METAHEURÍSTICAS

Práctica 3 - APC

Grupo 1

Mario Muñoz Mesa
mario43514@correo.ugr.es

12 de junio de 2022

Índice

1. Descripción del problema.

Nos ocupamos del problema del Aprendizaje de Pesos en Características (APC). Plantea, en el contexto de un problema de clasificación, elegir o ajustar un vector de pesos $\mathbf{w} = (w_1, \ldots, w_d) \in [0,1]^d$ asociado a las características en base a un criterio de mejora del clasificador. Hemos denotado con d al número de características.

En nuestro caso trabajaremos con clasificador tipo 1-NN, y el criterio será maximizar:

$$F(\mathbf{w}) := \alpha \cdot \text{tasa\_clas}(\mathbf{w}) + (1-\alpha) \cdot \text{tasa\_red}(\mathbf{w})$$

donde

$$\text{tasa-clas} := 100\frac{\text{nº instancias bien clasificadas en training}}{\text{nº instancias en training}}, \qquad \text{tasa-red} := 100\frac{\text{nº valores } w_i < 0.2}{\text{nº características}}$$

y $alpha = 0.5$ que pondera la importancia entre el acierto y la reducción de características para el nuevo mejor clasificador que se pretende encontrar.

De lo que nos ocupamos por tanto es de obtener $\arg\max_{\mathbf{w}\in[0,1]^d} F(\mathbf{w})$ para un clasificador 1-NN que utilizará la distancia ponderada

$$d_{\mathbf{w}}(u,v) = \sqrt{\sum_{i=1}^{d} w_i(u_i - v_i)^2}, \quad u, v \in \mathbb{R}^d$$

para clasificar. Queremos que aumente el acierto y reduzca el número de características, cualquier característica con peso menor estricto a 0.2 se descarta.

2. Descripción de la aplicación de los algoritmos empleados.

2.1. Asociados al preprocesado y representación. P1.

Partimos de un conjunto de entrenamiento T formado por instancias de una muestra. Cada instancia se ha representado mediante una estructura de datos `SampleElement` que recoge las valores de las características en un vector `vector<double>` y la clase como una cadena de caracteres.

De esta forma, cada dataset se lee mediante la función `read_arff` y lo almacenamos como un vector de elementos muestrales `vector<SampleElement>`.

Para la normalización utilizamos min-max normalization, se ha implementado en `normalization`.

Las particiones para la validación cruzada $k-fold$ se realizan respetando la proporción de clases en cada partición, para ello cada elemento de una clase se va asignando a una partición de forma cíclica.

La solución será un vector de pesos con valores entre 0 y 1, cada componente indica el peso de una característica, se representa como un `vector<double>`

2.2. Asociados al preprocesado y representación. P3.

Para los algoritmos de Búsquedas basadas en Trayectorias se ha decidido crear una estructura, `Solution`, que contiene un `vector<double>` con el vector de pesos y su fitness asociado.

Además se han definido los comparadores mayor, $>$, y menor, $<$. Una solución es mayor a otra si su función objetivo es mayor, y menor si la función objetivo es menor.

2.3. Funciones generales. P1.

Nuestro clasificador utiliza la distancia euclídea, para manejarla se han implementado dos funciones: `euclidean_distance2` y `euclidean_distance2_w`.

Algorithm 1: euclidean_distance2

Input: vector de reales a, vector de reales b
Output: la distancia euclídea al cuadrado entre a y b
begin
  dist $\leftarrow 0$
  for $i = 0$ *to* $a.size() - 1$ **do**
    dist $\leftarrow$ dist $+ (a[i] - b[i])^2$
  end
  return dist
end

Algorithm 2: euclidean_distance2_w

Input: vector de reales a, vector de reales b, vector de pesos $weights$
Output: la distancia euclídea ponderada y al cuadrado entre a y b
begin
  dist $\leftarrow 0$
  for $i = 0$ ***to*** $a.size() - 1$ **do**
    `// se descartan las características con peso <2`
    if $weights[i] \geq 0.2$ **then**
      dist $\leftarrow$ dist $+ (a[i] - b[i])^2$
    end
  end
  return dist
end

La función `euclidean_distance2` se utiliza exclusivamente para el cálculo de amigo y enemigo más cercano en el algoritmo Relief. Con `euclidean_distance2_w` calculamos la distancia ponderada descartando pesos menores a 0.2, se utiliza en 1-NN y Búsqueda Local.

En ambos casos calculamos la distancia euclídea al cuadrado por motivos de eficiencia y porque lo que nos interesa es comparar distancias, y como la raíz cuadrada es una función creciente, si $a > b \Rightarrow \sqrt{a} > \sqrt{b}$, no hay inconveniente en trabajar con la distancia al cuadrado.

El algoritmo 1-NN, haciendo uso de distancia ponderada, se ha implementado en `one_NN`.

