# Optimización de la mariposa monarca

Miguel García López

Junio 2022

# Índice

**1. Introducción** **1**

**2. Comportamiento migratorio** **2**
2.1. Operador de migración . . . . . 2
2.2. Operador de ajuste . . . . . 3

**3. Representación esquemática** **3**

**4. Adaptación para APC** **5**
4.1. Implementación . . . . . 5
4.2. Comparación con otros algoritmos . . . . . 6

**5. Hibridación con BL** **7**

**6. Mejoras de diseño/comportamiento** **8**

# 1. Introducción

En la naturaleza, la población de mariposas monarcas del este de América del Norte es conocida por su migración hacia el sur durante el final del verano/otoño desde el norte de EEUU y el sur de Canadá hasta México, cubriendo miles de millas. En este documento se explicará un tipo de algoritmo metaheurístico inspirado en la naturaleza, llamado optimización de la mariposa monarca (MBO). En MBO, todos los individuos de la mariposa monarca se encuentran en dos tierras distintas, a saber, el sur de Canadá y el norte de EEUU (Tierra 1) y México (Tierra 2). En consecuencia, las posiciones de las mariposas monarca se actualizan de dos maneras. En primer lugar, los descendientes son generados (actualización de población) por el operador de migración, que puede ajustarse por la tasa de migración. A continuación, se ajustan las posiciones para mariposas de la subpoblación 2 (tierra 2) por medio del operador de ajuste. Para mantener la población sin cambios y minimizar las evaluaciones de aptitud, la suma de las mariposas recién generadas de estas dos formas permanece igual a la población original.

Además, el operador de migración y el operador de ajuste de mariposas se pueden implementar simultáneamente. Por lo tanto, el método MBO es ideal para el procesamiento en paralelo y es capaz de compensar de manera equitativa entre intensificación y diversificación, un fenómeno muy importante en el campo de las metaheurísticas.

# 2. Comportamiento migratorio

El comportamiento migratorio de las mariposas monarcas puede ser descrito en las siguientes reglas:

1. Todas las mariposas monarcas están localizadas en la tierra 1 o en la tierra 2. La población total es la suma de ambas subpoblaciones.
2. Cada mariposa monarca hijo es generada por el operador de migración de una mariposa monarca de la tierra 1 o la tierra 2.
3. Para mantener la población inalterada, cuando un hijo es generado una mariposa monarca padre muere. En MBO esto se puede conseguir generando un hijo si tiene un fitness mejor que el del padre, de lo contrario el hijo es descartado.
4. Los individuos con el mejor fitness se mueven automáticamente a la siguiente generación y no pueden ser cambiados. Esto garantiza la calidad y efectividad del algoritmo, pues la población no se deteriora con el paso de generaciones.

El operador migratorio y el de ajuste se implementan de forma paralela, de modo que ambos procesos ocurren a la vez, de igual forma que con las mariposas monarcas reales.

## 2.1. Operador de migración

Las mariposas monarcas migran de la tierra 1 a la tierra 2 durante el mes de Abril, y de la tierra 2 a la tierra 1 durante Septiembre. Por lo que se podría decir que las mariposas de la tierra 1 están durante 5 meses allí y las de la tierra 2 durante 7 meses. El número de mariposas monarcas de la población se puede representar como ceil$(p*NP)(NP_1)$ y $NP-NP_1(NP_2)$. Donde ceil redondea el número al entero más cercano y mayor o igual que el número, NP es el número total de mariposas y p es el ratio de matiposas en la tierra 1. Cada población de cada tierra se llaman subpoblaciones 1 y 2. El proceso de migración es el que sigue:

$$x_{i,k}^{t+1} = x_{r_1,k}^{t} \tag{1}$$

donde $x_{i,k}^{t+1}$ indica el $k$ elemento de $x_i$ en la generación $t+1$ que representa la posición de la mariposa $i$. El otro término indica con $r_1$ la mariposa nueva generada aleatoriamente desde la subplobación 1. Cuando $r \leq p$, el elemento $k$ en la nueva mariposa monarca es generado por la ecuación (1). $r$ puede ser calculado de la siguiente forma:

$$r = rand * peri \tag{2}$$

Peri indica el período de migración, se suele fijar a 1,2 (12 meses). Rand es un número aleatorio generado por una distribución uniforme. Si $r > p$, el elemento $k$ se genera de la siguiente forma:

$$x_{i,k}^{t+1} = x_{r_2,k}^{t} \tag{3}$$

Donde la mariposa $r_2$ es seleccionada aleatoriamente de la subpoblación 2.

