Sección de Matemáticas
Universidad de La Laguna

Yaiza Alejandra Rodríguez García

Diseño de Rutas turísticas: itinerarios a pie en Santa Cruz de Tenerife

Design of Tourist Routes: walking itineraries in Santa Cruz de Tenerife

Trabajo Fin de Grado
Grado en Matemáticas
La Laguna, Septiembre de 2021

Dirigido por
Prof. D. Julio Brito Santana
Prof. D. José A. Moreno Pérez

Prof. D. Julio Brito Santana
Departamento de Ingeniería Informática y de Sistemas
Universidad de La Laguna
38200 La Laguna, Tenerife

Prof. D. José A. Moreno Pérez
Departamento de Ingeniería Informática y de Sistemas
Universidad de La Laguna
38200 La Laguna, Tenerife

Agradecimientos

Quiero agradecer en primer lugar a mi familia y amigos, por su apoyo incondicional, por dejarme contarles lo que iba aprendiendo, aunque no lo entendieran y por animarme a seguir adelante.

Gracias también a la Sociedad de Desarrollo del Ayuntamiento de Santa Cruz de Tenerife, por su interés y por darme acceso a muchos datos que han sido de gran utilidad a la hora de llevar a cabo este proyecto.

Agradecerles a todo el equipo docente de la facultad de Matemáticas por compartir sus conocimientos y ejercitar su paciencia para que los alumnos podamos entenderlos. Y por último agradecer a mis tutores de este proyecto su guía y consejo durante todo el proceso.

Muchas gracias a todos.

Yaiza Alejandra Rodríguez García La Laguna, 10 de septiembre de 2021

Resumen · Abstract

Resumen

Este trabajo tiene como objetivo diseñar rutas turísticas que se puedan realizar a pie en la ciudad de Santa Cruz de Tenerife considerando las preferencias generales de los turistas. Para resolver este problema se desarrollan y aplican modelos del tipo Orienteering Problem, en particular formulamos y encontramos soluciones utilizando los modelos Orienteering Problem Time Windows y Team Orienteering Problem Time Windows.

La metodología utilizada para encontrar las soluciones es una metodología aproximada, dado que este problema es complejo para encontrar soluciones a partir de un cierto número de puntos de interés. Así para la experimentación y obtención de resultados se aplica la metaheurística de Búsqueda por Entornos Variables Descendente, que es una variante de la Búsqueda por Entornos Variables. Los datos provienen del Observatorio de la Sociedad de Desarrollo del Ayuntamiento de Santa Cruz de Tenerife y del ISTAC.

Los resultados serán diferentes rutas, con diferentes duraciones, que tendrán como punto de partida la Plaza España, y que maximizarán las preferencias generales de los turistas.

Palabras clave: *Problemas de Diseño de Itineratios Turísticos, metaheurísticas, Problemas de Orientación, Búsqueda por entornos variables*

Abstract

The objetive of this work is to design tourist routes that can be done on foot in the city of Santa Cruz of Tenerife considering the general preferences of tourist. To solve this problem, are developed and applied models of the Orienteering Problem type, in particular we formulate and find solutions using the Orienteering Problem with Time Windows and Team Orienteering Problem with Time Windows models.

The methodology used to find the solutions is an approximate methodology, since this problem is complex to find solutions from a certain number of points of interest. Thus, for experimentation and obtaining results, the Variable Neighbourhood Descent metaheuristic is applied, which is a variant of the Variable Neighbourhood Search. The data come from the Observatorio de la Sociedad de Desarrollo del Ayuntamiento de Santa Cruz de Tenerife and the ISTAC.

The results will be differents routes with differents durations, which will have as their starting point the Plaza España, and which will maximize the generals preferences of tourists.

Keywords: *Trip Tourist Design Problem, Metaheuristic, Orienteering Problem, Variable Neighborhood Search*

Contenido

Introducción

Santa Cruz de Tenerife es una ciudad portuaria que se encuentra al noreste de Tenerife. La isla es un destino turístico a nivel mundial, un referente vacacional que cuenta con playas de arena negra, un buen clima y parques de atracciones; a su vez también atrae a amantes de la naturaleza y del deporte por su numerosa oferta de actividades acuáticas y al aire libre. Por otra parte, uno de los mayores atractivos de la isla es el parque Nacional del Teide que recibe millones de visitas al año. Eso conlleva que la economía de la isla dependa en gran medida del sector turístico, lo cual también afecta a la capital, Santa Cruz.

Siguiendo el ejemplo del Observatorio de la Sociedad de Desarrollo del Ayuntamiento de Santa Cruz, se puede clasificar a los turistas que vienen a Santa Cruz en tres categorías; los que se alojan en la ciudad; los excursionistas, los cuales se hospedan en otra parte de la isla y vienen a la ciudad de visita; y los cruceristas, que engloban a los turistas que llegan por mar en los cruceros y visitan la ciudad por un día. Los turistas que se alojan en la ciudad, son los que suelen disponer de más tiempo y más días para hacer rutas por la misma. Respecto a los excursionistas es difícil determinar el tiempo del que disponen, ya que la duración de la visita y las veces que vienen durante su estancia en la isla son complicadas de pronosticar. En cuanto a los cruceristas son los más predecibles, pues disponen de un día y de un número de horas limitado y quieren aprovechar al máximo el poco tiempo que tienen para visitar la ciudad.

A pesar de no ser una ciudad muy grande, Santa Cruz tiene muchos lugares de interés turístico, pero es muy probable que sus visitantes no tengan tiempo de verlos todos, por lo que solo apreciarán una parte. A la ciudad le interesa que los turistas que vienen se lleven una buena impresión, para que quieran volver o recomienden el destino y así prospere la economía. Por este motivo, resolver problemas de diseño de rutas turísticas que se ajusten a las preferencias de los turistas podría ser beneficioso. En el caso de los turistas alojados en la ciudad sería necesario crear

varias rutas para que puedan recorrer una cada día de estancia, mientras que los cruceristas solo necesitarían de una única ruta que se ajuste al tiempo disponible que pasarán en tierra. En cuanto a los excursionistas, si solo visitan Santa Cruz un día, sólo necesitarán una ruta, mientras que si tienen la intención de venir a la capital varios días sería mejor disponer de diferentes rutas.

Respecto a las preferencias de los turistas, se sabe que cada individuo tiene las suyas propias, pero nosotros tendremos en cuenta las preferencias obtenidas a partir del conocimiento previo de la experiencia en el destino. Basándonos en encuestas realizadas por el ISTAC, podemos saber qué actividades son las que más visitan los turistas en Santa Cruz. Este problema puede ser abordado teniendo en cuenta las preferencias a nivel individual, simplemente utilizando las valoraciones de preferencia de un turista que planifica su viaje o ponderando esta valoración con la obtenida de los datos previos disponibles. En este trabajo se utilizan para resolver el problema las preferencias obtenidas de información previa de los turistas, es decir se obtienen una ruta o varias (planificación recomendada) basada en la experiencia previa de otros turistas.

Otro aspecto importante en estos problemas es considerar los horarios de apertura de los lugares de interés turístico o en los que está disponible la visita. Por ejemplo, no tiene sentido que un turista visitara un museo cuando esté cerrado. Así, es importante disponer de la información de la hora inicial y final disponible de cada punto de interés, lo que denominamos ventana de tiempo y obtener una planificación que cuando el turista visite un punto de la ruta, esté disponible.

Este trabajo se concentra en resolver dos problemas que responden a dos necesidades de los turistas: el primer problema consistiría en diseñar una única ruta que planifique los lugares de interés a visitar, mientras que el segundo diseña un conjunto de rutas que puede realizar en un día o varios días. En ambos las rutas deben cumplir una serie de condiciones, como que los turistas las puedan hacer a pie, partiendo desde un punto de inicio (0), a una hora inicial y con diferentes duraciones (2, 3 o 4 horas). En los dos problemas el objetivo es crear rutas que maximicen las preferencias de los turistas respetando las limitaciones del tiempo límite de duración de la ruta y las ventanas de tiempo de cada punto de interés. En este trabajo ambos problemas se modelarán y formularan como problemas de programación lineal para posteriormente resolverlos utilizando un método aproximado, un procedimiento metaheurístico que dará como resultado diversas rutas factibles.

Respecto a los resultados del aprendizaje que con la realización de este trabajo se desarrollarán, se adquieren conocimientos de modelado y formulación de problemas reales, se amplían los conocimientos básicos sobre problemas de optimización combinatoria vistos en la asignatura de Optimización. También se manejan métodos de resolución aproximados y en qué difieren de los métodos exactos. Por último, se practican y amplían conceptos de programación utilizando el lenguaje Python para implementar un programa capaz de resolver los problemas planteados.

A continuación, los contenidos del documento están estructurados de la siguiente manera. En el capítulo 1, se comenzará definiendo un problema general de diseño de rutas turísticas para a continuación definir los problemas que resolveremos en el capítulo 3. También se describe la familia de modelos que nos va a ayudar a resolver los problemas planteados, los denominados problemas de orientación, así como algunos otros modelos relacionados. El siguiente capítulo, el capítulo 2, se centra en métodos de resolución aproximados; se definen las metaheurísticas y se presenta una de las formas en que se pueden clasificar. Este capítulo también describe de manera más detallada una de ellas, la búsqueda de entornos variables, presentamos su estructura, sus características y componentes, y algunas de sus variantes. Para terminar el capítulo, se detalla la variante de la búsqueda por entorno descendente que se va a utilizar en el siguiente capitulo para resolver los problemas planteados. En el capítulo 3 resolvemos los problemas particulares de diseño de una y varias rutas turísticas a pie en Santa Cruz de Tenerife y realizamos una serie de experimentos con el procedimiento metaheurístico utilizado para encontrar las mejores soluciones a dichos problemas. Por último, añadimos un capítulo de conclusiones y trabajos futuros.

1

Problemas de diseño de rutas turísticas: modelos y formulaciones

1.1. Introducción

En general cuando los turistas llegan a una ciudad quieren conocerla, recorrerla y descubrir sus secretos. Pero en las ciudades hay muchas cosas que ver y el tiempo de los turistas es limitado, por lo que plantea las siguientes preguntas: ¿Que sitios ver? ¿En qué orden los ven? ¿Como llegar de un sitio a otro? ¿Por dónde empezar? ¿Qué lugares estarán abiertos? ¿Dónde comer? etc.

Las ciudades quieren satisfacer las necesidades de los turistas que la visitan. Un problema de los destinos turísticos es que el turista sepa a donde ir o como moverse, y de ahí nace su interés en el diseño de rutas turísticas que satisfagan esas necesidades.

En este capítulo comenzaremos definiendo el problema del diseño de rutas turísticas de una forma general para después especificar los problemas que se resuelven en este trabajo, la obtención de itinerarios a pie en Santa Cruz de Tenerife. Conocido el problema, se presentan algunos de los modelos existentes en la literatura y se formulan. En particular se detallan los modelos que se van a abordar y resolver posteriormente, asociado con el supuesto práctico y con los datos disponibles.

1.2. El problema de diseño de itinearios turísticos

El problema de diseñar itinerarios turísticos o diseñar viajes o rutas turísticas, en inglés el Trip Tourist Design Problem (TTDP) ha sido definido por muchos autores. Entre otros autores Gavalas en [4] lo define como un problema de planificación de viajes para los turistas interesados en visitar múltiples puntos de interés o PdIs (de aquí en adelante utilizaremos el término PdI o PdIs para referirnos a los

puntos de interés). Estos mismos autores aportan que resolver el TTDP implica derivar diariamente recorridos turísticos que comprenden conjuntos ordenados de PdIs que coinciden con las preferencias del turista, maximizando la satisfacción del turista, teniendo en cuenta una multitud de parámetros y limitaciones (por ejemplo, distancias entre puntos de interés, tiempo estimado para ver cada PdI, horarios de apertura, ...) y respetando el tiempo disponible para hacer turismo.

