Inteligencia Artificial

Informe Final: Team Orienteering Problem

Alejandro Vilches Cornejo

14 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

El Team Orienteering Problem (TOP) es una extensión del Orienteering Problem (OP). Este consiste en que un equipo formado por varios miembros, los cuales inician en un punto de control específico, deben visitar tantos puntos como puedan dentro de una ventana de tiempo, llegando finalmente todos al punto de control final. De esta forma se busca maximizar el puntaje obtenido al pasar por cada punto, sin sobrepasar la ventana de tiempo y eligiendo la mejor ruta para cada miembro, considerando que esta tenga el menor *overlap* posible con puntos previamente visitados. En este informe se analizará y comparará diferentes resoluciones y heurísticas para este problema a lo largo de los años, además de desarrollar una resolución por medio de la heurística *Simulating Anealing*, analizando los resultados obtenidos.

1. Introducción

El Team Orienteering Problem (TOP) consiste de manera general en un grupo de puntos de control dado, donde cada uno posee un puntaje asociado. La meta es determinar para un grupo de personas las rutas a tomar por cada uno, de tal forma de acumular el mayor puntaje posible, considerando que todos deben llegar a la meta dentro de una ventana de tiempo y que pasar por un punto solo otorgará puntaje al grupo la primera vez que cualquier miembro lo visite. La resolución a esta problemática es muy útil para empresas encargadas en paquetes de viajes y recorridos turísticos o empresas que tengan que realizar visitas técnicas a sus clientes y deban organizar una flota de trabajadores para poder visitar la mayor cantidad de casas posibles dentro del día.

En este informe se estudiará este problema, con el fin de poder conocerlo y comparar soluciones existentes para este mismo, plantear un modelo que se ajuste de la mejor forma para

su resolución y finalmente, resolver el problema comparando los tiempo y resultados obtenidos para diferentes instancias. El trabajo estará separado diferentes secciones, primero se definirá el problema de manera general (sección 2), con sus respectivas variables, restricciones y objetivos. En la sección 3 se expondrá diferentes métodos y heurísticas que se han realizado a lo largo de los años para este problema, destacando el tipo de resolución y analizando cuan efectiva fue esta para el problema. Para la sección 4, se definirá un modelo matemático acorde al problema, dejando en claro los motivos de su elección por sobre otros. La sección 5 presenta la representación considerada para las soluciones, explicando cómo se verán y el por qué se eligió este tipo de representación por sobre a otras. Luego en la sección 6 se explicará el algoritmo de *Simulating Anealing* el cual fue usado para resolver el problema, haciendo mención especifica a diferentes partes de este, las cuales son importante de revisar en profundidad cómo se trabajaron. En la sección 7 se explica la forma en que se experimentó, considerando las instancias, técnicas, y parámetros usados, para luego mostrar los resultados (sección 8) de estos experimentos, tanto en tablas como gráficos de ser necesario. Finalmente, concluiremos el informe (sección 9) respecto de la técnica usada para la resolución, si fue la adecuada, sus ventajas y desventajas en base a los resultados obtenidos, el comportamiento bajo diferentes escenarios, entre otros aspectos.

2. Definición del Problema

El Team Orienteering Problem (TOP) es una extención del Orienteering Problem (OP)[3] el cual consiste en que dado un conjunto de punto de control repartidos por una zona, donde cada uno tiene un puntaje asociado, un jugador debe pasar por la mayor cantidad de puntos de control, acumulando la mayor cantidad de puntaje, hasta llegar a la meta, todo eso sin sobrepasar una ventana de tiempo dada. Ganará el jugador que logre obtener la mayor cantidad de puntos en su recorrido y a su vez su carrera cumpla con la ventana de tiempo.Por lo anterior, cualquier jugador que sobrepase el tiempo máximo, quedará descalificado y no será considerado.

El TOP consiste en seguir las mismas reglas que el OP, pero ahora ya no será solo un jugador el que esté presente, sino que será un grupo de ellos en forma de un solo equipo. La tarea esta vez es llegar a la meta acumulando la mayor cantidad de puntos como grupo, considerando que todos deben respetar la ventana de tiempo y que el paso por un mismo punto de control solo otorgará 1 vez el puntaje al grupo. Cabe destacar que el no cumplimiento del tiempo por parte de cualquier miembro del equipo, llevara a su descalificación y no serán considerados.

El problema se puede modelar como un problema de optimización multi-nivel, de tal forma que en el primer nivel tenemos que seleccionar un subconjunto de puntos que el equipo debe visitar. En el segundo nivel se debe asignar los puntos a cada miembro del equipo. Por ultimo en el tercer nivel, se debe construir un camino a través de los puntos asignados a cada miembro del equipo. Para poder representar el TOP utilizaremos un conjunto de puntos conectados entre ellos. Cada punto tiene un puntaje asociado, el cual será mayor o igual a cero. El número de puntos sera N donde la partida corresponde al punto 1 y la meta al N-ésimo punto. Para cada conexión entre los puntos existirá un costo asociado a este, el cual indicará el la distancia del punto i al j, o el tiempo que se tarda en viajar cualquier miembro entre los dos puntos. El equipo estará conformado por M miembros, por lo que es necesario generar M rutas, donde cada una inicie en el punto 1 y termine en el N-esimo, de tal forma de que el puntaje acumulado por el equipo sea máximo, además en necesario considerar que en cada ruta no aparezca el mismo punto mas de una vez. Por último, el tiempo máximo lo denotaremos por T_{max}, donde este corresponderá al valor máximo que pueden tardar cualquiera de las M rutas generadas [1].

