Instituto Nacional Electoral

# Modelo Matemático y Algoritmos

# Índice general

**Contenido** I

**Lista de Tablas** II

**1. Modelo Matemático** **1**
1.1. Función Multiobjetivo . . . . . 1
1.1.1. Objetivo Poblacional . . . . . 2
1.1.2. Objetivo Compacidad Geométrica . . . . . 3
1.1.3. Función Objetivo del Modelo Matemático . . . . . 5
1.2. Restricciones del Modelo . . . . . 5

**2. Recocido Simulado.** **7**
2.1. Algoritmo de Recocido Simulado . . . . . 8
2.2. Aplicación a distritos electorales . . . . . 11
2.2.1. Solución inicial . . . . . 11
2.2.2. Solución vecina . . . . . 12

**3. Colonia de Abejas Artificiales** **13**
3.1. Algoritmo Colonia de Abejas Artificiales . . . . . 14
3.2. Aplicación a Distritos Electorales . . . . . 15

**4. ABC-RS** **18**
4.1. Aplicación a Distritos Electorales . . . . . 19

**Bibliografía** **22**

# Índice de tablas

1.1. Valores de $k$ calculados con la expresión 1.4 tomando polígonos regulares de $m$ lados. . . . . 4

# Capítulo 1

# Modelo Matemático

Un Modelo de Optimización Multiobjetivo tiene dos componentes:

- Una Función Objetivo.
- Un Conjunto de Restricciones.

La tarea de la función objetivo, dentro del modelo de optimización, es calificar la calidad de las soluciones (o escenarios) que cumplen con todas las restricciones (soluciones factibles). Esta calificación se basa en dos criterios; el equilibrio poblacional en cada uno de los distritos y la forma de los distritos de acuerdo a su compacidad geométrica. La solución del modelo es cuando se obtiene aquel escenario cuyo valor objetivo es el menor posible. Este escenario puede ser único o puede que exista más de uno.

Otra tarea muy importante de la función objetivo es la de guiar a los algoritmos de búsqueda a ir mejorando las soluciones que se van obteniendo de tal manera que al final encuentren la mejor o una de las mejores soluciones al modelo.

## 1.1. Función Multiobjetivo

La función multiobjetivo debe conducir a los algoritmos a encontrar distritos con la menor desviación poblacional y que las formas geométricas de los distritos sean

lo más cercano a polígonos regulares pero además los valores de estos dos objetivos deben ser comparables para que sean "competitivos" dentro de esta función.

### 1.1.1. Objetivo Poblacional

La desviación del promedio poblacional de cada distrito electoral, debe estar a $\pm 15\,\%$, esto significa que si $P_D =$ Población del distrito y $P_M =$ Población media estatal, entonces la población de cualquier distrito está dentro de este criterio si cumple con la siguiente desigualdad:

$$P_M - 0.15P_M \leq P_D \leq P_M + 0.15P_M$$

Haciendo algunas operaciones algebraicas se llega a lo siguiente:

$$-0.15P_M \leq P_D - P_M \leq 0.15P_M$$

$$-1 \leq \frac{1}{0.15}\left(\frac{P_D}{P_M} - 1\right) \leq 1$$

o equivalentemente

$$0 \leq \left|\frac{1}{0.15}\left(\frac{P_D}{P_M} - 1\right)\right| \leq 1$$

De esta forma, para penalizar de forma más estricta a distritos fuera de rango, se toma un valor de $a$ mayor que 1, como se muestra en la siguiente expresión:

$$\left|\frac{1 - (\frac{P_D}{P_M})}{0.15}\right|^a \leq 1 \tag{1.1}$$

Se probaron algunos valores de $a \geq 1$ tales como: 1, 1.2, 1.5, 1.8 y 2 y después de varias experimentaciones se llegó a que el valor apropiado de $a$ es 2, ya que para valores menores de dos, en algunos ejercicios de prueba, se obtuvieron distritos fuera de rango.

En virtud de lo anterior, la fórmula de equilibrio poblacional, para un distrito $D$, es igual a:

$$C_1(D) = \left( \frac{1 - \left( \frac{P_D}{P_M} \right)}{0.15} \right)^2 \tag{1.2}$$

Donde:
$P_D$ = Población del distrito
$P_M$ = Población media estatal
0.15 = Desviación máxima permitida de la media estatal.

### 1.1.2. Objetivo Compacidad Geométrica

La compacidad geométrica de cada distrito electoral, se puede entender como la situación en la que el perímetro de los distritos adquiera una forma geométrica lo más cercana a **un polígono regular**.
Este criterio se incluye en la función multiobjetivo. La función que evalúa la compacidad se definirá como un cociente del perímetro del distrito entre la raíz cuadrada del área del mismo, multiplicada por un coeficiente definido experimentalmente para normalizarla con respecto al objetivo del equilibrio poblacional.

