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

UNIVERSIDAD DE GRANADA

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 Local y Algoritmos Greedy para el Problema del Agrupamiento con Restricciones

Curso 2020-2021

Tercer Curso del Grado en Ingeniería Informática

Práctica 1.b

Técnicas de Búsqueda Local y Algoritmos Greedy para el Problema del Agrupamiento con Restricciones

1. Objetivos

El objetivo de esta práctica es estudiar el funcionamiento de las *Técnicas de Búsqueda Local y de los Algoritmos Greedy* en la resolución del Problema del Agrupamiento con Restricciones (PAR) descrito en las transparencias del Seminario 2. Para ello, se requerirá que el estudiante adapte los siguientes algoritmos a dicho problema:

- Algoritmo *greedy* COPKM.
- Algoritmo de Búsqueda Local (BL).

El estudiante deberá comparar los resultados obtenidos por ambos algoritmos en una serie de casos del problema.

La práctica se evalúa sobre un total de **2 puntos**, distribuidos de la siguiente forma:

- BL (**1.25 puntos**).
- *Greedy* COPKM (**0.75 puntos**).

La fecha límite de entrega será el **domingo 11 de abril de 2021** antes de las 23:55 horas. La entrega de la práctica se realizará por internet a través del espacio de la asignatura en PRADO.

2. Trabajo a Realizar

El estudiante podrá desarrollar los algoritmos de la práctica siguiendo la modalidad que desee: trabajando con cualquier *framework* de metaheurísticas existente, implementándolos en un lenguaje de su elección, a partir del código 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 Agrupamiento con Restricciones

El PAR es una generalización del problema del agrupamiento clásico. Permite incorporar al proceso de agrupamiento un nuevo tipo de información: las restricciones. Dado un conjunto de datos X con n instancias, el problema consiste en encontrar una partición C={c_1, c_2, …, c_n} que agrupe dichas instancias en k grupos de forma que se minimice la distancia media entre las instancias del mismo grupo y que se cumplan las restricciones de instancia definidas en el conjunto R al máximo posible. Dichas restricciones implican que existen una serie de parejas de instancias que deben pertenecer (*Must-Link, ML)* o que no pueden pertenecer (*Cannot-Link, CL)* al mismo grupo.

El objetivo es obtener una partición *solución* que permita minimizar la función *fitness* definida como $fitness(solucion) = distancia_{intra-cluster}(solucion) + \lambda * infeasibility(solucion)$. Como se puede observar, es un problema con dos objetivos que se integran en una función mono-objetivo mediante una suma ponderada con el parámetro $\lambda \in [0,1)$, tal y como se describe en el Seminario 2. Para asegurar que el factor $infeasibility$ tiene la suficiente relevancia establecemos λ como **el cociente entre la distancia máxima existente en el conjunto de datos y el número de restricciones presentes en el problema, $|R|$.**

A continuación definimos cómo calcular tanto la distancia intra-cluster como la *infeasibility*. En primer lugar, dada una partición C, se puede calcular el centroide μ asociado a cada grupo c_i como el vector promedio de las instancias de X que lo componen:

$$\overrightarrow{\mu_i} = \frac{1}{|c_i|} \sum_{\overrightarrow{x_j} \in c_i} \overrightarrow{x_j}$$

Definimos la distancia media intra-cluster de un cluster C_i como la media de las distancias de las instancias que lo conforman a su centroide:

$$distancia_{intra-cluster}(C_i) = \frac{1}{|C_i|} \sum_{\overrightarrow{x_j} \in c_i} \|\overrightarrow{x_j} - \overrightarrow{\mu_i}\|_2$$

Definimos la desviación general de la partición C={c_1, c_2, …, c_n} como la media de las desviaciones intra-cluster de cada grupo o cluster:

$$distancia_{intra-cluster}(C) = \frac{1}{k} \sum_{c_i \in C} distancia_{intra-cluster}(C_i)$$