El TOP deriva del OP, por lo que si vemos esto desde el otro lado, el OP sería un caso específico de un TOP pero con solo un miembro en el equipo. De esta forma el Traveling Salesman Problem (TSP) [8] puede ser visto como una particularización del TOP con un solo miembro,

considerando las ciudades como los puntos, otorgándole puntaje a cada ciudad visitada y realizando la modificación de que al final del día no vuelva al punto de partida, sino que termine en una ciudad específica dentro de una ventana de tiempo más extensa que abarque un día completo. Tanto para OP como para TOP existe una de las variantes más conocidas, que es considerando ventanas de tiempo (OPTW y TOPTW) [3], la cual consiste en que a cada vértice se le asigna una ventana de tiempo y visitar este vértice solo puede hacerse una vez iniciada esta ventana. Otra de las variantes conocidas del TOP se ve en los servicios de guías turísticas, las cuales ofrecen al cliente la posibilidad de personalizar sus viajes, teniendo como objetivo maximizar el interés del cliente hacia lugares atractivos sujetos al tiempo que se quedarán en ellos, estos problemas son llamados Turist Trip Desing Problem (TTDPs) [8]. Por último, otro ejemplo de problema que se puede resolver mediante una variación del TOP es el problema de entrega de gasolina a las casas [1], de modo que la urgencia por gasolina será visto como el puntaje por cada vértice (casas), cada vehículo de la flota debe tener asignado un recorrido, de tal forma de poder maximizar el puntaje total obtenido, o viéndolo desde el sentido del problema, maximizar la cantidad gasolina entregada a las casas con mayor urgencia.

3. Estado del Arte

Team Orienteering Problem, tiene un origen reciente propuesto por Butt y Cavalier en 1994 con el nombre de *Multiple Tour Maximun Collection Problem*[7]. Dos años mas tarde, I-Ming Chao introdujo formalmente el problema [1], generando así el conjunto de instancias de referencia mas utilizados actualmente. Desde aquí en adelante, muchas heurísticas han sido propuestas para resolver el problema.

En [8] se menciona la primera de ellas, la cual surge en 1999 a cargo de Butt y Ryan, los cuales introducen un procedimiento basado en la formulación de *set covering*.

Para el año 2005 surge otra de las formulaciones más conocidas a cargo de Tang y Miller-Hooks [4], los cuales proponen una resolución en base a *Tabu Search* combinado con un procedimiento de memoria adaptativa (AMP), que permite alternar entre vecindarios pequeños y grandes mientras se genera la solución en la fase de mejora.

En el año 2007 un algoritmo tipo *Branch and Price* (BP) fue propuesto por Feillet y M. Gendreau [9], en el cual se usa un alcance de programación dinámica para solucionar el problema de precios, siendo esta solución muy adaptable a diferentes variables del problema, obteniendo que bajo la prueba de 320 problemas, esta nueva técnica consistentemente produce soluciones de mejor calidad, generando una baja en los tiempos de 29.1 % y 55.7 % tanto para vecindarios pequeños y largos respectivamente.

Luego, en 2008 se presenta una resolución basada en *Colonia de Hormigas* a cargo de Liangjun Ke [6], proponiendo los métodos: secuencial, concurrencia determinista, concurrencia aleatorias y método simultáneo. La heurística fue probada en base a 387 instancias obtenidas gracias a la formulación de Chao, obteniendo así un mejora en 15 de los grupos comparado con otros algoritmos, generando así una nueva heurística bastante prometedora. Además, gracias a su trabajo pudieron concluir que de los 4 métodos probados, el secuencial obtenía los mejores resultados en el menor tiempo.

Año siguiente, Pieter Vansteenwegen (2009) [5] presenta una heurística de *Busqueda Local* para el OP, la cual luego extrapola para la resolución tanto del TOP, como de las variantes con ventanas de tiempo para ambos problemas anteriores, obteniendo asi resultados de la misma

calidad que la mejor heuristica conocida hasta la fecha, sin embargo se redujo el tiempo computacional significativamente.

Luego Marcus Poggi [7] en el año 2010 introduce un modelo pseudo-polinomial lineal para el TOP, además de proponer un algoritmo tipo *Branch cut and Price* (BCP). Este algoritmo inicia con una fase de generación de columnas, para luego resolver el problema de precios y realizar los cortes, utilizando métodos como *min-cut* y *triangle clique*. Esta heurística fue probada con instancias entre 21 y 102 vértices, obteniendo una mejora de 2 minutos con respecto el método realizado por Boussier.

Para el año 2013 Dang [8] propuso un alcance *Branch and cut* (BC) basado en la formulación lineal y presenta un nuevo set de desigualdades validas y propiedades dominantes para acelerar el proceso de solución, obteniendo de esta forma una técnica efectiva que aceleró el proceso, resolviendo un gran y variado numero de instancias, siendo algunas de estas instancias indocumentadas con anterioridad.

Por último en [8] se menciona sobre Keshtkarana, quien mas recientemente presenta en 2016 un algoritmo BP con dos etapas de relajación (BP2R) y un enfoque BPC para resolver el TOP, donde un algoritmo de programación dinámica bidireccional acotado con espacio de estado decremental con relajación se utilizó para resolver los subproblemas.

