# PROBLEMA DE APRENDIZAJE DE PESOS EN CARACTERÍSTICAS
# BÚSQUEDAS POR TRAYECTORIAS
# MEMORIA P3 METAHEURÍSTICAS

Ignacio Garach Vélez MH Viernes 17:30

11 de junio de 2022

## Índice

# 1. Introducción

En estas prácticas vamos a abordar la implementación y aplicación de varios algoritmos para el cálculo de pesos de calidad de un clasificador 1-NN de modo que se optimice tanto la tasa de acierto del problema de clasificación, como la simplicidad del clasificador, esto es, la detección de que características no influyen de forma determinante en la labor de clasificación y por tanto podrían no tenerse en cuenta.
Como preliminares introducimos brevemente el problema de clasificación. Se trata de construir un modelo que a partir de las características (datos descritos numéricamente) de un objeto sea capaz de predecir su clase (tipo de objeto entre un número finito de posibilidades). Para ello se entrena al modelo con un conjunto de datos correctamente clasificados (aprendizaje supervisado) y despues se prueba su valía con un conjunto independiente de test. Existen numerosos enfoques para resolver este problema con gran éxito, nosotros nos vamos a centrar en probablemente el más sencillo, se trata del algoritmo $1 - NN$ conocido popularmente como el del vecino más cercano.

Este algoritmo, simplemente almacena los datos de entrenamiento en memoria junto con su clase asociada, matemáticamente esto se puede representar como un conjunto de puntos de $\mathbf{R}^n$ junto con un entero representando su clase asociada. Y para cada punto que quiera clasificar, calcula la distancia euclídea con todos los puntos de entrenamiento y escoge la clase del punto para el cuál es mínima, su vecino más cercano.
Nuestro trabajo va a consistir en conseguir pesos que ponderen la importancia en la distancia final de cada característica en cada caso concreto de entrenamiento, es decir si la distancia euclídea usual es:

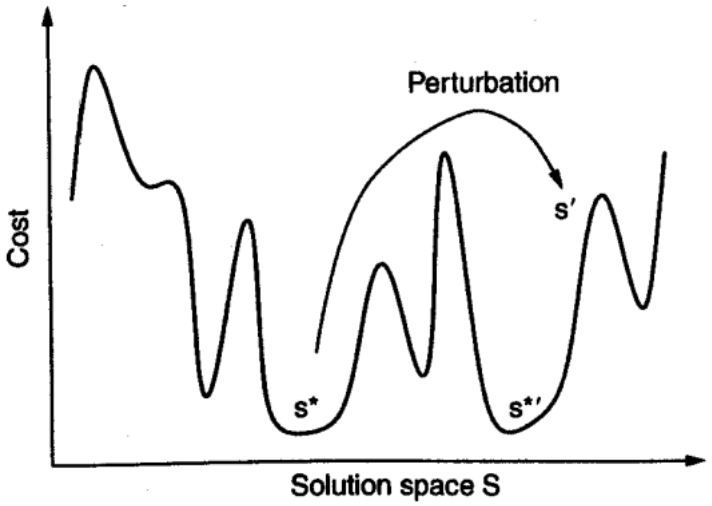
$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

nosotros consideramos el cálculo de los $w_i$ tal que sea:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} w_i (x_i - y_i)^2}$$

Nuestro objetivo es dual, por un lado queremos conseguir buenas tasas de acierto en el problema de clasificación y por otro obtener pesos bajos en las características menos influyentes para la clasificación. El problema de aprendizaje de pesos en características es determinar dichos pesos a partir del conjunto de datos de entrenamiento.
En la primera práctica se implementaron un algoritmo de búsqueda local con trayectorias simples y un algoritmo greedy (voraz) y se comparó su desempeño con el algoritmo 1-NN habitual.

En esta tercera práctica vamos a explorar diferentes técnicas de búsqueda basadas en trayectorias. Por un lado se implementará una búsqueda local con múltiples puntos de arranque escogidos aleatoriamente y por otro una reiteración de la misma con mutaciones fuertes. Son formas diferentes de ampliar la exploración de la superficie de optimización del fitness. Además implementaremos una técnica de optimización de inspiración física conocida como enfriamiento simulado, que permite aceptar soluciones peores con cierta probabilidad para evitar estancarse.




Escape de la búsqueda ILS

## 2. Datasets considerados

Para poder probar los algoritmos desarrollados se utilizarán los siguientes conjuntos de datos, todos ellos con clase binaria y algunos de ellos con distribución de clases no homogénea.

### 2.1. Ionosphere

Conjunto de datos recogidos en Goose Bay por un sistema con 16 antenas de alta frecuencia, el objetivo era captar electrones libres en la ionosfera, una devolución positiva indicaba la existencia de algún tipo de estructura. Los datos son las partes reales e imaginarias de 17 señales electromagnéticas complejas.

- 352 instancias
- 34 características
- 2 clases

### 2.2. Parkinsons

Conjunto de datos recogidos por el Centro Nacional de estudio de la voz de Oxford. Se trata de medidas asociadas a 195 grabaciones de voz de personas, algunas de ellas con enfermedad de Parkinson. La clase sería la presencia de dicha enfermedad.

- 195 instancias
- 22 características
- 2 clases

### 2.3. Spectf-Heart

Conjunto de datos de medidas asociadas a imágenes tomadas por tomografía computerizada cardíaca tomadas por el colegio de médicos de Ohio. Cada paciente se clasifica entre normal o con anomalías.

- 267 instancias
- 44 características
- 2 clases

## 3. Elementos comunes del problema

En este apartado vamos a describir las funciones que serán comunes durante todo el desarrollo de las prácticas de la asignatura:

- Para representar nuestros conjuntos de entrenamiento y test utilizaremos un vector de la STL de una nueva clase a la que llamaremos Instance que representará una instancia del problema, es decir su vector de características y su clase.
- Los datos proporcionados se encuentran en formato ARFF, un conjunto de cabeceras indicando las características y a continuación cada instancia es una fila con espacios como separadores y con la clase al final de la línea. Esto facilita su lectura para su posterior postprocesado. Hemos desarrollado para ello la funcion *read_arff* que recibe como parámetro el nombre del archivo y un par de vectores donde se almacenan las características y su clase. A posteriori en la función principal se genera el *vector de Instance* y se rellena con dichos datos.
- Los datos requieren ser normalizados para que no influyan las distintas escalas en nuestros algoritmos y no distorsionen los datos. Para esto se ha implementado la función *normalize* que modifica el vector de instancias, mediante escalado y translaciones hasta el intervalo $[0, 1]$, para ello se hace uso de la siguiente fórmula en cada característica:

$$x_i = \frac{x_i - mín(x_i)}{máx(x_i) - mín(x_i)}$$

- Para el cálculo de las distancias se implementa la función *weightedDistance* que recibe un vector de pesos y 2 vectores de características y devuelve la distancia ponderando por sus pesos.
- Para nuestro problema de aprendizaje de pesos, utilizaremos una versión del clasificador 1-NN que permitirá su generalización con la utilización de pesos. Recibe un valor booleano indicando si debe utilizar un vector de pesos al calcular las distancias. Atendiendo a estas características tendríamos el siguiente pseudocódigo:

Clasificador 1-NN (Instance, TrainingSet, Weighted?, Weights[])

Si no weighted
Entonces: Inicializar a 1 todos los pesos W
Para toda instancia de TrainingSet distinta de Instance (Leave One Out)
Hacer:

Calcular la distancia con respecto a Instance.
Si la distancia disminuye respecto al anterior guardamos su posición en pos.

