Optimización Multi-Objetivo: Problema de Asignación Cuadrática

Sergio Britos
Electiva 3 – Inteligencia Artificial, Ingeniería Informática, Universidad Nacional de Asunción, San Lorenzo
trojan.v6@gmail.com

Resumen. Los Problemas de optimización multi-objetivo consisten en problemas en los que se busca satisfacer varios objetivos que se contraponen entre sí. En específico se ataca el problema de la asignación cuadrática (QAP), el cual consiste en ubicar N instalaciones en N sitios considerando un costo que depende de las distancias y flujos entre las instalaciones. Para ello se utilizan dos estrategias: SPEA (Strength Pareto Evolutionary Algorithm) y MOACS (Multi-Objective Ant Colony System).

Palabras Clave: QAP, SPEA, Pareto, Multi-objetivo, MOACS.

1. Introducción

El problema de la asignación cuadrática, que se denota por sus siglas en inglés QAP (Quadratic assignment problem), fue planteado por Koopmans y Beckmann en 1957 como un modelo matemático para un conjunto de actividades económicas indivisibles, es uno de los problemas fundamentales de optimización combinatoria en las ramas de optimización e investigación de operaciones. QAP pertenece a los problemas no polinomiales duros, lo que sumado a que es un problema aplicable a un sinnúmero de situaciones, lo hacen un problema de gran interés para el estudio [1].

2. Descripción del Problema

El problema modela la siguiente situación de la vida real:

"Hay un conjunto de n instalaciones y un conjunto de n ubicaciones. Para cada par de ubicaciones se especifica una distancia y para cada par de instalaciones se especifica un peso o flujo (por ejemplo, la cantidad de suministros transportados entre las dos instalaciones). El problema es asignar todas las instalaciones a diferentes ubicaciones con el objetivo de minimizar la suma de las distancias multiplicadas por los flujos correspondientes".

La formulación matemática considera n instalaciones y n localidades, una matriz $A = \{a_{ij}\} \in \mathbb{R}^{nxn}$ donde a_{ij} representa la distancia entre las localidades i y j, y b matrices $B^q = \{b_{ij}^q\} \in \mathbb{R}^{nxn}$, q=1,..,b donde b_{ij}^q representa el q-ésimo flujo entre las instalaciones ubicadas en las localidades i y j, el QAP se define como:

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

Donde minimizar se refiere a la optimalidad Pareto. Para este trabajo se consideró en bQAP (QAP bi-objetivo) [1].

3. Algoritmos implementados

El problema fue resuelto mediante dos métodos:

3.1 SPEA (Strength Pareto Evolutionary Algorithm)

El SPEA (Strength Pareto Evolutionary Algorithm) es un algoritmo de optimización multi-objetivo (MOO). Básicamente el SPEA almacena externamente las soluciones Pareto-optimas. En cada punto en el tiempo el conjunto Pareto externo contiene las soluciones no dominadas del espacio de búsqueda muestreado hasta el momento.
Esto asegura que las soluciones Pareto-optimas no se pierden. Los individuos son evaluados en dependencia del número de puntos Pareto por los cuales son cubiertos [2].

