Recolección de materales peligrosos

[*]GitHub: https://github.com/aruzrojas/IAA-HAZMAT-COLLECTION-TS

Alexander Ruz Rojas
Departamento de Informática
Universidad Técnica Federico Santa María
Santiago, Chile
alexander.ruz@sansano.usm.cl

***Abstract*—En el presente artículo se presentarará el problema: 'Recolección de materiales peligrosos', donde se busca que estos materiales sean recogidos y transportados por un conjunto de camiones con un límite de carga cada uno. Dado que los materiales que se transportan están ubicados en lugares distintos y además tienen un costo de riesgo por ser peligrosos, lo que se busca es minimizar la distancia recorrida por cada camión y además en caso de accidente o situación similar, reducir la población afectada por dicha situación.**

***Index Terms*—Recolección, costo, minimizar, distancia recorrida, población afectada,**

I. Introducción

Muchas empresas requieren o trabajan con carga que desean transportar y contiene materiales que son nocivos, es decir, son perjudiciales para la sociedad y para el medio ambiente, si bien las empresas toman los resguardos necesarios para que el transporte de estos materiales sea lo más seguro posible, siempre existe la posibilidad [1] de que ocurra un accidente durante el transporte de la carga, lo que provocaría un gran impacto en la población expuesta a riesgos [2] como: incendio, explosión, fuga química o radiación de los materiales que estaban siendo transportados. A pesar de que el número de accidentes es muy bajo en relación a la cantidad de viajes que se realizan transportando estos materiales. Por esta misma razón, muchos investigadores comenzaron a desarrollar modelos matemáticos para poder identificar rutas que le permitan a cada vehículo transportar dichos materiales con el objetivo de que estas rutas sean de bajo riesgo. Para una empresa el costo de transporte en rutas de bajo riesgo resulta elevado, por lo cual este problema se vuelve multi-objetivo en donde se busca minimizar el costo asociado al transporte y reducir la población en riesgo, dando pesos distintos a cada objetivo dependiendo de la situación. Para darle aún más importancia al problema, estudios desde 1980 afirman [3] que en Estados Unidos la situación respecto al transporte de estos materiales se esta agravando, desde ese año más de 55.000 sustancias tóxicas fueron fabricados y procesados y al menos 250.000 envíos de materiales peligrosos se hacen diariamente y se espera que esto se duplique en al menos 10 años.

El resto del presente artículo, está organizado de la siguiente manera: la sección 2 contempla el estado del arte en donde se detallarán las técnicas que se han utilizado para resolver este problema, la sección 3 se determinará la técnica que se utilizó para resolver el problema (tanto el algoritmo como sus componentes), en la sección 4 se dan las características de las instancias utilizadas, como por ejemplo, cantidad de vehículos, cantidad de materiales peligrosos, etc, en la sección 5 se mostrarán los resultados obtenidos para las distintas instancias y en la última sección se dará a conocer la conclusión del artículo.

II. Estado del Arte

Los primeros estudios relacionados a recolección de materiales peligrosos datan a principios de la década del 80 [4], en un extensa investigación hecha por Gary L. Urbanek y Edward J. Barber, quienes desarrollan un criterio con el fin de designar rutas para el transporte de materiales peligrosos basado en la probabilidad de que ocurra un accidente en una ruta cualquiera, al año 1985, en el artículo [3] designan rutas para cada camión en base a 4 variables, las cuales son: el número de eventos peligrosos que ha ocurrido en una ruta, probabilidad de que el material provoque un accidente, población en el radio de riesgo y la cantidad de materiales peligrosos por cada tipo.

Al año 1997 [5] se utilizó un algoritmo llamado optimización por colonia de hormigas que fue diseñado por Marco Dorigo en el año 1992, quienes utilizaron este algoritmo fueron Richard F. Hartl y Christine Strauss para resolver el problema de enrutamiento de vehículos. Este problema es muy similar al visto en este artículo, en una breve descripción trata de que una flota de vehículos con capacidades iguales que deben realizar viajes desde una fábrica o lugar similar hacia distintos clientes recolectando distintos materiales, teniendo como objetivo solo reducir el costo de los viajes, ya sea el tiempo o el costo ecónomico involucrado. El poder resolver el VRP (problema de enrutamiento de vehículos) con el algoritmo optimización por colonia de hormigas fue un gran avance ya que los resultados obtenidos eran mucho mejor que utilizar búsqueda local y obtenían una diferencia de al menos entre 0-10% respecto a la mejor solución.

Para el año 2003 [6] se revisaron distintos modelos presentados para la resolución del problema de recolección de materiales peligrosos, los cuales representaban el valor esperado

de la consecuencia de que un camión cargado con materiales peligrosos escoja una ruta.

En el año 2010, los autores Rojee Pradhananga, Eiichi Taniguchia y Tadashi Yamada del artículo [1] "Ant colony system based routing and scheduling for hazardous material transportation", su enfoque es bastante similar al artículo "Applying the ANT System to theVehicle Routing Problem" [5], en donde utilizan el algoritmo basado en un sistema de colonia de hormigas para un problema multi-objetivo, a diferencia de VRP que solo era un problema de un único objetivo.