A continuación, profundizaremos en dos de las técnicas mas comunes que resuelven el TOP mencionadas anteriormente. Es evidente que debemos partir por quien introdujo formalmente el problema en 1996.

Formulación TOP por I-Ming Chao [1]

I-Ming Chao, Bruce L. Golden h y Edward A. Wasil, realizaron la primera formulación del problema en 1996, generando el conjunto de instancias más utilizadas actualmente. Para la generación de esta nueva heurística, ellos modificaron una de las existente para el OP, para asi poder solucionar el TOP. Esta corresponde al algoritmo estocástico de Tsiligrides. Esta nueva heurística consiste en dos pasos: Inicialización y luego Mejora. En la primera se construye una elipse sobre todo el conjunto de puntos considerando el inicio y final como los dos focos de esta y el tiempo máximo (T_{max}) como el largo del mayor de los vértices. De esta forma, cuando construyan los caminos, los cuales serán representados en conjuntos, solo considerarán los puntos que se encuentran dentro de la elipse, puesto que los de afuera no cumplirán con la restricción del tiempo máximo. Para esto se utiliza un método de Greedy, hasta que cada uno de las M rutas esté completa, donde ellos consideraban una ruta como completa cuando al insertarle un nuevo vértice, el tiempo que acumulaba esta superaba el valor de T_{max}, de modo que, se quedaban con las M rutas que sumaran mayor puntaje la cual sería la solución inicial.

Así buscaban construir una solución inicial de manera rápida, para luego en el paso de mejora, poder encontrar una solución que aumente el puntaje de la anterior, para lo que, permiten variar el puntaje del equipo, haciéndolo disminuir con la esperanza de luego encontrar una mejor solución. Para esto, ellos ocupan dos técnicas: *Two-point exchange y One-point movement*. La primero consiste en tomar un vértice del conjunto de vértices no seleccionados en la solución e intercambiarlo por alguno de los que forman parte del conjunto de la ruta solución, así, si el tiempo de la nueva ruta generado producto de este cambio es factible (menor o igual a T_{max}) la inmersión se considera factible y se verifica si el puntaje aumenta o disminuye: en caso de aumentar, el cambio se realiza como definitivo y se ignoran los demás cambios posibles, de no existir un candidato que aumente el puntaje, se considera alguno que disminuya este en la menor cantidad posible. La segunda de las técnicas, consiste en solo mover vértices entre los caminos ya creados. De esta forma, se moverá un vértice i desde su actual posición dentro una de las

rutas, para posicionarlo frente a otro punto dentro de cualquiera de las otras. Igual que para el caso anterior, solo se realizarán los cambios si estos generan una alza en el puntaje del equipo, de no existir ningún cambio que realice lo anterior, se considerará aquel que disminuya en menor cantidad el puntaje actual del equipo.

Utilizando este nuevo método, los autores pudieron evidenciar una mejora en los tiempos de procesamiento y resolución del problema, en comparación con los tiempos logrados por el algoritmo de Tsiligirides (TOH), arrojando que la nueva heurística entrega mejores resultados en un 60 % de los problemas probados por los autores.

Local Search

Esta segunda técnica de resolución surge en 2009, a cargo de Pieter Vansteenwegen [5] y la cual guiará el modelo presente en la sección 4. Esta nueva heurística combina otras para así poder llegar al algoritmo final a utilizar, cada una de estas es usada hasta que se alcanzan un óptimo local. Primero que todo, inicia con una fase de construcción, la cual es idéntica al método de Inicialización planteado por Chao en el punto anterior, utilizando conjuntos para representar las M rutas y los vertices aun no seleccionados. Luego, utilizar las heurísticas de *Insert and Replace*, que consiste en insertar nuevas locaciones a la ruta en la posición donde se consume menor tiempo de viaje. La segunda heurística se basa en el método de *Two-point exchange*, para mejorar las rutas encontradas. Finalmente, se utilizan 3 heurísticas con el fin de poder reducir el tiempo de viaje entre las localidades seleccionadas, que son *Swap, Move y TSP*. Swap consiste en intercambiar dos locaciones entre rutas, con el fin de mejorar el tiempo final; Move consiste en realizar el método *One-point movement* con el fin de poder despejar una ruta, para ingresar mas locaciones por medio de las otras heurísticas; y por ultimo TSP como se conoce, permitirá reducir el tiempo de viaje encontrando el camino más corto para cada ruta.

Este nuevo algoritmo fue probado con un gran número de problemas y comparado contra resoluciones anteriormente mencionadas como *Tabu Search*, el algoritmo de Tsiligrides, inclusive la formulación anteriormente mencionada de Chao, llegando al resultado de que esta formulación mejora en algunos casos, generando una nueva resolución para este problema que supera en muchos casos a las antes formuladas.

En general, se puede ver que no existió una técnica líder, si bien existieron muchas resoluciones en la actualidad basadas en *Branch Cut and Price* y sus derivados, estas no se destacan por sobre el resto de técnicas, cada una mejora en partes la resolución del problema, pero así como tienen ventajas, también poseen falencias, que otras técnicas pueden solucionar, pero también introducir nuevos problemas con ellas. Sin embargo, en algo en lo que se asemejan la mayoría de las soluciones es en la representación del problema, partiendo porque todas lo modelan como un gráfo, donde cada nodo tiene su correspondiente puntaje y los arcos que los conectan corresponden el tiempo que se tarde en viajar de uno a otro, de esta forma, las soluciones vienen representadas por un conjuntos de M rutas, donde cada una de ellas contiene los nodos que son visitados por esta. De aquí cada una de las técnicas trabaja de forma diferente los movimientos y cambios en las soluciones para llegar al resultado final.