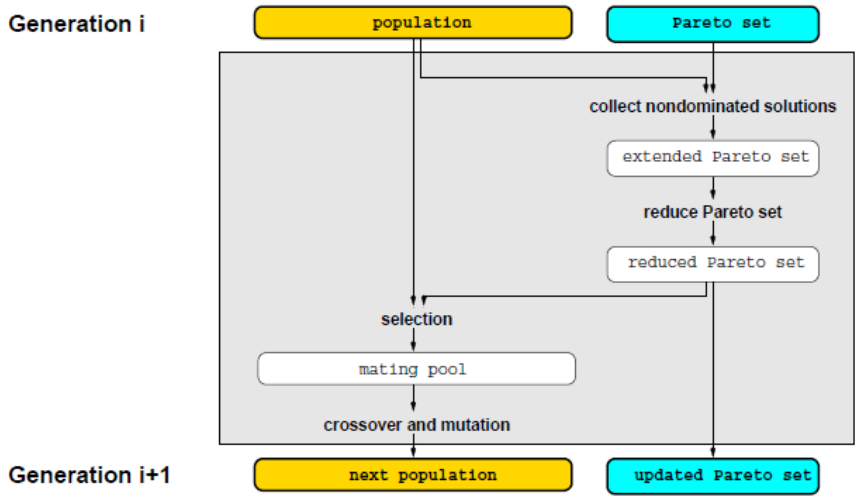


Figura 1: Algoritmo SPEA

```
PROCEDURE FitnessAssignemt
IN/OUT:
    population;
    paretoSet;
BEGIN
    (* update external Pareto set *)
    A := CollectNondominatedSolutions(population);
    B := CombineParetoSets(paretoSet, A);
    IF |B| > maxParetoPoints THEN
        paretoSet := ReduceParetoSet(B);
    ELSE
        paretoSet := B;
    FI
    (* calculate Pareto strengths *)
    FOR paretoInd IN paretoSet DO
        count := 0;
        FOR popInd IN population DO
            IF Covers(paretoInd, popInd) THEN
                count := count + 1;
            FI
        OD
        strength := count / (|population| + 1);
        SetFitness(paretoInd, strength);
    OD
    (* determine fitness values *)
    FOR popInd IN population DO
        sum := 0;
        FOR paretoInd IN paretoSet DO
            IF Covers(paretoInd, popInd) THEN
                sum := sum + GetFitness(paretoInd);
            FI
        OD
        SetFitness(popInd, sum + 1);
    OD
END
```

Algoritmo 1. Pseudocódigo de asignación de fitness del algoritmo SPEA .

También reduce el conjunto Pareto mediante clustering, ya que el conjunto Pareto-optimo puede ser extremadamente grande. Para ello al inicio cada elemento del conjunto Pareto original forma un cluster básico, entonces en cada paso dos clusters son elegidos para fusionarse en un cluster más grande hasta que el número dado de clusters es alcanzado. Los dos clusters son seleccionados de acuerdo al criterio del vecino más cercano, donde la distancia entre dos clusters es dada como la distancia promedio entre pares individuales a través de los dos clusters. Finalmente cuando el proceso de partición finaliza, el conjunto Pareto reducido se forma seleccionando un individuo representativo para cada grupo. Se considera el Centroide (el punto con distancia promedio mínima a todos los otros puntos en el cluster) como solución representativa. El Algoritmo 2 ilustra este proceso.

```
PROCEDURE ReduceParetoSet
IN/OUT:
    paretoSet;
BEGIN
    (* initialization: each Pareto point forms a cluster *)
    clusterSet := {};
    FOR paretoInd IN paretoSet DO
        clusterSet := clusterSet ∪ {{paretoInd}};
    OD
    (* join clusters until numbers of clusters remains under maximum *)
    WHILE |clusterSet| > maxParetoPoints DO
        (* select two clusters which have minimum distance *)
        minDistance := ∞;
        FOR {X, Y} SUBSET OF clusterSet DO
            IF ClusterDistance(X, Y) < minDistance THEN
                cluster1 := X;
                cluster2 := Y;
                minDistance := ClusterDistance(cluster1, cluster2);
            FI
        OD
        (* join the two selected clusters *)
        newCluster := cluster1 ∪ cluster2;
        clusterSet := clusterSet \ {cluster1, cluster2};
        clusterSet := clusterSet ∪ {newCluster};
    OD
    (* for each cluster pick out representative solution (centroid) *)
    paretoSet := {};
    FOR cluster IN clusterSet DO
        paretoInd := GetCentroid(cluster);
        paretoSet := paretoSet ∪ {paretoInd};
    OD
END
```

Algoritmo 2. Pseudocódigo del procedimiento de clustering en SPEA.

3.2 MOACS (Multi-Objective Ant Colony System)

MOACS es un algoritmo MOACO (Multi-Objective Ant Colony Optimization). Considera dos objetivos, utiliza una matriz de feromonas y dos visibilidades, una para cada objetivo del problema. Sigue una regla pseudo-aleatoria y se escoge el estado j como siguiente a visitar, estando en el estado i, según [3]:

$$j = \begin{cases} \max\limits_{j \in J_i} \left\{ \tau_{i,j} \left[\eta^0_{i,j} \right]^{\lambda\beta} \left[\eta^1_{i,j} \right]^{(1-\lambda)\beta} \right\} & si\ q < q_0 \\ \hat{i} & en\ caso\ contrario \end{cases} \qquad (2)$$

