

Inteligencia Artificial

Informe Final: Team Orienteering Problem

Guillermo Godoy Álvarez

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

En éste artículo se confecciona un estado del arte sobre el Problema de Orientación en Equipo (TOP) el cual, es un problema de enrutamiento en equipo, es decir, de selección de diferentes caminos o rutas, para cada integrante del equipo. En la cual, se optimiza la ganancia de puntaje, al visitar distintos lugares (puntos, vértices), cumpliendo un tiempo determinado. Cabe mencionar, que tiene un punto de inicio y un punto final. Es necesario considerar, que no se repite la puntuación, en caso que 2 o más compañeros pasen por el mismo lugar. También se profundiza sobre las estrategias (o heurísticas) recientes y antiguas utilizadas, para resolver el problema. Se debe tener en consideración, las limitaciones de las estrategias, como también las estrategias más prometedoras para la resolución de éstas.

1. Introducción

El propósito de éste artículo, es confeccionar un estado del arte relacionado al Problema de Orientación en Equipo (*Team Orienteering Problem* o TOP), realizar un análisis de los trabajos existentes sobre el problema, como las soluciones más recurrentes y efectuar una comparación de éstas.

El Problema de Orientación en Equipo (TOP) es una generalización del Problema de Orientación, que consiste en un deporte, en el cual los competidores tienen que elegir rutas para llegar a diversos puntos de control, que tienen una puntuación propia. El competidor que llega a la meta en un tiempo límite, y pase por los puntos de control con mayor puntuación, es el ganador.

Cuando se juega en equipo, los compañeros pueden ir a distintos puntos y obtener la puntuación de éstos, pero todos tienen que llegar a la meta. Es necesario considerar, que no se repite la puntuación, en caso que 2 o más compañeros pasen por el mismo lugar.

Al resolver este problema se busca optimizar la puntuación del equipo, escogiendo los mejores caminos que cumplan con la restricción de tiempo. Considerando que un jugador no se puede devolver a un punto de control que ya visitó. De esta forma, nacen varias heurísticas para lograr este objetivo. Una motivación de buscar mejores métodos de solución, es que éste tipo de problema, se puede llevar a otros ámbitos (se puede ver desde distintos puntos de vista) como por ejemplo, encontrar las mejores rutas en un tour, o problemas de enrutamiento de vehículos con ganancia (*Traveling Salesman Problems with Profits*[5]). Éste artículo, se constituye en primer lugar, con la Definición del Problema, en que se analizan las diferentes problemáticas vinculadas con el (TOP), y se describen variaciones del problema. El Estado del Arte, donde se detallan las técnicas más utilizadas, y se describen dos de los últimos algoritmos desarrollados. El Modelo Matemático, en el cual se muestran representaciones propuestas del problema y su descripción. Una Representación de una solución propuesta. La Descripción del algoritmo, en que se explica la solución utilizada para el problema. Experimentos, en el que se detallan los experimentos utilizados. Los resultados del experimento. Finalmente, se presentan las conclusiones del artículo.

2. Definición del Problema

El problema de Orientación en Equipo (*Team Orienteering Problem* o TOP) [12] se basa en el problema de Orientación (*Orienteering Problem* o OP), el cual es un problema NP-Hard descrito así en [6], donde en [11] es mencionado como un deporte, en el que un competidor tiene que seleccionar un camino desde un punto de partida hasta un destino o punto final, pasando por puntos de control a lo largo del camino. Cada punto de control tiene una puntuación asociada a él. El competidor tiene que seleccionar un conjunto de puntos de control para ser visitado, de modo que el puntaje total se maximice; mientras que el costo total del viaje no exceda un determinado límite. El costo normalmente se asocia al tiempo.

Cuando se juega en equipo (TOP), todos comienzan en el mismo punto de partida, los compañeros pueden ir a distintos puntos de control y obtener la puntuación de éstos, pero todos tienen que llegar a la meta. Es necesario considerar, que no se repite la puntuación, en caso que 2 o más compañeros pasen por el mismo punto, sólo se suma una vez esa puntuación al equipo. Es importante tomar en cuenta, que todo el equipo tiene que llegar al punto final, antes del tiempo límite.

La función objetivo es maximizar el puntaje total recolectado, las restricciones para esto son: En cada ruta, no se pueden repetir los puntos de control. Garantizar que cada ruta comienza en el punto de partida y termina en el punto final. Garantizar la conectividad de cada ruta. Asegurar el presupuesto de tiempo limitado para cada ruta. La variable de esta función, es decidir, que punto de control va a ser visitado o que ruta se va a utilizar.

Este tipo de problema se puede llevar a diferentes ámbitos. Como por ejemplo, en [17] se describe una aplicación TOP de reclutamiento de atletas de escuelas secundarias. Un reclutador tiene que visitar varias escuelas en un número determinado de días. Primero puede asignar un puntaje a cada escuela, según su potencial de reclutamiento. Como el tiempo disponible del reclutador es limitado, tiene que elegir las escuelas que visitará cada día, e intentar maximizar el potencial de reclutamiento.

Por otro lado, [14] describe una aplicación TOP de enrutamiento dónde técnicos tienen que atender a los clientes. Cada ruta TOP representa a un técnico, que sólo puede trabajar un número limitado de horas en un día. Por lo tanto, no todos los clientes que requieren servicio, pueden incluirse en los horarios diarios de los técnicos. Por ende, Deberá seleccionarse un subconjunto de clientes, teniendo en cuenta la importancia del cliente y la urgencia de la tarea.

De la misma manera, en que nace TOP como una modificación de OP, también surgen algunas variantes del TOP, la más conocida, por ejemplo:

El Problema de Orientación en Equipo con Ventana de Tiempo (*Team Orienteering Problems with Time Windows* o TOPTW) en la cual, se da un conjunto de nodos asociados con una ganancia, un tiempo de visita, y una ventana de tiempo para cada nodo. Lo que implica, que sólo puede ser visitado en un rango de tiempo, determinado por la ventana. Sin embargo, también se puede esperar a su apertura. Donde el objetivo del problema es maximizar la ganancia general obtenida, al visitar los nodos en todas las rutas cumpliendo con las restricciones antes mencionadas.

El *Tourist Trip Design Problem* (TTDP) el cual los turistas planifican un viaje basándose en sus preferencias personales y en la información de puntos de interés del destino, donde las preferencias sobre los puntos de interés y las restricciones de duración de las rutas son consideradas difusas.

3. Estado del Arte

El problema de Orientación en equipo, fue estudiado por [17] con el nombre *Multiple Path Maximum Collection Problem*, su nombre actual se introdujo en [12]. La primera heurística mencionada en ese artículo, es parecida a su heurística de cinco pasos (five-step heuristic) para el OP ([11]) dónde en lugar de seleccionar la mejor ruta, se seleccionan las mejores P rutas, y se usan dos pasos de reinicialización en lugar de uno.

Un algoritmo exacto para resolver el TOP es utilizar la generación de columnas, el cual, fue publicado en [13] donde resolvieron problemas con hasta 100 vértices, cuando el número de vértices en cada ruta sigue siendo pequeño. Al ser un algoritmo exacto, sólo puede resolver conjuntos pequeños de vértices, en un tiempo polinomial, por lo cual no es una buena estrategia para resolver este problema. Luego en [14] se desarrolla una heurística de búsqueda de tabú (*tabu search heurístico* TMH) incorporada en un procedimiento de memoria adaptativa (*Adaptive Memory Procedure* o AMP).

