UNIVERSIDAD DE GRANADA
E.T.S.I. INFORMÁTICA Y TELECOMUNICACIÓN

Departamento de Ciencias de la
Computación e Inteligencia Artificial

Metaheurísticas

http://sci2s.ugr.es/graduateCourses/Metaheuristicas
https://decsai.ugr.es

Guión de Prácticas

Práctica 1.b:
Técnicas de Búsqueda basadas en Poblaciones para el Problema del Aprendizaje de Pesos en Características

Curso 2016-2017

Tercer Curso del Grado en Ingeniería Informática

Práctica 1.b

Técnicas de Búsqueda basadas en Poblaciones para el Problema del Aprendizaje de Pesos en Características

1. Objetivos

El objetivo de esta práctica es estudiar el funcionamiento de las *Técnicas de Búsqueda basadas en Poblaciones* en la resolución del problema del aprendizaje de pesos en características (APC) descrito en las transparencias del Seminario 2. Para ello, se requerirá que el estudiante adapte las siguientes técnicas metaheurísticas a dicho problema:

- Algoritmos Genéticos: Dos variantes generacionales elitistas (AGGs) y otras dos estacionarias (AGEs), descritas en el Seminario 3. Aparte del esquema de evolución, la única diferencia entre los dos modelos de AGGs y AGE será el operador de cruce empleado.
- Algoritmos Meméticos: Tres variantes de algoritmos meméticos (AMs) basadas en un AGG, descritas en el Seminario 3. La única diferencia entre las tres variantes de AMs serán los parámetros considerados para definir la aplicación de la búsqueda local.

Para diseñar los AMs, será necesario también desarrollar un método de Búsqueda Local (BL) a la resolución del APC, siguiendo los pasos descritos en el Seminario 2.

El estudiante deberá comparar los resultados obtenidos con los proporcionados por el clasificador 1-NN generado considerando todas las características igualmente ponderadas (sin pesos) y con el clasificador 1-NN obtenido empleando los pesos aprendidos por el método *greedy* RELIEF, y con la propia BL en una serie de casos del problema.

La práctica se evalúa sobre un total de **3,5 puntos**, distribuidos de la siguiente forma:

- BL (1 punto).
- AGGs (0,75 puntos) y AGEs (1 punto).

- AMs (0,75 puntos).

La fecha límite de entrega será el **Lunes 17 de Abril de 2017** antes de las 23:59 horas. La entrega de la práctica se realizará por internet a través del acceso identificado de la web del departamento de CCIA (https://decsai.ugr.es).

2. Trabajo a Realizar

El estudiante podrá desarrollar los algoritmos de la práctica siguiendo la modalidad que desee: trabajando con cualquiera de los *frameworks* de metaheurísticas estudiados en el Seminario 1, implementándolos a partir del código C proporcionado en la web de la asignatura o considerando cualquier código disponible en Internet.

Los métodos desarrollados serán ejecutados sobre una serie de casos del problema. Se realizará un estudio comparativo de los resultados obtenidos y se analizará el comportamiento de cada algoritmo en base a dichos resultados. **Este análisis influirá decisivamente en la calificación final de la práctica**.

En las secciones siguientes se describen los aspectos relacionados con cada algoritmo a desarrollar y las tablas de resultados a obtener.

3. Problema y Casos Considerados

3.1. Introducción al Problema del Aprendizaje de Pesos en Características

El problema del APC consiste en optimizar el rendimiento de un clasificador basado en vecinos cercanos a partir de la inclusión de pesos asociados a las características del problema que modifican su valor en el momento de calcular las distancias entre ejemplos. En nuestro caso, el clasificador considerado será el 1-NN (k-NN, k vecinos más cercanos, con k=1 vecino). Así, el problema del APC se puede formular como:

$$\text{Maximizar } tasa\_clas(1-NN(s)) = 100 \cdot \frac{\text{nº de instancias bien clasificadas de } T}{\text{nº total de instancias de } T}$$

sujeto a que:

- $w_i = [0, 1], 1 \leq i \leq n$

donde:

- $W = (w_1, \ldots, w_n)$ es una solución al problema que consiste en un vector de números reales de tamaño n que define el peso que pondera a cada una de las características f_i.