Algorithm 3: one_NN

Input: elemento muestral *sam_el* a clasificar, conjunto de elementos muestrales sobre el que buscar el más cercano *sam_elements*, vector de pesos *weights*
Output: la clase o etiqueta del elemento más cercano, *min_l*

```
begin
    min_dist ← ∞
    for e in sam_elements do
        d ← euclidean_distance2_w(e.features, sam_el.features, weights)
        if d < min_dist then
            min_l ← e.label
            min_dist ← d
        end
    end
    return min_l
end
```

La versión con leave-one-out, que se utiliza en Búsqueda Local, simplemente consiste en dejar un elemento fuera.

Algorithm 4: one_NN_lo

Input: elemento muestral *sam_el* a clasificar, conjunto de elementos muestrales sobre el que buscar el más cercano *sam_elements*, vector de pesos *weights*, posición de elemento a dejar fuera *leave_out*
Output: la clase o etiqueta del elemento más cercano, *min_l*

```
begin
    min_dist ← ∞
    for i = 0 to sam_elements.size() − 1 do
        if i ≠ leave_out then
            d ← euclidean_distance2_w(sam_elements[i].features, sam_el.features, weights)
            if d < min_dist then
                min_l ← sam_elements[i].label
                min_dist ← d
            end
        end
    end
    return min_l
end
```

2.4. Funciones generales. P3.

Para evaluar la función objetivo asociada a los pesos de una solución y asignarlo a la solución se ha creado la función `obj_to_solution`:

Algorithm 5: obj_to_solution

Input: solución s, vector de elementos muestrales $training$
begin
  for $i = 0$ ***to*** $training.size()$ **do**
    $class\_labels.push\_back(\ one\_NN\_lo(training[i], training, s.w, i)\ )$
  end
  $s.obj \leftarrow obj\_function(class\_rate(class\_labels, training),\ red\_rate(s.w))$
end

El operador de mutación coincide con el de generación de vecinos en la BL, introduce una perturbación con una normal de media y desviación típica dadas por el argumento n_dist. La media es 0 en todos los algoritmos y respecto a la desviación típica: en BL y ES es 0.3, y en ILS 0.4.

Algorithm 6: mutation

Input: vector de pesos $weights$, índice a mutar i, distribución normal n_dist
begin
  $weights[i] \leftarrow weights[i] + e \in n\_dist$
  if $weigths[i] > 1$ **then**
    $weights[i] \leftarrow 1$
  end
  else
    if $weights[i] < 0$ **then**
      $weights[i] \leftarrow 0$
    end
  end
end

La búsqueda local que es utilizada por BMB e ILS sigue el mismo esquema que en la práctica 1. Ahora sin embargo: se adapta a la estructura de solución definida, el número máximo de evaluaciones es 1000, y comenzará a partir de una solución dada como argumento, esta solución acabará conteniendo la solución resultante de la búsqueda local.

Algorithm 7: local_search

Input: conjunto de elementos muestrales *training*, solución *sol*
begin
$n \leftarrow num\_feats$
$max\_gen\_neighbours \leftarrow 20 * n$
$max\_objf\_evals \leftarrow 1000$
$new\_best\_obj \leftarrow false$
$mod\_pos \leftarrow 0$
$gen\_neighbours,\ obj\_eval\_count \leftarrow 0,\ 0$
`// se inicializan índices de las componentes`
for $i = 0$ ***to*** $n - 1$ **do**
$comp\_indexes.push\_back(i)$
end
$best\_obj \leftarrow sol.obj$

while $gen\_neighbours < max\_gen\_neighbours$ ***and*** $obj\_eval\_count < max\_objf\_evals$ **do**
`// aleatorizamos componentes a mutar si se han recorrido todas o se mejoró f.obj`
if new_best_obj ***or*** $mod\_pos\,\%n == 0$ **then**
$new\_best\_obj \leftarrow false$
$shuffle(comp\_indexes)$
$mod\_pos \leftarrow 0$
end
`// Se toma componente a mutar y se muta`
$comp\_to\_mut \leftarrow comp\_indexes[mod\_pos\,\%n]$
$muted\_sol \leftarrow sol$
$mutation(muted\_sol.w,\ comp\_to\_mut,\ \mathcal{N}(0,\ 0.3))$
$gen\_neighbours \leftarrow gen\_neighbours + 1$
`// se asigna función objetivo a solución con los pesos mutados`
$obj\_to\_solution(muted\_sol,\ training)$
`// Si se ha mejorado, actualizamos mejor objetivo y weights, y vecinos generados`
if $muted\_sol.obj > best\_obj$ **then**
$sol \leftarrow muted\_sol$
$best\_obj \leftarrow sol.obj$
$gen\_neighbours \leftarrow 0$
$new\_best\_obj \leftarrow true$
end
$obj\_eval\_count \leftarrow obj\_eval\_count + 1$
$mod\_pos \leftarrow mod\_pos + 1$
end
end

2.5. Función objetivo.

Para el cálculo de las tasas se han implementado las funciones `class_rate` y `red_rate` que simplemente implementan la propia de definición de tasa-class y tasa-red.