Fin - Para todo

Devolver la clase de TrainingSet[pos]

### 3.1. Métricas utilizadas

Como medidas de la bondad de funcionamiento de los algoritmo se van a tomar las siguientes 4, relacionadas con los parámetros que se quieren optimizar, a saber, exactitud de clasificación, simplicidad y tiempo:

%**class** : Porcentaje de acierto en la labor de clasificación. Se utiliza el método de prueba basado en entrenamiento y test de 5 particiones cruzadas (se crean mediante la función *makeKFolds* inspirada en las prácticas de la asignatura de aprendizaje automático y que divide de forma homogénea y equilibrada las clases) y se realiza la media.

$$\%_{class} = 100\frac{\text{nº de instancias clasificadas correctamente en Test}}{\text{nº de instancias en Test}}$$

%**red** : Porcentaje de características que podrían obviarse, vamos a considerar aquellas cuyo peso asociado sea menor a 0.1. En efecto, para el algoritmo base no se reduce nada, porque todos los pesos son 1.

$$\%_{red} = 100\frac{\text{nº de características con peso menor que 0.1}}{\text{nº de características}}$$

**Agregado** : Valor de la función objetivo de nuestro problema. Se trata del promedio ponderado de las tasas de acierto y reducción, en nuestro caso consideramos una ponderación equiparada luego es simplemente la media de ambos valores

**Tiempo** : Tiempo de ejecución del algoritmo. En nuestro caso la media de 5 ejecuciones.

Para la implementación del cálculo de la tasa de clasificación se utilizará la función *score*, simplemente recorre el vector de predicciones del clasificador y cuenta los aciertos, devolviendo la división entre el número de instancias.
Para el cálculo de la tasa de reducción se hace algo análogo en *reduction* con el vector de pesos obtenido contando todos aquellos menores que 0.1 y devolviendo la división entre el número de características.
La función objetivo agregada se implementa de forma génerica con un parámetro $\alpha$ que pondera la importancia de precisión y simplicidad, aunque en nuestro caso usaremos $\alpha = 0.5$ luego coincidirá con la media de *score* y *reduction*, esto se realiza en *funcionObjetivo*.
Por último para el cálculo de tiempos se usará la librería *chrono* de C++ tomando el tiempo actual antes y después de las ejecuciones con *high_resolution_clock* y calculando su diferencia. Lo mediremos en segundos.

# 4. Búsqueda Multiarranque Básica

Una búsqueda con arranque múltiple trata de parecerse más a un algoritmo de búsqueda global pero mediante el uso de búsqueda local. Se trata de elegir soluciones iniciales que cubran (probablemente) más uniformemente el espacio de búsqueda y lanzar la búsqueda local a partir de cada una de ellas. El resultado final será el mejor de los resultados obtenidos a la finalización de todas las búsquedas locales. La Búsqueda Multiarranque Básica se caracteriza porque las soluciones iniciales se generan de forma aleatoria.
Este método converge al óptimo global del problema con probabilidad 1 cuando el número de soluciones iniciales generadas tienden a infinito. El problema de este algoritmo es que podrían generarse puntos muy cercanos entre sí de forma que se estuviera viendo comprometida la noción de exploración y la búsqueda local acabara con soluciones muy parecidas entre sí.
Para esta práctica se ha implementado un struct Solution que sustituye al Cromosoma de la práctica anterior, simplemente almacena un vector de pesos y su valor de fitness. Se ha modificado la local_search de la práctica 1 sólo con el objetivo de adaptarla a este cambio, por tanto es innecesario detallarlo más.

Por tanto se tiene este sencillo pseudocódigo:

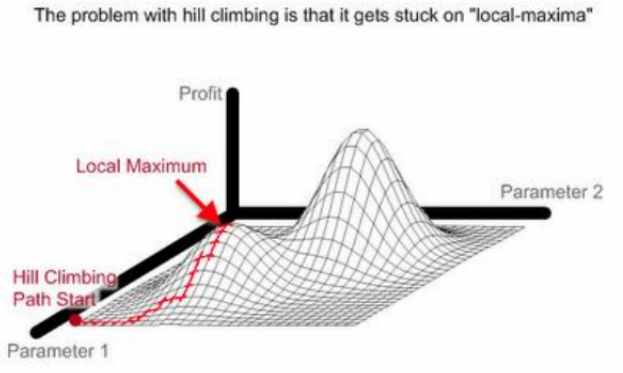
BMB(TrainingSet, Pesos)

```
Inicializar 15 soluciones aleatorias en un vector<Solution> inicios.
Para cada Solution en inicios (1 a 15):
    Local_Search(TrainingSet, inicios[i])
    (Ha quedado almacenado el resultado de la búsqueda en el vector)
Identificamos el mejor fitness y devolvemos dicha solución:
    best_sol = inicios[0]

    Para cada Solution en inicios (1 a 15):
        Si best_sol.fitness < inicios[i].fitness:
            best_sol = inicios[i]

Pesos = best_sol.pesos
```

En nuestro caso se realizarán 15 iteraciones, es decir, se generarán 15 soluciones iniciales aleatorias y se aplicará la Búsqueda Local sobre cada una de ellas. Cada aplicación de la búsqueda finalizará bien cuando no se encuentre mejora en todo el entorno o bien cuando se hayan realizado 1000 evaluaciones.

# 5. Enfriamiento Simulado

Tras la realización de las primeras prácticas tenemos claros los problemas de la búsqueda local, los óptimos locales, por tanto, una idea simple para solucionarlo es permitir que bajo determinadas condiciones el algoritmo avance hacia soluciones peores con cierta probabilidad, esta es la idea del enfriamiento simulado.


El Enfriamiento Simulado es un algoritmo de búsqueda por entornos con un criterio probabilístico de aceptación de soluciones basado en la leyes de la termodinámica. Como hemos mencionado, se tratará de permitir algunos movimientos hacia soluciones peores, pero estos movimientos de escape deben realizarse de un modo controlado.

En este caso se realizan mediante una función de probabilidad que hará disminuir la probabilidad de estos movimientos hacia soluciones peores conforme avanza la búsqueda (y supuestamente estemos más cerca del óptimo global). El enfoque es entonces claro, al principio se diversifica para una mejor exploración y al final se intensifica para una mejor explotación.