En la literatura sobre estudios de diseño de zonas, y en particular sobre distritación electoral, se proponen muchas medidas de compacidad tales como envolventes conexas, cuadriculado, comparación de círculos inscritos, etc.; pero una fórmula para el cálculo de la compacidad geométrica que da muy buenos resultados y su cálculo numérico requiere de pocas operaciones es:

$$k * \left( \frac{Perímetro\ del\ distrito}{\sqrt{Área\ del\ distrito}} \right) \tag{1.3}$$

Donde $k$ es una constante que hace la expresión 1.3 igual a uno dependiendo del polígono regular que se quiera. Por ejemplo, si deseamos que para un triángulo equilátero la expresión 1.3 sea igual a uno, $k = 0.21935$; para un cuadrado, $k = 0.25$, etc. En general para un polígono regular de $m$ lados la expresión 1.3 vale

uno, si el valor de $k$ es igual a:

$$k = \frac{1}{2\sqrt{m \tan\left(\frac{\pi}{m}\right)}} \tag{1.4}$$

En la Tabla 1.1 se calculan algunos valores de $k$ usando la expresión 1.4, tomando polígonos regulares con $m$ lados, para $m = 3, 4, ..., 10, ...\infty$.

| $\boldsymbol{m}$ | $\boldsymbol{k}$ |
|---|---|
| 3 | 0.21935 |
| 4 | 0.25000 |
| 5 | 0.26233 |
| 6 | 0.26864 |
| 7 | 0.27233 |
| 8 | 0.27467 |
| 9 | 0.27626 |
| 10 | 0.27738 |
| — | — |
| $\infty$ | 0.28209 |

Tabla 1.1: Valores de $k$ calculados con la expresión 1.4 tomando polígonos regulares de $m$ lados.

Por lo expresado anteriormente, para el cálculo de la compacidad geométrica de un distrito $D$ se propone la expresión

$$C_2(D) = \left(\left(\frac{\textit{Perímetro del distrito}}{\sqrt{\textit{Área del distrito}}} * 0.25\right) - 1\right) * 0.5 \tag{1.5}$$

Donde 0.25 corresponde al coeficiente que hace la expresión 1.3 igual a 1 si la forma del distrito corresponde a un cuadrado. Se eligió un cuadrado porque dentro de los polígonos regulares, los únicos capaces de formar una teselación [1] son: el triángulo equilátero, el cuadrado y el hexágono.

Por lo que se consideró restar a la expresión anterior la cantidad 1 para permitir que la mejor compacidad se encuentre cercana a cero y sea exactamente igual a cero cuando se forme un distrito de forma cuadrada.

[1] Un teselado es una regularidad o patrón de figuras que cubre o pavimenta completamente una superficie plana que cumple con dos requisitos: 1) Que no queden huecos, 2) Que no se superpongan las figuras.

### 1.1.3. Función Objetivo del Modelo Matemático

La función multiobjetivo que se propone para la distritación electoral será la suma ponderada del equilibrio poblacional y la compacidad geométrica dada por la siguiente expresión:

$$f(E) = \sum_{i=1}^{n} C_1(D_i) + 0.5 \sum_{i=1}^{n} C_2(D_i) \tag{1.6}$$

Donde, en la expresión 1.6, $E = (D_1, D_2, \ldots, D_n)$ es un escenario o plan distrital y el factor 0.5 refleja la importancia relativa de la compacidad geométrica con respecto al equilibrio poblacional. Se asigna al equilibrio poblacional un peso del doble con respecto a la compacidad geométrica, debido a la importancia relativa de ambos objetivos.

## 1.2. Restricciones del Modelo

Las restricciones que el modelo considera son las siguientes:

1. El estado se dividirá en $n$ distritos electorales.
2. Los distritos serán continuos y contiguos.
3. Cada distrito debe tener una desviación poblacional máxima dentro del 15 por ciento de la media estatal.
4. Integridad municipal, al considerar a los municipios que en forma integral se agruparán para formar un distrito.
5. Tiempos de traslado. Se calcula un tiempo de traslado de corte. Dos municipios se considerarán como no vecinos, si el tiempo de traslado entre ellos es mayor que el tiempo de corte.
6. Todo aquel municipio que pueda ser separado en un número entero de distritos dentro del 15 por ciento de desviación poblacional le serán asignados ese número de distritos y serán divididos hacia su interior.