Podemos ver pues que el peso que tiene cada subpoblación viene determinado por el ratio $p$. A mayor sea éste mayor importancia tendrá la subpoblación 1. Normalmente suele fijarse a $p = 5/12$ debido al período migratorio mencionado anteriormente (5 meses de 12 en la subpoblación 1).

`Operador de migración Pseudocódigo:`

**for** i=1,...,$NP_1$ **do**
  **for** k=1,...,D **do**
    $r = rand * peri$
    **if** $r \leq p$ **then**
      Seleccionar aleatoriamente una mariposa r1.
      Generar el $k$ elemento de $x_i^t + 1$ con ecuación (1).
    **else**
      Seleccionar aleatoriamente una mariposa r2.
      Generar el $k$ elemento de $x_i^t + 1$ con ecuación (3).

```
10:         end if
11:     end for
12: end for
```

## 2.2. Operador de ajuste

La posición de las mariposas monarca es ajustada por este operador. El proceso es el siguiente: para toda mariposa monarca $j$, si un número aleatorio generado es menor o igual que el ratio $p$ entonces:

$$x_{j,k}^{t+1} = x_{best,k}^{t} \tag{4}$$

El segundo término, $(x_{best,k}^{t})$, indica el $k$ elemento de $x_{best}$, que es la mejor mariposa de ambas tierras. Si por el contrario, el número generado es mayor que $p$, entonces:

$$x_{j,k}^{t+1} = x_{r_3,k}^{t} \tag{5}$$

Donde se escoge una mariposa $r_3$ aleatoria de la tierra 2. Bajo esta condición, si el número aleatorio $n$, es mayor que $BAR$, siendo $BAR$ el valor de ajuste de una mariposa, podemos actualizar de la siguiente manera:

$$x_{j,k}^{t+1} = x_{j,k}^{t+1} + \alpha(dx_k - 0{,}5) \tag{6}$$

Dx indica la longitud del paso de las mariposas monarcas, este puede ser calculado con el vuelo Lévy. Este es un tipo de paseo aleatorio en el cual los incrementos son distribuidos de acuerdo a una distribución de probabilidad. Podríamos calcular $dx = Levy(x_j^t)$. El valor $\alpha$ es una variable peso que se calcula $\alpha = S_{max}/t^2$ donde $S_{max}$ es el máximo paso que una mariposa individual puede dar, y $t$ es la generación actual. A mayor alpha mayor longitud de búsqueda.

`Operador de ajuste Pseudocódigo:`

```
1: for i=1,...,NP_2 do
2:     Calcular dx.
3:     Calcular α.
4:     for k=1,...,D do
5:         r = randomdistribution
6:         if r ≤ p then
7:             Generar el k elemento de x_j^{t+1} como en la eq (4)
8:         else
9:             Seleccionar aleatoriamente una mariposa r3 de subpoblación 2.
10:            Generar el k elemento de x_i^t + 1 con ecuación (5).
11:            if r > BAR then
12:                x_{j,k}^{t+1} = x_{j,k}^{t+1} + α(dx_k − 0,5)
13:            end if
14:        end if
15:    end for
16: end for
```

# 3. Representación esquemática

La posición de todas las mariposas monarcas se va actualizando paso a paso hasta que ciertas condiciones son satisfacidas o se alcanzan unas iteraciones (generaciones) máximas. Normalmente se suele utilizar para inicializar los parámetros los siguientes valores:

1. Paso máximo $(S_{max}) = 1{,}0$

2. Ratio de ajuste $(BAR) = \frac{5}{12}$
3. Periodo de migración $(peri) = 1{,}2$
4. Ratio de migración $(p) = \frac{5}{12}$

