UNIVERSIDAD DE GRANADA

Práctica 2.b:
Técnicas de Búsqueda basadas en poblaciones para el Problema del Aprendizaje de Pesos en Características (APC)

Brao Aissaoni, Isaac
Grupo de prácticas 3 Jueves(17:30-19:30)

Índice

1. Descripción del problema: Problema del Aprendizaje de Pesos en Características (APC)

El problema consiste en la realización de la clasificación de datos en clases que ya son conocidas. La pertenencia de cada dato a una clase ya se conoce a priori.

El objetivo es que el modelo de clasificación sea capaz de clasificar los datos correctamente con un error mínimo, además de aprender la importancia de cada característica que describe a los datos. Para ello debe asignar distinta importancia a cada característica a la que le es asignada un peso distinto, el cual se tendrá en cuenta al momento de clasificar.

2. Representación del problema y algoritmos empleados

2.1 Codificación de los datos del problema

2.1.1 Representación del problema

El problema se representa con 5 matrices $n_i$xm donde n_i es el número de datos de cada partición del problema y m es el número de característica.

$$X_i = \begin{pmatrix} x_{i11} & \cdots & x_{i1m} \\ \cdots & \cdots & \\ x_{in_i1} & \cdots & x_{in_im} \end{pmatrix}$$

Donde i corresponde al número de la partición.

2.1.2 Representación de la solución

Los pesos están representados en un vector de tamaño m donde m es el número de características

$$w_i = \begin{pmatrix} w_{i1} & \cdots & w_{im} \end{pmatrix}$$

Donde i corresponde al número de la partición.

2.2 Función objetivo

La función fitness tendrá en cuenta la tasa de acierto sobre el conjunto de entrenamiento y la tasa de reducción de los pesos. Buscamos maximizar el fitness.

$$fitness = \alpha \cdot \%aciertos + (1 - \alpha) \cdot \%reducción \quad donde\, \alpha = 0.75\, en\, los\, problemas$$

```python
def fitness(X_train, y_train, X_test, y_test, pesos, alfa=0.75):
    predicciones = clasificador1NN(X_train, y_train, X_test, pesos)
    aciertos = accuracy(y_test, predicciones)
    tasa_red = tasa_reduccion(pesos)
    return alfa*aciertos + (1-alfa)*tasa_red
```

```python
def tasa_reduccion(pesos):
    cont = 0
    for p in pesos:
        if p >= 0.1:
            cont += 1

    return 100*(cont / len(pesos))
```

```python
def accuracy(y_true, y_pred):
    correctos = 0

    for i in range(len(y_true)):
        if y_true[i] == y_pred[i]:
            correctos += 1

    return correctos / len(y_true)
```

```python
def clasificador1NN(X_train, y_test, X_test, pesos=None, k=1):
    # Inicializar pesos a 1 si no se especifican simulando un KNN normal
    if pesos is None:
        pesos = []
        for i in range(n_columnas(X_train)):
            pesos.append(1)

    predictions = np.array(n_columnas(X_test), dtype=str)

    # Calcular matriz de distancias
    for i in range(n_filas(X_test)):
        distancias = np.zeros(n_filas(X_train))
        for j in range(n_filas(X_train)):
            distancias[j] = distanciaEuclidiana(X_test[i], X_train[j], pesos)

        indices_ord = ordenarPorDistancia(distancias, eje='fila')

        # Si elegimos el numero 1, el vecino más cercano es el segundo porque el primero es el mismo
        indice_cercano = indices_ord[1]

        # Asignar la clase del vecino más cercano
        clase = y_test[indice_cercano]

        # Asignar la clase al conjunto de predicciones
        predictions[i] = clase

    return predictions
```

2.3 Generación de soluciones aleatorias

```python
def generar_poblacion_inicial(n_poblacion, n_caracteristicas):
    # Inicializar la poblacion con una matriz de n_poblacion x n_caracteristicas de pesos aleatorios de forma uniforme
    poblacion = np.empty((n_poblacion, n_caracteristicas))

    for i in range(n_poblacion):
        for j in range(n_caracteristicas):
            poblacion[i][j] = np.random.uniform(0, 1)
```

2.4 Operadores selección

2.4.1 AGG

```python
def seleccionar_padres_AGG(poblacion, fitness_poblacion):
    # Seleccionar padres por torneo
    padres = []

    for i in range(n_filas(poblacion)):
        # Seleccionar tres individuos aleatorios de la poblacion distintos
        indices_aleatorios = np.random.choice(range(0, n_filas(poblacion)), size=3)

        # Seleccionar el mejor de los tres
        mejor = np.argmax(fitness_poblacion[indices_aleatorios])

        # Añadir el mejor a la lista de padres
        padres.append(poblacion[indices_aleatorios[mejor]])

    return padres
```

2.4.1 AGE

```python
def seleccionar_padres_AGE(poblacion, fitness_poblacion):
    # Seleccionar padres por torneo
    padres = []

    # Solo se seleccionan dos padres
    for i in range(2):
        # Seleccionar tres individuos aleatorios de la poblacion distintos
        indices_aleatorios = np.random.choice(range(0, n_filas(poblacion)), size=3)

        # Seleccionar el mejor de los tres
        mejor = np.argmax(fitness_poblacion[indices_aleatorios])

        # Añadir el mejor a la lista de padres
        padres.append(poblacion[indices_aleatorios[mejor]])

    return padres
```

2.5 Operadores cruce

2.5.1 Operador de coeficientes aritméticos (CA)

```python
def cruce_CA(padres, prob_cruce):
    # Calcular cruces esperados
    cruces_esperados = int(prob_cruce * n_filas(padres) / 2)
    # Generar alphas aleatorios
    alphas = np.random.uniform(0, 1, size=cruces_esperados*2)

    hijos = []

    for i in range(cruces_esperados):
        hijo1 = padres[i*2] * alphas[i] + padres[i*2+1] * (1 - alphas[i])
        hijo2 = padres[i*2+1] * alphas[i+1] + padres[i*2] * (1 - alphas[i+1])

        hijos.append(hijo1)
        hijos.append(hijo2)

    return hijos
```

2.5.2 Operador Blend crossover (BLX)

```python
def cruce_BLX(padres, prob_cruce):
    # Calcular cruces esperados
    cruces_esperados = int(prob_cruce * n_filas(padres) / 2)
    # Alphas para el BLX-0.3
    ALPHA = 0.3

    hijos = []

    for i in range(cruces_esperados):
        hijo1 = np.empty(shape=n_columnas(padres))
        hijo2 = np.empty(shape=n_columnas(padres))

        for j in range(n_columnas(padres)):
            # Calcular minimo y maximo
            minimo = min(padres[i*2][j], padres[i*2+1][j])
            maximo = max(padres[i*2][j], padres[i*2+1][j])

            # Calcular rango
            I = maximo - minimo

            # Formulas de BLX-0.3
            # aleatorio( minimo - ALPHA*I, maximo + ALPHA*I)

            # Calcular valores de los hijos
            hijo1[j] = np.random.uniform(minimo - ALPHA*I, maximo + ALPHA*I)
            hijo2[j] = np.random.uniform(minimo - ALPHA*I, maximo + ALPHA*I)

        # Añadir hijos a la lista
        hijos.append(hijo1)
        hijos.append(hijo2)

    return hijos
```

2.6 Operadores mutación

2.6.1 AGG

```python
def mutar_AGG(hijos, prob_mutacion):
    mutaciones_esperadas = int(prob_mutacion * n_filas(hijos))

    for i in range(mutaciones_esperadas):
        # Seleccionar un hijo aleatorio
        hijo = hijos[np.random.randint(0, n_filas(hijos))]

        # Seleccionar un gen aleatorio
        gen = np.random.randint(0, n_columnas(hijo))

        # Mutar gen
        hijo[gen] = np.random.uniform(0, 1)
```

2.6.2 AGE

```python
def mutar_AGE(hijos, prob_mutacion):
    for i in range(n_filas(hijos)):
        for j in range(n_columnas(hijos[i])):
            if np.random.uniform(0, 1) < prob_mutacion:
                hijos[i][j] += np.random.normal(0, np.sqrt(0.3))
                hijos[i][j] = max(hijos[i][j], 0)
                hijos[i][j] = min(hijos[i][j], 1)
```