## 5.1. Modelo de Metrópoli

El fundamento de este control se basa en el trabajo de Metrópolis en el campo de la termodinámica estadística. Modeló el proceso de enfriamiento simulando los cambios energéticos en un sistema de partículas conforme decrece la temperatura, hasta que converge a un estado estable. Las leyes termodinámicas indican que a una temperatura $t$, la probabilidad de un incremento de energía $\delta E$ se aproxima por:

$$P[\delta E] = exp(\frac{-\delta E}{Kt})$$

En el modelo, se genera una perturbación aleatoria en el sistema y se calculan los cambios de energía resultantes:

- Si hay una caída energética, el cambio se acepta automáticamente.
- Si se produce un incremento energético, el cambio será aceptado con una probabilidad indicada por la expresión.

Para adaptarlo a una metaheurística, haremos uso de una variable llamada temperatura, cuyo valor indica cuando pueden ser aceptadas soluciones vecinas peores.

## 5.2. Cálculo de la temperatura inicial

Hay que tener en cuenta que estamos maximizando la función objetivo, luego hay que ajustar la resta del fitness de la solución anterior y de la mutada.

También debe adaptarse el cálculo de la temperatura inicial por esto, por ello la fórmula siguiente resulta en:

$$T_0 = \frac{\mu C(S_0)}{-ln(\phi)} = \frac{0.3(1 - \frac{sol_{ini}.fitness}{100})}{-ln(0.3)}$$

La temperatura final $T_{fin}$ se fijará a $10^{-3}$ comprobando siempre que sea menor que la temperatura inicial.

### 5.3. Esquema de enfriamiento

Se seguirá el esquema de Cauchy modificado para controlar el número de iteraciones, en el que para M iteraciones, en nuestro caso haremos:

$$T_{k+1} = \frac{T_k}{1 + \beta T_k}$$

$$\beta = \frac{T_0 - T_{fin}}{M T_0 T_{fin}}$$

Enfriamiento Simulado(TrainingSet, Pesos)

```
Iteraciones = 0
Inicializo Solution sol y la evaluo
Iteraciones++
Best_sol = sol
initial_temp = (mu*(1 - Best_sol.fitness / 100.0)) / (-ln(phi))
temp = initial_temp

Mientras final_temp >= temp:
    final_temp = temp / 100
Fin-Mientras
    beta = (initial_temp - final_temp)/(M*initial_temp*final_temp)
    successful = MAX_SUCCESS
    iter = 1

    Mientras Iteraciones<MAX_ITER y succesful!=0:
        vecino = 0
        succesful = 0
        Mientras no sobrepasemos iter, vecinos y mejoras:
            sol_mut = Mutar sol
            Recalcular fitness
            Iteraciones++
            Vecinos++

            diff = sol.fitness - sol_mut.fitness
            Si diff==0: (evito aceptar siempre que la diferencia es 0)
                diff = 0.001
            Si diff<0 o Aleatorio <= exp(-diff/temp):
                succesful++
                sol = sol_mut
                Si sol.fitness > best_sol.fitness:
                    best_sol = sol
        Fin-Mientras
        temp = temp/(1 + beta * temp) ESQUEMA DE CAUCHY
    Fin-Mientras
Pesos = best_sol.pesos
```

# 6. Búsqueda Local Reiterada

Este algoritmo esta basado en aplicar repetidamente la búsqueda local a una solución inicial obtenida mediante mutación de un óptimo local que ya se hubiera encontrado previamente. En nuestro caso consistirá en generar una solución aleatoria y aplicar búsqueda local sobre ella, despues de eso se comprueba si es mejor que la mejor hasta ese momento y se realizará una mutación sobre la mejor de esas 2. Posteriormente se volverá a lanzar búsqueda local sobre esta mutación. Repetiremos el proceso un número de veces y se devolverá la mejor encontrada en toda la búsqueda, luego el criterio de aceptación ILS será el criterio del mejor.

Además se añade un nuevo parámetro al método de mutación de anteriores prácticas para poder pasarle la varianza en el caso de la mutación especial de ILS.

## 6.1. Operador de mutación fuerte

Para distinguir este algoritmo de una búsqueda local, se realizan mutaciones mucho más perturbadoras sobre las soluciones que calcule la búsqueda local habitual, en nuestro caso vamos a mutar un 10 por ciento de los pesos en cada mutación, mediante el operador de mutación mediante una distribución normal, pero en este caso será de media 0 y varianza más alta, usaremos $\sigma = 0.4$.

Iterated Local Search(TrainingSet, Pesos)

```
Inicializar Solution s
Local_Search(TrainingSet, s)

Para un total de 15 iteraciones, hacer:
    Mutar fuertemente, esto es:
    s_mut = Mutar s en un 10 por ciento de componentes sin
            repeticiones con varianza 0.4

    Reiterar Local_Search:
        Calcular Fitness de s_mut
        Local_Search(TrainingSet, s_mut)

        Si s_mut.fitness > s.fitness:
            s = s_mut
Fin

Pesos = s.pesos
```

## 7. Hibridación de ILS y ES

En el contexto de estos algoritmos cabe plantearse una idea sencilla, sustituir en ILS la búsqueda local por el enfriamiento simulado. Esto tiene el objetivo de aumentar la variabilidad de soluciones mutando de dos formas diferentes a lo largo del algoritmo y así explorar mucho más el dominio. Adaptamos para ello los parámetros del enfriamiento poniendo 1000 iteraciones y un máximo de vecinos por cada una a 1, obtenemos el siguiente código.

Híbrido(TrainingSet, Pesos)

```
Inicializar Solution s
EnfriamientoSimuladoAuxiliar(TrainingSet, s)

Para un total de 15 iteraciones, hacer:
    Mutar fuertemente, esto es:
    s_mut = Mutar s en un 10 por ciento de componentes sin
            repeticiones con varianza 0.4

    Reiterar Enfriamiento Simulado:
        Calcular Fitness de s_mut
        EnfriamientoSimuladoAuxiliar(TrainingSet, s_mut)

        Si s_mut.fitness > s.fitness:
            s = s_mut
Fin

Pesos = s.pesos
```

## 8. Implementación y guía de usuario

Para la implementación se ha utilizado el lenguaje C++, las librerías habituales de entrada y salida y lectura de fichero, toma de tiempos y funciones auxiliares. Se usó un generador de números aleatorios inicializado con la semilla 2022.

Se ha especificado por secciones la estructura en funciones del código, por simplicidad se ha optado por implementar todo en un mismo archivo sin usar cabeceras.

Se utiliza optimización en la compilación con g++, concretamente se usa la siguiente orden para compilar:

g++ -std=c++11 -O3 main.cpp

Para ejecutar se llama en terminal a ./a.out seguido de la semilla que se desee.

