Implementación Vecino más cercano y colonia de hormigas en el problema del Agente Viajero(TSPTW)

Charlie Alberto Angulo Angulo
Candidato a Magister en Informática
Escuela Colombiana de Ingeniería Julio Garavito
Bogotá DC, Colombia
Charlie.angulo@mail.escuelaing.edu.co

Resumen- Este artículo presenta una solución mediante 2 heurísticas, tratando de descubrir una solución factible y óptima para el Viajero Problema del agente viajero con las ventanas de tiempo.

Palabras clave: Problema del viajante, ventanas de tiempo, colonia de hormigas, vecino ms cercano, generación de rutas para vehículos.

I. INTRODUCION

El problema del viajante se utiliza como referencia en muchos métodos de optimización y tiene numerosas aplicaciones en muchas áreas diferentes como artificial Inteligencia, aprendizaje automático, tecnología de software, etc.

En este artículo nos referimos en particular a la realización del Problema de TSP, que utiliza algunas restricciones adicionales como ventanas de tiempo. Esta version se conoce como TSP-TW (Traveling Salesman Problem with Time Windows). El TSPTW se puede definir como el problema de encontrar un recorrido de costo mínimo que comienza y termina en el mismo depósito, donde cada nodo debe ser visitado exactamente una vez dentro de su tiempo ventana. Se asocia un costo no negativo con cada arco, y el costo total se puede tomar como el tiempo de finalización de la ruta total del tiempo de viaje o distancia total recorrida, y donde la distancia total recorrida se considerará el valor de costo.

Este además del TSP, se debe visitar cada ciudad y se fue dentro de un intervalo de tiempo determinado como el camino de Hamilton, el problema es un sub problema, TSP, TSPTW y la mayoría de los demás TSP las variantes son computacionalmente difíciles, por lo que ningún algoritmo es de esperar un polinomio en N. [1]

El problema es NP- difícil, y Savelsberg mostró que incluso encontrar una solución viable de TSPTW es NP-completo. TSPTW tiene aplicaciones importantes en enrutamiento y programación.[2]

Al igual que en el resto de variantes del TSP el TSPTW busca como objetivo minimizar el costo de la ruta necesaria para visitar a todos los clientes, pero su gran diferencia con el resto de problemas es que este incluye un periodo de tiempo fijado para cada algoritmo. De esta forma podemos considerar un grafo completo donde los vértices representan a los clientes a visitar y un almacén y las aristas representan el costo de ir de un punto a otro. Además, cada vértice posee asociado un intervalo de tiempo el cual debe respetar el vehículo, este intervalo de tiempo es representado por $[ai, bi]$ donde ai representa la hora de apertura de la ventana y bi representa la hora de clausura de la ventana. [1].

Para la solución del STPWT se eligió dos heurísticas, el vecino más cercano y el algoritmo de colonia de hormigas (Ant Colony Optimization, ACO, buscando encontrar el camino con menor costo posible. Tomando como base un artículo de solución de problemas de generación de rutas para vehículos con ventanas de tiempo (VRPWT),[3] adaptándolo a la necesidad que requerimos de intentar resolver el STPWT.

II. CONTEXTO

A. QUE ES TSP:

El problema fue formulado por primera vez en 1930 y es uno de los problemas de optimización más estudiados, es computacionalmente complejo,

se conoce gran cantidad de heurísticas y métodos exactos, básicamente consiste en encontrar el circuito óptimo en términos del viaje más corto, visitándolos tan solo una vez y volviendo al punto de partida inicial.[4]
Resulta ser uno de los más prominentes dentro del campo de la optimización combinatoria pues todavía no ha sido resuelto eficientemente. Si queremos conocer su historia debemos remontarnos en principio a la de uno anterior que se origina en la teoría de grafos. Entendemos por grafo al par G = (V, E) donde V es un conjunto finito de elementos que llamamos vértices y E es un conjunto de pares de vértices que denominamos ramas. Un ciclo es una sucesión de vértices u1, u2, ..., up tales que u1, u2, ..., up−1 son distintos, up = u1 y (u_i , u_i+1) ∈ E. Si el ciclo contiene todos los vértices se llama hamiltoniano (en honor al matemático irland´es Sir William Rowan Hamilton).[5]

El TSP para un grafo que tiene asignado para cada una de sus ramas un cierto peso es el problema de encontrar un ciclo hamiltoniano de mínimo peso, entendiéndose por peso del ciclo a la suma de los pesos de todas las ramas que pertenecen a él. TSP: traveling salesman problema inversamente, el problema de decidir si un grafo tiene un ciclo hamiltoniano es un caso especial del TSP (si se asigna a todas las ramas del grafo peso 0 y a las ramas faltantes las agregamos asignándoles peso 1, se tendría otro grafo que tiene ciclos hamiltonianos. Resolviendo el TSP, el nuevo grafo tiene un ciclo de mínimo peso = 0 si y sólo si el grafo original contiene un ciclo hamiltoniano). [2]
El problema del agente viajero tiene una variación importante, y esta depende de que las distancias entre un nodo y otro sean simétricas o no, es decir, que la distancia entre A y B sea igual a la distancia entre B y A, puesto que en la práctica es muy poco probable que así sea.