El punto de partida de este problema, es la selección de lugares o PdIs. Estos PdIs representan los sitios turísticos que un turista puede visitar en una ciudad. Para representarlos de forma visual vamos a utilizar los grafos, donde los vértices representan los PdIs y los arcos los caminos que los unen.

El caso de que el turista disponga de tiempo y medios suficientes para visitarlos todos no es frecuente ni realista. Lo práctico y real es que el turista tenga limitado el tiempo disponible. Por eso, una de las características y restricciones principales del problema es el tiempo límite disponible, que denotamos T_{max}. Por otro lado, el turista tiene que escoger que PdIs visitar, y aquí entra en juego sus preferencias personales. La ruta será más satisfactoria o lo que es equivalente, reportará mayor beneficio, con la elección de los PdIs a visitar dentro de su preferencia. Para ello cada PdI tiene asociado una puntuación equivalente al grado de preferencia del mismo.

Determinados lugares de interés a visitar como restaurantes, museos o iglesias, tienen unos horarios de apertura y cierre, y los visitantes tienen que tenerlos en cuenta si quieren visitarlos y acceder a los mismos. Denominamos una ventana de tiempo, a la franja horaria donde el PdI está disponible para ser visitado. Esta ventana de tiempo del PdI tiene asociado dos valores, uno que indique la hora de apertura del PdI y la hora del cierre.

Un aspecto complementario a tener en cuenta es el tiempo que puede pasar o invertir en la visita de cada PdIs. La duración de la visita tiene que añadirse al tiempo a la duración del itinerario. Además, todos los turistas no tienen las mismas condiciones económicas. Los turistas disponen de un presupuesto y quieren tener en cuenta los costes de acceder a cada PdIs, en su caso. Teniendo en cuenta estos últimos dos parámetros, tendremos asociados a cada PdI cinco valores.

Por último, dentro de una ciudad todos los puntos están conectados, existe una red viaria y unos modos de transporte para acceder a ellos. Casi siempre hay más de una forma de llegar de un punto a otro, ya sea a pie, en coche, en transporte público, bicicleta, ..., y dependiendo del modo de transporte tardará más

o menos y tendrá un coste determinado diferente. Es por ello que existen varios caminos entre dos PdI, A y B. Los cuales tendrán asociado un coste expresado en tiempo, distancia o un coste monetario. La forma de representar este problema con grafos es con varios arcos que unan dos vértices concretos, cada arco tendrá asociado un valor que representará el coste con el que estemos trabajando, y cada vértice tendrá asociado unos valores que representarán los parámetros anteriores, obteniéndose el grafo siguiente:

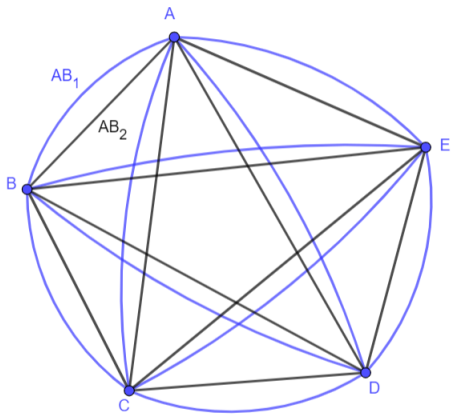


Figura 1.1: Grafo completo de un problema TTDP

En este ejemplo, el tamaño del arco no está relacionado de modo alguno con el coste que tenga asociado, las longitudes de los arcos son diferentes simplemente para simplificar el grafo. En la figura 1.1 se utilizan las líneas azules para representar un modo de transporte y las negras para otro modo de transporte.

Estos son algunos de los aspectos y restricciones que suelen tener estos problemas de encontrar itinerarios adecuado para el turista. También el problema puede ser planteado como obtener un itinerario, una ruta o tratar de encontrar varias rutas e itinerarios. Así es posible que el turista tenga diferentes rutas para varios días o en un mismo día quiera realizar varias rutas, en las cuales ir visitando los distintos PdIs disponible en el destino. En este caso estaríamos ante un problema de encontrar múltiples rutas con restricciones similares, y la satisfacción de preferencias.

En el capítulo 3 se plantearán dos problemas de diseño de rutas turísticas. En el primero, se desea encontrar una única ruta que maximice las preferencias generales de los turistas y que además cumpla con las restricciones de las ventanas de tiempo y un tiempo máximo de ruta. El segundo problema consiste en diseñar varias rutas, para diferentes días, y de diferente duración, que maximicen las preferencias generales de los turistas y tengan en cuenta las restricciones de las ventanas de tiempo y las duraciones máximas de las rutas. En ambos problemas sólo se tendrá en consideración un medio de transporte, y quedarán establecidos el mismo punto de partida y la misma hora de inicio para todas las rutas.

En la siguiente sección veremos algunos de los modelos que nos pueden ayudar para resolver estos problemas.

1.3. Modelos

Existen diferentes modelos descritos en la literatura para abordar el problema del diseño de itinerarios turísticos. En esta sección vamos a describir la familia de modelos OP, los cuales aparecen en la literatura con frecuencia y responden a las características descritas y asociada a nuestro caso de uso.

1.3.1. Orienteering Problem

El problema de la orientación o OP, surge y debe su nombre a los juegos de orientación. La idea es partir de un punto de salida o punto inicial, y el objetivo es llegar hasta el punto de llegada o punto final pasando previamente por una serie de ubicaciones específicas, donde cada ubicación o punto de interés tiene una puntuación predeterminada, y además cuenta con la restricción del tiempo, es decir, el recorrido que haga el jugador no puede exceder una cantidad estipulada de tiempo. Gana el jugador que completa el recorrido y consigue mayor puntuación.

La forma más sencilla de representar este modelo es mediante un grafo, donde los vértices serán los puntos de interés y los arcos los caminos que conectan los PdIs (puntos de interés). Donde, tenemos un punto de partida y otro de llegada, o bien, donde el punto de partida y llegada es el mismo.

En este modelo, no es obligatorio pasar por todos los PdIs, por lo que a un jugador solo se le sumara la puntuación de los PdIs por los que pase. Para representar la restricción del tiempo, hay que tener en cuenta que los arcos tendrán asociado un valor, que equivale al tiempo que se tarda en llegar de un PdI a otro, y la suma de estas cantidades no puede sobrepasar el tiempo límite estipulado.

En este modelo el objetivo o función objetivo, es maximizar la puntuación del jugador. Aunque también podría ser minimizar los costes (que en nuestro caso sería el tiempo) o se podría dar el caso de tener una función multiobjetivo.

Evidentemente a este juego se le puede añadir más restricciones que lo irán complicando cada vez más, como por ejemplo, pasar como máximo una vez por cada PdI o pasar por un PdI obligatoriamente. Algunas restricciones extras pueden dar paso a otros modelos particulares que mencionaremos más adelante.

Respecto al problema de rutas turísticas, este modelo correspondería a un problema bastante sencillo, con pocas restricciones. Se tendría una lista de PdIs, los cuales serían los puntos a visitar, cada uno con una puntuación que representa el grado de satisfacción o preferencia por el turista. Todos los puntos son accesibles y no es obligatorio que se visite cada uno de ellos. Todos los PdIs estarán conectados entre sí, y cada camino tendrá asociado un valor que representará el tiempo o la distancia que se tarda de un punto a otro.

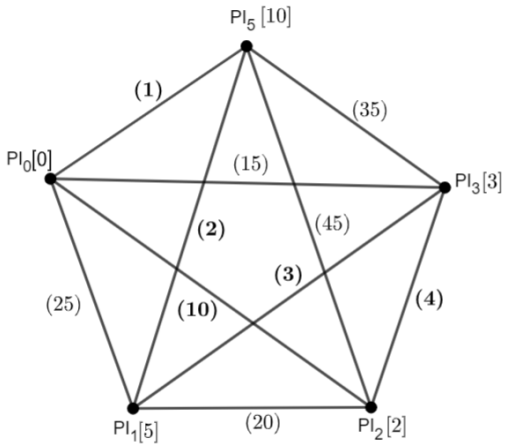


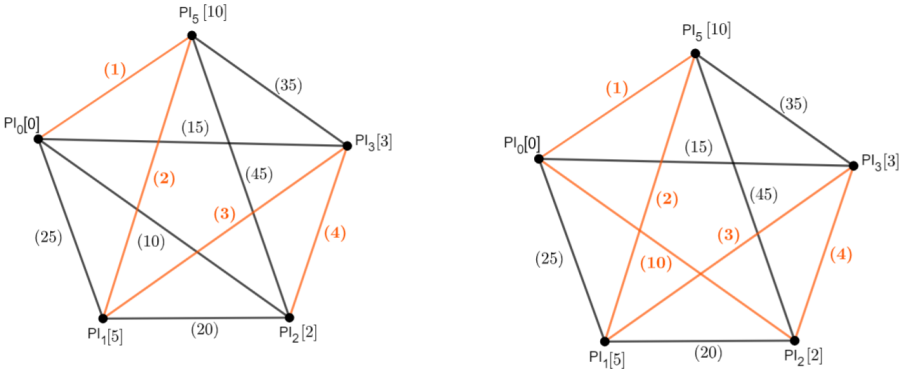
Figura 1.2: Representación de un modelo OP

Las restricciones son las siguientes:

- Un tiempo límite para la ruta.
- Un punto fijo de partida, es decir, obligamos al turista a empezar el recorrido en un punto concreto, tambien sería posible obligarlo a terminar en ese mismo punto.

- Solamente puede visitar un mismo lugar una vez, esta restricción es esencial en nuestro problema de rutas, ya que no interesaría al turista pasar una y otra vez por el mismo sitio teniendo tantos lugares a los que podría ir.

La solución que proporciona este modelo es una única ruta, que pretende ser la más atractiva para al turista.



(a) La mejor ruta que empieza en PI_0 y termina(b) La mejor ruta que empieza y termina en PI_0 en cualquier otro PI

Figura 1.3: Soluciones del modelo OP

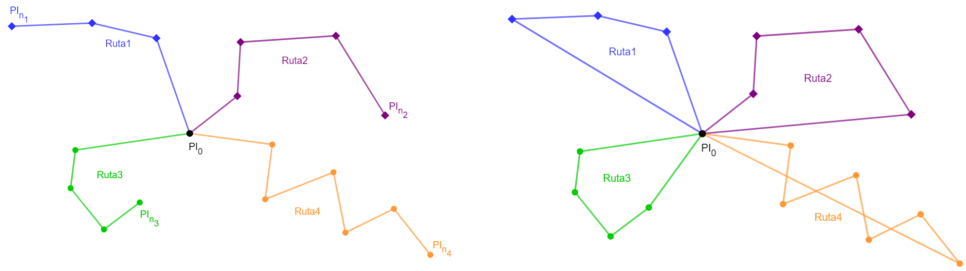
1.3.2. Team Orienteering Problem

El problema de orientación en equipo o TOP, como su nombre indica es un problema de orientación como el definido en el apartado anterior, con la diferencia de que ya no hay un único jugador, sino que son equipos de jugadores, donde cada equipo tendrá un número x de jugadores y por lo tanto se necesitarán x rutas, (una para cada jugador) en lugar de una. En este caso, se sumarían todas las puntuaciones de los jugadores del mismo equipo y ganaría el equipo con mayor puntación.