El algoritmo basado en un sistema de colonia de hormigas, es llamado asi por el comportamiento de las éstas en busca de su comida, al observarlas siempre están en una especie de hilera siguiendo un camino, esto se debe a que van dejando feromonas, que son sustancias químicas que simulan una firma para no perder el rastro por el que han pasado. Contextualizando el algoritmo para el problema de recolección de materiales, que es de tipo VRPTW multi-objetivo, es decir, problema de enrutamiento de vehículos con ventana de tiempo con más de un objetivo. La técnica que que emplea es construir una solución utilizando una población de hormigas y sus respectivas feromonas, donde estas últimas simulan el objetivo del problema que es reducir costos (ya sean económicos o de tiempo, etc.) y la población en riesgo. Cada hormiga realiza lo siguiente:

- Colocar la hormiga en el depósito, lugar de partida de los vehículos.
- Crear todas las rutas posibles desde el lugar actual (ya sea donde un cliente o nodo depósito).
- Seleccionar la ruta con el mayor valor de feromonas, las cuales fueron definidas en el paso anterior.
- Actualizar el valor de la feromona de la ruta actual.
- Actualizar el lugar en que se encuentra la hormiga (ya sea donde un cliente o nodo depósito).
- Repetir desde el segundo paso si es que se pueden atender a mas clientes.
- Actualizar la cantidad de vehículos disponibles.
- Terminar cuando se haya hecho lo mismo con todas las hormigas de la colonia.

Los pasos anteriores se deben repetir si es que aún no se alcanza el criterio de término, que puede ser un número de veces que se realizan estos pasos y cada vez que se terminen, se debe actualizar la solución general aplicando una búsqueda local de inserción a cada solución generada en los pasos anteiores y se actualiza el valor del rastro de feromona global (de la solución general).

Este algoritmo da muy buenos resultados muy cercanos a los óptimos globales, como se había mencionado el sistema de colonia de hormigas en un VRP presentaba soluciones entre un 0-10% de discrepancia respecto a las soluciones óptimas globales, para este caso que se trataba de un VRP con ventanas de tiempo multi-objetivo, presentaba soluciones entre un 0-10% de discrepancia respecto a los óptimos globales, pero la mayoría de los resultados tenía una diferencia entre 0-7%. Convergía a buenas soluciones en aproximadamente 1000 iteraciones, para una instancia de 50 clientes y utilizando 6 vehículos. Cabe destacar que en ninguna instancia utilizada se llegó al óptimo global.

En el año 2016 [2] Germán Paredes-Belmar, Andrés Bronfman, Vladimir Marianov, Guillermo Latorre-Núñez, propusieron una solución para este problema, en donde los camiones podían llevar distintos materiales siempre y cuando estos fueran compatibles entre sí, es decir, no había peligro alguno si se llevaban combinaciones de éstos en un mismo vehículo. Para resolver el problema lo que hacían era modelar el problema en un grafo, donde los nodos representan los clientes que buscan que su material sera recogido y transportado sumado al nodo depósito (punto de partida de los vehículos) y los arcos representan las rutas entre los nodos. Si un vehículo transportaba más de un material peligroso, consideraban solo el que tuviera mayor riesgo sobre la población.

El método que consideran es asumir un grafo auxiliar en donde todos los nodos estuvieran conectados entre sí, incluyendo al nodo depósito y consideran que cada arco tiene distintos pesos asociados a los distintos materiales, es decir, dependiendo del material que transporte el vehículo, se asocia el peso del arco, por lo tanto existen tantos grafos como tipos de materiales para el problema. Como consideran que minimizan la población en riesgo y el costo, el peso de los arcos ya no solo sería el que tan expuesta esta la población sino que se le agrega el costo, que correspondería al costo de viajar de un nodo i (un cliente cualquiera) a un nodo j (cliente cualquiera) utilizando ese arco. De la forma en que en este artículo relacionan ambos parámetros es la siguiente:

$$Z = \alpha \cdot EP + (1 - \alpha) \cdot C$$

Donde EP, corresponde a la población expuesta y C al costo, $\alpha \in$ [0,1] y corresponde al peso que se le asigna a la población expuesta, es decir, si se considera más o menos su valor frente al costo.

Por lo tanto para cada arco se obtiene un peso que relaciona la población expuesta y el costo de viajar de un cliente a otro, o del nodo depósito a un cliente (es importante destacar que en este caso no hay población en riesgo, porque no hay material cargado en el vehículo) o de un cliente al nodo depósito. Hecho lo anterior se resuelve el problema minimizando el valor de Z y cumpliendo restricciones, entre las que se destacan:

- No sobrepasar la capacidad de cada vehículo.
- Cada cliente es atendido por un solo vehículo.
- Evitar materiales incompatibles en el mismo vehículo.
- Evitar hacer ciclos entre los viajes del vehículo.

La métrica que utilizan para decidir que arco agregar a la ruta de cada vehículo es añadir el arco que menos contribuya al valor de Z, ya que se quiere minimizar, una vez resuelta la ruta para un vehículo, se actualizan los viajes ya que en la ruta se pudo haber agregado un arco que no exisitiera en el grafo original y dicho arco se actualiza con los arcos disponibles del grafo original.