B. Métodos heurísticos

Es un procedimiento para resolver un problema de optimización bien definido mediante una aproximación intuitiva, en la que la estructura del problema se utiliza de forma inteligente para obtener una buena solución. [5][7]

C. Heurísticas

Es una función matemática h (f) que sirve como una estimación del coste del camino más económico de un nodo dado hasta el nodo objetivo. [7]No garantizan una solución óptima, pero si aproximadas al resultado óptimo. Un buen algoritmo heurístico debe ser eficiente, bueno, y robusto. [5]

D. Mejora Iterativa

La idea básica consiste en comenzar con una solución inicial e ir realizando modificaciones para mejorar su calidad. El objetivo de esta técnica consiste en explorar iterativamente el estado de soluciones para encontrar las soluciones óptimas. Se detallan algunas técnicas a continuación.[6] [8]

E. ALGUNOS MÉTODOS DE SOLUCIÓN TRADICIONALES

La complejidad del cálculo del problema del agente viajero ha despertado múltiples iniciativas por mejorar la eficiencia en el cálculo de rutas. El método más básico es el conocido con el nombre de fuerza bruta, que consiste en el cálculo de todos los posibles recorridos, lo cual se hace extremadamente ineficiente y casi que se imposibilita en redes de gran tamaño. [6] También existen heurísticos que se han desarrollado por la complejidad en el cálculo de soluciones óptimas en redes robustas, es por ello que existen métodos como el vecino más cercano, la inserción más barata y el doble sentido. Por último se encuentran los algoritmos que proporcionan soluciones óptimas, como el método de branch and bound (ramificación y poda), que trabaja el problema como un algoritmo de asignación y lo resuelve por medio del método simplex.[4]

1) Método de la fuerza bruta

El método de la fuerza bruta no implica la aplicación de ningún algoritmo sistemático, tan solo consiste en explorar todos los recorridos posibles. Considerando la siguiente red simétrica, los caminos posibles se reducen a la mitad: [7]

2) Algoritmo de búsqueda local

Consiste en partir de una solución inicial t0, encontrar un s ∈ N(t) con coste menor, para de esa forma disminuir h(t), y repetir este procedimiento hasta llegar a un t en cuyo entorno resulte imposible disminuir el coste. Si esto pasa, se está en presencia de un mínimo local. [7][9]

3) Método de Branch and Bound

El método de branch and bound (ramificación y poda), nos proporciona una solución óptima del problema del agente viajero, calculando mediante el algoritmo simplex la solución del modelo. A medida que aumente el tamaño de la red el método puede tardar gran cantidad de tiempo en resolverse, sin embargo, para redes de mediano tamaño es una excelente alternativa. [7]

4) Algoritmos Genéticos (GA)

Son heurísticos inspirados en el proceso de evolución de los seres vivos. Funcionan entre el conjunto de soluciones de un problema llamado fenotipo, y el conjunto de individuos de una población natural, codificando la información de cada solución en una cadena, generalmente binaria, llamada cromosoma que representan los diferentes valores de las variables de la solución.

Se genera aleatoriamente la población inicial, que está constituida por un conjunto de cromosomas los cuales representan las posibles soluciones del problema, en estos apareamientos intercambiamos genes de ambas soluciones para generar nuevas soluciones, luego generamos mutaciones al azar para finalmente seleccionar los mejores 50 individuos de la población para conformar la población de la generación siguiente. [10][11]

5) Optimización por Colonia de Hormigas

El algoritmo de la colonia de hormigas, algoritmo hormiga u optimización por colonia de hormigas (Ant Colony Optimization, ACO) es una técnica probabilística para solucionar problemas computacionales que pueden reducirse a buscar los mejores caminos o rutas en grafos. [12]

III. SITUACION ACTUAL

Esta limitación de tiempo hace el problema es aún más difícil en la práctica. La hermosa simetría de TSP, donde cualquier permutación de ciudades es factible solución, se viola e incluso la búsqueda de soluciones viableses difícil [10]. Dado un depósito, un conjunto de clientes, el servicio tiempo (es decir, el tiempo que se debe dedicar al cliente) y un tiempo ventana (es decir, el tiempo de inicio y expiración), el TSP-TW El problema se refiere a encontrar un camino más corto, que comienza y termina en un depósito determinado, ya que cada cliente solo ha sido visitado una vez antes de su fecha de caducidad. La forma en que determinamos el depósito es importante para el TSPTW, aunque esto no es cierto para la versión general del TSP. [13]