3. Algoritmo genético generacional (AGG)

```python
def fit_AGG(X, y, max_evaluaciones, tipo_cruce, semilla=7, n_poblacion = 50, prob_cruce = 0.7, prob_mutacion = 0.08):
    # Semilla para inicializar pesos aleatorios
    np.random.seed(semilla)
    evaluaciones = 0

    # Inicializar la poblacion con pesos aleatorios de tamaño poblacion x columnas de X
    poblacion = generar_poblacion_inicial(n_poblacion, n_columnas(X))

    # Initializar fitness de la poblacion
    fitness_poblacion = np.empty_like(poblacion)

    for i in range(n_filas(poblacion)):
        fitness_poblacion[i] = fitness(X, y, poblacion[i])
        evaluaciones += 1

    # Obtener el mejor individuo
    fitness_mejor_individuo = np.max(fitness_poblacion)
    mejor_individuo = poblacion[np.argmax(fitness_poblacion)]

    # Mientras no se alcance el maximo de evaluaciones
    while evaluaciones < max_evaluaciones:
        # Seleccionar padres
        padres = seleccionar_padres_AGG(poblacion, fitness_poblacion)

        # Cruzar padres
        hijos = cruce(padres, prob_cruce, tipo_cruce)

        # Mutar hijos
        mutar_AGG(hijos, prob_mutacion)

        # Evaluar hijos
        for i in range(n_filas(hijos)):
            fitness_poblacion[i] = fitness(X, y, hijos[i])
            evaluaciones += 1

        #Reemplazar los peores individuos

        # Mejor de la nueva generacion
        mejor_hijo = np.argmax(fitness_poblacion)
        fitness_mejor_hijo = np.max(fitness_poblacion)

        # Si el mejor hijo es mejor que el mejor individuo, lo actualizamos
        if fitness_mejor_hijo > fitness_mejor_individuo:
            mejor_individuo = hijos[mejor_hijo]
            fitness_mejor_individuo = fitness_mejor_hijo
        else:
            # Si no, reemplazamos el peor hijo por el mejor individuo
            peor_hijo = np.argmin(fitness_poblacion)
            hijos[peor_hijo] = mejor_individuo
            fitness_poblacion[peor_hijo] = fitness_mejor_individuo

        # Actualizamos la poblacion
        poblacion = hijos

    # Devolvemos el mejor individuo
    return mejor_individuo
```

3.1 Versión mejorada del algoritmo

- Permito repeticiones de instancias en el torneo de la selección para mayor diversidad.
- La probabilidad de mutación es de 2.4 en vez de 0.08. He puesto ese valor porque es un valor óptimo para añadir diversidad.

4. Algoritmo genético estacionario (AGE)

```python
def fit_AGE(X, y, max_evaluaciones, tipo_cruce, semilla=7, n_poblacion = 50, prob_cruce = 1, prob_mutacion = 0.08):
    # Semilla para inicializar pesos aleatorios
    np.random.seed(semilla)
    evaluaciones = 0

    # Inicializar la poblacion con pesos aleatorios de tamaño poblacion x columnas de X
    poblacion = generar_poblacion_inicial(n_poblacion, n_columnas(X))

    # Initializar fitness de la poblacion
    fitness_poblacion = np.empty_like(poblacion)

    for i in range(n_filas(poblacion)):
        fitness_poblacion[i] = fitness(X, y, poblacion[i])
        evaluaciones += 1

    # Mientras no se alcance el maximo de evaluaciones
    while evaluaciones < max_evaluaciones:
        # Seleccionar 2 padres
        padres = seleccionar_padres_AGE(poblacion, fitness_poblacion)

        # Cruzar padres
        hijos = cruce(padres, prob_cruce, tipo_cruce)

        # Mutar hijos
        mutar_AGE(hijos, prob_mutacion)

        # Evaluar hijos en este caso son solo 2
        fitness_hijos = np.empty_like(hijos)
        for i in range(n_filas(hijos)):
            fitness_hijos[i] = fitness(X, y, hijos[i])
            evaluaciones += 1

        # Reemplazar los peores individuos
        # Los hijos compiten por entrar en la poblacion

        # Obtener los dos peores individuos
        peores = np.argsort(fitness_poblacion, order='desc')[:2]

        competidores = np.concatenate((hijos, poblacion[peores]))
        fitness_competidores = np.concatenate((fitness_hijos, fitness_poblacion[peores]))

        # Obtener los dos mejores individuos
        mejores = np.argsort(fitness_competidores, order='desc')[:2]

        # Reemplazar los peores por los mejores
        poblacion[peores] = competidores[mejores]
        fitness_poblacion[peores] = fitness_competidores[mejores]

    # Devolvemos el mejor individuo
    mejor_individuo = poblacion[np.argmax(fitness_poblacion)]
    return mejor_individuo
```

3.1 Versión mejorada del algoritmo

- Escoge dos individuos de manera aleatoria, que no son iguales
- La probabilidad de mutación es de 0.42 en vez de 0.08. He puesto ese valor porque es un valor óptimo para añadir diversidad.

5. Algoritmo memético (AM)

He implementado el algoritmo memético usando el cruce BLX, que, en general, me ha dado mejores resultados que AC.

```python
def fit_AM(X, y, max_evaluaciones, tipo_seleccion_bl, semilla=7, n_poblacion = 50, prob_cruce = 0.7, prob_mutacion = 0.08):
    # Semilla para inicializar pesos aleatorios
    np.random.seed(semilla)
    evaluaciones = 0
    generaciones = 0

    max_iter_bl = 2*n_columnas(X)

    # Inicializar la poblacion con pesos aleatorios de tamaño poblacion x columnas de X
    poblacion = generar_poblacion_inicial(n_poblacion, n_columnas(X))

    # Initializar fitness de la poblacion
    fitness_poblacion = np.empty_like(poblacion)

    for i in range(n_filas(poblacion)):
        fitness_poblacion[i] = fitness(X, y, poblacion[i])
        evaluaciones += 1

    generaciones += 1

    # Obtener el mejor individuo
    fitness_mejor_individuo = np.max(fitness_poblacion)
    mejor_individuo = poblacion[np.argmax(fitness_poblacion)]

    # Mientras no se alcance el maximo de evaluaciones
    while evaluaciones < max_evaluaciones:
        # Seleccionar padres
        padres = seleccionar_padres_AGG(poblacion, fitness_poblacion)

        # Cruzar padres
        hijos = cruce(padres, prob_cruce, tipo_cruce='BLX')

        # Mutar hijos
        mutar_AGG(hijos, prob_mutacion)

        # Evaluar hijos
        for i in range(n_filas(hijos)):
            fitness_poblacion[i] = fitness(X, y, hijos[i])
            evaluaciones += 1

        # Hacemos una seleccion de individuos de bl cada 10 generaciones
        if generaciones % 10 == 0:
            # Seleccionar individuos de la poblacion
            seleccionados = seleccion_BL(poblacion, fitness_poblacion, tipo_seleccion_bl)

            # Si hay sufitientes evaluaciones, aplicamos busqueda local
            if max_evaluaciones - evaluaciones < max_iter_bl * n_filas(seleccionados):
                for i in range(n_filas(seleccionados)):
                    seleccionados[i], fitness_poblacion[i], n_eval_bl = BL(X, y, seleccionados[i], fitness_poblacion[i], max_iter_bl)
                    evaluaciones += n_eval_bl


        #Reemplazar los peores individuos

        # Mejor de la nueva generacion
        mejor_hijo = np.argmax(fitness_poblacion)
        fitness_mejor_hijo = np.max(fitness_poblacion)

        # Si el mejor hijo es mejor que el mejor individuo, lo actualizamos
        if fitness_mejor_hijo > fitness_mejor_individuo:
            mejor_individuo = hijos[mejor_hijo]
            fitness_mejor_individuo = fitness_mejor_hijo
        else:
            # Si no, reemplazamos el peor hijo por el mejor individuo
            peor_hijo = np.argmin(fitness_poblacion)
            hijos[peor_hijo] = mejor_individuo
            fitness_poblacion[peor_hijo] = fitness_mejor_individuo

        # Actualizamos la poblacion
        poblacion = hijos
        generaciones += 1

    # Devolvemos el mejor individuo
    return mejor_individuo
```

5.1 Algoritmo de búsqueda local

```python
def BL(X, y, individuo, fitness, max_eval):
    # Inicializar el numero de evaluaciones
    n_evaluaciones = 0
    fitness_individuo = fitness

    while n_evaluaciones < max_eval:
        # Obtener un orden de mutacion aleatorio
        orden_mutacion = np.random.permutation(len(individuo))

        for mut in orden_mutacion:
            # Obtener un vecino
            vecino = obtenerVecino(individuo, mut)

            # Calcular fitness del vecino
            fitness_vecino = fitness(X, y, vecino)
            n_evaluaciones += 1

            # Si el vecino es mejor, actualizamos pesos
            if fitness_vecino > fitness_individuo:
                individuo = vecino
                fitness_individuo = fitness_vecino

            # Si se alcanza el maximo de evaluaciones, salimos
            if n_evaluaciones >= max_eval:
                break

    return individuo, fitness_individuo, n_evaluaciones
```

5.2 Selección de individuos para la búsqueda local

```python
def seleccion_BL(poblacion, fitness_poblacion, tipo_seleccion_bl):
    match tipo_seleccion_bl:
        case 'Mejores':
            # Seleccionar el 10% de los mejores individuos
            p = 0.1
            return np.argsort(fitness_poblacion, order='asc')[0:int(p*n_filas(poblacion))]
        case 'Aleatorios':
            # Seleccionar el 10% de los individuos aleatorios
            p = 0.1
            return np.random.choice(poblacion, size=int(p*n_filas(poblacion)))
        case 'Todos':
            # Seleccionar todos los individuos
            return range(n_filas(poblacion))
        case _:
            raise ValueError("Tipo de seleccion no válido.")
```

6. Procedimiento de desarrollo de la práctica

6.1 Software utilizado para el desarrollo del código fuente

La práctica se ha realizado en el lenguaje de programación python. Se ha usado **numpy** para las operaciones vectoriales y matriciales; **scipy** para el cálculo de distancias euclidianas, con peso y sin pesos para el KNN; **scikit learn** para la normalización y escalado de los datos; **pandas** para las tablas, estadísticas y guardar la salida en formato csv; el paquete **liac-arff** para la lectura de los datos y el paquete **tqdm** para las barras de progreso.

