. Application of genetic algorithms to diet optimization problems por

3. Marco teórico

3.1. Inteligencia artificial

La IA se puede definir como el estudio y diseño de agentes inteligentes, es decir, de sistemas que perciben su entorno y toman decisiones o acciones que maximizan sus posibilidades de éxito. Este campo se basa en disciplinas como la informática, la lógica o la neurociencia, que contribuyen a la simulación de capacidades cognitivas humanas en máquinas.

Los agentes inteligentes, que son la base de este campo, se clasifican según la capacidad de reconocer y actuar en su entorno. Podemos encontrar desde agentes simples, que responden directamente a estímulos, hasta agentes basados en utilidad, que evalúan si los resultados de sus acciones son satisfactorios. Es fundamental la racionalidad, la habilidad de realizar elecciones óptimas que maximicen la posibilidad de alcanzar objetivos. Utilizando distintas herramientas de lógica, los agentes inteligentes pueden formular y modificar conocimientos, deduciendo nueva información.

El razonamiento lógico es también fundamental para el desarrollo de algoritmos evolutivos, donde las decisiones sobre selección, cruzamiento o reproducción se basan en decisiones lógicas. Subclase de los métodos de aprendizaje automático inspirados en los procesos biológicos de evolución, aplican estos principios para desarrollar soluciones a complejos problemas.

A medida que la tecnología evoluciona, también lo hace la capacidad de la IA para aprender y adaptarse, al igual que los algoritmos genéticos mejoran a lo largo de las generaciones. Su implementación es cada vez mayor en campos como el desarrollo de software o la robótica, donde las mejoras en la optimización y en la resolución de los problemas conducen a mejoras significativas en la eficiencia y la funcionalidad.

3.2. Algoritmos genéticos

Un algoritmo se puede definir como un procedimiento computacional bien definido que toma un valor, o un conjunto de valores, como entrada y produce un valor, o un conjunto de valores, como salida. Se trata de un conjunto de instrucciones que permiten realizar una actividad mediante sucesivos pasos.

Un algoritmo evolutivo es un tipo de algoritmo que proviene de la computación evolutiva. Esta rama de la IA emplea principios inspirados en la evolución biológica para la resolución de problemas. Teoría propuesta por Charles Darwin en 1859, expone que en la naturaleza, en un entorno dado que puede albergar un número limitado de individuos, la selección natural favorecería a aquellos que poseyeran características que les permitiesen adaptarse mejor al medio, teniendo una mayor posibilidad de sobrevivir y reproducirse. A lo largo del tiempo, las características ventajosas se propagan a través de las generaciones, mientras que aquellas menos favorables tienden a desaparecer. Por lo tanto, estas poblaciones irán evolucionado y adaptándose gradualmente al entorno.

En 1975 John Holland propone imitar los procesos biológicos naturales que rigen la selección natural usando ordenadores, lo que sería el principio de los algoritmos genéticos (AGs). En un AG, las soluciones son modeladas como individuos o cromosomas, que generalmente se representan como cadenas de bits, aunque también se puede usar otro tipo de cadenas. Estas soluciones son evaluadas por una función de aptitud o fitness, y las más adecuadas son selecciondas para reproducirse mediante cruzamiento y mutación, procesos que mezclan y alteran aleatoriamente los cromosomas para generar diversidad. Los descendientes resultantes forman nuevas generaciones que vuelven a ser evaluadas, creando un ciclo que se repite hasta cumplir alguna condición de parada, como pudiera ser un número limitado de generaciones o que se alcance la solución deseada.

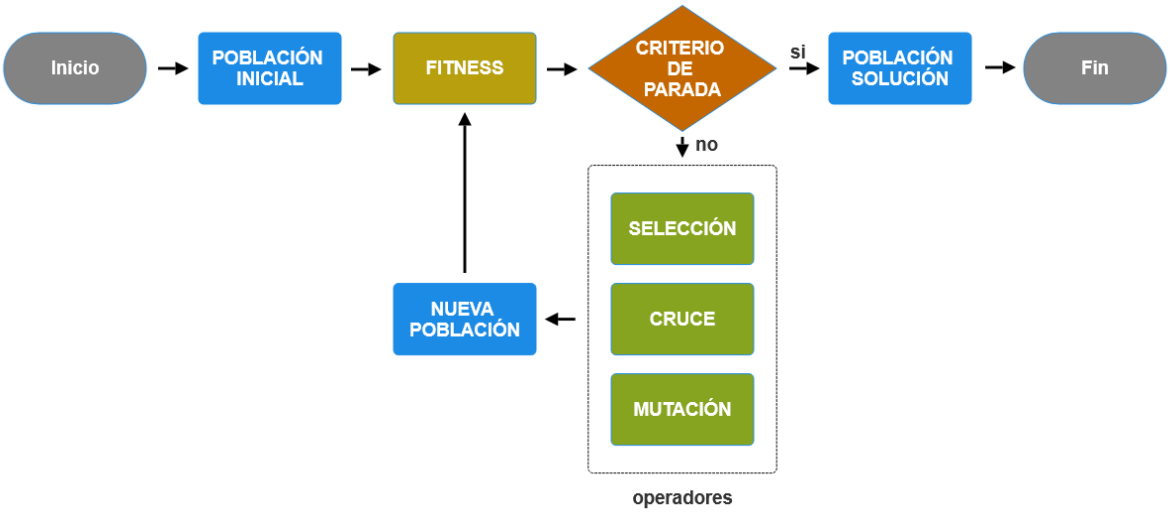


Figura 3.1. Algoritmo genético simple.

Explicado el concepto general, se va a desglosar cada una de las fases de la figura 3.1.