El vendedor ambulante o agente puede llegar al cliente antes de su tiempo listo, pero debe esperar. Obviamente, hay son caminos que no permiten al agente respetar los plazos de todos los clientes. Llamamos a estos caminos inviables, mientras que muchos Las rutas permiten al agente respetar los plazos de todos los clientes. llamados caminos factibles. El costo de un camino es la distancia total viajó [5]. Dado el gráfico G = (V, A), donde V = {1,2, ..., norte}. Suponga que 0 indica un depósito y A = (i, j) i, j ∈ V ∪ {0} es el conjunto de arcos entre clientes. El costo del tour (viajar costo) de i a j está representado por c_{ij}, que incluye tanto el tiempo de atención al cliente i, y el tiempo que se tarda en pasar de yo a j. Cada cliente i está asociado con una ventana de tiempo a_i,b_i, donde a_i y b_i representan el tiempo de preparación y madurez, respectivamente. Por tanto, el TSP-TW se puede formular como problema de encontrar un camino hamiltoniano, que comienza y termina en el depósito, satisfaciendo todas las limitaciones de tiempo y minimizando la distancia total recorrida. [13]

A. Definición del problema

El TSPTW se define formalmente de la siguiente manera. Sea G = (N, A) un gráfico finito, donde N = {0, 1,. . ., n} consiste en un conjunto de nodos que representan el depósito (nodo0) y n clientes, y A = N × N es el conjunto de arcos que conectan los nodos.

Para cada arco a_{ij} ∈ A entre dos nodos i y j, hay un costo asociado c (a_{ij}). Este costo generalmente representa el tiempo de viaje entre los clientes i y j, más un tiempo de servicio en el cliente i.

Para cada nodo i ∈ N, hay una ventana de tiempo asociada, [e_i, l_i], donde e_i representa la primera hora de inicio del servicio y l_i es la última hora de inicio del servicio. Una solución al problema es un recorrido que visite cada nodo una vez, comenzando y terminando en el depósito. Por tanto, un recorrido se representa como P = (p0 = 0, p1,..., P^n, $p^n + 1 = 0$), donde la subsecuencia (p^1,..., P^k,..., P^n) es una permutación de los nodos en N \ {0} y p^k denota el índice del cliente en la k-ésima posición del recorrido.

Dos elementos adicionales, $p^0 = 0$ y $p^n + 1 = 0$, representan el depósito.

Se supone que los tiempos de espera están permitidos, es decir, se puede llegar a un nodo i

antes del inicio de su ventana de tiempo e_i, pero no se puede dejar antes de e_i. Por lo tanto, la hora de salida del cliente p^k se calcula como $Dp^k = \max(Ap^k, ep^k)$, donde $Ap^k = Dp^k - 1 + c(ap^k - 1, p^k)$ es la hora de llegada al cliente p^k en el tour.[14]

La literatura define dos objetivos relacionados pero diferentes para este problema. Uno es la minimización del costo de los arcos atravesados a lo largo del recorrido $\Sigma_{\Pi}^{k} = 0$ (ap^k, $p^k + 1$). La otra alternativa es minimizar $Ap^n + 1$, la hora de llegada al depósito. En este trabajo nos centramos en lo primero y, por tanto, formalmente definió el TSPTW como:

$$\text{minimizar: } F(P) = \sum_{k=0}^{norte} c(a_{p_k, p_{k+1}})$$

sujeto a:

$$\Omega(P) = \sum_{k=0}^{n+1} \omega(p_k) = 0$$

dónde:

$$\omega(p_k) = \begin{cases} 1 \text{ si } A_{p_k} > l_{p_k} \\ 0 \text{ en caso contrario;} \end{cases},$$

$$A_{p_{k+1}} = \text{máximo}(A_{p_k}, e_{p_k}) + c(a_{p_k, p_{k+1}}).$$

En la definición anterior, Ω (P) denota el número de restricciones de ventana de tiempo que son violados por el recorrido P , que debe ser cero para soluciones factibles.

Implementaciones

Actualmente se han encontrado formas de resolver el problema con metaheurística de procesos como Variable Neighborhood Search (VNS) es una heurística de alto nivel, diseñada para encontrar, crear o seleccionar una heurística de nivel inferior (por ejemplo, un algoritmo de búsqueda), que puede proporcionar una solución bastante buena para un problema de optimización, particularmente con falta o incompleta información, o con capacidad informática limitadas.[13][15]

En el caso del agente viajero con restricciones de tiempo, [13]lo utilizaron para la búsqueda local seleccionando una solución inicial x, encontrando una dirección de descenso de x, en un vecindario N (x) y avanzando hacia el mínimo de f (x) en N (x) en la misma dirección. Si no hay dirección del descenso, luego la heurística se detiene; de lo contrario es repetido. En cada paso, la vecindad N (x) de x ha sido completamente explorado. VNS está diseñado para encontrar aproximaciones soluciones de problemas de optimización discretos y continuos y según ellos, resolviendo programación de enteros, mezclado programación entera, programación no lineal, utilizando backtracking y donde el resultado mostró que el algoritmo no logra obtener soluciones factibles para algunos casos con gran cantidad de clientes y ventanas de tiempo ajustadas.