Las heurísticas más utilizadas son *Tabu search* (utilizados en [14, 1]), *Variable neighbourhood Search* (utilizados en [1, 9]), *Particle swarm optimisation* (utilizados en [4, 16]).

Entre los algoritmos actuales se encuentran:

1. En [15], se propone una metaheurística, llamada *Pareto mimic algorithm*, la contribución se resume en:
 - a) Utilizar un nuevo operador, llamado operador *mimic operator*, para generar una nueva solución, al imitar una solución predominante. A través, de un parámetro. Éste operador controla fácilmente la similitud entre la solución generada y la solución predominante.
 - b) Actualizar una población de soluciones tradicionales, basadas en el dominio de Pareto. De hecho, para determinar las soluciones establecidas, pueden modelarse como un problema de decisión de criterios múltiples
 - c) Adoptar un nuevo operador, llamado *swallow operator*, para mejorar una solución. Éste operador intenta tragar (o insertar) un nodo no viable y luego reparar la solución no viable resultante.

A diferencia del *local search*, el *swallow operator* permite la búsqueda de un túnel a través del área de solución no viable. La idea de hacer túneles, a través del área de solución no factible, también se ha utilizado en [1, 8]

El *Pareto mimic algorithm* (PMA), es una técnica de solución iterativa. Al principio, se inicializan N soluciones incumbentes. La mejor solución hasta ahora x_b , se establece como la que tiene el mayor valor objetivo, entre estas N soluciones, donde el valor objetivo de

una solución x , denotado por $F(x)$, es la recompensa total recibida de esos caminos de x . En cada paso, funciona de la siguiente manera:

- A partir de cada solución predominante, el *mimic operator* genera una nueva solución.
- Posteriormente, la técnica *local search* y el *swallow operator* se emplean para mejorarla.
- Mediante el uso de las nuevas soluciones generadas y las antiguas soluciones establecidas, el nuevo conjunto de soluciones establecidas, se actualiza de acuerdo con el dominio de Pareto.
- La mejor solución hasta ahora x_b también se renueva.

PMA termina sólo cuando su parámetro de salida es satisfecho.

Comparación con otras técnicas:

- En la Figura 3 se muestra un extracto de 82 instancias donde se compara el PMA con PSOiA (*Particle Swarm Optimization-based Memetic Algorithm* [4]). Las columnas 1 a 5 presentan la información de cada instancia (es decir, el nombre, el número de puntos de control, el número de compañeros del equipo, el límite de tiempo y el método de generación de instancias). Para cada instancia, las Columnas 6–8 reportan el mejor (Best) y el valor medio (Mean) y el tiempo de ejecución (PSOiA) respectivamente, y las Columnas 9–11 reportan estos valores de PMA.

Instances				PSOiA				PMA		
Name	n	m	T _{max}	gen	Best	Mean	time(s)	Best	Mean	time(s)
cmt101c_m3	100	3	126.33	gen2	1300	1299.0	111.1	1300	1299.0	42.0
cmt151b_m3	150	3	116.67	gen2	1385	1373.8	754.0	1385	1374.5	169.9
cmt151c_m2	150	2	262.50	gen2	1963	1962.0	1799.6	1964	1962.0	368.5
cmt151c_m3	150	3	175.00	gen2	1916	1909.1	1376.2	1916	1909.2	441.5
cmt151c_m4	150	4	131.25	gen2	1880	1875.6	881.1	1880	1877.6	826.5
cmt200b_m2	199	2	191.00	gen2	2096	2088.2	4181.0	2096	2086.8	669.4
cmt200b_m3	199	3	127.33	gen2	2019	2005.0	2711.7	2019	2009.4	1351.7
cmt200b_m4	198	4	95.50	gen2	1894	1889.7	1515.2	1894	1891.6	974.5
cmt200c_m2	199	2	286.50	gen2	2818	2810.1	7320.3	2818	2810.6	1048.3
cmt200c_m3	199	3	191.00	gen2	2766	2751.2	4217.3	2766	2751.8	1200.0
cmt200c_m4	199	4	143.25	gen2	2712	2700.6	3004.1	2712	2703.3	1411.4
eil101b_m3	100	3	105.00	gen2	916	913.8	134.4	916	914.6	160.6
eil101c_m2	100	2	236.00	gen2	1305	1304.8	452.8	1305	1304.8	250.5
eil101c_m3	100	3	157.33	gen2	1251	1244.1	227.6	1251	1244.2	95.0
gil262a_m2	241	2	297.50	gen2	4078	4056.4	5907.3	4078	4066.3	2100.0

Figura 1: Extracto de la Tabla comparativa de 82 instancias PSOiA vs PMA[15]

- Se compara PMA con otros 2 algoritmos, GRASP \times ELS (*GRASP \times evolutionary local search hybrid* [3]) y MPNS-GRASP (*multiple phase neighborhood search-GRASP*[7]). Las columnas 1 a 6 de la figura 2 presentan la información de cada instancia. La columna 7 presenta los mejores resultados conocidos. Para cada instancia, se informa el mejor (Best) y el tiempo de ejecución (Time(s)) de cada algoritmo. Como se ve en los resultados, PMA es ligeramente mejor que MPNS-GRASP. Pero es GRASP \times ELS el mejor de los 3, y en los dos conjuntos de instancias de referencia, las brechas promedio entre PMA y GRASP ELS son 0.08 % y 0.76 % respectivamente.

Instances	Best known					GRASP × ELS ^a		MPNS-GRASP ^b		PMA	
	n	D_{max}	T_{max}	st	m	Best	Time (s)	Best	Time (s)	Best	Time (s)
1	51	160	∞	0	5	524.61	0.06	524.61	0.60	524.61	2.07
2	76	140	∞	0	10	835.26	15.01	836.39	5.40	835.26	9.71
3	101	200	∞	0	8	826.14	3.19	826.14	6.60	826.14	12.31
4	151	200	∞	0	12	1028.42	1029.48	1032.24	19.80	1029.79	23.98
5	200	200	∞	0	17	1291.29	1294.09	1314.25	55.20	1301.88	46.54
6	51	160	200	10	6	555.43	0.14	555.43	0.60	555.43	1.16
7	76	140	160	10	11	909.68	909.68	909.68	6.60	909.68	5.59
8	101	200	230	10	9	865.94	865.94	865.94	19.80	865.94	5.00
9	151	200	200	10	14	1162.55	1162.55	1175.86	35.40	1165.13	17.86
10	200	200	200	10	18	1395.85	1401.11	1412.11	72.60	1405.21	26.53
11	121	200	∞	0	7	1042.11	1042.11	1042.11	4.80	1042.11	12.43
12	101	200	∞	0	10	819.56	819.56	821.12	7.80	819.56	5.33
13	121	200	720	50	11	1541.14	1545.43	1548.53	10.20	1545.67	15.93
14	101	200	1040	90	11	866.37	866.37	868.62	7.20	866.37	8.54
Average						976.03	976.98	980.93	18.04	978.06	13.78

Figura 2: Tabla comparativa de 14 instancias GRASP×ELSa vs MPNS-GRASPb vs PMA

- En [10], se propone un nuevo método basado en dos algoritmos para resolver el TOP. Primero es crear una solución inicial, utilizando el algoritmo *k-mean algorithm* o *Lloyd algorithm* de una métrica en términos de ganancia y costo. El segundo es una nueva heurística que desarrolla la solución anterior para resolver el TOP.

Ik-means es uno de los algoritmos de aprendizaje sin supervisión más simple, que resuelve el conocido problema de agrupamiento.