También podemos definir el OP como un caso particular del TOP, en el que los equipos están formados por un único jugador.

Aplicado al problema de diseño de rutas turísticas, este modelo no presenta ninguna diferencia al OP en cuanto a datos y restricciones. La diferencia con el OP radica en el número de rutas que tiene la solución de este modelo. Mientras que en el OP el resultado es una ruta para el turista, en este caso el resultado son varias rutas. Esas rutas no tienen por qué tener el mismo tiempo límite. Es decir,

un día se podría hacer una ruta de 3 horas mientras que al día siguiente podría ser de 6.



(a) Las mejores rutas que empiezan en PI_0 y terminan en cualquier otro PI

(b) Las mejores rutas que empiezan y terminan en PI_0

Figura 1.4: Soluciones del modelo TOP

Cabe destacar que, aunque las restricciones son las mismas que en el OP, la restricción de no pasar dos veces por el mismo PdI, tiene un matiz diferente, ya que ahora no sólo hace referencia a no pasar por él dentro de la misma ruta, sino a que tampoco haya otra ruta dentro de la misma solución que pase por ese mismo PdI.

1.3.3. Team Orienteering Problem with Time Windows

El problema de orientación en equipo con ventanas de tiempo o TOPTW, es un TOP al que se le añade una restricción particular que se denomina ventanas de tiempo y que restringe el momento en el que se puede pasar por un PdI, es como ponerle un horario de apertura a cada PdI, si el jugador pasa cuando está abierto se cuenta como que si ha pasado y se le suma la puntuación de ese PdI, pero si pasa cuando está cerrado es como si no hubiese pasado por ahí a efectos de la puntuación.

Es interesante observar este modelo desde la perspectiva del problema de las rutas turísticas. A continuación, se estudiará la variante más sencilla, OPTW (Orienteering Problem with Time Windows).

En este problema los PdIs tienen un horario, es decir, una franja horaria en la que están abiertos y se pueden visitar, y que fuera de esa franja horaria no es conveniente ir puesto que al estar cerrado no ofrece ningún atractivo para el turista. A este horario es a lo que se denomina ventanas de tiempo. Por lo tanto, en

este caso, los PdIs contarán con tres parámetros más, a parte de las preferencias del turista, que serían: la hora de apertura, la hora de cierre y la duración de la visita.

Y a las restricciones vistas en el modelo OP le añadimos la restricción de tener que llegar a cada PdI dentro de la ventana de tiempo y también tenemos que tener en cuenta la duración de la visita de cada PdI para no superar el tiempo límite establecido.

La formulación del problema que se explica a continuación está basada en el modelo explicado en [15]:

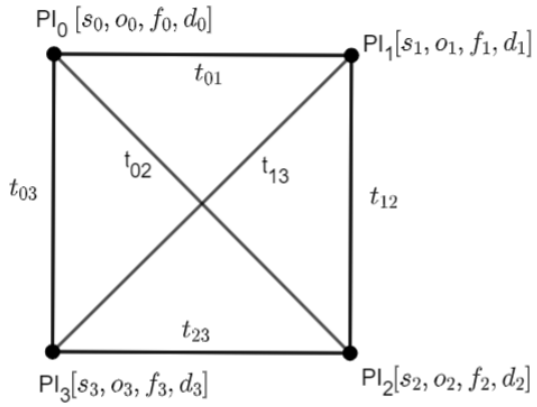


Figura 1.5: Representación de un modelo OPTW

Se define del conjunto de vértices $V = \{0, ..., v-1\}$ tal que v es el número de vértices del grafo. Para cada vértice $i \in V$ se definen los siguientes parámetros:

- s_i es la puntuación no negativa asociada al vértice i.
- o_i es el inicio de la ventana de tiempo en la que está abierto el vértice i.
- f_i es el final de la ventana de tiempo en que está abierto el vértice i
- d_i es la duración de la actividad en el vértice i.
- t_{ij} es el tiempo que se tarda de ir del vértice i al j.

También son necesarias dos variables binarias y una instrumental:

$$x_i = \begin{cases} 1 \text{ si visitamos } i \\ 0 \text{ en otro caso} \end{cases}$$

$$y_{ij} = \begin{cases} 1 \text{ si visitamos } j \text{ inmediatamente después de } i \\ 0 \text{ en otro caso} \end{cases}$$

z_i es el momento en el que llegamos a i.

Una vez definidos todos los parámetros y las variables que se necesitan, la formulación del problema es la que sigue:

$$\text{máx} \sum_{i \in V} s_i x_i \tag{1.1}$$

sujeto a:

$$\sum_{i \in V} y_{ij} \leq 1 \qquad \forall j \in V \tag{1.2}$$

$$\sum_{i \in V} y_{0i} = \sum_{i \in V} y_{i0} = 1 \tag{1.3}$$

$$\sum_{i,j \in V} t_{ij} y_{ij} + \sum_{i \in V} d_i x_i \leq T_{max} \tag{1.4}$$

$$z_i + d_i + (t_{ij} y_{ij}) - z_j \leq T_{max}(1 - y_{ij}) \tag{1.5}$$

$$o_i \leq z_i \leq f_i \qquad \forall i \in V \tag{1.6}$$

Donde T_{max} es el tiempo límite del que disponemos para la ruta turística.

La ecuación 1.1 es la función objetivo, que en este caso es maximizar la suma de las puntuaciones de los vértices que se visitan. Las ecuaciones de la 1.2 a la 1.6 muestran las restricciones que debe cumplir el problema:

- La ecuación 1.2 evita que pasemos dos o más veces por el mismo vértice.
- La ecuación 1.3 es opcional, se usaría en el caso de que se quiera que la ruta empiece y termine en el mismo PdI, a ese PdI se le asignaría el 0.
- Respecto a la ecuación 1.4 evita que la ruta sobrepase el tiempo máximo.
- En cuanto a la ecuación 1.5 ayuda a determinar la línea temporal de la ruta.
- Por último, la ecuación 1.6 responde a la restricción de las ventanas de tiempo, es decir, se asegura de visitar cada vértice después de que abra y antes de que cierre.

La solución vendrá en forma de vector, con los PdIs ordenados,

$$\boxed{0}\boxed{X_1}\boxed{\ldots}\boxed{X_r}$$

donde los X_i son los PdI escogidos para la ruta.

En el caso del modelo TOPTW, el problema resultaría muy similar que el anterior, los datos y restricciones son las mismos, pero hay ciertos cambios que se dan en la formulación y obviamente en el resultado. En el problema que vamos a formular, vamos a fijar el punto inicial en un PdI concreto, al que se denota como el vértice cero. Lo que significa que todas las rutas turísticas que se obtienen como resultado de este problema empezarán en el mismo PdI.

Respecto a la formulación del problema TOPTW:

Recordando que $V = \{0, ..., v-1\}$ es el conjunto de PdIs tal que v es el número de vértices y sea $P = \{1, \ldots, p\}$ donde p es el número de rutas que se quieren diseñar. Para cada vértice $i \in V$ se definen los siguientes parámetros:

- s_i es la puntuación no negativa asociada al vértice i.
- o_i es el inicio de la ventana de tiempo en la que está abierto el vértice i.
- f_i es el final de la ventana de tiempo en que está abierto el vértice i
- d_i es la duración de la actividad en el vértice i.
- t_{ij} es el tiempo que se tarda de ir del vértice i al j.
- $|V_p|$ es el número de vértices visitados en la parte p.
- cero = se usa para indicar el punto inicial y final.
- $|P| - 1$ es el número de ceros para separar las rutas.

Se necesitarán dos variables binarias y otra instrumental:

$$x_{ip} = \begin{cases} 1 \text{ si visitamos } i \text{ en } p \\ 0 \text{ en otro caso} \end{cases}$$

$$y_{ijp} = \begin{cases} 1 \text{ si visitamos } j \text{ inmediatamente después de } i \text{ en } p \\ 0 \text{ en otro caso} \end{cases}$$

z_{ip} es el momento en el que llegamos a i en la parte p.
Las ecuaciones siguientes definen el problema:

$$\text{máx} \sum_{p \in P} \sum_{i \in V} s_i x_{ip} \tag{1.7}$$

sujeto a:

$$\sum_{p \in P} x_{ip} \leq 1 \qquad \forall i \in V \tag{1.8}$$

$$\sum_{p \in P} \sum_{i \in V} y_{0ip} = \sum_{p \in P} \sum_{i \in V} y_{i0p} = |P| \tag{1.9}$$

$$\sum_{i,j \in V} t_{ij} y_{ijp} + \sum_{i \in V} d_i x_{ip} \leq T_{max} \qquad \forall p \in P \tag{1.10}$$

$$z_{ip} + d_i + (t_{ij} y_{ijp}) - z_{jp} \leq T_{max}(1 - y_{ijp}) \tag{1.11}$$

$$o_i \leq z_{ip} \leq f_i \qquad \forall i \in V \; y \; \forall p \in P \tag{1.12}$$

Se debe tener en cuenta que en este problema T_{max} no es un parámetro sino un vector:

T_1	...	T_p

donde p es el número de rutas que se quieren diseñar.

La ecuación 1.7 es la función objetivo de este modelo, que en este caso es maximizar la suma de las puntuaciones de los vértices que se visitan en cada ruta. Las ecuaciones de la 1.8 a la 1.12 muestran las restricciones que debe cumplir el problema:

- La ecuación 1.8 sirve para evitar que se pase dos veces por cualquier vértice, ya sea dentro de la misma ruta, o en rutas diferentes.
- La ecuación 1.9 obliga a la solución a empezar y terminar en el cero en cada una de las rutas.
- Respecto a la ecuación 1.10 evita que la ruta sobrepase el tiempo máximo, en cada ruta.
- En cuanto a la ecuación 1.11 ayuda a determinar la línea temporal de la ruta, en cada una.
- Por último, la ecuación 1.12 responde a la restricción de las ventanas de tiempo, es decir, se asegura de visitar cada vértice después de que abra y antes de que cierre.

En este caso la solución del problema también vendrá en forma de un único vector en el que se especificará el orden de los vértices que se visitan. Pero ya que

la solución consta de varias rutas, los ceros del vector marcarán el final de una ruta y el comienzo de la siguiente.

0	X_{11}	...	X_{1r_1}	0	X_{21}	...	X_{2r_2}	0	...	0	X_{p1}	...	X_{pr_p}

Estos son los dos modelos que se utilizarán para resolver los problemas planteados en el capítulo 3. Para el problema del diseño de una única ruta, se empleará el modelo OPTW mientras que el problema de diseñar varias rutas se ajustará a un modelo TOPTW. con la única diferencia de que en los problemas planteados no vamos a forzar a la solución a que termine en el mismo punto, solo se fijará el PdI de partida.

1.3.4. Otras variantes de la familia OP

En este apartado describe otras dos variantes del modelo TOP, la variante Multimodal, y Time Depend, que podemos encontrar de forma más detallada en [15] y [11], respectivamente.

- **MultiModal - Team Orienteering Problem with Time Windows:**

  Hasta ahora, sólo nos hemos planteado que existe un único camino o una única forma de llegar de un PdI a otro PdI concretos, pero el problema de orientación en equipo con ventanas de tiempo multimodal plantea la existencia de varias opciones para llegar del PdI_B al PdI_C y cada opción tendrá su propio coste que puede ser igual o diferente entre dos mismos PdIs.