- 1-NN es el clasificador *k*-NN con *k*=1 vecino generado a partir del conjunto de datos inicial, utilizando la técnica de validación leave-one-out y los pesos en *W* que se asocian a las *n* características.
- *T* es el conjunto de datos sobre el que se evalúa el clasificador, ya sea el conjunto de entrenamiento como el de prueba.

3.2. Conjuntos de Datos Considerados

El estudiante trabajará con los **3 conjuntos de datos** siguientes (obtenidos de la web http://www.ics.uci.edu/~mlearn/MLRepository.html y disponibles en la web de la asignatura):

1. **Sonar:** Conjunto de datos de detección de materiales mediante señales de sónar, discriminando entre objetos metálicos y rocas. 208 ejemplos con 60 características que deben ser clasificados en 2 clases.
2. **Wdbc (Wisconsin Database Breast Cancer)**: Esta base de datos contiene 30 características calculadas a partir de una imagen digitalizada de una aspiración con aguja fina (FNA) de una masa en la mama. Se describen las características de los núcleos de las células presentes en la imagen. La tarea consiste en determinar si un tumor encontrado es benigno o maligno (M = maligno, B = benigna). 569 ejemplos con 30 características que deben ser clasificados en 2 clases.
3. **Spambase:** Conjunto de datos de detección de SPAM frente a correo electrónico seguro. 460 ejemplos con 57 características que deben ser clasificados en 2 clases.

El formato de los ficheros es el ARFF de WEKA. Puede consultarse en las transparencias del Seminario 2 y en http://www.cs.waikato.ac.nz/ml/weka/.

3.3. Algoritmo de Comparación: RELIEF

El algoritmo escogido para ser comparado con las metaheurísticas implementadas es el *greedy* RELIEF, que deberá ser implementado por el estudiante. El objetivo de este algoritmo será generar un vector de pesos a partir de las distancias de cada ejemplo a su e*nemigo* más cercano y a su *amigo* más cercano. El *enemigo* más cercano a un ejemplo es aquel ejemplo de diferente clase más cercano a él. El *amigo* más cercano a un ejemplo es otro ejemplo de su misma clase más cercano a él.

La intuición del algoritmo es doble: primero se basa en incrementar el peso a aquellas características que mejor separan a ejemplos que son enemigos entre sí; también reduce el valor del peso en aquellas características que separan ejemplos que son amigos entre sí. Este/a incremento/reducción es directamente proporcional a la distancia entre los ejemplos en cada característica. A continuación, se describe el algoritmo en los siguientes pasos:

- Se inicializa el vector de pesos *W* a cero.
- Para cada ejemplo del conjunto de entrenamiento, se identifica el enemigo y amigo más cercano. *W* se actualiza sumándole la distancia existente entre el ejemplo y su enemigo más cercano (considerando todas las características, suma

componente a componente). Después, *W* se actualiza restándole la distancia existente entre el ejemplo y su amigo más cercano.

- Una vez hecho lo anterior para todos los ejemplos, *W* puede tener componentes negativas o que no estén en *[0, 1]*. Los valores negativos de *W* se truncarán a cero. El resto de valores se normalizarán dividiéndolo por el valor máximo encontrado en *W*.
- Nótese que el orden de actuación de los ejemplos no altera el resultado final de *W*.

Como el resto de algoritmos, el algoritmo RELIEF deberá ser ejecutado 10 veces para cada conjunto de datos de acuerdo a lo explicado en la siguiente sección.

3.4. Determinación de la Calidad de un Algoritmo

El modo habitual de determinar la calidad de un algoritmo de resolución aproximada de problemas de optimización es ejecutarlo sobre un conjunto determinado de instancias y comparar los resultados obtenidos con los mejores valores conocidos para dichas instancias (en caso de existir).

