Inteligencia Artificial

Informe Final: Team Orienteering Problem

Sebastián Ignacio Ramírez Vidal

15 de diciembre de 2018

Evaluación

Mejoras 1ra Entrega (10 %):	_____
Código Fuente (10 %):	_____
Representación (15 %):	_____
Descripción del algoritmo (20 %):	_____
Experimentos (10 %):	_____
Resultados (10 %):	_____
Conclusiones (20 %):	_____
Bibliografía (5 %):	_____
Nota Final (100):	_____

Resumen

Un problema común en la industria es la búsqueda de rutas óptimas para realizar servicios que requieren el despliegue de múltiples unidades para recubrir puntos de valor en un horario acotado. Este problema, generalizado, se le conoce como Team Orienteering Problem y en este artículo se procederá a revisar distintos acercamientos a este problema a través del tiempo con el objetivo de revisar las soluciones, o aproximaciones a soluciones, de cada uno comparando su eficiencia tanto en calidad de resultado como tiempo que demora en obtenerlo, para finalmente proponer una solución utilizando un algoritmo de Backtracking con GBJ.

Keywords: Team Orienteering Problem, Método exacto, Método inexacto, Metaheurística, Backtracking

1. Introducción

Orienteering es un deporte al aire libre que consiste en que un participante, desde de un punto de partida, debe visitar distintos puntos de control hasta terminar en un punto de llegada dentro de un margen delimitado de tiempo. Cada punto de control tiene un puntaje asociado el cual se le otorgará al participante sólo la primera vez que pase por allí, por lo que el ganador será aquel que haya juntado la mayor cantidad de puntos tras haber llegado a la meta dentro del tiempo disponible, implicando que no necesariamente visitará la mayor cantidad de puntos respecto a otros participantes, es por esto que responder cual es la ruta que se debe tomar para ganar no es trivial y se puede modelar como un problema de optimización.

El problema de Orienteering para un sólo participante, que es el que se acaba de describir, envuelve mas capas de complejidad cuando uno cambia el enfoque de un solo participante a un equipo participante con dos o mas integrantes. En este caso el problema pasa a ser intuitivamente conocido como uno de Team Orienteering Problem, o TOP para abreviar. Ya que el problema no sólo se trata de encontrar un camino que entregue la mayor cantidad de puntos, sino que varios caminos para que puedan ser recorridos en paralelo por los integrantes del grupo dentro del margen de tiempo y que en conjunto me den el mayor puntaje posible. Considerando que el problema de Orienteering para un solo participante es NP-difícil [1], TOP tiene al menos la misma complejidad que este al ser un derivado de el [2].

Debido a la naturaleza de este problema, es de esperarse que multiples cientistas de la computación en el área de la investigación operacional hayan intentado abordarlo a través del uso de distintas técnicas para encontrar soluciones que satisfagan lo planteado en un problema TOP de forma eficiente para una generalidad de instanciaciones, esto debido a la fuerte similitud que este problema tiene con respecto a operaciones en la industria actual, a raíz de los cuales se producen variantes de TOP las cuales serán discutidos mas adelante.

Finalmente es por todo lo ya explicado que se redacta este documento, con la finalidad de dar revisión a estos diversos acercamientos al problema TOP, las mejoras y variedades entre cada uno de ellos y a las variantes que han surgido del mismo. Además igualmente se propone una solución al problema utilizando una técnica de Backtracking con Graph Back Jump (GBJ), se describe su implementación en base a sus representaciones y se comparan sus resultados con los actualmente existentes. En lo que queda de este artículo, se describe el problema en detalle denotando sus variables, restricciones y función objetivo en la Sección 2. Luego, en la Sección 3 se presentan diversos acercamientos al problema resaltando su instante conológico y metodología aplicada. En la Sección 4 se entrega un modelo matemático del problema. Posteriormente en la Sección 5 se describe la representación utilizada en la solución propuesta, en la Sección 6 se describe el algoritmo utilizado y en las Secciones 7 y 8 se muestran los experimentos para las intancias que se resolvieron utilizando el código de la solución propuesta y sus respectivos resultados. Finalmente, en la Sección 9 se concluye de acuerdo a la revisión realizada y a la solución propuesta respecto de los resultados obtenidos y posibles mejoras al problema.

2. Definición del Problema

Aunque en la Sección 1 ya se dio una idea de como se puede observar un problema de TOP haciéndo la analogía con el deporte, no todo problema de este tipo estará necesariamente ligado al deporte. Es por esto que es necesario dar una definición mas general, en un lenguaje natural, como funciona un problema de Team Orienteering.