3.2.1. Población

Antes de explicar la generación de una población inicial, es necesario conocer algunos conceptos que son usados para representar y entender las soluciones. Son términos que son usados en la biología y en el campo de la genética.

- **Gen.** Es la unidad básica de información en un cromosoma. En una cadena de bits que representa una solución, cada bit puede considerarse un gen. En el caso que se trata en este PFG, el menú semanal, cada tipo de comida se podría considerar un gen. Por ejemplo, un gen sería bebida, plato principal o postre.
- **Alelo.** Es la forma específica o valor que puede tomar un gen. Siguiendo el ejemplo de bebida, posibles alelos serían agua, té o cerveza.
- **Cromosoma.** Es una colección de genes y representa una solución completa al problema de optimización. El cromosoma es la cadena de bits completa. En nuestro caso, la lista completa de alimentos seleccionados de un día determinado es el cromosoma.

- **Fenotipo.** Es la manifestación real de la solución codificada. En el PFG sería cómo se preparan y sirven estos alimentos en la realidad.

La generación de una población inicial implica crear un conjunto de soluciones candidatas. Generalmente, los individuos son seleccionados aleatoriamente dentro de los límites definidos en cada problema, asegurando que todas las áreas del espacio de búsqueda puedan ser exploradas. También existen métodos alternativos de generación que aplican ciertas restricciones para formar soluciones iniciales más prometedoras. Tomando de ejemplo el menú semanal, el espacio de búsqueda comprendería la base de datos en la que aparecen todos los alimentos con sus respectivas calorias y macronutrientes.

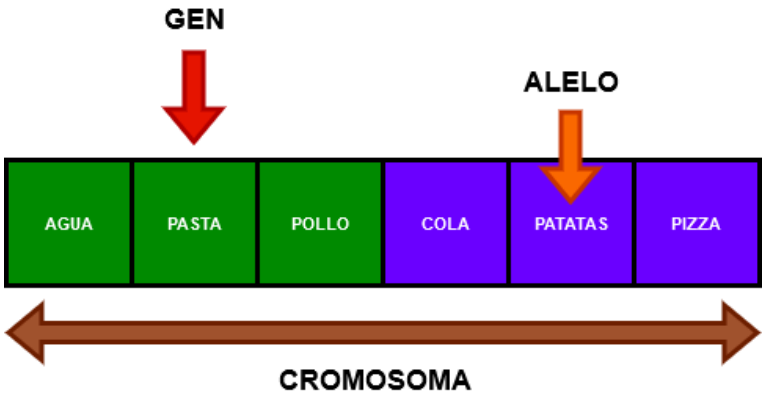


Figura 3.2. Estructura de un cromosoma.

3.2.2. Evaluación

Se mide lo bueno que es un individuo para nuestros propósitos (la calidad del individuo). Por lo tanto, lo más importante es definir una función de fitness correcta y representativa del problema a evaluar.

Según vayan pasando generaciones, la población inicial irá evolucionando hacia poblaciones candidatas que presentarán una mejor aptitud. Si la población ha alcanzado el objetivo, la condición de parada se activará, convirtiendo el conjunto de individuos candidatos en la población solución. En caso de no alcanzarlo, seguirá evolucionando hacia otra población candidata distinta.

3.2.3. Operadores

Son los procesos que se aplican a poblaciones de individuos para desarrollar generaciones futuras. Sirven para la exploración del espacio de soluciones y para la mejora de las poblaciones a través de las generaciones. Se busca que el conjunto de individuos presente una alta diversidad, ya que si es baja se corre el riego de caer en mínimos locales, lo que no permitiría explorar el espacio de soluciones ampliamente.

Selección

El primero de los operadores básicos. Elige un individuo para reproducirlo. Si bien se puede seleccionar de manera equiprobable, existen métodos basados en la aptitud del individuo. Estos son algunos:

- **Método estándar (rueda de la fortuna).** Asigna a cada individuo una probabilidad proporcional a su fitness, por lo que los más aptos tienen mayor probabilidad de reproducirse. Existe un derivado de este método en el que se normalizan las probabilidades, lo que favorece aún más a los adaptados.
- **Método del rango.** Se ordenan los individuos según su aptitud, y la probabilidad de selección se asigna según este ránking.
- **Selección por torneo.** Se escoge aleatoriamente un subconjunto de individuos de la población, y el mejor de este es elegido.

La selección puede incorporar un mecanismo de elitismo, que escoge los mejores individuos de una generación para que se mantengan en la siguiente. Esto ayuda a que la calidad del mejor individuo de una generación sea siempre igual o superior a su equivalente de la generación anterior, consiguiendo una progresión constante hacia la solución óptima, es decir, una mayor convergencia.

También se debe buscar un equilibrio con la diversidad. Si no se explora el espacio de búsqueda en amplitud, se podría caer en mínimos locales si la población es muy parecida entre sí, lo que no permitiría encontrar buenas soluciones.

Cruce

Se combina el material genético de dos individuos, padres, para producir descendencia, hijos. Es decir, se utiliza para intercambiar características de dos soluciones parentales con el objetivo de generar nuevas soluciones. Se aplica con probabilidad P_c . Al promover la mezcla de genes, ayuda a aumentar la diversidad y la convergencia, debido a la generación de nuevos descendientes con características deseables. Para problemas donde las soluciones están representadas como cadena de bits o enteros, se usan distintos métodos:

- **Cruce de un punto.** Se selecciona aleatoriamente una posición en el cromosoma. Todo lo que está antes de este punto se intercambia con todo lo que está después en la otra cadena, y viceversa, para producir dos descendientes.
- **Cruce de dos puntos.** Similar al cruce de un punto, pero se seleccionan dos puntos de corte. Las cadenas de genes entre estos dos puntos se intercambian entre los dos padres.
- **Cruce uniforme.** Cada gen tiene una probabilidad igual de ser elegido de uno de los dos padres.

