Universidad Eafit
Escuela de Ciencias Aplicadas e Ingeniería
CM0050 − Optimización II

Exploración y Evaluación de Algoritmos de Búsqueda de la Ruta Más Corta basados en la Contaminación del Aire

Efrain Jurado-Torres
Valentina Movil-Sandoval

15 de mayo de 2024

Resumen

La contaminación atmosférica, generada por la presencia de agentes nocivos en el aire, cobra la vida de 6.7 millones de personas en el mundo cada año [1]. En el Área Metropolitana del Valle de Aburrá (AMVA), el aumento de la concentración de contaminantes como el material particulado PM_{10} o el más peligroso $PM_{2,5}$ no solo desencadena enfermedades cardiorespiratorias, sino que también resulta en un promedio adicional de 15 defunciones anuales [2]. El parque automotor del AMVA es responsable del 82 % de las emisiones en la región [3]. Por ello, el gobierno local promueve el uso del transporte público y la bicicleta. Sin embargo, para los ciclistas que se desplazan regularmente en un entorno contaminado, esta práctica se convierte en una actividad de alto riesgo según [4]. Por tanto, el propósito de este estudio es explorar algoritmos como Dijkstra, A-star, BOA-star, NGSA II, Dijkstra-ϵ, A-star-ϵ y Pulse, para determinar la ruta ciclista más corta y con la menor contaminación del aire, y en consecuencia, ayudar a mitigar el riesgo para la salud asociado al uso de la bicicleta en ciudades con alta polucón. Finalmente, en este trabajo se evaluaran todos los algoritmos citados mediante estrategias de optimización multicriterio como SHANNON y TOPSIS, en aras de seleccionar el mejor algoritmo basándose en la calidad de la solución (distancia y cantidad de polución) y el tiempo de ejecución.

Palabras clave: Optimización Multiobjetivo, Problema de ruta más corta, Problema de ruta más corta multiobjetivo, Contaminación del aire, Salud Pública, Valle de Aburrá, Matemática Biológica.

1. Introducción

La contaminación atmosférica plantea una seria amenaza para la salud humana debido a la presencia de diversos agentes nocivos en la atmósfera, tales como el material particulado ($PM_{2,5}$ y PM_{10}), el monóxido de carbono (CO), el ozono (O_3), el hollín (BC), el dióxido de azufre y el óxido de nitrógeno (NO_x). Estos elementos pueden desencadenar enfermedades cardiorespiratorias e incluso ocasionar la muerte, como lo señala la Organización Mundial de la Salud (OMS). Es por ello que,

resulta alarmante que cerca del 95 % de la población mundial resida en zonas donde no se cumplen las directrices de calidad del aire establecidas por dicha organización. Prueba de ello es que, en Colombia, se han registrado concentraciones solo para $PM_{2,5}$ que duplican, triplican e incluso cuadruplican estos límites, especialmente en zonas de topología compleja y/o con una gran población [5].

El Área Metropolitana del Valle de Aburrá (AMVA), compuesta por 11 municipios que en conjunto albergan alrededor de 4 millones de habitantes, es una de las regiones más contaminadas de Colombia. Este problema se agrava con el reciente crecimiento del parque automotor, que es responsable del 82 % de las emisiones. En respuesta a esta situación, el gobierno local ha promovido el uso de medios de transporte sostenibles, como caminar y andar en bicicleta. Hoy en estos medios, se llevan a cabo más del 50 % de los viajes en la zona [2]. Sin embargo, estudios han revelado que en áreas con alta contaminación del aire, como el Valle de Aburrá, tanto peatones como ciclistas enfrentan un mayor riesgo a quebrantos de salud debido a su prolongada exposición a la polución [4]. Por lo tanto, es esencial identificar rutas que no solo ofrezcan comodidad al transeúnte, minimizando la distancia que debe recorrer, sino también, rutas en las que se reduzca al máximo la exposición humana a la contaminación atmosférica. Este problema se conoce en la literatura como el problema del camino más corto multiobjetivo (o simplemente MOSP, por sus siglas en inglés).

1.1. Estado del Arte

En el marco del problema del camino más corto biobjetivo (BOSP) encapsulado a su vez, en el problema del camino más corto multiobjetivo (MOSP), se busca determinar una ruta o conjunto de rutas que optimicen dos o múltiples funciones de costo simultáneamente [6]. Es por ello que se han ido desarrollando paulatinamente diversos enfoques para solucionar este problema. Dichos enfoques pueden agruparse en métodos de etiquetado, métodos de clasificación, métodos basados en programación dinámica y métodos basados en relajación de restricciones.

En el caso de los métodos de etiquetado, introducidos en 1980, estos fueron pioneros en la resolución del problema biobjetivo (BOSP) y multiobjetivo (MOSP) del camino más corto, ya que etiquetan los caminos de las rutas según su distancia, tiempo, etc. Un ejemplo clásico de algoritmo de etiquetado en el caso mono-objetivo son los algoritmos Dijsktra y A* [7], los cuales, tienen sus respectivas versiones multiobjetivo desarrolladas entre 1990 y el 2015, llamadas T-MDA y MOA*. Con esta nueva perspectiva, los estudiosos del tema han querido examinar el impacto de las funciones heurísticas en la resolución de problemas del camino más corto a gran escala [8].

Seguidamente, el enfoque de clasificación propuesto en 1981, plantea que las soluciones al MOSP puedan organizarse teniendo en cuenta los valores de solo una función objetivo hasta identificar todas las soluciones eficientes. Un ejemplo notable de esto es el algoritmo PULSE introducido por [9] en 2015. Este algoritmo emplea una búsqueda en profundidad junto con estrategias de poda para evitar el cálculo de soluciones dominadas o poco prometedoras, lo que resulta en una implentación rápida en términos de tiempo de ejecución en el contexto del MOSP o BOSP de medianas y altas dimensiones.

Por otro lado, la programación dinámica pretende resolver el MOSP descomponiendo el problema en subproblemas más pequeños y recurrentes para, en últimas, encontrar la solución óptima combinando las soluciones de estos subproblemas [10]. Mientras que, el enfoque basado en métodos de relajación

plantea resolver el problema del camino más corto multiobjetivo escalando el problema original, esto es, no considerar una o algunas de las restricciones, facilitando así la solución del problema. Sin embargo, esta técnica puede no ser eficiente, ya que puede generar una cantidad excesiva de caminos que no cumplen con el consumo máximo de recursos [11].

En últimas, otros métodos propuestos en la literatura incluyen un acercamiento a la técnica de multiplicadores de Lagrange generalizados, los cuales ofrecen una solución dual que podría diferir de la solución primal, lo que requiere el empleo de otros procedimientos para solventar este obstáculo según [12].

1.2. Planteamiento del Problema

El propósito de este trabajo es explorar y evaluar diversos algoritmos de ruteo para solucionar el problema del camino más corto y con la menor exposición humana a la contaminación atmosférica.

