1. Autores

Camila Montserrat Alderete González
Luis Alberto Cañete Baez
Oscar Leonardo Pedrozo González

2. Resumen

El paper trata sobre las implementaciones y pruebas de rendimientos para resolver los problemas del QAP por medio de algoritmos evolutivos como NSGA (Nondominated Sorting Genetic Algorithm) así también con el algoritmo MOACS (MultiObjective Ant Colony System).

3. Introducción

En este paper se muestra una comparación entre dos algoritmos empleados en la optimización multi-objetivo, el NSGA, el cual usa enfoque evolutivo y el MOACS que está basado en el modelo de comportamiento de las colonias de hormigas, denominada metaheurística ACO (Ant Colony Optimization), sobre el cual se realizaron pruebas con dos instancias del conocido problema QAP (Quadratic assignment problem) caracterizado por ser un problema de optimización combinatoria y ser del tipo NP-completo. Además explicaremos los algoritmos multi-objetivos utilizados, los resultados experimentales de la comparación y finalmente las conclusiones.

4. Descripción del Problema

El problema de asignación cuadrática o QAP es un problema de la clase NP-completo con diferentes aplicaciones en áreas como la investigación de operaciones, computación paralela distribuída, análisis combinatorio, etc. El objetivo del QAP es asignar n instalaciones o facilidades en n ubicaciones de modo a minimizar el costo total, siendo el costo la suma del producto entre los flujos y las distancias entre las instalaciones.

$$Minimizar\ \vec{f}(\vec{x}) = \begin{cases} \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} b_{ij}^{1} \\ \vdots \\ \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} b_{ij}^{b} \end{cases}$$

Para este trabajo se consideró el bQAP o QAP bi-objetivo, donde cada instancia del problema ofrece como entrada los siguientes datos:

- Las n localidades.
- La matriz distancias entre las localidades $A=\{a_{ij}\} \in \mathbb{R}n\times n$ donde a_{ij} representa la distancia entre las localidades i y j
- b matrices de flujo $B^q=\{b_{ij}^q\} \in \mathbb{R}n\times n$, $q=1,\dots, b$ donde b_{ij}^q representa el q-ésimo flujo entre las instalaciones ubicadas en las localidades i y j.

5. Algoritmos Implementados

5.1 NSGA

Se basa en la clasificación de individuos en varias capas o frentes. La clasificación consiste en agrupar a todos los individuos no dominados en un frente, con un valor de fitness (o adaptabilidad) igual para todos los individuos. Este valor es proporcional al tamaño de la población, para así proporcionar un potencial reproductivo igual para todos los individuos de este frente. Entonces el grupo de individuos clasificados es ignorado y otro frente de individuos no dominados es considerado. El proceso continúa hasta que se clasifican a todos los individuos en la población. Puesto que los individuos en el primer frente tienen el valor de fitness mayor, consiguen siempre más copias que el resto de la población.

Pseudocódigo:

```
function nsga( ):
        inicializar_poblacion()
        gen = 0
        mientras (gen < max_gen):
                frente = 1
                mientras ( la población no está clasificada ):
                        identificar_individuos_no_dominados()
                        asignar_dummy_fitness()
                        sharing_fitnes_frente_actual()
                        frente = frente + 1

                operador_reproducción_elitismo()
                operador_crossover()
                operador_mutación()
                gen = gen + 1
```

5.2 MOACS

Propuesto por Barán y Schaerer es una extensión del MACS-VRPTW. Fue implementado considerando dos objetivos, utiliza una matriz de feromonas y dos visibilidades, una para cada objetivo del problema. La regla de transición de estados se calcula como:

$$j = \{\left\{\tau_{i,j}\left[\eta_{i,j}^{0}\right]^{\lambda,\beta}\left[\eta_{i,j}^{1}\right]^{(1-\lambda),\beta}\right\} \; si\; q < q_0 \; \hat{i}\; en\; caso\; contrario$$