III. Técnica propuesta

Antes de detallar la resolución del problema, se definirán algunos parámetros y variables:

K: Conjunto de vehículos.
N: Conjunto de clientes incluyendo el nodo depósito.
N_c: Conjunto de clientes sin el nodo depósito.
M: Conjunto con los tipos de materiales peligrosos.
MC: Conjunto de cantidad de materiales peligrosos, donde el i-ésimo elemento corresponde a la cantidad que generó el cliente $n_i \in N$.
T: Conjunto del par: (nodo cliente, tipo de material peligroso), donde el nodo cliente genera un tipo de material.
A: Conjunto de arcos (rutas entre los nodos).
COM: Conjunto de pares de materiales peligrosos del tipo (m,r) que son compatibles entre sí, es decir, un vehículo puede transportar en la misma carga dichos materiales.
D_0: Lista de distancia desde el nodo depósito hacia los clientes.
D_m: Matriz de distancia entre los distintos nodos (clientes), si el vehículo transporta el material $m \in M$.
R_m: Matriz de riesgo entre los distintos nodos (clientes), si el vehículo transporta el material $m \in M$.
Q: Capacidad del vehículo.

Para la resolución de este problema se utilizará el algoritmo Tabu Search, el cual se caracteriza por ser una estrategia de búsqueda local, en donde es posible generar distintas soluciones a partir de una, lo que se conoce como vecindario de una solución. A través del uso de estructuras de memoria es posible aumentar el rendimiento de la búsqueda y en dicha estructura, se almacenan los movimientos realizados para llegar a una nueva solución, para este problema los movimientos que se utilizaron fueron:

- 2-Intercambio: De dos rutas de vehículos, se intercambian clientes que estén incluídos en sus rutas.
- Insert: De la ruta de un camión escoger un cliente, borrarlo de su ruta e insertarlo en la ruta de otro camión.

El objetivo de utilizar la estructura de memoria, conocida como Lista Tabú es que los movimientos realizados no se repitan y así la búsqueda de nuevas soluciones no se vea afectada. Para generar el espacio de soluciones (vecindario), se necesita una solución inicial la cual será generada a partir del algoritmo Greedy, el cual se caracteriza por ser una estrategia de búsqueda local en donde en cada paso escoge la opción de óptimo local, es decir, busca el beneficio del momento, sin importar lo que ocurrió o lo que puede ocurrir. Los componentes necesarios de Greedy para la resolución de este problema son:

- Representación de la solución: Matriz W de 4 dimensiones, que almacena el vehículo utilizado, material que carga, y los arcos utilizados, es decir:

$$w_{ij}^{km} = \begin{cases} 1, \text{ Si el vehículo } k \in K \text{ transporta el} \\ \text{material } m \in M \text{ entre los nodos } i,\ j \in N \\ 0, \text{ en otro caso} \end{cases}$$

- Función Miope: Para cada vehículo escoger el nodo que menos contribuye a la función evaluación y que los materiales que carga sean compatibles entre si.
- Punto de partida: Todos los vehículos parten del nodo depósito.
- Función de evaluación: Minimizar la relación entre costo y población expuesta mediante la siguiente expresión.

$$\text{Min } Z = \sum_{k \in K}^{N} \sum_{m \in M}^{N} \sum_{i,j \in A}^{N} a_{ij}^{mk} \cdot w_{ij}^{mk} \qquad (1)$$

Donde $a_{ij}^{mk} = \alpha \cdot EP_{ij}^{m} + (1 - \alpha) \cdot c_{ij}$

Con:
EP_{ij}^{M}: Cantidad de población en riesgo expuesta al material m entre los nodos i, j.
c_{ij}: Costo de transporte desde el nodo i al nodo j.
$\alpha \in [0, 1]$
Además de la variable w_{ij}^{km}, se definen dos variables m más las cuales son:

$$x_{ij}^{k} = \begin{cases} 1, \text{ Si el vehículo } k \in K \text{ viaja desde el nodo} \\ i \text{ al nodo } j;\ i,\ j \in N \\ 0, \text{ en otro caso} \end{cases}$$

$$y_{i}^{k} = \begin{cases} 1, \text{ Si el vehículo } k \in K \text{ carga el producto del} \\ \text{cliente } i \in N_c \\ 0, \text{ en otro caso} \end{cases}$$

C_i^k = Cantidad de material peligroso que lleva el vehículo $k \in K$ antes de visitar al nodo $i \in N$. Si se está en el nodo depósito, C_i^k = 0.

Para el algoritmo Tabu Search, los componentes necesarios son:

- Lista tabú: Estructura de memoria que almacena los movimientos realizados de cada solución encontrada, con el fin de no repetir los movimientos. Se debe definir el tamaño de la Lista y cada vez que se llene, se debe eliminar el primer movimiento hecho de los que se encuentran en la Lista.
- Función de evaluación: Minimizar la relación entre costo y población expuesta (1).

