Metaheurísticas en Paralelo

Actividad

Archivos. *proyecto_benchmark_par.cpp* : Funciones de prueba de la metaheurística. *proyecto_par.cpp* : Archivo con el proyecto. Del mismo modo se incluyen los códigos secuenciales en archivos del mismo nombre pero sin la terminación *_par*.

Project_CP_MetaHeu.ipynb : Archivo Colab para graficar los resultados.

1. Para benchmark:
   - **Compilar:** g++ -O3 -fopenmp proyecto_benchmark_par.cpp -o bench_main
   - **Ejecutar:** bench_main.exe
2. Para el problema principal
   - **Compilar:** g++ -O3 -fopenmp proyecto_par.cpp -o bench_main
   - **Ejecutar:** main.exe

0.1. Introducción

Las **metaheurísticas** son algoritmos de optimización que proporcionan un marco general para encontrar soluciones satisfactorias en problemas complejos y en tiempos razonables. En contraste, los métodos exactos resultan impracticos debido a la alta dimensionalidad o al costo computacional del problema. Estas técnicas son especialmente útiles en problemas de optimización combinatoria, diseño de sistemas, planificación y otros ámbitos donde el espacio de búsqueda es vasto y difícil de explorar exhaustivamente.

Clasificación:

- Basadas en Trayectorias: Operan sobre una única solución que se va mejorando iterativamente. Por ejemplo: Recodo Simulado y Búsqueda Tabú.
- Basadas en poblaciones: Operan sobre una población de soluciones que evoluciona con el tiempo. Por ejemplo: Algoritmos Genéticos, Enjambre de Partículas, Colonias de Hormigas.

Características Clave:

- *Flexibilidad.* Puedes aplicarse a una amplia gama de problemas de optmización sin modificarlos significativamente.

- *Probabilísticos*. Utilizan métodos aleatorios para guiar la búsqueda de soluciones.
- *Balance entre Exploración y Explotación*. Permite buscar soluciones en nuevas regiones y mejorar soluciones prometedoras.
- *No requieren derivadas*. No requieren información sobre las derivadas de la función objetivo a diferencia de los métodos de optimización basados en gradiente.

Ventajas. Son útiles para encontrar soluciones cercanas al óptimo en problemas de alta dimensionalidad y alta complejidad. Además, tiene la flexibilidad de adaptarse a distintos tipos de problemas sin requerir un conocimiento profundo del dominio como en el caso de métodos determinísticos.

Desafíos. Selección de parámetros adecuados (tasas de cruce, mutación, etc.). Equilibrio entre exploración y explotación para evitar estancamiento en óptimos locales. Evaluación de la calidad de las conclusiones encontradas y la determinación de criterios de parada.

0.2. Algoritmo de Estimación de Distribución

Los **Algoritmos de Estimación de Distribución** (EDA) constituyen una familia de meta-heurísticas derivadas de los algoritmos evolutivos. Su principal característica es que la generación de nuevas poblaciones se realiza por medio de muestreo de una distribución de probabilidad en lugar de operadores de cruce y mutación.

A continuación se muestra el pseudocódigo general de un algoritmo de estimación de distribución:

```
1 D_0 ← Generar M individuos  (poblacion inicial) al azar
2 for l = 1,2,...,L o hasta que se verifique el criterio de parada
3     a) D_{l-1}^{Best} ← Selecionar N < M individuos de D_{l-1} acorde con la funcion
    objetivo
4     b) p_l(x) = p(x|D_{l-1}^{Best}) ← Estimar la distribucion de probabilidad de que un
    individuo se encuentre en los indiviuos seleccionados
5     c) D_l Muestrear M individuos (la nueva poblacion) de p_l(x)
```

Listing 1: Pseudocódigo de la aproximación EDA

Considere el pseudocódigo anterior. Partimos de M individuos generados por medio de una distribución uniforme para cada variable en su respectivo dominio. Estos M individuos constituyen a *la población inicial*, D_0. En un primer paso, cada uno de dichos individuos es evaluado según cierto criterio específico del problema, *la función objetivo*, y sólo los mejores $N(N < M)$ de ellos es seleccionado. En un segundo paso, se lleva a cabo la inducción del modelo

probabilístico n-dimensional que mejor refleja las interdependencias entre las n variables. En un tercer paso, M nuevos individuos (*la nueva población*) es obtienen por medio de la simulación de la distribución de la probabilidad aprendida en el paso anterior. Estos tres pasos son repetidos hasta que se logra una condición de paro.