La fase 1, encuentra la solución inicial, utilizando k-mean clustering al no exceder las restricciones del coste total. La fase 2, desarrolla una búsqueda heurística de mejora, para mejorar así lo obtenido en la fase 1, basado en la maximización de $\frac{Ganancia}{Costo}$.

Comparación:

- Se presenta una comparación de soluciones logradas utilizando instancias de referencia propuestas por Chao et al. [12] con la heurística de Boussier et al. 2007 [2] y Dang et al. 2013b [16]

Table 3: Comparison with other exact methods.

set	Dang et al. (2013)	Boussier et al(2007)	our method
1	54/54	51/54	54/54
2	33/33	33/33	33/33
3	60/60	50/60	51/60
4	22/60	25/60	25/60
5	44/78	48/78	58/78
6	42/42	36/42	40/42
7	23/60	27/60	31/60
Total	278/387	270/387	292/387

Figura 3: La Tabla resume los resultados de la comparación, en términos de número total de los casos resueltos

4. Modelo Matemático

- Medelo obtenido de [18]

El objetivo es determinar las rutas P, cada uno limitado por T_{max} y maximiza el puntaje

total recolectado, esto se puede formular como un problema entero, con estas variables de decisión: $X_{ijp} = 1$ si, en la ruta p , una visita al vértice i va seguida de una visita al vértice j , de lo contrario $= 0$; $Y_{ip} = 1$ si, el vértice i se visita en la ruta p , de lo contrario $= 0$; U_{ip} = la posición del vértice i en el camino p .

$$\max \sum_{p=1}^P \sum_{i=2}^{N-1} S_i y_{ip} \quad (1)$$

$$\sum_{p=1}^P \sum_{j=2}^N x_{1jp} = \sum_{p=1}^P \sum_{i=1}^{N-1} x_{iNp} = P \quad (2)$$

$$\sum_{p=1}^P y_{kp} \leq 1; \quad \forall k = 2, \dots, N-1 \quad (3)$$

$$\sum_{i=1}^{N-1} x_{ikp} = \sum_{j=2}^N x_{kjp} = y_{kp}; \quad \forall k = 2, \dots, N-1; \quad \forall p = 1, \dots, P \quad (4)$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^N t_{ij} x_{ijp} \leq T_{max}; \quad \forall p = 1, \dots, P \quad (5)$$

$$2 \leq u_{ip} \leq N; \quad \forall i = 2, \dots, N; \quad \forall p = 1, \dots, P \quad (6)$$

$$u_{ip} - u_{jp} + 1 \leq (N-1)(1 - x_{ijp}); \quad \forall i, j = 2, \dots, N; \quad \forall p = 1, \dots, P \quad (7)$$

$$x_{ijp}, y_{ip} \in \{0, 1\}; \quad \forall i, j = 1, \dots, N; \quad \forall p = 1, \dots, P \quad (8)$$

La función objetivo (1) es maximizar el puntaje total recolectado. La restricción (2) garantiza que cada ruta comienza en el vértice 1 y termina en el vértice N. Las restricción (3) aseguran que cada vértice se visita como máximo una vez. Las restricción (4) garantizan la conectividad de cada ruta. Las restricción (5) aseguran el presupuesto de tiempo limitado para cada ruta. Las restricciones (6), (7) son necesarias para prevenir los subtours.

2. Medelo obtenido de[10]

TOP se modela como un grafo completo $G = (V, E)$, con $V = (\{1, \dots, n\} \cup \{d, a\})$ que son los puntos de control y $E = \{(i, j) | i, j \in V, i \neq j\}$ el conjunto de arcos.

Los vértices d y a son respectivamente el punto de inicio y punto final. Por conveniencia, se utilizan tres conjuntos \bar{V} , V^d y V^a que respectivamente son los puntos de control, los puntos de entrada y los puntos de llegada o punto final.

Cada vértice i está asociado con una ganancia P_i y el costo de viaje c_{ij} (utilizaremos costo como el tiempo de hacer el viaje) está asociado con cada arco $(i, j) \in E$.

Se asume que el costo del viaje satisfacen la desigualdad del triángulo. F está compuesto por m personas idénticos en capacidad y disponible para visitar los puntos de control.

Cada compañero de equipo debe comenzar su ruta desde d , visitar un cierto número de puntos de control y regresar a a sin exceder su costo de viaje predefinido (Tiempo limite) C_{max} .

El problema se puede formular como un *Mixed Integer Programming* (MIP) usando un número polinomial de variables de decisión x_{ijr} y y_{ir} : $x_{ijr} = 1$ si el arco (i, j) es utilizado por compañero de equipo r para visitar el punto de control i antes que j o $= 0$ de lo contrario; $y_{ir} = 1$ si el punto de control i es visitado por el compañero de equipo r o $= 0$

de lo contrario.

$$\max \sum_{i \in \bar{V}} \sum_{r \in F} y_{ir} p_i \quad (9)$$

$$\sum_{r \in F} y_{ir} \leq 1 \quad \forall i \in \bar{V} \quad (10)$$

$$\sum_{j \in V^d} x_{jar} = \sum_{j \in V^a} x_{djr} = 1 \quad (11)$$

$$\sum_{i \neq k} x_{kir} = \sum_{j \neq k} x_{kjr} = y_{kr} \quad \forall r \in F, \forall k \in \bar{V} \quad (12)$$

$$\sum_{i \in V^d} \sum_{j \in V^a \setminus \{i\}} C_{ij} x_{ij} \leq C_{max} \quad \forall r \in F \quad (13)$$

$$x_{ijr} \in \{0, 1\} \quad \forall i \in V, \forall j \in V, \forall r \in F \quad (14)$$

$$y_{ir} \in \{0, 1\} \quad \forall i \in \bar{V}, \forall r \in F \quad (15)$$

La función objetivo (9) es maximizar la suma de los beneficios recaudados. Las restricciones (10) garantizan que cada punto de control es visitado como máximo una vez. La conectividad de cada tour está asegurada por restricciones (11) y (12). Restricciones (13) describe la restricción de longitud de viaje. Finalmente, las restricciones (14) y (15) establece el requisito integral sobre las variables.

5. Representación

En esta sección se dará a conocer las expresiones matemáticas y estructuras ocupadas para resolver el problema.

5.0.1. Estructura General

NODELIST: Estructura tipo *typedef*, la cual se utiliza como representación de cada nodo y su estructura consiste de 4 variables:

- *X* e *Y*: Ambos de tipo float, donde representa la posición cartesiana del nodo, obtenida del archivo input.
- *score*: Tipo int, representa la puntuación de visitar ese nodo, obtenida del archivo input.
- **next*: Puntero a *NODELIST*, se utiliza para enlazar los nodos en una cadena, donde el primer nodo apunta al siguiente nodo agregado y el ultimo nodo apunta a *NULL*.

Consideraciones: Cada *NODELIST*, se va enlazando en forma de cola y en el orden que llegan del input, donde el nodo 0 siempre es el nodo inicial y el nodo final es el numero de nodos menos uno. La Distancia cartesiana de cada nodo es igual al tiempo que se demora en llegar al nodo. La ruta desde el nodo inicial al nodo final, es una ruta factible del problema.

5.0.2. Variables Globales

DistanciaTable : Vector de vectores tipo float, de tamaño *n* por *n*, donde *n* es la cantidad de nodos. Ésta matriz es global ya que solo se modifica una vez y es utilizada en la totalidad del problema. Su tamaño es modificado luego de obtener la cantidad de nodos y se almacena la distancia entre nodos.

Ejemplo: *DistanciaTable*[1, 2] = {3}, donde 3 es la distancia entre el nodo 1 y el nodo 2.