El cálculo de la variable $\hat{\imath}$ se realiza utilizando la probabilidad p_{ij} dada por:

$$p_{ij}=\begin{cases}\dfrac{\tau_{i,j}\left[\eta^{0}_{i,j}\right]^{\lambda\beta}\left[\eta^{1}_{i,j}\right]^{(1-\lambda)\beta}}{\sum_{x\in J_i}\tau_{i,x}\left[\eta^{0}_{i,x}\right]^{\lambda\beta}\left[\eta^{1}_{i,x}\right]^{(1-\lambda)\beta}} & si\ j\in J_i \\ 0 & en\ caso\ contrario\end{cases} \quad (3)$$

Input

G (V, A) … Graph: Set of nodes or cities joined by edges

α…relative influence of the pheromone traces

$\lambda_1, \lambda_2, .. \lambda_v..$ relative influence of the visibilities

ρ … evaporation rate

1: Read entry parameters
2: *Pareto_set* = empty
3: Initialize pheromone matrix $\tau_{ij} = \tau_0$ $\forall$ (i,j) $\in$ A
4: ***From*** w =1 until *Quantity_of_Generations*
5: ***From*** h = 1 until *Quantity_of_Ants* = *H*
6: Construct solution $\psi_{w,h}$
7: Calculate objective function $F(\psi_{w,h})$
8: ***End From*** h
9: Find no-dominated solutions (from the *Quantity _of_Ants* available)
10: Update *Pareto_Set*
11: Update pheromone matrix
12: ***End From*** w
13: Print *Pareto_set* of solutions
Exit

Output:

Pareto set of solutions $\psi_{(\mu)}$ with their respective objective functions $F(\psi_{(\mu)})$, where $F(\psi_{(\mu)}) = \left[F_1(\psi_{(\mu)}), F_2(\psi_{(\mu)}), \ldots F_u(\psi_{(\mu)})\right]$.
$\mu \in \mathbb{N} - \{0\}$ represents the number of solutions within the *Pareto_set*

Algoritmo 3. Pseudocódigo general del MOACS.

4 Resultados Experimentales

El ambiente de ejecución fue el siguiente:

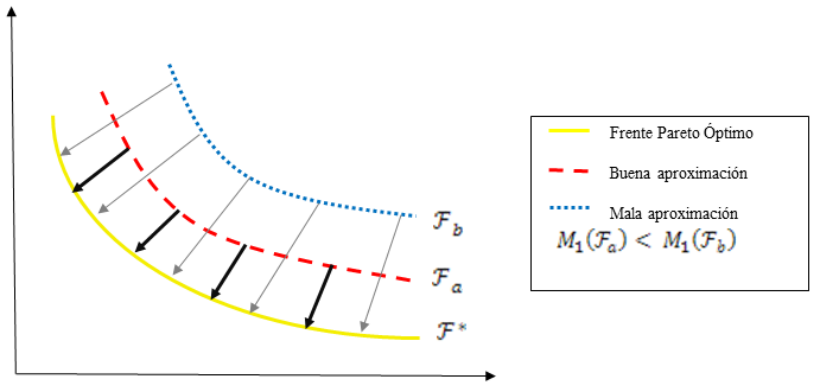
- Sistema operativo: Windows 10 Pro 64-bit
- Procesador: Intel(R) Core(TM) i7-4712MQ CPU @ 2.30GHz (8 CPUs)
- Memoria: 8192MB RAM

Implementación

- Python 3.5.1

Métricas de evaluación:

- Distancia al frente Pareto (M1): Para evaluar un conjunto de soluciones, se realiza la medición basada en la distancia a un conjunto Pareto óptimo de referencia, más precisamente, la distancia promedio desde cada solución de referencia a la más cercana. A menor distancia mejor evaluación tendrá el frente.
- **Figura 4.1**: Métrica M1

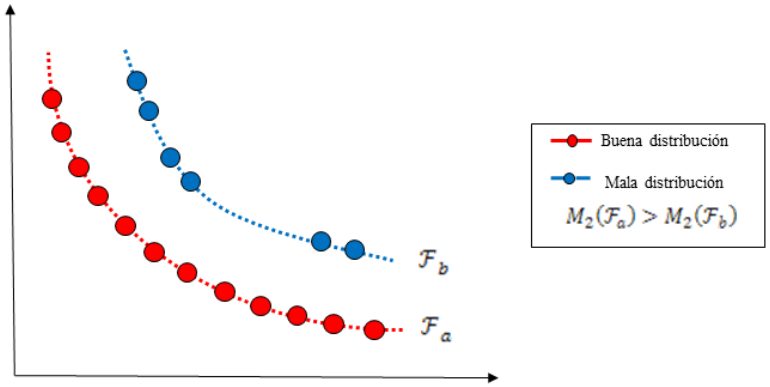


Distancia al Frente Pareto

$$M_1(\mathcal{F}) = \frac{1}{|\mathcal{F}|} \sum_{p \in \mathcal{F}} \min\{ d(p,q) \mid q \in \mathcal{F}^* \}$$

- Distribución del frente Pareto (M2): evalúa la distribución de los puntos en el frente Pareto mediante las distancias relativas entre las soluciones, tales que esas distancias sean superiores a un valor sigma (sigma=1000 para esta implementación).