El cálculo de $\hat{i}$ se realiza según la siguiente fórmula:

$$p_j = \{\frac{\tau_{i,j}\left[\eta_{i,j}^{0}\right]^{\lambda,\beta}\left[\eta_{i,j}^{1}\right]^{(1-\lambda),\beta}}{\sum_{x\in J_i}\tau_{i,x}\left[\eta_{i,x}^{0}\right]^{\lambda,\beta}\left[\eta_{i,x}^{1}\right]^{(1-\lambda),\beta}} \; si\; j\in J_i,\; 0\; en\; caso\; contrario$$

Cada vez que una hormiga se mueve del estado i al estado j, realiza la actualización local de feromonas según la ecuación:

$$\tau_{i,j} = (1 - \rho) \bullet \tau_{i,j} + \rho \cdot \Delta\tau$$

Pseudocódigo:

```
function moaco( ):
        inicializar_parametros()
        mientras (not condicion_parada()):
                generacion=generacion + 1
                        for ant=1 to m
                                construir_solucion()
                                evaluar_solucion()
                                actualizar_feromonas()
                                actualizar_conjunto_pareto()

function construir_solucion( ):
        sol={Ø}
        while existen_estados_no_visitados()
                siguiente=seleccionar_siguiente_estado()
                sol=sol U {siguiente}
                marcar_como_visitado(siguiente)
                if(actualizacion_paso_a_paso):
                        actualizar_feromonas_paso_a_paso()
```

6. Hardware utilizado

Los algoritmos fueron implementados en python 3.9 y fueron ejecutados en un entorno Windows 10, con un procesador AMD Ryzen 5 con 8GB de memoria. Se realizaron cinco ejecuciones del algoritmo NSGA para instancia.

