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

Departamento de Ciencias de la
Computación e Inteligencia Artificial

Metaheurísticas

http://sci2s.ugr.es/docencia/metah
https://decsai.ugr.es

Guión de Prácticas

Práctica 1.b:
Búsquedas por Trayectorias para el
Problema de la Selección de Características

Curso 2015-2016

Tercer Curso del Grado en Ingeniería Informática

Práctica 1.b

Búsquedas por Trayectorias para el Problema de la Selección de Características

1. Objetivos

El objetivo de esta práctica es estudiar el funcionamiento de los siguientes algoritmos: Búsqueda Local (BL), Enfriamiento Simulado (ES) y Búsqueda Tabú (BT). Para ello, se requerirá que el alumno adapte estos métodos para resolver el problema de la selección de características (SC) descrito en las transparencias del Seminario 2.a y que compare los resultados obtenidos con los proporcionados por el clasificador 3-NN generado considerando todas las características disponibles y con el clasificador 3-NN obtenido empleando las características seleccionadas por el método *greedy Sequential Forward Selection (*SFS*)* en una serie de casos del problema.

La práctica se evalúa sobre un total de **2,5 puntos**, distribuidos de la siguiente forma: BL (0,75 puntos), ES (0,75 puntos) y BT (1 punto). Además, se podrá implementar opcionalmente una variante extendida de la BT que puede suponer **1 punto adicional**.

La fecha límite de entrega será el **Lunes 4 de Abril de 2016** 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 alumno 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 el problema y los casos considerados, 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 de la Selección de Características

El problema de la SC consiste en extraer el subconjunto de características de un conjunto inicial proporcionado que permita obtener un clasificador con el máximo rendimiento. En nuestro caso, el clasificador considerado será el 3-NN (*k*-NN, *k* vecinos más cercanos, con *k*=3 vecinos). Así, el problema de la SC se puede formular como:

$$\text{Maximizar } tasa\_clas(3-NN(s)) = 100 \cdot \frac{\text{nº de instancias bien clasificadas de } T}{\text{nº total de instancias de } T}$$

sujeto a que:

- $s_i = \{0,1\}, 1 \leq i \leq n$

donde:

- $s = (s_1, \ldots, s_n)$ es una solución al problema que consiste en un vector binario de tamaño n que define si la característica f_i ha sido o no seleccionada.
- 3NN(s) es el clasificador k-NN con k=3 vecinos generado a partir del conjunto de datos inicial y las características seleccionadas en s.
- T es el conjunto de datos de prueba sobre el que se evalúa el clasificador.

3.2. Conjuntos de Datos Considerados

El alumno 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. **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.
2. **Movement_Libras**: Cada clase referencia a un tipo de movimiento de la mano en LIBRAS (nombre portugués 'Língua Brasileira de Sinais', la lengua oficial de señales brasileña). 360 ejemplos con 90 características que deben ser clasificados en 15 clases.

3. **Arrhythmia**: Distinguir entre la presencia y la ausencia de arritmia cardiaca y clasificarlo en uno de los 5 grupos mayoritarios. 386 ejemplos (seleccionados de los 452 originales, eliminando los ejemplos de clases minoritarias e imputando valores perdidos con la media/moda no supervisada) con 278 características (una original se elimina al tener muchos valores perdidos) que deben ser clasificados en 5 clases (de 16 originales, solo se mantienen las que superan un mínimo de 20 representantes).

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

3.3. Algoritmo de Comparación: SFS

El algoritmo escogido para ser comparado con las metaheurísticas implementadas es el *greedy Sequential Forward Selection (*SFS*)*, que deberá ser implementado por el alumno. El objetivo de este algoritmo será generar un subconjunto formado por las *m* características más prometedoras del conjunto inicial, escogidas de una en una mientras haya ganancia en el rendimiento del clasificador.