7. Tipología de municipios.

# Capítulo 2

# Recocido Simulado.

Recocido simulado es una de las técnicas heurísticas más conocidas, que por su simplicidad y buenos resultados en numerosos problemas, se ha convertido en una herramienta muy popular, con aplicaciones en diferentes áreas de optimización. El concepto fue introducido en el campo de la optimización combinatoria a inicios de la década de los 80 por Kirkpatrick [8] y Cerny [4]. Esta heurística, se inspira en una analogía entre el proceso de recocido de sólidos y la forma en que se resuelven problemas de optimización combinatoria. Dicha analogía resulta importante para comprender la forma en que trabaja este algoritmo.

El recocido de sólidos es un proceso de tratamiento térmico que se aplica a varios materiales como el vidrio y ciertos metales y aleaciones para hacerlos menos quebradizos y más resistentes a la fractura. El objetivo de este proceso es minimizar la energía interna de la estructura atómica del material y eliminar posibles tensiones internas provocadas en las etapas anteriores de su procesado. Para lograrlo, los metales ferrosos y el vidrio se recuecen calentándolos a alta temperatura y enfriándolos lentamente. Cada vez que se baja la temperatura, las partículas se reacomodan en estados de más baja energía hasta que se obtiene un sólido con partículas acomodadas conforme a una estructura con energía mínima.

El algoritmo de recocido simulado está basado en técnicas de Monte Carlo y genera una sucesión de estados del sólido de la siguiente manera. Dado un estado actual $i$ del sólido con energía $S_i$, entonces el siguiente estado $j$ es generado al aplicar una pequeña perturbación al estado actual. La temperatura del nuevo estado es $S_j$. Si

la diferencia de energía $S_i - S_j$, es menor o igual que cero, el estado $j$ es aceptado como estado actual. Si la diferencia es mayor que cero, el estado $j$ es aceptado con una probabilidad dada por:

$$\exp\left(\frac{S_i - S_j}{k_B T}\right) \tag{2.1}$$

Donde $T$ denota la temperatura a la cual se encuentra el sólido y $k_B$ es una constante física llamada la constante de Boltzmann. Este criterio de aceptación es conocido como el criterio de Metrópolis. Si el decremento de la temperatura es suficientemente lento, el sólido alcanzará un equilibrio térmico en cada temperatura.

## 2.1. Algoritmo de Recocido Simulado

La técnica de recocido simulado, se inspiró en el hecho de que el algoritmo de Metrópolis puede ser utilizado para generar soluciones a problemas de optimización combinatoria si se hacen las siguientes consideraciones:

- Las soluciones del problema de optimización son equivalentes a los estados del sólido.
- El costo de una solución, denotado por $f(E_i)$, es equivalente a la energía de cada estado.
- La temperatura será sustituida por un parámetro de control que regulará la probabilidad de aceptación de nuevas soluciones.

El algoritmo de Recocido Simulado comienza con una solución inicial y una temperatura inicial $T_0$. Se recomienda que al inicio del algoritmo la temperatura sea suficientemente alta para permitir todo, o casi todo, movimiento, es decir, que la probabilidad de pasar de la solución actual $E_i$ a la solución vecina $E_j$ sea muy alta, sin importar la diferencia entre los costos de ambas soluciones, $f(E_i) - f(E_j)$.

En cada iteración se genera una solución vecina de manera aleatoria, si la nueva solución mejora el valor de la función objetivo con respecto a la solución actual, esta

última es reemplazada. Cuando la nueva solución no mejora el valor de la función objetivo, se puede aceptar el cambio de la solución actual con cierta probabilidad dada por:

$$\exp\left(\frac{f(E_i) - f(E_j)}{T}\right) \tag{2.2}$$

Donde, $f(E_i)$ es el costo de la solución actual, $f(E_j)$ es el costo de la solución vecina y $T$ es la temperatura del proceso. Conforme el algoritmo avanza, el valor de la temperatura disminuye mediante un coeficiente de enfriamiento, $0 < \alpha < 1$, pero cada valor de $T$ se mantiene constante durante $L$ iteraciones, para permitir que el algoritmo explore distintas soluciones con la misma probabilidad de aceptación.

Debe observarse que al inicio, cuando la temperatura es alta, se tiene una mayor probabilidad de aceptar soluciones de menor calidad, lo cual permite la exploración del espacio de soluciones y evita la convergencia prematura a mínimos locales. Sin embargo, conforme el valor de la temperatura disminuye, el algoritmo se hace más selectivo y difícilmente acepta soluciones de menor calidad, iniciando una búsqueda que lo guía hacia un mínimo local. Finalmente, el algoritmo se detiene cuando la temperatura alcanza un valor límite, $T_f$, y devuelve la mejor solución encontrada.

Los parámetros de temperatura inicial $T_0$, coeficiente de enfriamiento $\alpha$, temperatura final $T_f$ y número de soluciones visitadas en cada temperatura $L$, son conocidos como *programa de enfriamiento* y juegan un papel importante en el desarrollo del algoritmo. Si se utilizan valores demasiado grandes el tiempo de ejecución podría ser excesivo, mientras que valores demasiado pequeños podrían provocar una convergencia prematura a un mínimo local. Para mayores detalles sobre esta técnica ver [1].

