Informe Proyecto VRP-wit-RMayor

Marié del Valle C311
Roxana Peña C312
Josué Rodríguez C312

Problema

El problema planteado es un problema de asignación de vehículos y rutas óptimas para una empresa de transporte con una cierta cantidad de vehículos y clientes. Los clientes tienen un conjunto fijo de paradas que deben ser visitadas por los vehículos de la empresa. La cantidad de vehículos no puede ser menor que la cantidad de clientes.La empresa debe asignar vehículos a cada cliente y planificar las rutas de los vehículos de forma óptima para minimizar los costos de transporte. El objetivo es conocer el costo total del recorrido de cada vehículo durante un número específico de días.

Diseño

Agentes

En una simulación, un agente es un objeto o un componente que representa una entidad con capacidad de actuar y tomar decisiones dentro del sistema simulado. Los agentes pueden interactuar entre sí y con el medio para lograr sus objetivos. Cada agente tiene un conjunto de reglas o comportamientos programados que determinan cómo actúa y cómo toma decisiones. En el problema planteado, los agentes serían los vehículos, la empresa de transporte y las autoridades.

Vehículo

Un agente vehículo es un objeto que representa un vehículo en la simulación. Tiene varias propiedades, como un identificador único, una capacidad de carga y un total de kilómetros que puede recorrer. La compañía asigna un cliente específico al vehículo y también las paradas que ese cliente necesita que se visiten. Entre las posibles acciones del vehículo están:

- Moverse a la siguiente parada en la ruta asignada.
- Recoger o entregar carga en la parada actual.
- Reportar su posición actual.

- Cambiar la ruta si es necesario.
- Informar sobre el estado del vehiculo.

Compañía

El agente compañía es un objeto que representa la empresa de transporte en la simulación. Tiene varias propiedades, como un deposito, una lista de vehículos, una lista de clientes y un presupuesto. Entre las acciones posibles de la compañía están:

- Asignar a cada vehículo su ruta óptima de acuerdo al cliente y las paradas asignadas previamente a ese veículo.
- Comprar un nuevo vehículo cuando sea necesario.
- Enviar a mantenimiento a los vehículos que lo necesiten y buscar reemplazo en lo posible.
- Pagar todos los gastos relacionados con los vehículos, como combustible y multas.
- Analizar y optimizar las rutas y los costos de transporte.

Autoridades

El agente autoridades es un objeto que representa las autoridades relevantes en la simulación, como las autoridades de tránsito y la policía. Interactúan con los vehículos y pueden tomar acciones como:

- Poner multas a los vehículos que incumplen las reglas de tránsito.
- Desviar a los vehículos de su ruta en caso de obstrucciones en el camino.

Medio

En una simulación, el medio es el entorno o el ambiente en el que se lleva a cabo la simulación. En el contexto del problema planteado, el medio podría ser el sistema de carreteras y ciudades en el que se mueven los vehículos de la empresa de transporte. Puede incluir aspectos como el tráfico, las condiciones climáticas, las regulaciones de tránsito, la infraestructura, entre otros. El medio puede ser modelado y simulado de forma detallada o de forma simplificada, en función de los objetivos de la simulación y de los recursos computacionales disponibles.

En el medio planteado anteriormente, los semáforos son un componente importante del sistema de carreteras y ciudades en el que se mueven los vehículos de la empresa de transporte. Los semáforos son dispositivos que controlan el flujo de tráfico en las intersecciones de las calles.

Los semáforos tienen luces de diferentes colores, como rojo, amarillo y verde, que indican a los conductores cuándo deben detenerse, reducir la velocidad o

continuar. En el medio planteado anteriormente, los semáforos son un aspecto importante a tener en cuenta en la planificación de las rutas de los vehículos y en la simulación del tráfico. Los vehículos deben detenerse en los semáforos en rojo y esperar a que cambie a verde antes de continuar. Los vehículos deciden si pasan o no cuando la luz del semáforo esté en amarrilla. Esto puede afectar el tiempo de viaje y el costo total del recorrido.

Implementación

La implementación computacional del problema fue abordada utilizando varios enfoques, como la programación de agentes, la inteligencia artificial y la optimización de rutas.