Mutación

Último de los operadores básicos. Se recorre toda la cadena, mutando cada gen con probabilidad P_m , es decir, eligiendo un nuevo valor mediante una elección equiprobable sobre el alfabeto. La mutación aumenta la diversidad y ayuda a no caer en mínimos locales. Algunos de los métodos usados son:

- **Mutación uniforme.** Cada gen puede cambiar a otro valor con probabilidad P_m.
- **Mutación por intercambio.** Dos genes aleatorios en el cromosoma son seleccionados y sus posiciones se intercambian.

Figura 3.3. Operadores.

4. Desarrollo

5. Configuración de la memoria

La estructura de esta plantilla está dividida en varios ficheros. Aunque puede parecer complicado, en realidad no es tanto:

- `./appendices/`: Los fuentes de los capítulos de apéndices, vamos, los que van al final y que se usan como adenda a la memoria como tal.
- `./chapters/`: Los fuentes de los capítulos que forman parte del cuerpo de la memoria.
- `./figures/`: Las figuras (imágenes, diagramas) que se usarán a lo largo de todo el documento.
- `./frontmatter/`: Los fuentes de todo aquello que se incluye antes del cuerpo de la memoria. Por ahora, todas las entradas del glosario.
- `./sources`: Ficheros con fuentes que se incluyen dentro de listados de fuentes del documento.
- `./references.bib`: Los fuentes en BibTeX de la bibliografía que se referencia en la memoria en la memoria.
- `./report.tex`: El fichero principal a partir del cual se compila todo el proyecto.
- `./upm-report`: El directorio que tiene todo el contenido que hace que la memoria se vea así. Si tocas aquí, que sea con mimo y cariño, porque es muy fácil cargárselo todo.

5.1. ¿Cómo empiezo a escribir la memoria?

Por el principio, es decir, con el fichero `report.tex`. Veamos la primera línea del fichero (listado 5.1).

Listado 5.1. Primeras líneas del fichero `report.tex`

```
\documentclass[%
    school=etsisi,%
    type=pfg,%
    degree=61IW,%
    authorsex=m,%
    directorsex=m,%
]{upm-report}
```

En este punto es donde se configura gran parte de la plantilla. Los parámetros y sus opciones son las siguientes:

- `school`: La escuela a la que pertenece el estudiante. La idea de la plantilla es que se use a lo largo de todas las escuelas de la UPM, y que cada una de ellas tenga su propia configuración. La escuela determinará, entre otras cosas, direcciones y colores principales. Las opciones se describen en el Apéndice A.
- `type`: El tipo de memoria. Modifica algunos textos, incluida la portada. Puede tomar los valores `pfg` (Proyecto Fin de Grado) y `pfm` (Proyecto Fin de Máster).
- `degree`: El grado al que aspira el estudiante. De momento sólo están definidos los grados que se imparten en la ETSISI (también en el Apéndice A).
- `authorsex`: Puede ser `m` (masculino) o `f` (femenino), y sirve para modificar algunos textos relacionados con el sexo del estudiante.
- `directorsex`: Similar al parámetro `authorsex`, pero para el director del proyecto.

Tras esta configuración, se incluye el fichero de referencias bibliográficas (listado 5.2):

Listado 5.2. Inclusión del fichero de referencias bibliográficas `references.bib`

```
\addbibresource{references.bib}
```

El tema de las referencias bibliográficas se explica en el Sección 6.9 del Capítulo 6. En principio no habría que tocar nada (bueno sí, añadir las fuentes y referenciarlas), pero si las referencias se tienen en otro fichero, bastaría con cambiar el nombre al de dicho fichero.

Los cinco siguientes comandos, como se ve en el listado 5.3, indican el título del proyecto, el nombre del autor y su entrada en la bibliografía, y el nombre del director y su entrada en la bibliografía.

Listado 5.3. Configurando autor, título del proyecto y director

```
\title{Planificación nutricional mediante algoritmos evolutivos
   }
\author{Javier Quesada Pajares}
\director{Cristian Ramírez Atencia}

\abstract{spanish}{
```

El único misterio es eso de las entradas bibliográficas para el autor y el director. No es más que los nombres que aparecen más adelante cuando se indica «cómo citar el proyecto». El día que aprenda cómo hacerlo automáticamente, será una configuración que desaparezca[1].

Tras ello, empieza el primer contenido de verdad: **resumen** y **abstract**, cada uno con sus palabras clave asociadas. Ambos dos son obligatorios y se añaden con la macro `\abstract`, donde se especificarán el idioma (`spanish` o `english`) y el contenido. De la misma manera, las palabras clave se añaden con la macro `\keywords`. Ni que decir tiene que ambos deben tener el mismo contenido, uno en español y el otro en inglés. Y además es obligatorio (según la normativa de la ETSISI).

Existe la opción de añadir agradecimientos a través de la macro `\acknowledgements`

[1] Bueno, y si tú, queridísimo lector o lectora sabes cómo hacerlo, hazme un *pull request* al repositorio de la plantilla: https://github.com/blazaid/UPM-Report-Template.

. Es opcional, así que si no se pone no se renderiza en el documento final, pero es algo bonito y a las abuelas les encanta aparecer ahí. Y las abuelas son de lo más bonito que existe en este mundo, así que cuidadlas.

Y ahora sí, se empieza con el grueso del documento. Tras incluir el glosario, del que se hablará en la sección 6.6 del 6, se comenzarán a incluir uno tras otro todos los capítulos de los que se compone nuestra memoria, tal y como se muestra en el listado 5.4.