Sea $T_k$ denote el valor de la temperatura y $L_k$ el número de transiciones generadas en la $k$-ésima iteración del algoritmo de Metropolis. El algoritmo de Recocido Simulado puede describirse en pseudo-código como se muestra en el Algoritmo 1.

El algoritmo de Recocido Simulado comienza llamando a un procedimiento de inicialización donde se definen la solución inicial, parámetro de control inicial y el número inicial de generaciones necesarias para alcanzar el equilibrio térmico para la temperatura inicial. La parte medular del algoritmo consta de dos ciclos. El externo **Repite**...**hasta** y el interno **Para**...**finpara**. El ciclo interno mantiene fija la

**Algoritmo 1:** Algoritmo de Recocido Simulado en pseudo-código

```
1  INICIALIZA (E_{i_inicial}, T_0, L_0)
2  k := 0
3  E_i := E_{i_inicial}
4  Repite
5  |  Para l := 1 a L_k hacer
6  |  |  GENERA (E_j vecino de E_i)
7  |  |  si f(E_j) ≤ f(E_i) entonces
8  |  |  |  E_i := E_j
9  |  |  fin
10 |  |  en otro caso
11 |  |  |  si exp((f(E_i)−f(E_j))/T_k) > número aleatorio en [0,1) entonces
12 |  |  |  |  E_i := E_j
13 |  |  |  fin
14 |  |  fin
15 |  fin
16 |  k := k + 1
17 |  CALCULA-LONGITUD (L_k)
18 |  CALCULA-CONTROL (T_k)
19 hasta Cumplir criterio de paro;
20 Termina algoritmo
```

temperatura hasta que se generan $L_k$ soluciones y se acepta o se rechaza la solución generada conforme el criterio de aceptación ya discutido. El ciclo externo disminuye el valor de la temperatura mediante el procedimiento CALCULA-CONTROL y calcula el número de soluciones a generar para alcanzar equilibrio térmico mediante el procedimiento CALCULA-LONGITUD. Este ciclo finaliza cuando la condición de paro se cumple.

Un rasgo característico del algoritmo de Recocido Simulado es que, además de aceptar mejoras en el costo, también acepta soluciones peores en costo. Inicialmente, para valores grandes de $T$, puede aceptar grandes soluciones deterioradas; cuando $T$ decrece, únicamente pequeñas desviaciones serán aceptadas y finalmente, cuando el valor de $T$ se aproxima a cero, no se aceptarán desviaciones. Este hecho significa que el algoritmo de Recocido Simulado tiene la capacidad de escapar de mínimos locales.

Note que la probabilidad de aceptar desviaciones está implementada al comparar el valor de $\exp(f(E_i) - f(E_j))/T$ con un número aleatorio generado de una distribución uniforme en el intervalo [0,1). Además, debe ser obvio que la velocidad de convergencia del algoritmo está determinada al escoger los parámetros $L_k$ y $T_k$, $k$ = 0,1,.... Si los valores $T_k$ decrecen rápidamente o los valores de $L_k$ no son grandes, se tendrá una convergencia más rápida que cuando los valores de $T_k$ decrecen lentamente o los valores de $L_k$ son grandes.

## 2.2. Aplicación a distritos electorales

La creación de distritos electorales, mediante recocido simulado, se realiza en dos etapas. Primero se crea una solución inicial formada por $r$ distritos conexos, donde $r$ es el número de distritos indicado para cada conjunto territorial. Posteriormente, se realiza un proceso de mejora destinado a explorar diferentes soluciones, a partir de la solución inicial, de tal forma que al final del proceso se obtenga una solución de buena calidad. Estas etapas se describen con más detalle en las siguientes secciones.

### 2.2.1. Solución inicial

El primer paso del algoritmo consiste en construir una solución inicial con distritos conexos, para lo cual selecciona de manera aleatoria $r$ Unidades Geográficas (UG) que asigna a distritos diferentes, y las marca como UG no disponibles. Después se realizan las siguientes instrucciones hasta que todas las UG están marcadas como no disponibles:

- Elegir un distrito.
- Generar una lista con las UG disponibles que colindan con el distrito seleccionado.
- Seleccionar al azar una UG de la lista.
- Incluir en el distrito la UG elegida y marcarla como no disponible.

De esta forma se obtiene una solución con $r$ distritos conexos ajenos que incluyen a todas las UG, cuya calidad no necesariamente es buena pero que podrá mejorarse en el proceso de búsqueda.