1.2.1. Planteamiento matemático del problema

El problema que se pretende solucionar, matemáticamente puede escribirse siguiendo la lógica asociada al BOSP, esto es:

Sea $G = (V, A)$ un grafo dirigido con un conjunto de nodos $V = \{1, \ldots, n\}$. Supongamos que tenemos un nodo de inicio $s \in V$ y un nodo destino $t \in V$. El conjunto $A = \{(i, j) : i, j \in V, i \neq j\}$ representa los arcos dirigidos, donde cada arco está asociado con dos pesos: la distancia c^1_{ij} y la contaminación atmosférica c^2_{ij}. Entonces, el objetivo es:

$$
\begin{aligned}
&\text{mín } c(x) = \left(\sum_{(i,j)\in A} c^1_{ij} x_{ij} \, , \sum_{(i,j)\in A} c^2_{ij} x_{ij} \right) \\
&\text{s.a.} \quad \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = \begin{cases} 1 & \text{si } i = s, \\ 0 & \text{si } i \neq s, n, \\ -1 & \text{si } i = n, \end{cases} \\
&\quad x_{ij} \in \{0, 1\}, \quad \forall\, (i, j) \in A.
\end{aligned} \tag{1}
$$

2. Materiales y Métodos

En esta sección se presentan los algoritmos que se emplearán para resolver el BOSP basado en la exposición humana a la contaminación atmosférica.

2.1. Métodos de etiquetado

Haciendo uso de métodos de etiquetado como el Dijstrak y el A*, se busca resolver en primer lugar el problema de encontrar: solo la ruta más corta y solo la ruta con menor contaminación del aire. Posteriormente, empleando la versión biobjetivo del algoritmo A*, es decir, el algoritmo BOA* y un enfoque génetico denominado NGSA-II, se soluciona el problema de encontrar la ruta más corta y con menor contaminación del aire. En consecuencia, en esta sección, se presentan brevemente

los algoritmos mencionados, y para cada uno, se expone su respectivo pseudocódigo, siguiendo la notación matemática ya planteada.

2.1.1. Algoritmo Dijkstra

El algoritmo de Dijkstra, conocido como algoritmo de caminos mínimos, encuentra el camino más corto desde un vértice origen hacia todos los demás vértices en un grafo ponderado mediante *pesos*. Fue propuesto por el científico de la computación Edsger Dijkstra en 1956 y publicado por primera vez en 1959.

Algoritmo 1: Pseudocódigo de Dijkstra

Entrada: $G = (V, A)$ un grafo, un nodo inicial
Salida : $\mathbf{d}$, las distancias más cortas desde el nodo inicial a todos los demás nodos

```
1  d[v] ← ∞ para cada v ∈ V
2  d[inicio] ← 0
3  Abiertos ← V
4  while Abiertos ≠ ∅ do
5      Sea v el nodo en Abiertos con la menor d[v]
6      Remover v de Abiertos
7      foreach vecino ∈ vecinos(v) do
8          if d[v] + coste(v, vecino) < d[vecino] then
9              d[vecino] ← d[v] + peso(v, vecino)
10 return d
```

2.1.2. Algoritmo A*

La heurística de búsqueda A* encuentra el camino de menor costo entre un nodo origen y uno objetivo, considerando tanto la función heurística (h) como el costo real de desplazamiento entre nodos. Desarrollada en 1968 por Hart, Nilsson y Raphael, es más efectiva que otros algoritmos informados al combinar ambos factores.

Algoritmo 2: Pseudocódigo del A*

Entrada: $G = (V, A)$ un grafo, un nodo inicial, un nodo meta, y una función heurística h
Salida : $\mathbf{d}$, las distancias más cortas desde el nodo inicial al nodo meta

```
1  d[v] ← ∞ para cada v ∈ V
2  d[inicio] ← 0
3  Abiertos ← {inicio}
4  while Abiertos ≠ ∅ do
5      Sea v el nodo en Abiertos con la menor d[v] + h(v, meta)
6      Remover v de Abiertos
7      if v es el nodo meta then
8          return d
9      foreach vecino ∈ vecinos(v) do
10         if d[v] + peso(v, vecino) + h(vecino, meta) < d[vecino] then
11             d[vecino] ← d[v] + peso(v, vecino)
12             Añadir vecino a Abiertos
```

2.1.3. Algoritmo BOA*

El algoritmo BOA*, propuesto por Hernández et al. [13], permite resolver el MOSP biobjetivo con revisiones de dominancia en tiempo constante. Similar al A* clásico, emplea valores g, h, y f, pero como tuplas para los dos costes del grafo. Usa una lista de nodos abiertos y selecciona el nodo con el menor valor lexicográfico de f. BOA* utiliza valores $g_2^{\text{mín}}(v)$ para cada nodo v, y realiza podas basadas en estos valores sin necesidad de considerar g_1 o f_1 debido al orden de la lista de abiertos. Cuando un nodo supera las podas y es la meta, se agrega a las soluciones. Nuestra implementación utiliza Dijkstra dos veces para calcular h_1 y h_2, las distancias mínimas al nodo final respecto a cada coste.

Algoritmo 3: Pseudocódigo del A* Bi-Objectivo (BOA*)

Input : $G = (V, A)$ un grafo, un nodo inicial, un nodo meta, y una función heurística consistente h

Output: *sols*, un conjunto de soluciones únicas en costo que son Pareto óptimas

```
1  sols ← ∅
2  foreach v ∈ V do
3      g_2^mín(v) ← ∞
4  x ← el nodo inicial
5  g(x) ← (0, 0)
6  padre(x) ← nulo
7  f(x) ← (h_1(x), h_2(x))
8  Inicializar Abiertos y añadirle x
9  while Abiertos ≠ ∅ do
10     Remover un nodo x de Abiertos con el valor lexicográficamente más pequeño de f
11     if g_2(x) ≥ g_2^mín(x) ∨ f_2(x) ≥ g_2^mín(meta) then
12         continue
13     g_2^mín(x) ← g_2(x)
14     if x = meta then
15         Añadir x a sols
16         continue
17     foreach y ∈ succ(x) do
18         g(y) ← g(x) + c_xy                    ▷ el coste del arco (x, y)
19         padre(y) ← x
20         f(y) ← g(y) + h(y)
21         if g_2(y) ≥ g_2^mín(y) ∨ f_2(y) ≥ g_2^mín(meta) then
22             continue
23         Añadir y a Abiertos
24 return sols
```

