Trabajo Fin de Grado
Grado en Ingeniería de Tecnologías Industriales

Aplicación de multiagentes en Python para optimización de rutas.

Autor: Sergio Fuentes Moreno
Tutor: José Miguel León Blanco

Dpto. de Organización Industrial y Gestión de Empresas I
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Trabajo Fin de Grado
Grado en Ingeniería de Tecnologías Industriales

Aplicación de multiagentes en Python para optimización de rutas.

Autor:
Sergio Fuentes Moreno

Tutor:
José Miguel León Blanco
Profesor Contratado Doctor

Dpto. de Organización Industrial y Gestión de Empresas I
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2021

Trabajo Fin de Grado: Aplicación de multiagentes en Python para optimización de rutas.

Autor: Sergio Fuentes Moreno

Tutor: José Miguel León Blanco

El tribunal nombrado para juzgar el Trabajo arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2021

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

Agradecido por estos años vividos en la escuela donde he conocido a grandes amigos. Amigos que me han acompañado a lo largo de la carrera, compartiendo trabajos, aprobados y suspensos, personas que me han acompañado en momentos de alegría y de tristeza.

Agredecer también a aquellos profesores que me han ayudado, tanto de la escuela como de la academia, profesores que sin ellos no estaría aquí redactando un trabajo de fin de grado. Sobre todo, a Javi de informática, gracias a él entendí la programación y me gustó hasta el punto de que, aquí estoy, realizando un trabajo que lleva por detrás un programa escrito por mí.

También quería agradecer a mi tutor José Miguel León Blanco por ayudarme a realizar este trabajo, por darme siempre ánimos, prestarme la ayuda necesaria y darme la oportunidad de realizar un trabajo que realmente me gusta. También darle las gracias por ayudarme en otras asignaturas en las que hemos coincidido a lo largo de la carrera.

Por último, agredecer sobre todo a mis padres Antonio Fuentes e Isabel Moreno, que me han animado a lo largo de la carrera. También agradecer a mi padre por intentar ayudarme con el trabajo, explicándome términos sobre la logística, el e-commerce y hablarme del momento actual que esta viviendo el sector del transporte.

Sergio Fuentes Moreno

Resumen

El trabajo consiste en la aplicación de un sistema multiagente para la optimización de rutas, en este caso un problema de rutado de vehículos con ventanas temporales. La primera parte es teórica, en ella se describen diferentes conceptos que son imprescindibles para la realización del trabajo.

El resto del trabajo describe cómo se ha realizado el sistema de multiagentes y qué algoritmos se utilizan. Finalmente se realizan numerosas experimentaciones con diversas instancias para poder sacar conclusiones al respecto y presentar una serie de resultados.

Abstract

The work consists of the application of a multiagent system for route optimization, in this case a vehicle routing problem with time windows. The first part is theoretical, it describes different concepts that are essential to the work.

The rest of the work describes how the multiagent system has been made and what algorithms are used. Finally, many experiments are carried out with differents instances in order to get conclusions and offer some results.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

c.q.d.	Como queríamos demostrar
e.o.c.	En cualquier otro caso
OA	Agentes Optimizadores
sa	Sujeto a
Min	Minimizar
max	Máximo
min	Mínimo
um	Unidades monetarias

1 OBJETO Y ALCANCE

Este 2020 y 2021 han sido especiales o diferentes respecto a otros años debido a la situación de pandemia en la que nos encontramos. La aparición de un nuevo virus a finales del año 2019 ha provocado la caída de la mayoría de los sectores comerciales. Sin embargo, el sector del transporte es aún más importante de lo que ya era antes, pues debido a esta pandemia, han aumentado las ventas en el canal e-commerce (comercio electrónico). Este canal de ventas ha aumentado en una gran cantidad de países, incluso en alguno de ellos como Turquía y México el aumento ha sido superior al 100% según informe del BBVA [1]. Este aumento obliga a las empresas de transporte a mejorar su modelo de distribución para poder mantener una cierta rentabilidad.

El objetivo es desarrollar un sistema multiagente capaz de abordar un problema de rutado de vehículos con ventanas de tiempo. Este tipo de programas no suelen utilizarse en problemas de optimización ya que, dotar de cierta inteligencia a un programa supone aumentar el tiempo de computación requerido. Sin embargo, los métodos de resolución exactos que existen para resolver el problema que se quiere abordar, tienen muchas limitaciones. Estos motivos y los mencionados en el apartado anterior, nos motivan a realizar el trabajo. Al final, se llevará a cabo un análisis de los resultados obtenidos, resultados que se compararán con las mejores soluciones encontradas para esos problemas.

2 INTRODUCCIÓN

Antes de determinar cuál es el problema que se quiere resolver, se debe conocer los distintos tipos de problemas de enrutamientos de vehículos llamados VRP (Vehicule Routing Problem) que existen, pero sin entrar en detalles. El problema de enrutamiento de vehículos es una generalización del problema del viajante (TSP, Travelling Salesman Problem [2]). La primera investigación sobre los problemas VRP fue en 1959 en un artículo de George Dantiz y John Ramser [3]: "El artículo trata de encontrar la ruta óptima de una flota de camiones de suministro de gasolina entre un terminal de granel y un largo número de estaciones de servicio suministrados por el terminal". Para solucionar este tipo de problemas se proponen distintas heurísticas y metaurísticas, pero en este proyecto se aplicará un sistema multiagentes (se entrará en detalle más adelante) para poder resolver el problema.

2.1 Variantes del VRP

Algunas de las variantes del problema VRP son las siguientes [4]:

-Problema de enrutamiento de vehículos con recogida y entrega (VRPPD): el vehículo tiene que ir hasta un punto de recogida para llevar la mercancía a un punto de entrega.

-Problema de enrutamiento de vehículos con ventanas temporales (VRPTW): las entregas se realizan en una ventana temporal específica.

-Problema de enrutamiento de vehículos con capacidad (CVRP).

-Problema de enrutamiento de vehículos con viajes múltiples (VRPMT): un vehículo puede realizar más de una ruta o viaje.

-Problema de enrutamiento de vehículos abiertos (OVRP): los vehículos no tienen por qué terminar en el depósito.

-Problema de enrutamiento de vehículos de flota mixta (MFVRP): en este caso no existe un solo tipo de vehículo, por tanto, la capacidad y el consumo de energía será diferente para cada tipo.

-Problema de enrutamiento de vehículos con múltiples depósitos (MDVRPR).

-Problema de enrutamiento de vehículos periódicos (PVRP): las entregas se realizan en distintos días.

-Problema de enrutamiento de vehículos con entrega dividida (SDVRP): la mercancía de un cliente puede dividirse en varios vehículos.

2.2 VRPTW

El problema por resolver es el VRP con ventanas temporales o VRPTW. En este apartado se definen las distintas variables, restricciones y datos de nuestro problema.

Para el problema, se propone utilizar el modelo propuesto por Cordone y Calvo en 2001 [5]:

Sea $G = (N, A)$ un grafo dirigido donde $N = \{0,1, \dots, n\}$ son nodos y $A = \{(i,j)\colon i,j \in N\ i \neq j\}$ el conjunto de arcos. El nodo 0 representa el almacén (depot) y $N' = \{1, \dots, n\}$ representa a los clientes o lugares a visitar. Cada arco (i, j) es asociado a un coste de viaje $c_{ij} \geq 0$ y a un tiempo $t_{ij} \geq 0$ .Cada nodo i tiene una demanda q_i, un tiempo de servicio s_i y una ventana de tiempo $[e_i, l_i]$ .Todos los vehículos tienen la misma capacidad Q y el mismo coste $h \geq 0$. A continuación, se asume que $c_{ij} = t_{ij}$,$\forall (i,j) \in A$ y que h es lo suficientemente alto para garantizar que la minimización del número de vehículos es el principal objetivo. La ventana de tiempo debe estrecharse estableciendo $e_i = max\,(e_0 + t_{0i}, e_i)$ y $l_i = min\,(l_0 - t_{0i}, l_i)$.

Establecer $x_{ij} = 1$ si se usa el arco (i, j) y 0 en caso contrario; p_i representa el comienzo del servicio en el nodo

i; y_i representa la carga del vehículo al abandonar el nodo i. La formulación matemática queda del siguiente modo:

$$\text{minimizar} \sum_{j \in N'} h x_{0j} + \sum_{(i,j) \in N'} c_{ij} x_{ij} \tag{1}$$

s. a:

$$\sum_{j \epsilon N} x_{ij} = 1 \qquad \forall i \in N' \tag{2}$$

$$\sum_{i \epsilon N} x_{ij} = 1 \qquad \forall j \in N' \tag{3}$$

$$si \quad x_{ij} = 1 \rightarrow p_i + s_i + t_{ij} \leq p_j \qquad \forall (i,j) \in A \tag{4}$$

$$e_i \leq p_i \leq l_i \qquad \forall i \in N' \tag{5}$$

$$si \quad x_{ij} = 1 \rightarrow y_i + q_j \leq y_j \qquad \forall (i,j) \in A \tag{6}$$

$$q_i \leq y_i \leq Q \qquad \forall i \in N' \tag{7}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{8}$$

Las restricciones (2) y (3) aseguran que cada cliente sea servido por una única ruta. Las ecuaciones (4) y (5) representan las restricciones de las ventanas de tiempo mientras que las ecuaciones (6) y (7) van referidas a la capacidad.

3 REVISIÓN DE LITERATURA

Para resolver el problema se aplica un sistema multiagente. La idea es aprovechar la capacidad de los sistemas de multiagentes de dividir un problema en problemas más pequeños, así como la comunicación entre los distintos agentes para acelerar la búsqueda de soluciones y en este caso para evitar óptimos locales. Jeffrey Wooldridge define a los sistemas multiagentes de la siguiente forma [6]: "Los sistemas multiagentes son sistemas compuestos por múltiples interacciones entre elementos, conocidos como agentes. Los agentes son sistemas con al menos dos importantes capacidades. La primera es que tengan cierta capacidad de realizar acciones de manera autónoma. La segunda, que puedan interactuar con otros agentes." ¿Pero qué es un sistema? Según la Real Academia Española (RAE), un sistema tiene varias definiciones:

-Conjunto de reglas o principios sobre una materia racionalmente enlazados entre sí.

-Conjunto de cosas que relacionadas entre sí ordenadamente contribuyen a determinado objeto.

3.1 Aplicación de sistemas multiagentes a un problema VRP.

En los últimos años se han hecho estudios sobre como implementar un sistema multiagente para resolver un problema VRP. Una de las aproximaciones más exitosas es la del concepto de un equipo asíncrono (A-Team), un grupo de agentes que intentan evolucionar una serie de soluciones. Dariusz Barbucha y Piotr Jedrzejowicz [7] proponen usar "JADE-based A-Team llamado JABAT". JADE por sus siglas en inglés significa Java Agent Development Framework.

JABAT usa un grupo de agentes, cada uno representa un algoritmo para solucionar el problema. Para evitar estancarse en un óptimo local, estas soluciones son mejoradas por otros agentes independientes y así tener mayor posibilidad de encontrar el óptimo global [3] [8].

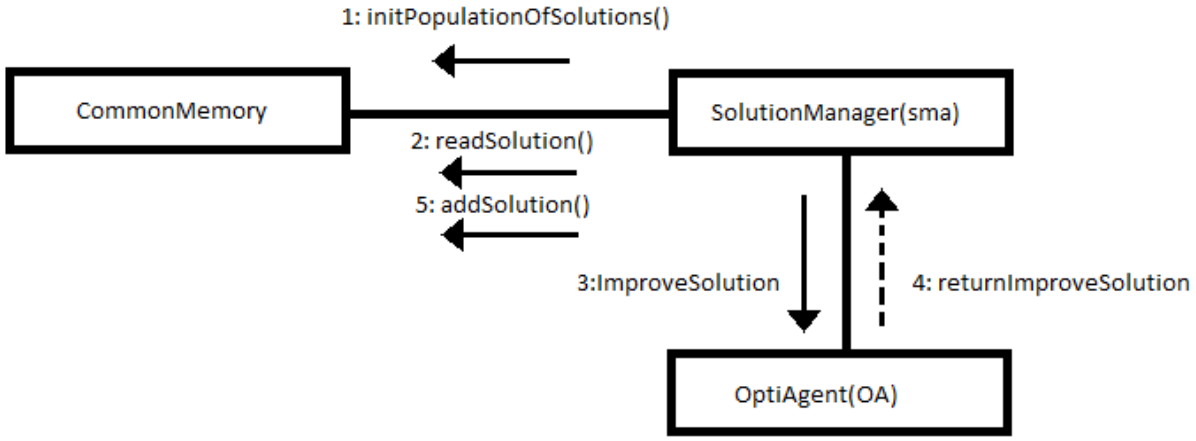


Ilustración 1 Diagrama de comunicación del proceso de resolución de JABAT [8]

En el anterior diagrama se puede observar los dos agentes de los que se hace uso:

-OptiAgent: formado por distintos algoritmos de mejora (optimization agents).

-SolutionManager: selecciona una solución de la población inicial de soluciones, la envía a los agentes de optimización y la sustituye en caso de mejora.

3.1.1 Heurísticas.

Antes de hablar de las heurísticas, se debe mencionar que existen métodos exactos para el VRPTW, pero estos métodos solo suelen encontrar soluciones para problemas de hasta 50 clientes. El objetivo de este trabajo es desarrollar un sistema multiagente capaz de resolver el VRPTW sin limitaciones de clientes y, por tanto, no se tendrán en cuentas estos métodos para el desarrollo del trabajo.

Linda Bibiana Rocha, Elsa Cristina González y Javier Arturo Orjuela definen las heurísticas como procedimientos que proporcionan soluciones de aceptable calidad mediante una exploración limitada del espacio de búsqueda. En su artículo clasifican las heurísticas en tres grupos: métodos constructivos, métodos de dos fases y heurísticas de mejora [9]. Edwin Montes Orozco explica muy bien estos tres grupos en su trabajo de fin de grado [10] , diferenciándolos de la siguiente forma.

Métodos constructivos: Rafael Marti los define como métodos que construyen paso a paso la solución a un problema. Basados en su mayoría, en la mejor elección en cada iteración [11].

Métodos de dos fases: Algunos de estos métodos son de asignación elemental, el algoritmo de ramificación y acotamiento truncados, los procedimientos de búsqueda local, etc.

Heurísticas de mejora: conocidos como procedimientos de búsqueda local. Partiendo de una solución inicial, se busca una solución vecina mejor y se reemplaza.

Esquema de la clasificación de heurísticas:

- Heurísticas
  - Métodos constructivos
  - Métodos de dos fases
  - Heurísticas de mejora

3.1.2 Métodos para obtener la población inicial de soluciones.

Para obtener una solución inicial, Marius M. Solomon en un artículo [12] propone dos tipos de heurísticas basadas en métodos constructivos para resolver el problema VRPTW.

La primera heurística es una heurística de ahorro. Se parte de una solución inicial en la que cada cliente es visitado por un vehículo, y se intenta reducir el número de vehículos uniendo las rutas en caso de que se produzca un ahorro en esa unión. Uno de los algoritmos más conocidos es el de Clarke-Wright [13].

La segunda es una heurística de inserción. Se parte de cero rutas o vehículos, se insertan clientes de uno en uno en la ruta según una serie de criterios, si deja de cumplir las restricciones del problema se abre una nueva ruta.