De acuerdo a como Boussier [3] lo define, un problema de Team Orienteering puede ser visto como un grafo constituido de los puntos de control, que serían vértices etiquetados con el valor del premio por visitarlos, y los arcos como las rutas entre ellos, cuya etiqueta sería la distancia entre los nodos que conecta. El objetivo de TOP es encontrar un conjunto de rutas (una ruta por cada participante) sujetas a un tiempo determinado para recorrerlas, recorriendo cada nodo intermedio a los mas una vez y cuyo premio acumulado por visitarlos sea máximo. Generalmente, el problema se simplifica asumiendo una velocidad de recorrido de 1 unidad de distancia por 1 unidad de tiempo, por lo que el tiempo máximo pasa a ser una distancia máxima recorrida, haciendo que la suma total de las distancias de cada ruta sea la distancia recorrida y que sea eso lo que se limite en vez del tiempo.

Así como se realiza la analogía de TOP con el deporte al aire libre, igualmente se le asocia el problema de enrutamiento de vehículos para la visita de clientes y con distancia máxima de recorrido, la cual constituye un gran esfuerzo de logística en muchos contextos de la industria actual, como por ejemplo la visita de técnicos a clientela, planificación de viajes turísticos, etc.

De igual forma que TOP proviene de OP, muchos problemas han sido derivados de TOP,

que a grandes rasgos constituyen variaciones donde se agregan restricciones a las dimensiones existentes en el problema. Por ejemplo, existe Team Orienteering Problem With Time Window, abreviado como TOPTW (Igualmente conocido como Selective Vehicle Routing Problem with Time Windows, SVRPTW [4]), donde se tiene un cierto margen de tiempo para visitar un nodo, lo cual aplica mas realistamente a un itinerario de un paquete turístico o a un itinerario de actividades en un evento. También, otra variante del problema es Capacitated Team Orienteering Problem (CTOP [5]) cuya primicia recae en que los clientes además poseen una demanda que cumplir en cada visita, mientras que los vehículos poseen una capacidad máxima de carga, lo cual se puede reflejar en la industria con un servicio de distribución. Finalmente, cabe destacar que así como existen variaciones a TOP, existen variaciones como especificaciones de variantes del mismo, o mezcla de variaciones que se convierten en una variación de por sí, como es el caso del Multiconstraint Team Orienteering Problem with Multiple Time Windows (MC-TOP-MTW), por lo que existen varias formas de especializar el problema y varios son arduamente trabajados debido a su aplicación en la logística de la industria actual.

3. Estado del Arte

En la literatura, la primera aparición del problema TOP aparece sin llamarse como tal en el artículo de Butt y Cavalier [6] sino que bajo el nombre de Multiples Tour de Máxima Colección. El término de TOP es introducido por primera vez en el artículo de Chao et al. [2] donde por primera vez se le realiza el ligado de este problema al deporte de Orienteering y por lo tanto como un derivado de OP. Las instanciaciones utilizadas para el testeo en el artículo de Chao et al. [2] son utilizadas hasta hoy en día para realizar el testeo de nuevos acercamientos al problema.

Para todo problema de este tipo existen dos formas de buscar soluciones, a través de métodos exactos y de métodos inexactos, donde estos últimos contemplan heurísticas y metaheurísticas. Como diferencia entre ambos se puede destacar entre todo las limitaciones que tienen, ya que un método exacto entregará óptimos globales pero a una cantidad discreta de instanciaciones del problema en tiempo razonable, mientras que un método inexacto puede ser mas rápido y abordar mas instancias pero lo hace a costo de que el óptimo que haya encontrado sea local y no global. Es por esto que se revisará el estado del arte para ambos métodos y los trabajos previos para las implementaciones mas eficientes actuales.

3.1. Metodología Exacta

En acercamientos exactos, es común ver que para obtener mejores resultados no tan sólo se buscan varaiciones de algoritmos de búsqueda combinatorial, sino que además en la mayoría de las ocaciones se acompaña con una reformulación del modelo matemático que describe el problema. El primer acercamiento con metodología exacta del cual se logró obtener información fue del artículo de Butt y Ryan [7] donde proponen utilizar el algoritmo de generación de columnas para obtener soluciones globales óptimas para el problema. Sin embargo, este método sólo puede resolver una cantidad muy limitada de instancias del problema en un tiempo razonable. Es por esto que en su trabajo, Boussier, Feillet y Gendreau [3] utilizan una metodología basada en el algoritmo de ramificación y precio, permitiendo resolver instancias mas grandes con hasta 100 vértices. En el trabajo de Poggi, Viana y Uchoa [8] logran generar un método competitivo con el anterior reformulando el problema y luego aplicando algoritmos de ramificación corte y precio. Lo más actual es el trabajo de Bianchessi, Mansini y Garzia Speranza [9] donde no sólamente resuelven de forma óptima 10 instancias más que la mejor anterior, utilizando su implementación de hebra simple y 26 para la implementación de multi-hebras, sino que además resuelve 24 instancias previamente sin resolver. Para esto, lo que hacen es reformular el problema de forma de matemáticamente modelarlo como un problema de dos índices y cantidad polinómica de variables, las cuales se agrupan en variables binarias, enteras y continuas; donde una solu-