2.1.4. Algoritmo NSGA-II

El NSGA-II es una versión mejora del algoritmo genético de ordenación no dominada (NSGA) propuesto en 2002. Este nuevo enfoque se emplea para abordar problemas de optimización multiobjetivo, como el MOSP. En este algoritmo, iterativamente se va generando una nueva población de soluciones a partir de la población inicial. En cada iteración, se ordenan las soluciones según su rango de no dominancia y se calcula la distancia de agrupamiento de cada una. Luego, la nueva población se compone seleccionando soluciones de acuerdo con su rango de no dominancia y distancia de agrupamiento. Cabe resaltar que, el rango de no dominancia de una solución indica cuántas otras soluciones domina, mientras que la distancia de agrupamiento mide su proximidad con otras soluciones en el mismo nivel de no dominancia, fomentando la diversidad en la nueva población. Posteriormente, el proceso iterativo descrito continúa hasta que se satisfaga un criterio de parada, comúnmente un número máximo de iteraciones, aunque pueden emplearse otros criterios. En últimas, la nueva población resultante se convierte en el punto de partida para la siguiente iteración del

algoritmo. Observe el mismo a continuación:

Algoritmo 4: Pseudocódigo del NGSA-II

```
P ← PoblaciónInicial()                                   ▷ Crear Población Inicial
OrdenamientoNoDominado(P)                                ▷ Clasificación No Dominada
CalcularDistanciaMultitud(P)                             ▷ Calcular Distancia Multitud
    while no SeCumpleCriterioTerminación() do
    Bucle Principal
// Reproducción
Padres ← SeleccionarPadres(P)                            ▷ Seleccionar Padres
Q ← CruzamientoYMutación(Padres)                         ▷ Obtener Descendientes
R ← P + Q                                                ▷ Fusión
// Clasificación y Selección
OrdenamientoNoDominado(R)                                ▷ Clasificación No Dominada
CalcularDistanciaMultitud(R)                             ▷ Calcular Distancia Multitud
P ← SeleccionarSobrevivientes(R)                         ▷ Seleccionar Próxima Generación
    if SeCumpleCriterioTerminación() then
    romper                                               ▷ Salir del Bucle
retornar P                                               ▷ Retornar Población Final
```

2.2. Métodos de clasificación

De manera simular al enfoque brevemente descrito. Para solucionar el BOSP, se emplean métodos de clasificación basados en restricciones ϵ. Por lo tanto, se modifica la estructura de los algoritmos Dijkstra y A* incorporando una restricción de recurso dada por ese ϵ y además, se trabaja con un algoritmo popular de clasficación propuesto en la literatura asociada al problea del camino más corto biobjetivo o multiobjetivo, es decir: el algoritmo Pulse. De maenra que, a continuación se reformula el problema matemático (1) para abordar la cosntrucción de Dijkstra-ϵ y A*-ϵ, y se presentan una breve descripción así como los pseudocódigos de los algoritmos ya mencionados.

2.2.1. Método ϵ-restringido

Una estrategia para abordar el problema (1) implica reformularlo de modo que una de las funciones objetivo se seleccione para optimización, mientras que las otras se transforman en restricciones. De acuerdo con [8], el problema reformulado se presenta de la siguiente manera:

$$\begin{aligned} \text{mín } & f_l(x) \\ \text{s.a. } & f_j(x) \leq \epsilon_j \\ \text{para todo } \quad & j = 1, ..., k \quad j \neq l \\ & x \in S \end{aligned} \tag{2}$$

Donde $l \in \{1, ..., k\}$ (siendo k el número de funciones objetivo), S es el conjunto de soluciones factibles, y ϵ_j es el limite superior para la función objetivo j.

Siguiendo este enfoque, se reemplaza el BOSP original por un *problema del camino más corto limitado por un único recurso* (SRCSP), un caso especial del *problema del camino más corto limitado por recursos* (CSP). Este tipo de problema busca el camino más corto sin exceder el uso de un recurso específico, en este caso, la contaminación atmosférica. La formulación matemática del SRCSP es la siguiente:

Dado un grafo dirigido $G = (V, A)$:

$$\begin{aligned}
&\text{mín} \sum_{(i,j)\in A} c_{ij}x_{ij} \\
&\text{s.a.} \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = \begin{cases} 1 & \text{si } i = s, \\ 0 & \text{si } i \neq s, n, \\ -1 & \text{si } i = n, \end{cases} \\
&\sum_{(i,j)\in A} t_{ij}x_{ij} \leqslant \epsilon \\
&x_{ij} \in \{0, 1\}, \text{ para todo } (i, j) \in A,
\end{aligned}$$

Donde V es el conjunto de nodos, el nodo inicial se etiqueta como s, y el final como n; A es el conjunto de arcos dirigidos con pesos asociados, c_{ij} representa el costo a minimizar (distancia) y t_{ij} el recurso a no exceder (polución); ϵ es el valor máximo permitido de la suma de los pesos de los arcos t_{ij}.

Se presentan los algoritmos para Dijkstra y A* ϵ-restringido el mismo a continuación:

- **Dijkstra ϵ-restringido**

Algoritmo 5: Pseudocódigo del Dijkstra-ϵ

Entrada: $G = (V, A)$ un grafo, un nodo inicial y un consumo máximo de recurso ϵ
Salida : **d**, las distancias más cortas desde el nodo inicial a todos los demás nodos

```
1  d[v] ← ∞ para cada v ∈ V
2  d[inicio] ← 0
3  Abiertos ← V
4  while Abiertos ≠ ∅ do
5      Sea v el nodo en Abiertos con la menor d[v]
6      Remover v de Abiertos
7      foreach vecino ∈ vecinos(v) do
8          if t[v] + recurso(v, vecino) > ε then
9              continue
10         if d[v] + coste(v, vecino) < d[vecino] then
11             d[vecino] ← d[v] + peso(v, vecino)
12 return d
```

- **A* ϵ-restringido**

Algoritmo 6: Pseudocódigo del A*-ϵ

Entrada: $G = (V, A)$ un grafo, un nodo inicial, un nodo meta, una función heurística h y un consumo máximo de recurso ϵ
Salida : **d**, las distancias más cortas desde el nodo inicial al nodo meta

```
1  d[v] ← ∞ para cada v ∈ V
2  d[inicio] ← 0
3  Abiertos ← {inicio}
4  while Abiertos ≠ ∅ do
5      Sea v el nodo en Abiertos con la menor d[v] + h(v, meta)
6      Remover v de Abiertos
7      if t[v] + recurso(v, vecino) > ε then
8          continue d
9      if v es el nodo meta then
10         return d
11     foreach vecino ∈ vecinos(v) do
12         if d[v] + peso(v, vecino) + h(vecino, meta) < d[vecino] then
13             d[vecino] ← d[v] + peso(v, vecino)
14             Añadir vecino a Abiertos
```

2.2.2. Método PULSE

Este algoritmo propuesto por Lozano y Medaglia [9] resuelve el CSP propagando *pulsos* desde v_s hasta v_e mediante recursión. Cada *pulso* crea un camino parcial P, con coste $c(P)$ y consumo de recurso $t(P)$. Si los *pulsos* se propagan sin restricciones, se encuentra el camino óptimo P^*.