Además, los algoritmos pueden tener diversos parámetros o pueden emplear diversas estrategias. Para determinar qué valor es el más adecuado para un parámetro o saber la estrategia más efectiva los algoritmos también se comparan entre sí.

La comparación de los algoritmos se lleva a cabo fundamentalmente usando dos criterios, la *calidad* de las soluciones obtenidas y el *tiempo de ejecución* empleado para conseguirlas. Además, es posible que los algoritmos no se comporten de la misma forma si se ejecutan sobre un conjunto de instancias u otro.

Por otro lado, a diferencia de los algoritmos determinísticos, los algoritmos probabilísticos se caracterizan por la toma de decisiones aleatorias a lo largo de su ejecución. Este hecho implica que un mismo algoritmo probabilístico aplicado al mismo caso de un problema pueda comportarse de forma diferente y por tanto proporcionar resultados distintos en cada ejecución.

Cuando se analiza el comportamiento de una metaheurística probabilística en un caso de un problema, se desearía que el resultado obtenido no estuviera sesgado por una secuencia aleatoria concreta que pueda influir positiva o negativamente en las decisiones tomadas durante su ejecución. Por tanto, resulta necesario efectuar varias ejecuciones con distintas secuencias probabilísticas y calcular el resultado medio y la desviación típica de todas las ejecuciones para representar con mayor fidelidad su comportamiento.

Dada la influencia de la aleatoriedad en el proceso, es recomendable disponer de un generador de secuencia pseudoaleatoria de buena calidad con el que, dado un valor semilla de inicialización, se obtengan números en una secuencia lo suficientemente grande (es decir, que no se repitan los números en un margen razonable) como para considerarse aleatoria. En las webs de la asignatura se puede encontrar una implementación en lenguaje C de un generador aleatorio de buena calidad (*random.h*).

Como norma general, el proceso a seguir consiste en realizar un número de ejecuciones diferentes de cada algoritmo probabilístico considerado para cada caso del problema. Es necesario asegurarse de que se realizan diferentes secuencias aleatorias en dichas ejecuciones. Así, el valor de la semilla que determina la inicialización de cada secuencia deberá ser distinto en cada ejecución y estas semillas deben mantenerse en los distintos algoritmos (es decir, la semilla para la primera ejecución de todos los algoritmos debe ser la misma, la de la segunda también debe ser la misma y distinta de la anterior, etc.). Para mostrar los resultados obtenidos con cada algoritmo en el que se hayan realizado varias ejecuciones, se deben construir tablas que recojan los valores correspondientes a estadísticos como el **mejor** y **peor** resultado para cada caso del problema, así como la **media** y la **desviación típica** de todas las ejecuciones. También se pueden emplear descripciones más representativas como los boxplots, que proporcionan información de todas las ejecuciones realizadas mostrando mínimo, máximo, mediana y primer y tercer cuartil de forma gráfica, Finalmente, se construirán unas tablas globales con los resultados agregados que mostrarán la calidad del algoritmo en la resolución del problema desde un punto de vista general.

Alternativamente, **en el caso de que se considere un número alto de casos del problema, se puede confiar en una única ejecución del algoritmo sobre cada caso del problema. Lo mismo ocurre en aquellos problemas de aprendizaje automático en los que se consideren varias ejecuciones del algoritmo sobre distintos subconjuntos del conjunto de datos original dentro del proceso de validación considerado. Esta condición se cumple en las prácticas que realizaremos con el APC**. Aun así, **será necesario inicializar la semilla del generador aleatorio para poder repetir el experimento** y obtener los mismos resultados si fuera necesario (en caso contrario, los resultados podrían variar en cada ejecución del mismo algoritmo sobre el mismo caso del problema).

En nuestro problema, consideraremos el método de validación ***5×2-cross validation*** (cv). Para ello, se generarán 5 particiones distintas del conjunto de datos proporcionado en subconjuntos de entrenamiento y prueba realizadas al 50%. Para cada partición, se aprenderá primero el clasificador con una parte y se validará con la otra y viceversa, resultando en 10 ejecuciones de cada algoritmo sobre cada conjunto de datos.