ción viable se representa con un arreglo binario de tamaño constante para indicar si el nodo es visitado o no, una matriz cuadrada binaria que representa los arcos y si estos son utilizados o no, y una matriz con valores continuos que indican el momento de llegada al pasar por ese arco; para luego introducir en el modelo restricciones de conectividad, y así resolviéndolo utilizando algoritmos de ramificación y corte. Ya que la cantidad de restricciones de conectividad crece exponencialmente con respecto a la cantidad de puntos de control, estos son agregados dinámicamente al árbol de ramificar-y-enlazar. Tanto el modelo originalmente utilizado como la refomulación de este último artículo se encontrarán presentes en la Sección 4.

3.2. Metodología Inexacta

Debido a la naturaleza NP-difícil de este problema, la mayoría de los esfuerzos de investigación sobre el se concentran en el uso de heurísticas y metaeurísticas, razón por la cual se encuentra una mayor cantidad de artículos con variedad de alogoritmos utilizados en el ámbito del acercamiento inexacto. En su trabajo, Butt y Cavalier [6] proponen un procedimiento greedy, cuando el problema aún no se conocía como TOP. A su vez Chao et al. [2] cuando propuso el nombre de TOP para este problema, propuso un método de cinco pasos y una heurística basada en el algoritmo estocástico de Tsiligirides [10]. Búsqueda tabú fue propuesta por Tang y Miller-Hooks [11], mientras que Archetti, Hertz y Speranza [12] proponen un algoritmo de búsqueda de dos tabúes y un algoritmo de búsqueda con vecindario cambiante. Ke, Archetti y Feng [13] fueron los primeros en utilizar algoritmos de metaheurísticas basados en la metodología de enjambre (swarming) donde a través de la técnica de Ant Colony Optimization [14] (ACO) se aplica en base a cuatro métodos por sobre el problema de TOP, un método secuencial, un método concurrente dividido en determinista y aleatorio, y un método simultaneo. En su trabajo, el cual ellos denominan ACO-TOP, obtienen resultados que a la fecha competían eficiente y efectivamente contra los que ya existían, donde el método secuencial destacaba por obtener la mejor calidad de resolución en menos de un minuto por cada instancia. Posterior a ACO-TOP, Vansteenwegen et al. [15] proponen un algoritmo de búsqueda local guiada y un algoritmo de vecindario con variable sesgada. Bouly et al. [16] luego proponen un algoritmo memético, que es básicamente un algoritmo genético con métodos híbridos dedicados específicamente a TOP, un procedimiento dividido óptimo para la evaluación cromosómica, y técnicas de búsqueda local para la mutación. Dang, Guidabj y Moukrim [17] proponen un algoritmo inspirado en enjambramiento de particulas logrando reducir el error relativo con respecto a las evaluaciones con metodologías exactas de un 0.0005 %, por lo cual es considerado como uno de los algoritmos mas efectivos a la fecha, a pesar de no ser tan eficiente en términos de tiempo de ejecución. Además, introdujeron un nuevo conjunto de instancias más grande para benchmarking. Recientemente Ke, Zhai, Li y Chan [18] introdujeron una nueva metaheurística basada en la metaheuristica de población, aplicándola por primera vez sobre un problema TOP, por lo que se le conoce como el acercamiento mas grande a resolver TOP, utilizando lo que ellos denominaron como un Algoritmo de Imitación de Pareto (Pareto mimic algorithm PMA) donde utilizan un nuevo operador, un operador de imitación, para generar una nueva solución representada como un arreglo de tamaño m con m caminos, al simplemente imitar una solución titular. Igualmente adopta un operador swallow, de forma de tragar, o insertar, un nodo inviable y luego reparar la solución inviable resultante. [19] Finalmente, Ke et al. [18] lograron obtener diez mejoras a las soluciones existentes ya conocidas para las instancias de benchmark de Chao et al. [2] y de Dang et al. [17].

3.3. Orientación de los Esfuerzos Actuales

Se observa cláramente una tendencia a reformular matemáticamente el modelo o las heurísticas con las que se trabaja el problema, la tendencia actual se orienta a tratar de abordar el problema desde nuevas perspectivas en vez de tratar de reimplementar métodos a modelos ya

existentes de alguna forma óptima de acuerdo a lo que el poder computacional o de estructuración de código permita.

4. Modelo Matemático

Como mencionado en la Sección 3, existen dos modelos matemáticos para este problema dignos de mencionar. El primero correspondería a la representación general del problema utilizado en la gran mayoría de los articulos revisados, el cual es extraído del articulo de Ke et al. [13]. Mientras que el segundo modelo a presentar corresponde al utilizado por Bianchessi et al. [9] para generar la solución que ellos plantean.