4. Modelo Matemático

A continuación, se presentara el modelo matemático para el TOP, explicitando tanto sus variables, parámetros, función objetivo y las restricciones. Guiándose por el trabajo realizado por Vansteenwegen [3], este modelo puede ser formulado como problema entero, de la siguiente forma.

En general, los métodos de solución propuestos a lo largo de los años son variados, hay que destacar que existen muchos estudios que utilizan un método *Branch cut* o *Branch cut and*

Price, sin embargo eso no deja de lado el desarrollo por Tabu Search, Local Search y el método planteado por Chao, el cual fue la base de mucho de las nuevas resoluciones formuladas.

Parámetros

Primero que todo se definen los parámetros obtenidos por medio del contexto del problema.

- N: es el numero de puntos de control o vértices del grafo que modela el problema.
- M: es el numero de miembros del equipo y su vez el numero de rutas a generar, ya que cada miembro necesita exactamente 1 sola ruta.
- T_{max}: es el tiempo máximo que pueden tardar cada una de las M rutas.
- t_{ij}: es el tiempo que cuesta en ir del nodo i al j

Variables

Este poblema puede ser modelado usando de variables enteras, de esta forma se tiene que:

$$x_{ijm} = \begin{cases} 1 & \text{si se va del vértice i al j en la ruta m} \\ 0 & \text{sino} \end{cases}$$

$$y_{im} = \begin{cases} 1 & \text{si voy del vértice i es visitado en la ruta m} \\ 0 & \text{sino} \end{cases}$$

u_{im} = la posición del vértice i en la ruta m,
entiéndase esto como el orden en que se visitarán los puntos dentro de la misma ruta

Función Objetivo

Se busca maximizar la cantidad de puntaje obtenido por el equipo, por lo que la función objetivo es de la forma:

$$Max \sum_{m=1}^{M} \sum_{i=1}^{N-1} s_i \cdot y_{im} \tag{1}$$

Donde como M es el numero de rutas que se desean crear (M miembros) y N sera el numero de puntos de control que existen (N vértices)

Restricciones

La primera de las restricciones a considerar, es el hecho de que todos deben iniciar en el punto 1 (partida) y llegar al punto N (meta):

$$\sum_{m=1}^{M} \sum_{j=2}^{N} x_{1jm} = \sum_{m=1}^{M} \sum_{i=1}^{N-1} x_{iNm} = M \tag{2}$$

La siguiente restricción nos asegura que cada vértice sea visitado a lo mas una vez dentro de una misma ruta.

$$\sum_{m=1}^{M} y_{km} \leq 1 \qquad \forall k = 2, ..., N-1 \tag{3}$$

Otra de la restricciones a considerar es que la conectividad de cada una de las M rutas.

$$\sum_{i=1}^{N-1} x_{ikp} = \sum_{j=2}^{N} x_{kjm} = y_{km} \qquad \forall k = 2, ..., N-1, \quad \forall m = 1, ..., M \tag{4}$$

Para el tiempo máximo que pueden tardar cada una de las M rutas se tiene lo siguiente.

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} t_{ij} x_{ijm} \leq T_{max} \qquad \forall m = 1, .., M \tag{5}$$

Por ultimo, es necesario definir restricciones para que no existan micro ciclos dentro de cada una de las rutas, además de la naturaleza de las variables.

$$2 \leq u_{im} \leq; \qquad \forall i = 2, ..., N; \quad \forall m = 1, ..., M \tag{6}$$

$$u_{im} - u_{jm} + 1 \leq (N-1)(1 - x_{ijp}) \qquad \forall i, j = 2, ...N; \quad \forall m = 1, ..., M \tag{7}$$

$$x_{ijm}, y_{im} \in \{0, 1\} \qquad \forall i, j = 1, ..., N; \quad \forall m = 1, ...M \tag{8}$$

$$u_{im} \in \{0, ..., N\} \qquad \forall i, j = 1, ..., N; \quad \forall m = 1, ...M \tag{9}$$

Espacio de Búsqueda

Considerando este modelo para el TOP, se puede definir el espacio de búsqueda en términos de las 3 variables que se posee, donde x_{ijm} e y_{im} son variables de tipo binarias y u_{im} es de tipo entera positiva. Así el espacio de búsqueda es igual a:

$$2^{N^2 * M} * 2^{NM} * N = N * 2^{N^2 + NM} \tag{10}$$

5. Representación

Para este problema, se considerará que cada uno de los nodos presentes, almacenarán la posición de este (x,y), el puntaje que tiene, si ha sido visitado o no por el equipo y el número que tiene con respecto al resto, así el inicio sera el 1 y el final n. Considerando esto, la representación de las soluciones será por medio de un vector que tendrá las M rutas solicitadas, cada una de estas es representada nuevamente por un vector, el cual contendrá los nodos en el orden que se visitarán en dicha ruta.

Se ha elegido este tipo de representación debido a que con esta se simplifica el trabajo de los movimientos que se realizaran para generar los diferentes vecinos random para cada iteración del algoritmo, permitiendo así un ingreso y sacado rápido de nodos, además de un trabajo sencillo intercambiando la posición de nodos dentro de una ruta. Además de que la visualización de la solución es inmediata, separando cada ruta de la otra, permitiendo así poder realizar movimiento para cada ruta por separado, sin que afecte a las otras que no esta involucradas.