El algoritmo emplea tres estrategias de poda: dominancia, factibilidad y límites. De esta manera:

- La poda por dominancia consiste en descartar aquellas rutas hacia un nodo específico que sean dominadas por otras rutas previamente identificadas hacia ese mismo nodo. También se asegura que el nodo a evaluar no se encuentre en el camino parcial P para así evitar bucles.

- Al tener un camino no dominado, se procede a verificar la posibilidad de alcanzar el nodo final v_e desde el actual v_k respetando la restricción de los recursos. En nuestra implementación esto se logra mediante el uso del Algoritmo de Dijkstra con el cual se hallan los caminos que minimizan el consumo del recurso desde el nodo final hacia los demás nodos del grafo. Este proceso determina la cantidad mínima de recursos necesarios $\underline{t}(v_k)$ desde el nodo en evaluación hasta el nodo final. Si esta cantidad, sumada al consumo de recursos a lo largo del camino $t(P)$ supera el limite ϵ, ya se sabe que desde ese punto es imposible alcanzar el nodo final manteniendo la restricción de recursos, por lo que este camino es podado.

- La poda por limites sigue un proceso similar a la poda por factibilidad. Se aplica el algoritmo de Dijkstra para conocer el coste mínimo $\underline{c}(v_k)$ desde el nodo final hacia los demás nodos. Además, al identificarse la primera solución factible, es decir, cuando se alcanza el nodo final respetando

ϵ, se guarda su costo total como $\bar{c}$. En cada iteración subsiguiente, cuando un camino parcial que termine en un nodo v_k supere tanto las podas de dominancia como las de factibilidad, se verifica si la suma del costo de ese camino parcial $c(P)$ con el costo mínimo para alcanzar el nodo final $c(v_k)$ es mayor que el costo previamente obtenido $\bar{c}$. Esto puede ser expresado con la siguiente desigualdad: $c(P) + \underline{c}(v_k) \geq \bar{c}$.

Seguidamente, el algoritmo Pulse inicializa P, c^0, y t^0, encuentra los caminos más cortos hacia los nodos y ejecuta la función Pulse recursivamente. Y finalmente, la Función Pulse efectúa las tres podas, añade el nodo actual a P, y se llama recursivamente a los sucesores del nodo actual. Su algoritmo es:

Algoritmo 7: Pseudocódigo del Pulse

Input : $G = (V, A)$ un grafo, v_s un nodo de inicio, v_e un nodo meta, ϵ el consumo máximo del recurso
Output: Camino óptimo $\mathcal{P}^*$

```
1  P ← {}
2  c^0 ← 0
3  t^0 ← 0
4  inicialización(G, ε)
5  pulse(v_s, c^0, t^0, P)
6  return P*
```

Algoritmo 8: Función Pulse

Input : Un nodo v_k, un coste c, un consumo de recurso t, un camino parcial $\mathcal{P}$
Output: Ninguno

```
1  if checkDominance(v_k, c, t) es falso then
2      if checkFeasibility(v_k, t) es verdadero then
3          if checkBounds(v_k, c) es falso then
4              P' ← P ∪ {v_k}
5              foreach v_i ∈ succ(v_k) do
6                  c' ← c + c_ki
7                  t' ← t + t_ki
8                  pulse(v_i, c', t', P')
9              end
10         end
11     end
12 end
```

2.3. Selección mejor algoritmo

Seleccionar el mejor algoritmo implica la creación de una matriz de decisión que evalúe varias alternativas, tales como los algoritmos Dijkstra, A*, BOA*, NSGA-II, Dijkstra-ϵ, A*-ϵ y Pulse. Los

criterios para evaluar las alternativas son, para cada ruta (solución); la distancia recorrida, la cantidad de contaminación detectada y el tiempo en el que se obtuvo dicha ruta (que es lo mismo que el tiempo de ejecución). De manera que, para elegir la alternativa más adecuada, se utiliza el método de SHANNON-TOPSIS seleccionar en función de una ponderación basada en el concepto de entropía. A continuación, se presenta el pseudocódigo asociado a tal fin.

Algoritmo 9: Cálculo de pesos por entropía de Shannon

Entrada: Matriz inicial $A_{n\times m} = [a_{ij}]$
Salida: Pesos representativos de cada criterio w_j
Paso 1: Normalizar la matriz inicial **for** $j = 1$ *to* n **do**
    Calcular $\sum_{i=1}^{m} a_{ij}$ **for** $i = 1$ *to* m **do**
    Calcular $r_{ij} = \frac{a_{ij}}{\sum_{i=1}^{m} a_{ij}}$
Paso 2: Calcular el valor de la entropía **for** $j = 1$ *to* n **do**
    Calcular $e_j = -\frac{1}{\ln(m)} \sum_{i=1}^{m} r_{ij} \ln(r_{ij})$
Paso 3: Calcular el peso que representa cada criterio
Calcular $w_j = \frac{1-e_j}{\sum_{j=1}^{n}(1-e_j)}$ para $j = 1, 2, ..., n$
Devolver w_j para $j = 1, 2, ..., n$

Algoritmo 10: Proceso de TOPSIS

Entrada: Matriz de decisión $X_{m\times n} = [x_{ij}]$, Pesos de criterios w_j
Salida: Orden de preferencia de alternativas
Paso 1: Normalización de la Matriz de Decisión **for** $j = 1$ *to* n **do**
    Calcular $x_{+j} = \text{máx}_i(x_{ij})$ y $x_{-j} = \text{mín}_i(x_{ij})$ **for** $i = 1$ *to* m **do**
    El criterio j es de beneficio
Calcular $r_{ij} = \frac{x_{ij}}{x_{+j}}$ **else**
    Calcular $r_{ij} = \frac{x_{ij}}{x_{-j}}$
Paso 2: Cálculo de la Matriz de Decisión Normalizada Ponderada **for** $i = 1$ *to* m **do**
    $j = 1$ to n
Calcular $v_{ij} = w_j \times r_{ij}$
Paso 3: Determinación de las Soluciones Ideal y Anti-Ideal **for** $j = 1$ *to* n **do**
    Calcular $v_j^* = \text{máx}_i(v_{ij})$ y $v_j^- = \text{mín}_i(v_{ij})$
Paso 4: Cálculo de las Medidas de Separación **for** $i = 1$ *to* m **do**
    Calcular $D_i^* = \sqrt{\sum_{j=1}^{n}(v_{ij} - v_j^*)^2}$
Calcular $D_i^- = \sqrt{\sum_{j=1}^{n}(v_{ij} - v_j^-)^2}$
Paso 5: Cálculo de la Cercanía Relativa a la Solución Ideal **for** $i = 1$ *to* m **do**
    Calcular $C_i^* = \frac{D_i^-}{D_i^* + D_i^-}$
Paso 6: Ranking del Orden de Preferencia
Ordenar las alternativas de acuerdo a C_i^* de mayor a menor
Devolver Orden de preferencia de alternativas

3. Resultados

En esta sección se exponen todos los resultados

3.1. Caso de estudio 1

El caso 1 se evaluaron todos los algoritmos, seleccionando un nodo de origen y un nodo cuya distancia fuese pequeña. Posteriormente, usando el método SHANNON-TOPSIS se generó al tabla a continuación, que presenta de mejor a peor, los algoritmos evaluados. Cabe resaltar que, en los algoritmos del tipo ϵ-restringidos y el PULSE, se trabajó con una restricción de contaminación igual a 30 (ug/m^3).