RutasFactibles: Vector de vectores de tipo int, su tamaño depende de la cantidad de rutas factibles encontradas y el tamaño de cada ruta factible. Ésta lista de rutas es global ya que solo es utilizada para almacenar rutas factibles y es utilizada en varias funciones.

Ejemplo: *RutasFactibles*[1] = {0, 5, 20}, la ruta factible numero 2, pasa por el nodo 0, el nodo 5 y el nodo 20.

mejorPuntaje : Tipo int, donde se almacena el mejor puntaje encontrado de una solución al problema, se utiliza solo para saber si alguna solución supero el mejor puntaje encontrado durante la ejecucion.

NumResultados : Tipo int, donde se almacena la cantidad total de resultados encontrados.

5.0.3. Variables

**nodelist*: Puntero a *NODELIST*, la cual apunta al nodo inicial.

tiempoocupado: Tipo float, el cual indica el tiempo acumulado en la instancia actual.

historialNodo: Vectores tipo int, indica los nodos visitados de cada instanciación, si la instanciación es una ruta factibles, el *historialNodo* es almacenado en *RutasFactibles*.

rutaSolucion: Vector tipo int, indica una combinación de rutas factibles de la instancia actual, el cual es una solución al problema.

Ejemplo: *rutaSolucion* = {4, 5, 18}, indica que la solución esta dada por *RutasFactibles*[4], *RutasFactibles*[5] y *RutasFactibles*[18], el largo de *rutaSolucion* siempre es igual al numero de rutas (o compañeros del equipo) del problema.

vectorConflicto: Vector tipo int, donde se almacenan los nodos con conflicto, más prematuramente instanciado.

5.0.4. Parámetro

Tmax: Tipo float, donde se guarda el tiempo limite para la solución del problema.

NumEquipo: Tipo int, donde se guarda el numero de rutas (o compañeros del equipo) para la solución del problema.

MaxNodos: Tipo int, donde se guarda la máxima cantidad de nodos del problema.

6. Descripción del algoritmo

La solución que se utilizo para éste problema consta de 4 pasos y es utilizado como técnica completa, ya que recorre todas las soluciones del problema. Como el problema TOP, es un problema NP-Hard, entonces es posible de transformar en otro problema NP-Hard, es por esto, que se transformo el problema en una combinación del problema OP y un coeficiente binomial para las soluciones OP.

- Paso 1: Se obtienen los parámetros del problema y se genera la tabla de distancia.
- Paso 2: Se utiliza el Algoritmo 1, para que en todas las ramificaciones del nodo inicial, se invoque un algoritmo recursivo.

Algoritmo 1: Iniciar BT

```
1: Si Distancia entre el nodo inicial y el final  $\leq T_{max}$  Entonces
2:   Agrego la ruta desde el inicial al final como ruta factible al problema.
3:   Agrego al historialNodo el nodo inicial
4:   Para cada arco del nodo inicial, sin contar el arco que lleva al nodo final Hacer
5:     Si distancia entre el nodo inicial al nodo del arco conectado  $\leq T_{max}$  Entonces
6:       tiempoocupado = Distancia entre el nodo inicial al nodo del arco conectado
7:       Agrego al historialNodo el nodo conectado
8:       BT+CBJ(tiempoocupado,historialNodo)
9:     Fin Si
10:  Fin Para
11: Si no
12:   No existe ninguna ruta factible al problema, se termina el programa, mostrando por
   consola que no hay solución
13: Fin Si
```

- Paso 3: Se utiliza el Algoritmo 2, basado en *Backtracking con Retorno Guiado por Conflictos* para encontrar todas las rutas factibles del problema, las cuales se guardan en *RutasFactibles*.

El algoritmo, que se describe a continuación, tiene como dominio en cada instanciación, todos sus arcos, menos los que conecten con los nodos del *historialNodo*. Su restricción viene dada por la suma del tiempo acumulado y la distancia del arco de su dominio, el cual no debe superar el T_{max} , en caso que no cumpla con la restricción, revisa si tiene un conflicto con alguna instancia anterior en orden cronológico en que fue creada.

Algoritmo 2: BT+CBJ

Entrada: *tiempoocupado* de la instancia anterior y *historialNodo* de la instancia anterior.

Salida: punto de regreso del nodo mas reciente en el conjunto de conflictos.

```
1: Para cada arco, sin contar el arco que lleve a algún nodo del historialNodo Hacer
2:   Si tiempoocupado + Distancia entre el ultimo nodo en historialNodo al nodo del arco
   conectado  $\leq T_{max}$  Entonces
3:     Agrego al historialNodo el nodo conectado
4:     Si nodo conectado = nodo final Entonces
5:       Agrego el historialNodo a RutasFactibles
6:       Quito del historialNodo el nodo conectado
7:       Devolver Ultimo nodo en historialNodo
8:     Si no
9:        $X \leftarrow$  BT+CBJ(tiempoocupado + Distancia del arco,historialNodo)
10:      Quito del historialNodo el nodo conectado
11:      Si nodo conectado  $\neq X$  Entonces
12:        Devolver  $X$ 
13:      Fin Si
14:    Fin Si
15:  Si no
16:    Agregar a vectorConflicto el retorno de la función Primer-Conflicto(historialNodo).
17:  Fin Si
18: Fin Para
19: Devolver Primer nodo en vectorConflicto
```

El método para revisar el conflicto se describe en el Algoritmo 3, el cual revisa desde el primer nodo del *historialNodo* de la instanciación, para ver cual fue la instancia que dio conflicto, éste se revisa, viendo si desde esa instancia se tenia tiempo suficiente para volver

al nodo final, la primera instancia que me de conflicto, retorno el nodo que fue escogido en esa instancia, de esta forma puedo revisar si me encuentro en esa instancia o se tiene que seguir devolviendo.

Algoritmo 3: Primer-Conflicto

Entrada: *historialNodo* de la instanciación.

Salida: el nodo que fue escogido por la instancia en conflicto.

- 1: *tiempoocupado* \leftarrow Distancia entre el primer nodo y el segundo nodo del *historialNodo*
 - 2: **Para** cada arco, que conecte los nodos en *historialNodo* **Hacer**
 - 3: **Si** *tiempoocupado* + Distancia del nodo conectado al ultimo nodo $> T_{max}$ **Entonces**
 - 4: **Devolver** el primer nodo del arco conectado
 - 5: **Si no**
 - 6: *tiempoocupado* + = Distancia del arco
 - 7: **Fin Si**
 - 8: **Fin Para**
-

- Paso 4: Se utiliza una función recursiva para combinar todos los índices de *RutasFactibles*, ésta obedece la función matemática de conminatoria sin repetición.

$$C_{n,k} = \binom{n}{k} = \frac{n!}{k!(n-k)!} \quad (16)$$

Donde n es la cantidad de rutas factibles encontradas y k es *NumEquipo* y cada combinación es guardada en *rutaSolucion*.

Para cada *rutaSolucion*, se suma uno al *NumResultados* y se obtiene el puntaje, sin contar los nodos repetidos. Si el puntaje es mejor que *mejorPuntaje*, este se modifica y se escribe en un archivo, como mejor solución, junto con el tiempo total de cada ruta.

Para comparar la heurística del Algoritmo 2, se creo un Algoritmo 4 basado en *Backtracking con Look-Ahead* y un Algoritmo 5 basado solo en *Backtracking* las cuales son invocadas de la misma forma que en el paso 2.

Algoritmo 4: BT+Look-ahead

Entrada: *tiempoocupado* de la instancia anterior y *historialNodo* de la instancia anterior.