La mayor dificultad de los EDAs es el problema de estimar la distribución de probabilidad $p_l(\mathbf{x})$. En este trabajo nosotros aproximaremos la distribución considerando una distribución normal para para individuo en D_{l-1}^{Best}, por lo que la aproximación para $p(\mathbf{x})$ será

$$p_l(\mathbf{x}) = \sum_{i=1}^{N} \mathcal{N}\left(\mathbf{x}|\mu = \mathbf{x}_i, \sigma = 0{,}1\right) \qquad \mathbf{x} \in D_{l-1}^{Best}. \tag{0.1}$$

0.3. Metodología

El problema que queremos resolver es el siguiente

$$\mathbf{x}^* = \underset{\mathbf{x}\in\Omega}{\operatorname{argmin}} f(\mathbf{x}), \qquad \text{con} \quad \Omega \subset \mathcal{R}^K. \tag{0.2}$$

Primero, implementaremos el algoritmo EDA con la aproximación (0.1). Para ello, primero presentamos las implementaciones hechas para el algoritmo secuencial y posteriormente las modificaciones para su ejecución en paralelo. Para verificar su funcionamiento lo probaremos con las benchmarks siguientes:

a) **Función Senoidal:**

$$f(x, y) = x^2 + y^2 + 3\sqrt{\sin^2(5x) + \sin^2(5y)} + 0{,}1 \tag{0.3}$$

con $\Omega = [-2, 2] \times [-2, 2]$. El mínimo se alcanza en $f(0, 0) = 0, 1$.

b) **Función Michaelwicz:**

$$f(x, y) = -\sin(x) * \sin^{20}\left(\frac{x^2}{\pi}\right) - \sin(y) * \sin^{20}\left(\frac{2y^2}{\pi}\right) \tag{0.4}$$

con $\Omega = [0, 4] \times [0, 4]$. El mínimo se alcanza en $f(2, 20319, 1, 57049) = -1{,}801$.

0.3.1. Implementaciones

Los **parámetros del algoritmo EDA** con los siguientes:

- N : *Tamaño de la población.* Inicialmente $D_0 = \{\mathbf{x}_i\}_{i=1}^N$ estará conformada de individuos generados uniformemente en Ω, posteriormente serán seleccionados como los mejores de

la siguiente generación.

- M : Número de descendientes. Esta será la cantidad de hijos de cada individuo $\mathbf{x} \in D_l$.
- *best_rate* : Proporción seleccionada de cada generación.
- L : Número de generaciones. Cantidad de veces que se obtendrá una nueva generación.
- $\mathcal{N}(\mu, \sigma)$: Distribución normal multivariable con vector de medias μ y de desviación estándar σ.

Las **variables del problema**:

- K : Número de parámetros, esto es, dimensión del dominio de búsqueda.
- Ω : Dominio del problema especificado por los límites superiores $Lsup_i$ e inferiores $Linf_i$ de cada variable, esto es, $\Omega = [Linf_1, Lsup_1] \times [Linf_2, Lsup_2] \times \cdots [Linf_k, Lsup_k]$.

A continuación se presenta la implementación del algoritmo EDA con distribución normal multivariable como generadora de nuevos individuos

```
// Poblacion Inicial: D_pop
for( int i = 0; i < N; i++ ) {
    getUniform( D_Pop + i*K, Linf, Lsup, K );
}

// Generamos una nueva elite
for( int gen = 0; gen < L; gen++ ) {
    // Para cada individuo x de D_pop
    for( int i = 0; i < N; i++ )
    {
        // Generamos M individuos con N(x, std)
        GenNormalSample( Desc_xi, M, D_Pop + i*K, Std_dev, Dom, K );

        // Evaluamos los nuevos individuos
        Evaluate( Desc_xi, Eval_xi, M, K );

        // Seleccionamos los D_best mejores
        BestSpecimen( Desc_xi, Eval_xi, BestDesc_xi, M, K, D_best );

        // Unimos todos los mejores
        JointBestDescend( BestDesc_xi, Eval_xi, BestDesc + i*D_best*K,
    Eval_BestDesc + i*D_best, D_best, K );

        // Evaluacion de D_pop
        Evaluate( D_Pop, Eval_D_pop, N, K );
```

```
25
26          // Unimos D_pop a la nueva generacion
27          JointBestDescend( D_Pop, Eval_D_pop, BestDesc + N*D_best*K,
    Eval_BestDesc + N*D_best, N, K );
28      }
29
30      // Seleccionamos los mejores de los nuevos y los viejos individuos
31      BestSpecimen( BestDesc, Eval_BestDesc, D_Pop, (D_best+1)*N, K, N );
32  }
```

Listing 2: Algoritmo EDA con distribucion normal multivariable como generadora de nuevos individuos.