Listado 5.4. Cómo se incluyen los capítulos y los apéndices

```
\input{chapters/estado-arte}
\input{chapters/marco-teorico}
\input{chapters/desarrollo}
\input{chapters/configuracion}
\input{chapters/componentes}
\input{chapters/licencia}

\appendix

\input{appendices/escuelas-y-titulos}
\input{appendices/ampliar}
\input{appendices/paquetes}
```

La macro `\appendix` del medio indica a partir de qué punto se añaden los apéndices. No son obligatorios, ni mucho menos, pero en algunos PFGs y Proyectos Fin de Máster (PFMs) se suelen incluir para dar información adicional de contexto que no es el objetivo de la memoria, pero sí interesante para complementar. Por ejemplo, en un PFM para el estudio del comportamiento de conductores al volante, uno de los apéndices podría ser cada uno de los formularios que se le ofrecieron para rellenar a cada uno de los conductores de dicho estudio.

Y ya estaría todo. Resumiendo, hay que configurar la plantilla, poner el autor, título y director del proyecto e incluir los capítulos y apéndices que queramos.

5.2. ¿Cómo estructurar la memoria?

La respuesta rápida es «como buenamente quieras/puedas». En realidad la estructura de la memoria va a depender del tipo de trabajo desarrollado, pero con carácter general, los trabajos suelen seguir ciertas estructuras.

Un PFG es un trabajo cuyo propósito es demostrar que se han llegado a adquirir las competencias asociadas con la titulación cursada. Con esto queremos decir que, a diferencia de otros tipos de trabajo académico, en éste no es necesario realizar aportaciones originales al estado de la cuestión.

Una estructura típica es la siguiente:

1. Resumen
2. Introducción
3. Estado de la cuestión
4. Metodología
5. Resultados y Discusión
6. Conclusiones
7. Apéndices
8. Referencias bibliográficas
9. Glosario

Un PFM, a diferencia de un PFG trata de profundizar más en un campo concreto de una disciplina, por lo que tiene a ser más extenso y mucho más específico.

En términos generales, la estructura es similar. Sin embargo es de esperar que el nivel de exigencia sea mayor, ya que el estudiante que lo realiza debe demostrar que es un titulado superior. Esto se nota más en la fase de documentación, ya que al tratar de profundizar en un tema más específico, el trabajo de contextualizar y argumentar es más tedioso.

Se pueden identificar dos tipos de proyectos diferentes, aquellos que podríamos catalogar de *profesionales*, con enfoque a la innovación o mejora en un área profesional concreta, y aquellos *de investigación*, más enfocados a la búsqueda de nuevo conocimiento en el área, y que suelen ser el comienzo de la carrera investigadora.

6. Componentes de la plantilla

En este capítulo hablaremos de los componentes principales con los que trabajaremos en nuestra memoria.

6.1. Columnas

TBD

6.2. Ecuaciones

La facilidad de composición de ecuaciones es una de las cosas que más atrae de LaTeX a muchos autores. LaTeXmantiene dos renderizadores diferentes, uno para el texto y otro para las ecuaciones, denominados modo párrafo y modo matemático[1]. El modo párrafo es el modo por defecto y no se le llama explícitamente. Al modo matemático, sin embargo, se le invoca de varias maneras diferentes.

6.2.1. Modo en párrafo

La forma más común es la forma “en línea”, donde el texto para el modo matemático se encierra entre dos signos $. Por ejemplo, veamos la frase del listado 6.1.

Listado 6.1. Ejemplo de inserción de fórmulas en linea

[1] Existe un tercer modo, denominado *LR mode* o *left-to-right mode*, raramente utilizado y que no trataremos aquí

```
El pequeño teorema de Fermat dice que si $p$ es un número primo
   , entonces, para cada número natural $a$, con $a>0$, $a^p \
   equiv a (\mod p)$
```

La frase quedaría como sigue:

El pequeño teorema de Fermat dice que si p es un número primo, entonces, para cada número natural a, con $a > 0$, $a^p \equiv a(mod p)$

6.2.2. Ecuaciones en bloque

Cuando en lugar de poner una ecuación dentro de un párrafo existente la queremos insertar en su propio espacio independiente hacemos uso de los entorno `equation` o `align`, dependiendo de si queremos una o más ecuaciones en el bloque, respectivamente. Por ejemplo, en el caso de una única ecuación, sería similar al ejemplo siguiente:

```
\begin{equation}
  \int_a^b f'(x) \, dx=f(b)-f(a
     )
\end{equation}
```

$$\int_a^b f'(x)\,dx = f(b) - f(a) \tag{6.1}$$

Sin embargo, en el caso de que quisiésemos más de una ecuación en el mismo bloque haríamos uso del carácter & para indicar en qué punto se alinean las ecuaciones; por ejemplo:

```
\begin{align}
    u  &= \arctan x \\
    du &= \frac{1}{1 + x^2}dx
\end{align}
```

$$u = \arctan x \tag{6.2}$$
$$du = \frac{1}{1+x^2}dx \tag{6.3}$$

Por último, si no tenemos por qué referenciarlas en el texto, podemos hacer

uso de los entornos `equation*` y `align*`

```
\begin{equation*}
  \int_a^b f'(x) \, dx=f(b)-f(a
     )
\end{equation*}
```

$$\int_a^b f'(x)\,dx = f(b) - f(a)$$

6.3. Elementos flotantes

Vamos a hablar un poco de los elementos denominados «flotantes» o *floats*. Éstos son bloques de contenido que «flotan» por la página hasta que LaTeX los coloca donde considera a través de ciertos algoritmos.

Lo general es **declarar el elemento flotante inmediatamente después del párrafo donde se ha referenciado** (las referencias cruzadas las vemos en la Sección 6.8 de este capítulo), y después dejar que LaTeX elija el mejor sitio. Y si después de haber escrito todo el documento hay algo que no cuadre, pues ahí modificarlo.