- 1: **Si** *tiempoocupado* + Distancia entre el ultimo nodo en *historialNodo* al nodo final $\leq T_{max}$ **Entonces**
 - 2: **Para** cada arco, sin contar el arco que lleve a algún nodo del *historialNodo* y sin contar el nodo final **Hacer**
 - 3: **Si** *tiempoocupado* + Distancia entre el ultimo nodo en *historialNodo* al nodo del arco conectado $\leq T_{max}$ **Entonces**
 - 4: Agrego al *historialNodo* el nodo conectado
 - 5: BT+Look-ahead(*tiempoocupado* + Distancia del arco, *historialNodo*)
 - 6: Quito del *historialNodo* el nodo conectado
 - 7: **Fin Si**
 - 8: **Fin Para**
 - 9: Agrego al *historialNodo* el nodo final
 - 10: Agrego el *historialNodo* a *RutasFactibles*
 - 11: Quito al *historialNodo* el nodo final
 - 12: **Fin Si**
-

Algoritmo 5: BT

Entrada: *tiempoocupado* de la instancia anterior y *historialNodo* de la instancia anterior.
Salida: punto de regreso del nodo mas reciente en el conjunto de conflictos.

- 1: **Para** cada arco, sin contar el arco que lleve a algún nodo del *historialNodo* **Hacer**
- 2: **Si** *tiempoocupado* + Distancia entre el ultimo nodo en *historialNodo* al nodo del arco conectado $\leq T_{max}$ **Entonces**
- 3: Agrego al *historialNodo* el nodo conectado
- 4: **Si** nodo conectado = nodo final **Entonces**
- 5: Agrego el *historialNodo* a *RutasFactibles*
- 6: **Si no**
- 7: BT(*tiempoocupado* + Distancia del arco, *historialNodo*)
- 8: **Fin Si**
- 9: Quito del *historialNodo* el nodo conectado
- 10: **Fin Si**
- 11: **Fin Para**
- 12: **Devolver** Primer nodo en *vectorConflicto*

7. Experimentos

Para todos los experimentos se utilizaron varias maquinas con las mismas características, las cuales son:

- Procesador Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz de 64 bits
- 4 núcleos
- RAM de 10 GiB
- Sistema Operativo Fedora 28

Las características del conjunto de prueba se muestra en la tabla (1):

	Numero de nodos	Numero de rutas
Set_21_234	21	2, 3, 4
Set_33_234	33	2, 3, 4
Set_66_234	66	2, 3, 4
Set_102_234	102	2, 3, 4

Tabla 1: Conjunto de pruebas

Se genero un programa en C++, que utiliza los algoritmos mencionados, el cual obtiene como input un archivo con los parámetros de las pruebas, retornando un output, con la mejor solución encontrada en una búsqueda completa y un segundo output, donde se menciona el tiempo utilizado para cada heurística, el tiempo en la combinatoria y la cantidad de soluciones encontradas.

8. Resultados

En la tabla 2, se saca el promedio final para cada heurística en las distintas pruebas y así comparar su rendimiento.

#Nodos	#Rutas	BT (s)	BT+CBJ (s)	BT+FC (s)
21	2	0,0667	0,1611	0,0085
21	3	0,0044	0,0081	0,0004
21	4	0,0004	0,0012	0,0001
33	2	0,1269	0,3087	0,0171
33	3	0,0100	0,0193	0,0009
33	4	0,0010	0,0027	0,0001
66	2	0,0627	0,1474	0,0031
66	3	0,0062	0,0155	0,0004
66	4	0,0005	0,0014	0,0000
102	2	3,7629	7,3414	0,0036
102	3	2,5387	4,9594	0,0025
102	4	0,0215	0,0476	0,0001
\bar{X}		0,5502	1,0845	0,0031

Tabla 2: Promedio de tiempo para cada Heurística utilizada, en los distintos conjuntos de prueba

Se revisa una diferencia de tiempo de ejecución, donde la técnica BT+FC, es superior, dado que al revisar con antelación los conflictos y elimina el dominio infactible antes de su revisión. En cambio BT+CBJ, tiene la peor puntuación para todos los casos, esto se debe, principalmente, ha que todos los nodos se encuentran conectados, por ende, en cada valor erróneo en el dominio de la instancia, se revisa su conflicto y si la instancia es infactible, habrá revisado todo su dominio antes de devolver el punto de retorno.

En las figuras 4,5 y 6, se muestra una relación entre el tiempo de ejecución de cada heurística, para cada T_{max} de las tablas 6,9,12 respectivamente.