  Desde el punto de vista del problema de rutas turísticas, estos caminos equivalen a los medios de transporte que se podrían usar, sabemos que dentro de una ciudad podemos movernos caminando desde cualquier PdI a otro, pero también existen otros medios de transporte, como el autobús, que no tiene por qué conectar todos los puntos de interés pero que si conectan algunos, y pueden suponer una reducción significativa del tiempo que se tarda en ir de un PdI a otro.

  En cuanto a la solución de este problema, constará de dos vectores, uno que especifique los PdI que se visitan de forma ordenada y otro que indica el tipo de transporte utilizado para ir de cada uno al siguiente.

- **Time Depend - Team Orienteering Problem with Time Windows:**

  En esta variante del TOPTW, se plantea la posibilidad que el tiempo que se tarda de un vértice a otro no sea un parámetro fijo sino una función no negativa

que varie en función del tiempo, por lo tanto, dependiendo de la hora de inicio del recorrido el trayecto del PdI_A al PdI_B se puede tardar más o menos.

Esta variante refleja el hecho, de que en una ciudad no siempre se tarda lo mismo de llegar del punto A al punto B, ya que existen horas en el día en que hay una mayor afluencia de personas que se mueven entre esos puntos, lo que se conoce como la hora punta.

1.4. Algunos modelos generales

Esta sección describe dos modelos más generales, que también pueden ser utilizados para planificar rutas turísticas cuando sus objetivos y limitaciones estén adaptados. Estos modelos tienen la particularidad que la ruta construida tiene que visitar todos los nodos o PdIs que están incluidos en el problema. Estos son, el problema del vendedor viajero [16], y el problema de enrutamiento de vehículos [18].

- **El problema del viajante de comercio** o TSP parte de la situación de un agente que tiene que visitar determinados PI en diferentes ciudades para vender su producto, las condiciones de este problema son que tiene que pasar una vez por todos los PdI, y el punto inicial y final del trayecto es el mismo. Respecto al objetivo, lo frecuente en este caso no es maximizan los beneficios, sino que se trata de minimizar los costes (distancia o tiempo), es decir, se busca la ruta que recorra todos los PdIs en el menor tiempo o distancia posible.

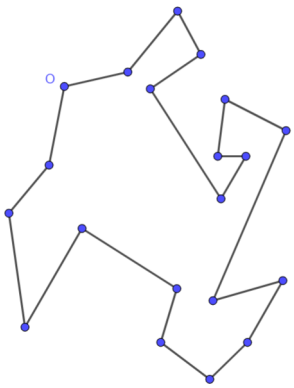


Figura 1.6: La solución es un circuito cerrado

El OP puede considerarse una versión de este problema, este caso se exige la condición de pasar por todos los PdI exactamente una vez, salvo el punto inicial, que al ser el mismo que el final lo se visita dos veces. Como en este caso, se va a pasar por todos los PdI no es necesario que estos tengan una puntuación, ya que sería irrelevante para el ejercicio, lo único que se debe tener en cuenta es el coste del trayecto entre dos PdIs.

- **El problema de rutas de vehículos** o VRP, responde a un problema de logística que puede presentarse a cualquier empresa que tenga que mover la mercancía desde un almacén a muchos puntos de venta y que además disponga de una flota de vehículos. Por lo tanto, si la flota consta de x número de vehículos, el objetivo es diseñar x rutas de transporte que cubran todos los puntos de venta y que minimice el coste del transporte.

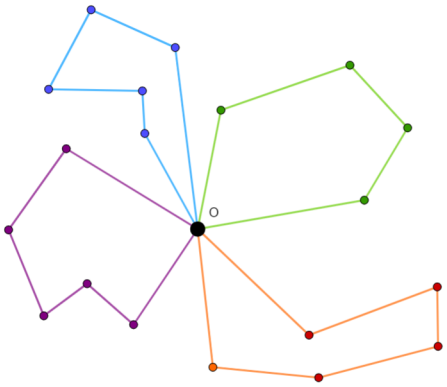


Figura 1.7: La solución son varios circuitos cerrados

Se puede plantear este modelo como una versión del TSP para varias rutas. De forma equivalente al descrito como TOP con la condición de pasar por todos los PdIs exactamente una vez, teniendo en cuenta que el punto inicial y final, debe ser el almacén y por lo tanto el mismo y que se pasará por ese punto un total de 2 veces con cada vehículo.

Con este modelo se termina el capítulo 1 para dejar paso a los métodos de resolución aproximados del capítulo 2.

2

Métodos aproximados

2.1. Introducción a métodos de resolución

Los problemas estudiados en el capítulo anterior pueden tener dos tipos de soluciones: las soluciones exactas, donde lo que se obtiene es la mejor solución o solución óptima; o soluciones aproximadas, que no tienen por qué ser las mejores, pero son factibles y bastantes buenas.

Los problemas planteados y descritos en el capítulo anterior son problemas complejos. En general, tienen establecidas muchas limitaciones, algunas difíciles de cumplir como las ventanas de tiempo, y además si tienen un gran número de nodos o PdIs, no siempre se puede garantizar encontrar una solución óptima en un tiempo de computación que no sea excesivo. En estos casos podría ser más razonable encontrar soluciones que sean factibles y buenas en un periodo de tiempo más corto y razonable. En este capítulo nos vamos a centrar en las soluciones aproximadas, estas soluciones se obtienen mediante procedimientos heurísticos o metaheurísticos.

En la siguiente sección se introducirá el concepto de las metaheurísticas, su definición y clasificación, para después describir una de ellas, la búsqueda por entornos variables. Se presentan diferentes variantes y en la última sección una variante en particular, la búsqueda por entornos variables descendente VND, que será la metaheurística que se utilizará, en el capítulo 3 en la experimentación para resolver el problema.

2.2. Metaheurísticas

Según la definición que utiliza Francisco Herrera en [6], se pueden definir a las metaheurísticas como una familia de algoritmos aproximados, que suelen ser

procedimientos iterativos, y guían a una heurística subordinada con el objetivo de recorrer y explotar adecuadamente el espacio de búsqueda.

Las metaheurísticas suelen ser procedimientos más rápidos que los métodos exactos, por eso en las ocasiones en las que no es necesario encontrar la mejor solución, sino que basta con una solución lo suficientemente buena, es recomendable usar este tipo de métodos.

Para establecer si un método es apropiado para resolver un problema, es necesario establecer un balance adecuado entre los dos conceptos siguientes:

- **Intensificación**: hace referencia al esfuerzo empleado en recorrer el espacio de búsqueda cercano a la solución actual.

- **Diversificación**: es la cantidad de esfuerzo que se emplea en la búsqueda de mejores soluciones en regiones alejadas de la solución actual.

Un balance adecuado entre estos conceptos ayuda a que el algoritmo de búsqueda optimice su tiempo de trabajo centrándose en regiones con buenas soluciones y desechando aquellas regiones que no las tiene.

En [6] se puede encontrar una posible clasificación de las metaheurísticas como la siguiente :

- **Basadas en métodos constructivos**: Estos métodos parten de una solución vacía y van añadiéndole componentes hasta obtener una solución adecuada. Un ejemplo de este tipo de metaheurísticas es GRASP.

- **Basadas en trayectorias**: Estas metaheurísticas parten de una solución inicial y aplicando un algoritmo de búsqueda local, van aplicando cambios a la solución de partida. Si se representa la búsqueda local en una gráfica se vería que sigue una trayectoria en el espacio de búsqueda, de ahí su nombre. Algunos ejemplos son: Búsqueda Local, Búsqueda TABU, Búsqueda por entornos variables, ...

- **Basadas en poblaciones**: Esta clase de metaheurística considera varios puntos en el espacio de búsqueda que evolucionan de forma paralela. Un ejemplo serían los Algoritmos genéticos.

A continuación, en la siguiente sección se tratará una metaheurística basada en la trayectoria, la Búsqueda por Entornos Variables o VNS por sus siglas en inglés.

2.3. Búsqueda por Entornos Variables

Esta sección describe la metaheurística VNS, su estructura, características y componentes, en sus diferentes variantes [12] y [13].

Pero antes, es necesario definir el concepto de entorno de una solución. El entorno de una solución es un conjunto de soluciones cercanas a la solución de partida que se obtienen aplicando pequeños cambios a la solución inicial. Estos cambios pueden realizarse en cualquier posición de la solución, también se pueden combinar.

En [12] se define la VNS como una metaheurística que se basa en la idea de un cambio sistemático de vecindario, tanto en la fase de descenso para encontrar un óptimo local como en una fase de perturbación para salir del valle correspondiente. Originalmente se diseñó para aproximar soluciones de problemas de optimización combinatoria, y más adelante se amplió para abordar problemas de enteros mixtos y problemas no lineales."

Un problema de optimización determinista se puede definir como:

$$min\{f(x)|x \in X \wedge X \subset S\}$$

donde S denota el conjunto de todas las soluciones y X es el conjunto de las soluciones factibles, por lo tanto si $x \in X$, significa que x es una solución factible, y f representa una función objetivo de valor real.

S va a determinar el tipo de problema al que nos enfrentamos, si S es grande pero finito, entonces estamos ante un problema de optimización combinatoria, en cambio si $S = R^n$ entonces se trata de un problema de optimización continuo. A continuación se estudiará el primer caso, ya que es con el que se va a trabajar.

Diremos que $x^* \in X$ es una solución óptima si $\forall x \in X$ se cumple que:

$$f(x^*) < f(x)$$

Las VNS se sustentan en tres principios:

1. Una solución óptima local de un determinado entorno de soluciones no tiene porqué ser una solución óptima en otro entorno.

2. Una solución óptima global es una solución óptima local en cualquier entorno.

3. En muchos problemas, los óptimos locales con el mismo o diferente entorno están relativamente cerca.

Un aspecto esencial en una VNS, es la opción de poder escapar de vecinadario (entorno) a otro, para evitar caer en óptimos locales, eso se consigue con la siguiente función:

- **función** Cambio de vecindario (x, x', k) , donde x es la mejor solución que se tiene, f es la función objetivo, x' es la solución a comparar y k es el vecindario.

Función: Cambio de vecindario (x, x', k)
if $f(x') < f(x)$ **then**
  $x \leftarrow x'$
  $k \leftarrow 1$
end
else
  $k \leftarrow k + 1$
end
Return : x, k

Algorithm 1: Cambio de vecindario

La siguiente función ayuda a escoger de forma aleatoria el $x' \in N_k(x)$ necesario para compararlo con la mejor solución en la función anterior. Teniendo en cuenta que $N_k(x)$ es el conjunto de soluciones factibles en el vecindario k, de forma que $N_k(x) = \{x^1, \ldots, x^{|N_k(x)|}\}$.

- **función** Agitar (x, k) : encuentra una solución aleatoria de un vecindario k.

Función: Agitar (x, k)
$w \leftarrow 1 + Rand(0, 1) * |N_k(x)|$
$x' \leftarrow x_w$
Return : x'

Algorithm 2: Agitar

A continuación se explicarán dos métodos que nos pueden ayudar a alcanzar un óptimo local.

- **función** Mejor mejora (x) : consiste en encontrar un mínimo local, que se denota por x', recorriendo todo el espacio de búsqueda $N(x)$ para cada x.

Función: Mejor mejora (x)

repeat
  $x' \leftarrow x$
  $x \leftarrow argmin_{y \in N(x)} f(y)$
until $f(x) \geq f(x')$
Return : x'

Algorithm 3: Mejor mejora

- **función** Primera mejora (x), en este caso, no se recorre todo el espacio de búsqueda, ya que desde que se encuentre una solución que mejore la actual, se cambia esa por la actual y se repite el proceso hasta que no se encuentre ninguna mejor.