4.1. Modelo General

Parámetros:

- $G = (V, E)$ un grafo completo donde $V = \{1, ..., n\}$ es un conjunto de vértices y $E = \{(i, j)|i, j \in V\}$ conjunto de arcos que los unen.
- S_i puntaje asociado al nodo $i \in N$ donde $S_1 = S_n = 0$
- C_{ij} costo asociado a recorrer el arco $(i, j) \in E$, que también se puede referir a las distancia entre los nodos.
- T_{max} es el tiempo o distancia máxima que puede demorar cada ruta.
- m cantidad de participantes, que definen la cantidad de rutas.

Variables:

- y_{ik} $(i = 1, ..., n;\ k = 1, ..., m)$ variable binaria que toma el valor de 1 si el vértice i es visitado en la ruta k, 0 sino.
- x_{ijk} $(i, j = 1, ..., n;\ k = 1, ..., m)$ variable binaria que toma el valor de 1 si el arco (i, j) es visitado en la ruta k, 0 sino. Ya que $C_{ij} = C_{ji}$, sólo está definido $x_{ijk}(i < j)$.
- U es un subconjunto de V.

Función Objetivo:

$$Max \sum_{i=2}^{|N|-1} \sum_{k=1}^{m} S_i y_{ik} \tag{1}$$

El objetivo de la función es maximizar la suma del puntaje obtenido entre las rutas.

Restricciones:

$$\sum_{j=2}^{n}\sum_{k=1}^{m} x_{1jk} = \sum_{i=1}^{n-1}\sum_{k=1}^{m} x_{ink} = m \tag{2}$$

Esta restricción obliga que toda ruta parta del nodo de inicio y llegue al nodo de llegada.

$$\sum_{i<j} x_{ijk} + \sum_{i>j} x_{jik} = 2y_{jk} \ (j = 2, ..., n-1;\ k = 1, ..., m) \tag{3}$$

Esta restricción asegura la conectividad de cada ruta.

$$\sum_{k=1}^{m} y_{ik} \leq 1 \ (i = 2, ..., n-1) \tag{4}$$

Esta restricción asegura que cada vértice, a excepción del de inicio y de llegada, sea visitado a lo mas una vez.

$$\sum_{i=1}^{n-1}\sum_{j>k} C_{ij}x_{ijk} \leq T_{max} \ (k = 1, ..., m) \tag{5}$$

Esta restricción limita el tiempo o la distancia a recorrer de cada ruta.

$$\sum_{i,j\in U} x_{ijk} \leq |U| - 1 \ (U \subset V \setminus \{1, n\};\ 2 \leq |U| \leq n-2;\ k = 1, ..., m) \tag{6}$$

Esta restricción se asegura que las subrutas estén prohibidos.

$$x_{ijk} \in \{0, 1\} \ (1 \leq i < j \leq n;\ k = 1, ..., m) \tag{7}$$

$$y_{1k} = y_{nk} = 1,\ y_{ik} \in \{0, 1\} \ (i = 2, ..., n-1;\ k = 1, ..., m) \tag{8}$$

La equaciones (7) y (8) aseguran la naturaleza de las variables del problema.
Se deduce que el espacio de búsqueda de este modelo debe ser del orden de $O(2^{n^3})$ ya que correspondería a una permutación binaria de una matriz tridimensional.

4.2. Modelo de Bianchessi et al.

Parámetros:

Los parámetros de este modelo son los mismos que los presentados en 4.1, por lo que se presentarán las variables, función objetivo y restricciones conforme a estos. Sólo, y por comodidad se considera un parámetro como una especificación de otro, donde N son los vértices intermedios del de inicio y el de llegada, es decir $N = V - \{v_1, v_n\}$

Variables:

Las variables binarias x_{ij} para cada $(i, j) \in E$ e y_i donde $i \in N$ toman valor 1 si el arco (i, j) es usado y el nodo i es visitado, 0 sino.
z_{ij} $(i, j) \in A \setminus \{1, n\}$ es una variable continua que representa la distancia que ha recorrido una ruta al momento de llegar a un vértice j viniendo de un vértice i.
Para cada conjunto $S \subseteq N$, se definen $\delta^+(S) = \{(i, j) \in A : i \in S, j \notin S\}$ y $\delta^-(S) = \{(i, j) \in A : i \notin S, j \in S\}$ como los conjuntos de arcos saliendo y entrando el conjunto S respectivamente, con $\delta^+(i) = \delta^+(\{i\})$ y $\delta^-(i) = \delta^-(\{i\})$.

Función Objetivo:

$$Max \sum_{i\in N} S_i y_i \tag{9}$$

Maximiza la suma de los puntajes de los nodos visitados.