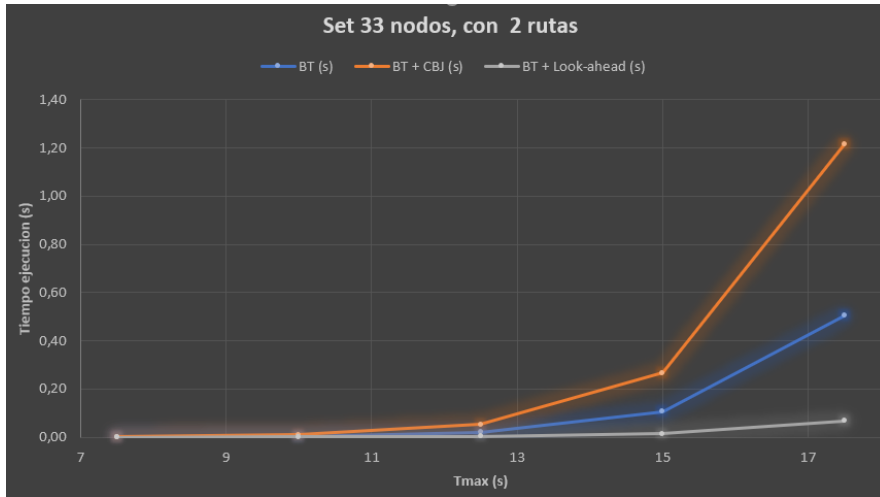


Figura 4: Relación entre el tiempo de ejecución de cada heurística y T_{max}

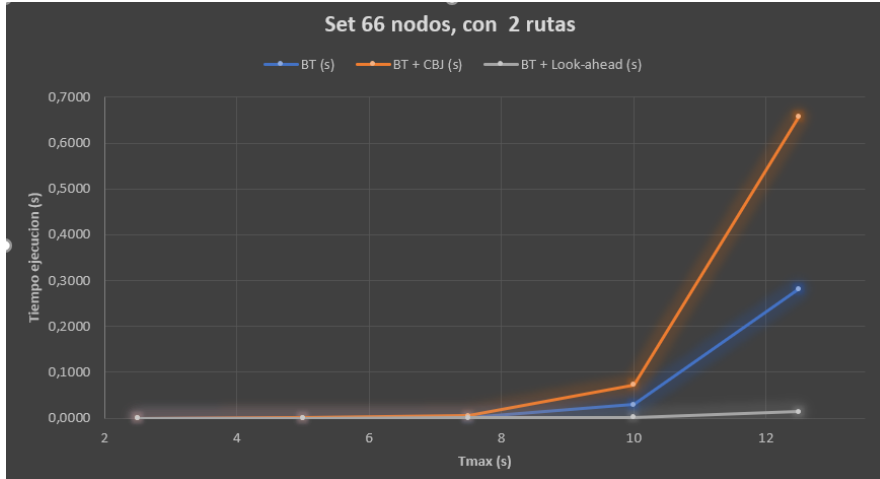


Figura 5: Relación entre el tiempo de ejecución de cada heurística y Tmax

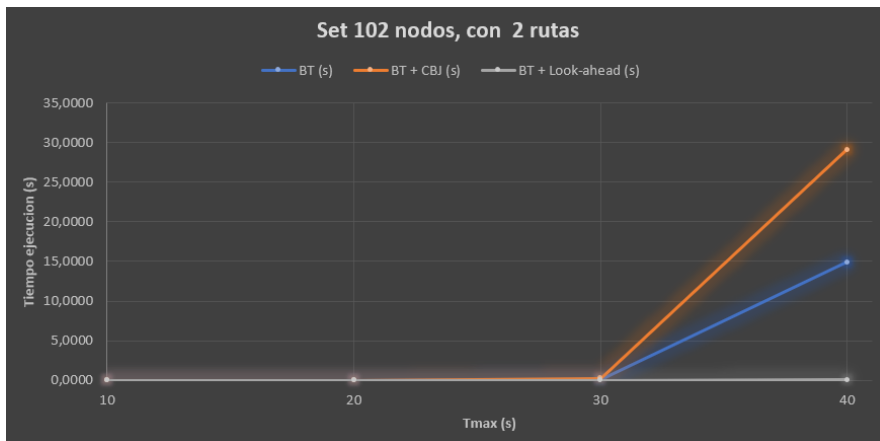


Figura 6: Relación entre el tiempo de ejecución de cada heurística y Tmax

Como al aumentar el Tmax, aumenta también la cantidad de instancias factibles del problema, las heurísticas de búsqueda completa comienzan a sufrir una subida circunstancial en su tiempo de ejecución, esto se debe a que estamos tratando de resolver un problema no determinista en tiempo polinomial.

En las figuras 7, 8 y 9, se revisa la suma del tiempo BT+CBJ y el tiempo de la combinatoria, llamada en las figuras como tiempo de ejecución total, para cada ruta de las tablas de las subsecciones 10.0.2, 10.0.3 y 10.0.4 respectivamente. Manteniendo constante el T_{max} .

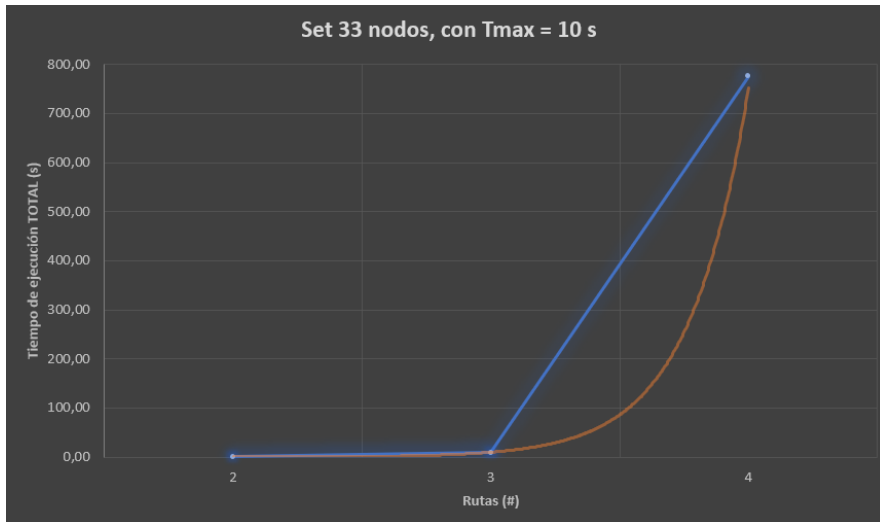


Figura 7: Relación entre el tiempo de ejecución Total (la suma de tiempo BT+CBJ y tiempo del binomial) Vs Numero de rutas. Con 33 nodos en 10 s de T_{max}

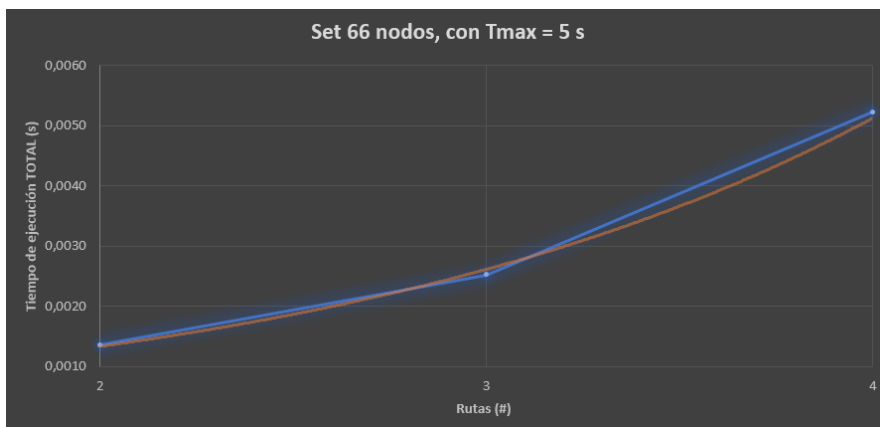


Figura 8: Relación entre el tiempo de ejecución Total (la suma de tiempo BT+CBJ y tiempo del binomial) Vs Numero de rutas. Con 66 nodos en 5 s de T_{max}

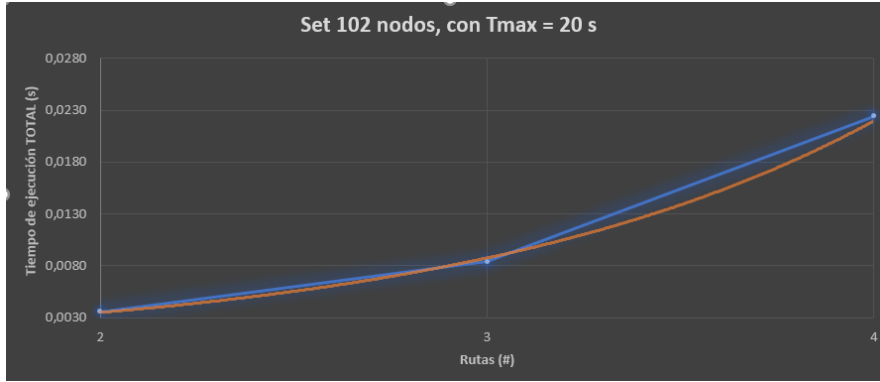


Figura 9: Relación entre el tiempo de ejecución Total (la suma de tiempo BT+CBJ y tiempo del binomial) Vs Numero de rutas. Con 102 nodos en 20 s de T_{max}

Donde la de color azul son líneas entre los puntos y la de color naranja es la exponencial del tiempo de ejecución total del programa, esto se debe a que se transformo el problema TOP, entonces la dependencia de las rutas no está asociado a la heurística, si no, a la combinatoria, la cual gráficamente es parecida a una exponencial.

9. Conclusiones

Para concluir, todas las estrategias apuntan a la resolución del problema. Sin embargo, se debe buscar la solución más óptima (con mayor puntuación), en el menor tiempo posible. Además, se debe buscar la estrategia que sirve para cada caso, ya que, hay heurísticas que resuelven casos que otras no pueden realizar en un tiempo determinado. Por lo cual, algunos métodos son más eficientes que otros, dependiendo de la instancia del problema.