Función: Primera mejora (x)

repeat
  $x' \leftarrow x$
  $i \leftarrow 0$
  repeat
    $i \leftarrow i + 1$
    $x \leftarrow argmin\{f(x), f(x^i)\}, x^i \in N(x)$
  until $(f(x) < f(x'))$ *or* $i = |N(x)|$
until $f(x) \geq f(x')$
Return : x'

Algorithm 4: Primera mejora

En ocasiones, el método de la mejor mejora puede llevar un tiempo excesivo, por eso en esas ocasiones es mejor emplear el método de la primera mejora ya que tarda menos tiempo en ejecutarse.

2.3.1. Variantes de VNS

Cuando hablamos de variantes de VNS, nos referimos a los diferentes enfoques que se pueden dar a la búsqueda de soluciones siendo fiel a la variabilidad del entorno de soluciones y la aplicación de una búsqueda local. Veremos algunas de las variantes que podemos encontrar en [13]

Estas estructuras pueden ser deterministas, aleatorias o ambas cosas. Cuando se dice que una estructura es determinista es porque los cambios que sufrirá la solución de partida se realizarán de una manera determinada, mientras que en el caso de que sea estocástica o aleatoria, se aplicarán los cambios de forma aleatoria a la solución. Veamos cuatro variantes de VNS:

- **Búsqueda de entorno variable descendente** o VND, partiendo de una solución actual y un entorno, hacemos una búsqueda local que determine iterativamente una solución óptima de ese entorno, una vez hallada, es reemplazada por la solución actual y se cambia el entorno en el que se estaba trabajando.

  Esta estructura de búsqueda es de tipo determinista. El algoritmo que la describe es:

```
entrada: x, {N_k, k = 1..k_max}, t_max
salida : x
begin
    t ← 0
    while t ≤ t_max do
        k ← 1
        repeat
            x' ← Mejora(x)
            if f(x') < f(x) then
                x ← x'
                k ← 1
            end
            else
                k ← k + 1
            end
        until k > k_max
    end
end
```

Algorithm 5: VNS Descendente; VND

- **Búsqueda de entorno variable básica** o BVNS: combina una estrategia determinista y aleatoria, ya que, como se verá a continuación, la estrategia consiste en que primero se selecciona una solución de un entorno (determinista), para después escoger al azar otra solución y aplicar la búsqueda desde ella (aleatoria). El algoritmo a seguir es:

entrada: x, $\{N_k, k = 1..k_{max}\}$, t_{max}
salida : x

begin
  $t \leftarrow 0$
  while $t \leq t_{max}$ **do**
    $k \leftarrow 1$
    repeat
      $x' \leftarrow Agita(x, k)$
      $x'' \leftarrow Mejora(x')$
      if $f(x'') < f(x)$ **then**
        $x \leftarrow x''$
        $k \leftarrow 1$
      end
      else
        $k \leftarrow k + 1$
      end
    until $k > k_{max}$
  end
end

Algorithm 6: VNS Básica; BVNS

- **Búsqueda de entorno variable reducida** o RVNS es una estrategia aleatoria, que suele ser útil cuando la búsqueda local es muy costosa. Los pasos a seguir son:

entrada: x, $\{N_k, k = 1..k_{max}\}$, t_{max}
salida : x

begin
  $t \leftarrow 0$
  while $t \leq t_{max}$ **do**
    $k \leftarrow 1$
    repeat
      $x' \leftarrow Agita(x, k)$
      if $f(x') < f(x)$ **then**
        $x \leftarrow x'$
        $k \leftarrow 1$
      end
      else
        $k \leftarrow k + 1$
      end
    until $k > k_{max}$
  end
end

Algorithm 7: VNS Reducida; RVNS

- **Búsqueda de entorno variable general** o GVNS es una combinación de las estrategias BVNS y VND. A continuación se verá cuales son los pasos de esta estrategia:

```
entrada: x, {N_k, k = 1..k_max}, {N'_j, j = 1..j_max}, t_max
salida  : x
begin
    t ← 0
    while t ≤ t_max do
        k ← 1
        repeat
            x' ← Agita(x, k)
            x'' ← VND(x', {N'_j, j = 1..j_max}, t_max)
            if f(x'') < f(x) then
                x ← x''
                k ← 1
            end
            else
                k ← k + 1
            end
        until k > k_max
    end
end
```

Algorithm 8: VNS General; GVNS

2.4. Variable Neighbourhood Descent o VND aplicada

En esta sección se va a explicar la VND que se va a utilizar para resolver los problemas TTDP en el siguiente capítulo.

Las soluciones de este problema vendrán en forma de vector. Por lo tanto, una solución x será de la forma $x = [x_0, \ldots, x_{n-1}]$ donde n es el tamaño de x.

Es indispensable crear una función cuyo objetivo sea verificar si la solución de entrada cumple las restricciones del problema o no, y devuelva True o False, respectivamente. Como esta función variará según las restricciones de cada problema, no se va a ver su estructura ni algoritmo, pero es importante saber que existe, y para futuras referencias en esta sección, a esta función se la denominará:

Función *factibilidad* (x)

Para empezar, se muestra el algoritmo que genera la solución inicial:

```
Función: GeneradorSolucion
begin
    m ←número máximo de iteraciones
    n ← número de elementos disponibles
    x ← [0]
    i ← 0
    while i ≤ m do
        j ← random(0, n)
        if j ∈ x then
            i ← i + 1
        end
        else
            x' ← x.append(j)
            factible ← factibilidad(x')
            if factible == True then
                x ← x'
                i ← i + 1
            end
            else
                x' ← x
                i ← i + 1
            end
        end
    end
end
Return : x
```

Algorithm 9: Generador de la solución inicial

A continuación, se explican los principales movimientos que se utilizarán para resolver el modelo OPTW en el siguiente capítulo:

- **Intercambio interno**: consiste en realizar intercambios de forma iterativa y determinada entre dos valores del vector solución x, creando así una nueva solución que contiene los mismos puntos en diferente orden. Cada vez que se realiza un intercambio, se comprueba que la solución obtenida x' es factible, en caso afirmativo se comparan el tiempo que consume cada solución, si resulta que la nueva solución consume menos tiempo que la actual, se renombra como solución actual y se vuelve a repetir el proceso desde el principio. Una vez realizados todos los intercambios posibles entre dos valores de la solución, se amplía el tamaño del entorno de búsqueda y se comienza de nuevo el procedimiento, así hasta que se alcanza el tamaño máximo k_{max}.

- **Intercambio externo**: consiste en intercambiar un valor de la solución por uno que no pertenece a ella. Este intercambio se realiza de forma determinada

e iterativa. Cada vez que se realice un intercambio se debe verificar que la nueva solución sea factible, en el caso de que lo sea, se comparan las funciones objetivos de ambas soluciones, y se desecha la que tenga menor valor en la función objetivo. Una vez que se han realizado todos los intercambios posibles en todas las posiciones de la solución sin obtener una solución mejorada, se procede a aumentar el tamaño del entorno de búsqueda y se repite el proceso. El proceso se termina cuando se alcanza el tamaño de entorno máximo y no se obtiene ninguna mejora.

- **Adición**: escoge un elemento que no pertenezca a la solución y lo añade a la solución actual siguiendo un proceso determinado e iterativo. Cada vez que se añade un elemento nuevo a la solución se comprueba si continúa siendo factible. En caso de que si lo sea se deja el punto añadido y se procede a elegir otro. En caso negativo se retira ese elemento de la posición en la que se añadió y se prueba en otra posición. Este proceso se realiza hasta que se han comprobado todas las posiciones con todos los elementos disponibles. Una vez realizado este procedimiento con un elemento se amplía el tamaño del entorno de búsqueda y se repite el proceso, hasta alcanzar el tamaño máximo de entorno k_{max}. Cabe destacar que en este movimiento simplemente se comprueba si es factible o no la nueva solución, no se compara de ninguna manera con la solución anterior, ya que es obvio que al añadir un elemento más a la solución, esta siempre tendrá mayor valor en la función objetivo.

Respecto a los movimientos que se aplicarán al modelo TOPTW, son bastantes similares a los del modelo OPTW, por eso se explicarán en el capítulo 3. Para terminar esta sección se va a analizar la estructura de la VND, con los movimientos que se han definido anteriormente:

1. Inicialización :
   * Seleccionar un conjunto de estructuras de entornos N_k tal que $k = 1, \ldots, k_{max}$
     # Es decir, tenemos que decidir el tamaño del entorno con el que se va a trabajar, este número entero será k_{max}, también tendremos que decidir qué movimientos hacer y en qué orden.

   * Seleccionar una solución inicial x
     # Tal y como explicamos anteriormente tenemos que generar una solución inicial aleatoria.

   * Iniciamos en $k = 1$
     # k será un contador que utilizaremos para llevar la cuenta del tamaño del entorno en el que se está trabajando.

2. Iteraciones:
   Repetir :
   a) Explorar el entorno $N_k(x)$: encontrar la mejor solución del entorno $N_k(x)$, x'.
   b) Moverse o no: se pueden dar dos situaciones:
      − x' es mejor que x, reemplazamos x por x', y volvemos a $k = 1$
      − en caso contrario, $k = k + 1$
   Hasta que $k > k_{max}$

   # Aquí, esencialmente se aplican los movimientos secuencialmente, ya que en cada movimiento se comparan las funciones objetivos de x y x', para determinar cuándo una solución es mejor que otra.

   Primero se trabaja con el tamaño de entorno $k = 1$, que significa aplicar los cambios de un elemento en un elemento, para después ir agrandando el tamaño de la estructura de entorno hasta alcanzar el tamaño máximo k_{max}. El bucle se detendrá cuando después de haber aplicado todos los movimientos a una solución no se haya obtenido ninguna mejor.

   Esta será la metaheurística que se emplea en el siguiente capítulo para resolver el problema de rutas turística que se planteado. El lenguaje de programación utilizando para imprimir el código de dicho procedimiento es Python.

3

Diseño y desarrollo de rutas en Santa Cruz de Tenerife

3.1. Descripción del problema y los datos

Santa Cruz de Tenerife, es una ciudad portuaria que vive del turismo, eso hace que sea de vital importancia recibir la visita de turistas cada año y que se marchen satisfechos con lo que han visto. Por ello, se plantea la cuestión de qué forma los turistas pueden ver Santa Cruz y quedar lo más satisfechos posibles, así surge para la ciudad el problema de rutas, basándose en las estadísticas publicadas por el ISTAC sobre sus preferencias para medir la satisfacción de los turistas.

En este capítulo se plantean dos problemas, el primero consiste en diseñar una única ruta turística que parte de la plaza España como punto de inicio a las 9:00 a.m. y tenga en cuenta los conocimientos previos de preferencias generales de los turistas. Esta ruta tendrá que satisfacer las siguientes restricciones: la ruta debe ajustarse a un tiempo máximo establecido y no podrá sobrepasarlo, también tendrá que respetar las ventanas de tiempo y no deberá pasar dos veces por ningún PdI. Existen muchos modelos que pueden resolver este problema, y en este caso se utilizará uno de los más sencillos un OPTW (explicado en el capítulo 1).

El segundo problema consiste en diseñar varias rutas turísticas de diferente duración, para el turista que tiene varios días disponibles o que quiere hacer varias rutas para recorrer los lugares de interés de la ciudad. En este caso, todas las rutas parten del mismo punto y a la misma hora (Plaza España, 9:00 a.m.) y cuenta con las mismas restricciones que el problema anterior, teniendo en cuenta que la restricción de no poder pasar dos veces por el mismo PdI, afecta no sólo a los puntos de la misma ruta, sino a los puntos de todas las rutas de la solución. El modelo utilizado es un TOPTW (también se explicó en el capítulo 1). Respecto a la función objetivo será maximizar las preferencias generales de los turistas.