7. Resultados Experimentales

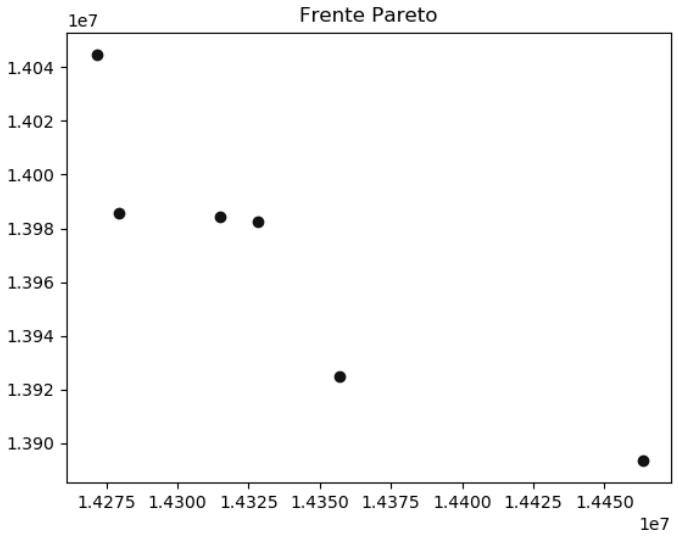

Figura 1. NSGA

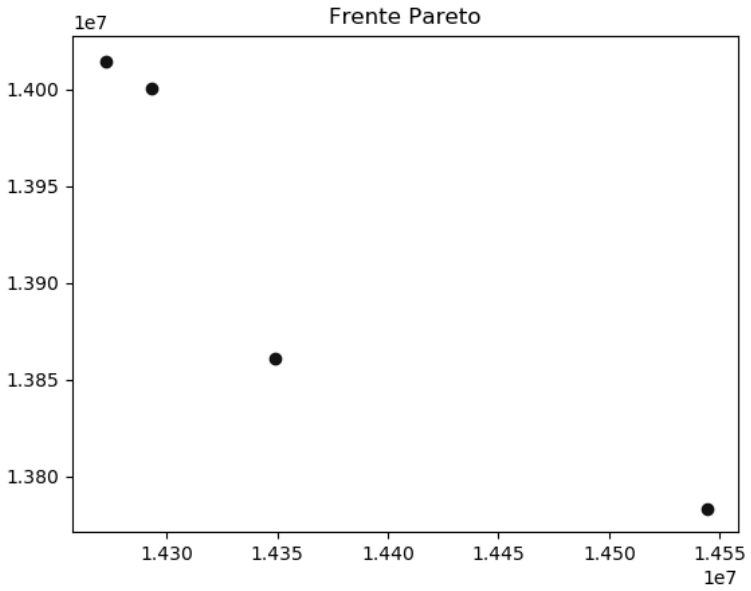

Figura 2. NSGA

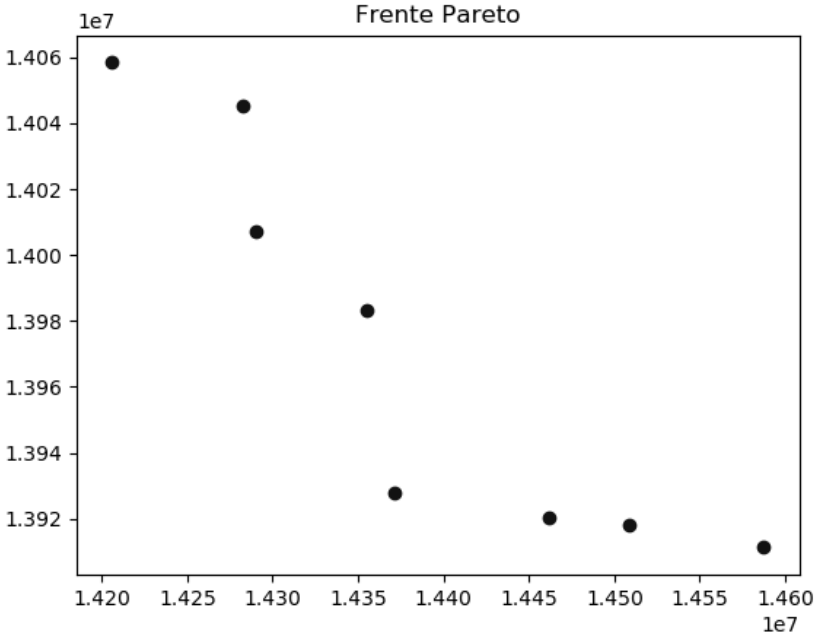

Figura 3. NSGA

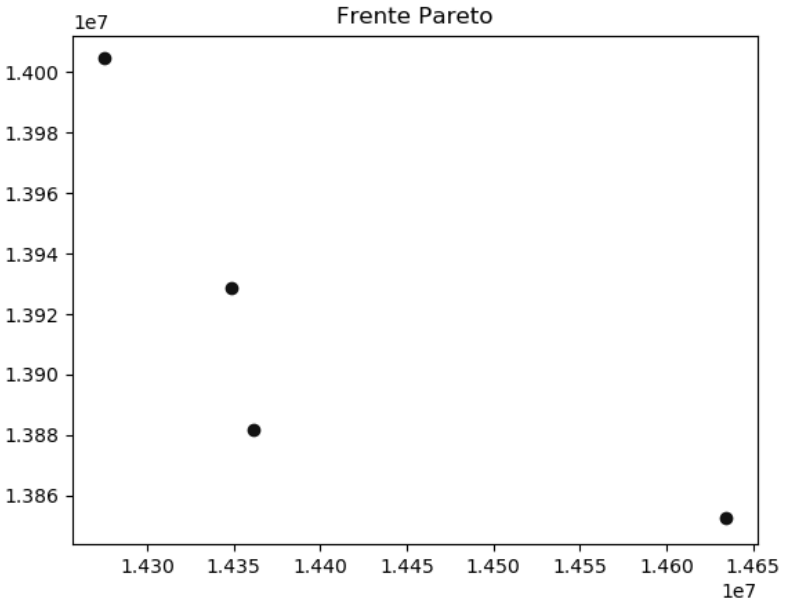

Figura 4. NSGA

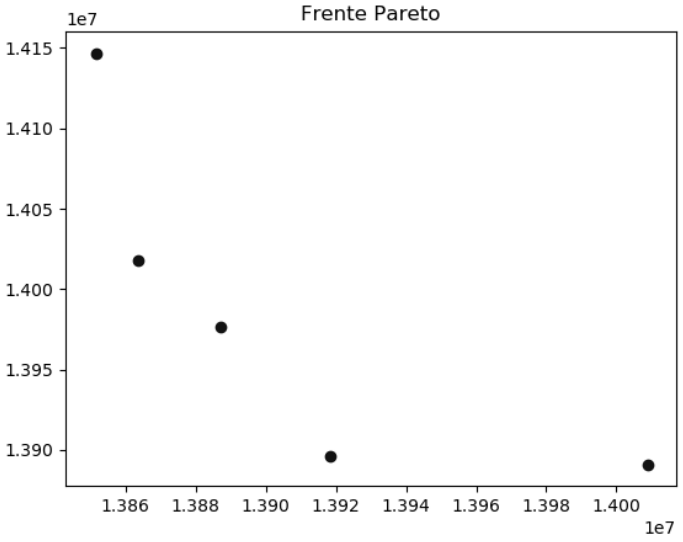

Figura 5. NSGA

Algoritmo NSGA			
Prueba	Distancia	Distribución	Extensión
1	0.0	6	244156.044136
2	0.0	4	356824.814799
3	0.0	8	409176.416378
4	0.0	4	390209.859619
5	0.0	5	300622.858805
Promedio	0.0	5,4	340197.9987474

8. Conclusión

No se pudo finalizar la implementación del algoritmo MOACS, pero se pudo verificar que el algoritmo de NSGA es adecuado para resolver los problemas QAP bi-objetivo. Buscando en la literatura los mejores frente pareto obtenidos por otras implementaciones NSGA y otros algoritmos, podemos concluir que nuestra implementación necesita ajustes para obtener paretos con mayor optimalidad. En base a la investigación y la implementación de los algoritmos se aprecia que debido a la complejidad de los problemas con soluciones multiobjetivos y su dificultad NP-completos no es fácil encontrar un algoritmo genérico para resolverlos todos en forma óptima. Algunos algoritmos tienen mejor desempeño en las métricas que otros, dependiendo del tipo del problema.