# 9. Análisis y presentación de resultados

Recordemos en primer lugar los resultados obtenidos en la práctica 1:

## 9.1. 1NN, Relief y Búsqueda Local

Trás la ejecución de los algoritmos, encontramos que era preocupante el tiempo de ejecución del algoritmo de búsqueda local, pues se elevaban a 30 minutos todas las ejecuciones. Sin embargo añadiendo optimización O3 al compilar lo reducimos en torno a los 5 minutos totales de ejecución. Se obtienen entonces los siguientes resultados, las primeras 5 filas son sobre las particiones y la sexta es la media:

Cuadro 1: Cuadro de resultados para 1-NN

| **Ionosphere** | | | | **Parkinsons** | | | | **Spectf-heart** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** |
| 84.51 | 0.00 | 42.25 | 0.003 | 95.00 | 0.00 | 47.50 | 0.00 | 58.57 | 0.00 | 29.29 | 0.002 |
| 87.32 | 0.00 | 43.66 | 0.003 | 92.50 | 0.00 | 46.25 | 0.00 | 71.43 | 0.00 | 35.71 | 0.002 |
| 87.32 | 0.00 | 43.66 | 0.003 | 95.00 | 0.00 | 47.50 | 0.00 | 70.00 | 0.00 | 35.00 | 0.002 |
| 91.55 | 0.00 | 45.77 | 0.001 | 97.50 | 0.00 | 48.75 | 0.00 | 65.71 | 0.00 | 32.86 | 0.002 |
| 80.60 | 0.00 | 40.30 | 0.001 | 100.00 | 0.00 | 50.00 | 0.00 | 63.77 | 0.00 | 31.88 | 0.002 |
| 86.26 | 0.00 | 43.13 | 0.002 | 96.00 | 0.00 | 48.00 | 0.00 | 65.90 | 0.00 | 32.95 | 0.002 |

Cuadro 2: Cuadro de resultados para Greedy Relief

| **Ionosphere** | | | | **Parkinsons** | | | | **Spectf-heart** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** |
| 83.10 | 2.94 | 43.02 | 0.01 | 95.00 | 0.00 | 47.50 | 0.002 | 72.86 | 13.64 | 43.25 | 0.01 |
| 90.14 | 2.94 | 46.54 | 0.01 | 92.50 | 0.00 | 46.25 | 0.002 | 74.29 | 20.45 | 47.37 | 0.01 |
| 88.73 | 2.94 | 45.84 | 0.01 | 97.50 | 0.00 | 48.75 | 0.002 | 74.29 | 20.45 | 47.37 | 0.01 |
| 90.14 | 2.94 | 46.54 | 0.01 | 95.00 | 0.00 | 47.50 | 0.002 | 74.29 | 22.73 | 48.51 | 0.01 |
| 80.60 | 2.94 | 41.77 | 0.01 | 100.00 | 0.00 | 50.00 | 0.002 | 71.01 | 31.82 | 51.42 | 0.01 |
| 86.54 | 2.94 | 44.74 | 0.01 | 96.00 | 0.00 | 48.00 | 0.002 | 73.35 | 21.82 | 47.58 | 0.01 |

Cuadro 3: Cuadro de resultados para búsqueda local

| **Ionosphere** | | | | **Parkinsons** | | | | **Spectf-heart** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** |
| 94.37 | 88,24 | 91.30 | 18.52 | 90.00 | 95.45 | 92.73 | 4.52 | 65.71 | 75.00 | 70.36 | 28.28 |
| 90.14 | 85.29 | 87.72 | 19.53 | 87.50 | 63.64 | 75.57 | 1.48 | 77.14 | 70.45 | 73.80 | 26.08 |
| 84.51 | 88.24 | 86.37 | 18.72 | 97.50 | 77.27 | 87.39 | 2.807 | 74.29 | 68.18 | 71.23 | 18.67 |
| 87.32 | 79.41 | 83.37 | 18.96 | 92.50 | 100.00 | 96.25 | 3.761 | 75.71 | 68.18 | 71.95 | 24.53 |
| 88.06 | 91.18 | 89.62 | 18.21 | 97.14 | 77.27 | 87.21 | 4.197 | 71.01 | 72.73 | 71.87 | 18.26 |
| 88.88 | 86.47 | 87.68 | 18.79 | 92.93 | 82.73 | 87.83 | 3.353 | 72.77 | 70.91 | 71.84 | 23.16 |

En primer lugar, observamos que el algoritmo 1-NN obtiene buenos resultados para Ionosphere, excelentes para Parkinsons y bastante malos para Spectf-Heart (un clasificador aleatorio tendría teóricamente un 50 por ciento de acierto). Evidentemente no puede reducir pesos por construcción, luego la función objetivo está acotada superiormente por 0.5.

Al comparar con el greedy Relief observamos que en general mejora ligeramente los porcentajes de acierto en clasificación para los 3 datasets. Por otro lado, reduce una característica en Ionosphere aunque era trivial de reducir pues si observamos los datos en crudo, casi todos sus valores son iguales

Cuadro 4: Cuadro resumen de resultados generales

| | **Ionosphere** | | | | **Parkinsons** | | | | **Spectf-heart** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** |
| **1-NN** | 86.26 | 0.00 | 43.13 | 0.002 | 96.00 | 0.00 | 48.00 | 0.002 | 65.90 | 0.00 | 32.95 | 0.002 |
| **RELIEF** | 86.54 | 2.94 | 44.74 | 0.01 | 96.00 | 0.00 | 48.00 | 0.002 | 73.35 | 21.82 | 47.58 | 0.01 |
| **BL** | 88.88 | 86.47 | 87.68 | 18.79 | 92.93 | 82.73 | 87.83 | 3.35 | 72.77 | 70.91 | 71.84 | 23.16 |

y por tanto no discriminan y pueden eliminarse. En Parkinson no logra reducir y por tanto se limita mucho la función objetivo. Finalmente en Spectf-Heart si logra reducir en torno a un 20 por ciento de características pero sigue obteniendo mala función objetivo agregada pues se le da gran importancia a la reducción en la ponderación.

Para búsqueda local la desventaja es el tiempo de ejecución, aunque podría acotarse cambiando las condiciones de parada. Para nuestros datasets es asumible. Observamos que se encuentran resultados mucho mejores que con Relief en todos los datasets, en Ionosphere y Parkinsons, la función objetivo se dispara casi al 90 por ciento, consiguiendose además reducir en torno al 80 por ciento de características. En Spectf-Heart conseguimos un resultado similar pero no tan bueno, nuestra tasa de clasificación sube al 72 por ciento y podemos descartar un 70 por ciento de características.

Por ello consideramos que es un buen punto de partida el algoritmo de búsqueda local para compararlo con los modelos bioinspirados de las siguientes prácticas y ver su funcionamiento, converge a un óptimo local al menos para nuestros casos aunque pueda tener periodos de oscilación por como se mutan las componentes, el algoritmo Greedy por su parte es muy rápido y genera una buena primera aproximación, aunque como damos mucha importancia a la simplificación de características no obtiene buenos resultados en estos datasets.