- Métodos constructivos
  - Heurísticas de ahorro
  - Heurísticas de inserción

3.1.3 Heurísticas de búsqueda local.

Algunos de los algoritmos de búsqueda local utilizados para resolver el VRPTW que aparecen en [10], son los siguientes.

- El operador de intercambio λ: Propuesto por Lin en 1965. Se dice que un intercambio λ consiste en eliminar λ aristas de la solución y reconectar los λ segmentos restantes. El valor de λ depende del intercambio que se quiera hacer, y puede tomar distintos valores. Una solución es óptima si no puede ser mejorada utilizando λ intercambios [10].

- El algoritmo de Lin-Kernighan: Propuesto en 1973. Para el algoritmo es necesario partir de una solución inicial T. Se inicia $i = 1$ y se selecciona x_i e y_i como el par más alejado en el paso i. Tras realizar una serie de iteraciones, si la mejor solución se da para $i = k$ entonces se intercambia $x_1, \dots, x_i$ con $y_1, \dots, y_i$ para dar una nueva solución T [14].

- El operador Or-opt: Propuesto por Ilhan Or en 1976 [15]. Consiste en eliminar un segmento de k clientes de una ruta y colocar ese segmento en otra parte de la ruta, sin variar el orden de los clientes dentro del segmento extraído [10].

Ilustración 2 Intercambio Or-Opt de 2 clientes

- GENI y US: Propuesto por M. Gendreau, A. Hertz y G. Laporte en 1992. GENI es un método de inserción generalizado (GENeralized Insertion). Adicionalmente, se propone un algoritmo posterior a la optimización llamado US, por sus siglas en inglés, Unstringing y Stringing [16].

- Algoritmos de transferencias cíclicas: Thompson y Psaraftis proponen este algoritmo en 1993 [17]. Estos algoritmos buscan eliminar clientes de una ruta y coolocarlos en otra de manera cíclica [18].

- Operadores de Van Breedam: En 1995 Van Breedam propone dos operadores para intercambiar clientes entre dos rutas diferentes. Estos operadores son denominados como String Relocation y String Exchange [19] [18].

3.1.4 Metaheurísticas.

En este apartado se explicarán algunas de las metaheurísticas más utilizadas para la resolución de un problema VRP.

En el blog de la Universidad Politécnica de Valencia [20] se definen las metaheurísticas como la combinación de manera inteligente de diversas técnicas con el fin de explorar el espacio de soluciones. También se proporciona la definición de Osman y Kelly (1996) [21]: *"Los procedimientos metaheurísticos son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son ni efectivos ni eficientes. Los metaheurísticos proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y la mecánica estadística".* A continuación, se explican algunas metaheurísticas.

- Algoritmos genéticos: fueron propuestos por Holland en 1975. En el año 1989 Goldberg los define como [22]: "*los Algoritmos Genéticos son algoritmos de búsqueda basados en la mecánica de*

selección natural y de la genética natural. Combinan la supervivencia del más apto entre estructuras de secuencias con un intercambio de información estructurado, aunque aleatorizado, para constituir así un algoritmo de búsqueda que tenga algo de las genialidades de las búsquedas humanas"

- Recocido simulado: propuesto por primera vez en 1983 por Kirkpatrick, Gelatt y Vecchi, resumen su artículo de la siguiente manera [23]: *"existe una conexión profunda y útil entre la mecánica estadística (el comportamiento de sistemas con muchos grados de libertad en equilibrio térmico a una temperatura finita) y optimización multivariante o combinatoria (encontrar el mínimo de una función dada dependiendo de muchos parámetros). Una analogía detallada con el recocido en sólidos proporciona un marco para la optimización de las propiedades de sistemas grandes y complejos. Esta conexión con la mecánica estadística proporciona una perspectiva desconocida sobre los problemas y métodos tradicionales de optimización."*

- Algoritmos de Hormigas: propuesto por Marcos Dorigo en 1992 [24]. El algoritmo se basa en el comportamiento de las hormigas a la hora de buscar comida. Las hormigas siguen a otras gracias a un rastro de feromonas que dejan a su paso. Cuanto más corto sea el camino, mayor es la concentración de feromonas, esto les permite encontrar la ruta más corta del hormiguero a la comida.

- Búsqueda Tabú: propuesto por Glover en 1986 [25]. Este algoritmo busca realizar métodos heurísticos de búsqueda local añadiendo una memoria, de manera que dota de cierta inteligencia a la heurística.

3.2 Programas considerados para programar el problema.

3.2.1 AnyLogic

AnyLogic es un programa que permite construir modelos de simulación de tres tipos: sistemas dinámicos, eventos discretos y basados en agentes. Crear un modelo es aparentemente sencillo, para modelos básicos no es necesario saber programación, se pueden crear seleccionando elementos con el ratón y arrastrándolos hacia la pantalla. Sin embargo, para modelos más realistas, es necesario conocer el lenguaje de programación llamado Java, ya que AnyLogic está basado en este.

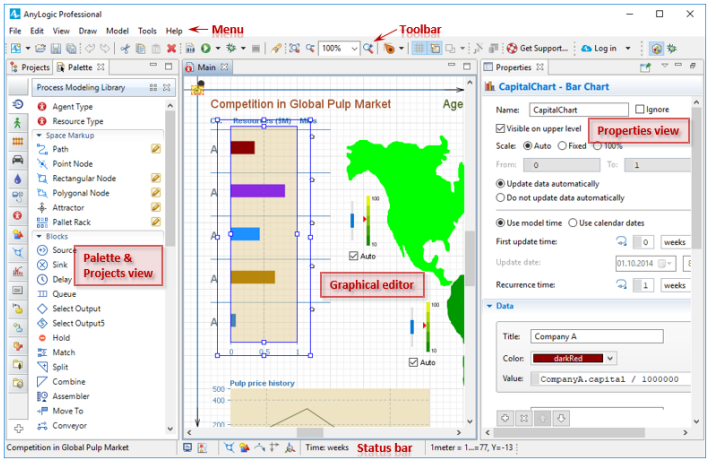


Ilustración 3 Interfaz de usuario de AnyLogic (help.anylogic.com)

3.2.1.1 Java

Java fue creado por James Goslin y su equipo en 1991, y posteriormente publicado por Sun Microsystems en el año 1995. El objetivo era crear un programa similar a C++ pero con su propia máquina virtual. Una de las principales y más importantes características de Java es su independencia de la plataforma (sistema operativo), es decir, que programas escritos en Java pueden ejecutarse en cualquier entorno (hardware) ya sea Linux, Windows, macOS, etc.

3.2.2 Python

Python es un lenguaje de alto nivel cuya primera aparición fue en el año 1991 de la mano de Guido van Rossum. En los años ochenta, Guido comenzó a trabajar en un equipo encargado de desarrollar un lenguaje llamado ABC. Finalmente, no tuvo éxito y Guido empezó en otro proyecto en 1986 llamado Amoeba, fue ahí cuando empezó a crear una variante del ABC seleccionando las mejores características de este y tratando de evitar sus fallos. Además, hizo posible que Python fuera un programa independiente de la plataforma y capaz de soportar librerías, características que no se encontraban en ABC [26].

Python es administrado por Python Software Fundation y tiene una licencia de código abierto, por lo que se puede hacer uso del programa de manera gratuita.

El conocimiento previo del lenguaje Python y el desarrollo previo de un sistema multiagente en Java, empujan al desarrollo del trabajo en Python.

3.3 Librerías/entornos para programar sistemas multiagentes en Python

Una librería es un archivo que puede usarse en un ejecutable y que tiene una serie de funcionalidades, el archivo aporta variables y funciones definidas por un tercero. Por ejemplo, una de las librerías más conocidas de Python es "Matplotlib" que sirve para generar gráficos a partir de unos datos. En nuestro trabajo, se ha usado Matplotlib para facilitar la interpretación de los resultados y así tener una referencia visual de la disposición de los clientes, almacén, rutas etc.

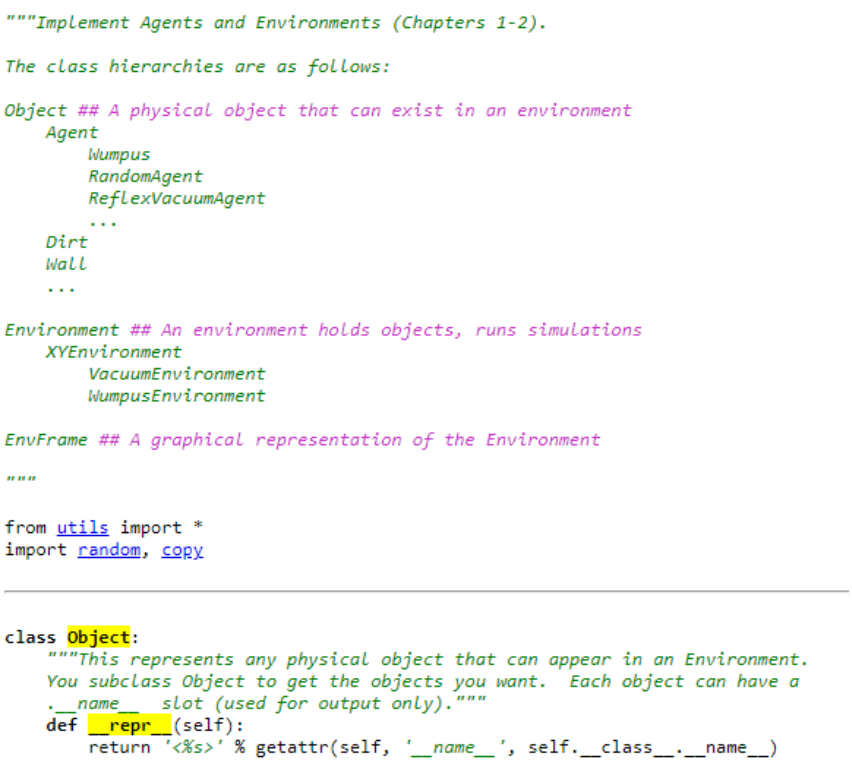
3.3.1 Librería Spade

Smart Python Agent Development Environment [27] es una plataforma para sistemas multiagentes escrita en Python y basada en XMPP (un protocolo de mensajería de respuesta instantánea). Algunas de sus características son las siguientes:

- Requiere una versión igual o superior a Python 3.6
- Basado en Asyncio (librería para entornos asíncronos) [28]
- Interfaz web gráfica
- Permite añadir características o funciones nuevas.
- Cualquier servidor XMPP es válido (hay que instalar un servidor)

3.3.2 Librería Aima [29]

El archivo (librería) contiene un resumen de los algoritmos que aparecen en el libro *Artifficial Intelligence: A Modern Approach.* El código es de libre uso y require una versión igual o superior a Python 2.2.



```
"""Implement Agents and Environments (Chapters 1-2).

The class hierarchies are as follows:

Object ## A physical object that can exist in an environment
    Agent
        Wumpus
        RandomAgent
        ReflexVacuumAgent
        ...
    Dirt
    Wall
    ...

Environment ## An environment holds objects, runs simulations
    XYEnvironment
        VacuumEnvironment
        WumpusEnvironment

EnvFrame ## A graphical representation of the Environment

"""

from utils import *
import random, copy

class Object:
    """This represents any physical object that can appear in an Environment.
    You subclass Object to get the objects you want.  Each object can have a
    .__name__  slot (used for output only)."""
    def __repr__(self):
        return '<%s>' % getattr(self, '__name__', self.__class__.__name__)
```

Ilustración 4 Ejemplo de parte del código de Aima

3.3.3 Librería Mesa [30]

Mesa es un entorno modular para construir, analizar y visualizar modelos de agentes. Para instalarlo se recomienda usar un entorno virtual y requiere de la versión 3 de Python. Para construir un modelo se puede:

- Escribir el código en un editor de texto.
- Crearlo de manera interactiva con Jupyter Notebook, un programa web de libre uso.

3.3.4 Resumen de librerías analizadas

	Spade	Aima	Mesa
Versión de Python	3.6	2.2	3
Herramienta de análisis	No	Sí	Sí
Interfaz gráfica web	Sí	No	Sí
Únicamente usa XMPP server	Sí	No	No
Claridad en la explicación de uso del código	Bien	Mal	Bien

Tabla 1 Comparativa librerías para programar multiagentes

3.3.5 Elección final

Tras comenzar a programar los algoritmos de mejora que acabarían siendo los agentes optimizadores, se ha observado que es más sencillo programar el entorno de intercambio de información entre los agentes sin el uso de una librería para ello y así no es necesario amoldar el código a la librería. De esta manera, se ha programado una función que, con un bucle, simula el intercambio de información entre un agente encargado de seleccionar e intercambiar soluciones con los agentes optimizadores, es decir, simula el diagrama de comunicación del proceso de resolución de JABAT que se puede observar en la ilustración 1.

4 APLICACIÓN PRÁCTICA

4.1 Datos del problema VRPTW

Los datos que se utilizan se obtienen de las instancias de Solomon [12] y [31]. Estas instancias son problemas del VRPTW que vienen clasificados en diferentes grupos. Según estos grupos varía la distribución de los clientes o las ventanas temporales, y para cada grupo se tienen instancias de 25, 50 y 100 clientes. Estas instancias vienen dadas por los grupos C1, C2, R1, R2, RC1 y RC2. Las instancias tipo C tienen los clientes de manera agrupada, en los tipos R están dispersos de manera uniforme y los tipos RC son una mezcla de ambos. En cuanto al número que sigue a la letra, va referido a las ventanas temporales.

En las siguientes ilustraciones se puede ver la distribución de clientes de algunas de las instancias de los grupos mencionados anteriormente. Estas gráficas han sido construidas con Matplotlib y representan un plano en dos dimensiones compuesto por los ejes de coordenadas x e y. El punto rojo indica el almacén de partida, el resto son los clientes, se representa una vista de planta.