La medida de validación será la tasa de acierto del clasificador 1-NN sobre el conjunto de prueba. El resultado final será la media de los 10 valores obtenidos, uno para cada ejecución del algoritmo.

Para facilitar la comparación de algoritmos en las prácticas del APC se considerarán dos estadísticos distintos denominados *Tasa_clas* y *Tiempo:*

- *Tasa_clas* se calcula como la media de los porcentajes de acierto (las tasas de clasificación) obtenidos por cada método en cada partición del conjunto de datos (es el valor final resultante del 5×2-cv).

- *Tiempo* se calcula como la media del tiempo de ejecución empleado por el algoritmo para resolver cada caso del problema (cada conjunto de datos). Es decir, en nuestro caso es la media del tiempo empleado por las 10 ejecuciones.

Cuanto mayor es el valor de *Tasa_clas* para un algoritmo, mejor calidad tiene dicho algoritmo, porque obtiene clasificadores más precisos en media. La elección entre un clasificador de mejor rendimiento o de mayor eficiencia depende de los requisitos de la aplicación concreta. Si dos métodos obtienen soluciones con la misma calidad (tienen valores de *Tasa_clas* similares), uno será mejor que el otro si emplea menos tiempo en media. En la web de la asignatura hay también disponible un código en C (*timer*) para un cálculo adecuado del tiempo de ejecución de los algoritmos metaheurísticos.

La hoja Excel *Tablas_APC_2016-17.xls*, disponible en la web de la asignatura, permite recopilar los estadísticos comentados para las tablas de resultados de la práctica.

4. Componentes de los Algoritmos

Los algoritmos de esta práctica tienen en común las siguientes componentes:

- *Esquema de representación*: Se seguirá la representación real basada en un vector *W* de tamaño *n* con valores en *[0, 1]* que indican el peso asociado a cada característica, explicado en las transparencias del seminario.

- *Función objetivo*: Será el porcentaje de acierto del clasificador 1-NN modificando el valor de las características con el vector *W,* y utilizando la técnica de validación *leave-one-out* explicada en las transparencias del seminario. El objetivo será maximizar esta función.

- *Generación de la solución inicial*: La solución inicial se generará de forma aleatoria utilizando una distribución uniforme en *[0, 1]* en todos los casos.

- *Esquema de generación de vecinos*: Se empleará el movimiento de cambio por mutación normal *Mov(W,σ)* que altera el vector *W* sumándole otro vector *Z* generado a partir de una distribución normal de media 0 y varianza σ^2. Su aplicación concreta dependerá del algoritmo específico.

- *Criterio de aceptación*: Se considera una mejora cuando se aumenta el valor global de la función objetivo.

- *Criterio de parada*: Se detendrá la ejecución del algoritmo bien cuando no se encuentre mejora al generar un máximo número de vecinos (BL) o bien cuando se hayan evaluado 15000 soluciones distintas (en cualquier caso, en la BL, se parará también después de 15000 evaluaciones aunque siguiera habiendo soluciones mejores en el entorno).

  A continuación veremos las particularidades de cada algoritmo.

4.1. Búsqueda Local

Algoritmo

Como algoritmo de BL para el APC consideraremos el esquema del primer mejor, tal y como está descrito en las transparencias del Seminario 2.

Valores de los parámetros y ejecuciones

Cuando la BL se ejecute como algoritmo individual, sin estar integrada en un AM, se detendrá la ejecución **cuando no se encuentre mejora tras generar un máximo de 20·n vecinos (n es el número de características) o cuando se hayan realizado 15000 evaluaciones de la función objetivo**, es decir, en cuanto se cumpla alguna de las dos condiciones. Además, se considerará un σ=0,3.