De hecho, la heurística VNS es utilizada en la actualidad para realizar una operación de importancia sustancial para el manejo de la programación diaria de una ciudad en el tráfico es la ruta de los camiones de basura.[16]

B. Heurística seleccionada para la solución del problema STPWT

1) Heurística el vecino más cercano(VNS)

Es una metaheurística basada en cambios sistemáticos de vecindarios combinados con búsqueda local. Estos cambios de vecindad pueden usarse tanto en la fase de intensificación (descenso) como en la de diversificación (perturbación). Para obtener más detalles sobre la búsqueda de vecindad variable y los procedimientos y variantes relacionados.[15]

El método del vecino más cercano es un algoritmo heurístico diseñado para solucionar el problema del agente viajero, no asegura una solución óptima, sin embargo, suele proporcionar buenas soluciones, y tiene un tiempo de cálculo muy eficiente.

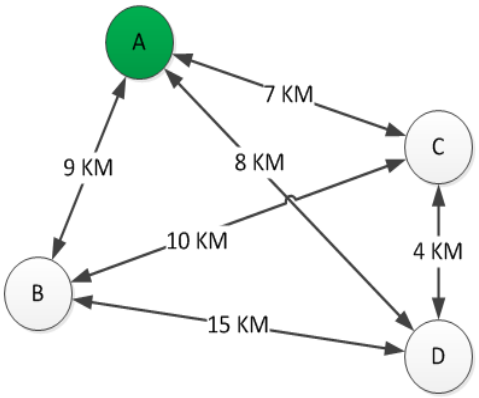

Figura.1Representacion de nodos en diferentes distancias.

El método consiste en una vez establecido el nodo de partida, evaluar y seleccionar su vecino más cercano. En este caso:

Vecinos de A	B	C	D
Distancia	9	7	8

En la siguiente iteración habrá que considerar los vecinos más cercanos al nodo C (se excluye A por ser el nodo de origen):

Vecinos de C	B	D
Distancia	10	4

En la siguiente iteración los vecinos más cercanos de D serán C, con quien ya tiene conexión, A quién es el nodo de origen y B, por esta razón B se debe seleccionar por descarte. Al estar en B todos los nodos se encuentran visitados, por lo que corresponde a cerrar la red uniendo el nodo B con el nodo A, así entonces la ruta solución por medio del vecino más próximo sería A, C, D, B, A = 7, 4, 15, 9 = 35 km.

Este es un caso en el que, a pesar de tener una red compuesta por pocos nodos, el método del vecino más cercano no proporciona la solución óptima, la cual se calcula con el método de fuerza bruta como 31 km. [6]

2) El algoritmo Colonia de hormigas(ACO)

La ACO pertenece a la clase de métodos heurísticos, los cuales son algoritmos aproximados utilizados para obtener soluciones lo suficientemente buenas a problemas. La fuente de inspiración de la Aco es el comportamiento real de las hormigas. Estos insectos cuando están en búsqueda de la comida inicialmente exploran el área alrededor de su nido de una forma aleatoria. Tan pronto encuentran fuentes de alimentos, evalúan su cantidad y calidad, y llevan alguna parte de esta comida para su nido. complejos en una cantidad razonable de tiempo de cómputo.

Durante el regreso al nido, las hormigas depositan una sustancia química llamada feromona sobre el camino, la cual servirá de guía futura para que las demás encuentren los alimentos. La cantidad de dicha sustancia depositada dependerá de la cantidad y calidad de los alimentos.[17]

Diferentes estudios han demostrado que la comunicación de las hormigas a través de caminos con feromonas les permite encontrar las rutas más cortas entre su nido y las fuentes de alimentos.

Esta característica es ampliamente utilizada para la solución de problemas de optimización que necesitan mejorar sustancialmente los tiempos de cómputo para la solución de una aplicación específica.[17]

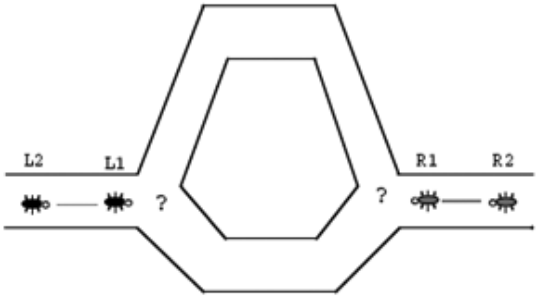

Figura.2 Las hormigas en el punto de decisión.

Las figuras 3 y 4 muestran lo que ocurre en los siguientes instantes, suponiendo que todas las hormigas caminan a la misma velocidad. El número de líneas punteadas es proporcional a la cantidad de feromonas que los insectos han depositado en él.