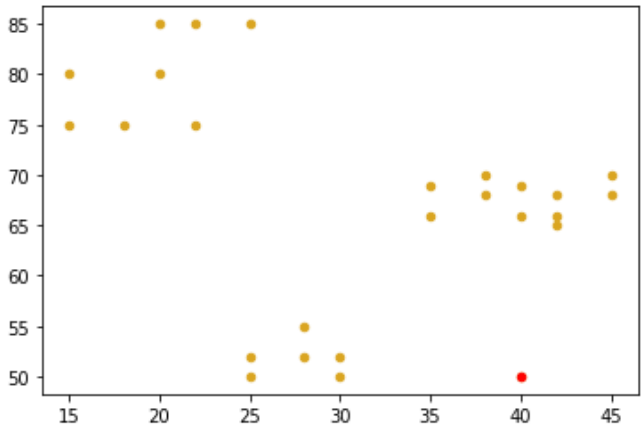


Ilustración 5 Distribución de instancia C103_025

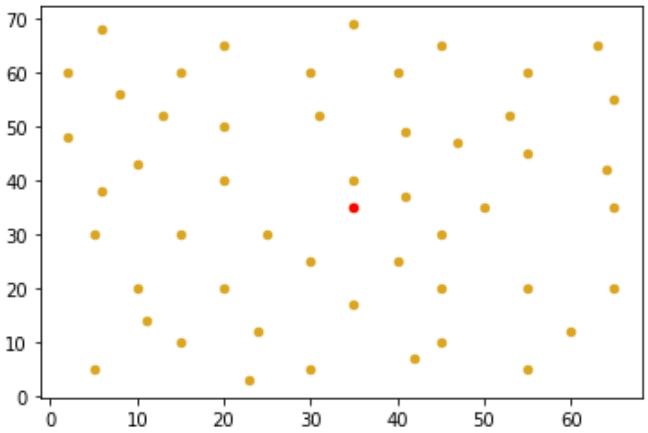


Ilustración 6 Distribución de instancia R101_050

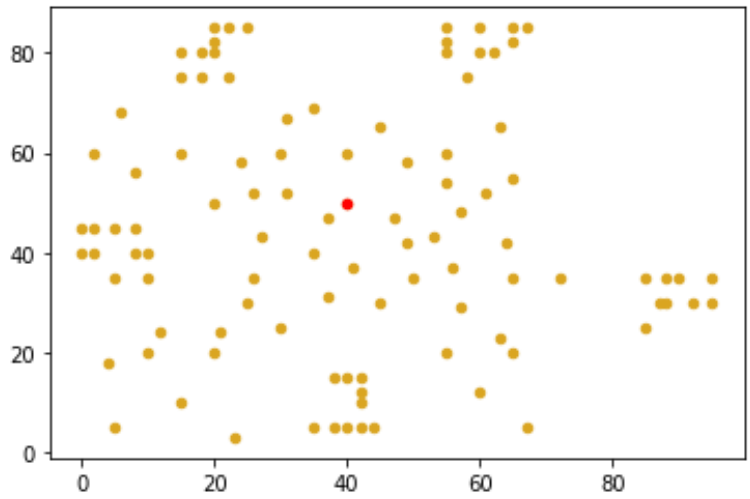


Ilustración 7 Distribución de instancia RC101_100

4.2 Población inicial de soluciones

El objetivo es resolver un problema de rutado de vehículos con ventanas temporales utilizando multiagentes. Para ello el primer paso es obtener una población inicial de soluciones para posteriormente mejorarlas. La población inicial es necesaria para evitar caer en un óptimo local. Que una solución inicialmente sea mejor que otra no quiere decir que tras aplicar una serie de mejoras y alcanzar un óptimo, este sea mejor que el óptimo alcanzado para la otra solución inicialmente peor. En la siguiente ilustración se puede observar un ejemplo de por qué es necesario partir de una población de soluciones.

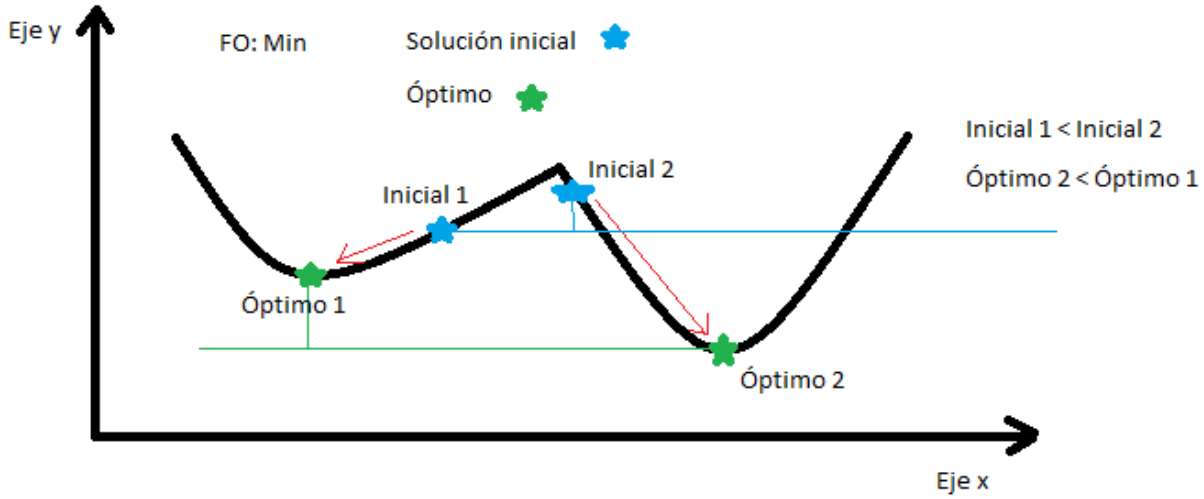


Ilustración 8 Ejemplo de gráfica con dos óptimos y dos soluciones iniciales.

Para obtener una solución inicial, hemos optado por heurísticas de inserción [12]. En total han sido cuatro soluciones las que se han obtenido, siguiendo una misma filosofía, pero cambiando el criterio (variando el paso 1 del pseudocódigo). Inicialmente todos los clientes están sin ruta. A continuación, podemos ver los pseudocódigos de los cuatro algoritmos utilizados.

Pseudocódigo del algoritmo correspondiente a la primera solución. Más cercano.

Hasta que todos los clientes tengan una ruta asignada:

1.Para iniciar la ruta, buscar el cliente sin ruta más cercano al almacén y añadir a la ruta si cumple las restricciones.

2. Buscar el cliente sin ruta más cercano al último cliente añadido en la ruta.

3. Comprobar si al añadir el cliente se cumplen las restricciones del problema.

4. Si cumple las restricciones se vuelve al paso 2, e.o.c iniciar una nueva ruta y volver al paso 1.

5. Devolver el valor de la función objetivo.

Pseudocódigo del algoritmo correspondiente a la segunda solución. Ventana estrecha.

Hasta que todos los clientes tengan una ruta asignada:

1.Para iniciar la ruta, buscar el cliente sin ruta con la ventana temporal más estrecha (esto quiere decir menor diferencia entre el tiempo final e inicial para realizar la entrega) y añadir a la ruta si cumple las restricciones.

2. Buscar el cliente sin ruta más cercano al último cliente añadido en la ruta.

3. Comprobar si al añadir el cliente se cumplen las restricciones del problema y calcular el valor de la función objetivo.

4. Si cumple las restricciones se vuelve al paso 2, e.o.c iniciar una nueva ruta y volver al paso 1.

5. Devolver el valor de la función objetivo.

Pseudocódigo del algoritmo correspondiente a la tercera solución. Más lejano.

Hasta que todos los clientes tengan una ruta asignada:

1.Para iniciar la ruta, buscar el cliente sin ruta más lejano al almacén y añadir a la ruta si cumple las restricciones.

2. Buscar el cliente sin ruta más cercano al último cliente añadido en la ruta.

3. Comprobar si al añadir el cliente se cumplen las restricciones del problema.

4. Si cumple las restricciones se vuelve al paso 2, e.o.c iniciar una nueva ruta y volver al paso 1.

5. Devolver el valor de la función objetivo.

Pseudocódigo del algoritmo correspondiente a la cuarta solución. Menor tiempo de inicio de entrega

Hasta que todos los clientes tengan una ruta asignada:

1.Para iniciar la ruta, buscar el cliente sin ruta con el tiempo de inicio de entrega más temprano y añadir a la ruta si cumple las restricciones.

2. Buscar el cliente sin ruta más cercano al último cliente añadido en la ruta.

3. Comprobar si al añadir el cliente se cumplen las restricciones del problema.

4. Si cumple las restricciones se vuelve al paso 2, e.o.c iniciar una nueva ruta y volver al paso 1.

5. Devolver el valor de la función objetivo.

4.3 Agentes Optimizadores

Para la creación de los distintos agentes optimizadores, aparte de los conocimientos adquiridos en la carrera, se han utilizado estrategias de búsqueda local similares a las desarrolladas en [7] , [32] y [10] . Algunas de estas heurísticas han sido ya comentadas en el apartado 3.1.3.

4.3.1 OA Vecinos

Con este algoritmo, los agentes tratan de mejorar el objetivo intercambiando el orden en que se visitan clientes vecinos dentro de una misma ruta.

Pseudocódigo

Para cada ruta i desde j=1

1.Intercambiar la posición del cliente j con el cliente j+1 de la ruta i.

2.Comprobar si cumple las restricciones y calcular el valor de la función objetivo.

3.En caso de no cumplir las restricciones invertir el cambio hecho, e.o.c mantener el cambio.

4.Si hay clientes para seguir intercambiando, volver al paso 1 incrementando el valor de j, es decir, intercambiar con el siguiente cliente.

5.Una vez se agotan los clientes a intercambiar en la ruta, pasar a la siguiente ruta incrementando el valor de i en el algoritmo y comenzar de nuevo por el primer cliente.

6. Devolver el valor de la función objetivo una vez finalizado el algoritmo con todas las rutas.

4.3.2 OA Buscar solitario

Tras obtener la población inicial de soluciones, se ha observado que se obtienen o se pueden obtener rutas en las que se visita a un solo cliente. Debido a que el coste de utilizar un vehículo es muy elevado, unos de los objetivos es minimizar el número de vehículos o rutas (hay un vehículo para cada ruta). Para ello se ha propuesto este algoritmo que busca rutas de un solo cliente y trata de colocar ese cliente como primer cliente a visitar en otra ruta.

Pseudocódigo

Mientras no encuentre una solución o se agoten las rutas.

1.Buscar la primera ruta en la que solo tenga un cliente y guardar el cliente en la variable solitario.

2.Para cada ruta distinta a la del solitario empezando por la primera, desplazar todos los clientes un espacio a la derecha.

3.Insertar el cliente solitario en la primera posición de la ruta.

4.Comprobar restricciones y función objetivo.

6.Si no cumple las restricciones o la solución es peor, deshacer el cambio y cambiar a la siguiente ruta volviendo al paso 2. Terminar el algoritmo e.o.c.

7.Si tras probar en todas las rutas no se ha encontrado una solución mejor para insertar el solitario encontrado, volver al paso 1 partiendo de la ruta posterior a la original del solitario anterior.

8. Devolver el valor de la función objetivo.

4.3.3 OA String Cross

Para este algoritmo nos hemos basado en el algoritmo llamado OA_StringCross que se propone en JABAT [7]. La diferencia entre ambos algoritmos se da en que en JABAT se contemplan todos los posibles puntos de reconexión, mientras que este algoritmo los genera de manera aleatoria.

Pseudocódigo

1. Generar dos rutas aleatorias, ruta_i y ruta_j, que al menos tengan un cliente y que sean distintas.

2.Generar dos puntos aleatorios pi y pj en cada ruta diviendo cada ruta en dos partes.

3. Crear una nueva ruta i formada por la primera parte de ruta_i desde 1 hasta pi, y la segunda de la ruta j formada desde pj+1 hasta el último cliente n de j (1…pi, pj+1…n).

4. Crear una nueva ruta j formada por la primera parte de ruta_j desde 1 hasta pj, y la segunda de la ruta_i formada desde pi+1 hasta el último cliente m de i (1…pj, pi+1…m).

5. Comprobar las restricciones y calcular el valor de la función objetivo.

6. Si no cumple las restricciones o el valor de la función objetivo es mayor, deshacer los cambios.

7. Devolver el valor de la función objetivo.

4.3.4 OA Insertion

El agente llamado Insertion es un algoritmo de inserción, el algoritmo propone un cliente aleatorio de una ruta y varía el orden de visita del cliente hasta lograr una mejora (en caso de existir esa mejora), manteniendo el resto de los clientes en el mismo orden.

Pseudocódigo

1. Generar una ruta aleatoria que tenga al menos 1 cliente en su ruta.

2. Generar una posición aleatoria pos e iniciamos j=1.

3. Mientras no exista una mejor solución y haya clientes sin intercambiar, intercambiar el cliente j con el cliente de la posición pos.

4. Comprobar restricciones y calcular el valor de la función objetivo.

5. Si no cumple las restricciones o el valor de la función objetivo es peor, deshacer el cambio, incrementar j y volver al paso 3. Finalizar el algoritmo e.o.c.

6. Devolver el valor de la función objetivo.

4.3.5 OA Intercambio

Este algoritmo proviene de la idea de mezclar el algoritmo de Insertion con el algoritmo String Cross. El algoritmo funciona igual que Insertion, pero esta vez intercambia las posiciones con una ruta distinta y no con la misma.

Pseudocódigo

1. Generar dos rutas aleatorias, ruta_i y ruta_j, con al menos un cliente en la ruta.

2. Generar una posición aleatoria para la ruta_i llamada pos e inicializar j en 1.

3. Mientras haya clientes para intercambiar en ruta_j y no exista una solución mejor. Intercambiar el cliente j con el cliente de la ruta_i en la posición pos.

4. Comprobar restricciones y función objetivo.

5. Si no cumple las restricciones o el valor de la función objetivo es peor, deshacer el cambio, incrementar j y volver al paso 3. Finalizar el algoritmo e.o.c.

6. Devolver el valor de la función objetivo.

4.4 Solution Manager y entorno

Como se dijo anteriormente, no se hará uso de librerías de multiagentes. Para simular la interacción entre los agentes optimizadores y el agente Solution Manager, se ha creado este último como una función en la que se llamará a otras funciones las cuales serán los agentes optimizadores. Para intercambiar información lo único que se deberá hacer es introducir la función SolutionManager en un bucle, los agentes optimizadores son funciones que devuelven el valor de la función objetivo y modifican las matrices de rutas y arcos, mientras que SolutionManager modifica un vector formado por la población inicial de soluciones y por otro lado las distintas matrices de rutas y arcos para cada solución. El programa en su conjunto simula la interacción entre diferentes agentes. A continuación, se puede observar el pseudocódigo y en la ilustración 8 un ejemplo de código.

Pseudocódigo

1. Seleccionar una solución con una probabilidad que depende de la calidad de la solución. Ordenando un vector que contiene las soluciones de mayor a menor valor de la función objetivo, se le da una puntuación a cada solución correspondiente a su posición en dicho vector, siendo la puntuación de uno a cinco puntos. De esta forma se genera un número aleatorio de uno a diez con el cuál podremos seleccionar la solución correspondiente.

2. Copiar en quince variables auxiliares (porque hay cinco agentes optimizadores y tres datos a copiar) el valor de la función objetivo, la matriz de ruta y la matriz de arcos correspondientes a la solución seleccionada.

3. Pasar los datos copiados a los agentes optimizadores para que proporcionen una solución igual o mejor a la original.

4. Seleccionar una de las cinco soluciones obtenidas según un criterio (aleatorio, la mejor solución y la segunda peor solución). En [7] proponen hasta seis criterios diferentes tanto para seleccionar una solución a mejorar, como para añadirla a la población de soluciones. El criterio se selecciona dando un valor para la variable “elección” comprendido entre 1 y 3. Para la resolución de los problemas, “elección” se ha asignado de las siguientes tres formas; fijando el valor a 1, fijando el valor a 2 y

generando un valor aleatorio entre 1 y 3.

4.1. Elección = 1: se selecciona la solución de manera aleatoria entre las cinco soluciones proporcionadas por los agentes.

4.2. Elección = 2: se selecciona la mejor solución entre las proporcionadas por los agentes.

4.3. Elección = 3: se selecciona la segunda peor solución entre las proporcionadas por los agentes.

5. Sustituir la solución de la población de soluciones que se había seleccionado por la nueva solución obtenida de los agentes proveniente del paso cuatro.