6.2 Manual de usuario

1. **Inicio del Programa:**
   Para ejecutar el script usar python3 main.py

   Al ejecutar el script, se te solicitará introducir la base de datos que deseas utilizar.
   Las opciones disponibles son:

   - **BreastCancer** (Por defecto): Base de datos sobre cáncer de mama.
   - **Ecoli**: Base de datos sobre secuencias de ADN de bacterias E. coli.
   - **Parkinson**: Base de datos sobre la detección de la enfermedad de Parkinson.

2. **Selección del Modelo:**
   Después de seleccionar la base de datos, se te pedirá que elijas el modelo de aprendizaje automático que deseas utilizar. Las opciones son:

   - **KNN** (Por defecto): K-Nearest Neighbors.
   - **Relief**: Algoritmo de selección de características Relief.
   - **BL**: Algoritmo Búsqueda Local.
   - **AGG**: Algoritmo Genético Generacional.
   - **AGE**: Algoritmo Genético Estacionario.
   - **AM**: Algoritmo Memético.
   - **ALL**: Ejecutar todos los modelos disponibles.

3. **Configuración Adicional:**
   Dependiendo del modelo seleccionado, puede ser necesario introducir configuraciones adicionales:
   - **Valor de k**: Relevante para el modelo KNN y la opción ALL. Representa el número de vecinos más cercanos a considerar.
   - **Valor de la semilla**: Requerido para los modelos BL, AGG, AGE, AM y la opción ALL. La semilla se utiliza para inicializar la generación de números aleatorios.
   - **Versión Mejorada**: Para los modelos AGG, AGE, AM y la opción ALL, se te preguntará si deseas utilizar la versión mejorada de los algoritmos genéticos.

- **Operador de Cruce**: Para AGG y AGE, se te solicitará seleccionar un operador de cruce entre 'CA' (Cruce Aritmético) y 'BLX' (Blend Crossover).
- **Operador de Selección en BL para el algoritmo memético**: Para el modelo AM, deberás seleccionar un operador de selección entre 'All', 'Random' y 'Best'.

4. **Ejecución del Modelo:**
   Una vez que se han introducido todas las configuraciones, el modelo seleccionado se ejecutará con las opciones especificadas.

5. **Guardar Resultados en Archivo CSV:**
   Puedes seleccionar `S` para guardar los resultados en un archivo CSV o `N` para no guardarlos.

6. **Requisitos del Programa:**
   Este programa requiere los siguientes paquetes de Python instalados en tu sistema:

   - **numpy**
   - **scipy**
   - **pandas**
   - **liac-arff**
   - **scikit-learn**
   - **tqdm**

7. **Errores y Salida del Programa:**
   Si introduces una base de datos o modelo no válido, el programa mostrará un mensaje de error correspondiente y se cerrará.

6.3 Consideraciones a tener en cuenta

1. Si tiene instalado el paquete arff de google tendrá que desinstalarlo e instalar a continuación liac-arff porque generan incompatibilidad entre ellos.
2. Los valores utilizados para el análisis de los resultados han sido k 1 y semilla 7.

7. Experimentos y resultados

Los experimentos han sido realizados por el parámetro k = 1 (KNN) y con la semilla igual a 7 (BL, Genéticos y Meméticos)

7.1 Ecoli

7.1.1 Resultados usando la versión normal

1NN						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	81.20	0.00	60.90	60.00	80.00	1.65E-03
2	83.46	0.00	62.59	54.64	72.86	7.62E-04
3	80.97	0.00	60.73	61.76	82.35	7.64E-04
4	77.99	0.00	58.49	62.87	83.82	7.53E-04
5	80.80	0.00	60.60	63.75	85.00	6.96E-04
Media	80.88	0.00	60.66	60.61	80.81	9.26E-04
Std dev	1.95	0.00	1.46	3.61	4.82	4.07E-04

Relief						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	79.32	28.57	66.64	65.00	77.14	7.61E-03
2	82.33	28.57	68.89	62.86	74.29	7.69E-03
3	79.48	28.57	66.75	66.70	79.41	7.54E-03
4	78.36	28.57	65.91	67.80	80.88	7.32E-03
5	79.35	28.57	66.65	72.14	86.67	7.66E-03
Media	79.77	28.57	66.97	66.90	79.68	7.56E-03
Std dev	1.50	0.00	1.13	3.47	4.63	1.45E-04

BL						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	80.08	57.14	74.34	73.21	78.57	0.09
2	83.08	42.86	73.03	67.50	75.71	0.07
3	78.73	57.14	73.33	72.74	77.94	0.07
4	77.61	57.14	72.49	76.05	82.35	0.06
5	78.26	57.14	72.98	80.54	88.33	0.07
Media	79.55	54.29	73.24	74.01	80.58	0.07
Std dev	2.17	6.39	0.69	4.78	4.95	0.01

AGG-AC						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	80.83	57.14	74.91	73.21	78.57	3.35
2	80.83	57.14	74.91	63.57	65.71	3.51
3	80.60	57.14	74.73	74.95	80.88	3.65
4	73.88	71.43	73.27	70.80	70.59	3.33
5	73.91	71.43	73.29	71.61	71.67	3.42
Media	78.01	62.86	74.22	70.83	73.48	3.45
Std dev	3.76	7.82	0.86	4.36	6.18	0.13

AGG-BLX						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	79.32	57.14	73.78	71.07	75.71	3.72
2	84.21	42.86	73.87	66.43	74.29	4.08
3	80.97	57.14	75.01	72.74	77.94	3.70
4	77.61	57.14	72.49	74.95	80.88	3.73
5	80.07	42.86	70.77	75.71	86.67	4.26
Media	80.44	51.43	73.19	72.18	79.10	3.90
Std dev	2.44	7.82	1.62	3.70	4.91	0.26

AGE-AC						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	80.45	42.86	71.05	69.64	78.57	4.67
2	82.71	42.86	72.74	65.36	72.86	4.66
3	74.25	57.14	69.98	72.74	77.94	4.41
4	75.75	42.86	67.52	72.48	82.35	4.83
5	68.48	71.43	69.22	71.61	71.67	4.34
Media	76.33	51.43	70.10	70.37	76.68	4.58
Std dev	5.57	12.78	1.96	3.05	4.39	0.20

AGE-BLX						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	76.32	57.14	71.52	72.14	77.14	4.74
2	80.08	57.14	74.34	67.86	71.43	4.69
3	73.51	71.43	72.99	68.59	67.65	4.42
4	77.61	42.86	68.92	70.27	79.41	5.11
5	79.71	57.14	74.07	78.04	85.00	4.87
Media	77.44	57.14	72.37	71.38	76.13	4.77
Std dev	2.69	10.10	2.22	4.07	6.79	0.25

AM-All						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	79.70	57.14	74.06	71.07	75.71	3.67
2	80.83	57.14	74.91	63.57	65.71	3.68
3	80.97	57.14	75.01	72.74	77.94	3.75
4	77.99	57.14	72.77	74.95	80.88	3.78
5	79.35	57.14	73.80	78.04	85.00	3.87
Media	79.77	57.14	74.11	72.07	77.05	3.75
Std dev	1.22	0.00	0.91	5.42	7.23	0.08

AM-Random						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	79.70	57.14	74.06	71.07	75.71	3.64
2	80.83	57.14	74.91	63.57	65.71	3.61
3	80.97	57.14	75.01	72.74	77.94	3.62
4	77.99	57.14	72.77	74.95	80.88	3.72
5	73.91	71.43	73.29	71.61	71.67	3.49
Media	78.68	60.00	74.01	70.79	74.38	3.62
Std dev	2.92	6.39	0.98	4.30	5.90	0.08

AM-Best						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	79.70	57.14	74.06	71.07	75.71	3.55
2	80.83	57.14	74.91	64.64	67.14	3.55
3	80.97	57.14	75.01	72.74	77.94	3.58
4	73.88	71.43	73.27	70.80	70.59	3.18
5	79.35	57.14	73.80	78.04	85.00	3.62
Media	78.94	60.00	74.21	71.46	75.28	3.50
Std dev	2.92	6.39	0.74	4.79	6.89	0.18

7.1.2 Resultados extra usando la versión mejorada

AGG-AC						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	81.58	57.14	75.47	73.21	78.57	3.36
2	80.83	57.14	74.91	63.57	65.71	3.39
3	80.97	57.14	75.01	72.74	77.94	3.41
4	78.36	57.14	73.05	74.95	80.88	3.42
5	79.35	57.14	73.80	78.04	85.00	3.59
Media	80.22	57.14	74.45	72.50	77.62	3.43
Std dev	1.32	0.00	0.99	5.41	7.21	0.09

