Sistemas Adaptativos
Entrega Final

Alvaro Matamala
Miguel Villa
Diego Maldonado

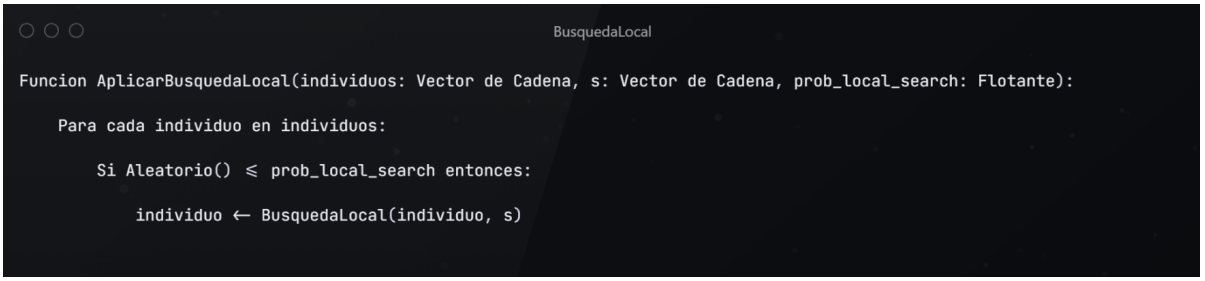
Pseudocódigo

Hibrido

```
hibrida(s: Vector de Cadena, tam_string: Entero, t_limite: Entero Largo, tunning: Entero, tam_pobl_inicial: Entero,
porcentaje_seleccionados: Flotante, prob_mutacion: Flotante, prob_cruce: Flotante, prob_local_search: Flotante) →
Tupla de Entero, Entero Largo:

    m ← tam_string
    tam_s ← Longitud de s

    best_dist ← -1
    best_time ← -1
    best_sol ← ""

    start_time ← Tiempo actual

    poblacion_inicial ← GenerarPoblacionInicialAleatoria(tam_pobl_inicial, m, s)  # Funcion auxiliar para generar
    población inicial aleatoria

    Mientras la duración actual no supere el límite de tiempo t_limite:

        # Torneo
        seleccionados ← SeleccionarPorTorneo(poblacion_inicial, porcentaje_seleccionados)

        # Cruza
        hijos ← CruzarIndividuos(seleccionados, poblacion_inicial, m, prob_cruce)

        # Mutación
        MutarIndividuos(hijos, m, prob_mutacion)

        # Busqueda local
        AplicarBusquedaLocal(hijos, s, prob_local_search)

        # Reemplazo
        ReemplazarPoblacion(poblacion_inicial, hijos, s)

        # Actualizar mejor solucion
        ActualizarMejorSolucion(poblacion_inicial, best_sol, best_dist, best_time, start_time, t_limite, tunning)

    Retornar Tupla (best_dist, best_time)
```

Cruza

```
Funcion CruzarIndividuos(seleccionados: Vector de Entero, poblacion: Vector de Cadena, m: Entero, prob_cruce:
Flotante) → Vector de Cadena:

    tam_seleccionados ← Longitud de seleccionados
    hijos ← Vector Vacío de Cadena

    Para i en rango(0, tam_seleccionados - 1, 2):

        padre1 ← poblacion[seleccionados[i]]
        padre2 ← poblacion[seleccionados[i + 1]]

        Si Aleatorio() < prob_cruce entonces:

            punto_cruza ← AleatorioEntre(0, m)
            hijo1 ← Concatenar(padre1[0:punto_cruza], padre2[punto_cruza:])
            hijo2 ← Concatenar(padre2[0:punto_cruza], padre1[punto_cruza:])

            Añadir hijo1 y hijo2 a hijos

        Sino:

            Añadir padre1 y padre2 a hijos

    Si tam_seleccionados es impar:

        Añadir poblacion[seleccionados[tam_seleccionados - 1]] a hijos

    Retornar hijos
```



Explicación

Variables de Inicio: Inicialización de variables, como tamaño de cadenas **tam_string**, límite de tiempo **t_limite**, tamaño de población inicial **tam_pobl_inicial**, y otros.

Generación de Población Inicial: Se crea una población inicial aleatoria de tamaño **tam_pobl_inicial**.

Iteración Principal (Bucle While): El algoritmo itera mientras no se exceda el límite de tiempo.

- **Selección por Torneo:** Selecciona individuos mediante torneo, comparando distancias y eligiendo ganadores hasta alcanzar un tamaño específico **tam_seleccionados**.

- **Cruzamiento:** Genera hijos a partir de los padres seleccionados. Con cierta probabilidad, se realiza un cruzamiento en un punto aleatorio; de lo contrario, se copian los padres.

- **Mutación:** Para cada hijo, con cierta probabilidad, se elige un índice aleatorio y se inserta un carácter aleatorio **ACGT**.

- **Búsqueda Local:** Para cada hijo, con cierta probabilidad, se inicia una búsqueda local que mejora iterativamente la solución haciendo cambios pequeños en la cadena de genes.

- **Reemplazo:** La población inicial se ordena por distancia de mayor a menor. Se obtiene la distancia de cada hijo y se reemplazan los individuos menos aptos de la población inicial con los nuevos hijos.

- **Actualizar Mejor Respuesta:** Se verifica si la mejor solución actual es superada por alguna de las soluciones generadas en la iteración actual. Si es así, se actualiza la mejor solución conocida (**best_sol**, **best_dist**, **best_time**).

Retorno de solución: Al terminar tiempo límite, el algoritmo sale del bucle while y retorna **best_dist** y **best_time**.