Restriciones del problema:

Restricción 1: No se puede superar la capacidad de carga de los vehículos.

$$\sum_{i \in N_c} y_i^k \cdot MC_i \leq Q \qquad \forall k \in K \qquad (2)$$

Restricción 2: Cada cliente que generó un material debe ser atendido por solo un vehículo, es decir, toda la carga se la lleva un solo vehículo.

$$\sum_{k \in K} y_i^k = 1 \qquad \forall i \in N \tag{3}$$

Restricción 3: El vehículo puede ser o no ser utilizado.

$$\sum_{j \in N_c} x_{0j}^k \leq 1 \quad \forall k \in K, \text{ 0 representa el nodo depósito.} \tag{4}$$

Restricción 4: Todos los vehículos que entren a un nodo cliente, deben salir de éste.

$$\sum_{i \in N_c} x_{ij}^k = \sum_{h \in N_c} x_{jh}^k \qquad \forall k \in K, \forall j \in N_c \tag{5}$$

Restricción 5: Un vehículo con carga de un material tipo r, no puede salir del nodo i (nodo actual) con un riesgo menor a r.

$$\sum_{j \in N} w_{ij}^{km} \leq 1 - y_i^k \quad \forall k \in K, \forall i \in N_c, \ m, \ r \in M, (i,r) \in T \tag{6}$$

Restricción 6: Se evita la combinacion de materiales incompatibles en el mismo vehículo.

$$y_i^k + y_j^k \leq 1 \ \ \forall k \in K, \forall i \in N_c, (i,m), \ (j,r) \in T, (m,r) \in COM \tag{7}$$

Restricción 7: Se registra el volumen de materiales después de que el vehículo haya pasado y recolectado los materiales de un nodo cliente y además se evita que el vehículo quede atrapado en un sub-tour (ciclos del grafo).

$$C_j^k \geq C_i^k + MC_i - Q \cdot (1 - x_{ij}^k) \quad \forall k \in K, i \in N, j \in N_c \tag{8}$$

Naturaleza de las variables:

$$x_{ij}^k, y_i^k, w_{ij}^{km} \in \{0,1\} \qquad \forall k \in K, \ i,j \in N, m \in M \tag{9}$$

Ya teniendo definido los parámetros, variables, restricciones y componentes para el algoritmo Greedy, se detallará el cómo se utiliza el algoritmo en el siguiente pseudo-código:

Algorithm 1 Algoritmo Greedy

```
1:  while (Clientes con carga) do
2:    for (k ∈ K) do
3:      min ← min(A)
4:      i,j ← min
5:      if (cliente j no descargado) then
6:        if (COMP(Materiales k, material j) then
7:          Visitar y descargar cliente j.
8:          Actualizar función objetivo
9:        end if
10:     end if
11:   end for
12: end while=0
```

El algoritmo anterior lo que realiza es que por cada vehículo recorre todos los nodos (clientes) posibles, siempre verificando las restricciones y sobre todo si es que el material a cargar es compatible con los materiales que ha cargado el vehículo, en caso de que cumpla las restricciones, es posible cargar dicho material en el vehículo hasta que regresa al nodo depósito y se realiza lo mismo con el siguiente vehículo.

Una vez generada la solución inicial utilizando Greedy, se deben definir los parámetros a utilizar del algoritmo Tabu Search para luego utilizarlo, se adjunta un pseudo-código a continuación:

Algorithm 2 Algoritmo Tabu Search

```
1:  ListaTabu ← []
2:  max_size ← Tamaño máximo lista tabú
3:  sBest ← Greedy()
4:  sbestCandidate ← Greedy()
5:  while (Criterio de término) do
6:    vecindario ← vecindario(sbestCandidate)
7:    sbestCandidate ← vecindario.firstElement
8:    for (sCandidate in vecindario) do
9:      if (ListaTabu no contiene sCandidate and
        eval(sCandidate) <= eval(sbestCandidate) then
10:       sbestCandidate ← sCandidate
11:     end if
12:   end for
13:   if (eval(sbestCandidate) <= eval(sBest) then
14:     sBest ← sbestCandidate
15:   end if
16:   ListaTabu.push(Movimiento(sbestCandidate)
17:   if (ListaTabu.size > max_size) then
18:     ListaTabu.remove(first)
19:   end if
20: end while
    =0
```

El algoritmo anterior, al comienzo inicializa los parámetros del problema, la estructura de memoria (ListaTabu), el tamaño máximo de la ListaTabu y se inicializa la solución inicial que se almacenará en las variables sBest y sbestCandidate, la primera mencionada se utilizará para guardar el valor de la mejor solución y la segunda se utilizará para guardar el valor de la mejor solución del vecindario y este último será generado a partir de sbestCandidate. Del vecindario se escoge la mejor solución que cumpla con la restricción de que el movimiento para llegar a dicha solución no se encuentre en la ListaTabu, este valor se almacena en sbestCandidate y si su valor es de mejor calidad que sBest, ésta última se actualiza. En la ListaTabu se agrega el movimiento realizado y por cada iteración se debe verificar si su tamaño no ha sobrepasado el max_size, en caso contrario se elimina el primer movimiento hecho de la ListaTabu. Después de una cantidad de iteraciones, se retorna el valor de sBest. Para una mejor comparación de los algoritmos, se almacenarán los valores de la función objetivo de los algoritmos:

- Algoritmo Greedy
- Algoritmo Tabu-Search con único movimiento (2-Intercambio).