De esta forma, un ejemplo de solución, considerando que existen 7 nodos (0 al 6) y se desea generar 3 rutas, se vería como lo siguiente:

$$solucion = [m_0, m_1, m_2] = [[0, 1, 4, 6], [0, 2, 6], [0, 5, 6]]$$

Donde m_0, m_1, m_2 son las rutas que se desean generar y 0..,6 son los nodos existente para poder visitar, cabe notar que no todos estos estarán en la solución como es el caso de 3 el cual no forma parte de la solución encontrada, esto debido a la restricción del tiempo máximo por cada ruta.

6. Descripción del algoritmo

Para la resolución del TOP se utilizó el algoritmo *Simulating Anealing*. A continuación se presenta una vista rápida del algoritmo, para poder luego explicar en profundidad como se trabajó cada una de sus partes.

```
Procedure simulated annealing

    inicializar max_iter//cantidad de iteraciones
    inicializar iter -> 0//iteracion actual
    inicializar Temp //temperatura inicial
    inicializar Temp_min //temperatura inicial y actual
    inicializar iter_temp // cantidad de iter. antes de modificar la Temp
    inicializar penalty //penalizacion si excede el tiempo maximo

    inicializar sc -> random //solucion random inicial (1)
    sbest -> sc //la mejor solucion parte siendo la primera

    Repetir
        Repetir
            sn = nueva solucion random N(sc)//sol. random por movimientos. (2)
            If F(sn) > F(sc) entonces: //(3)
                sc -> sn
            Else
            If random([0;1]) < exp(eval/Temp) entonces:
                sc -> sn
            If F(sc) > F(sbest) entonces:
                sbest -> sc
        Hasta iter_temp //se tiene que variar la Temp.
        Temp = g(T; t) //funcion de cambio de la temperatura //(4)
        iter = iter + 1
    Hasta max_iter  //no queden mas iteraciones por hacer

EndProcedure
```

Notemos que dentro de este algoritmos existen 4 puntos importantes en los que profundizar, el primero (1) es la solución random inicial, para este caso se consideró que la mejor opción era partir de una solución "vacía", que tuviese solo el nodo inicial y el final para las M rutas, de esta forma, no se condicionaba la solución final por medio de la inicial y se buscaba generar mayor diversidad en las soluciones finales.

El siguiente punto importante (2), en el cual nos detendremos a analizar el algoritmo utilizado en profundidad, es la generación de un vecino random dada la actual solución, el algoritmo utilizado para esta parte se ve de la siguiente forma.

```
Procedure Vecino_Random

    ruta -> random() //selecciona una ruta random a modificar
    mov -> random() //seleccionar un movimiento random y valido

    If mov es insertar entonces:
        punto -> random() //se selecciona un punto random que no este en la ruta

        If tamanno(ruta) > 2 entonces:
            insertar(ruta,punto) //lo ingresa entre el nodo inicial y final.
        Else:
            pos -> random() //selecciona una posicion random valida
            insertar(ruta,punto,pos)//se inserta en esa posicion

   If mov es sacar entonces:
        nodo -> random()//se selecciona un punto random que este en la ruta
        sacar(ruta,nodo)//saca el nodo de la ruta selecconada.

    If mov es swap entonces:
        nodo1 -> random()//nodo random a intercambiar de la ruta
        nodo2 -> random()//nodo random distinto a intercambiar
        swap(ruta,nodo1,nodo2)//se cambian de posicion en la ruta

EndProcedure
```

Como se puede apreciar, primero se selecciona una ruta random de las existente para realizar el movimiento sobre esta, luego de eso, se elige un movimiento válido a realizar, con esto nos referimos a que no se insertara si la ruta ya tiene todos los nodos, no se sacará si no posee nodos para quitar y no se hará swap si no posee a lo menos 2 nodos para intercambiar. Una vez elegido el movimiento, se procede a aplicarlos. Para insertar primero se elige un punto random de los existente que no pertenezca ya a la ruta y se verifica si está vacía la ruta (solo contiene el inicio y fin) o no, de ser el primer caso, el punto se inserta entre el nodo inicial y final, sino se elige una posición random donde se insertará el nuevo punto considerado. Para el movimiento de sacar un nodo, se elige en principio un nodo que exista en la ruta y luego se quita de esta. Finalmente para el ultimo movimiento, se intercambian dos nodos aleatorios de la ruta. Los fines de esto movimientos son intensificar la solución para el caso de insertar y swap (de ser el caso), y diversificar la solución en el caso de sacar y swap si el cambio genera un empeoramiento en la calidad de la solución.