## 9.2. Algoritmos genéticos

Pasamos a mostrar y comentar los resultados de la ejecución de los algoritmos genéticos implementados en esta práctica. Como nomenclatura adicional en la última tabla añadimos el sufijo 1 para las versiones del algoritmo que utilicen el operador de cruce BLX y el sufijo 2 para el cruce aritmético.

Cuadro 5: Cuadro de resultados para AGG con cruce BLX

| **Ionosphere** | | | | **Parkinsons** | | | | **Spectf-heart** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** |
| 77.46 | 70.59 | 74.03 | 118.73 | 90.00 | 86.36 | 88.18 | 30.98 | 72.86 | 59.09 | 65.97 | 126.74 |
| 88.73 | 73.53 | 81.13 | 119.36 | 95.00 | 72.73 | 83.86 | 30.88 | 85.71 | 56.82 | 71.27 | 126.74 |
| 83.10 | 67.65 | 75.37 | 120.57 | 85.00 | 90.91 | 87.95 | 30.91 | 67.14 | 68.18 | 67.66 | 127.12 |
| 90.14 | 61.76 | 75.95 | 117.26 | 85.00 | 77.27 | 81.14 | 30.98 | 68.57 | 61.36 | 64.97 | 126.05 |
| 88.06 | 73.53 | 80.79 | 121.82 | 94.29 | 81.82 | 88.05 | 32.96 | 73.91 | 56.82 | 65.37 | 126.79 |
| 85.50 | 69.41 | 77.46 | 119.55 | 89.86 | 81.82 | 85.84 | 31.34 | 73.64 | 60.45 | 67.05 | 126.69 |

Cuadro 6: Cuadro de resultados para AGG con cruce aritmético

| **Ionosphere** | | | | **Parkinsons** | | | | **Spectf-heart** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** |
| 90.14 | 58.82 | 74.48 | 119.41 | 87.50 | 72.73 | 80.11 | 30.97 | 74.29 | 59.09 | 66.69 | 126.87 |
| 95.77 | 67.65 | 81.71 | 118.27 | 85.00 | 81.82 | 83.41 | 30.76 | 71.43 | 56.82 | 64.12 | 126.12 |
| 85.92 | 67.65 | 76.78 | 119.24 | 100.00 | 77.27 | 88.64 | 30.86 | 74.29 | 54.55 | 64.42 | 126.57 |
| 85.92 | 64.71 | 75.31 | 119.03 | 92.50 | 72.73 | 82.61 | 31.03 | 72.86 | 56.82 | 64.84 | 127.58 |
| 85.07 | 67.65 | 76.36 | 121.56 | 82.86 | 72.73 | 77.79 | 33.04 | 55.07 | 56.82 | 55.95 | 127.14 |
| 88.56 | 65.29 | 76.93 | 119.50 | 89.57 | 75.45 | 82.51 | 31.33 | 69.59 | 56.82 | 63.20 | 126.86 |

En primer lugar con respecto a los AGG se observa que el cruce BLX considera resultados consistentemente mejores que el cruce aritmético para todos los datasets y sobre todo se acentúa en Parkinsons y Spectf-Heart. Creemos que esto es debido a la capacidad de exploración extra que tiene el cruce BLX fuera del intervalo de los padres mientras que el aritmético esta delimitado de forma estricta por los padres.
Por otro lado, si comparamos con el mejor algoritmo de la P1 que era búsqueda local, los resultados son algo peores con AGG-BLX que con búsqueda local, aunque similares. Sobre todo la tasa de clasificación es igual de buena pero reduce algo menos perdiendo por tanto mucho fitness al no conseguir tanta simplicidad como la búsqueda local. En los tiempos salen perdiendo claramente los genéticos aunque no por ello deben descartarse ya que factores como la naturaleza de los datasets, la sesgadez de las particiones de validación y los hiperparámetros usados en los algoritmos. Entran por tanto en juego los teoremas de No-Free-lunch para darnos certeza en que es probable que estos algoritmos sean útiles para otros problemas.

Cuadro 7: Cuadro de resultados para AGE con cruce BLX

| **Ionosphere** | | | | **Parkinsons** | | | | **Spectf-heart** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** |
| 85.92 | 97.06 | 91.49 | 118.86 | 95.00 | 100.00 | 97.50 | 30.87 | 65.71 | 70.45 | 68.08 | 127.50 |
| 95.77 | 94.12 | 94.95 | 119.25 | 95.00 | 100.00 | 97.50 | 30.87 | 81.43 | 77.27 | 79.35 | 126.07 |
| 91.55 | 97.06 | 94.30 | 119.53 | 92.50 | 100.00 | 96.25 | 31.98 | 77.14 | 70.45 | 73.80 | 126.85 |
| 88.73 | 91.18 | 89.95 | 118.61 | 92.50 | 100.00 | 96.25 | 31.15 | 81.43 | 75.00 | 78.21 | 126.14 |
| 82.09 | 94.12 | 88.10 | 124.78 | 100.00 | 100.00 | 100.00 | 33.19 | 81.16 | 75.00 | 78.08 | 126.99 |
| 88.81 | 94.71 | 91.76 | 120.21 | 95.00 | 100.00 | 97.50 | 31.61 | 77.37 | 73.64 | 75.51 | 126.71 |

Cuadro 8: Cuadro de resultados para AGE con cruce aritmético

| **Ionosphere** | | | | **Parkinsons** | | | | **Spectf-heart** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** |
| 88.73 | 100.00 | 94.37 | 119.65 | 85.00 | 95.45 | 90.23 | 31.26 | 70.00 | 79.55 | 74.77 | 127.05 |
| 94.37 | 97.06 | 95.71 | 119.32 | 95.00 | 100.00 | 97.50 | 30.99 | 74.29 | 72.73 | 73.51 | 125.95 |
| 90.14 | 97.06 | 93.60 | 119.08 | 95.00 | 100.00 | 97.50 | 30.96 | 74.29 | 79.55 | 76.92 | 126.03 |
| 90.14 | 100.00 | 95.07 | 121.14 | 87.50 | 100.00 | 93.75 | 31.01 | 75.71 | 75.00 | 75.36 | 127.65 |
| 83.58 | 100.00 | 91.79 | 122.69 | 100.00 | 100.00 | 100.00 | 33.40 | 73.91 | 79.55 | 76.73 | 128.89 |
| 89.39 | 98.82 | 94.11 | 120.38 | 92.50 | 99.09 | 95.80 | 31.52 | 73.64 | 77.27 | 75.46 | 127.11 |

Por otro lado con respecto a las variantes estacionarias, se observa que los cruces obtienen resultados muy similares, quizá algo mejor el cruce aritmético. Esto puede ser debido a que como el elitismo mantiene a los mejores en la población se explota muy bien una zona muy prometedora del espacio de soluciones. Se produce un fenómeno curioso con las tasas de reducción que llegan al 100 por 100 aunque si observamos dichos pesos pese a estar por debajo del umbral de 0.1 realmente son parejos, por tanto se esta produciendo un escalado para que mejore el fitness pero realmente la tasa de reducción está hinchada y no simplifica tanto como indica ya que no se podrían eliminar todas las características evidentemente.