AGG-BLX						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	81.58	57.14	75.47	72.14	77.14	3.49
2	80.83	57.14	74.91	65.71	68.57	3.46
3	80.97	57.14	75.01	72.74	77.94	3.53
4	78.36	57.14	73.05	74.95	80.88	3.54
5	79.35	57.14	73.80	78.04	85.00	3.68
Media	80.22	57.14	74.45	72.72	77.91	3.54
Std dev	1.32	0.00	0.99	4.55	6.06	0.08

AGE-AC						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	78.20	57.14	72.93	72.14	77.14	4.12
2	78.57	57.14	73.21	70.00	74.29	4.13
3	80.97	57.14	75.01	72.74	77.94	4.08
4	70.52	71.43	70.75	65.28	63.24	3.92
5	76.81	57.14	71.89	79.29	86.67	4.30
Media	77.01	60.00	72.76	71.89	75.85	4.11
Std dev	3.93	6.39	1.59	5.07	8.44	0.14

AGE-BLX						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	79.70	57.14	74.06	71.07	75.71	4.27
2	80.83	57.14	74.91	64.64	67.14	4.23
3	80.97	57.14	75.01	72.74	77.94	4.28
4	78.36	57.14	73.05	74.95	80.88	4.37
5	79.35	57.14	73.80	78.04	85.00	4.78
Media	79.84	57.14	74.17	72.29	77.34	4.39
Std dev	1.09	0.00	0.81	5.01	6.67	0.23

AM-All						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	81.20	57.14	75.19	73.21	78.57	3.59
2	80.83	57.14	74.91	63.57	65.71	3.68
3	80.97	57.14	75.01	72.74	77.94	3.78
4	78.36	57.14	73.05	74.95	80.88	3.70
5	79.35	57.14	73.80	78.04	85.00	3.71
Media	80.14	57.14	74.39	72.50	77.62	3.69
Std dev	1.23	0.00	0.93	5.41	7.21	0.07

AM-Random						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	81.58	57.14	75.47	73.21	78.57	3.55
2	80.83	57.14	74.91	67.86	71.43	3.50
3	80.97	57.14	75.01	72.74	77.94	3.53
4	78.36	57.14	73.05	74.95	80.88	3.53
5	79.35	57.14	73.80	78.04	85.00	3.69
Media	80.22	57.14	74.45	73.36	78.76	3.56
Std dev	1.32	0.00	0.99	3.71	4.95	0.07

AM-Best						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	81.95	57.14	75.75	72.14	77.14	3.50
2	80.83	57.14	74.91	64.64	67.14	3.49
3	80.97	57.14	75.01	72.74	77.94	3.56
4	78.36	57.14	73.05	74.95	80.88	3.52
5	79.35	57.14	73.80	78.04	85.00	3.65
Media	80.29	57.14	74.50	72.50	77.62	3.54
Std dev	1.43	0.00	1.07	4.96	6.62	0.07

7.1.3 Resultados globales

	Ecoli					
Algoritmo	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1NN	80.88	0.00	60.66	60.61	80.81	9.26E-04
Relief	79.77	28.57	66.97	66.90	79.68	7.56E-03
BL	79.55	54.29	73.24	74.01	80.58	0.07
AGG-AC	78.01	62.86	74.22	70.83	73.48	3.45
AGG-AC Mejor	80.22	57.14	74.45	72.50	77.62	3.43
AGG-BLX	80.44	51.43	73.19	72.18	79.10	3.90
AGG-BLX Mejor	80.22	57.14	74.45	72.72	77.91	3.54
AGE-AC	76.33	51.43	70.10	70.37	76.68	4.58
AGE-AC Mejor	77.01	60.00	72.76	71.89	75.85	4.11
AGE-BLX	77.44	57.14	72.37	71.38	76.13	4.77
AGE-BLX Mejor	79.84	57.14	74.17	72.29	77.34	4.39
AM-All	79.77	57.14	74.11	72.07	77.05	3.75
AM-All Mejor	80.14	57.14	74.39	72.50	77.62	3.69
AM-Rand	78.68	60.00	74.01	70.79	74.38	3.62
AM-Rand Mejor	80.22	57.14	74.45	73.36	78.76	3.56
AM-Best	78.94	60.00	74.21	71.46	75.28	3.50
AM-Best Mejor	80.29	57.14	74.50	72.50	77.62	3.54

7.2 Parkinson's

7.2.1 Resultados usando la versión normal

1NN						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	95.48	0.00	71.61	73.13	97.50	9.69E-04
2	94.19	0.00	70.65	65.63	87.50	6.23E-04
3	96.13	0.00	72.10	73.13	97.50	5.64E-04
4	94.19	0.00	70.65	75.00	100.00	5.10E-04
5	96.88	0.00	72.66	68.57	91.43	4.98E-04
Media	95.38	0.00	71.53	71.09	94.79	6.33E-04
Std dev	1.19	0.00	0.89	3.87	5.15	1.95E-04

Relief						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	96.13	0.00	72.10	73.13	97.50	3.76E-03
2	95.48	0.00	71.61	67.50	90.00	4.50E-03
3	96.77	0.00	72.58	71.25	95.00	3.59E-03
4	94.19	0.00	70.65	75.00	100.00	4.51E-03
5	96.88	0.00	72.66	68.57	91.43	3.75E-03
Media	95.89	0.00	71.92	71.09	94.79	4.02E-03
Std dev	1.10	0.00	0.83	3.11	4.15	4.48E-04

BL						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	94.19	81.82	91.10	89.83	92.50	0.18
2	96.13	77.27	91.41	90.57	95.00	0.13
3	98.71	77.27	93.35	88.69	92.50	0.15
4	96.13	72.73	90.28	83.81	87.50	0.11
5	98.75	72.73	92.24	91.04	97.14	0.16
Media	96.78	76.36	91.68	88.79	92.93	0.14
Std dev	1.95	3.80	1.17	2.92	3.60	0.03

AGG-AC						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	96.77	81.82	93.04	95.45	100.00	1.92
2	97.42	68.18	90.11	86.42	92.50	2.08
3	96.77	77.27	91.90	88.69	92.50	1.98
4	95.48	72.73	89.79	87.56	92.50	1.97
5	96.88	77.27	91.97	87.89	91.43	2.07
Media	96.67	75.45	91.36	89.20	93.79	2.00
Std dev	0.71	5.18	1.37	3.59	3.50	0.07

AGG-BLX						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	96.13	77.27	91.41	92.44	97.50	2.03
2	92.90	81.82	90.13	89.83	92.50	1.89
3	97.42	77.27	92.38	92.44	97.50	2.02
4	98.06	81.82	94.00	89.83	92.50	1.96
5	99.38	77.27	93.85	90.03	94.29	2.07
Media	96.78	79.09	92.36	90.92	94.86	2.00
Std dev	2.46	2.49	1.64	1.40	2.52	0.07

AGE-AC						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	90.32	86.36	89.33	89.09	90.00	2.54
2	85.16	90.91	86.60	82.73	80.00	2.38
3	89.03	81.82	87.23	91.70	95.00	2.53
4	87.74	86.36	87.40	85.34	85.00	2.48
5	86.25	86.36	86.28	88.02	88.57	2.53
Media	87.70	86.36	87.37	87.38	87.71	2.49
Std dev	2.07	3.21	1.19	3.46	5.61	0.07

AGE-BLX						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	100.00	63.64	90.91	87.16	95.00	3.06
2	98.06	68.18	90.59	80.80	85.00	2.97
3	95.48	77.27	90.93	90.57	95.00	2.99
4	93.55	90.91	92.89	88.35	87.50	2.58
5	98.75	77.27	93.38	83.60	85.71	2.94
Media	97.17	75.45	91.74	86.10	89.64	2.91
Std dev	2.61	10.46	1.29	3.89	4.97	0.19

AM-All						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	97.42	77.27	92.38	90.57	95.00	1.97
2	97.42	77.27	92.38	90.57	95.00	1.99
3	98.71	77.27	93.35	92.44	97.50	2.01
4	98.06	81.82	94.00	89.83	92.50	1.91
5	97.50	86.36	94.72	88.02	88.57	1.99
Media	97.82	80.00	93.37	90.29	93.71	1.97
Std dev	0.56	4.07	1.02	1.59	3.37	0.04

AM-Random						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	98.06	72.73	91.73	89.43	95.00	2.00
2	95.48	81.82	92.07	91.70	95.00	1.97
3	98.06	77.27	92.87	92.44	97.50	1.97
4	98.06	81.82	94.00	89.83	92.50	1.89
5	99.38	77.27	93.85	83.60	85.71	2.06
Media	97.81	78.18	92.90	89.40	93.14	1.98
Std dev	1.42	3.80	1.02	3.48	4.51	0.06

AM-Best						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	97.42	77.27	92.38	94.32	100.00	1.97
2	96.77	77.27	91.90	88.69	92.50	1.96
3	96.77	72.73	90.76	87.56	92.50	2.06
4	98.06	81.82	94.00	93.58	97.50	1.96
5	97.50	77.27	92.44	77.18	77.14	2.13
Media	97.31	77.27	92.30	88.26	91.93	2.02
Std dev	0.55	3.21	1.17	6.87	8.88	0.08