### 2.2.2. Solución vecina

El algoritmo de RS inicia con la construcción de una solución con distritos conexos, como ya se explicó en la sección 2.2.1, y durante el proceso de búsqueda y mejora, se garantizará que las nuevas soluciones conserven esta característica.

Para generar una solución vecina se elige de manera aleatoria un distrito, $D$, y se genera una lista con las UG que pueden ser enviadas a un distrito contiguo. Por lo tanto, en la lista se incluyen las UG que se encuentran en colindancia con otros distritos. Se selecciona aleatoriamente una UG, $i$, de la lista, y se cambia al distrito con el cual colinda, $D^{'}$; en caso de que colinde con dos o más distritos se hace una elección aleatoria. En caso de que el distrito elegido inicialmente esté formado por una sola UG se evita el cambio, ya que esto implicaría una disminución en el número de distritos.

Cuando el cambio de la UG provoca una disconexión en $D$ se recupera la conexidad de la siguiente forma:

1. Se identifican las diferentes componentes conexas en que se dividió $D$.
2. Se cuenta el número de UG que forman a cada componente conexa.
3. La componente conexa con el mayor número de UG es considerada como el distrito original, $D$.
4. El resto de las componentes conexas es enviado a $D^{'}$ junto con $i$.

Siguiendo este procedimiento, cada solución vecina es una solución con distritos conexos que se diferencia de la anterior por la ubicación de un conjunto de UG. Se debe mencionar que es importante elegir de manera aleatoria las UG que son cambiadas para evitar que el algoritmo favorezca algunas soluciones y para aumentar las posibilidades de visitar un mínimo global.

# Capítulo 3

# Colonia de Abejas Artificiales

*Colonia de Abejas Artificiales* (ABC por sus siglas en inglés) es una técnica metaheurística bio-inspirada, propuesta por Karaboga [5–7] que se basa en el comportamiento empleado por las abejas mielíferas para encontrar buenas fuentes de alimento y comunicar esta información al resto de la colmena. ABC utiliza parámetros como el tamaño de la colonia y el número máximo de generaciones. ABC es un algoritmo diseñado para resolver problemas de optimización combinatoria, que se basa en poblaciones donde las soluciones, llamadas fuentes de alimento, son modificadas por abejas artificiales. El objetivo de estas abejas es descubrir fuentes de alimento que tengan cada vez mayores cantidades de néctar, y finalmente devuelven la fuente de alimento más abundante que pudieron encontrar. De esta forma, se obtienen soluciones óptimas locales, e idealmente el óptimo global. En un sistema ABC, las abejas artificiales se mueven en un espacio de búsqueda multidimensional eligiendo fuentes de néctar dependiendo de su experiencia pasada y de sus compañeras de colmena. Cuando una abeja encuentran una mejor fuente de néctar, la memorizan y olvida la anterior. De este modo, ABC combina métodos de búsqueda local y búsqueda global, intentando equilibrar el balance entre exploración y explotación.

## 3.1. Algoritmo Colonia de Abejas Artificiales

El algoritmo ABC emplea varias soluciones del problema al mismo tiempo, cada solución es considerada como una fuente de alimento. La calidad de las soluciones es evaluada mediante la función objetivo, y puede ser vista como la cantidad de néctar en la fuente de alimento. A las funciones destinadas a producir modificaciones en las soluciones, se les llamada *abejas artificiales* y por el tipo de función, se clasifican en tres grupos: *abejas empleadas*, *abejas observadoras* y *abejas exploradoras*. El trabajo coordinado de los tres tipos de abejas produce cambios en las soluciones, de tal forma que se pueden encontrar fuentes de alimento cada vez de mejor calidad.

Las abejas empleadas están asociadas con una fuente de alimento en particular, y son las que explotan el alimento, a la vez que llevan consigo la información de esta fuente particular de alimento a las observadoras. Las abejas observadoras son aquellas que están esperando en el área de danza de la colmena para que las abejas empleadas les compartan la información sobre las fuentes de alimento, y entonces tomen una decisión de elección de alguna fuente de alimento. Las abejas que salen del panal en busca de una fuente de alimento al azar son las llamadas exploradoras.

En el algoritmo de Colonia de Abejas Artificiales, el número de abejas empleadas y el número de abejas observadoras, es igual al número de fuentes de alimento que están alrededor del panal. Cuando una fuente de alimento se agota debe ser abandonada, y la abeja empleada que la explotaba se convierte en una abeja exploradora que busca de forma aleatoria una fuente nueva.

La posición de una fuente de alimento representa una solución factible del problema de optimización y el monto de néctar o de alimento de la fuente corresponde a la calidad de la solución asociada a dicha fuente. En el Algoritmo 2 se presenta el pseudocódigo de colonia de abejas artificiales.

En la siguiente sección se da una descripción de la forma en que opera esta técnica.