Resultados

Para las funciones benchmarks consideramos los parámetros del método EDA mostrados en la tabla 1.

Función Senoidal y Michaelwicz	
N	4000
M	500
$best\_rate$	0.4
L	10
σ_i	$0{,}05 * (Lsup_i - Linf_i)$

Cuadro 1: Parámetros usados para el algoritmo EDA en la función senoidal.

Mostramos los mejores cinco individuos de la última generación para la función *Senoidal*, vea 2.

Generación $L = 10$	
1	[0.000055, -0.000024]
2	[-0.000364, 0.000023]
3	[-0.000461, 0.000138]
4	[-0.000160, -0.000473]
5	[0.000441, -0.000347]

Cuadro 2: Tabla de los mejores cinco puntos de la última generación para la función *Senoidal*.

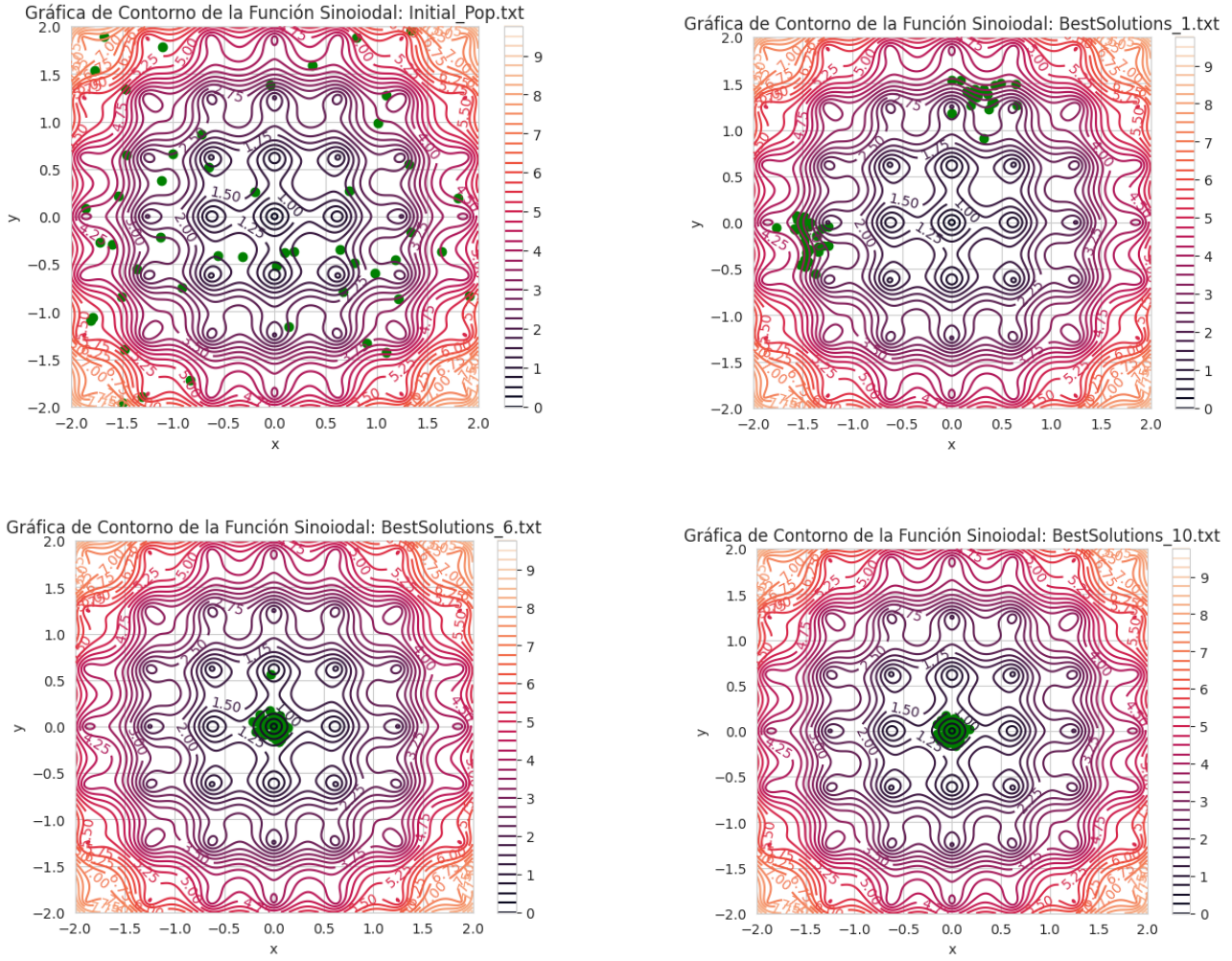

Figura 1: Gráficas de Contorno con los puntos obtenidos en las generaciones 0, 1, 6 y 10 para la función *Senoidal*. Solo se muestran los 50 mejores puntos cada generación, respectivamente.

Mostramos los mejores cinco individuos de la última generación para la función *Michaelwicz*, vea 3.

Generación $L = 10$	
1	[2.202677, 1.570917]
2	[2.202585, 1.570850]
3	[2.202910, 1.571106]
4	[2.203437, 1.570682]
5	[2.203357, 1.571192]

Cuadro 3: Tabla de los mejores cinco puntos de la última generación para la función *Michaelwicz*.