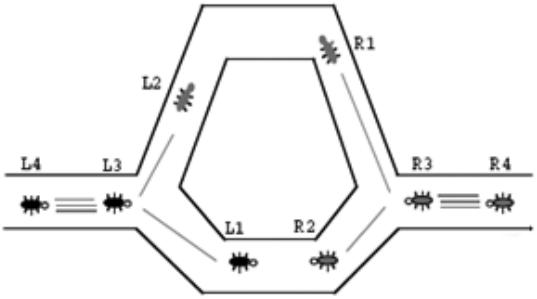

Figura.3: Elección de forma aleatoria entre los caminos inferior y superior

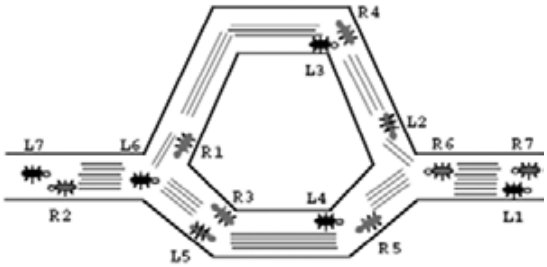

Figura 4. Efectos de la cantidad de hormigas que circulan por cada camino.

Como el camino inferior es más corto que el

superior, muchas más hormigas transitarán por éste durante el mismo periodo de tiempo. Esto implica que en el camino más corto se acumula más feromona mucho más rápido. Después de cierto tiempo, la diferencia en la cantidad de feromona en los dos caminos es lo suficientemente grande para influenciar la decisión de las nuevas hormigas que entren a recorrer estas vías.[17]

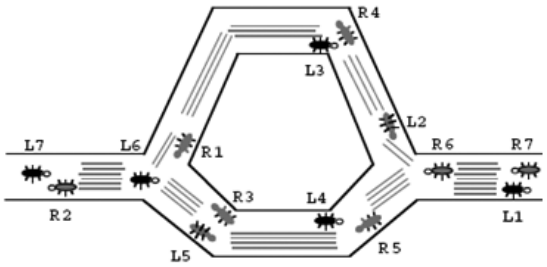

Figura 5. Mayor acumulación de feromonas en el camino más corto.

Teniendo en cuenta lo anterior las nuevas hormigas que entren al sistema preferirán escoger el camino inferior o más corto puesto que perciben una mayor cantidad de feromona en ése. Dicho fenómeno se incrementa como un efecto de retroalimentación positiva en el cual todas las hormigas utilizarán el camino más corto. Hay que mencionar que si se traslada este comportamiento directamente al pc para diseñar un algoritmo de búsqueda podemos encontrarnos con que éste se quede rápidamente con la estrategia más óptima.

Estos algoritmos, que están basados en una colonia de hormigas artificial, son agentes computacionales que trabajan de manera conjunta para poder comunicarse a través de rastros de feromonas artificiales. [17]

Los algoritmos ACO son programas constructivos:

en cada iteración del algoritmo cada hormiga construye una solución al problema a través de un grafo. Cada arista representa las posibles opciones que el insecto puede tomar y tiene asociada el siguiente tipo de información: .[17]

• Información heurística: en ésta se mide la preferencia heurística que tienen las hormigas para moverse de un nodo a otro. El camino recorrido de un nodo a otro es una arista. Esta información no es modificada durante la ejecución del algoritmo.

• Información de los rastros de feromonas artificiales: en ésta se mide la deseabilidad aprendida en el movimiento de un nodo a otro, lo cual busca imitar la feromona real que depositan las hormigas naturales. Este tipo de información es modificada mientras que se ejecuta el algoritmo dependiendo de las soluciones encontradas por los insectos.

C. Estrategia.

Se toma como referencia el articulo macs-vrptw a multiple ant colony system for vehicle routing problems with time windows,[3] adaptando a la necesidad de resolver el problema TSPWT y donde como estrategia se utiliza la heurística del vecino más cercano, para encontrar una solución y un algoritmo de colonia de hormigas.

Básicamente se pasa a la heurística del vecino más cercanos el número máximo de arcos y n ij y las ventanas de tiempo T_{ij}, este realiza un proceso en cuanto a costo y tiempo, devolviendo una solución a si no sea tan optima, y el resultado es utilizado para actualizar la matriz de feromonas locales y globales de las hormigas las cuales tomaran como base para iniciar el proceso de optimización el costo de recorrido hacia cada uno de los nodos.

En ACS se ha asociado dos medidas a cada arco del gráfico TSP: la cercanía η_{ij}, y el rastro de feromonas τ_{ij}. La cercanía, definida como la inversa de la longitud del arco, es un valor heurístico estático, que nunca cambia para una instancia de problema determinada, mientras que las hormigas cambian dinámicamente el rastro de feromonas en tiempo de ejecución. Por tanto, el componente más importante de ACS es la gestión de los rastros de feromonas que se utilizan, junto con la función objetivo, para construir nuevas soluciones. De manera informal, los niveles de feromonas dan una medida de lo deseable que es insertar un arco dado en una solución. Los rastros de feromonas se utilizan para exploración y explotación. La exploración se refiere a la elección probabilística de los componentes utilizados para construir una solución: la probabilidad se da a elementos con un fuerte rastro de feromonas. La explotación elige el componente que maximiza una combinación de valores de rastro de feromonas y evaluaciones heurísticas.