Caso 1			
Nodo origen			
(-75.6052191, 6.2318821)			
Nodo destino			
(-75.589614, 6.2384034)			
	Pesos Shannon		
Distancia (km)	0.175		
Cantidad de polución (ug/mˆ3)	0.173		
Tiempo de ejecución (s)	0.067		
Algoritmos monobjetivo			
Algoritmo	**Distancia (km)**	**Cantidad de polución (ug/mˆ3)**	**Tiempo de ejecución (s)**
Dijkstra min. Polución	2.697	23.680	0.223
A* min. Polución	2.728	23.680	0.250
Dijkstra min. Distancia	2.038	36.085	0.221
A* min. Distancia	2.038	36.085	0.239
Algoritmos multiobjetivo			
Dijkstra-ϵ	2.065	26.812	0.212
A*-ϵ	2.065	26.812	0.222
BOA*	2.089	23.818	0.407
Pulse	2.065	26.812	0.423
NGSA-II	3.200	39.575	1.964
Promedios	2.332	29.262	0.462

Adicionalmente, los resultados para el caso 1. se presentan resumidamente en la siguiente figura. Aquí, se muestra la calidad de la solución en términos de distancia y contaminación del aire para diferentes enfoques mono-objetivo y multiobjetivo.

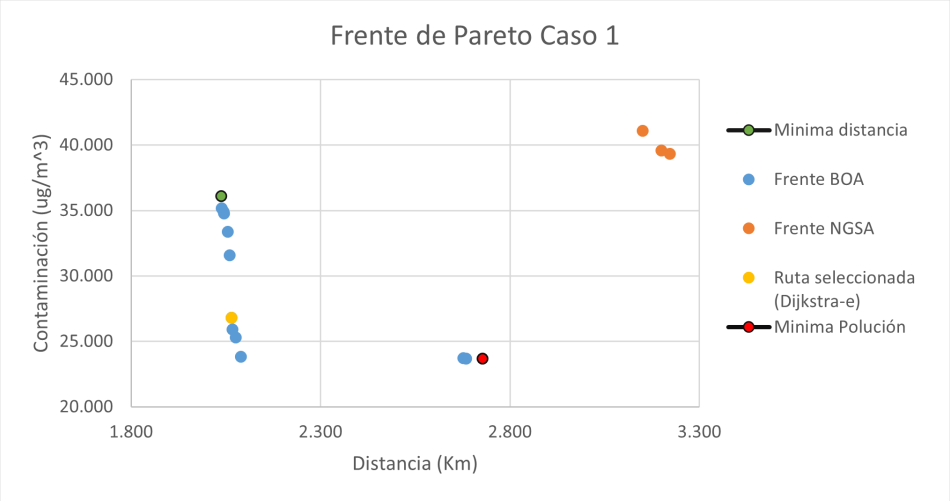

Figura 1: Conjunto de soluciones del Caso 1

3.2. Calidad de la solución

Distancia

Los resultados obtenidos en el caso multiobjetivo, se constrastaron con el mejor caso mono-objetivo obtenido, que fue el Dijsktra. Luego, para el caso 1, el Dijkstra-ϵ, siendo el mejor algoritmo, disminuyó la distancia que el ciclista tendrá que recorrer respecto al Dijkstra monobjetivo en un 23 %.

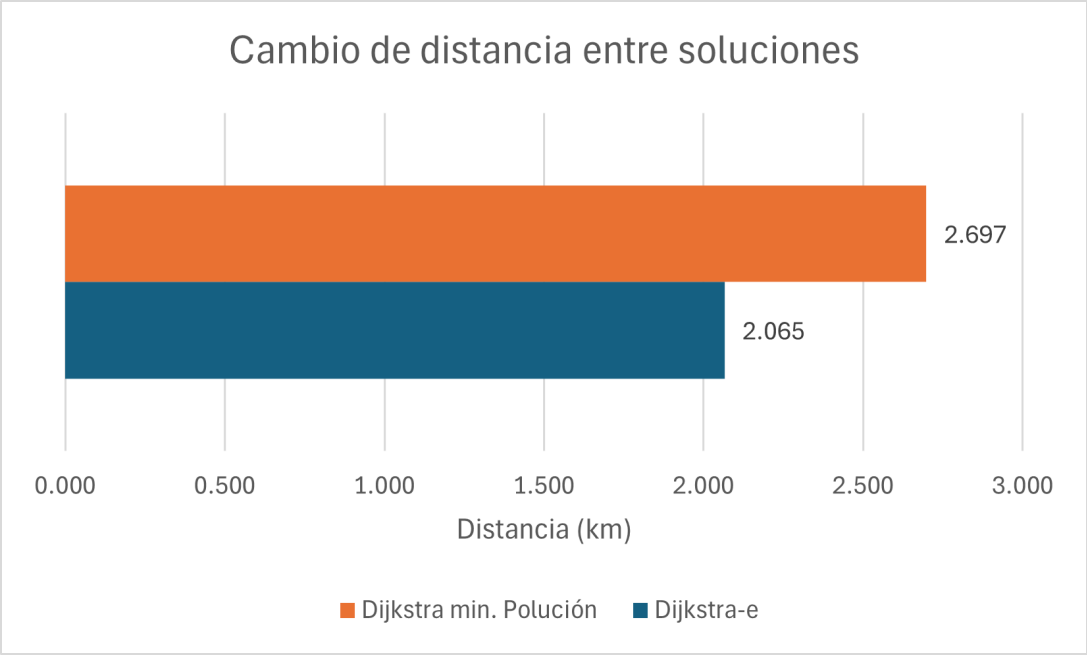

Figura 2: Cambio de distancia entre las mejores 2 soluciones objetivo único y multiobjetivo

Exposición a la contaminación Por otro lado, es posible evidenciar que el Dijkstra-ϵ aumentó la contaminación a la que el ciclista se va a exponer respecto al Dijkstra original en un 13 %.