Dado que disponemos de estructuras para almacenar las restricciones (una matriz de restricciones MR o una lista de restricciones LR), se puede calcular el valor de $infeasibility$ de manera sencilla. Definimos primero $V(\overrightarrow{x_i}, \overrightarrow{x_j})$ como la función que, dada una pareja de instancias, devuelve 1 si la asignación de dichas instancias viola una restricción y 0 en otro caso. A partir de ella, calculamos $infeasibility$ como el número de restricciones incumplidas por una partición C del conjunto de datos X de acuerdo a la siguiente expresión:

$$infeasability = \sum_{i=0}^{n} \sum_{j=0}^{n} V(\overrightarrow{x_i}, \overrightarrow{x_j})$$

Sin embargo, dado que el porcentaje de restricciones puede ser mucho menor que el total de pares de instancias, en la mayoría de algoritmos (a excepción del algoritmo Greedy) es conveniente calcularlo de forma más directa usando dos vectores de pares de elementos, el de restricciones de grupos distintos *ML* y el de restricciones del mismo grupo *CL*:

$$infeasibility(C) = \sum_{i=0}^{|ML|} bool2entero\left(h_c(\overrightarrow{ML_{i,1}}) \neq hc(\overrightarrow{ML_{[i,2]}})\right) + \sum_{i=0}^{|CL|} bool2entero\left(h_c(\overrightarrow{CL_{i,1}}) = hc(\overrightarrow{CL_{[i,2]}})\right)$$

donde **bool2entero** es la función indicadora, que devuelve 1 en caso de ser cierta la expresión Booleana que toma como argumento y 0 en otro caso, y *ML* y *CL* son los conjuntos de restricciones *Must Link* and *Cannot Link*.

3.2. Conjuntos de Datos Considerados

El estudiante trabajará con **6 instancias del PAR** generadas a partir de los **3 conjuntos de datos** siguientes (obtenidos de la web http://www.ics.uci.edu/~mlearn/MLRepository.html, procesados en algún caso y disponibles en el espacio de PRADO y en la web de la asignatura):

1. **Zoo:** En donde hay que agrupar animales en función de 16 atributos (si tiene cola, …). Tiene 7 clases (k = 7) y 101 instancias.
2. **Glass**: En donde hay que agrupar distintos vidrios en función de 9 atributos sobre sus componentes químicos. Tiene 7 clases (k = 7) y 214 instancias.
3. **Bupa:** En donde hay que agrupar personas en función de sus hábitos de consumo de alcohol, definido con 5 atributos. Tiene 16 clases (k = 16) y 345 instancias.

A partir de cada conjunto de datos se obtienen 2 instancias distintas generando 2 conjuntos de restricciones, correspondientes al 10% y 20% del total de restricciones posibles. Los llamaremos CS_{10} y CS_{20}, respectivamente.

Los ficheros *.dat* corresponden a los conjuntos de datos. Los ficheros *.const* corresponden a los conjuntos de restricciones. Por ejemplo: zoo*_set.dat* es el fichero de datos de zoo, y los conjuntos de restricciones son zoo*_const_set_10.const* e zoo*_set_const_20.const*. Los ficheros de datos contienen en cada fila una instancia del conjunto en cuestión con sus características separadas por comas (sin espacios). Los ficheros de restricciones almacenan una matriz de restricciones con el formato descrito en las transparencias del Seminario 2, con los valores separados por comas y sin espacios.

3.3. 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 el espacio de PRADO y 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. 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.

En el PAR consideraremos 5 ejecuciones con semillas de generación de números aleatorios diferentes para cada algoritmo en cada conjunto de datos con su conjunto de restricciones. Esto quiere decir que **un algoritmo cualquiera, se ejecutará 5 veces por cada conjunto de datos, por lo que supondrá un total de *5×6=30* ejecuciones por algoritmo**.