Algorithm 8: class_rate

Input: vector con clases o etiquetas de elementos clasificados $class\_labels$, elementos de test $test$

Output: el porcentaje de acierto

begin

$n \leftarrow 0$

$class\_labels\_s \leftarrow class\_labels.size()$

for $i = 0$ ***to*** $class\_labels\_s - 1$ **do**

if $class\_labels\_s[i] == test[i].label$ **then**

$n \leftarrow n + 1$

end

end

return $100\frac{n}{class\_labels\_n}$

end

Algorithm 9: red_rate

Input: vector de pesos $weights$

Output: el porcentaje de reducción

begin

$feats\_reduced \leftarrow 0$

for w ***in*** $weights$ **do**

if $w < 0.2$ **then**

$feats\_reduced \leftarrow feats\_reduced + 1$

end

end

return $100\frac{feats\_reduced}{initial\_number\_of\_features}$

end

La función objetivo, teniendo en cuenta que trabajamos con $\alpha = 0.5$, nos quedaría:

Algorithm 10: obj_function

Input: tasa de clasificación $class\_rate$, tasa de reducción red_rate

Output: el valor de la función objetivo o fitness

begin

return $\alpha \cdot class\_rate + (1 - \alpha) \cdot red\_rate$

end

3. Descripción de los algoritmos.

3.1. Enfriamiento simulado (ES).

Este algoritmo es un algoritmo de búsqueda por trayectorias simple. Persigue además no quedar atrapado en óptimos locales.

Para ello parte de una "temperatura" inicial que irá disminuyendo conforme aumentan las iteraciones. Esta temperatura irá determinando la probabilidad de aceptar soluciones peores que la actual, a menor temperatura menor probabilidad de aceptar. De este modo, cuando la temperatura es muy baja (cercana a la final) la probabilidad de aceptar una solución peor es muy baja, y cuando al temperatura es alta (al comienzo) se aceptarán bastantes soluciones peores, incentivando la exploración.

El operador de movimiento será la mutación normal con $\mu = 0$ y $\sigma = 0.3$. En nuestro caso una solución mutada, s_m, de la solución actual, s, será aceptada si se cumple una de las dos condiciones:

- $\Delta f < 0$ donde $\Delta f := fitness(s) - fitness(s_m)$, pues nuestro problema es de maximización.
- Se verifica $u < e^{-\frac{\Delta f}{T \cdot K}}$ con $u \in U(0,1)$, donde T es la temperatura actual y $K = 1$.

  Nota: en el caso de $\Delta f = 0$, se toma $\Delta f = 2.5 \cdot T_f$, donde T_f es la temperatura final, de modo que conforme disminuya T, acercándose a T_f, menor será la probabilidad de aceptar una solución con mismo fitness que la actual. Cuando $T = T_f$ la probabilidad sería 0.082 (8.2 %).

Adicionalmente, por cada mutación aceptada se comprueba si es la mejor solución hasta el momento, si lo es se guarda como la mejor.

Para el enfriamiento de la temperatura se emplea el esquema de Cauchy modificado:

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

donde M es el número de enfriamientos a realizar, T_0 la temperatura inicial y T_f la final. En nuestro caso:

$$M = \frac{15000}{\text{máx\_vecinos}} \; ; \quad T_0 = \frac{0.3 \cdot fitness(s_0)}{-\ln(0.3)} \; ; \quad T_f = 0.001$$

Donde máx_vecinos $= 10n$ donde n es el tamaño del vector de pesos.

Nota: como la temperatura inicial debe ser menor que la final se comprobará que sea así, en caso contrario se ha decidido reducir con $T_f = 0.1 \cdot T_f$ hasta que se cumpla.

Se enfriará la temperatura cada vez que se generen máx_vecinos o cada vez que se acepten máx_éxitos $:= 0.1 \cdot$ máx_vecinos vecinos.

La condición de parada es alcanzar 15000 evaluaciones de la función objetivo o bien no haber logrado ningún éxito en el enfriamiento.

El pseudocódigo es el siguiente:

Algorithm 11: sim_annealing

Input: vector de elementos muestrales *training*
Output: el vector de pesos de la mejor solución encontrada
begin
  $max\_neighbours \leftarrow 10 * n$
  $max\_success \leftarrow 0.1 * max\_neighbours$
  $max\_evals \leftarrow 15000$
  $evals,\ gen\_neighbours,\ num\_success \leftarrow 0,\ 0,\ 0$
  $max\_anns \leftarrow max\_evals/max\_neighbours$
  $T_f \leftarrow 0.001$
  `// Solución inicial`
  for $i = 0$ ***to*** n **do**
    $sol.w.push\_back(e \in U(0,1))$
  end
  $obj\_to\_solution(sol, training)$
  $evals \leftarrow 1$
  $best\_sol \leftarrow sol$
  $T_0 \leftarrow \frac{0.3 \cdot best\_sol.obj}{-\ln(0.3)}$
  $T \leftarrow T_0$
  while $T_f \geq T$ **do** `// Temperatura final debe ser menor que la inicial`
    $T_f \leftarrow T_f * 0.1$
  end
  $\beta \leftarrow \frac{T_0 - T_f}{max\_anns \cdot T_0 \cdot T_f}$
  $num\_success \leftarrow -1$
  while $evals < max\_evals$ ***and*** $num\_success \neq 0$ **do**
    $gen\_neighbours \leftarrow 0$
    $num\_success \leftarrow 0$
    while $evals < max\_evals$ ***and*** $gen\_neighbours < max\_neighbours$ ***and***
     $num\_success < max\_success$ **do**
      $muted\_sol \leftarrow sol$
      $mutation(muted\_sol.w, rand(0, n-1), \mathcal{N}(0, 0.3))$
      $obj\_to\_solution(muted\_sol, training)$
      $evals \leftarrow evals + 1$
      $gen\_neighbours \leftarrow gen\_neighbours + 1$
      $\Delta f \leftarrow sol.obj - muted\_sol.obj$
      `// Si tienen mismo fitness también se controla que la probabilidad de aceptar vaya disminuyendo con la temperatura`
      if $\Delta f == 0$ **then**
        $\Delta f \leftarrow 2.5 * T_f$
      end
      if $\Delta f < 0$ ***or*** $rand(0,1) \leq e^{-\frac{\Delta f}{T}}$ **then**
        $num\_success \leftarrow num\_success + 1$
        $sol \leftarrow muted\_sol$
        if $sol.obj > best\_sol.obj$ **then**
          $best\_sol \leftarrow sol$
        end
      end
    end
    $T \leftarrow \frac{T}{1 + \beta * T}$
  end
  return $best\_sol.w$
end