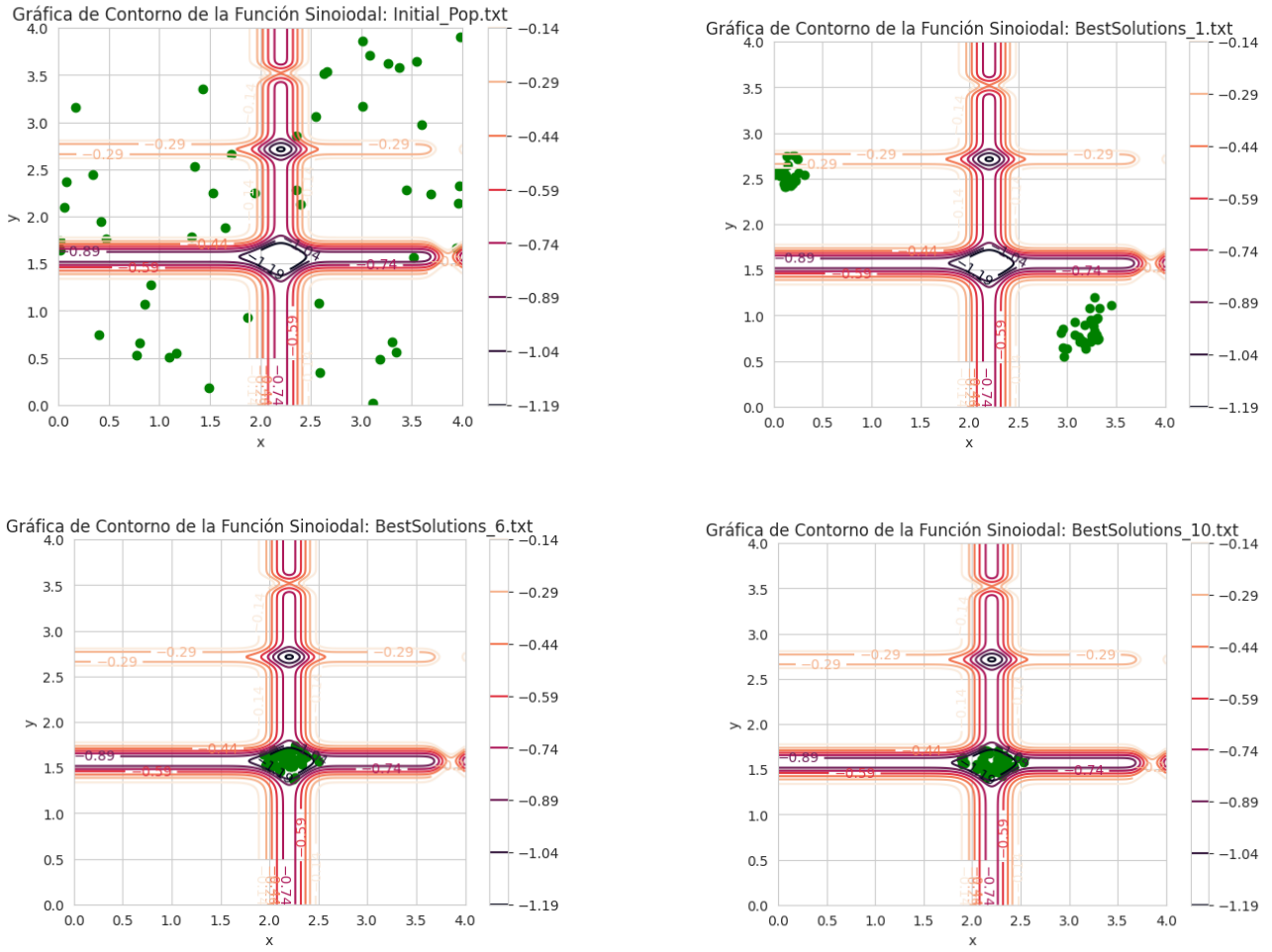

Figura 2: Gráficas de Contorno con los puntos obtenidos en las generaciones 0, 1, 6 y 10 para la función *Michaelwicz*. Solo se muestran los 50 mejores puntos cada generación, respectivamente.

0.3.2. Problema

Supongamos que tenemos la función

$$V(x) = E_0^* - b * log(i) - Re * i + C_1 \ln(1 - C_2 i). \tag{0.5}$$

Esta función describe el voltaje generado por una celda de combustible de metanol directo en función de la denominada densidad de corriente i.

De un experimento se tiene datos recolectados y se guardan en un documento llamado *datos_358.dat*. El documento contiene valores (i, V_i^{FC}), donde i es la densidad del corriente (A/cm^2), y V_i^{FC} es el voltaje a la densidad i.

Se debe estimar el valor de los 5 parámetros $\theta = \{E_0^*, b, Re, C_1, C_2\}$ definidos en los inter-

valos:

Parámetros	E_0^*	b	Re	C_1	C_2
Límite Inferior	0.0	0.0	0.0	0.0	0.0
Límite Superior	0.5	0.2	1.0	1.14	60.0

Cuadro 4: Tabla de límites para cada variable del problema.

Para hallar la estimación de estos parámetros utilizaremos las siguientes tres funciones

$$SSE(\theta) = \sum_{i=1}^{m} |V_i^{FC} - V(i)|^2 \tag{0.6}$$

$$SAE(\theta) = \sum_{i=1}^{m} |V_i^{FC} - V(i)| \tag{0.7}$$

$$MAE(\theta) = Median\{|V_i^{FC} - V(i)|\} \tag{0.8}$$