Se utilizó un sistema de programación de agentes, donde cada agente (vehículo, empresa y autoridades) fue programado con un conjunto de reglas y comportamientos que le permiten interactuar con el medio y con los demás agentes. Los agentes vehículos se mueven de acuerdo a la ruta asignada y toman decisiones basadas en la información del medio y en las comunicaciones con la empresa y las autoridades. La empresa asigna las rutas a los vehículos y toma decisiones sobre la adquisición de nuevos vehículos y el mantenimiento de los existentes. Por último, las autoridades interactúan con los vehículos para aplicar multas y desviar el tráfico en caso de obstrucciones.

También, se utilizaron técnicas de inteligencia artificial, como búsqueda, metaheurísticas y planificación.

Clases

MapNode

La clase `MapNode` representa un nodo en el grafo que es el mapa sobre el cual los vehículos se transportan. Esta clase tiene varias propiedades que se utilizan para representar el estado del mapa en un punto específico. Algunas de las propiedades más importantes son:

- `position`: esta propiedad indica la posición geográfica del nodo en el mapa.
- `people`: esta propiedad indica la cantidad de personas que se encuentran en ese punto en el mapa.
- `authority`: esta propiedad indica si hay autoridades presentes en ese punto. Contiene un objeto Authority que con información adicional sobre las autoridades presentes.
- `semaphore`: esta propiedad indica si hay un semáforo en ese punto en el mapa. Contiene un objeto Semaphore que con información adicional sobre el semáforo.

Vehicle

La clase `Vehicle` representa al agente vehículo en la simulación. Esta clase tiene varias propiedades y métodos que se utilizan para simular el comportamiento del vehículo y su interacción con el entorno. Algunas de las propiedades y métodos más importantes son:

- `id`: esta propiedad es el identificador único del vehículo.
- `capacity`: esta propiedad indica la capacidad del vehículo, es decir, la cantidad máxima de personas que puede transportar.
- `km_traveled`: esta propiedad indica la cantidad total de kilómetros que el vehículo ha recorrido.
- `route`: esta propiedad indica la ruta asignada al vehículo.
- `people_on_board`: esta propiedad indica la cantidad de personas que hay en el vehículo.
- `move()`: este método se utiliza para mover el vehículo a través del mapa, guiándose por la ruta asignada.
- `load()`: este método se utiliza para cargar a las personas en la posición actual en que se encuentra el vehículo.
- `unload()`: este método se utiliza para descargar a las personas en la posición actual en que se encuentra el vehículo.
- `at_semaphore()`: este método se utiliza cuando hay semáforos en el punto actual y para tomar decisiones sobre cómo evitarlas o adaptarse a las restricciones.
- `at_authority()`: este método se utiliza cuando hay autoridades presentes en el punto actual y para tomar decisiones sobre cómo evitarlas o adaptarse a las restricciones.
- `broken()`: este método se utiliza cuando el vehículo se rompe en medio de su ruta.
- `plan()`: este método se utiliza para hallar la próxima acción del vehículo en el punto actual.

Company

La clase `Company` representa a la empresa de transporte en la simulación. Algunas de las propiedades y métodos más importantes son:

- `vehicles`: esta propiedad es una lista de objetos `Vehicle`, que representan los vehículos de la empresa.

- `clients`: esta propiedad es un diccionario con los nombres de los clientes de la empresa y la lista de paradas de cada uno.
- `budget`: esta propiedad indica el presupuesto actual de la empresa.
- `assign()`: este método se utiliza para obterner asignación vehículos a los clientes y rutas a los vehículos.
- `start_route()`: este método se utiliza para asignar rutas a los vehículos.
- `pay_taxes()`: este método se utiliza para pagar los gastos de los vehículos, incluyendo el combustible, las multas y el mantenimiento.
- `buy_vehicle()`: este método se utiliza para comprar un vehículo.
- `check_vehicle()`: este método se utiliza para verificar el estado de un vehículo.
- `check_maintenance()`: este método se utiliza para conocer quq́ vehículos están en mantenimiento y reasignar, según las necesidades, a los que salgan.
- `find_replacement()`: este método se utiliza para hallar sustituto a vehículo en mantenimiento.
- `plan()`: este método se utiliza para hallar el plan de acción de la compañía por cada vehículo que posee.