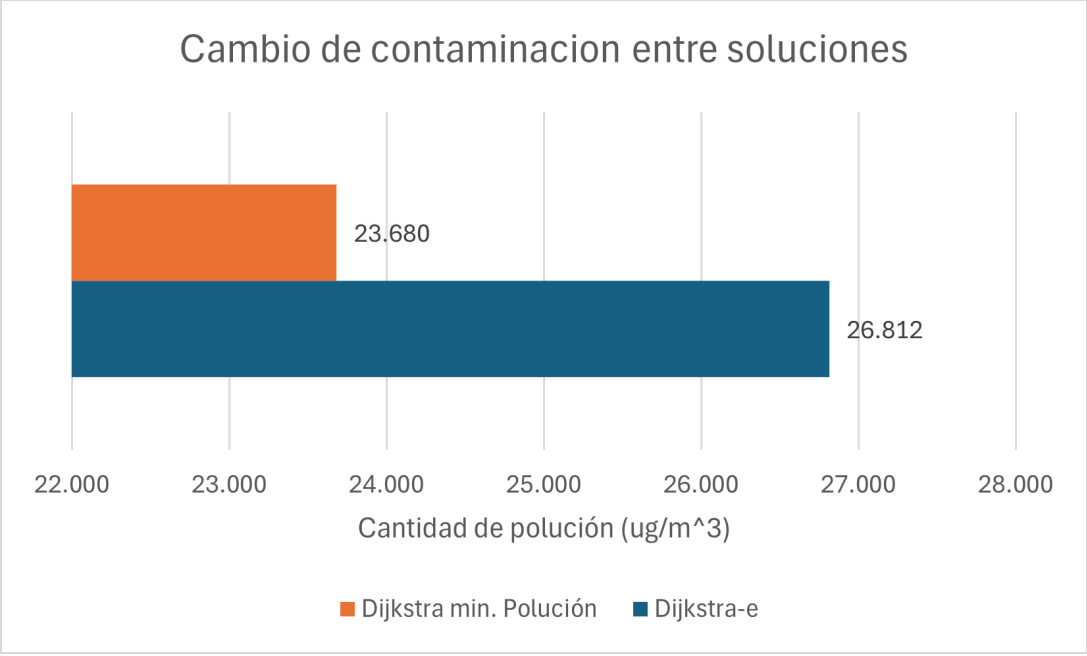

Figura 3: Cambio de contaminación entre las mejores 2 soluciones objetivo único y multiobjetivo.

Tiempo de ejecución
El Dijkstra-ϵ disminuyó en el tiempo de ejecucion respecto al Dijkstra original en un 5 %.

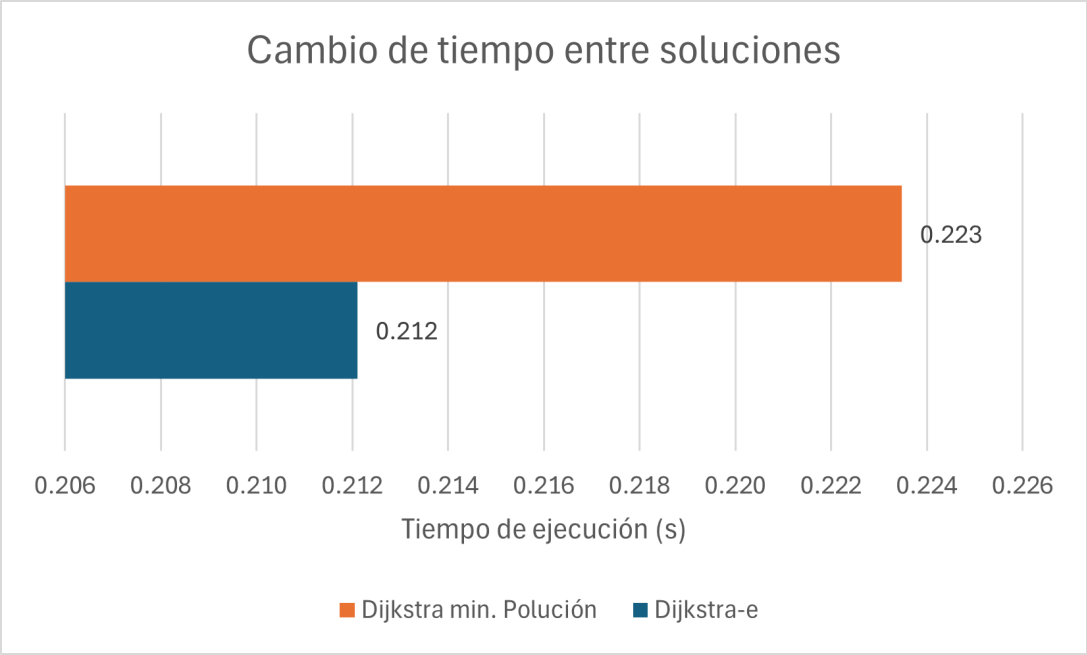

Figura 4: Cambio de tiempo de ejecución entre las mejores 2 soluciones objetivo único y multiobjetivo.

3.3. Caso 2

Para el segundo caso de estudio se seleccionaron nodos más distantes al caso anterior. De maenra similar, se consignaron los resultados con base a la clasificación (mejor/peor) hecha con SHANNON-TOPSIS en la tabla 2. Cabe resaltar que, los algoritmos ϵ-restringidos y el PULSE tuvieron un limite máximo de contaminación de 50 (ug/m^3).

Caso 2			
Nodo origen			
(-75.5636164, 6.2002022)			
Nodo destino			
(-75.5956039, 6.2453916)			
	Pesos Shannon		
Distancia (km)	0.1722		
Cantidad de polución (ug/mˆ3)	0.1506		
Tiempo de ejecución (s)	0.0143		
Algoritmos monobjetivo			
Algoritmo	**Distancia (km)**	**Cantidad de polución (ug/mˆ3)**	**Tiempo de ejecución (s)**
Dijkstra min. Polución	12.230	31.886	0.205
A* min. Polución	12.230	31.886	0.231
Dijkstra min. Distancia	7.231	94.053	0.264
A* min. Distancia	7.231	94.053	0.298
Algoritmos multiobjetivo			
BOA*	8.577	42.787	8.923
Pulse	7.863	49.899	23.355
Dijkstra-ϵ	9.702	49.503	0.225
A*-ϵ	9.702	49.503	0.258
NGSA-II	s.d	s.d	s.d
Promedios	9.346	55.446	4.220

La Figura 2 a continuación muestra la calidad de la solución en términos de distancia y contaminación del aire presente en la ruta para diferentes enfoques de solución MOSP caso 2. Contrario al caso anterior, se dibuja todo el frente de pareto para el algoritmo BOA.