Restricciones:

$$\sum_{j \in N} x_{1j} = \sum_{i \in N} x_{i,n} = m \tag{10}$$

Esta restricción asegura que la cantidad de rutas que salgan, sean los mismos que lleguen y tienen que ser m.

$$\sum_{(j,i) \in \delta^-(i)} x_{ji} = \sum_{(i,j) \in \delta^+(i)} x_{ij} = y_i \tag{11}$$

Esta restricción asegura que la cantidad de rutas que fluyan por el i-ésimo nodo sea a los mas uno.

$$z_{0j} = C_{0j} x_{0j} \tag{12}$$

Esta restricción asegura que el flujo se origine del nodo inicial.

$$\sum_{(i,j) \in \delta^+(i)} z_{ij} - \sum_{(j,i) \in \delta^-(i)} z_{ji} = \sum_{(i,j) \in \delta^+(i)} C_{ij} x_{ij} \tag{13}$$

Esta restricción asegura la continuidad del flujo actualizando el valor de llegada a cada nodo.

$$z_{ij} \leq (T_{max} - C_{j,n}) x_{ij} \ ((i,j) \in E \setminus \{1,n\}) \tag{14}$$

Esta restricción asegura que el costo de cada flujo resultante no sea mayor al máximo permitido.

$$z_{ij} \geq (C_{1i} + C_{ij}) x_{ij} \ ((i,j) \in E \setminus \{1,n\}) \tag{15}$$

Esta restricción impone un límite inferior en los valores que tome la variable **z** de forma que se restrinjan los rango de los valores posibles que esta variable pueda tomar en soluciones intermedias. Notar que si $x_{ij} = 1$, entonces z_{ij} denota la distancia de llegada al nodo j, mientras que $x_{ij} = 0 \implies z_{ij} = 0$.

$$y_i \in \{0,1\} \ (i \in N) \tag{16}$$

$$x_{ij} \in \{0,1\} \ ((i,j) \in E \setminus \{1,n\}) \tag{17}$$

$$0 \leq x_{1n} \leq m \tag{18}$$

$$\sum_{(i,j) \in A \setminus \{(1,n)\}} C_{ij} x_{ij} \leq m \times T_{max} \tag{19}$$

Las ecuaciones (16), (17) y (18) restrinjen la naturaleza de las variables, mientras que (19) es una restricción hecha para fortalecer la formulación e impone una duración máxima global a las rutas.
El espacio de búsqueda de este modelo tiene un orden de $O(2^{n^2})$ ya que corresponde a la permutación de una matriz binaria.

5. Representación

La representación utilizada en el algoritmo propuesto para las soluciones en esta entrega se deriva principalemente de una matriz de adyasencia que indican los arcos que conforman cada una de las rutas, de la cual junto a otras variables se construyen m listas con los nodos que conforman la ruta, una lista de tamaño m con el tiempo que se demora cada ruta y un puntaje asociado a la solución.

Esta solución se genera a partir de la extracción de datos que entregan tres variables que van cambiando durante la ejecución del algoritmo, las cuales son una lista binaria de tamaño

n que indica si el i-ésimo vecino fue visitado o no, una matriz binaria x de $n \times n$ que indica si el arco (i, j) fue utilizado, y una matriz de valores continuos z de tamaño $n \times n$ que indica el tiempo de llegada de al nodo i desde el nodo j.

El espacio de búsqueda viene dado a lo mas por la permutación de una matriz binaria de adyasencia de tamaño $n \times n$, lo cual nos da un orden máximo de $O(2^{n^2})$ pero que se ve disminuido dependiendo de la cantidad de rutas que tenga la isntanciación y dependiendo del tiempo máximo de cada ruta, lo cual igualmente nos va a limitar la cantidad de soluciones factibles.

6. Descripción del algoritmo

Debido a la naturaleza del problema, realizar BackTrack con GBJ era lo mismo que hacer un BackTrack completo, debido a que el grafo de restricciones era totalmente conexo, razón por la cual la implementación que se presenta a continuación es la de un BackTracking completo y no existe una función de GBJ.

Una vez leidos todos los parámetros se genera una lista de vértices, los cuales corresponden a clases con posición x, posición y, y un puntaje asociado, y una matriz de ejes de tamaño $n \times n$ con la distancia euclidiana entre (i, j) que es igual a (j, i), que luego se pasan por referencia a una clase instancia del problema la cual posee referencias a todas las variables de forma de utilizarlas como variables globales al momento de ejecutar el algoritmo e intentar mantener el espacio de memoria utilizada por las variables constante.