Figura 4.2: Métrica M2

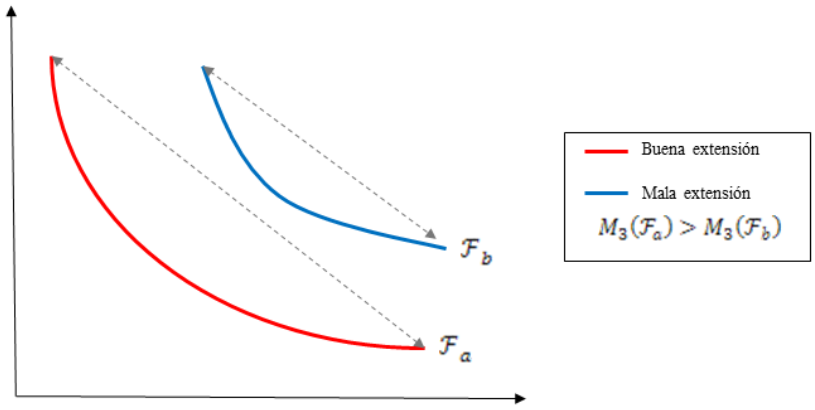


Distribución del frente Pareto

$$M_2(\mathcal{F}) = \frac{1}{|\mathcal{F}| - 1} \sum_{p \in \mathcal{F}} |\{ q \in \mathcal{F} \mid d(p,q) > \sigma \}|$$

- Extensión del frente Pareto (M3): simplemente evalúa las distancias entre los puntos extremos del frente Pareto.

Figura 4.4: Métrica M3

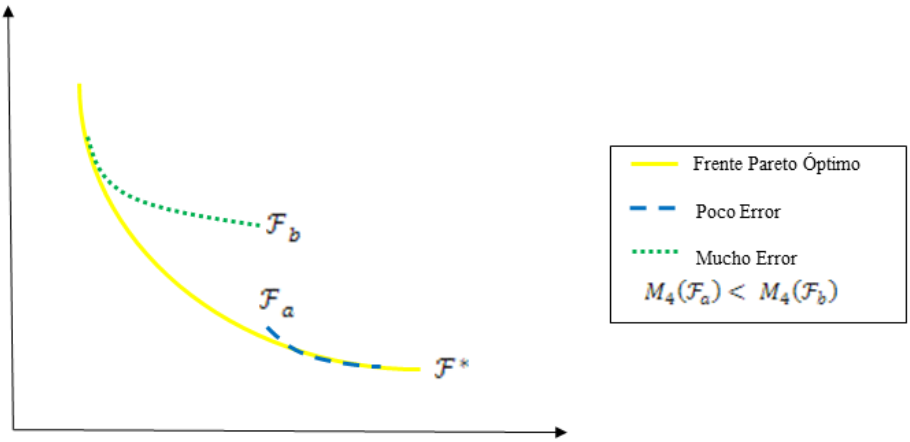


Extensión del frente Pareto

$$M_3(\mathcal{F}) = \sqrt{\sum_{i=1}^{k} \max\{(|p_i - q_i|) \mid p,q \in \mathcal{F}\}}$$

- Error (M4): realiza la evaluación de acuerdo a la cantidad de puntos del frente Pareto evaluado que coinciden con un frente Pareto óptimo o referencial.