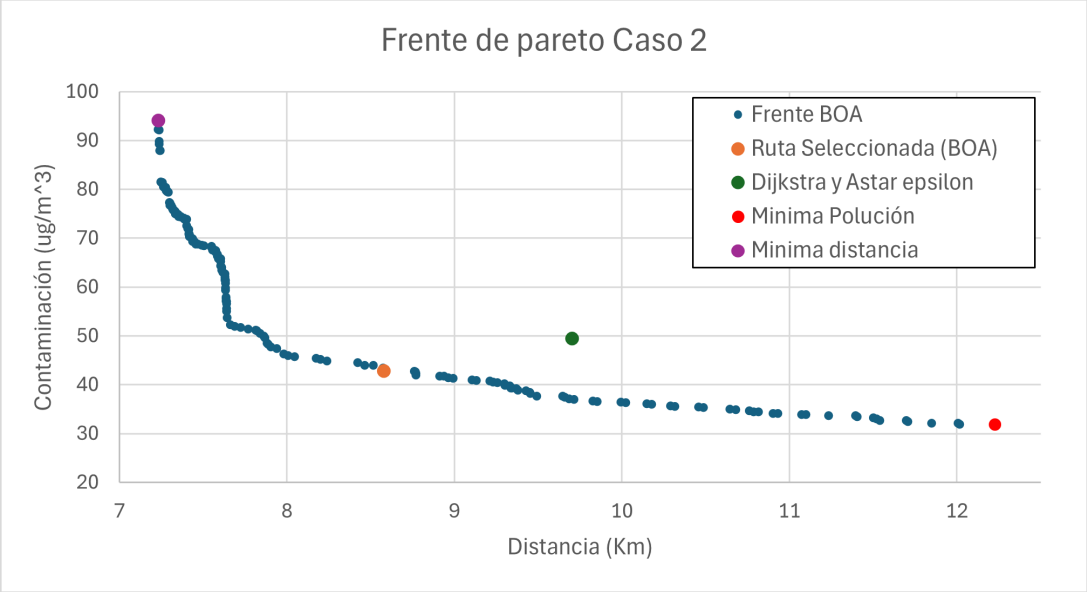

Figura 5: Conjunto de soluciones del Caso 2

3.4. Calidad de la solución

A continuación se muestran gráficos comparativos entre las mejores soluciones de objetivo único y multiobjetivo.

Distancia
BOA* disminuyo en la distancia respecto a Dijkstra en un 29 %.

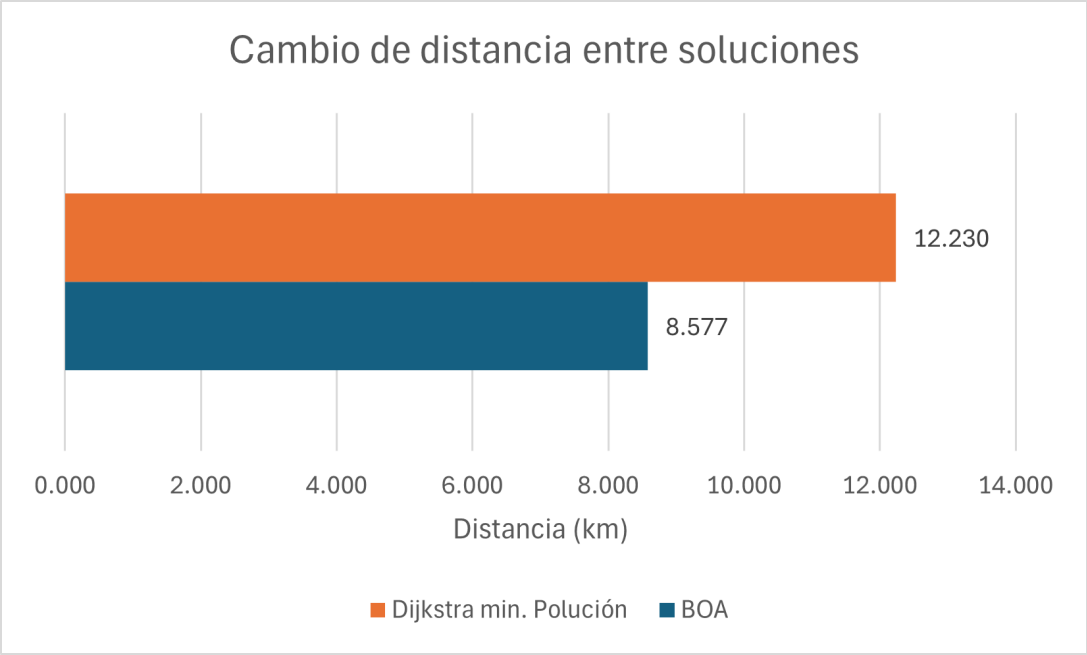

Figura 6: Cambio de distancia entre las mejores 2 soluciones objetivo único y multiobjetivo

Exposición a la contaminación
BOA* aumento en la contaminación respecto a Dijkstra en un 34 %.

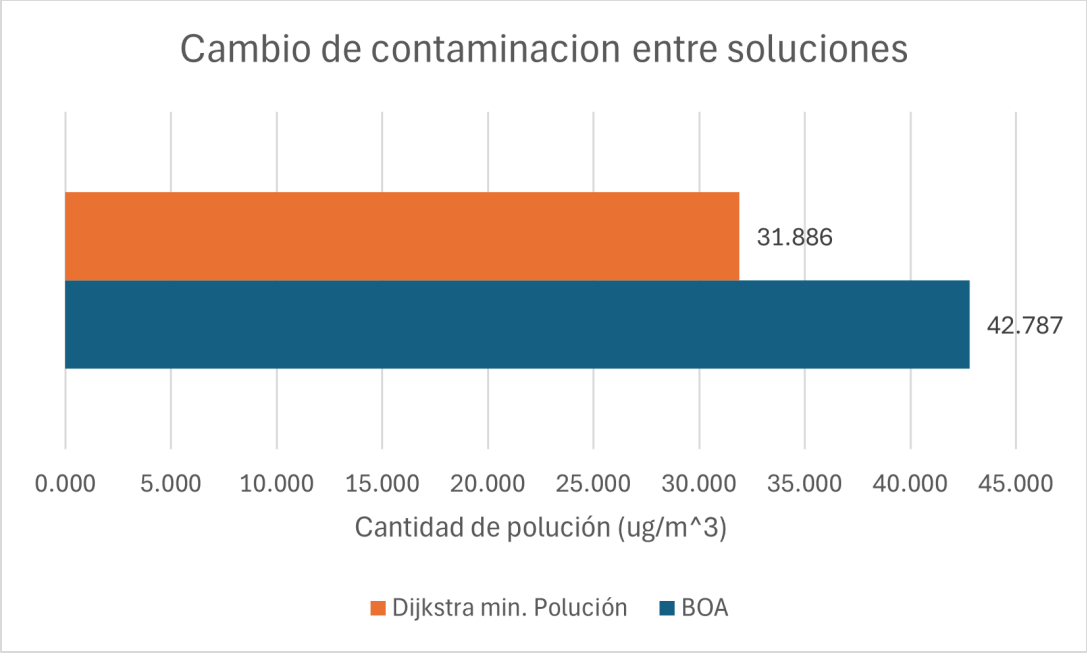

Figura 7: Cambio de contaminación entre las mejores 2 soluciones objetivo único y multiobjetivo.

Tiempo de ejecución

BOA* aumento en el tiempo de ejecución respecto a Dijkstra en un 4261 %.