En el caso de búsquedas completas de *Backtracking* es aconsejable utilizar un método de *Backtracking con Look-Ahead*, debido al resultado obtenido de los experimentos. Los beneficios son, que para instancias pequeñas se puede encontrar la mejor solución o encontrar que no existe ninguna solución. Para el caso que existan soluciones, se pueden encontrar todas las soluciones factibles. Como desventaja de este tipo de búsquedas, es que no se pueda lograr encontrar la mejor solución en un tiempo polinomial, tardando demasiado tiempo para instancias un poco más grandes (en relación a la cantidad de nodos, el tamaño de la ruta que se genere y el numero de compañeros). La mayoría de las técnicas estudiadas en otros trabajos, son técnicas incompletas, dado que TOP es NP-Hard y aunque se han logrado desarrollar varios algoritmos exactos, no existe un algoritmo que pueda encontrar todas las soluciones óptimas, dentro de un tiempo polinomial, en relación al tamaño de los nodos. Es por esto, que las técnicas incompletas se enfocan en mejorar la calidad de las soluciones, y otras tratan de mejorar el tiempo para encontrar soluciones.

El beneficio de encontrar mejores métodos para éste problema, es la diversidad de formas en que se puede plantear, logrando llevar este tipo de problema a

eventos cotidianos, por ejemplo, el manejo de drones no tripulados, en la entrega de paquetes o cartas, en donde los drones de manera conectada, tengan que decidir a través de inteligencia artificial, cual es el mejor camino en relación a la ubicación de sus compañeros, utilizando heurísticas incompletas.

10. Apéndice

10.0.1. Set_21_234

#Nodos: 21		#Rutas : 2				#Rutas	#Soluciones	Mejor Puntaje
Tmax	BT (s)	BT+CBJ(s)	BT+FC(s)	$\binom{N}{k}(s)$				
1	7,5	0,0012	0,0038	0,0002	0,0165	81	3.240	90
2	10	0,0080	0,0127	0,0005	0,2501	409	83.436	120
3	11,5	0,0169	0,0298	0,0015	1,9743	1153	664.128	140
4	12,5	0,0259	0,0595	0,0029	7,2196	2191	2.399.145	160
5	13,5	0,0497	0,1174	0,0060	29,2358	4388	9.625.078	190
6	15	0,1262	0,3139	0,0163	228,7810	12162	73.951.041	200
7	16	0,2392	0,5909	0,0321	871,0550	23574	277.854.951	200
\bar{X}		0,0667	0,1611	0,0085				

Tabla 3: Conjunto de pruebas de 21 nodos y 2 rutas

#Nodos:	21	#Rutas :				3			
Tmax	BT (s)	BT+CBJ(s)	BT+FC(s)	$\binom{N}{k}(s)$	#Rutas	#Soluciones	Mejor	Puntaje	
1	5	0,0002	0,0006	0,0000	0,0041	13	286	70	
2	6,7	0,0007	0,0024	0,0001	0,0475	39	9.139	70	
3	7,7	0,0015	0,0048	0,0003	0,6019	94	134.044	105	
4	8,3	0,0025	0,0059	0,0002	1,5677	129	349.504	105	
5	9	0,0043	0,0099	0,0003	7,6380	217	1.679.580	120	
6	10	0,0083	0,0131	0,0006	52,5444	409	11.319.484	120	
7	10,7	0,0135	0,0198	0,0010	224,9310	660	47.698.420	145	
\hat{X}		0,0044	0,0081	0,0004					

Tabla 4: Conjunto de pruebas de 21 nodos y 3 rutas

	#Nodos:	21	#Rutas :	4				
	Tmax	BT (s)	BT+CBJ(s)	BT+FC(s)	$\binom{N}{k}(s)$	#Rutas	#Soluciones	Mejor Puntaje
1	3,8	0,0000	0,0000	0,0000	0,0000	2	1	10
2	5	0,0001	0,0003	0,0000	0,0041	13	715	70
3	5,8	0,0002	0,0006	0,0000	0,0325	21	5.985	70
4	6,2	0,0003	0,0008	0,0000	0,1751	31	31.465	70
5	6,8	0,0004	0,0013	0,0001	0,5324	42	111.930	70
6	7,5	0,0007	0,0021	0,0001	7,6238	81	1.663.740	105
7	8	0,0010	0,0030	0,0002	26,2534	109	5.563.251	105
\widehat{X}		0,0004	0,0012	0,0001				

Tabla 5: Conjunto de pruebas de 21 nodos y 4 rutas

10.0.2. Set_33_234

	#Nodos:	33	#Rutas :	2				
	Tmax	BT (s)	BT+CBJ(s)	BT+FC(s)	$\binom{N}{k}(s)$	#Rutas	#Soluciones	Mejor Puntaje
1	7,5	0,0005	0,0015	0,0001	0,0027	30	435	90
2	10	0,0034	0,0095	0,0004	0,1231	215	23.005	150
3	12,5	0,0203	0,0529	0,0027	4,1402	1210	731.445	180
4	15	0,1054	0,2655	0,0143	114,8470	6212	19.291.366	220
5	17,5	0,5049	1,2141	0,0680	2.461,1200	28206	397.775.115	260
\widehat{X}		0,1269	0,3087	0,0171				

Tabla 6: Conjunto de pruebas de 33 nodos y 2 rutas

	#Nodos:	33	#Rutas :	3				
	Tmax	BT (s)	BT+CBJ(s)	BT+FC(s)	$\binom{N}{k}(s)$	#Rutas	#Soluciones	Mejor Puntaje
1	5	0,0002	0,0004	0,0000	0,0020	10	120	30
2	6,7	0,0006	0,0017	0,0001	0,0149	21	1.330	90
3	8,3	0,0017	0,0049	0,0002	0,1054	49	18.424	120
4	10	0,0063	0,0109	0,0004	9,2224	215	1.633.355	170
5	11,7	0,0159	0,0253	0,0013	392,2530	732	65.102.860	200
6	13,3	0,0352	0,0725	0,0036	8.612,1000	2017	1.365.589.680	230
\widehat{X}		0,0100	0,0193	0,0009				

Tabla 7: Conjunto de pruebas de 33 nodos y 3 rutas

	#Nodos: 33		#Rutas : 4					
	Tmax	BT (s)	BT+CBJ(s)	BT+FC(s)	$\binom{N}{k}(s)$	#Rutas	#Soluciones	Mejor Puntaje
1	3,8	0,0000	0,0001	0,0000	0,0003	4	1	20
2	5	0,0001	0,0003	0,0000	0,0019	10	210	30
3	6,2	0,0002	0,0007	0,0000	0,0248	18	3.060	90
4	7,5	0,0006	0,0016	0,0001	0,2259	30	27.405	100
5	8,8	0,0014	0,0040	0,0002	10,3667	75	1.215.450	140
6	10	0,0034	0,0095	0,0004	775,9360	215	86.567.815	190
\widehat{X}		0,0010	0,0027	0,0001				

Tabla 8: Conjunto de pruebas de 33 nodos y 4 rutas

10.0.3. Set.66.234

	#Nodos: 66		#Rutas : 2					
	Tmax	BT (s)	BT+CBJ(s)	BT+FC(s)	$\binom{N}{k}(s)$	#Rutas	#Soluciones	Mejor Puntaje
1	2,5	0,0000	0,0000	0,0000	0,0000	1	1	0
2	5	0,0002	0,0004	0,0000	0,0009	11	55	20
3	7,5	0,0021	0,0056	0,0002	0,0175	61	1.830	50
4	10	0,0298	0,0731	0,0016	1,0476	461	106.030	80
5	12,5	0,2814	0,6579	0,0138	71,1957	3673	6.743.628	180
\widehat{X}		0,0627	0,1474	0,0031				