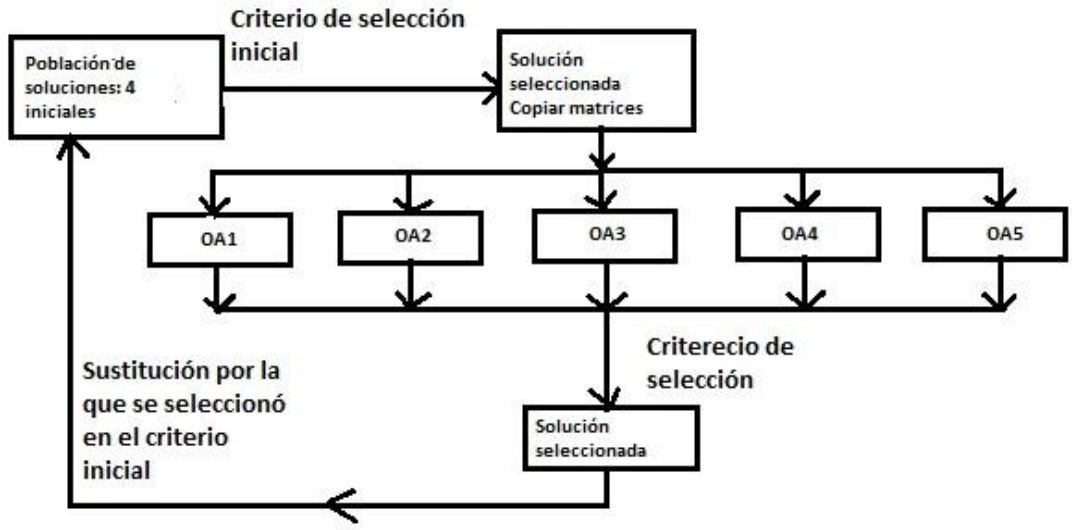


Ilustración 9 Esquema SolutionManager. Ejemplo de intercambio de información

5 EXPERIENCIA COMPUTACIONAL

En este apartado se darán las características del ordenador utilizado, el programa y la versión de Python, así como los distintos criterios utilizados a la hora de solucionar las distintas instancias y recoger los datos.

5.1 Características del ordenador utilizado.

Para este trabajo, se ha utilizado un ordenador de sobremesa con la última versión de Windows 10 de 64 bits. El ordenador está compuesto por un procesador Ryzen 5 1600 AF y una gráfica AMD Radeon RX 570.

5.2 Programa y versión de Python

El código para la resolución del problema ha sido desarrollado en el programa Spyder, un entorno de desarrollo gratuito escrito en Python. En concreto se hace uso de la versión 5 de Spyder y la versión 3.7.9 de Python.

5.3 Criterios para la recogida de datos

El código incluye aleatoriedad, por tanto, la decisión ha sido generar treinta soluciones con el mismo criterio para obtener posteriormente una media. Cada instancia se resuelve con los tres criterios comentados en el apartado 4.4 para dar un valor a la variable "elección". De esta manera se han obtenido tres resultados para "elección" =1, tres para "elección" =2 y tres para "elección" =aleatorio (1,3).¡

La función SolutionManager se ha introducido en un bucle de 300 iteraciones para simular el intercambio de información, en algún caso se incrementará el número de iteraciones para observar como mejora la solución en periodo de tiempo mayor. El número de iteraciones en el bucle equivale a tiempo de intercambio de información entre agentes.

En cuanto a las instancias utilizadas, se utilizan tres instancias de cada grupo de instancias de Solomon ya comentadas en el apartado 4.1. Las instancias son de 50 clientes, no se ha optado por las instancias de 100 clientes para no incrementar el tiempo de ejecución. Los resultados para poder realizar una comparación se obtienen de [33], para poder utilizarlos es necesario tener en cuenta el número de vehículos ya que no se tienen en cuenta en la función objetivo de [33] y sí en la de este trabajo. En algunos casos se utiliza la instancia de 25 clientes debido a que no se dispone de resultados de la instancia de 50, el motivo por el cual esas instancias no aparecen es porque los métodos utilizados en [33] no han conseguido resolverlas y no se han encontrado en otros documentos.

Para las restricciones, se supone un valor de 100 para cada vehículo o ruta y una velocidad de 1, es decir, si la distancia es 10 tardará en recorrer la distancia un total de 10 unidades temporales.

6 RESULTADOS Y CONCLUSIONES

Para la obtención de los resultados, se han utilizado las siguientes instancias de 50 clientes de Solomon:

C101.50, C102.50, C103.50, RC101.50, RC102.50, RC103.50, R201.50, R202.50, R205.50, C201.50, C202.50, C203.50, RC201.50, RC201.25, RC202.25, R101.50, R102.50 y R103.50.

En todo momento se utiliza el punto como separador decimal y la coma como separador de miles.

6.1 Obtención de resultados.

En el apartado 5.3 se explicaron los criterios para la obtención de resultados. Cuando se habla de solución se habla del valor de la función objetivo. A continuación, se definen los diferentes términos que se utilizan en las tablas.

- Inicial x: solución inicial número x en um. Hay un total de 4.
- Mejorada x: solución inicial número x en um. tras correr el código.
- % Mejora x: $\frac{Inicial\ x - Mejorada\ x}{Inicial\ x} * 100$
- Tiempo ejec: es el tiempo de ejecución del código en segundos.
- Selec elección 1: el criterio para seleccionar la solución recibida de los agentes es escoger una solución de manera aleatoria.
- Selec elección 2: el criterio para seleccionar la solución recibida de los agentes es escoger la mejor solución obtenida.
- Selec elección rand: el criterio para seleccionar la solución recibida de los agentes es, de manera aleatoria seguir el criterio de selección 1, de selección 2 o escoger la segunda peor solución obtenida de los agentes.

A continuación, se observan las diferentes tablas que se han generado para cada instancia. Las tablas muestran la media de las treinta experimentaciones que se han realizado para cada criterio e instancia. En amarillo se subraya la mejor solución de las que conforman la tabla.

	C101.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	2220.90	2220.90	2220.90
Inicial 2	3650.41	3650.41	3650.41
Inicial 3	4963.21	4963.21	4963.21
Inicial 4	1857.55	1857.55	1857.55
Mejorada 1	1961.90	1933.97	2072.46
Mejorada 2	2416.81	3438.66	2086.28
Mejorada 3	4024.74	4728.64	4017.10
Mejorada 4	1551.32	1569.78	1583.25
% Mejora 1	11.66%	12.92%	6.68%
% Mejora 2	55.55%	9.53%	70.43%
% Mejora 3	42.26%	10.56%	42.60%
% Mejora 4	13.79%	12.96%	12.35%
Tiempo ejec	43.02	41.93	42.51

Tabla 2 Resultados medios de la instancia C101.50

	C102.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	1997.35	1997.35	1997.35
Inicial 2	3353.34	3353.34	3353.34
Inicial 3	3952.80	3952.80	3952.80
Inicial 4	2474.19	2474.19	2474.19
Mejorada 1	1732.58	1674.99	1750.71
Mejorada 2	2168.96	2343.79	2312.98
Mejorada 3	2787.13	2568.16	2682.97
Mejorada 4	2003.56	1906.60	1910.61
% Mejora 1	13.26%	16.14%	12.35%
% Mejora 2	59.30%	50.54%	52.09%
% Mejora 3	58.36%	69.32%	63.58%
% Mejora 4	23.56%	28.42%	28.22%
Tiempo ejec	44.25	43.32	43.44

Tabla 3 Resultados medios de la instancia C102.50

	C103.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	1427.08	1427.08	1427.08
Inicial 2	2890.85	2890.85	2890.85
Inicial 3	2815.50	2815.50	2815.50
Inicial 4	1925.50	1925.50	1925.50
Mejorada 1	1264.29	1084.69	1281.67
Mejorada 2	1840.46	1659.30	1946.05
Mejorada 3	2052.00	2027.23	1953.62
Mejorada 4	1522.94	1488.54	1466.02
% Mejora 1	11.41%	23.99%	10.19%
% Mejora 2	73.60%	86.30%	66.21%
% Mejora 3	53.50%	55.24%	60.40%
% Mejora 4	28.21%	30.62%	32.20%
Tiempo ejec	44.63	44.59	45.22

Tabla 4 Resultados medios de la instancia C103.50

	RC101.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	4441.00	4441.00	4441.00
Inicial 2	4552.80	4552.80	4552.80
Inicial 3	4298.73	4298.73	4298.73
Inicial 4	3538.67	3538.67	3538.67
Mejorada 1	4191.58	4097.72	4086.55
Mejorada 2	4323.53	4262.28	4132.05
Mejorada 3	4234.14	4058.27	4261.56
Mejorada 4	3037.18	3021.22	3212.19
% Mejora 1	5.62%	7.73%	7.98%
% Mejora 2	5.16%	6.54%	9.47%
% Mejora 3	1.45%	5.41%	0.84%
% Mejora 4	11.29%	11.65%	7.35%
Tiempo ejec	43.10	42.59	44.06

Tabla 5 Resultados medios de la instancia RC101.50

	RC102.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	4116.95	4116.95	4116.95
Inicial 2	3914.73	3914.73	3914.73
Inicial 3	3607.88	3607.88	3607.88
Inicial 4	3250.66	3250.66	3250.66
Mejorada 1	3733.67	3640.73	3167.60
Mejorada 2	3542.24	3682.38	3867.56
Mejorada 3	3533.02	3320.58	3534.20
Mejorada 4	2887.41	3000.90	3014.51
% Mejora 1	9.31%	11.57%	23.06%
% Mejora 2	9.05%	5.64%	1.15%
% Mejora 3	1.82%	6.98%	1.79%
% Mejora 4	8.82%	6.07%	5.74%
Tiempo ejec	44.34	44.29	44.16

Tabla 6 Resultados medios de la instancia RC102.50

	RC103.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	3008.03	3008.03	3008.03
Inicial 2	3204.26	3204.26	3204.26
Inicial 3	3404.33	3404.33	3404.33
Inicial 4	2905.55	2905.55	2905.55
Mejorada 1	2830.12	2387.05	2247.36
Mejorada 2	2787.93	2949.32	2606.88
Mejorada 3	3362.43	3186.19	3346.48
Mejorada 4	2422.40	2443.75	2471.73
% Mejora 1	5.91%	20.64%	25.29%
% Mejora 2	13.84%	8.48%	19.86%
% Mejora 3	1.39%	7.25%	1.92%
% Mejora 4	16.06%	15.35%	14.42%
Tiempo ejec	43.32	43.45	43.57

Tabla 7 Resultados medios de la instancia RC103.50

	R201.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	3754.14	3754.14	3754.14
Inicial 2	4890.56	4890.56	4890.56
Inicial 3	4251.89	4251.89	4251.89
Inicial 4	3112.54	3112.54	3112.54
Mejorada 1	3229.29	3143.04	2971.45
Mejorada 2	4372.95	4580.28	4375.35
Mejorada 3	3566.70	3888.65	3607.32
Mejorada 4	2696.25	2849.89	2484.30
% Mejora 1	13.98%	16.28%	20.85%
% Mejora 2	13.79%	8.27%	13.72%
% Mejora 3	18.25%	9.68%	17.17%
% Mejora 4	11.09%	7.00%	16.73%
Tiempo ejec	43.38	42.09	43.29

Tabla 8 Resultados medios de la instancia R201.50

	R202.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	2961.83	2961.83	2961.83
Inicial 2	4120.89	4120.89	4120.89
Inicial 3	3288.38	3288.38	3288.38
Inicial 4	2787.15	2787.15	2787.15
Mejorada 1	2478.43	2257.79	2381.78
Mejorada 2	3720.09	3730.58	2849.86
Mejorada 3	2832.65	2869.86	2501.77
Mejorada 4	2385.75	2191.96	2217.86
% Mejora 1	16.32%	23.77%	19.58%
% Mejora 2	13.53%	13.18%	42.91%
% Mejora 3	15.39%	14.13%	26.56%
% Mejora 4	13.55%	20.10%	19.22%
Tiempo ejec	43.75	43.56	44.18

Tabla 9 Resultados medios de la instancia R202.50

	R205.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	3201.02	3201.02	3201.02
Inicial 2	3692.48	3692.48	3692.48
Inicial 3	3366.50	3366.50	3366.50
Inicial 4	2710.15	2710.15	2710.15
Mejorada 1	2572.78	2489.17	2882.71
Mejorada 2	2874.54	2460.26	2235.87
Mejorada 3	2579.54	3085.05	2236.07
Mejorada 4	2386.98	2165.25	2184.01
% Mejora 1	19.63%	22.24%	9.94%
% Mejora 2	25.55%	38.49%	45.50%
% Mejora 3	24.58%	8.79%	35.31%
% Mejora 4	10.10%	17.02%	16.44%
Tiempo ejec	43.67	43.34	43.70

Tabla 10 Resultados medios de la instancia R205.50

	C201.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	3227.87	3227.87	3227.87
Inicial 2	3889.33	3889.33	3889.33
Inicial 3	3867.72	3867.72	3867.72
Inicial 4	1754.56	1754.56	1754.56
Mejorada 1	2208.47	2861.24	2442.95
Mejorada 2	3071.83	3455.87	2549.84
Mejorada 3	2229.13	3590.35	1871.55
Mejorada 4	1734.58	1580.67	1717.04
% Mejora 1	31.58%	11.36%	24.32%
% Mejora 2	25.33%	13.43%	41.50%
% Mejora 3	50.76%	8.59%	61.84%
% Mejora 4	0.62%	5.39%	1.16%
Tiempo ejec	43.40	41.87	44.45

Tabla 11 Resultados medios de la instancia C201.50

	C202.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	3155.24	3155.24	3155.24
Inicial 2	3030.95	3030.95	3030.95
Inicial 3	2919.50	2919.50	2919.50
Inicial 4	2461.33	2461.33	2461.33
Mejorada 1	2461.25	1775.34	1537.72
Mejorada 2	2058.41	1912.69	2101.22
Mejorada 3	1445.12	1457.05	1529.55
Mejorada 4	2149.20	1947.89	1939.25
% Mejora 1	21.99%	43.73%	51.26%
% Mejora 2	30.82%	35.44%	29.47%
% Mejora 3	46.73%	46.35%	44.05%
% Mejora 4	9.89%	16.27%	16.55%
Tiempo ejec	44.05	43.96	44.74

Tabla 12 Resultados medios de la instancia C202.50

	C203.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	2510.83	2510.83	2510.83
Inicial 2	2374.43	2374.43	2374.43
Inicial 3	2193.46	2193.46	2193.46
Inicial 4	2258.81	2258.81	2258.81
Mejorada 1	1950.03	1723.72	1783.51
Mejorada 2	1690.45	1751.68	1473.25
Mejorada 3	1233.92	1422.17	1318.95
Mejorada 4	1709.95	1522.92	1646.64
% Mejora 1	22.33%	31.35%	28.97%
% Mejora 2	27.24%	24.80%	35.89%
% Mejora 3	38.22%	30.72%	34.83%
% Mejora 4	21.86%	29.31%	24.38%
Tiempo ejec	44.42	45.10	45.28

Tabla 13 Resultados medios de la instancia C203.50

	RC201.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	3756.97	3756.97	3756.97
Inicial 2	4063.63	4063.63	4063.63
Inicial 3	3868.40	3868.40	3868.40
Inicial 4	2985.49	2985.49	2985.49
Mejorada 1	3104.46	3374.33	3082.57
Mejorada 2	3673.64	3705.00	3066.71
Mejorada 3	3347.65	3561.90	2708.30
Mejorada 4	2930.82	2903.09	2824.11
% Mejora 1	17.37%	10.18%	17.95%
% Mejora 2	10.38%	9.55%	26.54%
% Mejora 3	13.86%	8.16%	30.88%
% Mejora 4	1.46%	2.19%	4.30%
Tiempo ejec	44.38	42.79	43.53