El resultado final de las medidas de validación será calculado como la media de los 5 valores obtenidos para cada conjunto de datos y restricciones.
Para facilitar la comparación de algoritmos en las prácticas del PAR se considerarán cuatro estadísticos distintos denominados *Agregado, Tasa_inf, Tasa_C* y *Tiempo*:

- *Agregado* corresponde al valor de la función objetivo que está optimizando el algoritmo; es decir, al valor a minimizar proveniente de la fórmula $fitness(solucion) = distancia_{intra-cluster}(solucion) + \lambda * infeasibility(solucion)$. Igualmente, se indicará el valor medio de las 5 ejecuciones. Es el criterio usado por cada uno de los algoritmos para optimizar.
- *Tasa_inf* se corresponde al valor de *infeasability* promedio obtenido por cada método en cada ejecución en el conjunto de datos.
- *Error_distancia* se calcula como el valor promedio del valor absoluto de la distancia intra-cluster con respecto a un valor de referencia para dicho problema. Usamos esta medida ya que si se incumplen las restricciones es posible obtener una distancia intra-cluster mejor que si se cumpliesen, así que medimos cómo de cercana es la distancia intra-cluster frente a la óptima cuando se cumplen todas las restricciones.
- *Tiempo* se calcula como la media del tiempo de ejecución empleado por el algoritmo para resolver cada caso del problema (cada conjunto de datos con su conjunto de restricciones). Es decir, en nuestro caso es la media del tiempo empleado por las 5 ejecuciones.

En principio cuanto menor es el valor de *Agrupamiento* mejor guía el algoritmo la búsqueda. De todos modos, para valorar cómo de bueno es el algoritmo para el problema es necesario valorar la calidad de la solución final obtenida. Para ello, miraremos primordialmente *Tasa_Inf*, ya que cuanto menor sea obtiene agrupamientos más respetuosos con las restricciones. Del mismo modo, cuanto menor es el valor de *Error_distancia*, mejor calidad tiene también el algoritmo, porque genera agrupamientos más parecidos al ideal. Si dos métodos obtienen soluciones con la misma calidad (tienen valores de *Tasa_Inf* y *Error_distancia* 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_PAR_2020-21.xls*, disponible en la web de la asignatura, permite recopilar los estadísticos comentados para generar las tablas de resultados de la práctica.

4. Componentes del Algoritmo de Búsqueda Local

El algoritmo de BL tiene las siguientes componentes, varias de las cuales serán comunes a los algoritmos metaheurísticos de las siguientes prácticas:

- *Esquema de representación*: Se seguirá la representación entera basada en un vector *S* de tamaño *n* (número de instancias del conjunto de datos) con valores en *{1, ..., k}* (siendo *k* el número de agrupamientos, que coincidirá con el del clases del conjunto de datos real) que indican el cluster asociado a cada instancia, explicado en las transparencias del seminario.
- *Función objetivo*: Será la expresión ya explicada, $fitness(solucion) = distancia_{intra-cluster}(solucion) + \lambda * infeasibility(solucion)$. El valor de λ está explicado en las transparencias del Seminario 2.
- *Generación de la solución inicial*: La solución inicial se generará de forma aleatoria escogiendo un valor en *{1, ..., k}* en cada posición del vector *S*, **comprobando que cada cluster tiene al menos una instancia asignada**.

- *Esquema de generación de vecinos*: Se emplea el movimiento de cambio de cluster que altera el vector *S* sustituyendo el valor s_i en una posición *i*-ésima por otro valor $l \in \{1, ..., k\}$ y $l \neq s_i$, **comprobando que no deja ningún cluster vacío**.
- *Criterio de aceptación*: Se considera una mejora cuando disminuye el valor global de la función objetivo.
- *Exploración del entorno*: Se realizará una exploración en orden aleatorio del entorno en cada iteración.

Algoritmo

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

Valores de los parámetros y ejecuciones

Se detendrá la ejecución del algoritmo de BL cuando no se encuentre mejora en todo el entorno o cuando se hayan realizado 100000 evaluaciones de la función objetivo, es decir, en cuanto se cumpla alguna de las dos condiciones.