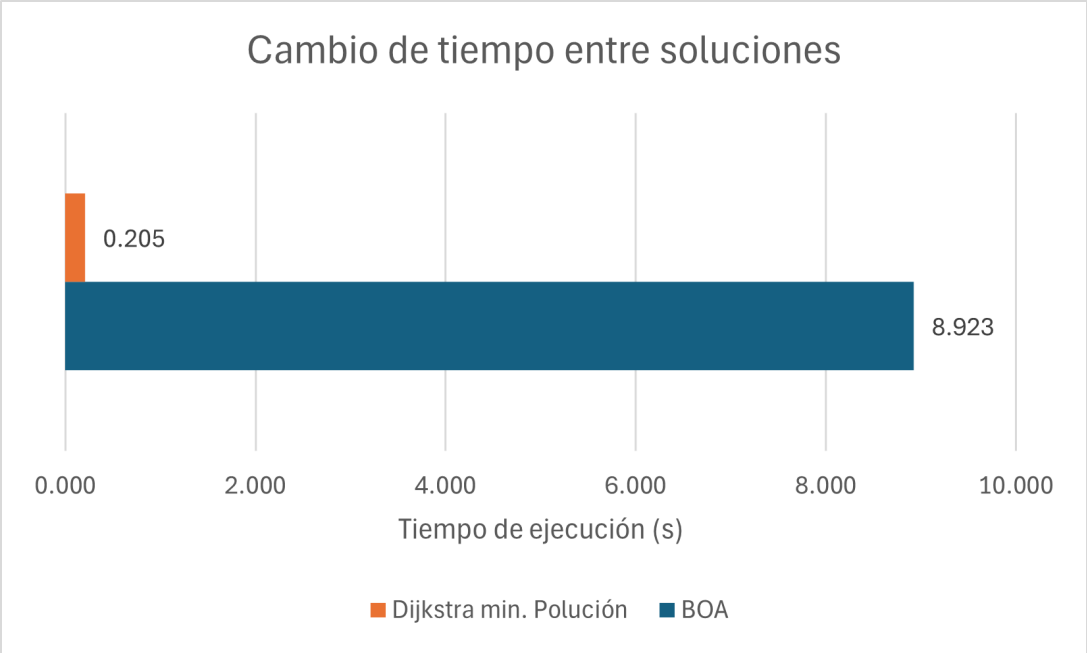

Figura 8: Cambio de tiempo de ejecución entre las mejores 2 soluciones objetivo único y multiobjetivo.

4. Discusión

- Los algoritmos que abordan el RSCSP presentan la desventaja de requerir que el decisor tenga conocimiento previo de los valores adecuados del recurso, como la contaminación en este caso. Existe el riesgo de elegir un límite demasiado bajo, lo que podría resultar en la inexistencia de una solución factible. Por otro lado, seleccionar un límite excesivamente alto podría llevar a pasar por alto caminos con un menor consumo de recursos, los cuales podrían ser de interés para el decisor.
- Dijkstra-ϵ y A*-ϵ son algoritmos cuyo rendimiento es cuestionable. Aunque su tiempo de ejecución no excede el de sus versiones originales, los caminos encontrados no son necesariamente no dominados, lo que puede llevar en ciertas circunstancias a la falta de una solución.
- Los algoritmos que resuelven el BOSP proporcionan más de una solución no dominada, lo que representa una ventaja al ofrecer al decisor múltiples opciones. El uso de Topsis puede ayudar al decisor a elegir entre estas soluciones, especialmente cuando el número de soluciones no dominadas aumenta exponencialmente con el tamaño del grafo. Además, la entropía de Shannon proporciona una forma rápida y sencilla de definir los pesos utilizados por Topsis, aunque es preferible que estos hayan sido previamente ajustados según las preferencias o necesidades del decisor.
- El algoritmo Pulse exhibe una ejecución relativamente lenta en grafos grandes, lo que puede comprometer su capacidad para encontrar una solución óptima para el SRCSP. Esto es evidenciable revisando los resultados de cada caso, estos consignados en las tablas 1 y 2. Este resultado es sorprendente, dado que en la literatura este tiempo es considerablemente inferior.

- El NGSA-II resulta incapaz de resolver problemas cuyos nodos estén a una distancia mayor a 2 kilómetros debido a su largo tiempo de ejecución. Por no mencionar que las soluciones que retorna no son no dominadas. Por eso no se presentan los datos asociados en la Figura 5., que es el caso 2. donde el nodo de inicio y el nodo de llegada estan lejos.
- En conclusión, consideramos que el BOA* es un algoritmo capaz de abordar eficazmente el problema planteado, siendo relativamente rápido y capaz de devolver múltiples soluciones, y como se mencionó anteriormente retorna más de una solución otorgandole más flexibilidad al usuario.

Referencias

[1] "World health organization - ambient (outdoor) air quality and health."

[2] "Encuesta: Medellín cómo vamos - valle de aburrá: entre prevenciones, alertas y retos por la gobernanza del aire."

[3] "Área metropolitana del valle de aburrá - condiciones especiales."

[4] "Área metropolitana del valle de aburrá - evaluación de la exposición de ciclistas a la contaminación del aire: una revisión de la literatura."

[5] IQAir, "World air quality."

[6] P. Hansen, "Bicriterion path problems," in *Multiple Criteria Decision Making Theory and Application* (G. Fandel and T. Gal, eds.), (Berlin, Heidelberg), pp. 109–127, Springer Berlin Heidelberg, 1980.

[7] A. Sedeño-noda and M. Colebrook, "A biobjective dijkstra algorithm," *European Journal of Operational Research*, vol. 276, no. 1, pp. 106–118, 2019.

[8] K. Miettinen, *Introduction to Multiobjective Optimization: Noninteractive Approaches*, pp. 1–26. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[9] L. Lozano and A. L. Medaglia, "On an exact method for the constrained shortest path problem," *Computers & Operations Research*, vol. 40, no. 1, pp. 378–384, 2013.

[10] O. Salzman, A. Felner, C. Hernandez, H. Zhang, S.-H. Chan, and S. Koenig, "Heuristic-search approaches for the multi-objective shortest-path problem: progress and research opportunities," in *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, IJCAI '23, 2023.

[11] P. Festa, "Constrained shortest path problems: state-of-the-art and recent advances," in *2015 17th International Conference on Transparent Optical Networks (ICTON)*, pp. 1–17, 2015.

[12] Y. P. Aneja and K. P. K. Nair, "The constrained shortest path problem," *Naval Research Logistics Quarterly*, vol. 25, no. 3, pp. 549–555, 1978.

[13] C. Hernández, W. Yeoh, J. A. Baier, H. Zhang, L. Suazo, S. Koenig, and O. Salzman, "Simple and efficient bi-objective search algorithms via fast dominance checks," *Artificial Intelligence*, vol. 314, p. 103807, 2023.