Authority

La clase `Authority` representa al agente autoridad en la simulación. Algunas de las propiedades y métodos más importantes son:

- `probability`: esta propiedad indica la probabilidad de la autoridad de parar el vehículo y poner multa.
- `stop_vehicle()`: este método se utiliza cuando la autoridad para el vehículo.
- `change_place()`: este método se utiliza cuando la autoridad cambia su posición en el mapa.

Semaphore

La clase `Semaphore` representa los semáforos en el medio de la simulación. Esta clase tiene varias propiedades y métodos que se utilizan para simular el comportamiento de los semáforos y su interacción con los vehículos. Algunas de las propiedades y métodos más importantes son:

- `position`: esta propiedad indica la posición en el mapa donde se encuentra el semáforo.

- `state`: esta propiedad indica el color actual del semáforo.
- `color_range`: esta propiedad indica la duración de cada color del semáforo.
- `update_color()` : dado el tiempo global y la duración de cada color del semáfore, actualiza el color del mismo.

Inteligencia Artificial

Búsqueda

La búsqueda es una técnica de inteligencia artificial que se utiliza para encontrar soluciones óptimas en problemas de navegación en espacios de estados. Se utilizó junto al algoritmo de planificación para encontrar la siguiente acción de la compañía, dado un estado y un dominio. El algoritmo de búsqueda utilizado fue el algoritmo de costo uniforme.

Recocido Simulado

El enfoque de recocido simulado (Simulated Annealing, SA) es un algoritmo de optimización que se utiliza para encontrar soluciones óptimas en problemas de optimización complejos. Se inspira en el proceso de enfriamiento de los metales al ser forjados.

En el problema, la compañía de transporte tiene que asignar vehículos a los clientes y paradas a los vehículos. El proceso de asignación de vehículos y paradas es un problema de optimización complejo debido a las restricciones y las diferentes variables involucradas, como la capacidad de carga de los vehículos y la cantidad de personas en cada parada.

El algoritmo de recocido simulado se utiliza para encontrar la asignación óptima de vehículos y paradas. El algoritmo comienza con una solución inicial que es una asignación aleatoria y luego aplica un proceso iterativo de mejora de la solución. En cada iteración, el algoritmo genera una nueva solución modificando ligeramente la solución actual. La nueva solución se evalúa y se decide si se acepta o no en función de una función de probabilidad que depende de la diferencia entre la solución actual y la nueva solución y de un parámetro de temperatura que se va reduciendo a medida que el proceso avanza.

El proceso continúa hasta que se alcanza un criterio de parada, que en este caso es un número fijo de iteraciones.

Colonia de Hormigas

La metaheurística colonia de hormigas (ACO, por sus siglas en inglés) es un algoritmo de optimización que se utiliza para encontrar soluciones óptimas en problemas de optimización de rutas. Se basa en el comportamiento de las hormigas al buscar comida.

En el problema, la compañía de transporte tiene que planificar las rutas de los vehículos de forma óptima para minimizar los costos de transporte. El algoritmo de colonia de hormigas se utiliza para encontrar la ruta óptima para

cada vehículo. El algoritmo comienza con una solución inicial y luego aplica un proceso iterativo de mejora de la solución.

En cada iteración, el algoritmo simula el comportamiento de las hormigas para recolectar comida. Cada hormiga se mueve a través de las diferentes paradas y decide qué parada visitar en función de una función de probabilidad que depende de la cantidad de feromona depositada en cada parada y de la distancia entre las paradas. La feromona es una sustancia que las hormigas depositan en el camino para indicar a las demás hormigas qué ruta es más prometedora. Cuanto más feromona hay en un camino, más probable es que las hormigas elijan esa ruta.

A medida que las hormigas recorren las paradas, depositan feromona en el camino que han seguido. Esto aumenta la probabilidad de que las hormigas futuras elijan esa ruta. Al final de cada iteración, el algoritmo actualiza la feromona en las paradas, evapora una parte de la feromona para evitar que se acumule en una sola ruta y añade nueva feromona para fomentar la exploración de nuevas rutas.

Todos los vehículos tendrán en como primera parada el depósito de la compañía y como última parada el depósito del cliente al que fueron asignados. El criterio de parada es un número fijo de iteraciones.