Tabla 14 Resultados medios de la instancia RC201.50

	RC201.25		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	2237.63	2237.63	2237.63
Inicial 2	2239.04	2239.04	2239.04
Inicial 3	2020.57	2020.57	2020.57
Inicial 4	1364.01	1364.01	1364.01
Mejorada 1	1470.17	1674.48	1486.72
Mejorada 2	1924.36	1967.15	2056.46
Mejorada 3	1969.59	1798.68	1961.56
Mejorada 4	1161.24	1165.76	1170.65
% Mejora 1	34.30%	25.17%	33.56%
% Mejora 2	14.06%	12.15%	8.16%
% Mejora 3	2.28%	9.92%	2.64%
% Mejora 4	9.06%	8.86%	8.64%
Tiempo ejec	13.26	13.42	13.02

Tabla 15 Resultados medios de la instancia RC201.25

	RC202.25		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	1556.21	1556.21	1556.21
Inicial 2	1884.39	1884.39	1884.39
Inicial 3	1895.82	1895.82	1895.82
Inicial 4	1664.75	1664.75	1664.75
Mejorada 1	1014.02	1323.93	1017.19
Mejorada 2	1371.07	1471.69	1545.17
Mejorada 3	1505.79	1687.58	1143.14
Mejorada 4	1495.73	1309.24	1353.81
% Mejora 1	34.84%	14.93%	34.64%
% Mejora 2	32.99%	26.52%	21.80%
% Mejora 3	25.06%	13.38%	48.37%
% Mejora 4	10.86%	22.84%	19.98%
Tiempo ejec	13.43	13.46	13.37

Tabla 16 Resultados medios de la instancia RC202.25

	R101.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	4886.90	4886.90	4886.90
Inicial 2	5189.93	5189.93	5189.93
Inicial 3	5528.42	5528.42	5528.42
Inicial 4	4036.71	4036.71	4036.71
Mejorada 1	4455.30	4617.69	4655.33
Mejorada 2	5107.66	4925.60	5089.99
Mejorada 3	5331.06	5289.81	5285.50
Mejorada 4	3839.63	3808.39	3518.37
% Mejora 1	8.83%	5.51%	4.74%
% Mejora 2	1.68%	5.41%	2.05%
% Mejora 3	4.04%	4.88%	4.97%
% Mejora 4	4.03%	4.67%	10.61%
Tiempo ejec	40.62	40.35	41.94

Tabla 17 Resultados medios de la instancia R101.50

	R102.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	4093.94	4093.94	4093.94
Inicial 2	4330.44	4330.44	4330.44
Inicial 3	4710.70	4710.70	4710.70
Inicial 4	3361.53	3361.53	3361.53
Mejorada 1	3235.63	3778.20	3125.11
Mejorada 2	3893.71	3936.87	4022.62
Mejorada 3	4505.38	4393.20	4464.81
Mejorada 4	2796.29	2853.84	2736.56
% Mejora 1	20.97%	7.71%	23.66%
% Mejora 2	10.67%	9.61%	7.52%
% Mejora 3	5.02%	7.76%	6.01%
% Mejora 4	13.81%	12.40%	15.27%
Tiempo ejec	42.66	41.83	43.07

Tabla 18 Resultados medios de la instancia R102.50

	R103.50		
	selec eleccion 1	selec eleccion 2	selec eleccion rand
Inicial 1	3525.19	3525.19	3525.19
Inicial 2	3626.70	3626.70	3626.70
Inicial 3	3736.01	3736.01	3736.01
Inicial 4	2968.47	2968.47	2968.47
Mejorada 1	2398.20	2712.86	2507.38
Mejorada 2	3120.09	3193.98	3166.05
Mejorada 3	3478.02	3431.31	3481.16
Mejorada 4	2516.14	2389.88	2217.92
% Mejora 1	31.97%	23.04%	28.87%
% Mejora 2	14.37%	12.27%	13.07%
% Mejora 3	7.32%	8.64%	7.23%
% Mejora 4	12.83%	16.41%	21.29%
Tiempo ejec	43.62	42.76	43.06

Tabla 19 Resultados medios de la instancia R103.50

6.2 Resultados y conclusiones respecto a la población inicial de soluciones.

Como ya se explicó, la población inicial de soluciones se obtiene al variar el paso del algoritmo encargado de encontrar el primer cliente para añadir a la ruta. En la tabla "Inicial 1" corresponde al algoritmo cuyo primer cliente en la ruta es el más cercano al almacén, "Inicial 2" al cliente con la ventana temporal más estrecha, "Inicial 3" al cliente más lejano y, por último, "Inicial 4" al cliente con menor tiempo (fecha) de inicio de entrega.

A continuación, se observa una serie de gráficas en relación con la población inicial de soluciones y que ha sido obtenida de los resultados expuestos anteriormente. Para la obtención de las gráficas, se ha seguido el siguiente orden para representar las instancias en el "eje x": C101.50, C102.50, C103.50, RC101.50, RC102.50, RC103.50, R201.50, R202.50, R205.50, C201.50, C202.50, C203.50, RC201.50, RC201.25, RC202.25, R101.50, R102.50 y R103.50.

En la siguiente tabla, se representa el valor de la función objetivo de las soluciones iniciales generadas mediante métodos constructivos (recordemos que hay 4, en cada valor del eje x hay 4 valores representados en el eje y) para cada una de las 18 instancias (eje x), siendo C101.50 el valor x=1 y R103.50 el valor x=18.

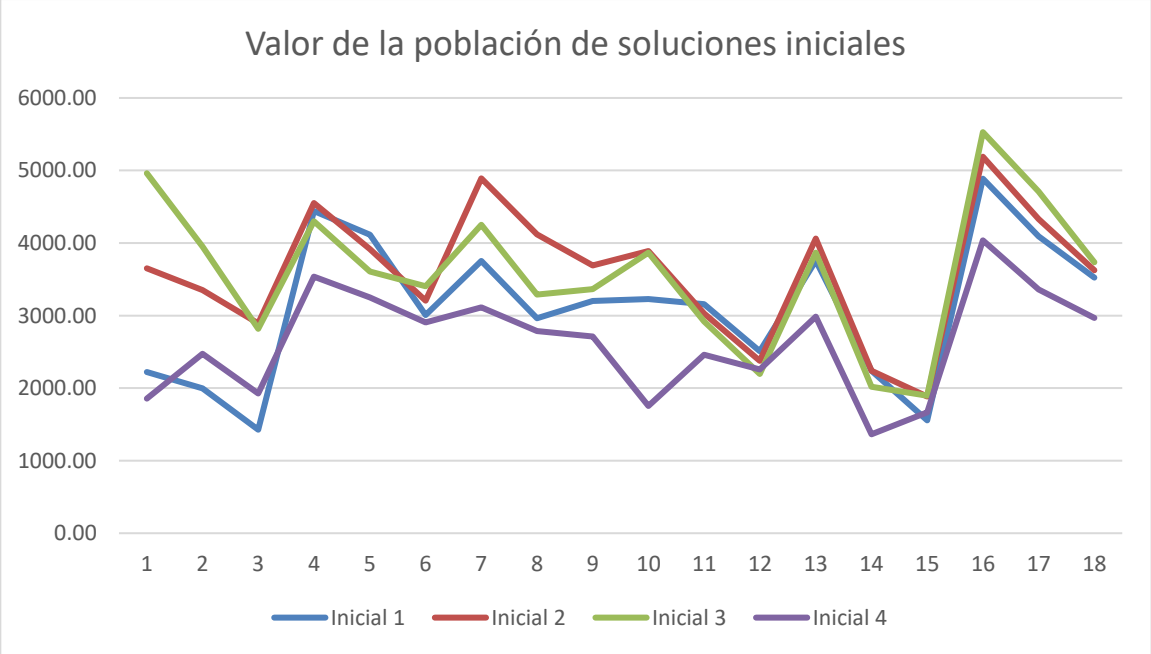


Ilustración 10 Gráfico del valor de la población inicial para cada instancia.

El siguiente gráfico muestra el número de veces en el "eje y" en el que una solución inicial ha quedado, siendo los puestos del 1º al 4º. Por ejemplo, la solución inicial obtenida a raíz del algoritmo 4 ha sido la mejor en 14 ocasiones tras finalizar el programa.

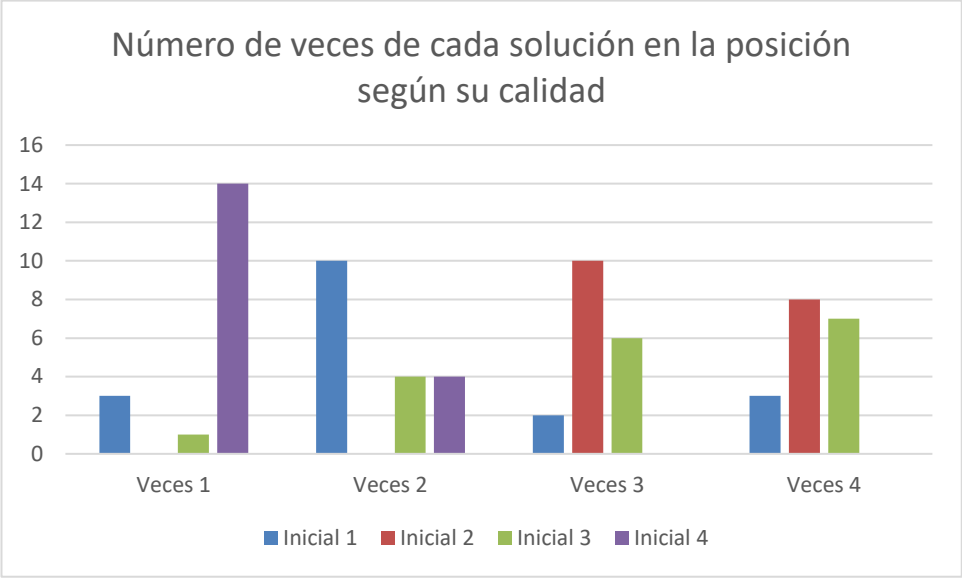


Ilustración 11 Gráfico del número de veces de la población inicial en un puesto.

Este gráfico muestra el número de veces en el "eje y" en el que una solución inicial que era la mejor/peor/otra ha terminado siendo la mejor al finalizar el programa. De este modo, sabemos que en 13 ocasiones la solución inicial que era la mejor al comienzo del programa ha terminado siendo la mejor.

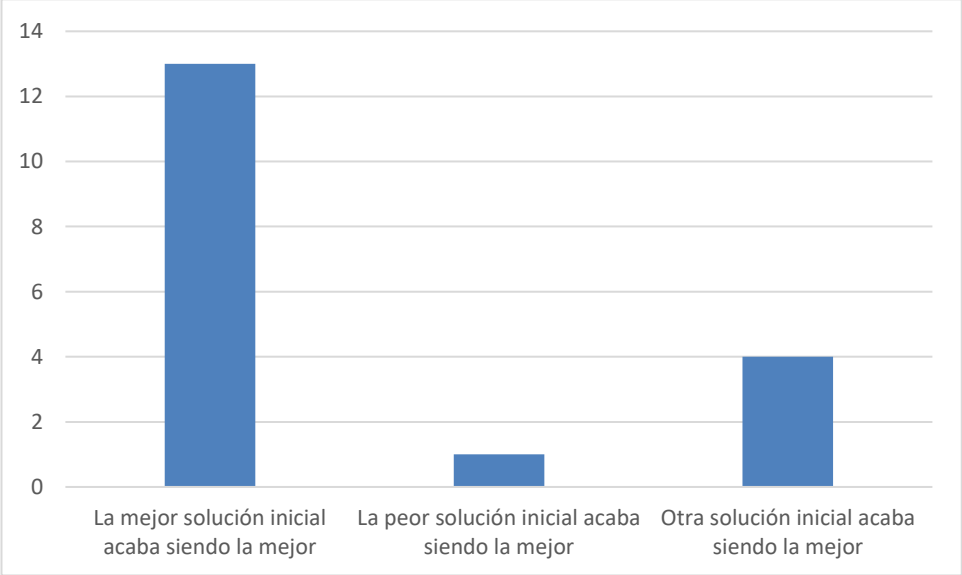


Ilustración 12 Gráfico que relaciona la solución inicial con la mejor solución final.

De estos gráficos podemos concluir que, de los métodos usados, el mejor para obtener una solución inicial corresponde al último algoritmo, cuyo criterio es seleccionar como primer cliente aquel con una fecha inicial de entrega menor. Por tanto, se puede decir que es más importante el poder entregar antes la mercancía que la cercanía al cliente. En cuanto al algoritmo que utiliza como criterio buscar al cliente más lejano, se puede concluir que no es una solución inicial buena.

Tras estas conclusiones sobre la solución inicial, no se debe pensar en trabajar con una única solución de partida. Aproximadamente un tercio de las soluciones finales han sido obtenidas de soluciones de partida que no eran las mejores, incluso en un caso se trataba de la peor. Esto demuestra que el algoritmo trata de no quedarse en un óptimo local.

6.3 Resultados y conclusiones del porcentaje de mejora.

En este apartado se exponen diferentes gráficos relacionados con los porcentajes de mejora de las soluciones. El objetivo es realizar un análisis del funcionamiento del programa en cuanto a la mejora de las soluciones.

Recordemos que el % Mejora x se calcula como la diferencia entre "Inicial x" y "Mejorada x", y todo ello dividido por "Inicial x".

Las ilustraciones que se encuentran a continuación representan en el "eje y" el porcentaje de mejora medio de cada criterio e instancia. De este modo, en el "eje x" se representa el resultado medio para los 3 criterios de cada instancia, siendo el número total de instancias de 3 por cada tipo (C1, RC1, etc.) haciendo un total de 9 puntos representados en este eje. Por tanto, para las tres instancias de cada tipo se tendrían los rangos 1-3 para la primera instancia, 4-6 para la segunda y 7-9 para la tercera.