5. Algoritmo de Comparación: COPKM

El algoritmo escogido para ser comparado con las metaheurísticas implementadas en esta práctica (la BL) y en las siguientes es el *greedy* COPKM, que también deberá ser implementado por el estudiante. El objetivo de este algoritmo será generar un agrupamiento basado en el famoso algoritmo k-medias pero teniendo en cuenta la satisfacción de restricciones asociadas al conjunto de datos.

El algoritmo COPKM hace también una interpretación débil de las restricciones. El algoritmo k-medias clásico minimiza únicamente la desviación general. Al modificarlo para tener en cuenta las restricciones, **el algoritmo acomete un problema más complejo y puede tardar más en encontrar una solución**.

A continuación, se describe el algoritmo en los siguientes pasos:

- Se generan *k* centroides iniciales de forma aleatoria dentro del dominio asociado a cada dimensión del conjunto de datos.
- Se barajan los índices de las instancias para recorrerlas de forma aleatoria sin repetición.
- Se asigna cada instancia del conjunto de datos al grupo más cercano (aquel con centroide más cercano) que viole el menor número de restricciones *ML* y *CL*.
- Se actualizan los centroides de cada grupo de acuerdo al promedio de los valores de las instancias asociadas a su grupo.
- Se repiten los dos pasos anteriores mientras que al menos un grupo haya sufrido algún cambio.

Como la BL, el algoritmo COPKM deberá ser ejecutado 5 veces para cada conjunto de datos de acuerdo a lo explicado en la sección anterior.

6. Tablas de Resultados a Obtener

Se diseñará una tabla para cada algoritmo (COPKM, BL) y cada nivel de restricciones donde se recojan los resultados de la ejecución de dicho algoritmo en los conjuntos de datos considerados. Tendrá la misma estructura que la Tabla 6.1.

Tabla 6.1: Resultados obtenidos por el algoritmo X en el PAR con XX% de restricciones

	Zoo				**Glass**				**Bupa**			
	Tasa_Inf	***Error_Dist***	***Agr.***	**T**	***Tasa_Inf***	***Error_Dist***	***Agr.***	**T**	***Tasa_Inf***	***Error_Dist***	***Agr.***	**T**
Ejecución 1	x	x	x	x	x	x	x	x	x	x	x	x
Ejecución 2	x	x	x	x	x	x	x	x	x	x	x	x
Ejecución 3	x	x	x	x	x	x	x	x	x	x	x	x
Ejecución 4	x	x	x	x	x	x	x	x	x	x	x	x
Ejecución 5	x	x	x	x	x	x	x	x	x	x	x	x
Media	x	x	x	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 6.2. 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 6.2: Resultados globales en el PAR con XX% de restricciones

	Zoo				**Glass**				**Bupa**			
	Tasa_Inf	***Error_Dist***	***Agr.***	**T**	***Tasa_Inf***	***Error_Dist***	***Agr.***	**T**	***Tasa_Inf***	***Error_Dist***	***Agr.***	**T**
COPKM	x	x	x	x	x	x	x	x	x	x	x	x
BL	x	x	x	x	x	x	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 desviación general de la partición obtenidas (capacidad del algoritmo para obtener soluciones de calidad), número de restricciones no satisfechas y tiempos requeridos para obtener las soluciones (rapidez del algoritmo). En esta práctica se comparará el rendimiento de la metaheurística considerada, la BL, con respecto al algoritmo de referencia, el COPKM. En las siguientes prácticas se comparará también el rendimiento de las distintas metaheurísticas consideradas entre sí.

7. 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.

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 incluirá la descripción en pseudocódigo del método de exploración del entorno, el operador de generación de vecino y la generación de soluciones aleatorias empleadas en el algoritmo de BL.

f) Descripción en **pseudocódigo** de los algoritmos 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:

   a) 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**).

   b) Resultados obtenidos según el formato especificado.

   c) 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 (si se ha hecho).

Aunque lo esencial es el contenido, 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 de la plataforma PRADO.

8. 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.