3.2. Búsqueda Multiarranque Básica (BMB).

El algoritmo BMB consiste en aplicar BL a varias soluciones generadas aleatoriamente, guardando las obtenidas tras al BL y devolviendo la mejor.

En nuestro caso se generarán 15 soluciones aleatorias, cada BL local aplicada realizará un máximo de 1000 evaluaciones de la función objetivo.

Algorithm 12: bmb

Input: vector de elementos muestrales *training*
Output: el vector de pesos de la mejor solucion encontrada
begin
  $max\_its \leftarrow 15$
  $best\_sol.obj \leftarrow -1$
  for $i = 0$ ***to*** $max\_its - 1$ **do**
    `// solución aleatoria`
    for $j = 0$ ***to*** $n - 1$ **do**
      $sol.w[j] \leftarrow u \in U(0, 1)$
    end
    $obj\_to\_solution(sol, training)$
    $local\_search(training, sol)$
    if $sol.obj > best\_sol.obj$ **then**
      $best\_sol \leftarrow sol$
    end
  end
  return $best\_sol.w$
end

3.3. Búsqueda Local Reiterada (ILS).

El algoritmo ILS consistirá en generar una solución inicial aleatoria y aplicar el algoritmo de BL sobre ella. Una vez obtenida la solución optimizada, se estudiará si es mejor que la mejor solución encontrada hasta el momento y se realizará una mutación sobre la mejor de estas dos, volviendo a aplicar el algoritmo de BL sobre esta solución mutada. Este proceso se repite un determinado número de veces, devolviéndose la mejor solución encontrada en todo el proceso.

La mutación a aplicar consite en mutar conjuntamente $t = 0.1 \cdot n$ características aleatorias con el operador de mutación que venimos usando pero esta vez con $\sigma = 0.4$.

En total se realizarán 15 llamadas a BL, y cada BL realizará un máximo de 1000 evaluaciones de la función objetivo.

Algorithm 13: ils

Input: vector de elementos muestrales *training*
Output: el vector de pesos de la mejor solucion encontrada
begin
  $max\_its \leftarrow 15$
  $t \leftarrow round(0.1 * n)$
  `// Solución inicial`
  for $i = 0$ ***to*** $n - 1$ **do**
    $sol.w.push\_back(e \in U(0, 1))$
  end
  $obj\_to\_solution(sol, training)$
  `// Índices a mutar`
  for $i = 0$ ***to*** $n - 1$ **do**
    $indexes.push\_back(i)$
  end
  $local\_search(training, sol)$
  for $i = 1$ ***to*** $max\_its - 1$ **do**
    $shuffle(indexes)$
    $muted\_sol \leftarrow sol$
    for $j = 0$ ***to*** $t - 1$ **do**
      $mutation(muted\_sol.w, indexes[j], \mathcal{N}(0, 0.4))$
    end
    $obj\_to\_solution(muted\_sol, training)$
    $local\_search(training, muted\_sol)$
    if $muted\_sol.obj > sol.obj$ **then**
      $sol \leftarrow muted\_sol$
    end
  end
  return $sol.w$
end

3.4. Algoritmo híbrido ILS-ES.

Este algoritmo sigue el mismo esquema que ILS a excepción de utilizar ES en lugar de BL.