7.2.2 Resultados extra usando la versión mejorada

AGG-AC						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	97.42	77.27	92.38	90.57	95.00	2.15
2	98.06	77.27	92.87	88.69	92.50	2.08
3	98.71	77.27	93.35	92.44	97.50	2.04
4	97.42	81.82	93.52	89.83	92.50	1.99
5	98.13	81.82	94.05	93.31	97.14	2.08
Media	97.95	79.09	93.23	90.97	94.93	2.07
Std dev	0.54	2.49	0.64	1.89	2.41	0.06

AGG-BLX						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	96.77	81.82	93.04	86.08	87.50	2.10
2	96.13	81.82	92.55	91.70	95.00	2.10
3	98.71	77.27	93.35	92.44	97.50	2.15
4	98.06	81.82	94.00	89.83	92.50	2.03
5	98.75	81.82	94.52	89.03	91.43	2.12
Media	97.69	80.91	93.49	89.82	92.79	2.10
Std dev	1.18	2.03	0.78	2.50	3.77	0.04

AGE-AC						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	91.61	81.82	89.16	89.83	92.50	2.81
2	92.90	81.82	90.13	89.83	92.50	2.77
3	92.90	81.82	90.13	87.95	90.00	2.83
4	96.13	72.73	90.28	85.68	90.00	2.88
5	95.00	81.82	91.70	84.74	85.71	2.89
Media	93.71	80.00	90.28	87.61	90.14	2.83
Std dev	1.82	4.07	0.91	2.34	2.77	0.05

AGE-BLX						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	99.35	72.73	92.70	93.18	100.00	3.12
2	96.13	86.36	93.69	89.09	90.00	2.91
3	98.06	77.27	92.87	92.44	97.50	3.03
4	97.42	81.82	93.52	89.83	92.50	3.09
5	98.75	81.82	94.52	93.31	97.14	3.12
Media	97.94	80.00	93.46	91.57	95.43	3.06
Std dev	1.25	5.18	0.73	1.97	4.07	0.09

AM-All						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	97.42	81.82	93.52	91.70	95.00	2.17
2	96.77	77.27	91.90	83.07	85.00	2.17
3	98.06	77.27	92.87	92.44	97.50	2.15
4	98.71	81.82	94.49	89.83	92.50	2.10
5	98.13	81.82	94.05	89.03	91.43	2.24
Media	97.82	80.00	93.36	89.21	92.29	2.17
Std dev	0.74	2.49	1.02	3.70	4.70	0.05

AM-Random						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	98.71	77.27	93.35	92.44	97.50	2.07
2	96.13	81.82	92.55	91.70	95.00	1.96
3	98.06	81.82	94.00	91.70	95.00	2.02
4	97.42	81.82	93.52	89.83	92.50	1.99
5	98.75	81.82	94.52	89.03	91.43	2.05
Media	97.81	80.91	93.59	90.94	94.29	2.02
Std dev	1.09	2.03	0.74	1.44	2.38	0.04

AM-Best						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	96.77	81.82	93.04	86.08	87.50	1.95
2	96.13	81.82	92.55	91.70	95.00	1.92
3	98.06	77.27	92.87	92.44	97.50	1.99
4	98.71	81.82	94.49	89.83	92.50	2.02
5	96.88	86.36	94.25	88.02	88.57	2.07
Media	97.31	81.82	93.44	89.62	92.21	1.99
Std dev	1.05	3.21	0.87	2.62	4.22	0.06

7.2.3 Resultados globales

	Parkinsons					
Algoritmo	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1NN	95.38	0.00	71.53	71.09	94.79	6.33E-04
Relief	95.89	0.00	71.92	71.09	94.79	4.02E-03
BL	96.78	76.36	91.68	88.79	92.93	0.14
AGG-AC	96.67	75.45	91.36	89.20	93.79	2.00
AGG-AC Mejor	97.95	79.09	93.23	90.97	94.93	2.07
AGG-BLX	96.78	79.09	92.36	90.92	94.86	2.00
AGG-BLX Mejor	97.69	80.91	93.49	89.82	92.79	2.10
AGE-AC	87.70	86.36	87.37	87.38	87.71	2.49
AGE-AC Mejor	93.71	80.00	90.28	87.61	90.14	2.83
AGE-BLX	97.17	75.45	91.74	86.10	89.64	2.91
AGE-BLX Mejor	97.94	80.00	93.46	91.57	95.43	3.06
AM-All	97.82	80.00	93.37	90.29	93.71	1.97
AM-All Mejor	97.82	80.00	93.36	89.21	92.29	2.17
AM-Rand	97.81	78.18	92.90	89.40	93.14	1.98
AM-Rand Mejor	97.81	80.91	93.59	90.94	94.29	2.02
AM-Best	97.31	77.27	92.30	88.26	91.93	2.02
AM-Best Mejor	97.31	81.82	93.44	89.62	92.21	1.99

7.3 Breast Cancer

7.3.1 Resultados usando la versión normal

1NN						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	95.15	0.00	71.37	73.04	97.39	6.60E-03
2	94.93	0.00	71.20	72.39	96.52	4.21E-03
3	96.92	0.00	72.69	69.13	92.17	3.31E-03
4	93.83	0.00	70.37	73.70	98.26	3.59E-03
5	95.43	0.00	71.58	70.18	93.58	4.23E-03
Media	95.25	0.00	71.44	71.69	95.59	4.39E-03
Std dev	1.11	0.00	0.83	1.95	2.60	1.30E-03

Relief						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	94.71	3.33	71.87	73.88	97.39	2.67E-02
2	94.27	3.33	71.54	73.88	97.39	2.62E-02
3	98.24	13.33	77.01	70.51	89.57	2.63E-02
4	94.05	0.00	70.54	73.04	97.39	2.52E-02
5	96.09	0.00	72.07	70.87	94.50	2.43E-02
Media	95.47	4.00	72.60	72.44	95.25	2.57E-02
Std dev	1.74	5.48	2.53	1.63	3.41	9.72E-04

BL						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	96.04	86.67	93.69	92.10	93.91	0.78
2	96.04	86.67	93.69	92.10	93.91	1.22
3	98.02	76.67	92.68	86.99	90.43	0.79
4	96.48	90.00	94.86	90.98	91.30	1.16
5	96.74	80.00	92.55	90.87	94.50	0.84
Media	96.66	84.00	93.50	90.61	92.81	0.95
Std dev	0.82	5.48	0.93	2.11	1.82	0.21

AGG-AC						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	96.48	83.33	93.19	91.27	93.91	13.10
2	96.04	90.00	94.53	94.89	96.52	12.52
3	98.46	83.33	94.68	90.62	93.04	13.39
4	95.59	90.00	94.20	92.28	93.04	13.63
5	97.83	80.00	93.37	91.56	95.41	13.25
Media	96.88	85.33	93.99	92.12	94.39	13.18
Std dev	1.22	4.47	0.68	1.66	1.54	0.42

AGG-BLX						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	96.70	80.00	92.52	91.09	94.78	12.33
2	95.15	83.33	92.20	91.27	93.91	12.35
3	98.24	83.33	94.51	85.40	86.09	12.87
4	96.04	86.67	93.69	94.06	96.52	10.97
5	96.30	83.33	93.06	90.33	92.66	12.25
Media	96.49	83.33	93.20	90.43	92.79	12.15
Std dev	1.13	2.36	0.93	3.15	4.00	0.70

AGE-AC						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	90.97	93.33	91.56	91.81	91.30	9.54
2	91.41	86.67	90.22	92.75	94.78	10.65
3	96.92	86.67	94.35	92.10	93.91	10.59
4	88.55	90.00	88.91	90.33	90.43	10.10
5	93.04	86.67	91.45	91.85	93.58	10.87
Media	92.18	88.67	91.30	91.77	92.80	10.35
Std dev	3.10	2.98	2.02	0.89	1.84	0.53

AGE-BLX						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	94.27	86.67	92.37	93.41	95.65	10.77
2	94.93	83.33	92.03	89.96	92.17	11.49
3	97.36	90.00	95.52	89.02	88.70	9.90
4	95.15	86.67	93.03	90.14	91.30	10.72
5	93.91	90.00	92.93	93.37	94.50	10.31
Media	95.13	87.33	93.18	91.18	92.46	10.64
Std dev	1.34	2.79	1.37	2.06	2.73	0.59

AM-All						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	97.14	90.00	95.35	94.89	96.52	11.90
2	96.70	86.67	94.19	93.41	95.65	11.90
3	97.80	86.67	95.01	86.23	86.09	12.17
4	96.26	90.00	94.69	92.28	93.04	12.82
5	96.30	90.00	94.73	93.37	94.50	12.18
Media	96.84	88.67	94.80	92.04	93.16	12.20
Std dev	0.64	1.83	0.43	3.37	4.16	0.38

AM-Random						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	97.14	90.00	95.35	95.54	97.39	9.83
2	96.48	86.67	94.02	92.75	94.78	10.22
3	98.24	86.67	95.35	88.84	89.57	10.24
4	96.70	90.00	95.02	90.98	91.30	9.99
5	95.87	86.67	93.57	90.47	91.74	10.30
Media	96.88	88.00	94.66	91.72	92.96	10.12
Std dev	0.88	1.83	0.82	2.55	3.11	0.20