donde m es la cantidad de datos en el archivo *datos_*358.*dat*. Entonces el problema de minimización es minimizar alguna de las funciones objetivo.

Parámetros	E_0^*	b	Re	C_1	C_2	f_obj
Reported	0,432	0,123	0,041364	0,108	29,4	$\cdots$
SSE	0,3976	0,14452	0,01969	0,12847	28,993	**0,00279**
SAE	0,39393	0,14751	0,02277	0,12253	29,1531	**0,18965**
MAE	0,40408	0,13833	0,01331	0,11995	29,209	**0,01194**

Figura 3: Resultados proporcionados según cada función error.

La paralelización se ejecuto sobre la obtención de las poblaciones, para ello se utilizaron *n_threads* = *4*.

Resultados

De la misma manera que para las pruebas con las benchmarks. Consideramos los siguientes valores para los parámetros del algoritmo EDA para este problema, vea 5.

Se obtuvo un **speed_up** $\approx$ 2,18 entre el método paralelizado sobre el método secuencial.

También mostramos el mejor resultado de cada prueba hecha a partir de cada función error, vea 6.

Parámetros	
N	4000
M	500
$best\_rate$	0.3
L	20
σ_i	$0{,}01 * (Lsup_i - Linf_i)$
$n\_threads$	4

Cuadro 5: Parámetros usados para el algoritmo EDA en la función principal.

Parámetros	E_0^*	b	Re	C_1	C_2	**FuncObj**
SSE	0.450395	0.123283	0.582946	0.141850	28.563723	0.004975
SAE	0.498393	0.098637	0.479931	0.147505	28.568213	0.162498
MAE	0.492222	0.092540	0.587427	0.129544	27.999712	0.015978

Cuadro 6: Tabla de resultados para cada función objetivo.