«¿Y si no hago referencia a un elemento flotante, dónde lo pongo?» Bueno, pues en general no se pone. Si algo no es nombrado, suele ser porque no aporta información, y si no la aporta, pues para qué lo vamos a meter. Ojo, que lo mismo existe algún caso donde sí tiene sentido, pero es muy raro.

Aunque más adelante veremos los diferentes tipos de *floats*, en caso de que queramos modificar su comportamiento todos tienen los especificadores de posición indicados en la tabla 6.1:

6.3.1. Código fuente

Para la gestión de los listados de código fuente se utiliza el paquete `listings`. El estilo usado es una modificación[2] de Solarized desarrollado por Ethan Schoonover.

[2] En realidad la modificación es cambiar el fondo de crema a blanco

Tabla 6.1. Opciones para los elementos flotantes de LaTeX

Especificador	Acción
H	Colocar exactamente en el sitio indicado
h	Colocar aproximadamente en el sitio indicado
t	Colocar al comienzo de la página
b	Colocar al final de la página
p	Colocar en una página exclusiva para elementos “flotantes”
!	Forzar las opciones obviando los mecanismos internos de LaTeX

Existen muchas formas diferentes de incluir listados de código en una memoria. Aquí introducimos los más comunes.

Código dentro del propio párrafo

TBD

Bloques de código

Insertar código en párrafos no es tan común como insertar bloques enteros. Para ello haremos uso del entorno `lstlisting`. Por ejemplo:

```
\begin{lstlisting}
from collections import Counter

def is_anagram(w1, w2):
    return Counter(w1) == Counter(w2)
\end{lstlisting}
```

Nos daría el siguiente resultado:

```
from collections import Counter
```

```
def is_anagram(w1, w2):
    return Counter(w1) == Counter(w2)
```

Una cosa que hay que tener en cuenta es que dentro de un entorno `lstlisting` se ignoran todos los comandos de LaTeX [3] y el texto se imprime tal y como se ha introducido. Esto incluye los tabuladores y espacios de principio de línea.

Al igual que en el código de párrafo, también podemos especificar en qué lenguaje está escrito el código para que se resalten en éste las palabras reservadas. Por ejemplo:

```
\begin{lstlisting}[language=python]
from collections import Counter

def is_anagram(w1, w2):
    return Counter(w1) == Counter(w2)
\end{lstlisting}
```

Nos daría como resultado el siguiente bloque de código:

```
from collections import Counter

def is_anagram(w1, w2):
    return Counter(w1) == Counter(w2)
```

Código directamente desde fichero

Esta forma es muy común, ya que se usa tanto para hacer referencia al código fuente de la aplicación directamente, como a código separado en ficheros para mantener el tamaño de la memoria manejable.

Suponiendo que tenemos el fichero `sources/snippets.py`, para incluirlo en-

[3] En realidad no todos, y si no mira en estos fuentes cómo hemos metido el fin de bloque `lstlisting` dentro del propio bloque.

tero basta con usar el comando `lstinputlisting`:

```
\lstinputlisting[language=python]{sources/snippets.py}
```

Con este comando conseguiríamos el siguiente resultado:

```
def all_unique(l):
    return len(l) == len(set(l))

def is_palindrome(l):
    return l == l[::-1]
```

En caso de que se desease importar sólo parte del fichero, se pueden indicar las filas que delimitan el trozo de código. Por ejemplo:

```
\lstinputlisting[
    language=python,
    firstline=4,
    lastline=5
]{sources/snippets.py}
```

Esto haría que sólo se imprimiesen las filas 4 y 5, correspondientes a la segunda función:

```
def is_palindrome(l):
    return l == l[::-1]
```

Ambas opciones son opcionales, y los valores por defecto de `firstline` y `lastline` serán el principio y el final del fichero respctivamente.

Etiquetando bloques de código

Al igual que con el resto de bloques *float* (e.g. figuras o tablas), se pueden[4] (**deben**) añadir pies de texto a los bloques de código, lo cual hace más legible su cometido.

Para ello basta con añadir el argumento `caption` a las opciones del bloque. Por ejemplo:

```
\begin{lstlisting}[language=python, caption=Función para
   determinar cuando una palabra \textt{w1} es anagrama de otra
    palabra \textt{w2}]
from collections import Counter

def is_anagram(w1, w2):
    return Counter(w1) == Counter(w2)
\end{lstlisting}
```

Nos daría como resultado el siguiente bloque de código:

Listado 6.2. Función para determinar cuando una palabra `w1` es anagrama de otra palabra `w2`

```
from collections import Counter

def is_anagram(w1, w2):
    return Counter(w1) == Counter(w2)
```

6.3.2. Figuras

El código siguiente (listado 6.3) renderizará una imagen.

[4] Pongo *pueden* porque es opcional, pero en realidad se **deben** poner, porque si no los listados de fuentes quedan horribles, como el de esta plantilla por ejemplo (échale un vistazo si no lo has hecho antes).

Listado 6.3. Inserción de una figura

```
\begin{figure}[H]
    \centering
    \includegraphics[width=0.25\textwidth]{figures/vault-boy.
       png}
    \caption{\label{fig:img-vault-boy} Vault Boy approves that}
\end{figure}
```

La sintaxis es bastante autoexplicativa. El entorno `figure` es el que delimita el contenido de la figura. El comando `centering` determina que se tiene que centrar . Luego, el comando `caption` determina el pie de imagen, el cual además incluye una etiqueta (comando `label`) que sirve para referenciar. Por último, se incluye la imagen con el comando `includegraphics`.

En definitiva, la imagen (figura 6.1) se mostrará con un ancho igual al 25 % del ancho que ocupa el texto, centrada, con un pie de foto y una etiqueta para referenciar.