Figura 4.1: Métrica M4



Error

$$M_4(\mathcal{F}) = 1 - \frac{|\mathcal{F} \cap \mathcal{F}^*|}{|\mathcal{F}|}$$

Tabla 1. Comparaciones entre los diferentes métodos implementados

	Instancia 1									
	SPEA					MOACS				
Ejecución	M1	M2	M3	M4	Tiempo (seg.)	M1	M2	M3	M4	Tiempo (seg.)
1	41.061,5781	2,0	104.189,6535	1,00	0,7589	77.554,8456	5,0	331.097,6106	1,00	645,9319
2	69.077,6012	6,0	236.227,0369	1,00	0,6494	29.953,0712	13,0	479.621,0338	1,00	658,9248
3	63.287,6296	4,0	249.898,8124	1,00	0,7253	74.444,3688	6,0	614.957,1164	1,00	577,4550
4	80.305,0211	6,0	217.944,0698	1,00	0,7079	51.832,3580	9,0	577.890,6092	1,00	751,6772
5	107.625,1192	5,0	250.975,9583	1,00	0,7651	52.216,3287	6,0	449.146,5700	1,00	691,3279
Promedio:	72.271,3898	4,6	211.847,1062	1,00	0,7213	57.200,1945	7,8	490.542,5880	1,00	665,0633

	Instancia 2									
	SPEA					MOACS				
Ejecución	M1	M2	M3	M4	Tiempo (seg.)	M1	M2	M3	M4	Tiempo (seg.)
1	76.646,2351	3,0	137.984,5216	1,00	0,8908	63.610,4481	11,0	466.008,2148	1,00	814,1166
2	46.496,8003	7,0	322.556,9013	1,00	0,7942	70.866,3317	7,0	436.012,4543	1,00	771,8748
3	30.227,7655	3,0	157.808,7835	1,00	0,8891	48.327,0491	6,0	456.448,2780	1,00	657,8778
4	46.442,9947	5,0	275.535,9207	1,00	0,8266	67.464,3654	7,0	456.061,6995	1,00	739,8399
5	50.985,1657	5,0	338.432,2206	1,00	0,6563	45.516,1697	11,0	423.841,7137	1,00	590,2478
Promedio:	50.159,7923	4,6	246.463,6695	1,00	0,8114	59.156,8728	8,4	447.674,4721	1,00	714,7914

Dado los resultados de las pruebas realizadas (Tabla 1), se puede deducir que: el algoritmo MOACS obtiene mejores resultados en promedio en la instancia 1 para las métricas M1, M2 y M3 por una amplia ventaja, no así con la métrica M4 en la que empata en todas la ejecuciones. Para la instancia 2, de las cinco ejecuciones y considerando la métrica M1, SPEA fue mejor en promedio, ganando en 3 de los 5 casos, no obstante considerando las otras métricas (M2 y M3) MOACS fue altamente superior, nuevamente hubo un empate con la métrica M4, esto se debe a que en ninguna de las ejecuciones algún punto del frente Pareto coincidió con el frente Pareto de referencia. Otro detalle importante que se puede observar es la diferencia en el tiempo de ejecución entre ambos algoritmos, siendo el MOACS, por mucho, más lento que el SPEA (Con el tiempo promedio del SPEA menor a 1 segundo en todas las instancias y ejecuciones, mientras el MOACS supera los nueve minutos en todas las ejecuciones).

5 Conclusiones

Con las pruebas realizadas se concluye que para el problema de asignación cuadrática, y considerando las métricas de evaluación presentadas, el algoritmo MOACS presenta una performance bastante superior al algoritmo SPEA, a cambio de un tiempo de ejecución notablemente superior. Esta gran diferencia principalmente se debe a que MOACS utiliza una colonia de hormigas para minimizar simultáneamente cada función objetivo, considerando todos los objetivos igual de importantes, de esa forma un conjunto Pareto óptimo puede encontrarse en solo una ejecución de la meta heurística.

Para trabajos futuros se podrían considerar una mayor cantidad de instancias de pruebas y realizar una muestra mayor de ejecuciones, usando otras métricas de evaluación. También se podrían mejorar los frentes Pareto de referencia obteniéndolos algoritmos distintos.

Referencias

1. Quadratic assignment problem. (2016, 3 de noviembre). *Wikipedia, La enciclopedia libre.* https://en.wikipedia.org/wiki/Quadratic_assignment_problem. Fecha de consulta: noviembre 11, 2016,
2. E. Zitzler; L. Thiele: An Evolutionary Algorithm for Multiobjective Optimization: *The Strength Pareto Approach*.Computer Engineering and Communication Networks Lab. Swiss Federal Institute of Technology, Mayo 1998.
3. B. Barán; M. Schaerer: A Multiobjective Ant Colony System for Vehicle Routing Problem with Time Windows. Centro Nacional de Computación, Universidad Nacional de Asunción San Lorenzo, Febrero 2003.