El objetivo de ACS es encontrar un recorrido más corto. Las hormigas construyen recorridos en paralelo, donde la distancia es un parámetro. Cada hormiga se asigna aleatoriamente a un nodo inicial y tiene que construir una solución, es decir, un recorrido completo. Un recorrido se construye nodo por nodo: cada hormiga agrega iterativamente nuevos nodos hasta que todos los nodos han sido visitados. Cuando hormiga k se encuentra en el nodo i, eso elige el siguiente nodo j probabilísticamente en el conjunto de nodos factibles N_i^K . La regla probabilística utilizada para construir un recorrido es la siguiente: con probabilidad q0 un nodo con el más alto$T_{ij}[\cap ij]^B, J \in N_i^k$ se elige (explotación), mientras que con probabilidad (1- q 0) el nodo j se elige con un probabilidad p_{ij} proporcional a $T_{ij}[\cap ij]^B, J \in N_i^k$.

$$p_{ij} = \begin{cases} \dfrac{\tau_{ij} \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k} \tau_{il} \cdot [\eta_{il}]^{\beta}} & \text{if } j \in N_i^k \\ 0 & \text{otherwise} \end{cases}$$

dónde β y q 0 son parámetros: β sopesa la importancia relativa del valor heurístico, mientras que q 0 determina la importancia relativa de la explotación frente a la exploración: el menor q 0 cuanto mayor sea la probabilidad de utilizar la regla probabilística descrita con la Ecuación 1.

Una vez que cada hormiga ha creado una solución completa, esta se mejora tentativamente y se actualiza los rastros de feromonas. Luego, el proceso se repite comenzando de nuevo hasta que se cumpla una condición de terminación. El ACS termina cuando se cumple al menos una de las siguientes condiciones: se ha generado un número fijo de soluciones optimas, hasta que haya logrado ninguna mejora durante un número determinado de iteraciones.

Las soluciones construidas se utilizan para actualizar los rastros de feromonas. La razón es que de esta manera se memoriza una "ruta preferida" en la matriz del camino de feromonas y las futuras hormigas utilizarán esta información para generar nuevas soluciones de las generadas por el algoritmo del vecino más cercano de esta ruta preferida. los τ_{ij} se actualizan de la siguiente manera:

$$\tau_{ij} = (1-\rho)\cdot \tau_{ij} + \rho / J_{\psi}^{gb} \qquad \forall (i,j) \in \psi^{gb}$$

dónde ρ ($0 \le \rho \le 1$) es un parámetro y J_{Ψ}^{gb} es la longitud deΨ^{gb}, el camino más corto generado por las hormigas desde el comienzo del cálculo. Este procedimiento de actualización global se aplica al final de cada ciclo, es decir, cada vez que se ha completado la fase constructiva.

La actualización de hormigas local de feromonas se realiza de la siguiente manera:

Cuando una hormiga se mueve desde el nodo I al nodo j, la cantidad de rastro de feromonas en el arco (i , j) es disminuir de acuerdo con la siguiente regla:

$$\tau_{ij} = (1-\rho)\cdot \tau_{ij} + \rho \cdot \tau_0$$

dónde τ es el valor inicial de los senderos. Se encontró que $T_0 = {}^{1}/_{(\cap . J_{\Psi}^{h})}$ es un buen valor para este parámetro donde J_{ψ}^{h} es la longitud de la solución inicial producida por la heurística del vecino más cercano.

D. Resultados computacionales

La práctica se realizó con un Windows 10 pro x64 bits, Procesador Intel(R) core™ i7-6700T cpu @2.80Ghz, memoria Ram 8,00 Gb. El experimento utilizo muestras de SolomonPotvinBengio con los siguientes ajustes de parámetros: m = 10 hormigas, q 0 = 0,1, β = 1 y ρ = 0,1, Beta =2, max_iter =200. El código fue escrito en Python.

Un ejempo de ejecucion para el archivo SolomonPotvinBengio\rc_201.1.txt, donde muestra la solución no tan optima de la heurística del vecino más cercano y como las colonias de hormigas optimizan en el costo del recorrido e incluso el tiempo de ejecución, aunque no sea el objetivo.

Recorrido vecino [0, 14, 15, 2, 0, 19, 12, 11, 3, 10, 1, 0, 13, 18, 17, 16, 7, 0, 9, 8, 5, 6, 4, 0] costo

342.29862605164163, tiempo 0.04686236381530762.

Recorrido Hormigas [0, 14, 16, 7, 0, 15, 0, 12, 0, 13, 19, 1, 10, 3, 2, 0, 11, 17, 0, 18, 4, 9, 5, 6, 8, 0] tiempo 0.0 [iteration 0]: costo 354.829420.

Recorrido Hormigas [0, 14, 13, 19, 15, 0, 18, 12, 10, 3, 2, 0, 9, 5, 6, 8, 7, 4, 17, 11, 1, 0, 16, 0] tiempo 0.01564931869506836 [iteration 1]: costo 294.248969.