El algoritmo de Backtrack se hace realizando una recursión con paso de un parámetro entero sin declariación de variables dentro de el y con llamado a funciones para ser inmediatamente evaluadas, por lo cual la memoria stack utilizada por el algoritmo recursivo es a lo mas un árbol de profundidad $log(n)$ con a lo mas m hijos por nodo raíz relativo, lo cual genera una cantidad de hojas de $m^{log(n)}$.

El algoritmo recibe como parámetro un índice i y evalúa como caso base si este i corresponde al índice $n - 1$ para saber si es que ya se recorrieron todos los i-ésimos vecinos anteriores, luego llama a un chequeo para ver si del n-ésimo nodo se pueden conectar los nodos que hasta el momento generan las rutas sin pasarse del tiempo máximo permitido, en caso de poderse, se verifica si la solución actual es mejor que la existente y en caso de no ser así simplemente se ignora. Luego se eliminan todos los cambios hechos en la matriz al intentar conectar el último nodo con los de las rutas hasta el momento y se retorna al nivel anterior de recursión. El segundo caso que el algoritmo verifica es si es que no existen las cantidades de rutas que la instancia pide, en cuyo caso intenta conectar el i-ésimo nodo actual al nodo de origen, si puede lo conecta y llama a la recursión para el siguiente índice y al volver deshace el cambio. Finalmente el tercer caso se verifica independiente de los otros dos, de forma que al volver al nivel recursivo del segundo caso habiendo sido exitoso, de todas formas se evalúa el tercer caso, utilizando así el i-ésimo nodo ya sea, sin necesidad de que hayan m rutas, o habiéndolas, evalúandolo de todas formas si es que no ha sido visitado, para esto se verifica si este nodo se puede conectar con los últimos nodos que componen las rutas actuales, de forma de mantener la conectividad y la cantidad de rutas actuales, evitando así que existan subrutas discontinuas o con mayor grado de salida que 1. En caso de poder conectarse con alguno lo hace, para luego al volver deshacer los cambios, para luego igualmemente seguir con el siguiente índice sin haberse conectado.

A continuación se muestra un pseudo-código de formageneral que explica el algoritmo recursivo:

```
Data: routesAmm = 0, addRoutes = true
Result: Best route for instance
begin bTrack(i);
if i is end node then
    for last nodes in routes do
        try to connect with end node;
    end
    if connections possible then
        if this is best solution then
            update best solution;
        else
            ignore it;
        end
    end
end
else if addRoutes AND i is not visited then
    if connection to initial node possible then
        connect;
        bTrack(i+1);
        undoChanges;
        /* Connecting and undoing updates routesAmm and addRoute          */
    else
        bTrack(i+1)
    end
end
if i is not visited then
    for last nodes in routes do
        if connection possible then
            connect;
            bTrack(i+1);
            undo;
        end
    end
    bTrack(i+1)
end
/* Initial call is done with bTrack(1)                                    */
```

A nivel de implementación, las conexiones se hacen al verificarse, es decir, invoca a una función que retorna verdadero si es se pudo actualizar las matrices x y z, al igual que el arreglo y indicando que se agregó el nodo, y falso si no se pudo. Ya que las conexiones y las verificaciones son en tiempo constante debido a que se acceden a índices específicos de arreglos fijos, no agrega mucha complejidad a la recursividad.

La función de recursividad de este algoritmo viene representada de forma muy general como:

$$T(k) = mT(k+1) + m$$
$$T(n) = m$$

Lo cual indica que aproximadamente el tiempo de ejecución del algoritmo es de $O(m^n)$ sin considerar el impacto que tiene el tiempo máximo de cada ruta.

7. Experimentos

Los experimentos se realizaron en un Notebook HP Pavilion 15-cx0003la, con procesador Intel Core i5 de 2,3 GHz, 8GB de RAM y en sistema operativo Ubuntu 18.04LTS.

Los parámetros se definieron según los trata la literatura, teniendo una cantidad n de nodos en el problema, una cantidad m de rutas, un tiempo $Tmax$ que restringe las rutas, y posteriormente la ubicación cartesiana de los puntos y su puntaje correspondiente. Como ya mencionado anteriormente, se almacenaron en un arreglo de vertices con puntaje asociado y una matriz de adyasencia cuyo valor (i, j) es la distancia euclideana entre i y j.

Ya que el tiempo de ejecución del algoritmo implementado igualmente es alto y hasta el último momento de implementación del código se obtenían algunos resultados erroneos, solamente se recolectó datos del set de instancias de 24 nodos que existen actualmente en la literatura, de los cuales se obtuvo un mejor puntaje según el algoritmo y el tiempo de ejecución que se demoró en obtenerlo. Otros sets igualmente fueron testeados, pero ya que por razones de tiempo no se pudieron probar todas las instancias de estos, no se recolectaron los datos.