Figura 6.1. Vault Boy approves that

El comando `includegraphics` puede importar los formatos típicos de imagen, como jpeg, png o pdf. También admite una serie de opciones como rotación, alto, ancho (éste le hemos especificado con `width`), etcétera. ¡Ojo! Siempre que se pueda, hay que intentar insertar imágenes vectoriales. De esta manera, se mantiene la calidad de la imagen. Si no, puede ocurrir que se pixele y no quede nada bien.

Subfiguras

¿Y qué pasa cuando queremos incluir múltiples imágenes dentro de una figura? Bueno, pues aquí hay que usar el entorno `subfigure`. En el listado 6.4 vemos un ejemplo de cómo se manejan.

Listado 6.4. Inserción de varias subfiguras

```
\begin{figure}[H]
  \centering
  \begin{subfigure}{.3\textwidth}
    \includegraphics[width=\linewidth]{figures/vault-boy.png}
    \caption{\label{fig:subfigure-1}Vault Boy 1}
  \end{subfigure}%
  \begin{subfigure}{.3\textwidth}
    \includegraphics[width=\textwidth]{figures/vault-boy.png}
    \caption{Vault Boy 2}
  \end{subfigure}%
  \begin{subfigure}{.3\textwidth}
    \includegraphics[width=\textwidth]{figures/vault-boy.png}
    \caption{Vault Boy 3}
  \end{subfigure}
  \caption{\label{fig:subfigures}Todos los Vault Boy}
\end{figure}
```

En realidad cada subfigura se trata como una figura normal, pero en relación con el *float* contenedor. Cuando a una subfigura se le especifica un ancho, se le está diciendo al compilador de qué ancho es esa subfigura en concreto (en nuestro caso 0.3 veces el ancho de la línea). Sin embargo, a la imagen se le da un ancho total de `linewidth`, porque al estar dentro de su espacio de subfigura, el ancho ha cambiado. El resultado es el que se observa en la figura 6.2. Por cierto, también podemos referenciar a los pies de las subfiguras (e.g. Así: 6.2a).

(a) Vault Boy 1 (b) Vault Boy 2 (c) Vault Boy 3

Figura 6.2. Todos los Vault Boy

6.3.3. Tablas

Las tablas (en realidad «cuadros») son una forma muy eficaz de presentar información. En los resultados de casi cualquier trabajo existen cuadros de algún tipo para que los datos se comprendan de un único vistazo (o para que al menos sea más fácil identificarlos.

¿Por qué "cuadro" en lugar de "tabla"?
Tal y como se indica en el FAQ de CervanTex, *table* (inglés) y *tabla* (español) son falsos amigos; el inglés *table* tiene un sentido más general que el español *tabla*, cuyo uso es únicamente para aquellos cuadros dedicados a la disposición de números (e.g. tabla de multiplicar o tabla de logaritmos).

Sin embargo, las tablas suelen ser bastante complicadas en LaTeX, por lo menos para la gente que empieza. Para no escribir demasiado, la respuesta para casi toda maquetación de tabla está en https://www.tablesgenerator.com/. En serio, no perdáis el tiempo si no es estrictamente necesario. La maquetáis visualmente, la generáis (con estilo *booktabs*) y a correr.

6.4. Enlaces de hipertexto

TBD

6.5. Fórmulas matemáticas

TBD

6.6. Glosario

El glosario de una memoria es el lugar donde se encuentran los términos que se usan a lo largo del documento y que se considera que requieren una aclaración. En esta plantilla, en el momento que generemos un término, se creará un capítulo al final de la memoria con el listado de todos aquellos términos definidos.

Para gestionar el glosario se hace uso del paquete `glossaries` el cual es relativamente complejo de configurar. También su documentación es muy extensa[5], así que en esta sección hablaremos únicamente de lo esencial.

6.6.1. ¿Cuándo y cómo especificar términos?

La regla general del «cuándo» es una vez terminada la memoria. En ese punto, seremos conscientes de qué términos son los más interesantes para incluir en el glosario. En ese punto deberemos ir término por termino sustituyéndolo por la entrada del glosario para que el proceso automático se encargue de la indexación y numeración de páginas.

El «cómo» se refiere a de qué manera escribirlos. La regla general en el cas-

[5] Pero aún así es el sitio donde ir a buscar información. El paquete, junto con su documentación está disponible en la dirección https://www.ctan.org/pkg/glossaries.

tellano (y hasta donde el autor de la plantilla sabe, en cualquier idioma) es de la manera en la que aparecería en medio del texto. Es decir, si la palabra se escribe generalmente en minúscula (e.g. *El jugador blandía un hacha de batalla*) se deberá incluir dentro del glosario en minúscula, mientras que si se escribe generalmente en mayúscula (e.g. *Encontró el Arco de la Perdición*) irá en mayúscula.

6.6.2. Definiendo los términos del glosario

Las entradas se escribirán dentro del fichero `frontmatter/glossary.tex`. La forma estándar de definir un término es la que se muestra en el listado 6.5.

Listado 6.5. Código para crear una entrada en el glosario

```
\newglossaryentry{hacha-batalla}{
    name={hacha de batalla},
    description={Herramienta antigua utilizada en combate}
}
```

Luego, dentro del texto, podremos hacer referencia a dichas entradas con los comandos que se muestran en la tabla 6.2.

Tabla 6.2. Comandos para incluir términos del glosario en el texto de la memoria

Comando	Ejemplo con la clave **`hacha-batalla`**
`\gls`	hacha de batalla
`\Gls`	Hacha de batalla
`\glspl`	hacha de batallas
`\Glslp`	Hacha de batallas

Como los plurales los gestiona automáticamente, puede ser que queramos, como en este caso, modificar el plural de nuestro término. Para ello debemos añadir la opción `plural` a la entrada para especificar cómo es el plural de la entrada, como se muestra en el listado 6.6.