**Algoritmo 2:** Algoritmo de Colonia de Abejas Artificiales en pseudo-código

```
1  INICIALIZA. Generar de forma aleatoria M fuentes de alimento E_1, ..., E_M.
2  Repite
3      Para i := 1 a M hacer
4          Enviar una abeja empleada a la fuente de alimento E_i.
5          Modificar la fuente de alimento E_i y determinar la cantidad de néctar de
           la nueva solución, E'_i, mediante la función objetivo.
6          si la nueva solución es mejor entonces
7              E'_i ⟵ E_i
8          fin
9      fin
10     Para i := 1 a M hacer
11         Calcular la probabilidad de cada fuente de alimento para ser elegida por
           una abeja observadora.
12     fin
13     Para i := 1 a M hacer
14         Elegir en probabilidad una fuente de alimento E_j.
15         Enviar una abeja observadora a la fuente de alimento seleccionada.
16         Modificar la fuente de alimento E_j y determinar la cantidad de néctar de
           la nueva solución, E'_j, mediante la función objetivo.
17         si la nueva solución es mejor entonces
18             E'_j ⟵ E_j
19         fin
20     fin
21     Detener la exploración de las fuentes de alimento que han sido agotadas por
       las abejas.
22     Enviar a las abejas exploradoras para encontrar nuevas fuentes de alimento
       de forma aleatoria.
23     Memorizar la mejor fuente de alimento encontrada hasta el momento.
24 hasta Cumplir criterio de paro;
25 Termina el algoritmo
```

## 3.2. Aplicación a Distritos Electorales

Para el diseño de distritos electorales, una fuente de alimento se define como una solución $E = \{D_1, D_2, ..., D_n\}$, donde $D_s$ es un conjunto de UG para $s = 1, 2, \ldots, n$. Una población inicial de $M$ fuentes de alimento se genera utilizando la estrategia descrita en la sección 2.2.1, de tal manera que cada solución está formada por $n$ distritos conexos.

De acuerdo a la técnica propuesta por Karaboga, cada fuente de alimento, $E_i$, debe

ser modificada por exactamente una abeja empleada. En este caso, cada solución sufre un cambio para obtener la solución $E_i^{'}$ siguiendo la estrategia descrita en la sección 2.2.2. Si la nueva solución $E_i^{'}$ tiene una cantidad de néctar mejor que $E_i$, $E_i^{'}$ sustituye a $E_i$ y se convierte en una nueva fuente de alimento explotada por la colmena. En otro caso, $E_i^{'}$ se rechaza y $E_i$ se conserva.

En cuanto el proceso de las abejas empleadas se ha realizado en todas las $M$ fuentes de alimento, empieza el turno de las abejas observadoras. Cada abeja observadora evalúa la calidad de las soluciones obtenidas hasta este momento, y elije una fuente de alimento con base en la probabilidad $p_i$, dada por:

$$p_i = \frac{\text{Calidad}_i}{\sum_{j=1}^{M} \text{Calidad}_j} \tag{3.1}$$

Donde:
$\text{Calidad}_i$ es la calidad de la fuente de alimento $E_i$, calculada mediante la función objetivo.
$M$ es el número de fuentes de alimento.

Una vez que ha seleccionado una solución, $E_{Origen}$, la abeja observadora intentará mejorarla mediante un proceso que combina algunas características de esta fuente de alimento con otra solución, $E_{Destino}$, elegida de forma aleatoria. Este proceso de combinación se resume en los siguientes pasos.

Sea $E_{Origen}$ la fuente de alimento que se va a modificar y $E_{Destino}$ la solución con la que deberá combinarse. Se elige una UG, $k$, de forma aleatoria. Entonces, existen un distrito $D_i \in E_{Origen}$ y un distrito $D_j \in E_{Destino}$ tales que $k \in D_i \cap D_j$ . Ahora se deben considerar los siguientes conjuntos:

$$H_1 = \{l : x_{li} = 0, x_{lj} = 1\} \tag{3.2}$$

$$H_2 = \{l : x_{li} = 1, x_{lj} = 0\} \tag{3.3}$$

Donde:

$$x_{li} = \begin{cases} 1, & \text{si la UG } l \text{ pertenece al distrito } i \\ 0, & \text{en otro caso} \end{cases}$$

Entonces una UG en $H_1$ es insertada en $D_i$, y una UG en $H_2$ es extraída de $D_i$, e insertada en un distrito contiguo a $D_i$.

Es importante observar que estos movimientos pueden producir disconexión en el distrito $D_i$, por lo que debe realizarse un proceso que la repare. Para esto, se cuenta el número de componentes conexas en $D_i$. Si el número de componente conexas es 1, entonces el distrito no perdió su conexidad. En otro caso, la componente conexa que contiene a $k$ (i.e., la UG usada en el proceso de combinación mencionado anteriormente) se define como el distrito $D_i$, mientras que el resto de las componentes son asignadas a otros distritos adyacentes.