8. Resultados

Los resultados y sus comparaciones se muestran en el Cuadro 1, donde se evidencia el alto impacto que Tmax tiene sobre el tiempo de ejecución del algoritmo, ya que como es Backtrack, inmediatamente desecha las ramas donde se generan soluciones no válidas, al aumentar Tmax, aumentan las soluciones válidas y por lo tanto se recorre mas el espacio de búsqueda completo.

Además notamos como para las instancias p2.2.d, p2.2.e, p2.2.j, p2.2.k y p2.3.h no se obtuvieron rutas óptimas, lo cual representa a un 85 % de efectividad con respecto de la solución mostrada en Boussier et al[3], y obteniendo en un 61 % de los casos un tiempo de ejecución considerablemente peor según la notación y las sifras significativas utilizadas.

Esta diferencia en resultados de rutas óptimas se debe a probablemente un error en la implementación, se utiliza hasta el último momento para revisar el algoritmo pero hasta el momento no se ha logrado identificar donde recae el problema por el que se ignoran estas soluciones. En cuanto al tiempo de ejecución, es de esperarse que un Backtrack se demore esta cantidad de tiempo y aumente de esa forma cuando la cantidad de soluciones factibles aumenta, y aunque se revisó persistentemente la implementación buscando donde podría mejorarse el tiempo de ejecución, las mejoras posibles sólo implicaban una mejora en cantidad constante y poco significativa para la ejecución final del problema.

Instance	m	Tmax	Boussier 2007		BackTrack	
			Score	CPU(s)	Score	CPU(s)
p2.2.a	2	7.5	90	0	90	0
p2.2.b		10	120	0	120	0
p2.2.c		11.5	140	0	140	0
p2.2.d		12.5	160	0	150	0
p2.2.e		13.5	190	0	170	1
p2.2.f		15	200	0	200	2
p2.2.g		16	200	0	200	2
p2.2.h		17.5	230	0	230	5
p2.2.i		19	230	0	230	9
p2.2.j		20	260	1	230	14
p2.2.k		22.5	284	0	240	34
p2.3.a	3	5	70	0	70	0
p2.3.b		6.7	70	0	70	0
p2.3.c		7.7	105	0	105	0
p2.3.d		8.3	105	0	105	0
p2.3.e		9	120	0	120	1
p2.3.f		10	120	0	120	2
p2.3.g		10.7	145	0	145	4
p2.3.h		11.7	165	0	160	9
p2.3.i		12.7	200	0	200	24
p2.3.j		13.3	200	0	200	36
p2.3.k		15	200	0	200	122
p2.4.a	4	3.8	10	0	10	0
p2.4.b		5	70	0	70	0
p2.4.c		5.8	70	0	70	0
p2.4.d		6.2	70	0	70	0
p2.4.e		6.8	70	0	70	0
p2.4.f		7.5	105	0	105	1
p2.4.g		8	105	0	105	2
p2.4.h		8.8	120	0	120	4
p2.4.i		9.5	120	0	120	11
p2.4.j		10	120	0	120	18
p2.4.k		11.2	180	0	180	78

Cuadro 1: Resultados comparados con los de Boussier et al.[3]

9. Conclusiones

Respecto a aproximaciones anteriores, no tan solamente la metodología utilizada juega un rol importante en la calidad de las soluciones y en el tiempo en que estas se obtienen, sino que además es relevante las representaciones que se utilizan para esto. Es así como en Bianchessi et al.[9] notamos importantes resultados nuevos en cuanto a benchmarking y a la obtensión de nuevos óptimos para considerar globales al cambiar el modelo matemático del algoritmo exacto para que este funcione con variables de a lo sumo dos indices de dimensión. Mientras que en Ke et al.[18] notamos que en busquedas locales se logran mejores resultados al cambiar las operaciones internas del algoritmo al momento de elegir nuevas posibles soluciones para diversificar e intensificar, lo cual llevó a la creación de una nueva metaheurística. Aunque la mayoría de los papers visitados en este documento resuelven instancias del problema TOP, ya que así fueron buscados los documentos, no todos logran obtener resultados favorables en

cuantro a encontrar nuevos resultados globales o tiempos de ejecución notoriamente mas rápidos, en general muestran nuevas aproximaciones competitivas con respecto a los mejores, es tan sólo en los últimos dos papers mencionados recientemente donde se logran resultados significativos en años, debido a, como ya se mencionó, la utilización de nuevas representaciones, modelos, y metaheurísticas.

Respecto a este acercamiento, la elección de BackTrack con GBJ terminó llevando a la implementación de un BackTrack común y corriente, debido a que como el problema posee un grafo de restricciones totalmente conexo, el salto al paso anterior parece tan bueno como cualquier otro para que cumpla con el requisito de ser un salto GBJ. Quizás CBJ podría haber resultado mas eficiente, debido a que por la naturaleza de los caminos, al momento del algoritmo revisar las conexiones finales habría solamente deshecho los últimos vértices que visitan las rutas conflictivas, lo cual podría haber reducido tiempo de ejecución de manera considerable, aunque a medida que se aumenta el tiempo máximo de cada ruta ambas heurísticas de salto habrían convergido igualmente al mismo tiempo de ejecución.