En resumen esta última tabla nos muestra una importante superioridad de AGE frente a AGG e incluso frente a búsqueda local, sobre todo en Ionosphere y Parkinsons aunque como hemos comentado antes las tasas de reducción pueden producir resultados algo engañosos.

Cuadro 9: Cuadro resumen de resultados generales

| | **Ionosphere** | | | | **Parkinsons** | | | | **Spectf-heart** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** |
| **1-NN** | 86.26 | 0.00 | 43.13 | 0.002 | 96.00 | 0.00 | 48.00 | 0.002 | 65.90 | 0.00 | 32.95 | 0.002 |
| **RELIEF** | 86.54 | 2.94 | 44.74 | 0.01 | 96.00 | 0.00 | 48.00 | 0.002 | 73.35 | 21.82 | 47.58 | 0.01 |
| **BL** | 88.88 | 86.47 | 87.68 | 18.79 | 92.93 | 82.73 | 87.83 | 3.35 | 72.77 | 70.91 | 71.84 | 23.16 |
| **AGG1** | 85.50 | 69.41 | 77.46 | 119.55 | 89.86 | 81.82 | 85.84 | 31.34 | 73.64 | 60.45 | 67.05 | 126.69 |
| **AGG2** | 88.56 | 65.29 | 76.93 | 119.50 | 89.57 | 75.45 | 82.51 | 31.33 | 69.59 | 56.82 | 63.20 | 126.86 |
| **AGE1** | 88.81 | 94.71 | 91.76 | 120.21 | 95.00 | 100.00 | 97.50 | 31.61 | 77.37 | 73.64 | 75.51 | 126.71 |
| **AGE2** | 89.39 | 98.82 | 94.11 | 120.38 | 92.50 | 99.09 | 95.80 | 31.52 | 73.64 | 77.27 | 75.46 | 127.11 |

## 9.3. Algoritmos meméticos

Mostramos en esta sección los resultados de los algoritmos meméticos. La nomenclatura utilizada en las tablas es AM1 para AM(10, 1), AM2 para AM(10, 0.1) y AM3 para AM(10, 0.1M).

Cuadro 10: Cuadro de resultados para AM1 con cruce BLX

| **Ionosphere** | | | | **Parkinsons** | | | | **Spectf-heart** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** |
| 92.96 | 94.12 | 93.54 | 122.58 | 92.50 | 86.36 | 89.43 | 30.78 | 77.14 | 75.00 | 76.07 | 130.32 |
| 95.77 | 85.29 | 90.53 | 123.18 | 87.50 | 90.91 | 89.20 | 30.67 | 77.14 | 79.55 | 78.34 | 131.16 |
| 90.14 | 82.35 | 86.25 | 122.44 | 92.50 | 95.45 | 93.98 | 30.90 | 74.29 | 65.91 | 70.10 | 130.54 |
| 94.37 | 91.18 | 92.77 | 123.42 | 92.50 | 100.00 | 96.25 | 30.66 | 77.14 | 70.45 | 73.80 | 129.95 |
| 88.06 | 94.12 | 91.09 | 126.89 | 97.14 | 90.91 | 94.03 | 32.73 | 76.81 | 72.73 | 74.77 | 131.63 |
| 92.26 | 89.41 | 90.84 | 123.70 | 92.43 | 92.73 | 92.58 | 31.15 | 76.51 | 72.73 | 74.62 | 130.72 |

Cuadro 11: Cuadro de resultados para AM2 con cruce BLX

| **Ionosphere** | | | | **Parkinsons** | | | | **Spectf-heart** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** |
| 91.55 | 88.24 | 89.89 | 119.16 | 95.00 | 95.45 | 95.23 | 30.76 | 75.71 | 77.27 | 76.49 | 125.37 |
| 95.77 | 79.41 | 87.59 | 118.59 | 92.50 | 86.36 | 89.43 | 30.73 | 74.29 | 70.45 | 72.37 | 124.91 |
| 88.73 | 76.47 | 82.60 | 119.94 | 92.50 | 86.36 | 89.43 | 30.74 | 68.57 | 81.82 | 75.19 | 125.11 |
| 91.55 | 88.24 | 89.89 | 118.76 | 95.00 | 90.91 | 92.95 | 30.62 | 75.71 | 68.18 | 71.95 | 124.80 |
| 88.06 | 88.24 | 88.15 | 123.40 | 100.00 | 90.91 | 95.45 | 32.46 | 60.87 | 79.55 | 70.21 | 126.45 |
| 91.13 | 84.12 | 87.63 | 119.97 | 95.00 | 90.00 | 92.50 | 31.06 | 71.03 | 75.45 | 73.24 | 125.33 |

Cuadro 12: Cuadro de resultados para AM3 con cruce BLX

| **Ionosphere** | | | | **Parkinsons** | | | | **Spectf-heart** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** |
| 84.51 | 88.24 | 86.37 | 118.04 | 90.00 | 90.91 | 90.45 | 30.73 | 67.14 | 70.45 | 68.80 | 125.09 |
| 88.73 | 82.35 | 85.54 | 118.95 | 87.50 | 95.45 | 91.48 | 30.78 | 78.57 | 75.00 | 76.79 | 125.25 |
| 92.96 | 88.24 | 90.60 | 117.13 | 92.50 | 95.45 | 93.98 | 30.83 | 82.86 | 68.18 | 75.52 | 125.22 |
| 92.96 | 82.35 | 87.66 | 117.64 | 82.50 | 95.45 | 88.98 | 30.58 | 77.14 | 75.00 | 76.07 | 125.28 |
| 80.60 | 97.06 | 88.83 | 121.91 | 97.14 | 100.00 | 98.57 | 32.56 | 69.57 | 77.27 | 73.42 | 126.48 |
| 87.95 | 87.65 | 87.80 | 118.74 | 89.93 | 95.45 | 92.69 | 31.10 | 75.06 | 73.18 | 74.12 | 125.46 |