AM-Best						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	97.14	90.00	95.35	94.89	96.52	9.47
2	96.48	86.67	94.02	92.75	94.78	10.16
3	98.24	86.67	95.35	89.49	90.43	9.96
4	97.80	90.00	95.85	91.63	92.17	9.92
5	97.17	83.33	93.71	93.08	96.33	11.55
Media	97.36	87.33	94.86	92.37	94.05	10.21
Std dev	0.68	2.79	0.93	1.99	2.67	0.79

7.3.2 Resultados extra usando la versión mejorada

AGG-AC						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	96.04	90.00	94.53	92.93	93.91	11.24
2	96.48	86.67	94.02	92.75	94.78	11.42
3	99.12	86.67	96.01	89.49	90.43	10.78
4	95.59	93.33	95.03	89.86	88.70	10.84
5	96.74	83.33	93.39	91.02	93.58	12.27
Media	96.79	88.00	94.59	91.21	92.28	11.31
Std dev	1.37	3.80	1.00	1.60	2.59	0.60

AGG-BLX						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	97.14	90.00	95.35	94.89	96.52	10.71
2	96.48	90.00	94.86	93.59	94.78	10.44
3	97.80	90.00	95.85	92.28	93.04	10.39
4	96.70	90.00	95.02	90.98	91.30	10.69
5	96.09	90.00	94.57	94.75	96.33	10.43
Media	96.84	90.00	95.13	93.30	94.40	10.53
Std dev	0.66	0.00	0.49	1.67	2.22	0.16

AGE-AC						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	93.83	90.00	92.87	91.63	92.17	12.11
2	92.07	93.33	92.39	94.42	94.78	11.59
3	96.04	90.00	94.53	90.33	90.43	12.01
4	94.93	90.00	93.70	94.89	96.52	11.97
5	92.83	90.00	92.12	91.31	91.74	12.63
Media	93.94	90.67	93.12	92.52	93.13	12.06
Std dev	1.59	1.49	0.99	2.02	2.47	0.37

AGE-BLX						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	96.70	90.00	95.02	94.89	96.52	12.04
2	95.37	90.00	94.03	92.93	93.91	12.28
3	96.70	90.00	95.02	91.63	92.17	12.29
4	96.26	90.00	94.69	92.28	93.04	12.22
5	95.87	90.00	94.40	94.06	95.41	12.38
Media	96.18	90.00	94.63	93.16	94.21	12.24
Std dev	0.57	0.00	0.42	1.32	1.76	0.13

AM-All						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	97.14	90.00	95.35	94.89	96.52	11.63
2	96.04	90.00	94.53	94.24	95.65	12.08
3	99.12	86.67	96.01	89.49	90.43	13.11
4	96.26	90.00	94.69	92.28	93.04	13.63
5	96.52	90.00	94.89	95.44	97.25	12.46
Media	97.01	89.33	95.09	93.27	94.58	12.58
Std dev	1.25	1.49	0.60	2.42	2.81	0.80

AM-Random						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	97.36	90.00	95.52	94.89	96.52	10.71
2	96.48	90.00	94.86	94.24	95.65	10.92
3	97.58	86.67	94.85	85.58	85.22	11.00
4	96.70	90.00	95.02	90.98	91.30	10.44
5	96.09	90.00	94.57	90.62	90.83	11.00
Media	96.84	89.33	94.96	91.26	91.90	10.81
Std dev	0.62	1.49	0.35	3.70	4.52	0.24

AM-Best						
Partición	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1	96.92	90.00	95.19	94.89	96.52	9.79
2	96.04	90.00	94.53	94.89	96.52	10.32
3	98.02	90.00	96.01	93.59	94.78	10.21
4	98.02	90.00	96.01	91.63	92.17	9.72
5	96.09	90.00	94.57	95.44	97.25	10.64
Media	97.01	90.00	95.26	94.09	95.45	10.14
Std dev	0.98	0.00	0.73	1.53	2.04	0.38

7.3.3 Resultados globales

	Breast Cancer					
Algoritmo	Tasa de clasificación	Tasa de reducción	Fitness train	Fitness test	Accuracy	Tiempo de ejecución
1NN	95.25	0.00	71.44	71.69	95.59	4.39E-03
Relief	95.47	4.00	72.60	72.44	95.25	2.57E-02
BL	96.66	84.00	93.50	90.61	92.81	0.95
AGG-AC	96.88	85.33	93.99	92.12	94.39	13.18
AGG-AC Mejor	96.79	88.00	94.59	91.21	92.28	11.31
AGG-BLX	96.49	83.33	93.20	90.43	92.79	12.15
AGG-BLX Mejor	96.84	90.00	95.13	93.30	94.40	10.53
AGE-AC	92.18	88.67	91.30	91.77	92.80	10.35
AGE-AC Mejor	93.94	90.67	93.12	92.52	93.13	12.06
AGE-BLX	95.13	87.33	93.18	91.18	92.46	10.64
AGE-BLX Mejor	96.18	90.00	94.63	93.16	94.21	12.24
AM-All	96.84	88.67	94.80	92.04	93.16	12.20
AM-All Mejor	97.01	89.33	95.09	93.27	94.58	12.58
AM-Rand	96.88	88.00	94.66	91.72	92.96	10.12
AM-Rand Mejor	96.84	89.33	94.96	91.26	91.90	10.81
AM-Best	97.36	87.33	94.86	92.37	94.05	10.21
AM-Best Mejor	97.01	90.00	95.26	94.09	95.45	10.14

7.4 Todos los resultados globales

	Breast Cancer			Ecoli			Parkinsons		
Algoritmo	Tasa clas	Tasa red	Fitness	Tasa clas	Tasa red	Fitness	Tasa clas	Tasa red	Fitness
1NN	95.25	0.00	71.69	80.88	0.00	60.61	95.38	0.00	71.09
Relief	95.47	4.00	72.44	79.77	28.57	66.90	95.89	0.00	71.09
BL	96.66	84.00	90.61	79.55	54.29	74.01	96.78	76.36	88.79
AGG-AC	96.88	85.33	92.12	78.01	62.86	70.83	96.67	75.45	89.20
AGG-AC Mejor	96.79	88.00	91.21	80.22	57.14	72.50	97.95	79.09	90.97
AGG-BLX	96.49	83.33	90.43	80.44	51.43	72.18	96.78	79.09	90.92
AGG-BLX Mejor	96.84	90.00	93.30	80.22	57.14	72.72	97.69	80.91	89.82
AGE-AC	92.18	88.67	91.77	76.33	51.43	70.37	87.70	86.36	87.38
AGE-AC Mejor	93.94	90.67	92.52	77.01	60.00	71.89	93.71	80.00	87.61
AGE-BLX	95.13	87.33	91.18	77.44	57.14	71.38	97.17	75.45	86.10
AGE-BLX Mejor	96.18	90.00	93.16	79.84	57.14	72.29	97.94	80.00	91.57
AM-All	96.84	88.67	92.04	79.77	57.14	72.07	97.82	80.00	90.29
AM-All Mejor	97.01	89.33	93.27	80.14	57.14	72.50	97.82	80.00	89.21
AM-Rand	96.88	88.00	91.72	78.68	60.00	70.79	97.81	78.18	89.40
AM-Rand Mejor	96.84	89.33	91.26	80.22	57.14	73.36	97.81	80.91	90.94
AM-Best	97.36	87.33	92.37	78.94	60.00	71.46	97.31	77.27	88.26
AM-Best Mejor	97.01	90.00	94.09	80.29	57.14	72.50	97.31	81.82	89.62

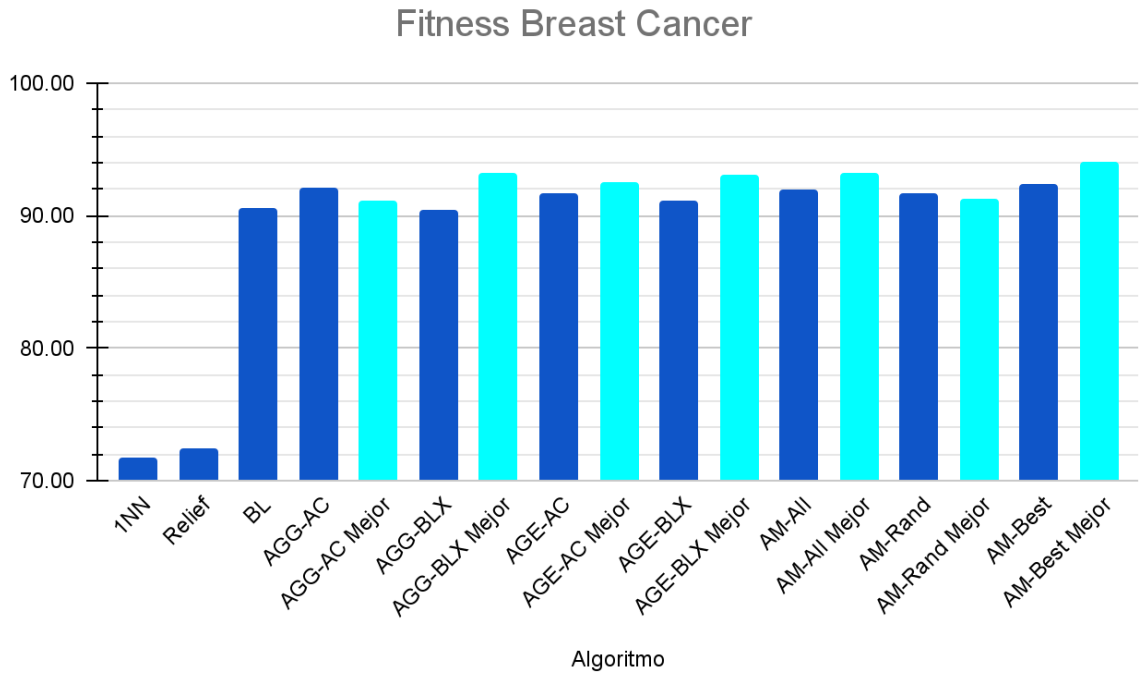
8. Análisis de resultados

8.1 1-NN, Greedy y BL frente a los algoritmos genéticos

Los algoritmos genéticos y BL suelen tener resultados similares en cuanto a fitness. 1NN y Greedy son muy malos en fitness porque no utilizan o casi no utilizan información del problema.