Listado 6.6. Especificando el plural para un término del glosario

```
\newglossaryentry{python}{
    name={Python},
    plural={Pythonacos},
    description={El mejor lenguaje de programación}
}
```

Así, el plural de la clave `python` descrita quedaría como Pythonacos, en lugar del valor por defecto que sería *Pythons*.

Un caso particular de términos del glosario son las siglas y los acrónimos. No vamos a entrar en detalle aquí[6] sino que vamos a introducir las siglas como caso especial de entrada de glosario. Cuando tengamos una sigla, la crearemos en el glosario como se muestra en el listado 6.7.

Listado 6.7. Entrada genérica de una sigla o acrónimo en el glosario

```
\newacronym[
    description={Proyecto Fin de Grado. Proyecto a realizar al
       final de una titulación de Grado},
    longplural={Proyectos Fin de Grado}
    ]{pfg}{PFG}{Proyecto Fin de Grado}
```

En el ejemplo se puede ver que hay dos entradas, `longplural` y `description` que son opcionales. La primera es la equivalente a `plural` de `newglossaryentry`, y no necesita más explicación.

La segunda, `description` suele utilizarse para acrónimos, cuando necesitamos describir la entrada. Cuidado en este caso porque si hace referencia a varias palabras estas se deberían incluir dentro de la descripción (como en el ejemplo, «Proyecto Fin de Grado»).

La regla general de los acrónimos y las siglas es que la primera vez que aparecen en el texto, deben aparecer con el nombre completo mientras que el

[6] Pero recomendamos visitar https://www.fundeu.es/recomendacion/siglas-y-acronimos-claves-de-redaccion/ y darle una leída porque es interesante.

resto de veces pueden aparecer indistintamente como sigla o forma larga. De esto se encarga automáticamente el comando `gls`. Es decir, si tenemos la sigla `special`, la primera vez que incluyamos la sigla con `\gls{special}` saldrá *Strenght, Perception, Endurance, Charisma, Intelligence, Agility & Luck* (SPECIAL) mientras que el resto de veces que la incluyamos se verá simplemente SPECIAL.

Con los acrónimos se incluyen comandos adicionales para controlar su presentación. Estos son los mostrados en la tabla 6.3

Tabla 6.3. Comandos específicos para controlar la presentación de acrónimos

Comando	Ejemplo con la clave `rpg`
`\acrshort`	RPG
`\acrshortpl`	RPG
`\acrlong`	*Role-Playing Game*
`\Acrlong`	*Role-Playing Game*
`\acrlongpl`	*Role-Playing Games*
`\Acrlongpl`	*Role-Playing Games*
`\acrfull`	*Role-Playing Game* (RPG)
`\Acrfull`	*Role-Playing Game* (RPG)
`\acrfullpl`	*Role-Playing Games* (RPG)
`\Acrfullpl`	*Role-Playing Games* (RPG)

Por cierto, en castellano las siglas **no incluyen la «s» al final**, así que no deberíamos usar los comandos que terminan en `pl`. Por eso la definición que se ha hecho de la sigla `rpg` es la mostrada en la figura 6.8.

Listado 6.8. Entrada de `rpg` en `glossaries.tex`

```
\newacronym[
    description={Role-Playing Game. Juego de rol},
    shortplural={RPG}
    ]{rpg}{RPG}{\textit{Role-Playing Game}}
```

6.7. Notas

TBD

6.8. Referencias cruzadas

Las etiquetas (*label*) son una herramienta muy útil en el proceso de composición tipográfica. Se puede pensar en ellas como punteros a zonas de interés del documento, de tal manera que se les pueda referenciar sin necesidad de conocer su posición final en la composición.

Por ejemplo, lo normal es que en un libro, a la hora de referenciar una figura, aparezca una frase del estilo "[...] como muestra la Figura 3 [...]". Lo que es bastante raro son las frases del estilo "[...] como muestra la Figura de los moñecos amarillos [...]" o "[...] como muestra la siguiente Figura [...]"[7].

Una de las propiedades más útiles y, en ocasiones, infravaloradas de LaTeXes la facilidad y potencia de su sistema de etiquetado. Este sistema permite referenciar tablas, listados de código fuente, ecuaciones, capítulos, secciones, etc., con facilidad y flexibilidad. Además, LaTeXlas numera y referencia automáticamente, cambiando la numeración en función de las adiciones y supresiones sin que el autor tenga que hacer nada.

Para referenciar un elemento, lo primero que hay que crear es una etiqueta **después** del elemento a referenciar. Esto ya lo hemos visto anteriormente, por ejemplo en el listado 6.3. Si nos fijamos, se declara una etiqueta justo después de la etiqueta `caption` con el nombre `fig:img-vault-boy`. De esta manera, podemos referenciar varios indicadores de la figura, como se muestra en el listado 6.9

Listado 6.9. Referenciando una figura y su página

[7] Sí, bueno, quizá la segunda frase no es tan rara, pero siempre es preferible referenciar directamente a dar posiciones relativas.

```
Mira la Figura~\ref{fig:img-vault-boy} en la página~\nameref{
   fig:img-vault-boy}.
```

Dicho listado daría el siguiente resultado:

«Mira la Figura 6.1 titulada Vault Boy approves that en la página 31.»

