Sistemas Adaptativos
Metaheurística Poblacional para 2-Opt. Consensus Problem for DNA Motif Profiles

Álvaro Darío Matamala Gutiérrez
Diego Alexander Maldonado Muñoz
Miguel Ángel Villa Ríos

Pseudocódigo:

Genetico

```
Algoritmo genetico(s, tam_string, tam_poblacion, alpha, t_limite, tunning):
    n = tam_poblacion
    m = tam_string
    tam_s = longitud de s

    poblacion_inicial = []  // Lista para almacenar la población inicial
    distancias_poblacion_inicial = []  // Lista para almacenar las distancias de la población inicial

    start_time = obtener_tiempo_actual()  // Marcar el tiempo de inicio

    // Generar una población inicial aleatoria
    Para cada i en rango(n):
        cromosoma = generar_cromosoma_aleatorio(m)
        agregar cromosoma a poblacion_inicial
        agregar calcularDistancia(cromosoma, s) a distancias_poblacion_inicial

    mejor_solucion = ""
    mejor_distancia = -1
    mejor_tiempo = -1

    // Bucle principal hasta que se alcance el límite de tiempo
    Mientras el tiempo actual - start_time sea menor que t_limite:
        seleccionados = seleccionar_individuos_por_torneo(n, distancias_poblacion_inicial)

        // Cruzar los individuos seleccionados
        hijos = cruzar_individuos(seleccionados, poblacion_inicial)

        // Mutación de los hijos
        mutar_hijos(hijos, alpha)

        // Reemplazar individuos con mayor distancia en la población inicial por los hijos
        reemplazar_individuos(distancias_poblacion_inicial, poblacion_inicial, hijos)

        // Buscar el mejor individuo en la población actual
        mejor_indice = encontrar_mejor_individuo(distancias_poblacion_inicial)

        end_time = obtener_tiempo_actual()  // Marcar el tiempo de finalización
        duration = end_time - start_time

        // Actualizar la mejor solución si es necesario
        Si mejor_solucion está vacía o distancias_poblacion_inicial[mejor_indice] < mejor_distancia:
            mejor_solucion = poblacion_inicial[mejor_indice]
            mejor_distancia = distancias_poblacion_inicial[mejor_indice]
            mejor_tiempo = duration

            Si no tunning:
                Imprimir "Solucion: ", mejor_solucion
                Imprimir "Distancia: ", mejor_distancia
                Imprimir "Tiempo: ", duration

    Devolver tupla(mejor_distancia, mejor_tiempo)
```

Explicación del código:

Las partes importantes del algoritmo genético se pueden subdividir en 7 pasos, los cuales son:

1. Generación de la Población Inicial

Para cada individuo en la población, se genera un cromosoma aleatorio de longitud m utilizando las letras "A", "C", "G" y "T". La distancia de cada cromosoma respecto a las cadenas objetivo (s) se calcula y se almacena en distancias_poblacion_inicial.

2. Selección de progenitores por torneo

Se realiza la selección de individuos mediante el método de torneo. Se elige aleatoriamente un par de individuos y se selecciona el que tiene la menor distancia respecto a las cadenas objetivo. Este proceso se repite hasta tener la mitad de la población como individuos seleccionados.

3. Cruce de los ganadores

La etapa de cruce combina los cromosomas de los individuos seleccionados. Cada par de individuos seleccionados da lugar a dos descendientes obtenidos al intercambiar las mitades de sus cromosomas.

4. Mutación de los hijos

La mutación se aplica a cada hijo con una probabilidad determinada por el parámetro alpha. Si se cumple esta probabilidad, se elige aleatoriamente un índice en el cromosoma y se modifica la letra en ese índice. Esto para agregar variabilidad en la población.

5. Reemplazo de Individuos en la Población Inicial

Los descendientes generados reemplazan a los individuos de la población inicial que tienen las mayores distancias respecto a las cadenas objetivo. Esto contribuye a la evolución de la población hacia soluciones más óptimas.

6. Evaluación y Actualización de la Mejor Solución

Se busca el mejor individuo en la población actual y se actualiza la mejor solución conocida si es necesario. Volvemos al paso 2. También se imprime la información de la mejor solución si la opción de ajuste (tunning) está desactivada.

7. Finalización y Retorno de Resultados

El algoritmo termina cuando se alcanza el límite de tiempo especificado. Se marca el tiempo de finalización y se devuelve una tupla con la mejor distancia y el tiempo total de ejecución.