Una vez que han concluido los procesos de combinación y reparación antes mencionados, se obtiene una nueva fuente da alimento $E'_{Origen}$. Si la nueva solución $E'_{Origen}$ tiene una cantidad de néctar mejor que $E_{Origen}$, $E'_{Origen}$ sustituye a $E_{Origen}$ y se convierte en una nueva fuente de alimento explotada por la colmena. En otro caso, $E'_{Origen}$ es rechazada y $E_{Origen}$ es conservada.

Se considera una iteración del algoritmo después de que las $M$ abejas empleadas y las $M$ abejas observadoras hacen su labor. El algoritmo finaliza después de $L$ iteraciones.

En cada iteración se lleva una estadística del número de veces que cada fuente de alimento sufre un cambio. Si una fuente de alimento no cambia después de un número $N$ de iteraciones, entonces una abeja exploradora sustituye esta fuente de alimento usando la estrategia descrita en la sección 2.2.1.

# Capítulo 4

# ABC-RS

La idea de combinar varios conceptos y técnicas de diferentes algoritmos es una de las líneas de investigación más fuertemente desarrolladas en el campo de la optimización desde los años noventa [2]. El objetivo de una hibridación es crear técnicas mejores y más robustas, que combinen y exploten las ventajas de diferentes métodos.

En la literatura especializada, no existe un consenso sobre el concepto **heurística híbrida** [2, 3, 10]. Sin embargo, en términos generales, una técnica heurística híbrida puede definirse como una estrategia robusta que combina de forma armónica las características de diferentes heurísticas. El objetivo principal de generar un híbrido es explotar de forma sinérgica los beneficios de diferentes técnicas de optimización, mientras que sus aspectos negativos se disminuyen, de tal forma que permite producir un algoritmo capaz de resolver problemas difíciles de forma eficiente. Para lograrlo, se requiere la combinación adecuada de ideas, paradigmas o conceptos complementarios.

La creación de una estrategia híbrida es un proceso que implica conocimiento y creatividad. Se requieren conocimientos sobre el problema de interés, métodos heurísticos y técnicas de optimización, de tal forma que se puedan seleccionar los elementos que serán utilizados en el híbrido. Por otra parte, se requiere la creatividad para combinar adecuadamente estas ideas. Por lo anterior, se puede decir que la hibridación no es un proceso trivial, que se basa en la experiencia, la creatividad y la inteligencia para la creación de un algoritmo capaz de resolver de

forma eficiente problemas de optimización difíciles [9]. Por lo tanto, una combinación adecuada de ideas complementarias, paradigmas o conceptos de optimización puede ser la clave para lograr el máximo rendimiento en la resolución de muchos problemas.

Existen diferentes taxonomías, o clasificaciones, de las técnicas de híbridación. Por ejemplo, Talbi propuso en [11] una clasificación basada en las funciones y en la arquitectura del algoritmo híbrido. De acuerdo a esta clasificación, se pueden considerar algoritmos híbridos de bajo y de alto nivel. Un híbrido es de bajo nivel cuando una de las estrategias de optimización de una técnica heurística es reemplazada por otra heurística. Por otro lado, cuando varias heurísticas trabajan dentro del mismo algoritmo se trata de un híbrido de alto nivel. Los híbridos de alto nivel se subdividen en dos categorías: por relevos y cooperativos. Se dice que es por relevos cuando la salida de una técnica es la entrada de otra, y se le llama cooperativos cuando varias estrategias trabajan por separado y periódicamente comparten información. Bajo esta taxonomía, el algoritmo ABC-RS puede clasificarse como un híbrido cooperativo de alto nivel.

En la siguiente sección se da una descripción de la forma en que opera el algoritmo ABC-RS.

## 4.1. Aplicación a Distritos Electorales

Por ser un híbrido entre recocido simulado y colonia de abejas artificiales, este algoritmo hereda muchos de los conceptos mencionados en las secciones 2 y 3, por ejemplo, será necesario establecer la temperatura inicial, $T_0$, la temperatura final, $T_f$, el procedimiento CALCULA-LONGITUD para determinar el número de iteraciones que se realizará en cada temperatura y el procedimiento CALCULA-CONTROL empleado para disminuir la temperatura. Asimismo, se define una fuente de alimento como una solución $E = \{D_1, D_2, ..., D_n\}$, donde $D_s$ es un conjunto de UG para $s = 1, 2, \ldots, n$.

Al iniciar el algoritmo, se genera una población de $M$ fuentes de alimento utilizando la estrategia descrita en la sección 2.2.1, de tal manera que cada solución está formada por $n$ distritos conexos.