Los algoritmos genéticos suelen mejorar al BL porque exploran soluciones, y es más fácil encontrar óptimos locales mejores a los que encuentra el BL. Solamente en el database Ecoli el BL sea mejor a los genéticos, que creo que se deba a la semilla específica utilizada. En cuanto a los meméticos siempre mejoran a los genéticos pero en alguna de las variantes, generalmente cuando se hace BL a todos los cromosomas o a los mejores.

8.2 Análisis en Breast Cancer

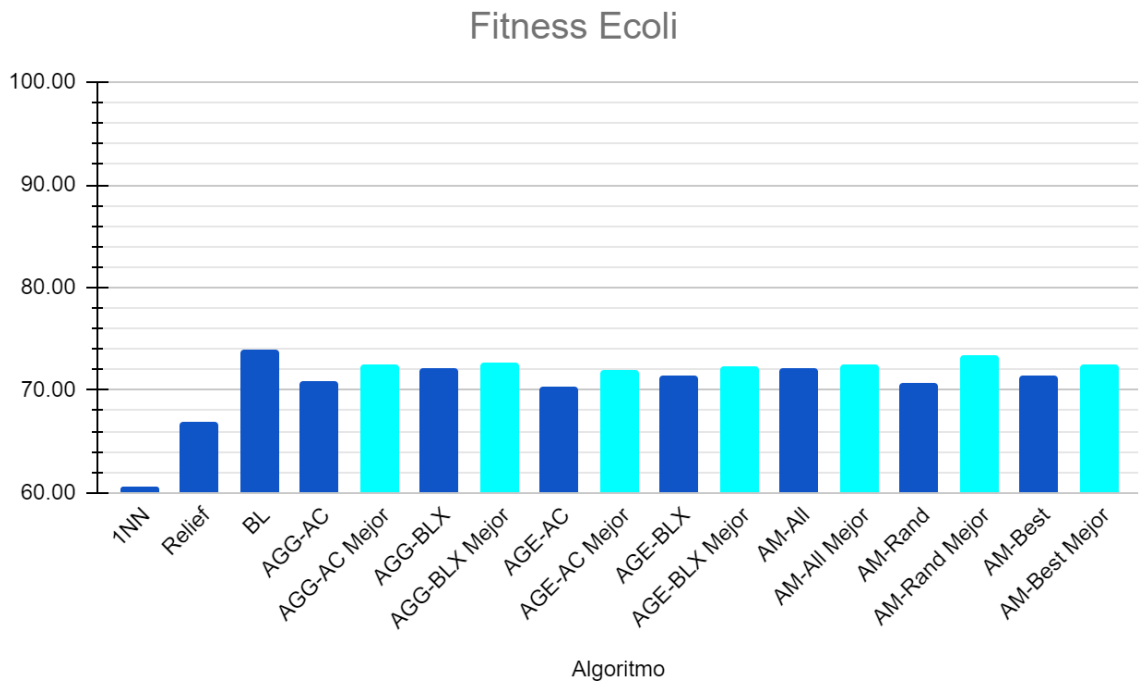


Comparación en fitness de los algoritmos en Breast Cancer

Se puede observar como la mayoría de los genéticos mejoran a BL. Además, también se observa como en la versión normal de los algoritmos genéticos, el cruce AC es mejor que BLX en todos los algoritmos, pero pasa lo contrario en la versión mejorada. Veremos como después en otros databases es mejor el BLX en la mayoría de casos. En la versión mejorada casi siempre es mejor el BLX.

Los algoritmos que mejores resultados dan son los meméticos destacando AM-All, la versión mejorada de AM-Best y AGG-AC.

8.3 Análisis en Ecoli

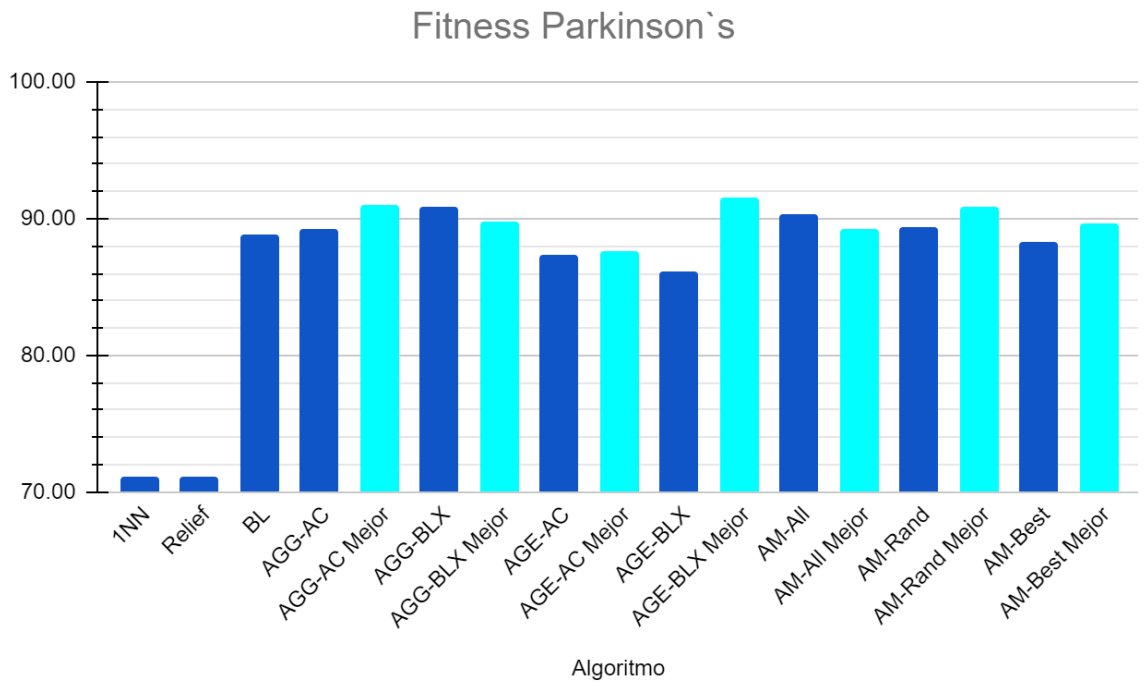


Comparación en fitness de los algoritmos en Ecoli

Aquí vemos cómo el BL es mejor que los genéticos pero creo que es por suerte ya que probando con otras semillas da peores resultados que los genéticos. En general, el operador AC es peor que el BLX. Entre los mejores están AGG-BLX, tanto la versión normal y mejorada y los algoritmos meméticos AM-All y AM-Random usando la versión mejorada, dando los mejores resultados entre los genéticos.

Cabe destacar que los algoritmos genéticos no mejoraron mucho al BL en este dataset.

8.4 Análisis en Parkinson`s



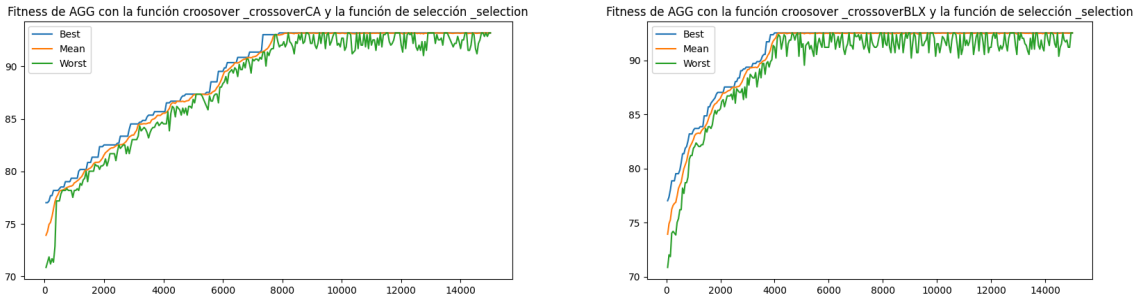
Comparación en fitness de los algoritmos en Parkinson´s

En Parkinson`s se ve como AGE en casi todas las variantes en el peor de entre todos los algoritmos genéticos, excepto AGE-BLX con la versión mejorada, que es el algoritmo que mejor resultado ha dado entre todos los algoritmos. Esto creo que se debe a suerte, tanto para mal como para bien, y a que AGE en general tenga menor diversidad que AGG, cosa que aclararé más tarde.