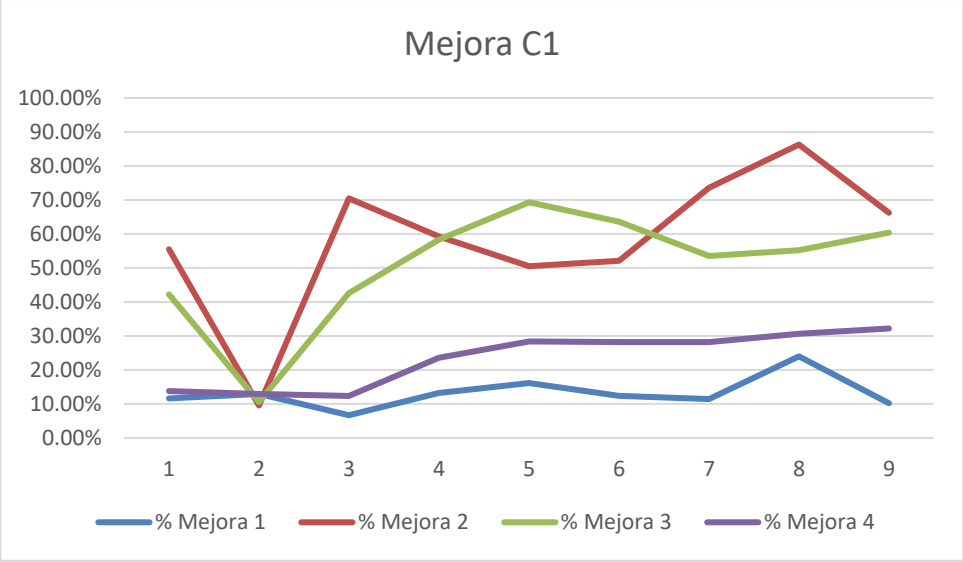


Ilustración 13 Porcentaje de mejora para las instancias de tipo C1

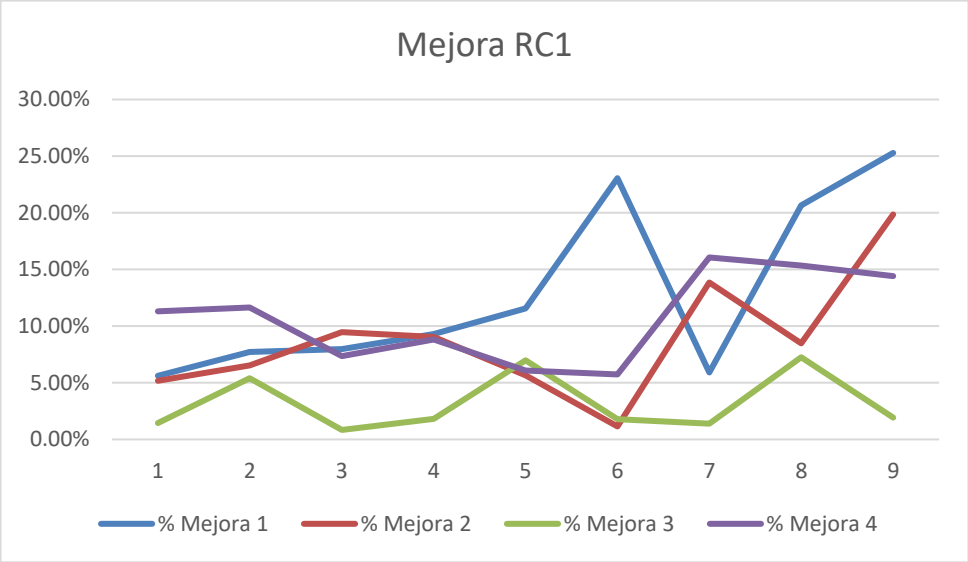


Ilustración 14 Porcentaje de mejora para las instancias de tipo RC1

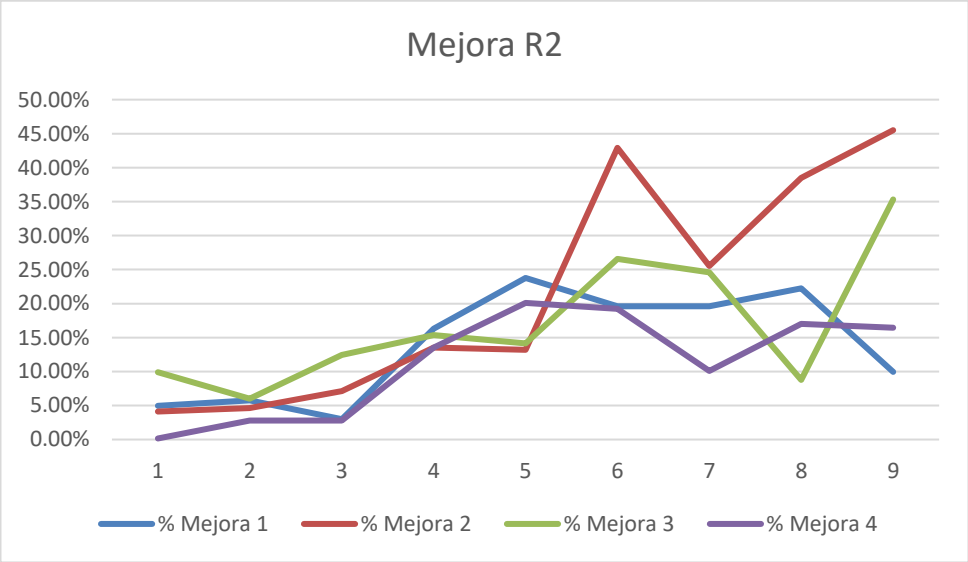


Ilustración 15 Porcentaje de mejora para las instancias de tipo R2

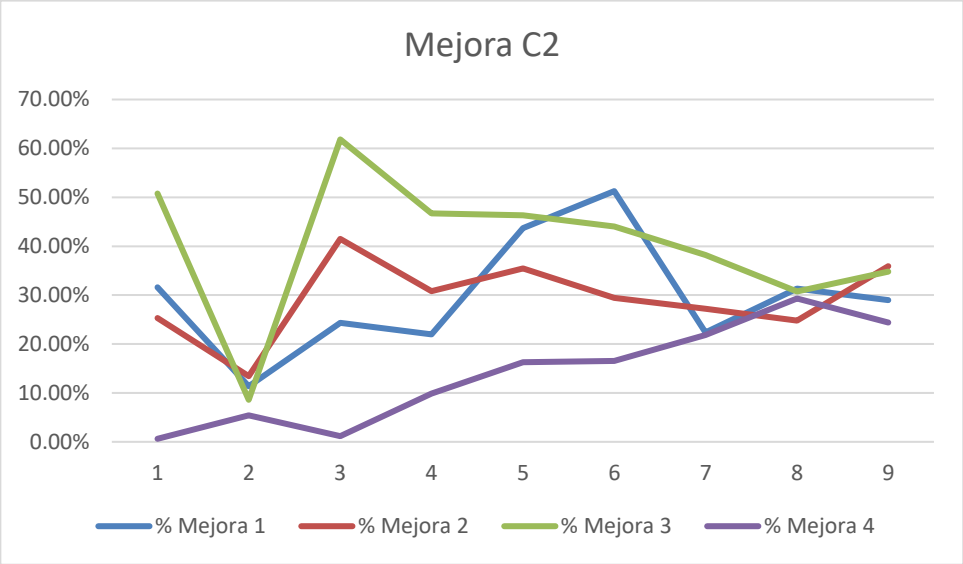


Ilustración 16 Porcentaje de mejora para las instancias de tipo C2

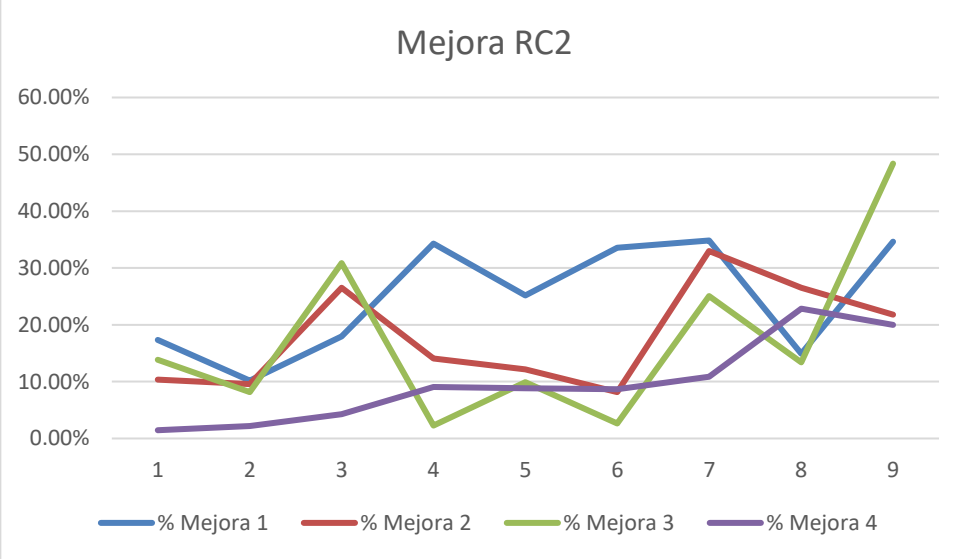


Ilustración 17 Porcentaje de mejora para las instancias de tipo RC2

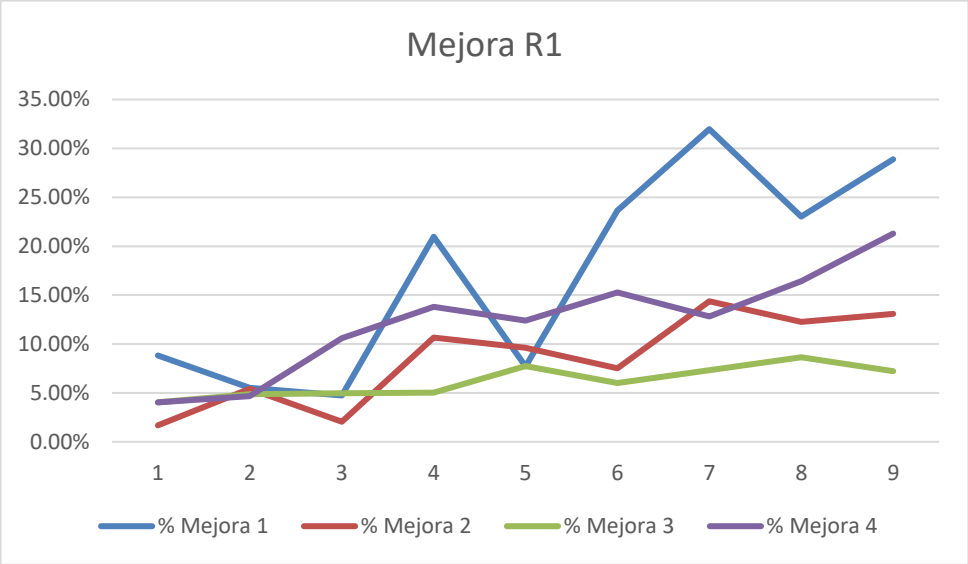


Ilustración 18 Porcentaje de mejora para las instancias de tipo R1

A continuación, se representa el porcentaje de mejora medio para cada solución inicial por tipo de instancia.

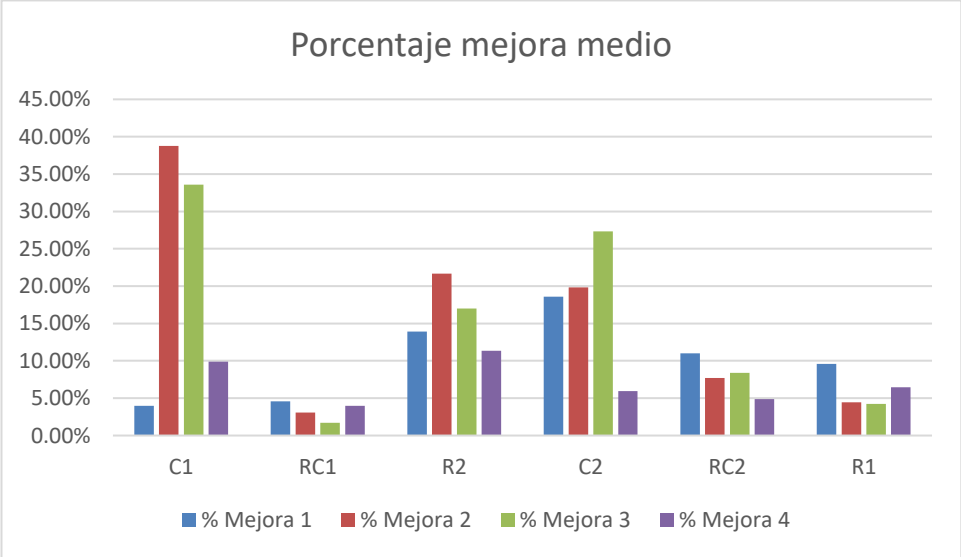


Ilustración 19 Porcentaje de mejora medio para cada instancia

En las siguientes ilustraciones se puede observar el porcentaje de mejora medio de cada solución según la instancia y el criterio (viene definido en el título del gráfico).

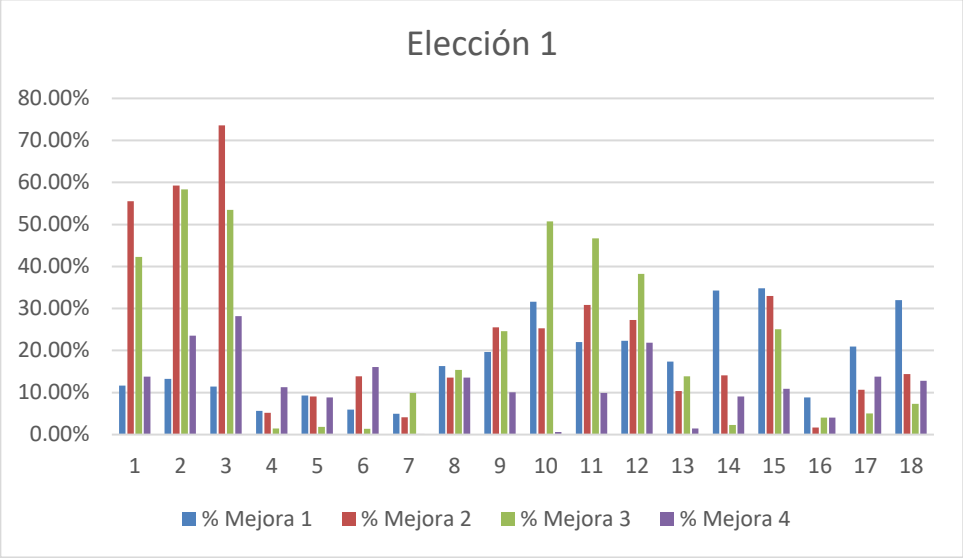


Ilustración 20 Porcentaje de mejora según el criterio Elección=1

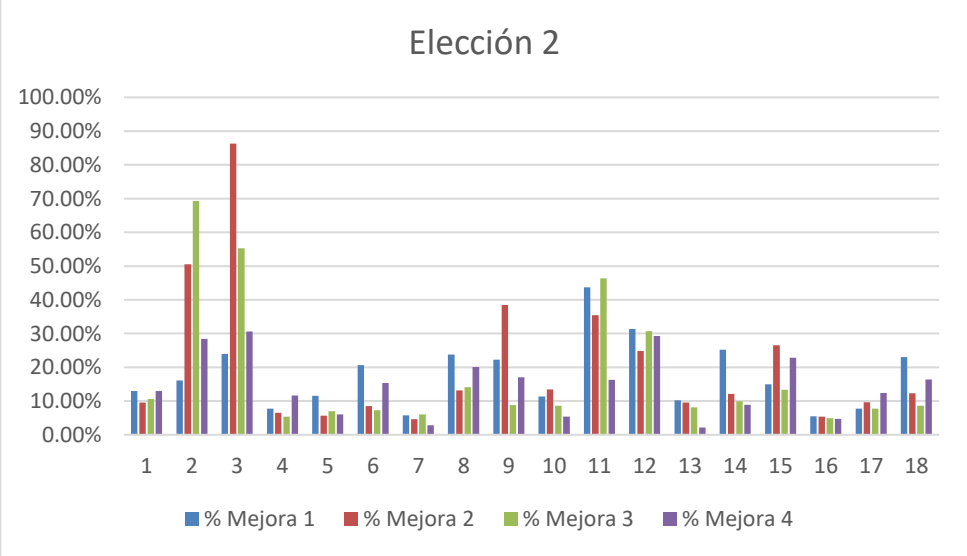


Ilustración 21 Porcentaje de mejora según el criterio Elección=2

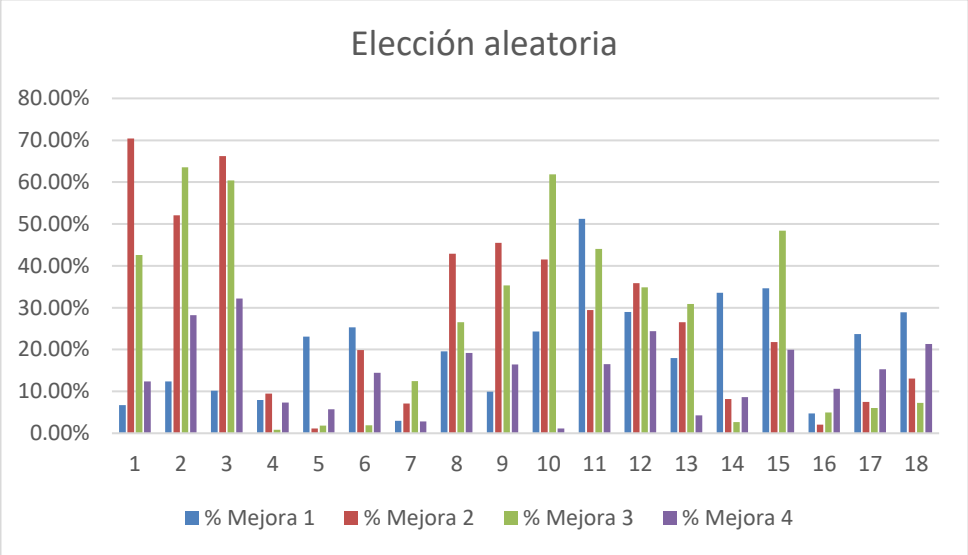


Ilustración 22 Porcentaje de mejora según el criterio Elección=aleatorio (1,3)