4.2. Algoritmos Genéticos

Algoritmos

Los AGs de esta práctica presentarán las siguientes componentes:

- *Esquema de evolución*: Como se ha comentado, se considerarán dos versiones, una basada en el esquema generacional con elitismo (AGG) y otra basada en el esquema estacionario (AGE). En el primero se seleccionará una población de padres del mismo tamaño que la población genética mientras que en el segundo se seleccionarán únicamente dos padres.

- *Operador de selección*: Se usará el torneo binario, consistente en elegir aleatoriamente dos individuos de la población y seleccionar el mejor de ellos. En el esquema generacional, se aplicarán tantos torneos como individuos existan en la población genética, incluyendo los individuos ganadores en la población de padres. En el esquema estacionario, se aplicará dos veces el torneo para elegir los dos padres que serán posteriormente recombinados (cruzados).

- *Esquema de reemplazamiento*: En el esquema generacional, la población de hijos sustituye automáticamente a la actual. Para conservar el elitismo, si la mejor solución de la generación anterior no sobrevive, sustituye directamente la peor solución de la nueva población. En el estacionario, los dos descendientes generados tras el cruce y la mutación (esta última aplicada según una determinada probabilidad) sustituyen a los dos peores de la población actual, en caso de ser mejores que ellos.

- *Operador de cruce*: Se emplearán dos operadores de cruce para representación real, explicados en las transparencias de teoría del tema de Algoritmos Genéticos y del Seminario 3. Uno de ellos será el operador BLX-α, con α=0.3. El otro será el cruce aritmético. **Esto resultará en el desarrollo de cuatro AGs distintos, dos generacionales (AGG-BLX y AGG-CA) y dos estacionarios (AGE-BLX y AGE-CA)**.

- *Operador de mutación*: Se considerará el operador de mutación normal para representación real, explicado en las transparencias del Seminario 2 y de teoría. Para aplicarlo, se considerará un valor σ=0,3.

Valores de los parámetros y ejecuciones

El tamaño de la población será de 30 cromosomas. La probabilidad de cruce será 0,7 en el AGG y 1 en el AGE (siempre se cruzan los dos padres). La probabilidad de mutación (por gen) será de 0,001 en ambos casos. El criterio de parada en las dos versiones del AG consistirá en realizar **15000 evaluaciones de la función objetivo**.

4.3. Algoritmos Meméticos

Algoritmos

El AM consistirá en hibridar el algoritmo genético generacional (AGG) que mejor resultado haya proporcionado con la BL desarrollada. Se estudiarán las tres posibilidades de hibridación siguientes:

1. AM-(10,1.0): Cada 10 generaciones, se aplica la BL sobre todos los cromosomas de la población.
2. AM-(10,0.1): Cada 10 generaciones, se aplica la BL sobre un subconjunto de cromosomas de la población seleccionado aleatoriamente con probabilidad p_{LS} igual a 0.1 para cada cromosoma.
3. AM-(10,0.1mej): Cada 10 generaciones, aplicar la BL sobre los $0.1 \cdot N$ mejores cromosomas de la población actual (N es el tamaño de ésta).

Valores de los parámetros y ejecuciones

El tamaño de la población del AGG será de 10 cromosomas. Las probabilidades de cruce y mutación serán 0,7 (por cromosoma) y 0,001 (por gen) en ambos casos. Se detendrá la ejecución de la BL aplicada sobre un cromosoma **cuando se hayan evaluado $2 \cdot n$ vecinos distintos en la ejecución,** siendo *n* el tamaño del cromosoma. El criterio de parada del AM consistirá en realizar **15000 evaluaciones de la función objetivo**, incluidas por supuesto las de la BL.

5. Tablas de Resultados a Obtener

Se diseñará una tabla para cada algoritmo (1-NN, RELIEF, BL, AGG-BLX, AGG-CA, AGE-BLX, AGE-CA, AM-(10,1.0), AM-(10,0.1) y AM-(10,0.1mej)) donde se recojan los resultados de la ejecución de dicho algoritmo en los conjuntos de datos considerados. Tendrá la misma estructura que la Tabla 5.1.