Cuadro 13: Cuadro resumen de resultados generales

| | **Ionosphere** | | | | **Parkinsons** | | | | **Spectf-heart** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** |
| **1-NN** | 86.26 | 0.00 | 43.13 | 0.002 | 96.00 | 0.00 | 48.00 | 0.002 | 65.90 | 0.00 | 32.95 | 0.002 |
| **RELIEF** | 86.54 | 2.94 | 44.74 | 0.01 | 96.00 | 0.00 | 48.00 | 0.002 | 73.35 | 21.82 | 47.58 | 0.01 |
| **BL** | 88.88 | 86.47 | 87.68 | 18.79 | 92.93 | 82.73 | 87.83 | 3.35 | 72.77 | 70.91 | 71.84 | 23.16 |
| **AGG1** | 85.50 | 69.41 | 77.46 | 119.55 | 89.86 | 81.82 | 85.84 | 31.34 | 73.64 | 60.45 | 67.05 | 126.69 |
| **AGG2** | 88.56 | 65.29 | 76.93 | 119.50 | 89.57 | 75.45 | 82.51 | 31.33 | 69.59 | 56.82 | 63.20 | 126.86 |
| **AGE1** | 88.81 | 94.71 | 91.76 | 120.21 | 95.00 | 100.00 | 97.50 | 31.61 | 77.37 | 73.64 | 75.51 | 126.71 |
| **AGE2** | 89.39 | 98.82 | 94.11 | 120.38 | 92.50 | 99.09 | 95.80 | 31.52 | 73.64 | 77.27 | 75.46 | 127.11 |
| **AM1** | 92.26 | 89.41 | 90.84 | 123.70 | 92.43 | 92.73 | 92.58 | 31.15 | 76.51 | 72.73 | 74.62 | 130.72 |
| **AM2** | 91.13 | 84.12 | 87.63 | 119.97 | 95.00 | 90.00 | 92.50 | 31.06 | 71.03 | 75.45 | 73.24 | 125.33 |
| **AM3** | 87.95 | 87.65 | 87.80 | 118.74 | 89.93 | 95.45 | 92.69 | 31.10 | 75.06 | 73.18 | 74.12 | 125.46 |

Destacamos que la comparación clara de estos algoritmos ha de ser con los algoritmos generacionales pues se basan en ellos, se nota claramente que la búsqueda local explotadora unida al poder explorador generacional genera un algoritmo con muy buenos resultados, mejores que ambas por separado (búsqueda local y AGGs). Se acercan también bastante a los logrados por las variantes estacionarias. Concluimos que como parecía indicar la lógica las mejores versiones meméticas son las que exploraban todos los cromosomas y la que exploraba a los mejores de ellos, quedando algo por detrás la aleatoria.

## 9.4. Algoritmos de búsqueda por trayectorias

En esta práctica, hemos obtenido algunos resultados bastante interesantes sobre todo en el aspecto temporal. Pasamos a mostrar las tablas de resultados. En primer lugar para búsqueda multiarranque básica y búsqueda local reiterada:

Cuadro 14: Cuadro de resultados para Búsqueda Multiarranque Básica

| **Ionosphere** | | | | **Parkinsons** | | | | **Spectf-heart** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** |
| 88.73 | 94.12 | 91.43 | 118.75 | 90.00 | 90.91 | 90.45 | 29.31 | 65.71 | 79.55 | 72.63 | 123.65 |
| 90.14 | 82.35 | 86.25 | 120.84 | 92.50 | 100.00 | 96.25 | 30.27 | 80.00 | 75.00 | 77.50 | 123.60 |
| 91.55 | 85.29 | 88.42 | 119.02 | 90.00 | 95.45 | 92.73 | 26.96 | 71.43 | 72.73 | 72.08 | 124.34 |
| 94.37 | 85.29 | 89.83 | 119.41 | 92.50 | 100.00 | 96.25 | 29.54 | 75.71 | 72.73 | 74.22 | 124.26 |
| 83.58 | 88.24 | 85.91 | 123.06 | 97.14 | 90.91 | 94.03 | 31.07 | 66.67 | 70.45 | 68.56 | 124.78 |
| 89.67 | 87.06 | 88.37 | 120.21 | 92.43 | 95.45 | 93.94 | 29.43 | 71.90 | 74.09 | 73.00 | 124.13 |

Observamos que la búsqueda multiarranque básica consigue consistentemente resultados ligeramente mejores que la búsqueda local que parece el *baseline* con el que comparar. En términos de rendimiento y tiempo necesita de bastante más tiempo en su ejecución la multiarranque y dependiendo de la aplicación en la que se quiera utilizar y del número de datos disponibles podría no compensar invertir tanto más tiempo, veremos más adelante otras técnicas de búsqueda multiarranque que mantengan una alta exploración con tiempos más razonables. Podría ser interesante comenzar la búsqueda desde aún más puntos de partida para aumentar la exploración aunque para obtener tiempos razonables deberían disminuir laa evaluaciones máximas complicando de este modo la explotación.Por otro lado, cabe destacar que en el dataset Parkinsons los tiempos son más razonables debido a su tamaño y el rendimiento aumenta de forma más notable, por ende se hace patente que todo dataset es particular y pueden funcionar algoritmos diferentes para ellos.

Cuadro 15: Cuadro de resultados para Búsqueda Local Reiterada

| **Ionosphere** | | | | **Parkinsons** | | | | **Spectf-heart** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** |
| 90.14 | 94.12 | 92.13 | 93.34 | 95.00 | 100.00 | 97.50 | 17.90 | 78.57 | 72.73 | 75.65 | 113.14 |
| 90.14 | 88.24 | 89.19 | 90.34 | 90.00 | 90.91 | 90.45 | 16.11 | 74.29 | 72.73 | 73.51 | 113.71 |
| 88.73 | 88.24 | 88.48 | 92.12 | 97.50 | 100.00 | 98.75 | 18.51 | 80.00 | 81.82 | 80.91 | 114.89 |
| 90.14 | 91.18 | 90.66 | 95.54 | 87.50 | 95.45 | 91.48 | 15.62 | 81.43 | 77.27 | 79.35 | 114.86 |
| 88.06 | 100.00 | 94.03 | 104.09 | 100.00 | 100.00 | 100.00 | 16.26 | 75.36 | 72.73 | 74.04 | 113.43 |
| 89.44 | 92.35 | 90.90 | 95.08 | 94.00 | 97.27 | 95.64 | 16.88 | 77.93 | 75.45 | 76.69 | 114.01 |

En cuanto a la búsqueda local reiterada, se observa que es bastante efectivo en tiempos inferiores que el algoritmo anterior, aunque aún resulta bastante más costoso que la búsqueda local. Es interesante reseñar que para Spectf-Heart se obtiene el mejor resultado de todas las prácticas y esto es muy particular porque dicho dataset era el más complicado en el sentido de tamaño y de la capacidad inicial del clasificador 1-NN. La explicación es el escape de óptimos locales mediante mutación fuerte, lo que permite obtener alta variabilidad a partir de soluciones buenas, esperando que en explotaciones posteriores se obtengan mejores aún.