Recorrido Hormigas [0, 14, 13, 19, 11, 17, 12, 1, 10, 3, 2, 0, 15, 0, 18, 4, 6, 8, 7, 5, 16, 0, 9, 0] tiempo 0.03124213218688965 [iteration 2]: costo 278.056911.

Recorrido Hormigas [0, 14, 13, 19, 15, 0, 18, 12, 0, 9, 7, 5, 6, 8, 4, 17, 11, 1, 10, 3, 2, 0, 16, 0] tiempo 0.04686141014099121 [iteration 4]: costo 253.048443.

Recorrido Hormigas [0, 14, 13, 19, 15, 0, 9, 7, 5, 6, 8, 4, 17, 11, 1, 10, 3, 2, 0, 18, 12, 0, 16, 0] tiempo 0.14061284065246582 [iteration 12]: costo 253.048443.

Recorrido Hormigas [0, 14, 13, 19, 15, 0, 9, 7, 5, 6, 8, 4, 16, 12, 11, 1, 10, 3, 2, 0, 18, 17, 0] tiempo 0.3285515308380127 [iteration 31]: costo 239.331335.

Recorrido Hormigas [0, 14, 13, 19, 15, 0, 18, 12, 11, 1, 10, 3, 2, 0, 9, 7, 5, 6, 8, 4, 16, 17, 0] tiempo 0.3285515308380127 [iteration 32]: costo 236.092817.

Recorrido Hormigas [0, 14, 13, 19, 15, 0, 9, 7, 5, 6, 8, 4, 16, 17, 11, 1, 10, 3, 2, 0, 18, 12, 0] tiempo 0.35979747772216797 [iteration 34]: costo 229.844764.

Recorrido Hormigas [0, 18, 12, 11, 1, 10, 3, 2, 15, 0, 14, 13, 19, 17, 0, 9, 7, 5, 6, 8, 4, 16, 0] tiempo 1.1096141338348389 [iteration 120]: costo 217.783205.

Recorrido Hormigas [0, 14, 13, 19, 17, 11, 1, 10, 3, 2, 15, 0, 18, 12, 0, 9, 7, 5, 6, 8, 4, 16, 0] tiempo 1.1408584117889404 [iteration 123]: costo 211.535153.

El menor costo obtenido con hormigas es 211.5351526645167 en 1.8128561973571777 segundos.

Volver varias veces al depósito es menos atractivos para las hormigas, ya que las reglas de actualización de feromonas, acepta un único depósito.

Los resultados generales con todas las muestras son los siguientes:

	Vecino mas cecano		Hormigas	
data	costo	tiempo	costo	tiempo
rc_201.1.txt	342.298	0.04	211.535	1.14
rc_201.2.txt	540.836	1.90	306.296	1.68
rc_201.3.txt	417.337	4.64	307.996	1.28
rc_201.4.txt	562.329	7.84	385.554	1.23
rc_202.1.txt	430.017	10.12	250.052	2.98
rc_202.2.txt	202.545	13.37	140.608	0.04
rc_202.3.txt	531.172	14.36	425.881	1.42
rc_202.4.txt	614.690	17.22	444.307	1.96
rc_203.1.txt	331.134	19.84	232.817	0.15
rc_203.2.txt	661.901	21.40	378.645	1.00
rc_203.3.txt	770.499	24.25	456.965	3.06
rc_203.4.txt	215.399	27.83	147.566	0.07
rc_204.1.txt	486.325	28.95	343.088	1.50
rc_204.2.txt	511.146	33.17	345.677	0.43
rc_204.3.txt	209.346	36.01	156.981	0.28
rc_205.1.txt	288.120	37.84	202.762	0.24
rc_205.2.txt	476.188	38.97	250.538	2.29
rc_205.3.txt	587.885	41.55	344.424	0.76
rc_205.4.txt	959.799	44.77	571.895	0.45
rc_206.1.txt	84.883	47.34	84.883	0.0
rc_206.2.txt	438.843	47.56	298.564	1.09
rc_206.3.txt	473.832	51.02	258.562	0.60
rc_206.4.txt	506.286	53.14	246.227	3.28
rc_207.1.txt	410.636	56.91	347.911	2.29
rc_207.2.txt	390.987	60.33	286.824	0.81
rc_207.3.txt	365.120	63.03	287.162	1.26
rc_207.4.txt	79.585	66.10	71.852	0.01
rc_208.1.txt	276.005	66.50	0	0
rc_208.1.txt	276.005	0.01	0	0
rc_208.2.txt	297.991	0.062	195.508	0.34
rc_208.2.txt	238.591	0.06	195.706	0.24

IV. CONCLUSIÓN

Tomar como base el problema VRPWT, y adaptar su lógica al problema TSPWT no genero mucho desgaste y se obtuvo un gran resultado, intentando construir soluciones factibles y optimizando el costo de recorrido mediante las feromonas de la heurística de colonia de hormigas(ACO). Obteniendo como resultado soluciones no tan factibles por parte del algoritmo del vecino más cercano (VNS) y un porcentaje mayoritario para la ACO.