Para este algoritmo se adapta el pseudocódigo de ES para partir ya de una solución dada y realizar como máximo 1000 evaluaciones de la función objetivo (para que en total realice 15000 evaluaciones de función objetivo como el resto de algoritmos y sea equiparable)

Pseudocódigo de ES adaptado:

Algorithm 14: sim_annealing_s

Input: vector de elementos muestrales *training*, solución de partida s
Output: la mejor solución encontrada
begin
  $max\_neighbours \leftarrow 10 * n$
  $max\_success \leftarrow 0.1 * max\_neighbours$
  $max\_evals \leftarrow 1000$
  $max\_anns \leftarrow max\_evals/max\_neighbours$
  $evals,\ gen\_neighbours,\ num\_success \leftarrow 0,\ 0,\ 0$
  $T_f \leftarrow 0.001$
  $sol \leftarrow s$
  $best\_sol \leftarrow sol$
  $T_0 \leftarrow \frac{0.3 \cdot best\_sol.obj}{-\ln(0.3)}$
  $T \leftarrow T_0$
  while $T_f \geq T$ **do** `// Temperatura final debe ser menor que la inicial`
    $T_f \leftarrow T_f * 0.1$
  end
  $\beta \leftarrow \frac{T_0 - T_f}{max\_anns \cdot T_0 \cdot T_f}$
  $num\_success \leftarrow -1$
  while $evals < max\_evals$ ***and*** $num\_success \neq 0$ **do**
    $gen\_neighbours \leftarrow 0$
    $num\_success \leftarrow 0$
    while $evals < max\_evals$ ***and*** $gen\_neighbours < max\_neighbours$ ***and*** $num\_success < max\_success$ **do**
      $muted\_sol \leftarrow sol$
      $mutation(muted\_sol.w, rand(0, n-1), \mathcal{N}(0, 0.3))$
      $obj\_to\_solution(muted\_sol, training)$
      $evals \leftarrow evals + 1$
      $gen\_neighbours \leftarrow gen\_neighbours + 1$
      $\Delta f \leftarrow sol.obj - muted\_sol.obj$
      `// Si tienen mismo fitness también se controla que la probabilidad de aceptar vaya disminuyendo con la temperatura`
      if $\Delta f == 0$ **then**
        $\Delta f \leftarrow 2.5 * T_f$
      end
      if $\Delta f < 0$ ***or*** $rand(0,1) \leq e^{-\frac{\Delta f}{T}}$ **then**
        $num\_success \leftarrow num\_success + 1$
        $sol \leftarrow muted\_sol$
        if $sol.obj > best\_sol.obj$ **then**
          $best\_sol \leftarrow sol$
        end
      end
    end
    $T \leftarrow \frac{T}{1 + \beta * T}$
  end
  return *best_sol*
end

Con esta modificación el pseudocódigo de ILS-ES es exactamente igual que ILS pero sustituyendo las llamadas a `local_search` por llamadas a `sim_annealing_s`.

4. Procedimiento considerado en el desarrollo.

Para el desarrollo de la práctica no se ha utilizado ningún framework, se han implementado las funciones necesarias en C++. Todo el código se encuentra en `p3.cpp` .

Para la ejecución basta indicar un argumento con el archivo a ejecutar, en este caso se utiliza la semilla por defecto 1234, que es la misma que se utilizó para los resultados de esta memoria.

También se incluye un fichero `p3_all_datasets.sh` que ejecuta, con la semilla por defecto, `p3` sobre los 3 datasets.

5. Experimentos y análisis de resultados.

Los 3 datasets con los que se ha trabajado son:

- **Ionosphere**: Conjunto de datos de radar que fueron recogidos por un sistema *Goose Bay*, Labrador. Consta de 352 instancias con 34 características y 2 clases.
- **Parkinsons**: Conjunto de datos orientado a distinguir entre la presencia y ausencia de la enfermedad de Parkinson. Consta de 195 ejemplos con 22 características y 2 clases.
- **Spectf-Heart**: Conjunto de datos de detección de enfermedades cardiacas a partir de imágenes médicas de tomografía computerizada del corazón de pacientes. Consta de 267 ejemplos con 44 características y 2 clases.

Los resultados obtenidos son en 1-NN, BL y Relief fueron:

1-NN

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
P1	87.3239	0	43.662	0.373	95	0	47.5	0.181	85.7143	0	42.8571	0.468
P2	64.2857	0	32.1429	0.325	25	0	12.5	0.073	72.8571	0	36.4286	0.407
P3	64.2857	0	32.1429	0.325	25.641	0	12.8205	0.072	72.8571	0	36.4286	0.397
P4	64.2857	0	32.1429	0.325	23.6842	0	11.8421	0.07	72.8571	0	36.4286	0.397
P5	64.2857	0	32.1429	0.381	23.6842	0	11.8421	0.071	73.5294	0	36.7647	0.388
Media	68.89334	0	34.44672	0.3458	38.60188	0	19.30094	0.0934	75.563	0	37.78152	0.4114

Relief

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
P1	88.7324	2.94118	45.8368	1.753	95	9.09091	52.0455	0.364	90	0	45	2.28
P2	84.2857	5.88235	45.084	1.757	97.5	4.54545	51.0227	0.366	87.1429	0	43.5714	2.222
P3	90	2.94118	46.4706	1.76	94.8718	4.54545	49.7086	0.371	80	0	40	2.201
P4	88.5714	2.94118	45.7563	1.761	92.1053	4.54545	48.3254	0.369	84.2857	0	42.1429	2.172
P5	87.1429	2.94118	45.042	1.766	100	0	50	0.374	77.9412	0	38.9706	2.237
Media	87.74648	3.529414	45.63794	1.7594	95.89542	4.545452	50.22044	0.3688	83.87396	0	41.93698	2.2224

Búsqueda Local (BL)