Gráfica que representa para cada solución inicial, la media de los porcentajes de mejora según el criterio escogido.

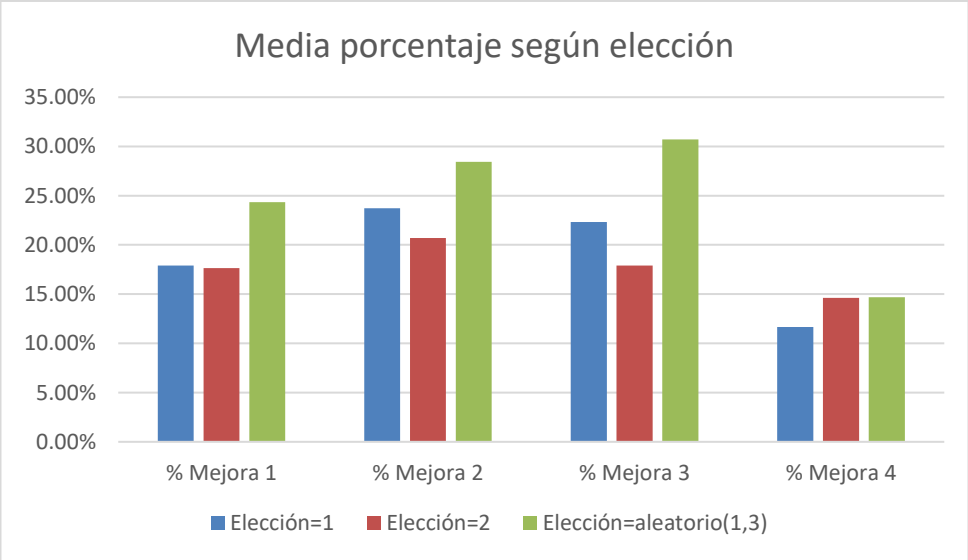


Ilustración 23 Porcentaje de mejora medio según el criterio de elección

Representación de la media de porcentaje de mejora para cada solución inicial según el tipo de instancia y criterio (el criterio viene definido en el título del gráfico).

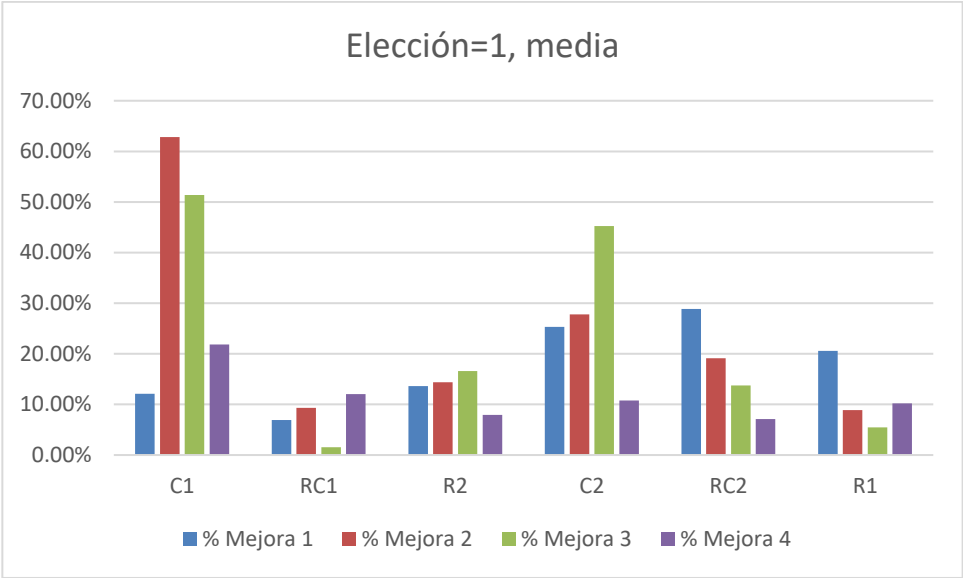


Ilustración 24 Porcentaje de mejora medio del tipo de instancia para el criterio Elección=1

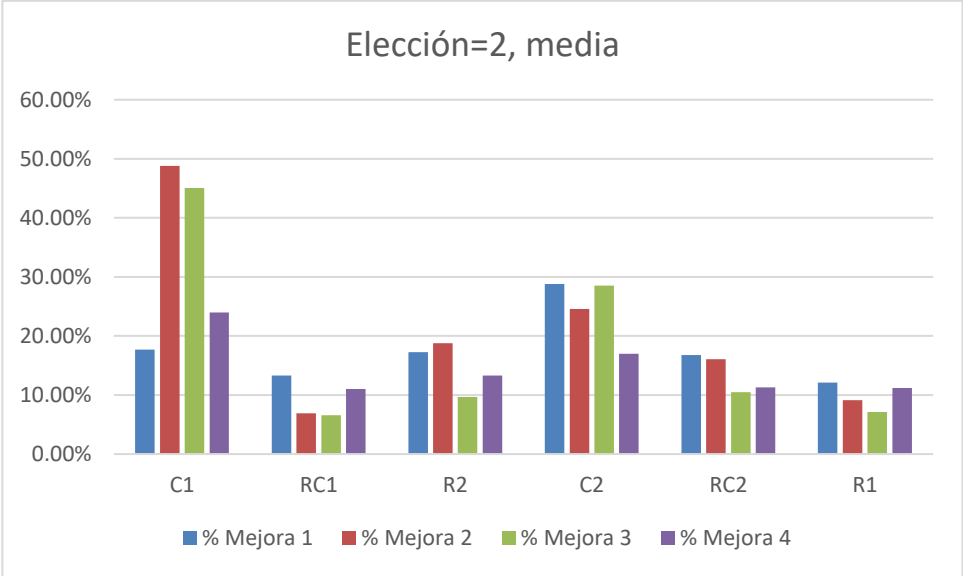


Ilustración 25 Porcentaje de mejora medio del tipo de instancia para el criterio Elección=2

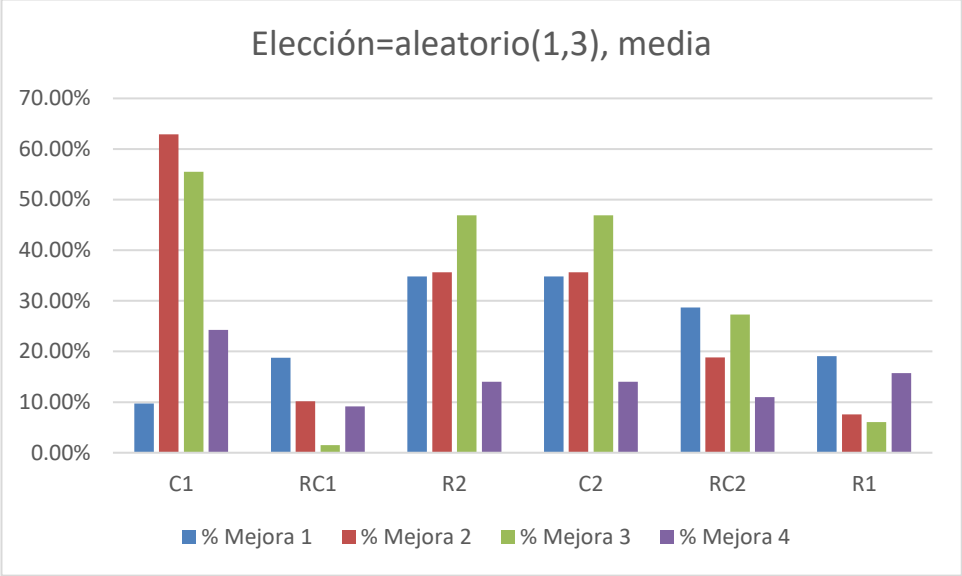


Ilustración 26 Porcentaje de mejora medio del tipo de instancia para el criterio Elección=aleatorio (1,3)

Se puede observar que en general, el porcentaje de mejora medio es mayor para los problemas de clientes agrupados, seguidos de los de tipo R2 y RC2. La mejora llega en un caso a ser casi del 40%, sin embargo, estas mejoras elevadas se dan para unas soluciones iniciales no tan buenas como la solución inicial cuatro. Para esta última solución (la cuatro), el porcentaje de mejora se mueve entre el 5% y el 10% aproximadamente, un porcentaje que podríamos decir que es un poco bajo, sobre todo si se parte de una solución inicial lejana a la óptima.

Por norma general, aunque los distintos criterios de selección de soluciones por los agentes optimizadores tienen resultados similares, tras la experimentación llevada a cabo, el mejor criterio es "elección=aleatorio (1,3)" para todos los tipos de problema excepto las instancias RC1.

En resumen, la mejor opción es elegir de manera aleatoria entre las siguientes opciones; escoger la mejor solución disponible (elección = 2), escoger la segunda peor solución (elección = 3) y escoger una totalmente aleatoria (elección = 1). Una posible explicación de estos resultados es que al incluir la posibilidad de que el agente Solution Manager se quede con la segunda peor solución, se pueden evitar óptimos locales y la solución pueda mejorar más. De los tres criterios tenidos en cuenta, el peor ha sido escoger siempre la mejor opción.

A continuación, a modo recordatorio se exponen los distintos criterios utilizados:

- Elección = 1: escoger de manera aleatoria una solución proporcionada por los agentes optimizadores.
- Elección =2: escoger la mejor solución (menor valor objetivo) proporcionada por los agentes optimizadores.
- Elección = aleatorio (1,3) de manera aleatoria se escoge:
  - Elección = 1.
  - Elección = 2.
  - Elección = 3: escoger la segunda peor solución proporcionada por los agentes optimizadores.

6.4 Resultados y conclusiones de la solución final.

La siguiente tabla es una comparativa entre los resultados encontrados con un método exacto (Relajación Lagrangiana), obtenidos del documento de [33], y los resultados obtenidos de la experimentación con un sistema multiagente. Hay que tener en cuenta que la función objetivo era distinta en [33] y, por tanto, se ha tenido que añadir un valor de 100 um por cada vehículo o ruta a los valores de dicho documento.

Instancias	Método exacto, solución encontrada	Tiempo ejec	Resultado multiagentes(res1)	Tiempo ejec	Resultado Multiagentes/Resultado Método exacto
C101.50	862.40	0.50	1551.32	43.02	1.80
C102.50	861.40	1.50	1674.99	43.32	1.94
C103.50	861.40	12.40	1084.69	44.59	1.26
RC101.50	1650.02	1.70	3021.22	42.59	1.83
RC102.50	1419.90	1029.80	2887.41	44.34	2.03
RC103.50	1243.13	17.10	2247.36	43.57	1.81
R201.50	1388.43	9.20	2484.30	43.29	1.79
R202.50A	1192.74	1707.70	2191.96	43.56	1.84
R205.50	1166.60	55507.50	2165.25	43.34	1.86
C201.50A	660.20	1.20	1580.67	41.87	2.39
C202.50A	660.20	18.80	1445.12	44.05	2.19
C203.50	659.80	167.10	1233.92	44.42	1.87
RC201.50	1170.15	61.80	2708.30	43.53	2.31
RC201.25	656.65	0.60	1161.24	13.26	1.77
RC202.25A	590.41	6351.70	1014.02	13.43	1.72
R101.50	2243.37	0.70	3518.37	41.94	1.57
R102.50	2009.00	3.50	2736.56	43.07	1.36
R103.50	1665.95	16.20	2217.92	43.06	1.33
Media					1.82

Tabla 20 Comparación de las soluciones obtenidas con las óptimas encontradas.

En las siguientes ilustraciones se puede observar la evolución para la instancia C103.50. El círculo rojo corresponde al almacén, mientras que los amarillos representan los clientes. En azul se pueden observar los primeros clientes de cada ruta. Las líneas azules simulan la ruta que sigue el vehículo.

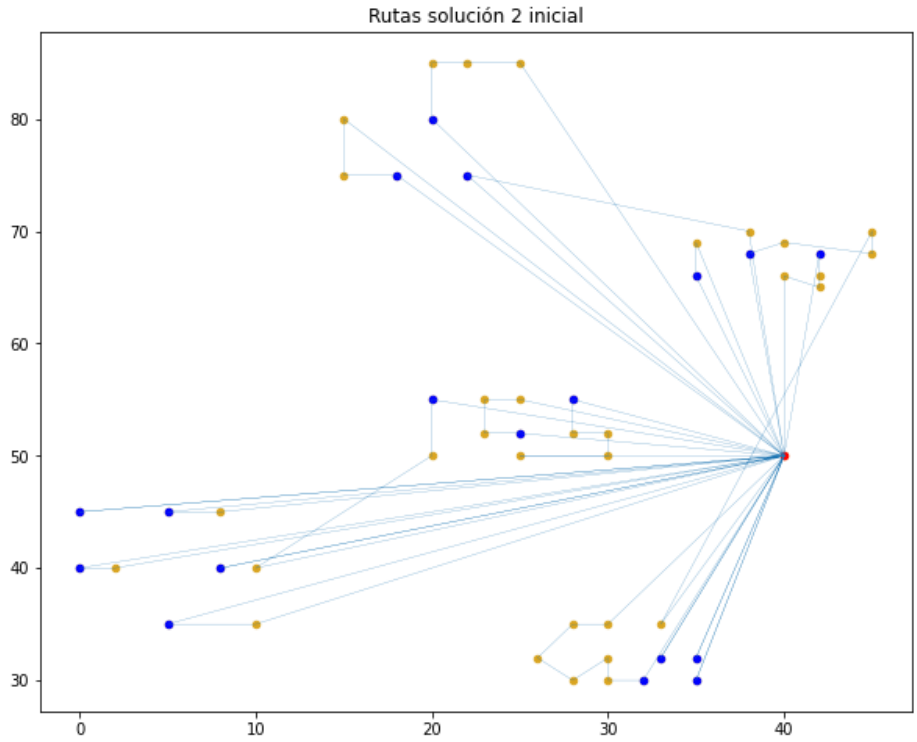


Ilustración 27 Representación gráfica de la solución inicial 2 para la instancia C103.50

Inicialmente esta solución tiene 18 rutas como se puede observar en la ilustración.