Posteriormente, cada fuente de alimento, $E_i$, entra a un proceso de búsqueda y mejora de acuerdo a la siguiente estrategia.

Primero se emplea RS. Se genera una solución vecina, $E_j$, como se describe en la sección 2.2.2. Si $E_j$ mejora el valor de la función objetivo con respecto a $E_i$, entonces esta última es reemplazada por $E_j$. En caso contrario, la $E_j$ puede reemplazar a $E_i$ con una probabilidad dada por la ecuación 2.2. Posteriormente, se utiliza ABC. Se elige de forma aleatoria una solución $E_{Destino}$, la cual es combinada con la solución $E_i$ de acuerdo al proceso descrito en la sección 3.2 para obtener una nueva solución, $E_i^{'}$. La nueva solución $E_i^{'}$ reemplaza a $E_i$ cuando tiene un costo menor de la función objetivo, y es rechazada en caso contrario.

Este proceso de creación y reemplazo de soluciones se repite durante $L_k$ iteraciones, dadas por el procedimiento CALCULA-LONGITUD, al término de las cuales se pasa a la siguiente fuente de alimento, $E_{i+1}$, en la cual se aplican los mismos pasos. Cuando todas las fuentes de alimento han sido afectadas por este ciclo, se realiza un decremento en la temperatura mediante el procedimiento CALCULA-CONTROL, y se ejecuta nuevamente el ciclo. Este proceso termina cuando se alcanza la temperatura final propuesta por el usuario. En el Algoritmo 3 se presenta el pseudocódigo del algoritmo ABC-RS.

**Algoritmo 3:** Algoritmo ABC-RS en pseudo-código

```
1  INICIALIZA (E_{i_inicial}, T_0, L_0)
2  k := 0
3  E_i := E_{i_inicial}
4  Generar de forma aleatoria M fuentes de alimento E_1, ..., E_M.
5  Repite
6      Para i := 1 a M hacer
7          Para l := 1 a L_k hacer
8              Inicia RS
9              GENERA (E_j vecino de E_i)
10             si f(E_j) ≤ f(E_i) entonces
11                 E_i ⟵ E_j
12             fin
13             en otro caso
14                 si exp((f(E_i) − f(E_j)) / T_k) > número aleatorio en [0,1) entonces
15                     E_i ⟵ E_j
16                 fin
17             fin
18             Inicia ABC
19             Seleccionar de forma aleatoria una fuente de alimento E_Destino.
20             Combinar las fuentes de alimento E_i y E_Destino para obtener una
               fuente de alimento nueva E'_i.
21             Determinar el costo de la solución E'_i mediante la función objetivo.
22             si la nueva solución es mejor entonces
23                 E_i ⟵ E'_i
24             fin
25         fin
26     fin
27     k := k + 1
28     CALCULA-LONGITUD (L_k)
29     CALCULA-CONTROL (T_k)
30 hasta Cumplir criterio de paro;
31 Termina algoritmo
```

# Bibliografía


[1] S. G. de los Cobos, J. Goddard, M. A. Gutiérrez y A. E. Martínez, "Búsqueda y exploración estocástica", Ed. México: UAM-I (2010)

[2] C. Blum, J. Puchinger, G. R. Raidl y A. Roli, "A brief survey on hybrid metaheuristics", 4th International Conference on Bioinspired Optimization Methods and their Applications, pp 3-16 (2010)

[3] C. Blum, J. Puchinger, G. R. Raidl, A. Roli, "Hybrid metaheuristics in combinatorial optimization: A survey". Applied Soft Computing, 11(6), pp.4135-4151 (2011)

[4] V. Cerny, "A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm", Journal of Optimization Theory and Applications, 45, pp. 41-55 (1985)

[5] D. Karaboga, "An idea based on honey bee swarm for numerical optimization", Technical Report TR06, Computer Engineering Department, Erciyes University, Turkey (2005)

[6] D. Karaboga y B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm", Journal of Global Optimization, 39, pp. 459-471 (2007)

[7] D. Karaboga y B. Basturk, "On the performance of artificial bee colony (ABC) algorithm", Applied Soft Computing, 8, pp. 687-697 (2008)

[8] S. Kirkpatrick, C. D. Gellat y M. P. Vecchi, "Optimization by simulated annealing", Science, 220, pp. 671-680 (1983)

[9] J. T. Palma-Méndez y R. Marín-Morales, "Inteligencia artificial. Técnicas, métodos y aplicaciones". Mc Graw Hill (2008)

[10] R. G. Raidl, "A unified view on hybrid metaheuristics". Hybrid Metaheuristics, Springer Berlin Heidelberg (2006)

[11] E. G. Talbi, "A taxonomy of hybrid metaheuristics". Journal of heuristics, 8(5), 541-564 (2002)