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
P1	78.8732	85.2941	82.0837	3311.36	87.5	86.3636	86.9318	390.064	85.7143	88.6364	87.1753	6719.91
P2	92.8571	85.2941	89.0756	4673.61	82.5	90.9091	86.7045	408.803	85.7143	86.3636	86.039	7458.06
P3	95.7143	82.3529	89.0336	1986.46	87.1795	86.3636	86.7716	339.172	88.5714	88.6364	88.6039	5512.85
P4	88.5714	88.2353	88.4034	3914.9	86.8421	50	68.4211	265.049	85.7143	77.2727	81.4935	3448.21
P5	90	85.2941	87.6471	2388.89	97.3684	86.3636	91.866	581.204	80.8824	88.6364	84.7594	3612.81
Media	89.2032	85.2941	87.24868	3255.044	88.278	79.99998	84.139	396.8584	85.31934	85.9091	85.61422	5350.368

Los resultados de ES, BMB, ILS e ILS-ES:

ES

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
P1	84.507	88.2353	86.3712	22259	87.5	86.3636	86.9318	4260.59	77.1429	86.3636	81.7532	27566.4
P2	84.2857	85.2941	84.7899	24425.6	87.5	86.3636	86.9318	4846.22	88.5714	84.0909	86.3312	24123.4
P3	85.7143	85.2941	85.5042	25713.9	84.6154	86.3636	85.4895	4339.23	90	86.3636	88.1818	28748.5
P4	85.7143	88.2353	86.9748	22850.6	86.8421	90.9091	88.8756	4912.97	82.8571	86.3636	84.6104	26001.8
P5	90	88.2353	89.1176	25145	89.4737	86.3636	87.9187	4296.52	85.2941	84.0909	84.6925	29975.9
Media	86.0443	87.0588	86.5515	24078.8	87.1862	87.2727	87.2295	4531.11	84.7731	85.4545	85.1138	27283.2

BMB

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
P1	88.7324	88.2353	88.4838	21180.7	87.5	86.3636	86.9318	4202.84	87.1429	84.0909	85.6169	24765.9
P2	84.2857	91.1765	87.7311	21030.1	100	81.8182	90.9091	4005.66	90	84.0909	87.0455	24605.9
P3	92.8571	85.2941	89.0756	21208.9	94.8718	90.9091	92.8904	4504.04	84.2857	81.8182	83.0519	24914.8
P4	87.1429	85.2941	86.2185	20306.1	89.4737	81.8182	85.6459	4169.52	88.5714	84.0909	86.3312	25037.6
P5	90	88.2353	89.1176	20431.2	92.1053	90.9091	91.5072	4418.87	83.8235	90.9091	87.3663	25600
Media	88.6036	87.6471	88.1253	20831.4	92.7901	86.3636	89.5769	4260.19	86.7647	85	85.8824	24984.8

ILS

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
P1	87.3239	91.1765	89.2502	20627.8	87.5	90.9091	89.2045	2451.35	78.5714	90.9091	84.7403	26641.3
P2	90	91.1765	90.5882	21020	95	90.9091	92.9545	2840.84	90	93.1818	91.5909	27344.6
P3	84.2857	88.2353	86.2605	19157.3	94.8718	90.9091	92.8904	2677.5	87.1429	93.1818	90.1623	28861.9
P4	85.7143	91.1765	88.4454	21414.7	89.4737	90.9091	90.1914	2462.07	85.7143	93.1818	89.4481	27934.9
P5	87.1429	91.1765	89.1597	18949.2	86.8421	90.9091	88.8756	2587.95	82.3529	88.6364	85.4947	27081.3
Media	86.8934	90.5882	88.7408	20233.8	90.7375	90.9091	90.8233	2603.94	84.7563	91.8182	88.2872	27572.8

ILS-ES

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
P1	84.507	94.1176	89.3123	21557.7	87.5	86.3636	86.9318	4612.44	78.5714	90.9091	84.7403	25224.6
P2	84.2857	91.1765	87.7311	24182.3	85	90.9091	87.9545	4563.85	88.5714	88.6364	88.6039	26319.7
P3	88.5714	91.1765	89.8739	23002.1	94.8718	90.9091	92.8904	4446.55	80	90.9091	85.4545	26299.9
P4	84.2857	91.1765	87.7311	22168.8	89.4737	90.9091	90.1914	4833.17	87.1429	90.9091	89.026	27123.8
P5	82.8571	94.1176	88.4874	22922.7	94.7368	90.9091	92.823	4799.14	85.2941	90.9091	88.1016	26896.5
Media	84.9014	92.3529	88.6272	22766.7	90.3165	90	90.1582	4651.03	83.916	90.4545	87.1853	26372.9

Se muestra la tabla resumen:

Nº part.	Ionosphere				Parkinsons				Spectf-Heart			
	%clas	%red	Agr.	T	%clas	%red	Agr.	T	%clas	%red	Agr.	T
1-NN	68.89334	0	34.44672	0.3458	38.60188	0	19.30094	0.0934	75.563	0	37.78152	0.4114
Relief	87.74648	3.529414	45.63794	1.7594	95.89542	4.545452	50.22044	0.3688	83.87396	0	41.93698	2.2224
BL	89.2032	85.2941	87.24868	3255.044	88.278	79.99998	84.139	396.8584	85.31934	85.9091	85.61422	5350.368
ES	86.0443	87.0588	86.5515	24078.8	87.1862	87.2727	87.2295	4531.11	84.7731	85.4545	85.1138	27283.2
BMB	88.6036	87.6471	88.1253	20831.4	92.7901	86.3636	89.5769	4260.19	86.7647	85	85.8824	24984.8
ILS	86.8934	90.5882	**88.7408**	20233.8	90.7375	90.9091	**90.8233**	2603.94	84.7563	91.8182	**88.2872**	27572.8
ILS-ES	84.9014	92.3529	88.6272	22766.7	90.3165	90	90.1582	4651.03	83.916	90.4545	87.1853	26372.9

En cuanto a las tasas, en primer lugar vemos gráficamente la tasa de clasificación:

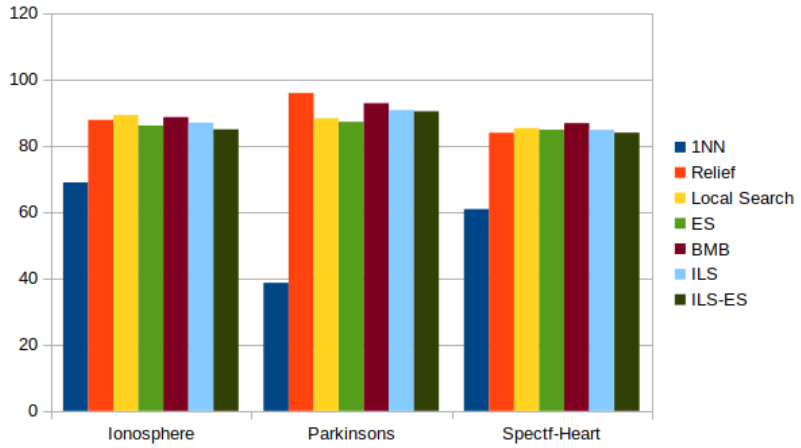

Figura 1: Tasa de clasificación.

Se observa que, a parte de 1NN, el resto de algoritmos proporcionan tasas de clasificación similares entre sí, dependiendo del dataset ganan unos u otros, no hay un patrón claro. Aunque Relief da buen resultado tampoco es de mucho interés pues sabemos que no proporciona apenas tasa de reducción. Lo que sí podemos destacar es que los algoritmos basados en trayectorias múltiples han dado resultados muy competitivos con BL, de hecho marcan nuevos máximos en Parkinsons y Spect-Heart. BMB destaca en todos los casos frente a ILS e ILS-ES, además ILS también da mejores tasas que ILS-ES. Por otra parte el algoritmo basado en trayectoria simple, ES, no ha superado a BL en ningún dataset aunque sí da valores cercanos, en Ionosphere y Spect-Heart supera a ILS-ES y queda cerca de ILS.

Vemos ahora las tasas de reducción:

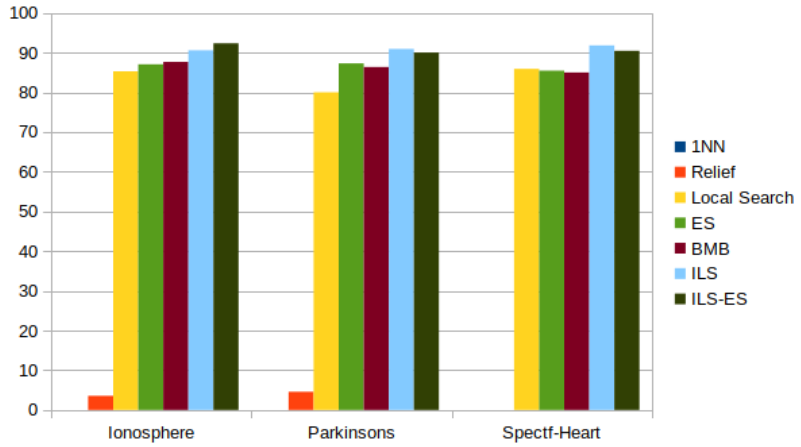

Figura 2: Tasa de reducción.

BL era el que mejor resultado proporcionaba en los algoritmos de referencia. Vemos que se han mejorado los resultados en los 3 datasets. Los máximos los marcan los algoritmos de trayectorias múltiples ILS e ILS-ES, aunque ILS con enfriamiento simulado solo a superado a ILS (con BL) en Ionosphere. También cabe destacar que ES, a pesar de ser de trayectoria simple, ha dado tasas de reducción equiparables a BMB, incluso mejores tanto en Parkinsons y Spect-Heart.