En ambos problemas vamos a utilizar las mismas preferencias. Estas preferencias se han extraído de encuestas realizadas por el ISTAC, y han sido estudiadas por el Observatorio de la Sociedad de Desarrollo del Ayuntamiento de Santa Cruz de Tenerife. Se debe tener en cuenta que estas preferencias son de un conjunto de turistas, no de un turista en particular. Se podrían plantear diferentes problemas teniendo en consideración las preferencias particulares de cada turista de varias formas; se podría tener en cuenta solo la opinión particular de cada turista o bien se podría ponderar con las preferencias generales, esto se asemejaría más a un sistema de recomendación. En cualquier caso, la elección de cómo obtener las preferencias de los turistas no afectan a la metodología de resolución del problema, ni al modelo escogido, simplemente marca una diferencia en cuanto a quien está dirigida la solución que se obtiene.

El primer paso para resolver los problemas, es recabar la información necesaria. Los datos utilizados son comunes a ambos problemas. La lista de puntos de interés (PdIs) que se ofrecerán a los turistas consta de 44 lugares distribuidos por el centro de Santa Cruz. Las preferencias son generales, basadas en opiniones previas que tienen los turistas obtenidas mediante encuestas por el ISTAC. Los puntos de interés se clasificán en categorías que nos ayudán a valorar los PdIs en función de las usadas por el ISTAC en sus encuestas.

También se tienen en cuenta las ventanas de tiempo, es decir el tiempo en el que están disponible o abiertos los lugares de interés. Por ello es necesario conocer sus horarios de apertura y de cierre, así como la duración de la visita. En los casos que no haya un horario de cierre o de apertura, como pasaría con los espacios abiertos como las calles, fijamos que abren a las 0:00 horas y cierran a las 24:00 horas.

En [19] se puede encontrar un archivo denominado DatosPdIs.pdf que dispone de una tabla con todos los datos relevantes para estos problemas.

Estos puntos de interés se representan en el siguiente mapa, donde se muestran los PdIs con un rombo verde:

Figura 3.1: Mapa de Santa Cruz con los puntos de interés marcados

Las rutas a visitar estan dentro de la ciudad, así todos los puntos de interés están conectados. De hecho, existen diversas formas de llegar de uno a otro: caminando, en tranvia, en guagua, en el autobús turístico, en taxi, bicicleta o patinete eléctrico. El problema podría usar una combinación de cualquiera de ellos, pero los problemas específicos a resolver en este trabajo están centrados en un único medio de transporte. Como en Santa Cruz suele hacer buen clima y no es una ciudad excesivamente grande, vamos limitar la búsqueda a rutas que se puedan hacer caminando. Para ello se construye una matriz de tiempos que indica cuanto se tarda de un punto a otro cualquiera y utilizamos los minutos como unidad de medida. En [19] está disponible un archivo denominado matriztiempo.pdf donde se puede visualizar la matriz de tiempo utilizada para resolver estos problemas.

Los resultados se obtienen variando ciertos parámetros iniciales. Se fijan una hora inicial de partida, que será las 9:00 horas y un punto inicial de donde saldrán todas las rutas, el punto 0, que en este caso es la Plaza España.

Una ves especificadas las instancias utilizadas con los datos disponibles y establecidos los parámetros de los problemas, la siguiente sección describirá la metodología utilizada para resolver los problemas planteados.

3.2. Experimentación

En esta sección explicamos la metodología utilizada, así como las pruebas de ejecución realizadas. Para resolver los problemas planteados vamos a utilizar una VND, que ya hemos explicado y analizado en el capítulo anterior.

3.2.1. Experimentación con el modelo OPTW

En este apartado, se va a realizar un experimento para determinar la estructura de entornos utilizada para resolver el modelo OPTW (diseño de una única ruta).

Para que la experimentación es necesario fijar dos parámetros iniciales, el tiempo máximo que se establece en 240 min., o lo que es equivalente, 4 horas, y la hora de inicio de la ruta, que es a las 9:00 de la mañana. El punto de partida también estará fijado, en el punto 0.

Lo primero es generar una solución inicial, en el método que se va a aplicar, esta solución se genera de forma aleatoria, por ello se trabajará con un número de iteraciones. Para estudiar qué número de iteraciones es más adecuado para generar la solución inicial se va realizar un experimento en el cual se generarán cinco soluciones para cada número de iteraciones m que se quiera estudiar y se compararán el valor de sus funciones objetivos. Los resultados de este experimento se reflejan en la tabla siguiente:

Soluciones	**Función objetivo**					**Tiempo medio**
	$m = 10$	$m = 40$	$m = 50$	$m = 60$	$m = 80$	**de ejecución**
Prueba 1:	59.5	113.5	91	116.5	87	0
Prueba 2:	79	97.5	137	147	68.5	0
Prueba 3:	92	87	105	138.5	97.5	0
Prueba 4:	89.5	142	122.5	92	110	0
Prueba 5:	76.5	88	122	119	109.5	0
Media	79.3	105.6	115.5	122.6	94.5	0

Tabla 3.1: Número de iteraciones en la función generadora de la solución inicial

En la tabla 3.1 se aprecia que con 60 iteraciones se obtiene de media una mayor puntuación en la función objetivo, y dado que el tiempo de ejecución de la función generadora es prácticamente nulo, a partir de ahora se fijará las iteraciones de la función generadora de soluciones en $m = 60$.

Es importante implementar una función que compruebe en cada paso si la solución es factible o no, es decir que cumpla con las restricciones que se han establecido. Esta función será indispensable cuando se ejecuten cambios en la solución actual.

Al utilizar una VND para resolver el problema, se debe tener en cuenta que es una variante de la VNS y que el tamaño del entorno de búsqueda debe variar. En este ejercicio particular que estamos resolviendo, el tamaño máximo de la estructura, que se denota como k_{max}, será $k_{max} = 3$.

Como se ha explicado en la última sección del capítulo 2, se van a utilizar tres tipos de búsquedas locales:

- **Intercambio interno**: comienza eligiendo el primer PdI que se visita, y se prueba a cambiar de posición de forma iterativa, después de cada cambio se comprueba que la solución nueva es factible y en caso afirmativo se compara el tiempo que se tarda en recorrer la nueva solución y se compara con la solución actual, y se escoge la solución que tarde menos tiempo en recorrerse. Una vez que hemos probado todos los puestos con el primer punto, pasamos al segundo y repetimos el proceso, y así hasta probarlo con todos los PdIs de la solución actual. Una vez que se ha realizado este movimiento con todos los PdIs de la solución se amplía el tamaño del entorno de búsqueda, y en vez de intercambiar un elemento por otro se intercambiaran de dos en dos, o de tres en tres, hasta alcanzar k_{max}.

- **Intercambio externo**: se elige el primer PdI de la lista de puntos, si pertenece a la solución actual, escogemos el siguiente, y si no pertenece lo intercambiamos por el punto que está en la posición 1, primero se comprueba que es factible y en caso afirmativo se comparan los valores de las funciones objetivo, se elegirá la solución con mayor valor en la función objetivo. Si la nueva solución es mejor, comenzamos desde el principio el movimiento, y si es mejor la solución actual, pasamos a la siguiente posición de la solución. Repetimos el proceso hasta pasar por todas las posiciones, y una vez recorrida toda la solución elegimos el siguiente PdI de la lista de puntos de interés. Una vez que se ha realizado el cambio con todos los PdIs disponibles, ampliamos el tamaño del entorno de búsqueda como en el movimiento anterior, y se realiza el cambio de dos en dos,

etc.

- **Adición**: se busca el primer PdI que no pertenece a la solución actual y se introduce en ella, primero en la primera posición, si esta nueva solución es factible entonces será la nueva solución actual y se vuelve a empezar el procedimiento, en caso contrario el elemento escogido se introduce en la siguiente posición, y así hasta recorrer la solución entera. Este proceso se repite con cada PdI que se ha quedado fuera de la solución actual, y una vez que se ha probado con todos, entonces se amplía el entorno de búsqueda como en los movimientos anteriores.

Para establecer la estructura de entornos adecuada, se va a experimentar con el orden de los movimientos. Para ello se utilizarán siempre las mismas 5 soluciones generadas con 60 iteraciones, y un tiempo máximo de 240 min.:

Pruebas	**Solución inicial**	**Función objetivo**	**Tiempo de ejecución**
1	[0 35 34 7 43 31 32 33 12 16 20 17]	114	0.0
2	[0 3 34 7 10 14 1 37 41 27 33 43 26]	116.5	0.0
3	[0 17 39 43 25 15 23 22 33 27 34]	98	0.0
4	[0 10 41 16 5 28 13 11 22 15 27]	89.5	0.0
5	[0 38 23 7 42 33 18 34 36]	80	0.0

Tabla 3.2: Soluciones iniciales

A continuación, aplicamos cada uno de los tres movimientos a las soluciones propuestas en la tabla 3.2 para comparar con qué movimiento obtenemos una mayor puntuación de la función objetivo. En la siguiente tabla se representa dicha comparación:

Movimientos	**Función objetivo**					**Tiempo medio**
	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5	**de ejecución**
Solución inicial:	114	116.5	98	89.5	80	0
Intercambio interno:	114	116.5	98	89.5	80	0,00385108
Adición:	134	146.5	118	119.5	100	0,00499959
Intercambio externo:	120	130	110	109	90	0,024472666

Tabla 3.3: Comparación de un movimiento

Como observamos en la tabla anterior, el movimiento de intercambio interno no afecta a la función objetivo, ya que se trabajan con los mismos puntos de interés y lo que optimiza este movimiento es el tiempo que se tarda en recorrer dichos puntos, respecto a los otros dos movimientos, se puede apreciar claramente como el de adición aumenta más la función objetivo que el intercambio externo en to-

das las pruebas realizadas. La última columna hace referencia al tiempo medio de ejecución de cada movimiento, como se puede apreciar el movimiento más rápido es el intercambio interno, seguido muy de cerca por la adición.

A continuación estudiamos que combinación de dos movimientos es más beneficiosa.

Movimientos	**Función objetivo**					**Tiempo medio de ejecución**
	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5	
Solución inicial:	114	116.5	98	89.5	80	0
Int. interno + adición:	191	165.5	128	139.5	129	0,011731005
Int. interno + externo:	120	129	110	89.5	90	0,043532753
Adición + int. interno:	134	146.5	118	119.5	100	0,014973974
Adición + int. externo:	140	160	130	140	110	0,039616823
Int. externo + interno:	120	130	110	109	90	0,028169203
Int. externo + adición:	162	177	166	175	175	0,028737879

Tabla 3.4: Combinaciones de dos movimiento

Como se puede apreciar en la tabla 3.4 en la mayoría de los casos, la mejor combinación de dos movimientos es hacer primero el intercambio externo y después la adición. Esto se cumple en todas las pruebas a excepción de en la prueba 1, que su mejor resultado se obtiene al hacer primero un intercambio interno y después una adición. Se puede apreciar que en ambos casos se ejecuta el movimiento de adición, que había resultado ser el mejor movimiento de la tabla 3.3. Para salir de dudas se compararán las combinaciones de los tres movimientos.