Los algoritmos meméticos no consiguen superar los resultados de AGG-BLX, pero en las versiones mejoradas los algoritmos meméticos sí que suelen mejorar al AGG mejorado.

Cabe destacar que en este database la versión mejorada mejora en menor medida que en otros databases, salvo en algunos casos.

8.5 Comparación del cruce AC y BLX



Comparación de la convergencia entre el cruce AC y el cruce BLX en AGG(*)

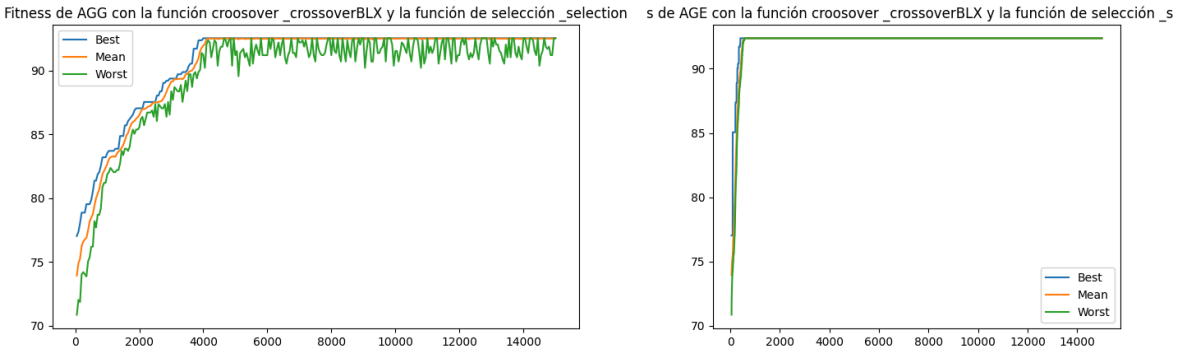
Observando la gráfica de convergencia se puede observar que BLX converge antes, y hace mejoras de manera más continuas. En cuanto a diversidad, se puede observar que son iguales, ya que la diferencia entre el peor, el medio y el mejor es muy poco.

Analizando los gráficos, es más fácil mejorar el algoritmo con BLX, porque, al converger antes, se podría hacer un reinicio de la población, para explorar más y poder encontrar posibles mejores soluciones.

En los resultados, BLX ha sido mejor que AC, salvo en Breast Cancer usando la versión normal, y otras ocasiones que creo que se deban a la suerte.

Como conclusión BLX es mejor que AC al converger más rápido que AC, manteniendo la misma diversidad que AC.

8.6 Comparación de los algoritmos AGG y AGE



Comparación de la convergencia entre AGG-BLX y AGE-BLX(*)

Se puede observar que AGE converge muy rápido (en menos de 2000 evaluaciones), mientras que AGG tarda bastante más (alrededor de 4000 evaluaciones).

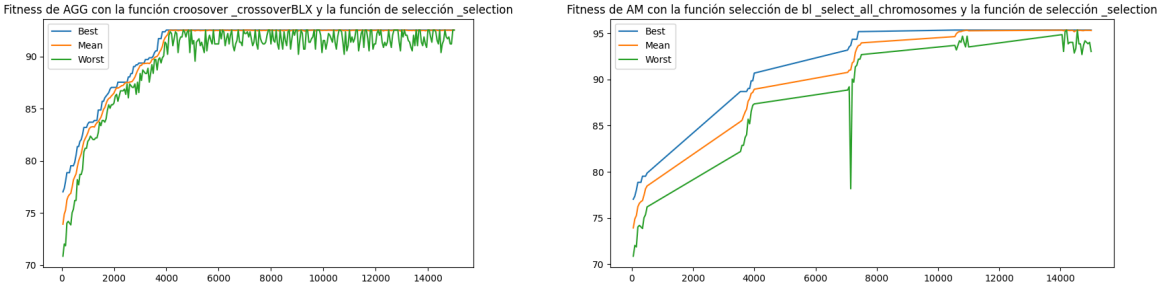
En cuanto a diversidad, AGG tiene bastante más diversidad que AGE, el cual, en las soluciones finales tienen casi la misma valoración, lo que hace más difícil mejorar a partir

del cruce de los individuos de la población. Pienso que AGE sería mejor con reinicios de la población cuando la población converja.

En general, AGG da mejores resultados que AGE, salvo en algunas excepciones que creo que se deban a la suerte.

Como conclusión AGG es mejor a AGE, ya que tiene mayor diversidad, aunque creo que AGE sería mejor a AGG, aplicando un reinicio a la población cada vez que la población converja.

8.7 Comparación de los algoritmos AGG-BLX y AMs



Comparación de la convergencia entre AGG-BLX y AM-All(*)

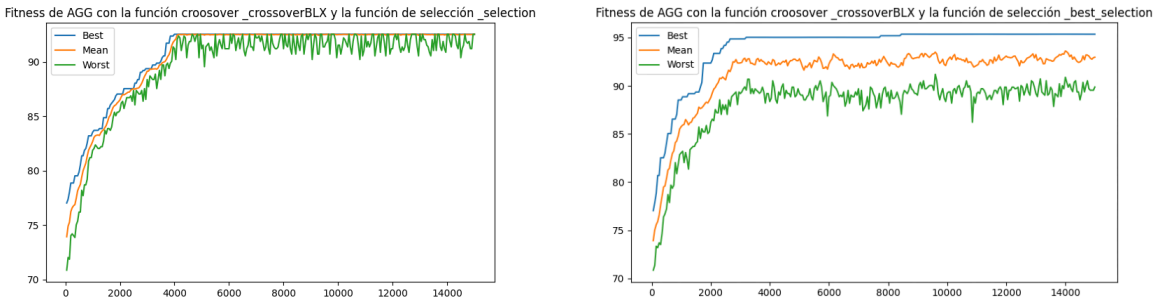
Observando los gráficos, se observa que el AM converge más lentamente, pero alcanza mejores resultados que AGG, eso indica que las evaluaciones que AGG no mejora explorando, se utilizan en el BL del algoritmo memético para mejorar localmente, en el entorno de las soluciones. También podemos observar que AM tiene mayor diversidad.

Esto puede solucionar el problema del AGG de converger tan rápido, y así no tener que necesitar un reinicio de la población para intentar mejorar los resultados.

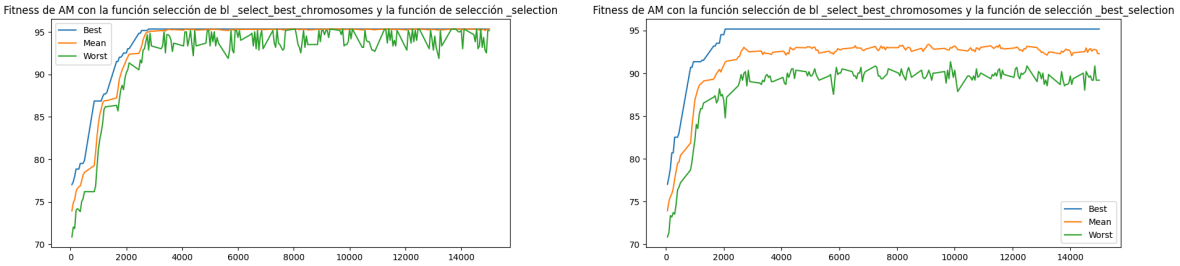
En los resultados, generalmente, los AMs mejoran al AGG-BLX.

Como conclusión AM es mejor que AGG ya que explora localmente, tiene mayor diversidad, y utiliza evaluaciones que AGG no hubiera aprovechado para mejorar.

8.8 Comparación entre la versión normal y mejorada



Comparación de la convergencia entre el AGG-BLX y AGG-BLX mejorado(*)



Comparación de la convergencia entre el AM-Best y AM-Best mejorado(*)

En ambas comparaciones se puede observar que la versión mejorada tiene mayor diversidad que la versión normal. Además de eso el algoritmo converge antes en la versión mejorada.

Esto se puede deber a la alta tasa de mutaciones, que eleva en gran medida la diversidad, permitiendo al algoritmo explorar mucho más que al original.

En general, en los resultados la versión mejorada tiene mejor fitness que la normal.

En conclusión, la versión mejorada es mejor, porque le permite al algoritmo converger antes y explorar más.

***(*)** Las gráficas pertenecen a Breast Cancer utilizando como partición de validación la partición 1.*

9. Conclusión

Los algoritmos genéticos son, en general, mejores que los algoritmos implementados en la práctica 1.

Entre los algoritmos genéticos, el mejor cruce es el BLX, por converger antes, y el mejor algoritmo el AGG por tener mayor diversidad.

En referencia a los algoritmos meméticos, mejorar al AGG-BLX, porque utilizan las evaluaciones que casi no se usan para mejorar en AGG, para mejorar localmente.

La versión mejorada es mejor a la versión normal, gracias a aportar mayor diversidad a la población.