Planificación

La planificación de inteligencia artificial (AI Planning) es una técnica que se utiliza para encontrar soluciones óptimas en problemas de planificación en los que existen varias acciones posibles y restricciones. Puede ser utilizado para asistir en la toma de decisiones de la empresa y optimizar las acciones que realiza.

Esta herramienta de utilizó en la compañía de transporte debido a que las acciones de la compañía son estáticas y consisten en tareas repetitivas como asignar rutas a los vehículos, pagar multas, combustible y mantenimiento, y verificar el estado de los vehículos. Con esto se logra automatizar y optimizar estas tareas mediante la generación de planes de acción que consideran las restricciones y limitaciones existentes.

DSL

Un lenguaje de dominio específico (DSL, por sus siglas en inglés) es un lenguaje de programación diseñado para resolver problemas específicos de un dominio particular. En el caso del Problema de Enrutamiento de Vehículos, se puede crear un DSL para codificar los problemas y facilita la tarea de resolverlos utilizando algoritmos de optimización. El lenguaje de dominio específico (DSL) creado para codificar problemas de enrutamiento de vehículos se puede conectar a la simulación para evaluar las soluciones propuestas y determinar cuál es la más adecuada. Esto permite que los problemas sean resueltos de manera más eficiente y permite predecir el comportamiento del sistema en diferentes escenarios.

Este lenguaje esta desarrollado en python, con una gramatica LALR. Para el desarrollo del compilador se usó PLY, que es una implementación de Python pura del constructor de compilación lex/yacc. Incluye soporte al parser LALR(1) así como herramientas para el análisis léxico de validación de entrada y para el reporte de errores. El análisis sintáctico se divide en 2 fases: en una se realiza el análisis léxico, con la construcción de un lexer, y en la otra se realiza el proceso de parsing, definiendo la gramática e implementando un parser para la construcción del Árbol de Sintaxis Abstracta (AST). El programa fuente se procesa de izquierda a derecha y se agrupan en componentes léxicos (tokens) que son secuencias de caracteres que tienen un significado. Todos los espacios en blanco, comentarios y demás información innecesaria se elimina del programa fuente. El lexer, por lo tanto, convierte una secuencia de caracteres (strings) en una secuencia de tokens.

El parser también se implementó mediante PLY, especificando la gramática y las acciones para cada producción. Para cada regla gramatical hay una función cuyo nombre empieza con p_. El docstring de la función contiene la forma de la producción, escrita en EBNF. PLY usa los dos puntos (:) para separar la parte izquierda y la derecha de la producción gramatical. El símbolo del lado izquierdo de la primera función es considerado el símbolo inicial. El cuerpo de esa función contiene código que realiza la acción de esa producción. En cada producción se construye un nodo del árbol de sintaxis abstracta.

El procesador de parser de PLY procesa la gramática y genera un parser que usa el algoritmo de shift-reduce LALR(1), que es uno de los más usados en la actualidad. Aunque LALR(1) no puede manejar todas las gramáticas libres de contexto, la gramática usada fue refactorizada para ser procesada por LALR(1) sin errores.

Para realizar los recorridos en el árbol de derivación se hace uso del patrón visitor. Este patrón nos permite abstraer el concepto de procesamiento de un nodo. Cada elemento del nodo se procesa y se envia a la simulacion para ejecutarla.

Se provee al ususario una simple interfaz para escribir el codigo y ver el resultado de la simulación.

Recomendaciones

En la simulación de una compañía de transporte, es esencial probar el modelo sobre un mapa real para obtener una mayor precisión en las rutas y los tiempos de viaje. Utilizar un mapa real del área en la que operará la empresa permitirá una mejor representación de la realidad y una mayor confiabilidad en los resultados obtenidos.

Además, es importante tener en cuenta el sentido de las calles en el mapa al generar las rutas. El sentido de las calles puede afectar significativamente los tiempos de viaje y, por lo tanto, debe ser considerado en la simulación.

Otro factor importante a tener en cuenta es la posibilidad de que la empresa pierda clientes o adquiera nuevos clientes. Es importante ajustar la simulación

en consecuencia para reflejar estos cambios en el negocio.