La primera característica seleccionada será aquella que produzca una mayor tasa de clasificación inicial al ser empleada en solitario para generar el clasificador. A continuación, se añaden progresivamente los atributos más prometedores procediendo del siguiente modo:

- Se evalúa el valor de la función de selección de cada característica candidata (de la lista de no seleccionadas hasta el momento) midiendo la tasa de clasificación resultante de añadir cada característica candidata al conjunto de características actual de forma individual.
- Se añade la característica que proporcione el mayor valor de la función de selección, es decir, la que produce la mayor ganancia en la función de evaluación con respecto al conjunto de características ya seleccionado.
- Se detiene la ejecución del algoritmo en el momento en que no se encuentre alguna característica que produzca ganancia.

Como el resto de algoritmos, el SFS 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 la web 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. Alternativamente, 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 la SC**. 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 3-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 de la SC se considerarán tres estadísticos distintos denominados *Tasa_clas*, *Tasa_red* 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).

- *Tasa_red* corresponde al porcentaje de reducción obtenido en la selección del subconjunto de características respecto al total. Para cada subconjunto seleccionado *s*, se calcula de la siguiente forma:

$$tasa\_red(s) = 100 \cdot \frac{\text{nº total de características - nº de características seleccionadas}}{\text{nº total de características}}$$

  El valor global se obtiene como la media de los porcentajes de reducción obtenidos por cada método en cada partición del conjunto de datos.

- *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. Del mismo modo, cuanto mayor es el valor de *Tasa_red*, mejor calidad tiene también el algoritmo, porque obtiene clasificadores más simples en media. La elección entre un clasificador de mejor rendimiento o de mayor simplicidad depende de los requisitos de la aplicación concreta. Finalmente, si dos métodos obtienen soluciones con la misma calidad (tienen valores de *Tasa_clas* y *Tasa_red* 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_SC_2015-16.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 binaria basada en un vector s de tamaño n con valores en $\{0,1\}$ que indica la selección o la eliminación de las características, explicada en las transparencias del seminario.

- *Función objetivo*: Será el porcentaje de acierto del clasificador 3-NN generado a partir de la selección de características codificada en la solución actual. El objetivo será maximizar esta función.

- *Generación de la solución inicial*: La solución inicial se generará de forma aleatoria en todos los casos.

- *Esquema de generación de vecinos*: Se empleará el movimiento de cambio de pertenencia $Flip(s,i)$ que altera la selección de la característica correspondiente a la posición i en la solución s (pasa de 0 a 1 o viceversa). Su aplicación concreta dependerá del algoritmo específico.

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

- *Criterio de parada*: Se detendrá la ejecución del algoritmo bien cuando no se encuentre mejora en todo el entorno (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 la SC consideraremos el esquema del primer mejor, tal y como está descrito en las transparencias del Seminario 2.a.

4.2. Enfriamiento Simulado

Algoritmo

Se ha de emplear un algoritmo ES con las siguientes componentes:

- *Esquema de enfriamiento*: Se empleará el esquema de Cauchy modificado:

$$T_{k+1} = \frac{T_k}{1+\beta \cdot T_k} \qquad ; \qquad \beta = \frac{T_0 - T_f}{M \cdot T_0 \cdot T_f}$$

donde M es el número de enfriamientos a realizar, T_0 es la temperatura inicial y T_f es la temperatura final que tendrá un valor cercano a cero [1].

- *Operador de Vecino y exploración del entorno para L(T)*: En cada iteración del bucle interno *L(T)*, se aplicará un único movimiento *Flip*(*s*,*i*) para generar una única solución vecina que será comparada con la solución actual. Se escogerá aleatoriamente la característica *i* a la que se le cambiará su pertenencia (de 0 a 1 o de 1 a 0).

- *Condición de enfriamiento L(T)*: Se enfriará la temperatura, finalizando la iteración actual, bien cuando se haya generado un número máximo de vecinos *máx_vecinos* (independientemente de si han sido o no aceptados) o bien cuando se haya aceptado un número máximo de los vecinos generados *máx_éxitos*.

- *Condición de parada*: El algoritmo finalizará bien cuando haya alcanzado el número máximo de evaluaciones prefijado o bien cuando el número de éxitos en el enfriamiento actual sea igual a 0.