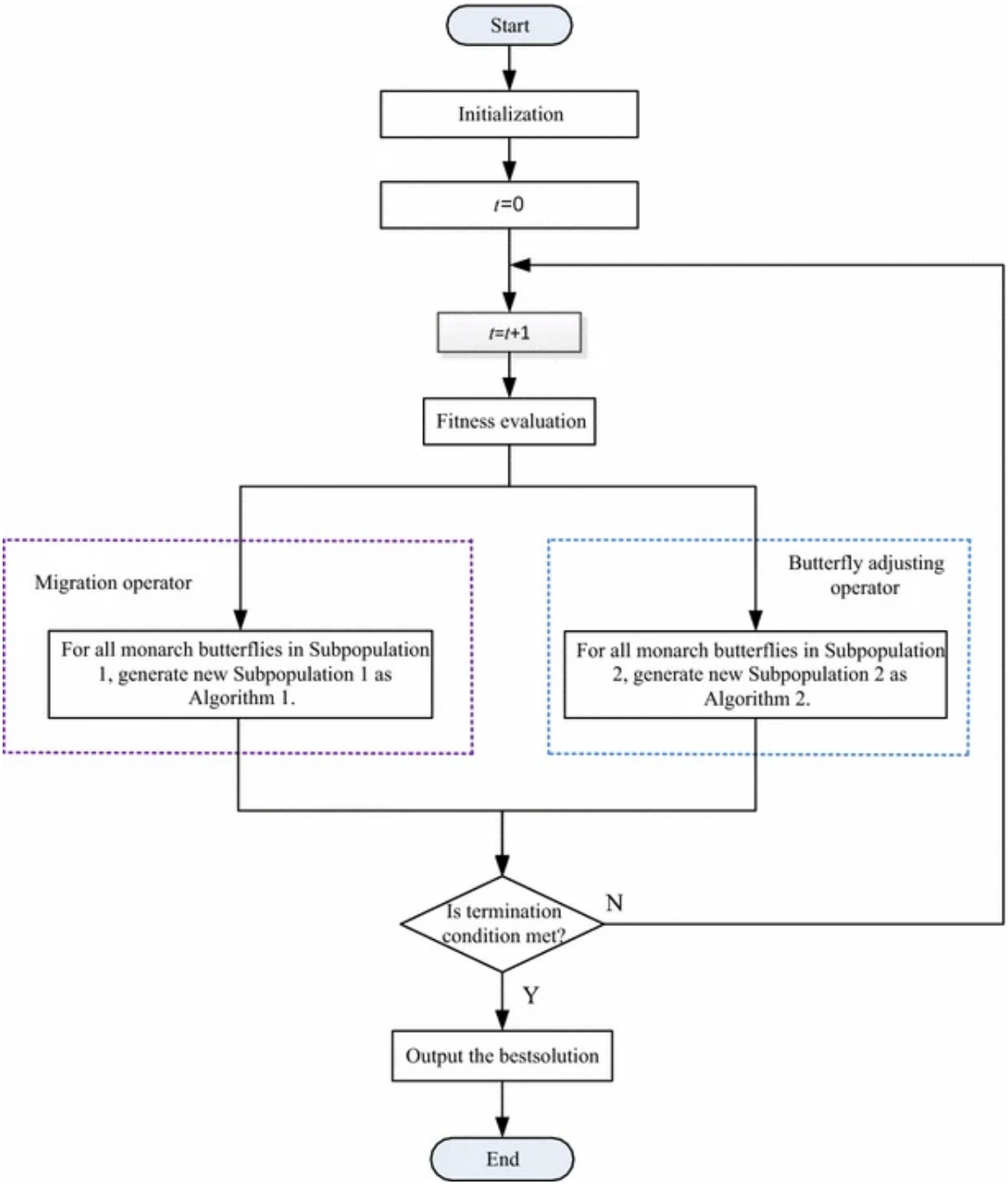


Figura 1: Esquema del algortimo MBO

# 4. Adaptación para APC

## 4.1. Implementación

El algoritmo de optimización ha sido adaptado al problema del aprendizaje de pesos en características (APC). En este tipo de problemas se conocen las clases existentes y se conoce la clase concreta de cada individuo/objeto del conjunto de datos. Existen una cantidad abrumadora de técnicas para abordar el aprendizaje de los sistemas de clasificación, pero en nuestro caso nos basaremos en ténicas de k vecinos más cercanos. Concretamente hemos usado un clasificador 1NN (kNN con $k = 1$) que nos servirá para calcular los aciertos de nuestro sistema de clasificación.