- Algoritmo Tabu-Search con único movimiento (Insert).
- Algoritmo Tabu-Search con ambos movimientos (2-Intercambio e Insert).

IV. Escenario experimental

La ejecución del algoritmo fue realizada en un computador con un procesador AMD Ryzen 52500U (8 CPUs) 2 GHZ, Memoria Ram 16 GB y Sistema Operativo Windows 10, el algoritmo fue programado en C++. Lo primero que se realizó es leer el archivo para obtener los datos necesarios para la resolución del problema. Se utilizan estructuras para los nodos y los vehículos, por cada nodo (cliente), se almacenaban los datos de su id, tipo de material que producen y la carga que producen del material. Para el vehículo en su estrucutra se almacena la capacidad actual, el nodo actual en que se encuentra y la ruta que ha hecho. En matrices se va almacenando la distancia entre todos los nodos y el riesgo de transitar entre todos los nodos para cada material. Se define la matriz W, que es la representación de la solución del problema, la cual se va construyendo utilizando el algoritmo Greedy con la técnica descrita en Algorithm 1.

Para el algoritmo Tabu-Search (Algorithm 2), se utiliza como Lista Tabú un vector de tamańo máximo 7 y como criterio de término realizar 100 iteraciones. Para la elección del movimiento (2-Intercambio o Insert) se genera un número al azar entre 0 y 1 a partir de una semilla, si dicho número es menor a 0.5, se usa el movimiento Insert, en caso contrario se utiliza 2-Intercambio. Para una mejor comparación en la sección resultados se detallarán los valores obtenidos utilizando cada movimento por separado y la combinación de ambos, además de los resultados utilizando Greedy.

Las instancias que se utilizaron para obtener resultados del algoritmo implementado son las mismas que se utilizaron en el artiículo "Hazardous materials collection with multiple-product loading" [2]. Todas las instancias tienen 5 tipos de materiales, distinta cantidad de vehículos pero todos con la misma capacidad. Las características de estos son las siguientes:

TABLE I
DATOS DE CANTIDAD DE MATERIALES POR TIPO EN CADA ZONA.

Instancias	# MC_A	# MC_B	# MC_C	# MC_D	# MC_E
Zona 1	7.640	6.680	3.040	4.260	5.720
Zona 2	5.440	10.890	14.420	20.540	11.740
Zona 3	8.040	2.170	4.170	4.120	1.970
Zona 4	12.440	1.840	3.560	5.940	3.390
Zona 5	1.560	4.370	1.820	2.760	2.810
Zona 6	5.600	3.270	6.060	2.310	3.770
Zona 7	3.840	2.490	2.170	4.980	2.230

TABLE II
DATOS DE CANTIDAD DE CLIENTES Y VEHÍCULOS DISPONIBLES.

Instancias	# clientes[a]	# vehículos disponibles
Zona 1	32	3
Zona 2	36	3
Zona 3	15	3
Zona 4	30	5
Zona 5	21	2
Zona 6	22	5
Zona 7	11	4

[a]No se considera el nodo depósito.

TABLE III
MATRIZ COM: COMPATIBILIDAD ENTRE LOS LOS MATERIALES Y TIPO DOMINANTE.

Tipo	A	B	C	D	E
A	A	-	C	D	-
B	-	B	C	D	E
C	C	C	C	-	E
D	D	D	-	D	E
E	-	E	E	E	E

Las tablas I y II muestran las características de cada instancia, donde cada zona representa una instancia, es decir, un archivo distinto. Las características a destacar son la cantidad de material peligroso de cada tipo, cantidad de clientes y vehículos disponibles de cada instancia, notar que todos los vehículos tienen la misma capacidad que es de 40.000. La instancia también considera las distancias desde el nodo depósito hacia todos los otros nodos (clientes) y matrices de distancia entre los nodos y riesgos entre estos mismos, dependiendo del material que se transporta. Cabe destacar que por cada tipo de material existe una matriz de distancia, ya que el vehículo debe tomar rutas distintas dependiendo del material cargado.

La tabla III muestra la compatibilidad entre los tipos de materiales, es decir, cada tipo de material puede ser o no compatible respecto a los otros. En el caso de que un vehículo cargue más de un tipo de material que sean compatibles entre si, se considera un solo riesgo para efectos de la resolución.