A partir de ahora, se utilizará la siguientes abreviaturas referente a los movimientos:

- II : Intercambio interno
- IE : Intercambio externo
- A : Adición

En la siguiente tabla se muestran las diferentes combinaciones de tres movimientos que se están estudiando y los valores de la función objetivo que se obtienen con cada una de ellas:

En la tabla anterior destacan dos combinaciones de los movimientos. La combinación “intercambio interno + intercambio externo + adición” ha obtenido los mejores resultados en 2 de las 5 pruebas, mientras que la otra combinación los ha obtenido en las restantes tres pruebas, además si se compara el tiempo medio de ejecución se aprecia que la combinación “intercambio externo + intercambio interno + adición” es más rápida.

Movimientos	**Función objetivo**					**Tiempo medio de ejecución**
	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5	
Solución inicial:	114	116.5	98	89.5	80	0
II + A + IE:	199	179	140	159	140	0,050202036
II + IE + A:	207	177	167	206	185	0,04355998
A + II + IE:	140	159	130	140	109	0,043018627
A + IE + II:	140	160	130	140	110	0,051764441
IE + II + A:	194	182	206	163	194	0,035887384
IE + A + II:	162	177	166	175	175	0,039819527

Tabla 3.5: Combinaciones de los tres movimiento

Antes de tomar una decisión entre estas dos combinaciones, se plantea otro experimento. Ya que el objetivo del problema es diseñar rutas de diversos tiempos de duración, podría ser esclarecedor comparar estas dos combinaciones según la duración de la ruta, para ello generaremos tres nuevas soluciones para cada tiempo máximo de ruta. Recordar, que se quieren diseñar rutas de 2, 3 y 4 horas. Las soluciones generadas en función del tiempo máximo de ruta aparecen en la siguiente tabla:

Tiempo máximo	**Pruebas**	**Soluciones**	**Función objetivo**
$T_{max} = 2$ horas	Prueba 2.1	[0 14 6 5 42 32 23 22 25]	89
	Prueba 2.2	[0 33 5 6 12 22 26 2 30 23]	97
	Prueba 2.3	[0 4 10 16 21 42 26]	63
$T_{max} = 3$ horas	Prueba 3.1	[0 22 29 10 35 12 7 3]	75
	Prueba 3.2	[0 34 20 39 11 36 19 42 14]	85
	Prueba 3.3	[0 33 25 1 2 18 13 28 3 16 26]	97
$T_{max} = 4$ horas	Prueba 4.1	[0 42 10 7 22 8 37 17 25 29]	93
	Prueba 4.2	[0 11 26 33 32 43 29 16 17 35 36 5 28 41 13 10]	146.5
	Prueba 4.3	[0 7 11 31 19 33 16 17 43 12]	91

Tabla 3.6: Soluciones iniciales para rutas de diferente tiempo máximo

En la tabla anterior no están los tiempos de ejecución ya que como se puede apreciar en el experimento anterior, el tiempo de ejecución de la función generadora de soluciones iniciales es muy pequeño y cercanos a cero.

A continuación, se expresan en una tabla el valor de la función objetivo de las mejores soluciones obtenidas con ambas combinaciones de movimientos en cada una de las soluciones propuestas en la tabla 3.6.

Movimientos	**Función objetivo**									**Tiempo medio de ejecución**		
	$T_{max} = 2$			$T_{max} = 3$			$T_{max} = 4$					
	2.1	2.2	2.3	3.1	3.2	3.3	4.1	4.2	4.3	$T_{max} = 2$	$T_{max} = 3$	$T_{max} = 4$
Solución inicial:	89	97	63	75	85	97	93	146.5	91	0	0	0
II + IE + A:	158	149	138	186	176	166	225	208	225	0,013	0,029	0,042
IE + II +A:	178	167	117	136	167	186	175	236	194	0,017	0,027	0,023

Tabla 3.7: Combinaciones de los tres movimiento

Como se puede apreciar en la tabla 3.7 ambas combinaciones de movimientos tienen mejores soluciones, "intercambio interno + externo +adición"genera mejores soluciones en las pruebas 2.3, 3.1, 3.2, 4.1 y 4.3 mientras que la otra combinación de movimientos las genera en el resto de pruebas, por lo que están bastante igualadas. Asique para tomar una decisión se tendrá en cuenta el tiempo medio de ejecución de cada combinación. Como se aprecia en la tabla anterior, cuando el tiempo máximo es menor ($T_{max} = 2$) la combinación más rápida es la primera, aunque existe poca diferencia con la segunda, pero a medida que aumenta el tiempo máximo, la combinación "intercambio externo +interno + adición"parece ser más rápida, y la diferencia va en aumento. Por lo tanto, escogeremos esta última combinación para resolver nuestro problema particular.

La estructura de la metaheurística para obtener la máxima puntuación seguiría el siguiente esquema:

1. Generamos una solución inicial
2. solucion-actual1 = intercambio externo (solucion inicial)
3. solucion-actual2 = intercambio interno (solucion-actual1)
4. Solucion-actual3 = adición (solucion-actual2)

Al estudiar las soluciones obtenidas sobre cartografía se aprecia que el orden de los puntos a visitar no es el más apropiado, por eso considero que es necesario añadir al final del esquema anterior el movimiento de intercambio interno para que ordene los puntos de una forma más satisfactoria, quedando entonces la siguiente estructura:

1. Generamos una solución inicial
2. solucion-actual1 = intercambio externo (solucion inicial)
3. solucion-actual2 = intercambio interno (solucion-actual1)
4. solucion-actual3 = adición (solucion-actual2)
5. Solucion-final = intercambio interno (solucion-actual3)

Aplicando esta estructura se obtienen las siguientes soluciones:

- Para una ruta con tiempo máximo de dos horas ($T_{max} = 120$):

Proceso	**Solución actual**	**Valor**
Solución generada:	[0 7 10 12 36 20 2]	58
IE:	[0 1 10 12 36 20 2]	60
IE + II:	[0 36 1 10 12 20 2]	60
IE + II +A:	[0 25 13 11 5 4 3 36 1 10 12 20 2]	116
Solución final:	[0 11 5 25 13 36 4 3 1 10 12 20 2]	116

Tabla 3.8: Soluciones encontradas aplicando un VND

La última columna hace referencia al valor de la función objetivo. El tiempo de duración de la última solución actual es de 119 minutos y el tiempo de ejecución del programa ha sido 0.00841279220581055.

- Para una ruta con el tiempo máximo de tres horas ($T_{max} = 180$):

Proceso	**Solución actual**	**Valor**
Solución generada:	[0 3 23 7 38 33 14]	54
IE:	[0 2 10 11 5 6 14]	60
IE + II:	[0 11 2 10 5 14 6]	60
IE + II + A:	[0 25 26 23 22 20 17 36 16 13 12 4 3 1 11 2 10 5 14 6]	185
Solución final:	[0 11 4 2 22 20 17 3 1 16 36 13 25 10 12 26 23 5 14 6]	185

Tabla 3.9: Soluciones encontradas aplicando un VND

El tiempo de duración de la última solución actual es de 179 min. y el tiempo de ejecución del programa es 0,014923810958862305.

- Y por último, una ruta cuyo tiempo máximo es de cuatro horas ($T_{max} = 240$):

Proceso	**Solución actual**	**Valor**
Solución generada:	[0 26 20 8 29 43 31 1 5 42]	84
IE:	[0 10 11 14 29 16 17 1 5 42]	90
IE + II:	[0 29 17 1 10 16 11 5 14 42]	90
IE + II + A:	[0 25 26 23 22 20 13 12 6 4 3 2 29 17 1 10 16 11 5 14 42]	195
Solución final:	[0 11 29 25 20 23 26 16 13 3 4 2 22 17 1 10 12 5 14 6 42]	195

Tabla 3.10: Soluciones encontradas aplicando un VND

El tiempo de duración de la última solución obtenida es de 238 min. y el tiempo de ejecución del programa es 0,024193525314331055.

3.2.2. Experimentación del modelo TOPTW

A continuación, se tratará de establecer una estructura de entorno para el problema del diseño de varias rutas, el modelo TOPTW.

En este caso en particular, vamos a diseñar tres rutas diferentes, con tiempos máximos diferentes:

$$T_{max} = [T_1, T_2, T_3] = [240, 180, 120]$$

donde T_i es el tiempo máximo en la ruta i en minutos.

Este modelo presenta algunas diferencias respecto al OPTW. Comenzando con la función generadora de soluciones, ahora esa función que sigue determinada por el número de iteraciones, se repite tres veces, es decir, le pedimos que nos genere una solución factible que empiece en cero, y que no supere el tiempo máximo T_1, una vez creada la primera ruta, añadimos un cero a la solución y le pedimos que nos cree otra solución a partir de ese cero y que no sobrepase T_2, y así con la tercera también. Sacando provecho de la experimentación anterior, utilizaremos 60 iteraciones para generar cada una de las rutas de la solución. De este modo, obtendremos tres soluciones separadas por ceros en el mismo vector.

Los movimientos para este problema son los siguientes:

- **Intercambio interno dentro de cada ruta**: Este intercambio es muy similar que el visto en el modelo anterior. Simplemente es aplicar el intercambio interno a cada ruta por separado, intercambiando solo los PdIs pertenecientes a la misma ruta.

- **La adicion**: No supone alguna diferencia con la adición anterior, simplemente se aplica a las tres rutas simultáneamente y se debe tener presente en todo momento que cada ruta tiene un tiempo máximo distinto que se debe respetar.

- **Intercambio externo**: se aplica de forma similar que el intercambio externo del modelo OP salvo, que en esta ocasión se debe tener especial consideración con los ceros de la ruta, es decir, se debe añadir el condicionante de que si el elemento de la solución que queremos intercambiar por uno externo es el punto

cero, entonces se debe dejar la solución sin realizar el intercambio y pasar al siguiente elemento de la solución.

- **Intercambio interno entre rutas**: Este movimiento es una nueva versión del movimiento de intercambio interno del modelo OP, en este caso, a diferencia de en el intercambio interno dentro de cada ruta, en este movimiento se intercambian puntos que pertenecen a la solución sin importar a que ruta pertenezcan. Es importante recalcar que con este movimiento el valor de la función objetivo de la solución no va a variar, si puede variar el valor de la función objetivo de cada ruta que forman la solución, pero en esta experimentación no se tendrá en cuenta el valor individual de cada ruta sino el de toda la solución.

En este caso, al tener cuatro movimientos las combinaciones son mucho más numerosas que en el caso anterior con tan solo tres movimientos, por ello, vamos a aprovechar los resultados obtenidos en la experimentación anterior, y partiendo de la estructura anterior:

" intercambio externo + intercambio interno + adición "

donde el intercambio interno hace referencia al intercambio interno dentro de la misma ruta, se va a estudiar en que posición es mejor introducir el movimiento de intercambio interno entre rutas manteniendo el orden de los otros tres movimientos.

Para este experimento vamos a necesitar de cinco pruebas como en el experimento anterior, y así comparar los resultados obtenidos en cada una de ellas. Por lo que se comienza generando 5 soluciones iniciales, recordemos que cada una de estas soluciones iniciales son tres rutas divididas por cero.

Pruebas	**Solución inicial**	**F. objetivo**
1	[0 16 8 30 2 18 26 29 1 37 10 20 0 5 38 33 17 42 27 14 6 0 39 32 4 34]	210
2	[0 37 20 33 18 5 25 36 12 42 13 10 1 29 21 0 31 38 4 32 41 14 0 23 43 16 2 28 30 3]	241.5
3	[0 40 38 42 29 15 32 3 1 0 41 20 33 23 35 21 14 25 0 31 8 37 2 22]	183
4	[0 30 6 3 14 43 16 35 22 21 5 13 29 1 0 39 33 41 23 32 36 31 2 25 0 11 42 20 40 28 17]	257.5
5	[0 18 20 15 1 31 40 12 10 0 39 32 2 17 21 38 26 23 0 41 7 35]	163.5

Tabla 3.11: Soluciones iniciales TOP

El tiempo de ejecución es tan ínfimo (muy cercano a 0) que no es de interés.