Para diseñar el clasificador son necesarias además dos técnicas, el aprendizaje y la validación. Para ello hemos utilizado la técnica de 5-fold cross validation, en la que se dividen los datos en 5 particiones disjuntas al 20 %. Utilizamos un 80 % de los datos (4 particiones de 5) para entrenar nuestro modelo, la partición restante se utiliza para testear los datos. La calidad final de nuestro modelo se mide con la media de los 5 porcentajes arrojados por las particiones.

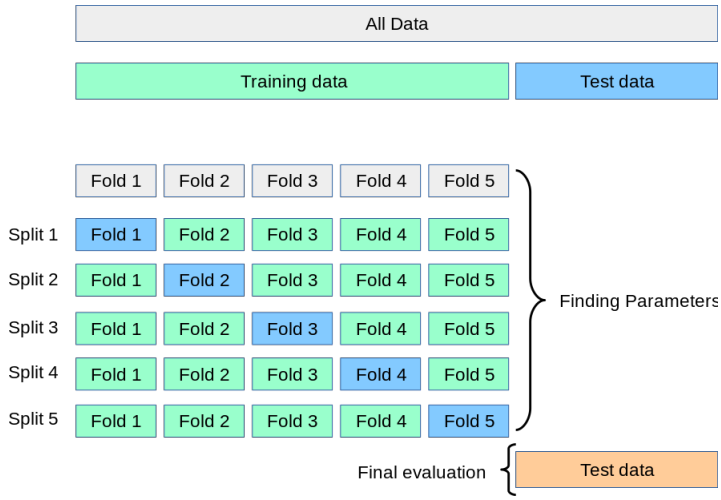


Figura 2: 5-fold cross validation ilustrado

APC optimiza el rendimiento del clasificador kNN. Este asigna valores reales a las características, de tal forma que se pondera la relevancia de cada una de ellas en el problema de aprendizaje.

La evaluación que hemos utilizado son los siguientes:

1. Tasa de reducción: asociada al número de características utilizadas por el clasificador con respecto al total.

2. Tasa de clasificación: porcentaje de instancias correctamente clasificadas.

En nuestro caso queremos maximizar ambas dándoles igual peso a ambos porcentajes (50 %). De modo que la función objetivo a maximizar es $F(W) = \alpha \cdot tasa_{clas}(W) + (1 - \alpha) \cdot tasa_{red}(W)$ siendo $\alpha = 0{,}5$.