V. Resultados obtenidos

TABLE IV
Resultados Zona 1

ZONA 1												
α	**Greedy**			**2-Int**			**Insert**			**2-Int + Insert**		
	EP	Costo	Total	EP	Costo	Total	EP	Costo	Total	EP	Costo	Total
0	640669	242720	242720	640669	242720	242720	627258	195031	195031	588399	192931	192931
0.1	656606	221273	264806	647871	219634	262458	647871	219633	262457	647871	219631	262455
0.2	656606	215889	304032	647871	214250	300974	647871	214249	300973	647871	214248	300972
0.3	656606	210504	344335	647871	208865	340567	647871	208864	340566	647871	208862	340565
0.4	656606	205120	385715	647871	203481	381237	644594	205531	381156	644594	205531	381156
0.5	635809	208605	422207	628695	205345	417020	602916	198123	400520	598254	202125	400190
0.6	635809	203221	462774	623636	206086	456616	602916	192739	438845	602916	192739	438845
0.7	621102	183481	489816	608929	186346	482154	588209	172999	463646	583547	177001	461583
0.8	626212	183418	537653	617477	181779	530338	616188	169940	526938	616188	169940	526938
0.9	626033	174029	580833	617298	172390	572807	617270	172532	572796	616225	165444	571147
1	626033	168645	626033	617240	175815	617240	617240	175815	617240	616225	160060	616225

TABLE V
Resultados Zona 2

ZONA 2												
α	**Greedy**			**2-Int**			**Insert**			**2-Int + Insert**		
	EP	Costo	Total	EP	Costo	Total	EP	Costo	Total	EP	Costo	Total
0	983578	137617	137617	$1{,}0 \cdot 10^6$	119387	119387	965343	111701	111701	965343	111701	111701
0.1	$1{,}0 \cdot 10^6$	141263	233959	$1{,}1 \cdot 10^6$	117100	214547	$1{,}0 \cdot 10^6$	111846	201989	896730	106601	185614
0.2	968482	137367	303590	964682	136351	302017	888992	99218.2	257173	888992	99219.2	257174
0.3	968482	132746	383467	964682	131730	381616	888992	94597.3	332916	889665	95060.3	333442
0.4	959117	144406	470291	949937	147868	468696	839291	110091	401771	839291	110092	401772
0.5	909115	99907.5	504511	906417	99571.5	502994	875436	97533.5	486485	874911	97100.5	486006
0.6	909115	95286.6	583584	906417	94950.6	581830	875436	92912.6	562427	874911	92479.6	561938
0.7	909115	90665.7	663580	906417	90329.7	661591	875436	88291.7	639293	874911	87858.7	638795
0.8	905047	86772.8	741392	902349	86436.8	739167	852866	81983.8	698690	851800	81486.8	697737
0.9	905047	82151.9	822758	902349	81815.9	820296	852866	77361.9	775316	851800	76865.9	774307
1	905047	77531	905047	902349	77195	902349	852866	72741	852866	851800	72245	851800

TABLE VI
Resultados Zona 3

ZONA 3												
α	**Greedy**			**2-Int**			**Insert**			**2-Int + Insert**		
	EP	Costo	Total	EP	Costo	Total	EP	Costo	Total	EP	Costo	Total
0	295254	53459	53459	295254	53459	53459	295254	53459	53459	295254	53459	53459
0.1	307573	52514.5	78020.4	307573	52514.5	78020.4	307573	52514.5	78020.4	307573	52514.5	78020.4
0.2	307573	49154	100838	307573	49154	100838	307573	49154	100838	307573	49154	100838
0.3	376259	50087.5	147939	376259	50087.5	147939	376259	50087.5	147939	376259	50087.5	147939
0.4	376259	46727	178540	376259	46727	178540	376259	46727	178540	376259	46727	178540
0.5	376259	43366.5	209813	376259	43366.5	209813	376259	43366.5	209813	376259	43366.5	209813
0.6	283134	38688.2	185356	283134	38688.2	185356	283134	38688.2	185356	283134	38688.2	185356
0.7	283134	35352.4	208800	283134	35352.4	208800	283134	35352.4	208800	283134	35352.4	208800
0.8	283134	32016.6	232911	283134	32016.6	232911	283134	32016.6	232911	283134	32016.6	232911
0.9	283134	28680.8	257689	283134	28680.8	257689	283134	28680.8	257689	283134	28680.8	257689
1	283134	25345	283134	283134	25345	283134	283134	25345	283134	283134	25345	283134