Tabla 5.1: Resultados obtenidos por el algoritmo X en el problema del APC

	Sonar		**Wdbc**		**Spambase**	
	%_clas	**T**	***%_clas***	**T**	***%_clas***	**T**
Partición 1-1	x	x	x	x	x	x
Partición 1-2	x	x	x	x	x	x
Partición 2-1	x	x	x	x	x	x
Partición 2-2	x	x	x	x	x	x
Partición 3-1	x	x	x	x	x	x

Partición 3-2	x	x	x	x	x	x
Partición 4-1	x	x	x	x	x	x
Partición 4-2	x	x	x	x	x	x
Partición 5-1	x	x	x	x	x	x
Partición 5-2	x	x	x	x	x	x
Media	x	x	x	x	x	x

Finalmente, se construirá una tabla de resultados global que recoja los resultados medios de calidad y tiempo para todos los algoritmos considerados, tal como se muestra en la Tabla 5.2. Para rellenar esta tabla se hará uso de los resultados medios mostrados en las tablas parciales. Aunque en la tabla que sirve de ejemplo se han incluido todos los algoritmos considerados en esta práctica, naturalmente sólo se incluirán los que se hayan desarrollado.

Tabla 5.2: Resultados globales en el problema del APC

	Sonar		**Wdbc**		**Spambase**	
	%_clas	**T**	***%_clas***	**T**	***%_clas***	**T**
1-NN	x	x	x	x	x	x
RELIEF	x	x	x	x	x	x
BL	x	x	x	x	x	x
AGG-BLX	x	x	x	x	x	x
AGG-CA	x	x	x	x	x	x
AGE-BLX	x	x	x	x	x	x
AGE-CA	x	x	x	x	x	x
AM-(10,1.0)	x	x	x	x	x	x
AM-(10,0.1)	x	x	x	x	x	x
AM-(10,0.1mej)	x	x	x	x	x	x

A partir de los datos mostrados en estas tablas, el estudiante realizará un análisis de los resultados obtenidos, ***que influirá significativamente en la calificación de la práctica***. En dicho análisis se deben comparar los distintos algoritmos en términos de las tasas de clasificación obtenidas (capacidad del algoritmo para obtener soluciones de calidad) y el tiempo requerido para obtener las soluciones (rapidez del algoritmo). Se comparará el rendimiento de las metaheurísticas entre sí, así como con respecto a los algoritmos de referencia, el 1-NN original y el RELIEF.

6. Documentación y Ficheros a Entregar

En general, la **documentación** de ésta y de cualquier otra práctica será un fichero pdf que deberá incluir, al menos, el siguiente contenido:

a) Portada con el número y título de la práctica, el curso académico, el nombre del problema escogido, los algoritmos considerados; el nombre, DNI y dirección e-mail del estudiante, y su grupo y horario de prácticas.

b) Índice del contenido de la documentación con la numeración de las páginas.

c) **Breve** descripción/formulación del problema (**máximo 1 página**). Podrá incluirse el mismo contenido repetido en todas las prácticas presentadas por el estudiante.

d) Breve descripción de la aplicación de los algoritmos empleados al problema (**máximo 4 páginas**): Todas las consideraciones comunes a los distintos algoritmos se describirán en este apartado, que será previo a la descripción de los algoritmos específicos. Incluirá por ejemplo la descripción del esquema de representación de soluciones y la descripción en pseudocódigo (no código) de la función objetivo y los operadores comunes. En este caso, el operador de generación de vecino/mutación, la generación de soluciones aleatorias, el mecanismo de selección de los AGs y los operadores de cruce y mutación empleados.

e) Descripción en **pseudocódigo** de la **estructura del método de búsqueda** y de todas aquellas **operaciones relevantes** de cada algoritmo. Este contenido, específico a cada algoritmo se detallará en los correspondientes guiones de prácticas. El pseudocódigo **deberá forzosamente reflejar la implementación/ el desarrollo realizados** y no ser una descripción genérica extraída de las transparencias de clase o de cualquier otra fuente. La descripción de cada algoritmo no deberá ocupar más de **2 páginas**.