¿Por qué a mí me aparece el símbolo ?? en lugar de una referencia? Pues lo más seguro es que sea un error a la hora de escribir la etiqueta. Menos común, pero también puede pasar, es que el documento no se haya compilado bien. Hay que tener en cuenta que LaTeX fue creado en una época donde las máquinas tenían poca (¡poquísima!) RAM, y para funcionar lo que se hacían eran varias compilaciones sobre el documento, almacenando los valores temporales en ficheros. Y como nadie quiere perder tiempo en cambiar y *debuggear* algo que funciona estupendamente bien, no se reimplementa. De todas formas, si te animas, ahí tienes un buen proyecto que si lo sacas adelante te va a hacer muy famoso.

6.9. Referencias bibliográficas

Hay muchas formas diferentes de gestionar las referencias bibliográficas, así que aquí hemos decidido elegir una de ellas por considerarla la más cómoda y simple, que es mediante el paquete *biblatex*.

El fichero de referencias, `references.bib`, incluirá una entrada por cada una de las referencias que se citan durante la memoria. Luego, en el cuerpo del texto, se podrán hacer referencias a dichas entradas y será LaTeX después quien se encargue de indexar correctamente, crear los hipervínculos y maquetar automáticamente.

El fichero `references.bib` puede tener muchas más de las referencias que se citan en el cuerpo del texto. Sin embargo, sólo aparecerán las referencias que se citen en el texto.

No has dicho en ningún momento *bibliografía* Sí. Las referencias bibliográficas, también conocidas como lista de referencias o simplemente referencias, son todas aquellas fuentes bibliográficas que han sido citadas a lo largo del documento. La bibliografía, también conocida como referencias externas, es simplemente una lista de recursos utilizados, citados o no. Como generalmente los no referenciados no se usan para dar soporte a un texto científico se suelen descartar.

6.9.1. ¿Cómo creamos nuevas referencias?

El fichero `references.bib` contará cero o más entradas con la estructura mostrada en el listado 6.10.

Listado 6.10. Estructura general de una referencia

```
@tipo{id,
    author = "Autor",
    title = "Título de la referencia (libro, artículo, enlace,
       ...)",
    campo1 = "valor",
    campo2 = "valor",
    \ldots
}
```

En esta entrada, `@tipo` indica el tipo de elemento (p. ej. `@article` para artículos o `@book` para libros) e `id` es un identificador **único en todo el documento** para el elemento. El resto de campos dependerán del tipo de la referencia, aunque generalmente casi todos los tipos comparten los campos de `author`, `title` o `year`.

6.9.2. ¿De qué manera puedo citar las referencias?

TBD **mcculloch1943logical**

6.10. Referencias cruzadas

TBD

6.11. Referencias a recursos externos

TBD

7. Licencia

Cuando se publica la obra en el archivo digital, por defecto lo hace con la licencia de *Creative Commons* Reconocimiento - Sin obra derivada - No comercial. Aunque usar esta licencia es correcto, no es una licencia libre y a algunos nos parece algo malo.

Considero que todo el conocimiento generado en una universidad pública ha de ser público y libre. Por ello esta obra se publica con la licencia *Creative Commons* Reconocimiento - Sin obra derivada - No comercial - Compartir igual, de tal manera que se garantiza que la obra se comparte con las mismas libertades y así, todo el mundo puede hacer de esta obra el uso que quiera.

A. Escuelas y títulos

A continuación se describen todas las opciones de grados y títulos disponibles en la memoria.

A.1. Escuelas

Las escuelas disponibles se describen en el cuadro A.1.

Clave	Valor
`etsidi`	E.T.S. de Ingeniería y Diseño Industrial
`etsisi`	E.T.S. de Ingeniería de Sistemas Informáticos

Tabla A.1. Relación entre el código de la plantilla y la escuela a la que se refiere

De momento no están todas, así que si te apetece añadir la tuya puedes, o bien contactar con los autores, o bien modificarlo (mira el apéndice B) y también contactar con los autores, así lo podemos hacer público con la mayor cantidad de usuarios posible.

A.2. Titulaciones

Cada una de las escuelas poseen ciertas titulaciones que se han de añadir a la configuración.

A.2.1. E.T.S. de Ingeniería y Diseño Industrial

La ETSIDI tiene configurados como colores principal y de link el RGB $(223, 30, 64)$. El logo es el mostrado la figura A.1.

Figura A.1. Logo de la ETSIDI utilizado en la cubierta trasera de la memoria

Las titulaciones que existen la última vez que se actualizó este documento son las siguientes:

- `56IE`: Grado en Ingeniería Eléctrica.
- `56IA`: Grado en Ingeniería Electrónica Industrial y Automática.
- `56IM`: Grado en Ingeniería Mecánica.
- `56IQ`: Grado en Grado en Ingeniería Química.
- `56DD`: Grado en Ingeniería en Diseño Industrial y Desarrollo de Producto.

A.2.2. E.T.S. de Ingeniería de Sistemas Informáticos

La ETSISI tiene configurados como colores principal y de link el RGB $(0, 177, 230)$. El logo es el mostrado la figura A.2.

Figura A.2. Logo de la ETSISI utilizado en la cubierta trasera de la memoria

Las titulaciones que existen la última vez que se actualizó este documento son las siguientes:

- `61CD`: Grado en Ciencia de Datos e Inteligencia Artificial.
- `61CI`: Grado en Ingeniería de Computadores.
- `61IW`: Grado en Ingeniería del Software.
- `61SI`: Grado en Sistemas de Información.
- `61TI`: Grado en Tecnologías para la Sociedad de la Información.

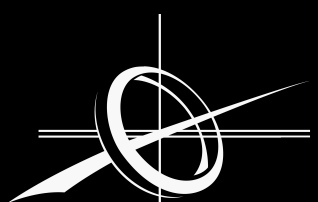
B. ¿Cómo ampliar la plantilla?

TBD

C. Lista de paquetes incluidos

TBD

Índice de términos



Universidad Politécnica de Madrid

ETSI **SISTEMAS INFORMÁTICOS**