Finalmente respaldar que, al abordar una variante del problema, podría ser una buena estrategia adaptar primero los algoritmos más conocidos de la literatura a la variante del

problema, antes de comenzar para desarrollar algoritmos completamente nuevos.

V. TRABAJOS FUTUROS

Lograr construir la solución completa de una MACS TSPWT, mediante la heurística de búsqueda tabú y otro método de colonia de hormigas, estas 2 metahuristicas enfocadas a optimizar el tiempo que toma recorrer cada uno de los nodos para encontrar soluciones eficientes en cuestiones de tiempo.

VI. REFERENCIAS

[1] S. Edelkamp, M. Gath, T. Cazenave, and F. Teytaud, "Algorithm and knowledge engineering for the TSPTW problem," *Proc. 2013 IEEE Symp. Comput. Intell. Sched. CISched 2013 - 2013 IEEE Symp. Ser. Comput. Intell. SSCI 2013*, no. September 2020, pp. 44–51, 2013, doi: 10.1109/SCIS.2013.6613251.

[2] F. Focacci, M. Milano, and A. Lodi, "Solving TSP with Time Windows with Constraints," *Log. Program.*, 2020, doi: 10.7551/mitpress/4304.003.0043.

[3] Y. For, V. Routing, P. With, and T. Windows, "MACS-VRPTW : A M ULTIPLE A NT C OLONY S YSTEM FOR VEHICLE ROUTING PROBLEMS WITH TIME WINDOWS In D . Corne , M . Dorigo and F . Glover , editors New Ideas in Optimization Chapter 5 MACS-VRPTW : A M ULTIPLE A NT C OLONY S YSTEM FOR," pp. 1–17, 1999.

[4] "Problema del agente viajero - TSP | Ingenieria Industrial Online." https://www.ingenieriaindustrialonline.com/investigacion-de-operaciones/problema-del-agente-viajero-tsp/ (accessed Feb. 04, 2021).

[5] L. Stockdale and D. S. Puddu, "El problema del viajante: un algoritmo heurístico y una aplicación.," p. 114, 2011, [Online]. Available: http://cms.dm.uba.ar/academico/carreras/licenciatura/tesis/2011/Stockdale_Lorena.pdf.

[6] "Problema del viajante - Wikipedia, la enciclopedia libre." https://es.wikipedia.org/wiki/Problema_del_viajante (accessed Feb. 01, 2021).

[7] C. Rego, D. Gamboa, F. Glover, and C. Osterman, "Traveling salesman problem heuristics: Leading methods, implementations and latest advances," *Eur. J. Oper. Res.*, vol. 211, no. 3, pp. 427–441, 2011, doi: 10.1016/j.ejor.2010.09.010.

[8] A. P. Martinez, L. M. López, U. Nacional, and F. De Informática, "Variante Heurística para el Problema del Viajante . Caso de Aplicación : Circuito de Pesca Deportiva," pp. 695–704, 2018.

[9] N. Christofides, "Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem.," no. February, 1976.

[10] "Tres métodos diferentes para resolver el problema del viajante." https://baobabsoluciones.es/blog/2020/10/01/problema-del-viajante/ (accessed Feb. 05, 2021).

[11] "Algoritmo genético - Wikipedia, la enciclopedia libre." https://es.wikipedia.org/wiki/Algoritmo_genético (accessed Feb. 05, 2021).

[12] "Algoritmo de la colonia de hormigas - Wikipedia, la enciclopedia libre." https://es.wikipedia.org/wiki/Algoritmo_de_la_colonia_de_hormigas (accessed Feb. 04, 2021).

[13] C. Papalitsas, K. Giannakis, T. Andronikos, D. Theotokis, and A. Sifaleras, "Initialization methods for the TSP with Time Windows using Variable Neighborhood Search," *IISA 2015 - 6th Int. Conf. Information, Intell. Syst. Appl.*, 2016, doi: 10.1109/IISA.2015.7388106.

[14] M. López-Ibáñez and C. Blum, "Beam-ACO based on stochastic sampling: A case study on the TSP with time windows," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5851 LNCS, pp. 59–73, 2009, doi: 10.1007/978-3-642-11169-3_5.

[15] B. Gendron, K. Amghar, J. Cordeau, and B. Gendron, "A General Variable Neighborhood Search Heuristic for the

Traveling Salesman Problem with Time Windows under Completion Time Minimization A General Variable Neighborhood Search Heuristic for the Traveling Salesman Problem with Time Windows under Completion," no. July, 2019.

[16] C. Papalitsas, P. Karakostas, T. Andronikos, S. Sioutas, and K. Giannakis, "Combinatorial GVNS (general variable neighborhood search) optimization for dynamic garbage collection," *Algorithms*, vol. 11, no. 4, pp. 1–18, 2018, doi: 10.3390/a11040038.

[17] A. S., C. O., F. de Viana I., and H. F, "La Metaheur□stica de Optimizaci□n Basada en Colonias de Hormigas: Modelos y Nuevos Enfoques," no. May 2014, pp. 261–314, 2004.