Valores de los parámetros y ejecuciones

La temperatura inicial se calculará en función de la siguiente fórmula:

$$T_0 = \frac{\mu \cdot C(S_0)}{-\ln(\phi)}$$

donde T_0 es la temperatura inicial, $C(S_0)$ es el coste de la solución inicial y $\phi \in [0,1]$ es la probabilidad de aceptar una solución un μ por 1 peor que la inicial. En las ejecuciones se considerará $\phi=\mu=0{,}3$. La temperatura final T_f se fijará a 10^{-3} (*¡comprobando siempre que sea menor que la inicial!*).

Los parámetros que definen el bucle interno *L(T)* tomarán valor *máx_vecinos=10·n* (tamaño del caso del problema) y *máx_éxitos=0.1*máx_vecinos*. El número máximo de evaluaciones será 15000. Por lo tanto, el número de iteraciones (enfriamientos) *M* del algoritmo ES será igual a 15000/*máx_vecinos* [2].

[1] **NOTA**: Si el alumno observa que este esquema de enfriamiento enfría demasiado rápido, puede sustituirlo por un esquema proporcional: $T_{k+1} = \leftarrow \alpha \cdot T_k$ con $\alpha \in [0.9, 0.99]$.

[2] **NOTA 1**: Es posible que se realicen menos de 15000 evaluaciones durante la ejecución debido a la condición de enfriamiento cuando se alcanzan *máx_éxitos*.

NOTA 2: Puede que con *máx_vecinos*=10·*n* se generen demasiados vecinos para cada enfriamiento. El alumno puede probar también con *máx_vecinos*=5·*n* o directamente *n*.

4.3. Búsqueda Tabú

Algoritmo versión básica

Se trabajará con la versión del algoritmo BT estudiada en clase, la cual utiliza una lista de movimientos tabú y tres mecanismos de reinicialización:

- *Lista tabú*: Almacena el índice *i* de las características a las que se les cambió la selección en los movimientos que generaron las soluciones aceptadas en las iteraciones recientes. No se permiten movimientos que restauren el estado (seleccionado/no seleccionado) de la característica s_i durante la tenencia tabú de ese movimiento.

- *Operador de Vecino y exploración del entorno para L(T)*: En cada iteración se generarán 30 soluciones vecinas aplicando el movimiento *Flip*(*s*,*i*) para obtener cada una de ellas. Se escogerá aleatoriamente el índice *i* de la característica a cambiar, sin repetidos. Se seleccionará el mejor de esos 30 vecinos de acuerdo a los criterios tabú.

- *Criterio de aspiración*: Tener mejor coste (mayor) que la mejor solución obtenida hasta el momento.

Valores de los parámetros y ejecuciones de la versión básica

El número máximo de evaluaciones será 15000. El tamaño inicial de cada lista tabú será *n*/3.

Algoritmo versión extendida

La versión extendida consiste en incorporar una memoria de largo plazo y un procedimiento de reinicialización a la versión básica anterior. Para ello, se aplicará una oscilación estratégica basada en tres estrategias distintas, escogidas aleatoriamente según una probabilidad: construcción de una solución inicial aleatoria (diversificación), nuevo arranque de la búsqueda desde la mejor solución encontrada hasta el momento (intensificación) y uso de la memoria a largo plazo para generar una nueva solución diferente a las visitadas (diversificación controlada).

El uso de la memoria a largo plazo consistirá en mantener una estadística en forma de vector *frec* de tamaño *n* de modo que $frec_i$ almacene el número de veces que la característica *i* ha sido seleccionada en las soluciones aceptadas durante la búsqueda. **La memoria de largo plazo no se actualiza tras cada reinicialización**.

Para reinicializar con diversidad usando la memoria a largo plazo, se genera una nueva solución inicial de forma aleatoria con una probabilidad inversa a la de la memoria de frecuencias. Se incluirá la característica *i* en la solución inicial *s* con menor probabilidad cuanto mayor sea su valor de $frec_i$:

$$s_i = \begin{cases} 1, & \text{si } u_{\text{aleatorio}} < 1 - \dfrac{frec_i}{num\_soluciones} \\ 0, & \text{si } u_{\text{aleatorio}} \geq 1 - \dfrac{frec_i}{num\_soluciones} \end{cases}$$

Además de saltar a una solución concreta según las reinicializaciones comentadas, también se alterará un parámetro del algoritmo de búsqueda para provocar un cambio de comportamiento del algoritmo más efectivo. Este consistirá en variar el tamaño de la lista tabú, incrementándola o reduciéndola en un tanto por ciento según una decisión aleatoria uniforme.