## 4.2. Comparación con otros algoritmos

| | Ionosphere | | | | Parkinsons | | | | Spectf-heart | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *%_clas* | *%red* | *Agr.* | T | *%_clas* | *%red* | *Agr.* | T | *%_clas* | *%red* | *Agr.* | T |
| **1-NN** | 63,140 | 0,000 | 31,570 | 1,000 | 76,410 | 0,000 | 38,200 | 0,000 | 68,000 | 0,000 | 40,410 | 1,000 |
| **RELIEF** | 60,850 | 37,050 | 48,950 | 101,000 | 77,900 | 22,720 | 50,330 | 20,000 | 69,460 | 11,300 | 34,450 | 84,000 |
| **BL** | 61,400 | 8,230 | 34,830 | 15.774,000 | 76,410 | 0,900 | 38,650 | 2.398,000 | 70,330 | 0,400 | 35,390 | 20.940,000 |
| **ES** | 0,794 | 0,906 | 0,850 | 98449,000 | 0,754 | 0,991 | 0,872 | 6058,000 | 0,696 | 0,945 | 0,821 | 31335,000 |
| **ILS** | 0,834 | 0,912 | 0,873 | 999124,000 | 0,913 | 0,909 | 0,911 | 55809,000 | 0,779 | 0,918 | 0,849 | 316157,000 |
| **BMB** | 0,831 | 0,918 | 0,875 | 212358,000 | 0,923 | 0,909 | 0,916 | 87576,000 | 0,765 | 0,914 | 0,839 | 321453,000 |
| **ILS-ES** | 0,840 | 0,924 | 0,882 | 974105,000 | 0,831 | 0,973 | 0,902 | 88043,000 | 0,452 | 1,000 | 0,726 | 89449,000 |
| **AGG-BLX** | 0,851 | 0,894 | 0,873 | 206409,000 | 0,918 | 0,900 | 0,909 | 47924,000 | 0,823 | 0,886 | 0,855 | 286540,000 |
| **AGG-CA** | 0,846 | 0,906 | 0,876 | 206270,000 | 0,918 | 0,900 | 0,909 | 48247,000 | 0,804 | 0,923 | 0,864 | 292600,000 |
| **AGE-BLX** | 0,854 | 0,859 | 0,857 | 205851,000 | 0,887 | 0,864 | 0,875 | 47423,000 | 0,775 | 0,923 | 0,806 | 266190,000 |
| **AGE-CA** | 0,837 | 0,900 | 0,869 | 205536,000 | 0,923 | 0,909 | 0,916 | 47500,000 | 0,794 | 0,900 | 0,847 | 259204,000 |
| **AM-(10,1.0)** | 0,869 | 0,888 | 0,878 | 205532,000 | 0,913 | 0,918 | 0,916 | 48488,000 | 0,797 | 0,800 | 0,799 | 268152,000 |
| **AM-(10,0.1)** | 0,849 | 0,912 | 0,880 | 207176,000 | 0,954 | 0,909 | 0,931 | 46904,000 | 0,748 | 0,918 | 0,833 | 260714,000 |
| **AM-(10,0.1mej)** | 0,886 | 0,900 | 0,893 | 207509,000 | 0,892 | 0,918 | 0,905 | 47054,000 | 0,760 | 0,895 | 0,827 | 259117,000 |
| **AGG-BLX** | 0,851 | 0,894 | 0,873 | 206409,000 | 0,918 | 0,900 | 0,909 | 47924,000 | 0,823 | 0,886 | 0,855 | 286540,000 |
| **AGG-CA** | 0,846 | 0,906 | 0,876 | 206270,000 | 0,918 | 0,900 | 0,909 | 48247,000 | 0,804 | 0,923 | 0,864 | 292600,000 |
| **AGE-BLX** | 0,854 | 0,859 | 0,857 | 205851,000 | 0,887 | 0,864 | 0,875 | 47423,000 | 0,775 | 0,923 | 0,806 | 266190,000 |

Figura 3: Tabla comparativa

**[NOTA]: En BL, 1-NN y Relief hay un pequeño error de formato, se muestran porcentajes sobre 100 en vez de sobre 1.**

La tabla comparativa muestra resultados en cada tasa de las anteriormente descritas, la función objetivo y el tiempo total de ejecución. Estos resultados se calculan para tres conjuntos de datos distintos. Los distintos conjuntos utilizados han sido los siguientes:

1. Parkinsons – datos que se utilizan para distinguir presencia o ausencia de la enfermedad Parkinson. Consta de 195 ejemplos, 23 atributos y 2 clases.

2. Spectf-heart – contiene atributos calculados a partir de imágenes médicas de tomografía computarizada. La tarea consiste en saber si la fisiología del corazón es correcta. Consta de 267 ejemplos, 45 atributos y 2 clases.

3. Ionosphere – datos de radar recogidos en Goose Bay. Su objetivo es el de clasificar electrones por "buenos" y "malos" dependiendo de si tienen algún tipo de estructura en la ionosfera. Consta de 352 ejemplos, 34 atributos y 2 clases.

Los algoritmos comparados son los utilizados en las prácticas de la asignatura *Metaheurísticas* de la rama de Computación y Sistemas Inteligentes en Ingeniería Informática (UGR).

Los resultados arrojados para estos conjuntos de datos por el algoritmo MBO que hemos implementado son los siguientes:

1. Spectf-Heart:

   *a*) Tasa clas media: 0.7315068493

   *b*) Tasa red media: 0.7318181634

   *c*) Fitness medio: 0.7316625064

   *d*) Tiempo ejecucion medio: 68775 ms

2. Parkinsons:

    *a*) Tasa clas media: 0.9435897436
    *b*) Tasa red media: 0.7909090757
    *c*) Fitness medio: 0.8672494097
    *d*) Tiempo ejecucion medio: 6046 ms

3. Ionosphere:

    *a*) Tasa clas media: 0.9028571429
    *b*) Tasa red media: 0.7176470518
    *c*) Fitness medio: 0.8102520973
    *d*) Tiempo ejecucion medio: 48211 ms

Por sí solo el algoritmo consigue resultados muy competentes con respecto a la función a maximizar, consiguiendo alcanzar de media más del 80 % de fitness medio en en dos de los conjuntos de datos y un 70 % en el otro. Podemos observar también que sus tiempos son muy razonables si lo comparamos con el resto de algoritmos, siendo muy inferiores a la mayoría (obviamente algoritmos menos complejos y de menos calidad como relief son más rápidos). Esto se debe a la rápida convergencia que tiene MBO y a la paralelización de sus dos operadores. Vemos pues que el algoritmo base obtiene resultados muy a la par de técnicas de calidad como los genéticos, enfriamiento simulado o las búsquedas reiteradas.