No existe un único algoritmo óptimo para resolver estos problemas, dependiendo de lo que se desee resolver, se escogerá uno u otro algoritmo. A pesar de eso, estos enfoques algorítmicos se aproximan bastante bien a los resultados deseados en un tiempo razonable.

9. Referencias

[1] K. Deb. “Evolutionary Algorithms for Multi-Criterion Optimización in Engineering Design”. In Proceedings of Evolutionary Algorithms in Engineering and Computer Science EUROGEN’99. 1999.

[2] C. Coello. An updated Survey of Evolutionary Multiobjective Optimización Techniques: state of the art and future trends. In Congress on Evolutionary Computation. Piscataway, N. J., IEEE Service Center. 3–13.

[3] D. Pinto y B. Barán. “Solving Multiobjective Multicast Routing Problem with a new Ant Colony Optimización approach”. LANC’05, Cali, Colombia. 2005.

[4] C. García-Martínez, O. Cordón y F. Herrera. “An Empirical Análisis of Multiple Objective Ant Colony Optimización Algorithms for the Bi-criteria TSP”. ANTS Workshop 61-72. 2004

[5] J. Paciello, H. Martínez, C. Lezcano and B. Barán. Algoritmos de Optimización multi-objetivos basados en colonias de hormigas. Proceedings of CLEI’2006. Latin-American Conference on Informatics (CLEI). Santiago, Chile.