El siguiente parte del algoritmo a profundizar (3) es el de la función de evaluación, para esto solo se trabajaron dos punto, el primero fue comprobar que ninguna de las rutas superase el tiempo máximo permitido, sino el puntaje del grupo automáticamente llevaría una penalización correspondiente al tiempo excedido, luego se procedía a calcular el puntaje, primero se marcaban los puntos que se visitaban por la solución y sumaban los puntajes de cada uno de los puntos marcados como visitados dentro del vector que contiene todos los puntos existentes, si el tiempo de alguna ruta superaba el máximo, se le descontaba al puntaje la penalización correspondiente a este tiempo. De esta forma, el algoritmo se ve así.

```
Procedure Funcion_de_evaluacion

    inicializar puntaje -> 0 //puntaje inicial
```

```
inicializar Pts -> Puntos//se copian los puntos en una variable local.
inicializar exceso -> 0 //tiempo que se excede alguna ruta
inicializar se_excede -> false //booleano para saber si se excede

For ruta in Rutas://para cada ruta dentro del vector de Rutas
    If tiempo(ruta) > T_max entonces://si el tiempo es mayor que lo maximo
        exceso = tiempo(ruta) - T_max;//se calcula cuanto excede
        se_excede -> true;

For ruta in Rutas://se marcan los puntos que se visitan de la solucion.
    For nodos in ruta://ver que el nodo este en la ruta y no este marcado
        If Esta_en(nodo, pts) && No_visitado(nodo,pts) entonces:
            marcar_visitado(punto)//se marca como visitado el punto

For punto in Puntos: //por cada punto en vector de puntos.
    If Visitado(punto) entonces://si se visito el punto
        puntaje += score_punto //se le suma el puntaje del punto visitado

If se_excede entonces:
    puntaje -= penalty*exceso;
```

EndProcedure

Por último, para el punto (4) del algoritmo, el cual modifica la temperatura. solo se consideró una disminución de esta en un valor fijo y dado al inicio del programa, sin embargo, esta función de cambio de temperatura puede ser lo que cada uno desee, puesto que su único fin es hacer decrecer o aumentar la temperatura para aumentar o disminuir la diversificación de las soluciones.

7. Experimentos

Primero que todo, como la técnica utilizada (*Similatiing Anealing*) es de tipo incompleta y estocástica, es necesario repetir el algoritmo una cierta cantidad de veces para una misma instancia, para así poder estudiar los resultados y obtener un promedio para esa instancia. De esta forma se decide que cada una de las estas será repetida 50 veces para tener los resultado necesarios para el estudio.

Con respecto a los parámetros que influirán en el algoritmo, tenemos los siguientes:

- Iteraciones Máximas: Definen la cantidad de iteraciones generales que se realizan del algoritmo, a mayor sea su valor, mayores la posibilidades de encontrar una mejor solución. Para estas, se consideraran una cantidad fija de 100 iteraciones
- Iteraciones Máximas de Temperatura: Definen la cantidad de iteraciones internas antes de realizar un cambio en la temperatura, para este caso, cuándo se disminuye su valor. Se considera que un valor razonable para estas es de 10.
- Temperatura Inicial: Define la temperatura máxima del algoritmo y con la cual iniciará su ejecución. Los valores para esta irán desde los $50 - 100$ para casos normales y serán constantes por cada ejecución del algoritmo.
- Temperatura Mínima: Define la temperatura mínima del algoritmo, no se podrá bajar mas de esta temperatura, se considera un valor constante entre $10 - 20$

- Variación de Temperatura: Define cuanto disminuye la temperatura cuando corresponda, se considera un valor constante entre $1 - 5$ unidades cada vez que se deba disminuir.
- Penalización: Define la cantidad por la que se multiplicará el tiempo excedido de una de las rutas, para luego restarlo del puntaje obtenido por el equipo, el valor de la penalización dependerá de la instancia a tratar, por razones que se analizaran mas adelante.

Por último, con respecto a las instancias que se trabajarán, se tendrán 9 instancias en total, en grupos de 3 según la cantidad de nodos a considerar, siendo estos: $21 - 66 - 102$ nodos respectivamente. Para cada uno de los grupos de instancias, se considerará 3 tamaños de rutas diferentes: $2 - 3 - 4$, donde cada una tendrá un tiempo limite que puede ser variable o constante para todas.

8. Resultados

Para la primera parte del estudio, se consideraron los 9 problemas y se fijaron parámetros similares para todos, según lo especificado anteriormente. Sin embargo es necesario destacar que para el primer grupo de instancias (21 nodos) fue necesario aumentar la penalización, ya que con un valor de 100, sumado al resto de parámetros fijos para todos igualmente, el algoritmo no era capaz de encontrar soluciones factibles en este caso, ya que comenzaba a aceptar muchas soluciones infactibles, hasta el punto en que el algoritmo no terminaba, ya que quería insertar un nuevo nodo en una ruta, pero no existían mas nodos disponibles. Esta anomalía se verá con mayor detalle en la segunda parte del estudio.

Pasando a los parámetros y resultados correspondientes a esta primera parte, se tiene lo siguiente:

Parámetros

- Iteraciones Máximas = 100
- Iteraciones Máximas de Temperatura = 5
- Temperatura inicial = 70
- Temperatura Mínima = 10
- Variación de Temperatura = 3
- Penalización (caso 21) = 1000
- Penalización (caso 66 y 102) = 100

Cant Nodos	Cant. Rutas	Tmax	Puntaje promedio
21	2	22,5	187
21	3	15,0	179
21	4	11,2	131

Tabla 1: *Resumen de los resultado obtenidos para la primera parte del estudio, donde se ejecutó el primer trío de instancias*

Cant Nodos	Cant. Rutas	Tmax	Puntaje promedio
66	2	65,0	914
66	3	43,3	908
66	4	32,5	1035

Tabla 2: *Resumen de los resultado obtenidos para la primera parte del estudio, donde se ejecutó el segundo trío de instancias*