Mostramos también los tiempos de ejecución a los que se hará mención:

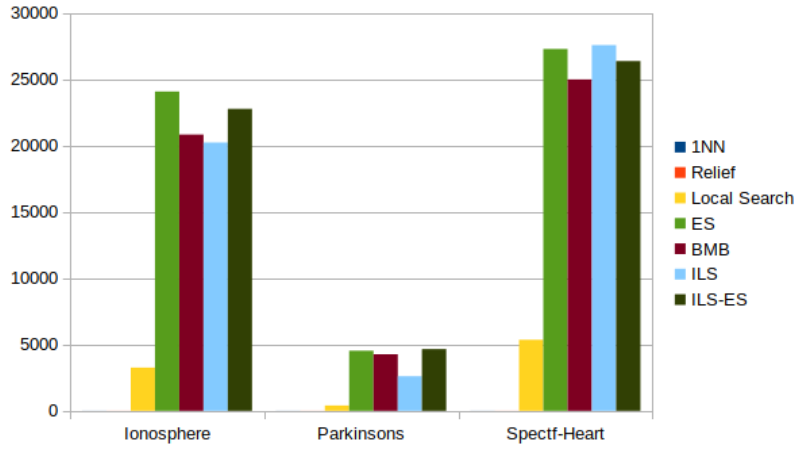

Figura 3: Tiempos de ejecución.

Por último el valor agregado:

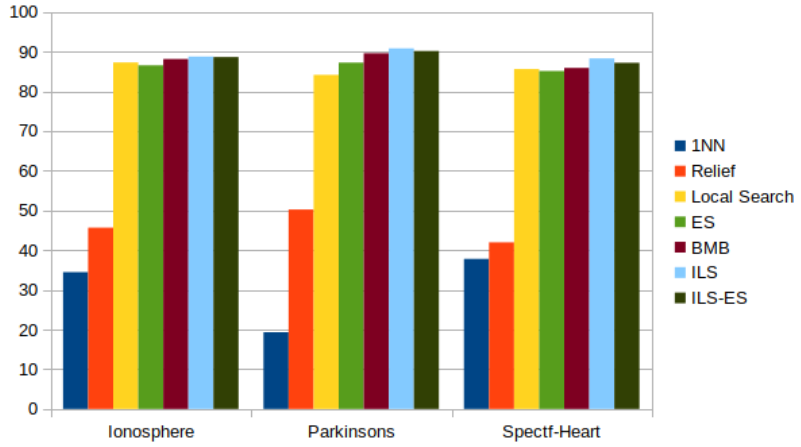

Figura 4: Valor de función objetivo o agregado.

De nuevo se han marcado nuevos máximos respecto a los resultados de BL en los 3 datasets. Los mejores resultados vienen dados por los algoritmos basados en trayectorias múltiples, siendo el mejor ILS en los 3 datasets. El segundo mejor algoritmo ha sido ILS-ES (en los 3 datasets). La búsqueda local iterativa ha resultado ser la estrategia que mayor fitness proporciona. En tercer lugar encontramos BMB (en los 3 datasets), otra técnica de trayectoria múltiple, que sigue siendo mejor que las estrategias de trayectorias simples. Entre los algoritmos de trayectorias simples encontramos resultados similares y competidos; en Ionosphere y Spect-Heart gana BL, y en Parkinsons gana ES, no hay un ganador consistente.

Se comparan las técnicas según tipología:

Trayectorias simples contra trayectorias múltiples

Como podemos ver, los algoritmos basados en trayectorias múltiples (BMB, ILS, ILS-ES) proporcionan un fitness mayor a los de trayectorias simples (BL y ES).

Lo anterior puede comprobarse con BL vs BMB, BL vs ILS, y ES vs ILS-ES. En todos los casos ganan las estrategias de trayectorias múltiples que introducen más diversidad de soluciones.

Lo que se observa por tanto es que da mejor resultado intentar, para un mismo número de evaluaciones de función objetivo, diversificar esas evaluaciones en varias BLs o ESs y explorar varios máximos locales, sin explotarlos demasiado, que invertir todas las evaluaciones en explotar un

máximo que aunque se alcance puede nos ser tan bueno.

BMB contra ILS

Vemos que ILS ha dado mejor tasa agregada que BMB en los 3 datasets.

Esto es, ha dado mejor resultado partir de una solución con buen fitness (ya aplicada una BL) e ir realizando mutaciones fuertes que partir de soluciones aleatorias independientes como en el caso de BMB.

También cabe mencionar que BMB requiere generar 15 soluciones aleatorias lo cual juega en contra de su tiempo de ejecución, sin embargo no ha supuesto gran diferencia con ILS pues en la implementación de ILS también se ha requerido elegir aleatoriamente las componentes a mutar, de hecho da mayor tiempo de ejecución en Spect-Heart.

BL contra ES

Podemos ver que se han obtenido tasas agregadas muy similares entre BL y ES, siendo ligeramente mejor BL en Ionosphere y Spectf-Heart. Cabe destacar que BL ha requerido además mucho menos tiempo de ejecución, esto tiene sentido pues, a diferencia de ES, no requiere generar soluciones aleatorias.

Además sabemos que ES acepta soluciones peores a la actual, aumentando así la capacidad de exploración. Es decir, dependiendo del dataset y los óptimos locales que se tengan puede ser más efectivo introducir más exploración con ES o buscar más explotación con BL; esto podría explicar que en Parkinsons se obtenga mejor resultado con ES, aunque siempre hay una componente de aleatoriedad.

Lo anterior también se aplica a ILS y ILS-ES, donde se han obtenido resultados similares aunque ligeramente mejores a favor de ILS.