Valores de los parámetros y ejecuciones de la versión extendida

Se realizará una reinicialización cuando transcurran ***10 iteraciones*** sin mejorar la mejor solución obtenida hasta el momento en la ejecución del algoritmo (mejor global). La probabilidad de reinicializar desde una solución inicial aleatoria es 0,25, la de partir de la mejor solución obtenida es 0,25 y la de usar la memoria a largo plazo parar generar la solución más diversa es 0,5.

El tamaño de la lista tabú cambiará después de las reinicializaciones, aumentando o disminuyendo en un 50%.

5. Tablas de Resultados a Obtener

Se diseñará una tabla para cada algoritmo (3-NN, SFS, BL, ES, BT básica y BT extendida) 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 de la SC

	Wdbc			**Movement_Libras**			**Arrhythmia**		
	%_clas	***%_red***	**T**	***%_clas***	***%_red***	**T**	***%_clas***	***%_red***	**T**
Partición 1-1	x	x	x	x	x	x	x	x	x
Partición 1-2	x	x	x	x	x	x	x	x	x
Partición 2-1	x	x	x	x	x	x	x	x	x
Partición 2-2	x	x	x	x	x	x	x	x	x
Partición 3-1	x	x	x	x	x	x	x	x	x
Partición 3-2	x	x	x	x	x	x	x	x	x
Partición 4-1	x	x	x	x	x	x	x	x	x
Partición 4-2	x	x	x	x	x	x	x	x	x
Partición 5-1	x	x	x	x	x	x	x	x	x
Partición 5-2	x	x	x	x	x	x	x	x	x
Media	x	x	x	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 de la SC

	Wdbc			**Movement_Libras**			**Arrhythmia**		
	%_clas	***%_red***	**T**	***%_clas***	***%_red***	**T**	***%_clas***	***%_red***	**T**
3-NN	x	0	x	x	0	x	x	0	x
SFS	x	x	x	x	x	x	x	x	x
BL	x	x	x	x	x	x	x	x	x
ES	x	x	x	x	x	x	x	x	x
BT básica	x	x	x	x	x	x	x	x	x
BT extendida	x	x	x	x	x	x	x	x	x

A partir de los datos mostrados en estas tablas, el alumno 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 y de reducción 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 3-NN original y el SFS.

6. Documentación y Ficheros a Entregar

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

1. 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 alumno, y su grupo y horario de prácticas.
2. Índice del contenido de la documentación con la numeración de las páginas.
3. **Breve** descripción/formulación del problema (**máximo 1 página**). Podrá incluirse el mismo contenido repetido en todas las prácticas presentadas por el alumno.
4. Breve descripción de la aplicación de los algoritmos empleados al problema (**máximo 2 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 de la función objetivo y los operadores comunes (en este caso, el de **generación de vecino**), etc.
5. 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, además de la descripción del esquema de búsqueda, se deberá incluir al menos:

- Para el algoritmo BL, descripción en pseudocódigo del método de exploración del entorno.
- Para el algoritmo ES, descripción del cálculo de la temperatura inicial y del esquema de enfriamiento.
- Para el algoritmo BT básico, descripción en pseudocódigo del manejo de la lista tabú.
- Para el algoritmo BT extendido, en caso de haberlo realizado, descripción en pseudocódigo del manejo de los mecanismos de reinicialización.

6. **Breve** descripción del algoritmo de comparación.
7. 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**.
8. 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.
9. 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 alumno (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, como la BT extendida de esta práctica, experimentación con diferentes parámetros, prueba con otros operadores o versiones adicionales del algoritmo, análisis extendido, etc.) podrá incrementar la nota final.

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.