Cant Nodos	Cant. Rutas	Tmax	Puntaje promedio
102	2	200,0	370
102	3	133,3	345
102	4	100,0	432

Tabla 3: *Resumen de los resultado obtenidos para la primera parte del estudio, donde se ejecutó el tercer trío de instancias*

Como se puede apreciar en la tabla 1, cuando se tiene pocos nodos el tiempo máximo influye de mayor manera en el puntaje que la cantidad de rutas, esto porque el tiempo condicionará mayormente qué nodos se agregan a las rutas, y no importa cuantas rutas tenga, como se poseen pocos nodos, el algoritmo privilegiará que ninguna supere el tiempo máximo, generando que la solución tienda a tener menor puntaje. Caso contrario que ocurre cuando se comienza a aumentar la cantidad de rutas, como en la tabla 2, donde mientras mas rutas se tienen, los puntajes aumentan, sin importar que el tiempo disminuya, esto porque tiene mas espacio para almacenar nodos, además de que tiene mayor cantidad de donde seleccionar nodos no repetidos para las rutas, resultados que se corresponden con los presentados para el caso de 102 nodos (tabla 2).

Por otro lado, como se mencionaba, el primer grupo de casos sufría una anomalía con el valor de penalización normal, por lo que valores inferiores a 1000 no eran suficientes para poder obtener una ejecución correcta del algoritmo y con esta, soluciones factibles. Es por esto, junto con que se quiere estudiar la influencia de la temperatura inicial y final, además de la penalización, que se realiza este segundo estudio.

Para este, se consideró solo 1 de los 3 casos de prueba por grupo, y se varió los valores de la temperatura inicial, temperatura mínima y la penalización, de tal forma de estudiar cuál de estos generaba mayores cambios considerables con respecto a las soluciones anteriormente obtenidas. De esta forma, se tiene que los resultados son los siguientes.

Parámetros

- Iteraciones Máximas = 100
- Iteraciones Máximas de Temperatura = 10
- Variación de Temperatura = 5

T_inicial	T_minima	Penalización	Puntaje Promedio
10	10	10000	140
80	60	10	426
10	10	10	-

Tabla 4: *Resultados para el caso de 21 nodos, 4 rutas a generar y tiempo máximo de 11.2*

T_inicial	T_minima	Penalización	Puntaje Promedio
10	10	10000	599
80	60	10	1187
10	10	10	1670

Tabla 5: *Resultados para el caso de 66 nodos, 4 rutas a generar y tiempo máximo de 32.5*

T_inicial	T_minima	Penalización	Puntaje Promedio
10	10	10000	367
80	60	10	794
10	10	10	1171

Tabla 6: *Resultados para el caso de 102 nodos, 4 rutas a generar y tiempo máximo 100*

Se puede observar que para la primera instancia, la cual sufría de esta anomalía. Tomando un valor de penalización excesivamente alto, y temperatura inicial y mínima iguales, se obtiene un resultado promedio dentro del rango similar al de una ejecución normal del algoritmo, esto porque se busca intensificar directamente, dados los valores de aceptación por parte de la temperatura y la penalización excesiva de soluciones infactibles. Por otro lado cuando la penalización se reducía a un mínimo de 10, pero se variaba la temperatura, se esperaría que no arrojase resultados por lo antes dicho, sin embargo el cambio en la temperatura mínima, permitía que se diversificase lo suficiente para poder obtener soluciones de mejor calidad que el promedio encontrado para el estudio anterior, sin embargo cabe destacar que al momento de buscar estas soluciones, el algoritmos mas de alguna vez se estancaba puesto que no encontraba solución debido a que diversificó demasiado la solución, al punto de que ya no poseía nodos para agregar a esta. Por ultimo se ve el caso que ya confirma nuestra hipótesis de que una penalización baja no ayuda demasiado a la búsqueda de soluciones factibles y mucho menos a una correcta ejecución del algoritmo, puesto que para este caso, donde se considero la temperatura inicial y mínima como iguales y muy baja, el algoritmo no fue capaz de entregar resultados y mucho menos de terminar una ejecución debido a lo ya comentado de la cantidad de nodos.

Dejando de lado la instancia que producía estas anomalías, se estudió igualmente un caso para el grupo 2 y 3, siguiendo el mismo patrón que para el caso anterior, se inició con una temperatura baja y una alta penalización, luego se procedió a aumentar la temperatura y disminuir la penalización. Y finalmente se considera todo al mínimo, tanto en temperatura como en penalizaciones. De esta forma, se puede ver que el tener una mayor penalización no ayuda a mejorar la solución ni a evita la diversificación por medio de movimientos a soluciones infactibles, sin embargo si asegura obtener un resultado factible con un puntaje promedio aceptable. Con respecto a la temperatura, se puede ver claramente que este es el parámetro que genera mayor cambio en la calidad de las soluciones, ya que es uno de los centrales para la consideración de si se acepta o no la solución, dada la comparación entre un numero random perteneciente a $[0; 1]$ y la exponencial de la diferencia entre las calidades nueva y actual sobre la temperatura actual. Es así como un aumento en la temperatura y una disminución en la penalización, genera diversificación y con esto una mejor búsqueda de soluciones de calidad superior, pero es mucho mas efectivo una baja temperatura y penalización, para poder intensificar directamente en búsqueda de una solución de excelente calidad, considerando leves movimientos entre soluciones infactibles generados por la baja penalización.