TABLE VII
RESULTADOS ZONA 4

ZONA 4												
α	**Greedy**			**2-Int**			**Insert**			**2-Int + Insert**		
	EP	Costo	Total	EP	Costo	Total	EP	Costo	Total	EP	Costo	Total
0	1,4 $\cdot 10^6$	139369	139369	1,4 $\cdot 10^6$	136375	136375	1,3 $\cdot 10^6$	128370	128370	1,3 $\cdot 10^6$	128370	128370
0.1	1,4 $\cdot 10^6$	131484	255452	1,2 $\cdot 10^6$	128555	240111	1,3 $\cdot 10^6$	126196	239434	1,3 $\cdot 10^6$	126966	241675
0.2	1,4 $\cdot 10^6$	125856	390188	1,2 $\cdot 10^6$	122772	348126	1,2 $\cdot 10^6$	122100	335559	1,2 $\cdot 10^6$	121688	334192
0.3	1,4 $\cdot 10^6$	118985	517543	1,2 $\cdot 10^6$	115901	455992	1,2 $\cdot 10^6$	115229	437478	1,2 $\cdot 10^6$	115165	435730
0.4	1,4 $\cdot 10^6$	112113	646273	1,2 $\cdot 10^6$	109029	565233	1,2 $\cdot 10^6$	108357	540772	1,2 $\cdot 10^6$	106825	542804
0.5	1,4 $\cdot 10^6$	105241	776378	1,2 $\cdot 10^6$	102157	675848	1,2 $\cdot 10^6$	101833	645368	1,2 $\cdot 10^6$	104855	657642
0.6	1,4 $\cdot 10^6$	98369.2	907856	1,2 $\cdot 10^6$	95285.2	787838	1,2 $\cdot 10^6$	94961.2	751326	1,2 $\cdot 10^6$	97983.2	765451
0.7	1,4 $\cdot 10^6$	88179.4	995397	1,3 $\cdot 10^6$	85843.4	947243	1,3 $\cdot 10^6$	99810.4	903475	1,2 $\cdot 10^6$	83517.4	883598
0.8	1,4 $\cdot 10^6$	77939.6	1,1 $\cdot 10^6$	1,3 $\cdot 10^6$	81455.6	1,0 $\cdot 10^6$	1,2 $\cdot 10^6$	78147.6	959399	1,1 $\cdot 10^6$	77673.6	955070
0.9	1,3 $\cdot 10^6$	73141.8	1,1 $\cdot 10^6$	1,2 $\cdot 10^6$	72927.8	1,1 $\cdot 10^6$	1,1 $\cdot 10^6$	70782.8	1,0 $\cdot 10^6$	1,1 $\cdot 10^6$	70370.8	1,0 $\cdot 10^6$
1	1,4 $\cdot 10^6$	70998	1,4 $\cdot 10^6$	1,3 $\cdot 10^6$	69436	1,3 $\cdot 10^6$	1,2 $\cdot 10^6$	78264	1,2 $\cdot 10^6$	1,1$\cdot 10^6$	76681	1,1 $\cdot 10^6$

TABLE VIII
RESULTADOS ZONA 5

ZONA 5												
α	**Greedy**			**2-Int**			**Insert**			**2-Int + Insert**		
	EP	Costo	Total	EP	Costo	Total	EP	Costo	Total	EP	Costo	Total
0	612229	73258	73258	612229	73258	73258	612229	73258	73258	612229	73258	73258
0.1	613232	72560.4	126628	613232	72560.4	126628	613232	72560.4	126628	613232	72560.4	126628
0.2	613232	70524.8	179066	613232	70524.8	179066	613232	70524.8	179066	613232	70524.8	179066
0.3	613232	68489.2	231912	613232	68489.2	231912	613232	68489.2	231912	613232	68489.2	231912
0.4	613232	66453.6	285165	613232	66453.6	285165	613232	66453.6	285165	613232	66453.6	285165
0.5	613232	64418	338825	613232	64418	338825	613232	64418	338825	613232	64418	338825
0.6	613232	62382.4	392892	613232	62382.4	392892	613232	62382.4	392892	613232	62382.4	392892
0.7	613232	60346.8	447366	613232	60346.8	447366	613232	60346.8	447366	613232	60346.8	447366
0.8	613232	58311.2	502248	613232	58311.2	502248	613232	58311.2	502248	613232	58311.2	502248
0.9	613232	56275.6	557536	613232	56275.6	557536	613232	56275.6	557536	613232	56275.6	557536
1	613232	54240	613232	613232	54240	613232	613232	54240	613232	613232	54240	613232

TABLE IX
RESULTADOS ZONA 6

ZONA 6												
α	**Greedy**			**2-Int**			**Insert**			**2-Int + Insert**		
	EP	Costo	Total	EP	Costo	Total	EP	Costo	Total	EP	Costo	Total
0	553072	93187	93187	553072	93187	93187	546091	92999	92999	561101	93142	93142
0.1	553072	89832.1	136156	553072	89832.1	136156	540931	89645.1	134774	547912	89833.1	135641
0.2	553072	86477.2	179796	553072	86477.2	179796	540931	86290.2	177218	547912	86478.2	178765
0.3	553072	83122.3	224107	553072	83122.3	224107	540931	82935.3	220334	553072	83122.3	224107
0.4	535018	97438.4	272470	526857	96166.4	268443	514836	93591.4	262089	492480	95680.4	254400
0.5	535018	94083.5	314551	526857	92811.5	309834	514836	90236.5	302536	492480	92325.5	292403
0.6	535018	90728.6	357302	526857	89456.6	351897	514836	86881.6	343654	492480	88970.6	331076
0.7	438473	72084.7	328556	428039	71390.7	321044	356687	70879.7	270945	355494	73165.7	270796
0.8	438473	68729.8	364524	428039	68035.8	356038	356687	67524.8	298855	355494	69810.8	298357
0.9	438473	65374.9	401163	428039	64680.9	391703	356687	64169.9	327435	355494	66455.9	326590
1	438473	62020	438473	428039	61326	428039	356687	60815	356687	355494	63101	355494