Ahora que ya se tienen las 5 soluciones generadas, se estudiará el valor objetivo de cada una, al aplicarle los cuatro movimientos en diferente orden, estos resultados se expresarán en la siguiente tabla. Por motivos de espacio, vamos a utilizar la siguiente abreviatura:

- IIint : Intercambio interno dentro de la misma ruta.
- A : Adición
- IE : intercambio externo
- IIext : Intercambio interno entre diferentes rutas.

Movimientos	**Función objetivo**					**Tiempo medio de ejecución**
	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5	
Solución inicial:	210	241.5	183	257.5	163.5	0
IIext + IE + IIint + A:	301	324	282	332.5	305	0,180161953
IE + IIext + IIint + A :	319	319	305	324	310	0,168920135
IE + IIint + IIext + A :	319	319.5	310	324	310	0,169985437
IE + IIint + A + IIext:	319	318	296	319	307	0,304596043

Tabla 3.12: Combinaciones de los cuatro movimiento

Tal y como se puede apreciar en la tabla 3.12, el primer orden de movimientos " IIext + IE + IIint + A ", puede ser el mejor en algunos casos y el peor en otros, y en algunas ocasiones como en el la prueba 1 el resto de los órdenes no presentan ninguna diferencia en cuanto al valor de la función objetivo de la solución, por lo tanto se calculará la media de los valores de las funciones objetivo para cada orden de movimientos:

Movimientos	**Función objetivo**
IIext + IE + IIint + A:	308.9
IE + IIext + IIint + A :	315.4
IE + IIint + IIext + A :	316.5
IE + IIint + A + IIext:	311.8

Tabla 3.13: Media de los valores de la función objetivo

En la tabla 3.13 se aprecia que, de media, el valor de la función objetivo es mayor cuando el orden de los movimientos es " IE + IIint + IIext + A ". Pero al igual que en el caso OPTW, es conveniente ordenar los PdIs que conforman la última solución para que así el orden de la ruta sea algo más favorable para el turista. Por lo que la estructura de entornos que se va a utilizar será:

1. Generamos una solución inicial
2. solucion-actual1 = intercambio externo (solucion inicial)
3. solucion-actual2 = intercambio interno dentro de la misma ruta (solucion-actual1)
4. solucion-actual3 = intercambio interno entre diferentes rutas (solucion-actual2)
5. solucion-actual4 = adición (solucion-actual3)
6. Solucion-final = intercambio interno dentro de la misma ruta (solucion-actual4)

Asique ese orden de movimientos determinará la estructura del entorno de búsqueda.

Siguiendo este orden de movimientos se ha encontrado la siguiente solución, con el vector de tiempos máximos definido anteriormente:

Proceso	**Solución actual**	**Valor**
Solución generada:	[0 4 32 28 2 5 14 30 33 29 40 39 0 1 25 18 8 12 0 16 6 17 35 31]	188
IE:	[0 36 10 11 2 5 42 43 14 29 40 39 0 25 32 1 34 12 0 16 6 22 23 31]	208
IE + IIint :	[0 11 5 14 42 2 10 43 36 29 40 39 0 25 12 1 34 32 0 16 6 22 23 31]	208
IE + IIint + IIext:	[0 40 39 0 25 10 12 1 0 11 29 36 16 43 32 5 14 6 42 23 22 34 2 31]	208
IE + IIint + IIext + A:	[0 13 8 7 4 3 40 39 0 33 30 26 20 21 18 17 25 10 12 1 0 11 29 36 16 43 32 5 14 6 42 23 22 34 2 31]	305
Solución final:	[0 13 3 4 7 8 40 39 0 33 18 26 21 20 30 17 1 10 12 25 0 11 29 36 16 43 32 5 14 6 42 23 22 34 2 31]	305

Tabla 3.14: Una solución del modelo TOPTW

Los tiempos de duración de las tres rutas obtenidas son 238, 179 y 119 respectivamente. El tiempo de ejecución del programa es de 0.16063666343688965.
*

* Los programas en python utilizados para resolver estos problemas están disponibles en [19] dentro de una carpeta denominada programas.

4

Conclusiones

De este trabajo se pueden extraer, entre otras, las conclusiones siguientes:

- En la literatura se pueden encontrar muchos modelos para resolver los problemas de diseño de itinerarios turísticos. En este trabajo hemos modelado dos problemas para encontrar rutas a pie por Santa Cruz de Tenerife, teniendo en cuenta limitaciones de duración de la ruta y ventanas de tiempo en los lugares de interés. Para ello hemos utilizado los modelos OPTW y TOPTW variantes de la familia de los problemas de Orientación. Estos dos modelos simples que nos facilitan formular los problemas como problemas de programación lineal e identificar objetivos, restricciones, parámetros y problemas para resolverlos.

  Para resolver estos problemas hemos optado por métodos de resolución aproximados y el uso del lenguaje de programación Python para implementarlos. Específicamente hemos utilizado la metaheurística VND para encontrar soluciones. Como se observa en la descripción y experimentación realizada, con el uso unos pocos y sencillos movimientos de entornos se ha conseguido soluciones en un tiempo bastante corto. Así hemos podido demostrar que este tipo de aproximaciones facilitan encontrar soluciones a este tipo de problemas complejos. No siempre es posible encontrar soluciones exactas en estos problemas con muchos puntos de interés que visitar y con las restricciones propuestas.

- Respecto a los resultados de aprendizaje con en este proyecto, he ampliado los conocimientos que poseía sobre optimización combinatoria. He aprendido a identificar, especificar, modelar y formular problemas de optimización del mundo real. Existen muchos modelos diferentes que pueden resolver los problemas según sus objetivos y las restricciones tratadas. También he aprendido conceptos totalmente nuevos, como los métodos aproximados metaheurísticos para resolver problemas NP. Por último, he sido capaz de implementar en Python

programas y funciones con el método seleccionado.

- Las soluciones obtenidas son soluciones factibles que satisfacen todas las restricciones, pero para aquellos que conocen Santa Cruz, es fácil ver que se pueden mejorar, desde el punto de vista práctico. Para ello sería necesario profundizar y ajustar los modelos, incorporando nuevas restricciones y criterios en los objetivos.

  En futuros trabajos se pueden modelar estos problemas con otras o más restricciones, también se puede resolver el problema exacto y utilizar otras metaheurísticas y comparar los resultados obtenidos en cada una de ellas, evaluando los mejores métodos y soluciones para este problema.

  Puede ser de interés en el futuro abordar problemas en el cual se puede tener en cuenta la selección de medios de transporte y la dependencia del tiempo. Un problema que en su momento nos planteamos pero con las limitaciones de tiempo disponible para realizar el TFG, no pudimos abordar.

Bibliografía

[1] Abraham Duarte. *Optimización heurística: Búsqueda de Vecindad Variable* Grafo Research. Universidad Rey Juan Carlos. C. Tulipán, s/n, 28933 Móstoles, Madrid https://grafo.etsii.urjc.es/

[2] Alancay, N. Villagra, S. Villagra, A. *Algoritmos metaheurísticos trayectoriales para optimizar problemas combinatorios* UNPA- UACO (Universidad Nacional de la Patagonia Austral - Unidad Académica) Departamento de Ciencias Exactas y Naturales LabTEm- Laboratorio de Tecnologías Emergentes Caleta Olivia, 2015

[3] Ali Azizi, Farid Karimipour & Ali Esmaeily (2017) . *Time-dependent, activity-based itinerary personal tour planning in multimodal transportation networks.* Annals of GIS, 23:1, 27-39. Disponible en: https://doi.org/10.1080/19475683.2016.1276100

[4] Damianos Gavalas, Grammati Pantziou, Vlasios Kasapakis, Nikolaos Vathis, Charalampos Konstantopoulos y Christos Zaroliagis. *A Personalized Multimodal Tourist Tour Planner* November 25 - 28 2014, Melbourne, VIC, Australia

[5] Damianos Gavalas, Vlasios Kasapakis, Charalampos Konstantopoulos, Grammati Pantziou, Nikolaos Vathisy Christos Zaroliagis. *The eCOMPASS multimodal tourist tour planner* Expert Systems with Applications. Página de la revista: www.elsevier.com/locate/eswa

[6] Francisco Herrera *Introducción a los Algoritmos Metaheurísticos* Grupo de Investigación "Soft Computing and Intelligent Information Systems" Dpto Ciencias de la Computación e I A Dpto. Ciencias de la Computación e I.A. Universidad de Granada, 18071 – ESPAÑA. http://sci2s.ugr.es

[7] José Andrés Moreno Pérez. *Metaheurísticas de Búsqueda por Entornos Variables, VNS.* Disponible en: https://www.youtube.com/watch?v=Ztrtfl5Y4MA

[8] Jorge Brito Dávila *Optimización de rutas turísticas* Trabajo de fin de grado, grado en Matemáticas, Universidad de la Laguna.

[9] Julian Dibbelt, Charalampos Konstantopoulos, Dorothea Wagner, Damianos Gavalas, Spyros Kontogiannis, Christos Zaroliagis, Vlasios Kasapakis y Grammati Pantziou. *Multimodal Route and Tour Planning in Urban Environments* Conference Paper · July 2017.

[10] Julio Brito Santana. *Sistemas de Recomendación Gestión del Conocimiento en las Organizaciones.* Diciembre 2019

[11] Krzysztof Ostrowski. *An effective metaheuristic for tourist trip planning in public transport networks.* Faculty of Computer Science, Białystok University of Technology, Wiejska 45A, 15-001 Białystok, Poland.

[12] Michel Gendreau · Jean-Yves Potvin (Editors) International series in Operations Research & Management Science *Handbook of Metaheuristics* Tercera edición (2019) Springer.
Más información en :https://www.springer.com/series/6161

[13] Pierre Hansen, Nenad Mladenovia y Jose A. Moreno Pérez. *Búsqueda de Entorno Variable* (2003 - 11 - 05)

[14] Pierre Hansen, Nenad Mladenovic, José Andrés Moreno Pérez. Variable Neighbourhood Search. *Revista Iberoamericana de Inteligencia Artificial. No.19 (2003),pp. 77-92.*
Disponible en: http://www.aepia.org/revista.

[15] Yalan Zhou, Chen Li y Yanyue Li. *A Simulated Annealing for Multimodal Team Orienteering Problem with Time Windows.* Springer Nature Singapore Pte Ltd. 2018 J. Qiao et al. (Eds.). Disponible en: https://doi.org/10.1007/978-981-13-2829-9_3

[16] Wikipedia. *Problema del viajante.* Disponible en: https://es.wikipedia.org/wiki/Problema_del_viajante

[17] Wikipedia. *Problema de rutas de vehículos*.Disponible en: https://es.wikipedia.org/wiki/Problema_de_rutas_de_veh%C3%ADculos

[18] Wikipedia. *Problema de enrutamiento de vehículos*.Disponible en: https://es.wikipedia.org/wiki/Problema_de_enrutamiento_de_veh%C3%ADculo

[19] En la siguiente dirección se encuentran algunas tablas relevantes en el capítulo 3 y el programa en python que se ha utilizado para resolver los problemas planteados. https://drive.google.com/drive/folders/1IIFxJF0JpPiTKCI3msIGeSifArhnhBON?hl=es