9. Conclusiones

El TOP a pesar de ser un problema relativamente nuevo (24 años aproximadamente), este ya ha sido atacado por muchas de las heurísticas y técnicas mas conocidas dentro del área de la Inteligencia Artificial. Si bien, todas estas tienen sus propios métodos y algoritmos, parten de la misma base, considerando la representación del problema como un gráfo y posteriormente las soluciones como un conjuntos de rutas con los puntos que visita cada una, obteniendo estas por medio de pequeñas modificaciones según requiera la técnica. En general, se puede ver que no existió una técnica líder, si bien existieron muchas resoluciones en la actualidad basadas en *Branch Cut and Price* y sus derivados, estas no se destacan por sobre el resto de técnicas, cada una mejora en partes la resolución del problema, pero así como tienen ventajas, también poseen falencias, que otras técnicas pueden solucionar, pero también introducir nuevos problemas con ellas.

Con respecto a la solución que se plantea en este informe, la cual esta basada en la técnica de *Simulating Anealing*, se puede concluir que los tiempos que tarde el algoritmo en resolver y encontrar soluciones para diferentes instancias son relativamente bajos, esto debido a que solo realiza movimientos random, por lo que solo debe preocuparse de si se acepta o no la nueva solución, es aquí donde según los resultados presenten en la experimentación se logra comprobar que los parámetros que tienen mayor influencia en la calidad de la solución son los de penalización y temperaturas, siendo este ultimo al mas importante, ya que generaba mayor porcentaje de mejora en las soluciones obtenidas.

Por otro lado, gracias a la experimentación nuevamente, se pudo desligar las ventajas y desventajas que ofrece esta técnica. Dentro las ventajas se tiene que es un algoritmo que encuentra soluciones rápidamente, ya que no requiere de mucha complejidad para llevarlo a cabo, además de que es fácil hacer que el programa diversifique. Sin embargo, también trae consigo una serie de desventajas, dentro de estas podemos destacar el tema de la aleatoriedad de los movimientos, donde muchas veces ocurría que se ingresaba un nuevo nodo, para luego quitar se mismo nodo de la ruta, por lo que se volvía al punto anterior, perdiendo parte del avance hacia nuevas zonas, esto se podría solucionar haciendo uso de un símil a la lista tabú, la cual pueda almacenar el ultimo nodo ingresado, con el fin de que al siguiente movimiento, no se elimine este mismo, sin embargo eso ya seria entrar a modificar la aleatoriedad de SA por lo que se deja propuesto como una modificación a la técnica con el fin de poder mejorar las soluciones y diversificar de mejor manera. Otra de las desventajas en parte presentadas, fue el tema de la condición de aceptación, ya que esta depende de una exponencial y un valor random, es necesario realizar una correcta penalización de las soluciones infactibles, para que el algoritmo no se mueva solo dentro de soluciones infactibles hasta un punto en el que ya no pueda salir de esas zonas y volver a soluciones factibles. Para esto, la mejor forma es jugar con los parámetros antes mencionados de temperatura y penalización para obtener la mejor combinación para la instancia de tal forma de que se siga dando oportunidad de diversificar e intensificar.

Finalmente, con respecto al trabajo que queda por delante, sería imprudente mencionar que ya está todo realizado, día a día surgen nuevas formas de mejorar el problema, generar heurísticas más rápidas y mejores. Además al tratarse de un problema que tiene una alta implementación en la vida actual para las compañías de Viajes y Tours, deja un gran trabajo por delante para hacer de estas soluciones mas óptimas y así entregar un mejor soporte de soluciones para el problema, tal como se menciona, se puede considerar una variante de la técnica, combinando con las prohibiciones de *Tabu Search* para así evitar repetir movimientos que lleven a puntos que ya se visitaron y caminar hacia mejores soluciones, en vez de estancarnos dentro de una zona por movimientos repetitivos que no generan nada de calidad.

Referencias

[1] Chao, I. M., Golden, B. L., & Wasil, E. A. (1996b). A fast and effective heuristic for the orienteering problem. European journal of operational research, 88(3), 475-489.

[2] Chao, I.-M., Gold n, B. L., & Wasil, E. A. (1996a). The team orienteering problem. European Journal of Operational Research 88 (1996) 464-474

[3] Vansteenwegen, P., Souffriau, W., & Van Oudheusden, D. (2011). The orienteering problem: A survey. European Journal of Operational Research 209 (2011) 1–10

[4] Tang, H., & Miller-Hooks, E. (2005). A tabu search heuristic for the team orienteering problem. Computers & Operations Research, 32(6), 1379-1407.

[5] Vansteenwegen, P., Souffriau, W., Berghe, G. V., & Van Oudheusden, D. (2009). A guided local search metaheuristic for the team orienteering problem. European journal of operational research, 196(1), 118-127.

[6] Ke, L., Archetti, C., & Feng, Z. (2008). Ants can solve the team orienteering problem. Computers & Industrial Engineering, 54(3), 648-665.

[7] Poggi, M., Viana, H., & Uchoa, E. (2010). The team orienteering problem: Formulations and branch-cut and price. In OASIcs-OpenAccess Series in Informatics (Vol. 14). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[8] El-Hajj, R., Dang, D. C., & Moukrim, A. (2016). Solving the team orienteering problem with cutting planes. Computers & Operations Research, 74, 21-30.

[9] Boussier, S., Feillet, D., & Gendreau, M. (2007). An exact algorithm for team orienteering problems. 4or, 5(3), 211-230.