Tabla 9: Conjunto de pruebas de 66 nodos y 2 rutas

	#Nodos: 66		#Rutas : 3					
	Tmax	BT (s)	BT+CBJ(s)	BT+FC(s)	$\binom{N}{k}(s)$	#Rutas	#Soluciones	Mejor Puntaje
1	1,7	0,0000	0,0000	0,0000	0,0000	1	1	0
2	3,3	0,0000	0,0001	0,0000	0,0003	5	10	15
3	5	0,0001	0,0004	0,0000	0,0021	11	165	20
4	6,7	0,0011	0,0029	0,0001	0,0919	37	7.770	60
5	8,3	0,0054	0,0139	0,0004	3,4735	119	273.819	95
6	10	0,0305	0,0755	0,0017	216,6070	461	16.222.590	110
\widehat{X}		0,0062	0,0155	0,0004				

Tabla 10: Conjunto de pruebas de 66 nodos y 3 rutas

	#Nodos: 66		#Rutas : 4					
	Tmax	BT (s)	BT+CBJ(s)	BT+FC(s)	$\binom{N}{k}(s)$	#Rutas	#Soluciones	Mejor Puntaje
1	1,2	0,0000	0,0000	0,0000	0,0000	1	1	0
2	2,5	0,0000	0,0000	0,0000	0,0000	1	1	0
3	3,8	0,0001	0,0001	0,0000	0,0003	5	5	20
4	5	0,0001	0,0004	0,0000	0,0048	11	330	20
5	6,2	0,0007	0,0019	0,0000	0,0194	15	1.365	20
6	7,5	0,0021	0,0057	0,0002	7,8716	61	521.855	80
\widehat{X}		0,0005	0,0014	0,0000				

Tabla 11: Conjunto de pruebas de 66 nodos y 4 rutas

10.0.4. Set_102_234

#Nodos: 102		#Rutas : 2						
	Tmax	BT (s)	BT+CBJ(s)	BT+FC(s)	$\binom{N}{k}(s)$	#Rutas	#Soluciones	Mejor Puntaje
1	10	0,0000	0,0001	0,0000	0,0004	3	3	30
2	20	0,0007	0,0019	0,0001	0,0017	14	91	64
3	30	0,1169	0,2568	0,0005	0,0747	103	5.253	101
4	40	14,9339	29,1069	0,0137	42,7578	2323	2.697.003	190
\widehat{X}		3,7629	7,3414	0,0036				

Tabla 12: Conjunto de pruebas de 102 nodos y 2 rutas

#Nodos: 102		#Rutas : 3						
	Tmax	BT (s)	BT+CBJ(s)	BT+FC(s)	$\binom{N}{k}(s)$	#Rutas	#Soluciones	Mejor Puntaje
1	6,7	0,0000	0,0000	0,0000	0,0000	1	1	0
2	13,3	0,0001	0,0002	0,0000	0,0004	4	4	46
3	20	0,0008	0,0020	0,0001	0,0064	14	364	79
4	26,7	0,0208	0,0483	0,0002	0,3669	51	20.825	117
5	33,3	0,6084	1,2644	0,0014	70,7708	285	3.817.670	175
6	40	14,6021	28,4415	0,0135	43.541,0000	2323	2.086.581.321	247
\widehat{X}		2,5387	4,9594	0,0025				

Tabla 13: Conjunto de pruebas de 102 nodos y 3 rutas

#Nodos: 102		#Rutas : 4						
	Tmax	BT (s)	BT+CBJ(s)	BT+FC(s)	$\binom{N}{k}(s)$	#Rutas	#Soluciones	Mejor Puntaje
1	5	0,0000	0,0000	0,0000	0,0000	1	1	0
2	10	0,0000	0,0001	0,0000	0,0000	3	1	30
3	15	0,0001	0,0003	0,0000	0,0002	4	1	46
4	20	0,0007	0,0019	0,0001	0,0205	14	1.001	79
5	25	0,0086	0,0207	0,0002	2,2283	41	101.270	123
6	30	0,1195	0,2624	0,0005	103,9990	103	4.421.275	164
\widehat{X}		0,0215	0,0476	0,0001				

Tabla 14: Conjunto de pruebas de 102 nodos y 4 rutas

Referencias

- [1] Alain y Speranza Maria Grazia Archetti, Claudia y Hertz. Metaheuristics for the team orienteering problem. *Journal of Heuristics*, 13(1):49–76, Feb 2007.
- [2] Dominique y Gendreau Michel Boussier, Sylvain y Feillet. An exact algorithm for team orienteering problems. *JOR*, 5(3):211–230, Sep 2007.

- [3] Prins C. A grasp \times evolutionary local search hybrid for the vehicle routing problem. page 35–53, 2009.
- [4] Rym Nesrine y Moukrim Aziz Dang, Duc-Cuong y Guibadj. A pso-based memetic algorithm for the team orienteering problem. In Anthony y Di Caro Gianni A. y Drechsler Rolf y Farooq Muddassar y Grahlf Jörn y Greenfield Gary y Prins Christian y Romero Juan y Squillero Giovanni y Tarantino Ernesto y Tettamanzi Andrea G. B. y Urquhart Neil y Uyar A. Şima Di Chio, Cecilia y Brabazon, editor, *Applications of Evolutionary Computation*, pages 471–480, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
- [5] Pierre y Gendreau Michel Feillet, Dominique y Dejax. Traveling salesman problems with profits. *Transportation Science*, 39(2):188–205, 2005.
- [6] Larry y Vohra Rakesh Golden, Bruce L. y Levy. The orienteering problem. *Naval Research Logistics (NRL)*, 34(3):307–318.
- [7] Yannis Marinakis. Multiple phase neighborhood search-grasp for the capacitated vehicle routing problem. *Expert Systems with Applications*, 39(8):6807 – 6815, 2012.
- [8] P. y Vanden Berghe G. y Van Oudheusden D. Souffriau, W. y Vansteenwegen. A path relinking approach for the team orienteering problem. *Computers y Operations Research*, 37(11):1853–1859, 2010. cited By 78.
- [9] Wouter y Berghe Greet Vanden y Oudheusden Dirk Van Vansteenwegen, Pieter y Souffriau. *Metaheuristics for Tourist Trip Planning*, pages 15–31. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- [10] Bochar Laarabi y Bouchaib Radi. A two-phase heuristic method for solving team orienteering problem. *International Journal of Advances in Mathematics*, Volume 2017:10–19, 2017.
- [11] I-Ming Chao y Bruce L. Golden y Edward A. Wasil. A fast y effective heuristic for the orienteering problem. *European Journal of Operational Research*, 88(3):475 – 489, 1996.
- [12] I-Ming Chao y Bruce L. Golden y Edward A. Wasil. The team orienteering problem. *European Journal of Operational Research*, 88(3):464 – 474, 1996.
- [13] Steven E. Butt y 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.
- [14] Hao Tang y Elise Miller-Hooks. A tabu search heuristic for the team orienteering problem. *Computers & Operations Research*, 32(6):1379 – 1407, 2005.
- [15] Liangjun Ke y Laipeng Zhai y Jing Li y Felix T.S. Chan. Pareto mimic algorithm: An approach to the team orienteering problem. *Omega*, 61:155 – 166, 2016.
- [16] Duc-Cuong Dang y Rym Nesrine Guibadj y Aziz Moukrim. An effective pso-inspired algorithm for the team orienteering problem. *European Journal of Operational Research*, 229(2):332 – 344, 2013.

- [17] Steven E. Butt y Tom M. Cavalier. A heuristic for the multiple tour maximum collection problem. *Computers & Operations Research*, 21(1):101 – 111, 1994.
- [18] Pieter Vansteenwegen y Wouter Souffriau y Dirk Van Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1 – 10, 2011.