TABLE X
RESULTADOS ZONA 7

ZONA 7												
α	**Greedy**			**2-Int**			**Insert**			**2-Int + Insert**		
	EP	Costo	Total	EP	Costo	Total	EP	Costo	Total	EP	Costo	Total
0	199788	68050	68050	199788	68050	68050	199788	68050	68050	199788	68050	68050
0.1	235395	87343.8	102149	235395	87343.8	102149	235395	87343.8	102149	235395	87343.8	102149
0.2	235395	86302.6	116121	235395	86302.6	116121	235395	86302.6	116121	235395	86302.6	116121
0.3	235395	85261.4	130301	235395	85261.4	130301	235395	85261.4	130301	235395	85261.4	130301
0.4	235395	84220.2	144690	235395	84220.2	144690	235395	84220.2	144690	235395	84220.2	144690
0.5	235395	83179	159287	235395	83179	159287	235395	83179	159287	235395	83179	159287
0.6	207805	75433.8	154857	207805	75433.8	154857	207805	75433.8	154857	207805	75433.8	154857
0.7	207805	74392.6	167781	207805	74392.6	167781	207805	74392.6	167781	207805	74392.6	167781
0.8	207805	73351.4	180914	207805	73351.4	180914	207805	73351.4	180914	207805	73351.4	180914
0.9	207805	72310.2	194256	207805	72310.2	194256	207805	72310.2	194256	207805	72310.2	194256
1	191883	68602	191883	191883	68602	191883	191883	68602	191883	191883	68602	191883

Cada tabla representa una instancia utilizada, es decir una zona, y por cada una de ellas se especifican los valores de alfa, el algoritmo utilizado, los objetivos (población expuesta y costo) y el total que equivale al valor de la función objetivo (1). Los algoritmos mostrados en cada tabla son Greedy, Tabu-search con movimiento 2-Intercambio, Tabu-search con movimiento Insert y Tabu search con ambos movimientos.

VI. Conclusiones

En comparación de los resultados de los algoritmos Greedy y Tabu-search, se puede afirmar que en todas las instancias con Tabu search, utilizando cualquier movimiento o la combinación de estos, el resultado que se obtiene es igual o mejor que lo obtenido usando Greedy, por lo que los movimientos propuestos para el algoritmo Tabu-search resultaron ser positivos.

En cuanto a los movimientos de Tabu-search, se observa que en todas las instancias y considerando todos los valores de α, se obtiene mejor calidad de la función objetivo utilizando solo Insert sobre usar solo 2-Interambio. Comparando los resultados de Insert con la combinación de los movimientos (2-Intercambio e Insert), en la gran mayoría de las instancias se obtienen el mismo o mejores resultados usando la combinación de ambos movimientos, con la excepción de la instancia 4 (Tabla VII), en donde por ejemplo con los valores de α igual a 0.1, 0.4, 0.5, 0.6, se obtienen mejores resultados usando solo el movimiento Insert que la combinación de ambos. Es posible que se deba a que el movimiento 2-Intercambio diversifique hacia espacios de soluciones donde no se pueda encontrar un buen óptimo, impidiendo intensificar la solución. Notar también que en la instancia 4 (Tabla VII) con el valor de α igual a 0.1, con ambos movimientos por separado se obtiene una mejor solución que combinando ambos movimientos.

En cuanto a tiempo de ejecución, no fue considerado ya que en la totalidad de ejecuciones de Greedy, la obtención del resultado era casi instantánea y con Tabu-search, el tiempo de ejecución era menor a 2 segundos, por lo que no es relevante considerarlo. Una forma de poder obtener mejores resultados es aceptar soluciones infactibles, con el objetivo de diversificar aún más el espacio de soluciones, para poder realizar dicha diversificación, se debe modificar la función objetivo, penalizándola de tal forma de que dicha solución infactible no sea parte de las soluciones factibles. También es posible modificar la función objetivo, tal como lo hacen en [1], donde además de tener un parámetro más en la función objetivo, buscan distintas formas de relacionarlas y de considerar la más óptima.

References

[1] Rojee Pradhananga, Eiichi Taniguchia, Tadashi Yamada, "Ant colony system based routing and scheduling for hazardous material transportation". Department of Urban Management, Kyoto University, Katsura campus C-1, Kyoto 615-8540, Japan, pp. 1-8, 2010.

[2] Germán Paredes-Belmar, Andrés Bronfman, Vladimir Marianov, Guillermo Latorre-Núñez, "Hazardous materials collection with multiple-product loading", Pontificia Universidad Católica de Chile, Universidad Andres Bello, Chile, pp 1-7, 2016.

[3] K. David Pijawka, Steve Foote, and Andy Soesilo, "Improving Transportation of Hazardous Materials Through Risk Assessment and Routing", WASHINGTON, D. C., EE.UU, pp 1-10, 1985.

[4] Gary L. Urbanek and Edward J. Barber, "Development of criteria to designate routes for transporting hazardous materials", Washington, D.C., EE UU, pp 1-3, 1980.

[5] Richard F. Hartl, Christine Strauss, "Applying the ANT System to the Vehicle Routing Problem", University of Vienna, France, pp 1-3 1997.

[6] Erhan Erkut, Armann Ingolfsson "Transport Risk Models for Hazardous Materials: Revisited", University of Alberta, pp 1-3, 2003.

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.