A continuación, vea Fig.4 se muestran los diferentes ajustes al modelo 0.5 con los parámetros obtenidos bajo cada una de las funciones objetivo, vea Tabla 6.

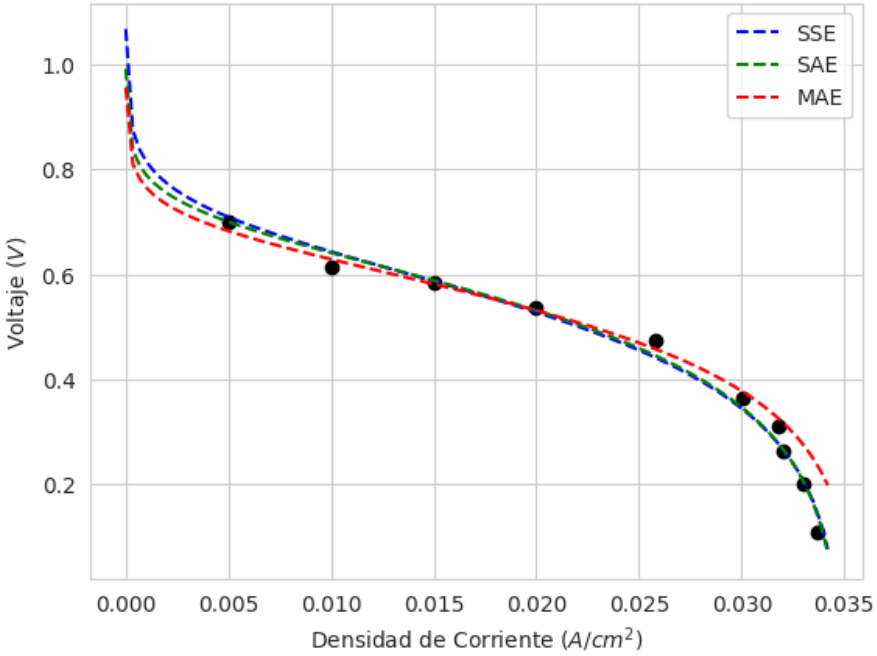

Figura 4: Gráfica de Ajustes con los parámetros obtenidos. Los puntos negros son los datos de *datos_358*.

Como observamos, los resultados obtenidos distan mucho de los resultados proporcionados. Primero, las variables Re y C_2 no son cercanas de ninguna forma. Además, no es posible graficar

una curva con los parámetros proporcionados debido a que el valor parámetro C_2 produce estar fuera del dominio de la función $\ln(\cdot)$.

Por otro lado, los resultados gráficos que obtuvimos son comparables con los proporcionados. La curva que tiene un ajuste más 'horizontal' es justamente la obtenida por la función de objetivo MAE, mientras que las dos restantes tienen pendientes ligeramente mayores. Lo anterior se explica por el error que mide cada una, donde los errores pequeños se hacen más pequeños y los grandes más grandes, por eso la función SSE es la que mayor peso le da a los extremos y en segundo lugar la función SAE. Por otro lado, la función MAE se ajusta por la mediana de los datos, lo que proporciona este efecto.

0.4. Conclusiones:

Los resultados numéricos obtenidos, desde el punto de vista de resolver el problema de optimización, fueron muy buenos porque nos acercamos a los proporcionados tanto visual como numéricamente.

El uso de metaheurísticas para encontrar soluciones a problemas de optimización fue muy gratificante debido a la facilidad para encontrar la solución sin una teoría compleja sobre el algoritmo ni tampoco un coste computacional grande debido a que para encontrar las mejores soluciones solo requiere de evaluar la función objetivo. De este modo, este tipo de métodos de optimización son muy útiles.

La naturaleza de metaheurísticas poblacionales tiene característica de la gran cantidad de cálculos independientes en una gran cantidad de individuos. Debido a lo anterior, es de gran utilidad contar con herramientas de paralelización para reducir el tiempo de computo y aprovechar los recursos computacionales disponibles.

0.5. Bibliografía:

1 Notas del Curso Computo Paralelo. Tercera Parte: Blanco-Cocom Luis Daniel. (2023) CIMAT.

2 Engineering Optimization: An Introduction with Metaheuristic Applications. Yang Xin-She. Wiley (2010).

3 Metaheuristics: Outlines, MATLAB Codes and Examples. Kaveh Ali & Bakhshpoori Taha. Springer (2019).