   Para esta primera práctica, se incluirán al menos las descripciones en pseudocódigo de:
   - Para el algoritmo BL, descripción en pseudocódigo del método de exploración del entorno.
   - Para los AGs, el esquema de evolución y de reemplazamiento considerados.
   - Para los AMs, el esquema de búsqueda seguido por cada algoritmo en lo que respecta a la integración de la BL dentro del AG.

f) Descripción en **pseudocódigo** del algoritmo de comparación.

g) Breve explicación del **procedimiento considerado para desarrollar la práctica**: implementación a partir del código proporcionado en prácticas o a partir de cualquier otro, o uso de un framework de metaheurísticas concreto. Inclusión de un pequeño **manual de usuario describiendo el proceso para que el profesor de prácticas pueda replicarlo**.

h) Experimentos y análisis de resultados:
   - Descripción de los casos del problema empleados y de los valores de los parámetros considerados en las ejecuciones de cada algoritmo (**incluyendo las semillas utilizadas**).
   - Resultados obtenidos según el formato especificado.
   - Análisis de resultados. El análisis deberá estar orientado a **justificar** (según el comportamiento de cada algoritmo) **los resultados** obtenidos en lugar de realizar una mera "lectura" de las tablas. Se valorará la inclusión de otros elementos de comparación tales como gráficas de convergencia, boxplots, análisis comparativo de las soluciones obtenidas, representación gráfica de las soluciones, etc.

i) Referencias bibliográficas u otro tipo de material distinto del proporcionado en la asignatura que se haya consultado para realizar la práctica (en caso de haberlo hecho).

Aunque lo esencial es el contenido, también debe cuidarse la presentación y la redacción. **La documentación nunca deberá incluir listado total o parcial del código fuente**.

En lo referente al **desarrollo de la práctica**, se entregará una carpeta llamada **software** que contenga una versión ejecutable de los programas desarrollados, así como los ficheros de datos de los casos del problema y el código fuente implementado o los ficheros de configuración del framework empleado. El código fuente o los ficheros de configuración se organizarán en la estructura de directorios que sea necesaria y deberán colgar del directorio FUENTES en el raíz. Junto con el código fuente, hay que incluir los ficheros necesarios para construir los ejecutables según el entorno de desarrollo empleado (tales como *.prj, makefile, *.ide, etc.). La versión ejecutable de los programas y los ficheros de datos se incluirán en un subdirectorio del raíz de nombre BIN. En este mismo directorio se adjuntará un pequeño fichero de texto de nombre LEEME que contendrá breves reseñas sobre cada fichero incluido en el directorio. Es importante que los programas realizados puedan leer los valores de los parámetros de los algoritmos desde fichero, es decir, que no tengan que ser recompilados para cambiar éstos ante una nueva ejecución. Por ejemplo, la semilla que inicializa la secuencia pseudoaleatoria debería poder especificarse como un parámetro más.

El fichero pdf de la documentación y la carpeta software serán comprimidos en un fichero .zip etiquetado con los apellidos y nombre del estudiante (Ej. Pérez Pérez Manuel.zip). Este fichero será entregado por internet a través del acceso identificado de la web del departamento de CCIA (https://decsai.ugr.es).

7. Método de Evaluación

Tanto en esta práctica como en las siguientes, se indicará la puntuación máxima que se puede obtener por cada algoritmo y su análisis. La inclusión de trabajo voluntario (desarrollo de variantes adicionales, experimentación con diferentes parámetros, prueba con otros operadores o versiones adicionales del algoritmo, análisis extendido, etc.) podrá incrementar la nota final por encima de la puntuación máxima definida inicialmente.

En caso de que el comportamiento del algoritmo en la versión implementada/ desarrollada no coincida con la descripción en pseudocódigo o no incorpore las componentes requeridas, se podría reducir hasta en un 50% la calificación del algoritmo correspondiente.