Aunque el algoritmo propuesto entrega soluciones a la mayoría de los casos de prueba existente en la literatura en tiempos que bordean las horas, no está entregando resultados similares a los que de referencia debiese dar, es por esto que aunque se considera un buen acercamiento para obtener resultados globales, no está bien logrado. A pesar de esto, se considera dentro de las posibles implementaciones de BackTrack para este problema, como una implementación competitiva y hasta mas eficiente que la media al ocupar depresentaciones de a lo mas dos índices, inspirándose en el modelo propuesto por Bianchessi et al.[9].

Para mejorar este algoritmo, una buena forma sería arreglándolo para que entregue las soluciones globales en las intancias que no están correspondiendo, lo cual se deduce es debido a un problema puntual de implementación y no del algoritmo en si, ya que revisión tras revisión se ha verificado que la recursión en su para recorrer el espacio de búsqueda está bien hecho. Respecto al tiempo de ejecución, no es posible al momento de entregar este informe determinar alguna forma para mejorarlo.

10. Referencias

[1] Bruce L. Golden, Larry Levy, and Rakesh Vohra. The orienteering problem. *Naval Research Logistics (NRL)*, 34(3):307–318, 1987.

[2] I-Ming Chao, Bruce L. Golden, and Edward A. Wasil. The team orienteering problem. *European Journal of Operational Research*, 88(3):464 – 474, 1996.

[3] Feillet Boussier and Gendreau. An exact algorithm for team orienteering problems. *4OR*, 5:211 – 230, 2007.

[4] Cyrille Gueguen. *Méthodes de résolution exacte pour les problèmes de tournées de véhicules*. PhD thesis, Châtenay-Malabry, Ecole centrale de Paris, 1999.

[5] C Archetti, D Feillet, A Hertz, and M G Speranza. The capacitated team orienteering and profitable tour problems. *Journal of the Operational Research Society*, 60(6):831–842, 2009.

[6] Steven E. Butt and Tom M. Cavalier. A heuristic for the multiple tour maximum collection problem. *Computers & Operations Research*, 21(1):101 – 111, 1994.

[7] Steven E. Butt and David M. Ryan. An optimal solution procedure for the multiple tour maximum collection problem using column generation. *Computers & Operations Research*, 26(4):427 – 441, 1999.

[8] Marcus Poggi, Henrique Viana, and Eduardo Uchoa. The Team Orienteering Problem: Formulations and Branch-Cut and Price. *10th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'10)*, 14:142–155, 2010.

[9] Mansini Bianchessi and Garzia Speranza. A branch-and-cut algorithm for the team orienteering problem. *International Transactions in Operational Research*, 25(2):627–635, 2018.

[10] T. Tsiligirides. Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, 35(9):797–809, Sep 1984.

[11] Hao Tang and Elise Miller-Hooks. A tabu search heuristic for the team orienteering problem. *Computers & Operations Research*, 32(6):1379 – 1407, 2005.

[12] Claudia Archetti, Alain Hertz, and Maria Grazia Speranza. Metaheuristics for the team orienteering problem. *Journal of Heuristics*, 13(1):49–76, Feb 2007.

[13] Liangjun Ke, Claudia Archetti, and Zuren Feng. Ants can solve the team orienteering problem. *Computers & Industrial Engineering*, 54(3):648 – 665, 2008.

[14] Marco Dorigo and Mauro Birattari. Ant colony optimization. *Encyclopedia of machine learning*, pages 36–39, 2011.

[15] Pieter Vansteenwegen, Wouter Souffriau, Greet Vanden Berghe, and Dirk Van Oudheusden. A guided local search metaheuristic for the team orienteering problem. *European Journal of Operational Research*, 196(1):118 – 127, 2009.

[16] Hermann Bouly, Duc-Cuong Dang, and Aziz Moukrim. A memetic algorithm for the team orienteering problem. *4OR*, 8(1):49–70, Mar 2010.

[17] Duc-Cuong Dang, Rym Nesrine Guibadj, and Aziz Moukrim. An effective pso-inspired algorithm for the team orienteering problem. *European Journal of Operational Research*, 229(2):332–344, 2013.

[18] Liangjun Ke, Laipeng Zhai, Jing Li, and Felix T.S. Chan. Pareto mimic algorithm: An approach to the team orienteering problem. *Omega*, 61:155 – 166, 2016.

[19] Aldy Gunawan, Hoong Chuin Lau, and Pieter Vansteenwegen. Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, 255(2):315 – 332, 2016.