# 5. Hibridación con BL

Para mejorar la propuesta de MBO he utilizado un método de búsqueda local que mejorá de forma sustancial la explotación del algoritmo. La búsqueda local se ha implementado a la mejor solución obtenida por el algoritmo (de forma que el tiempo de ejecución se incrementa ligeramente) y los resultados obtenidos son hasta un 10 % mejores en algunos casos. Claramente vemos que la búsqueda local le aporta un extra al poder explotar aún más la calidad de la solución añadiendo una búsqueda en el vecindario de la mejor solución encontrada. Los resultados de mejora son los siguientes para MBO-BL:

1. Spectf-Heart:

    *a*) Tasa clas media: 0.7803057375
    *b*) Tasa red media: 0.8954545498
    *c*) Fitness medio: 0.8378801437
    *d*) Tiempo ejecucion medio: 85977 ms

2. Parkinsons:

    *a*) Tasa clas media: 0.9230769231
    *b*) Tasa red media: 0.9000000238
    *c*) Fitness medio: 0.9115384735
    *d*) Tiempo ejecucion medio: 9635 ms

3. Ionosphere:

    *a*) Tasa clas media: 0.8257142857
    *b*) Tasa red media: 0.8999999881
    *c*) Fitness medio: 0.8628571369
    *d*) Tiempo ejecucion medio: 77001 ms

# 6. Mejoras de diseño/comportamiento

Sabemos que el equilibrio entre intensificación y diversificación dentro de un algoritmo metaheurístico lleva a que este sea más efectivo. La intensifiación está directamente relacionada con la explotación, mientras que la diversificación de una metaheurística se relaciona con su grado de exploración.

Dentro de nuestro algoritmo de optimización de la mariposa monarca, la diversificación tiene parte en el proceso del operador migratorio, el cuál actua para diversificar la subpoblación número 1 creando mariposas a partir de las migraciones de las mariposas padre, las cuales son elegidas aleatoriamente en función del ratio de migración. En el operador de ajuste en cambio se lleva un proceso de intensifiación por el cuál vamos generando mariposas hijas a partir de las características de la mejor mariposa de ambas tierras (siempre que ocurra que el número aleatorio esté dentro del ratio especificado). Además se implementa el vuelo Levy, el cual se encarga de aplicar una caminata aleatoria de distribución Levy en la cual se intensifica el proceso de búsqueda local.

En concreto, el algoritmo MBO está determinado por estos dos operadores, los cuales pueden ser implementados simultáneamente, lo cual hace que haya una buena compensación entre ambos operadores, dando lugar a un equilibrio entre diversificación e intensificación.

Igualmente podríamos aplicar mejoras en el equilibrio introduciendo componentes tales como CHC. Este algoritmo es una propuesta de Algoritmo Generacional que convina una selección estratégica (siempre preserva los mejores individuos) y aplicando un cruce uniforme (HUX) el cual intrcambia exactamente la mitad de los alelos (una de dos o más versiones de una secuencia de ADN) que son distintos en los padres.

`Algoritmo CHC Pseudocódigo:`

1: t = 0
2: inicializar $P(t)$
3: evaluar estructuras en $P(t)$
4: **while** not end **do**
5: $t = t + 1$
6: seleccionar $C(t) = P(t-1)$
7: recombinar $C^{'}(t)$="prevencción de incesto"+$HUX(C^{'}(t))$
8: reemplazar $P(t)$ de $C^{''}(t)$ y de $P(t-1)$
9: **if** convergencia($P(t)$) **then**
10: divergencia(P(t))

El operador de cruce tiene la característica de que genera hijos muy diferentes a los padres, de manera que genera una solución más diversa.