Cuadro 16: Cuadro de resultados para Enfriamiento Simulado

| **Ionosphere** | | | | **Parkinsons** | | | | **Spectf-heart** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** |
| 88.73 | 88.24 | 88.48 | 117.75 | 92.50 | 100.00 | 96.25 | 30.62 | 70.00 | 61.36 | 65.68 | 10.68 |
| 88.73 | 100.00 | 94.37 | 118.71 | 92.50 | 100.00 | 96.25 | 30.41 | 74.29 | 68.18 | 71.23 | 22.72 |
| 85.92 | 100.00 | 92.96 | 118.27 | 97.50 | 100.00 | 98.75 | 30.32 | 74.29 | 50.00 | 62.14 | 11.55 |
| 91.55 | 100.00 | 95.77 | 117.78 | 95.00 | 100.00 | 97.50 | 30.40 | 74.29 | 56.82 | 65.55 | 11.58 |
| 80.60 | 100.00 | 90.30 | 120.17 | 100.00 | 100.00 | 100.00 | 32.16 | 69.57 | 72.73 | 71.15 | 18.94 |
| 87.11 | 97.65 | 92.38 | 118.53 | 95.50 | 100.00 | 97.75 | 30.78 | 72.48 | 61.82 | 67.15 | 15.10 |

El algoritmo de enfriamiento simulado obtiene muy buenos resultados tanto en Ionosphere y Parkinsons, de hecho en este último obtiene los mejores resultados de entre todos los algoritmos considerados en las prácticas, además en unos tiempos muy razonables, incluso inferiores en la búsqueda local. De nuevo se da un resultado que ocurría en los algoritmos genéticos y es que las tasas de reducción llegan al 100 por 100 pero con pesos parejos, luego ocurre un escalado para que mejore el fitness y no tanto la simplicidad. El compromiso entre exploración y explotación es parejo porque el algoritmo permite fugarse de óptimos locales con cierta probabilidad a lo largo del algoritmo. Un estudio del esquema de Cauchy mas en profundidad podría mejorar los resultados porque es posible que estemos enfriando demasiado rápido o demasiado lento, impidiendo introducir suficiente diversidad o introduciendo demasiada.

Cuadro 17: Cuadro de resultados para Hibridación de Búsqueda Reiterada y Enfriamiento Simulado

| **Ionosphere** | | | | **Parkinsons** | | | | **Spectf-heart** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** |
| 88.73 | 100.00 | 94.37 | 118.37 | 92.50 | 100.00 | 96.25 | 30.20 | 75.71 | 59.09 | 67.40 | 40.58 |
| 94.37 | 100.00 | 97.18 | 117.87 | 92.50 | 100.00 | 96.25 | 6.53 | 77.14 | 70.45 | 73.80 | 39.33 |
| 85.92 | 100.00 | 92.96 | 117.70 | 92.50 | 95.45 | 93.98 | 4.69 | 74.29 | 72.73 | 73.51 | 39.22 |
| 95.77 | 100.00 | 97.89 | 117.18 | 92.50 | 100.00 | 96.25 | 4.23 | 81.43 | 61.36 | 71.40 | 35.98 |
| 86.57 | 100.00 | 93.28 | 120.83 | 100.00 | 90.91 | 95.45 | 5.14 | 81.16 | 68.18 | 74.67 | 35.98 |
| 90.27 | 100.00 | 95.14 | 118.39 | 94.00 | 97.27 | 95.64 | 10.16 | 77.95 | 66.36 | 72.15 | 38.22 |

Cuadro 18: Cuadro resumen de resultados generales

| | **Ionosphere** | | | | **Parkinsons** | | | | **Spectf-heart** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** | ***%cl*** | ***%red*** | ***Agr.*** | **T** |
| **1-NN** | 86.26 | 0.00 | 43.13 | 0.002 | 96.00 | 0.00 | 48.00 | 0.002 | 65.90 | 0.00 | 32.95 | 0.002 |
| **RELIEF** | 86.54 | 2.94 | 44.74 | 0.01 | 96.00 | 0.00 | 48.00 | 0.002 | 73.35 | 21.82 | 47.58 | 0.01 |
| **BL** | 88.88 | 86.47 | 87.68 | 18.79 | 92.93 | 82.73 | 87.83 | 33.50 | 72.77 | 70.91 | 71.84 | 23.16 |
| **AGG1** | 85.50 | 69.41 | 77.46 | 119.55 | 89.86 | 81.82 | 85.84 | 31.34 | 73.64 | 60.45 | 67.05 | 126.69 |
| **AGE2** | 89.39 | 98.82 | 94.11 | 120.38 | 92.50 | 99.09 | 95.80 | 31.52 | 73.64 | 77.27 | 75.46 | 127.11 |
| **AM1** | 92.26 | 89.41 | 90.84 | 123.70 | 92.43 | 92.73 | 92.58 | 31.15 | 76.51 | 72.73 | 74.62 | 130.72 |
| **BMB** | 89.67 | 87.06 | 88.37 | 120.21 | 92.43 | 95.45 | 93.94 | 29.43 | 71.90 | 74.09 | 73.00 | 124.13 |
| **ILS** | 89.44 | 92.35 | 90.90 | 95.08 | 94.00 | 97.27 | 95.64 | 16.88 | 77.93 | 75.45 | 76.69 | 114.01 |
| **ES** | 87.11 | 97.65 | 92.38 | 118.53 | 95.50 | 100.00 | 97.75 | 30.78 | 72.48 | 61.82 | 67.15 | 15.10 |
| **ILS-ES** | 90.27 | 100.00 | 95.14 | 118.39 | 94.00 | 97.27 | 95.64 | 10.16 | 77.95 | 66.36 | 72.15 | 38.22 |

La hibridación de la búsqueda reiterada mediante enfriamiento simulado tambien parece una opción bastante competitiva con respecto al resto, en algunos casos también obtiene reducciones falsamente elevadas como ocurría en el enfriamiento simulado. En este caso es la opción ganadora absoluta para Ionosphere, obtiene de las mejores tasas de clasificación y la mejor de reducción. La introducción de las 2 formas de mutación tanto en la reiteración como la del propio enfriamiento simulado obtiene la mayor variabilidad de entre todos los algoritmos considerados en esta práctica y proporciona muy buenos resultados de forma sostenida.

En la tabla general se han incluido las mejores variantes de los algoritmos genéticos de la P2 y los algoritmos de la P1 así como los de esta práctica. Es reseñable que la labor de aprendizaje de pesos en Parkinsons parece la más fácil de los datasets ya que obtiene muy buenos resultados en todos los algoritmos con cierta complejidad. En general, resulta que los algoritmos de esta práctica mejoran ligeramente a los meméticos y genéticos basados en poblaciones, además estos últimos tienen mayor simplicidad de codificación (por estar basados en búsqueda local) y no necesitan de gran estructuración de datos. Los resultados que hemos obtenido globalmente indican que en cada Dataset el mejor algoritmo es distinto, casual o no esto pone de manifiesto la veracidad del teorema de No Free Lunch, no existe un algoritmo universal y debemos hacer un estudio en cada problema dependiente de su dominio y características. Una metaheurística es buena, no por si misma sino por su adaptabilidad a distintos datasets y superficies de optimización, debe encontrarse por tanto en cada caso el correcto compromiso entre exploración y explotación.