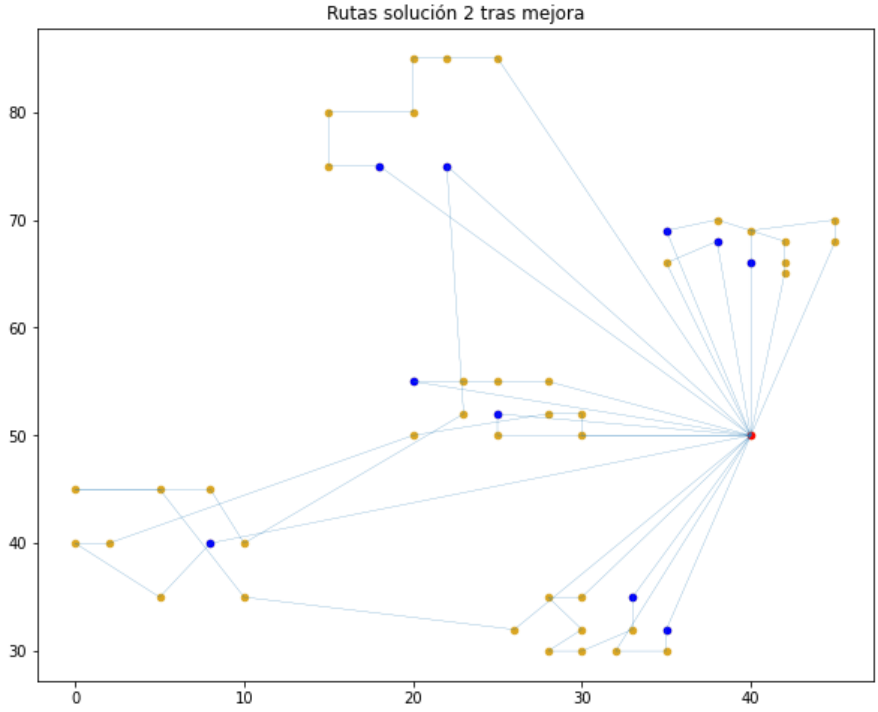


Ilustración 28 Representación gráfica de la solución final 2 para la instancia C103.50

Tras la mejora, el número de rutas ha sido disminuido de 18 a 10 rutas, lo que hace un total de 8 rutas ahorradas. Además de este ahorro, se pueden percebir ciertos cambios en las rutas, clientes pertenecientes a una ruta ahora se encuentran en otra. Este ejemplo muestra cómo el sistema de multiagentes cumple su función. Gracias a los distintos agentes pueden eliminarse rutas, intercambiar clientes de rutas diferentes e incluso cambiar el orden de los clientes de una misma ruta, opciones que no se pueden dar al utilizar un único algoritmo o agente.

En este caso, una solución encontrada en una experimentación para la instancia C103.50 ha sido la siguiente:

Valor de la función objetivo: 1038.73 um

Número de rutas: 6

Tiempo de ejecución: 745.54 segundos

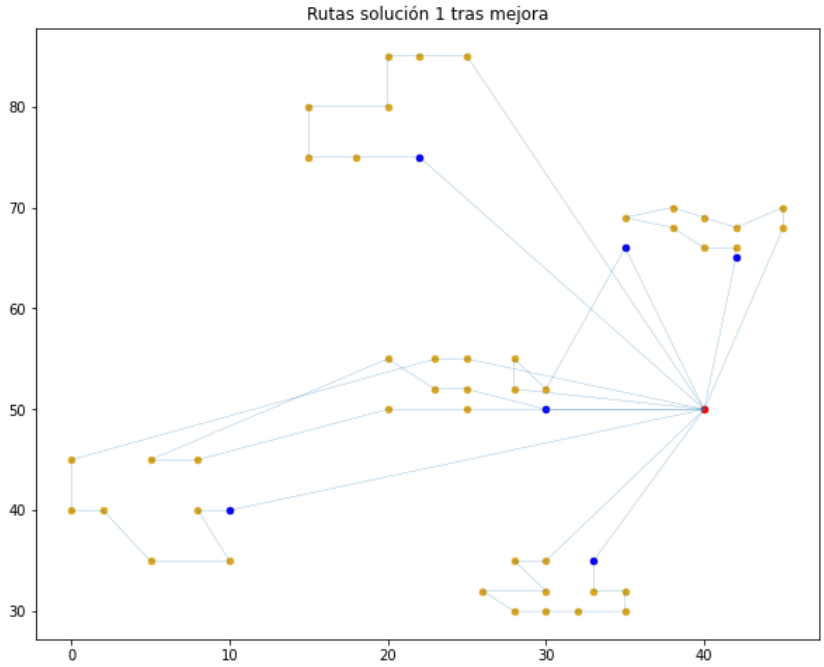


Ilustración 29 Representación gráfica de la solución final 1 para la instancia C103.50

Los resultados de la tabla 20 nos indican que el sistema de multiagentes nos proporciona una solución que aproximadamente duplica el coste de la solución óptima, además el tiempo de ejecución es en la mayoría de los casos superior. Sin embargo, para los problemas de tipo R1 y en la instancia C103.50 se observa que la calidad de las soluciones es bastante buena y se aproxima al óptimo.

El sistema de multiagentes requiere de un tiempo elevado, pero teniendo en cuenta de que los algoritmos utilizados como agentes no son algoritmos eficaces, son algoritmos que hemos realizado a partir de los conocimientos adquiridos en la carrera y en la realización del trabajo, se puede decir que el programa funciona correctamente y cumple sus funciones, c.q.d.

Además de esto, hay que tener en cuenta que el problema VRPTW es un problema difícil de resolver y que, para instancias con un número de clientes elevado, utilizar métodos como el empleado en [33] no proporciona una buena solución o directamente no resuelven el problema (se ha tenido que sustituir algunas instancias por otras al no disponer de la solución óptima). Sin embargo, el sistema de multiagentes es capaz de resolver ese tipo de instancias ya que se emplean heurísticas.

Finalmente, se puede decir que al menos hasta 50 clientes no es necesario utilizar un sistema multiagente. Sin embargo, para las instancias que no son tipo clustered comienza a ser necesario el empleo de al menos una

heurística o metaheurística, pues se puede observar en [33] que hay instancias que no están resueltas y otras en las que se emplea un tiempo elevado con los métodos exactos. También hay que destacar que los algoritmos utilizados son algoritmos que se han creado en base a otros que sí funcionan mejor, pero que han sido modificados para la realización del programa.

6.5 Propuestas de mejora para el sistema multiagente.

Para mejorar el funcionamiento del sistema multiagente se proponen los siguientes cambios:

- OA Vecinos, no varía.

-OA Insertion, añadir que, si no encuentra el algoritmo una solución, genere un número aleatorio nuevo hasta recorrer todos los clientes de la ruta.

-OA String Cross, sustituir por un algoritmo de búsqueda tabú similar.

-OA Intercambio, sustituir por un algoritmo de búsqueda tabú similar.

-OA Buscar solitario, sustituir por un algoritmo que, de una ruta aleatoria compruebe cliente por cliente de esa ruta si hay un cliente cercano (definir una distancia) perteneciente a otra ruta y tratar de insertarlo antes o después (comprobar cuál de las dos opciones sería mejor).

Después de realizar estos cambios, el tiempo de ejecución de cada agente aumentaría porque trataríamos con agentes más complejos. Sin embargo, el sistema multiagente podría mejorar y sin duda, sería mejor opción que utilizar un único algoritmo ya que el abanico de posibilidades que ofrece es mucho mayor. El sistema de multiagentes permite el intercambio de clientes entre diferentes rutas y el de una misma, además permite no caer en un óptimo local gracias a la comunicación entre los agentes optimizadores y Solution Manager (podemos con criterios hacer que se contemple la posibilidad de escoger una solución peor y así evitar óptimos locales, además de disponer de una población de soluciones inicial).

Otro cambio para realizar sería la sustitución del algoritmo de la solución inicial 3 por un algoritmo nuevo. Este algoritmo propone utilizar el algoritmo de la población inicial cuatro, pero en vez de buscar el cliente más cercano una vez añadido un primer cliente a la ruta, propone introducir en la ruta al cliente cuyo valor de la ecuación $distancia + \frac{fecha\ de\ inicio\ de\ la\ entrrega}{1000}$ sea menor. Este nuevo algoritmo no se usará en el programa, pero se ha hecho una prueba para ver si la solución inicial es mejor que el resto y se ha observado que en trece ocasiones ha sido así. Aunque no se utilice en el programa, si se probará en las instancias C201.50, C202.50 y RC201.50 para ver si se consigue una mejora respecto al resultado final, estos resultados se obtendrán siguiendo el criterio "elección=aleatorio (1,3)". Con las tablas 21 y 22 se concluye que, para obtener una solución inicial, un método que pondere la distancia de un cliente a otro junto con la fecha inicial de entrega (el mínimo de la ventana temporal) es mejor en la mayoría de las ocasiones que métodos en los que se excluya a uno de los dos (distancia y tiempo).

	C101.50	C102.50	C103.50	RC101.50	RC102.50	RC103.50	R201.50	R202.50ª	R205.50
Inicial 1	2220.90	1997.34	1427.08	4440.99	4116.95	3008.03	3754.13	2961.83	3201.02
Inicial 2	3650.40	3353.34	2890.85	4552.80	3914.72	3204.26	4890.56	4120.89	3692.4845
Inicial nuevo algoritmo	1695.11	1900.14	2353.68	3775.99	3245.76	2711.01	2831.09	2640.00	2529.62
Inicial 4	1857.55	2474.18	1925.50	3538.66	3250.66	2905.55	3112.53	2787.14	2710.14

Tabla 21 Soluciones iniciales con nuevo algoritmo, parte 1

	C201.50A	C202.50A	C203.50	RC201.50	RC201.25	RC202.25A	R101.50	R102.50	R103.50
Inicial 1	3227.87	3155.24	2510.82	3756.96	2237.63	1556.21	4886.90	4093.94	3525.19
Inicial 2	3889.33	3030.95	2374.42	4063.63	2239.04	1884.39	5189.93	4330.43	3626.70
Inicial nuevo algoritmo	1612.65	2005.30	2137.29	3335.62	1200.69	1664.56	3901.60	3324.69	3231.53
Inicial 4	1754.561309	2461.328499	2258.81	2985.49	1364.01	1664.75	4036.71	3361.53	2968.47

Tabla 22 Soluciones iniciales con nuevo algoritmo, parte 2

Instancias	Mejor resultado	Tiempo ejec	Resultado multiagentes algoritmo 3 (res1)	Resultado Multiagentes/Resultado Método exacto	Resultado multiagentes nuevo algoritmo	Resultado Multiagentes/Resultado Método exacto	% Mejora sobre res1
C201.50A eleccion=aleatorio	660.20	1.20	1871.55	2.60	1449.83	2.20	22.53%
C202.50A eleccion=aleatorio	660.20	18.80	1529.55	2.32	1499.25	2.27	1.98%
RC201.50 eleccion=aleatorio	1,170.15	61.80	2708.30	2.31	3169.62	2.71	-17.03%

Tabla 23 Comparación resultados del algoritmo 3 con el nuevo algoritmo, todos con el criterio "elección=aleatorio (1,3)"

Instancias	Mejor resultado	Tiempo ejec	Mejor Resultado multiagentes (res1)	Resultado Multiagentes/Resultado Método exacto	Resultado multiagentes nuevo algoritmo	Resultado Multiagentes/Resultado Método exacto	% Mejora sobre res1
C201.50A eleccion	660.20	1.20	1580.67	2.39	1449.83	2.20	8.28%
C202.50A	660.20	18.80	1445.12	2.19	1499.25	2.27	-3.75%
RC201.50	1,170.15	61.80	2708.30	2.31	3169.62	2.71	-17.03%

Tabla 24 Comparación mejores resultados encontrados con el nuevo algoritmo, este último con el criterio "elección=aleatorio (1,3)"

Tras los resultados obtenidos de la tabla 23 y 24 se puede concluir que, conseguir una mejor solución inicial no implica obtener mejores resultados finales, al menos en las instancias en las que se ha hecho la comprobación. Por tanto, se propone para un futuro trabajo, realizar las mejoras de los agentes optimizadores mencionados anteriormente para comprobar si mejoran los resultados. Tras realizar los cambios mencionados, se podría probar de nuevo con el algoritmo propuesto para obtener una solución inicial.

7 CONCLUSIONES GENERALES

El primer paso para comenzar con la realización del sistema multiagente es conseguir una solución inicial. Para ello se han utilizado métodos constructivos con la restricción de número de vehículos relajado, es decir, no se ha puesto límite al número de vehículos o rutas de la solución. Tras conseguir los resultados para distintos criterios, se decidió probar a utilizar un criterio que incluyera la distancia y el tiempo de entrega de manera ponderada (se ha comprobado que el programa no mejora con este único cambio) con el cual se ha podido concluir que en el VRPTW es importante tener en cuenta ambas partes (distancia y tiempo).

El segundo paso es seleccionar los agentes que se van a utilizar en el sistema. Por un lado, están los agentes encargados de optimizar las soluciones y por otro, el agente encargado de proporcionar y extraer soluciones al resto. La elección de los agentes es muy importante y se ha comprobado que en el trabajo estos agentes hay que cambiarlos si se quieren obtener unos mejores resultados. En cuanto al intercambio de información, se ha demostrado que el peor criterio a seguir de los escogidos es el de seleccionar siempre la mejor solución que proporcionan los agentes, pues no evita óptimos locales.

Por último, una vez realizado el programa, hay que decidir el criterio de parada de este. En este trabajo se ha decidido realizar 300 interacciones entre el agente SolutionManager y los agentes optimizadores, esto equivale a unos 42 segundos aproximadamente de tiempo de ejecución para instancias de 50 clientes. Sin embargo, este criterio podría cambiarse por otro sin ningún problema, por ejemplo, que el valor de la última columna de la tabla 22 (Resultado multiagentes/Resultado método exacto) fuera menor o igual a cierto valor además de una condición que controle el tiempo por si nunca se llega a dicho objetivo (hay que tener en cuenta que para la realización del trabajo se han hecho 1620 experimentaciones y, por tanto, se dispone de poco tiempo para cada experimentación).

Con esto, queda diseñado un sistema multiagente para abordar el problema de rutado de vehículos con ventanas temporales con un almacén y cualquier cantidad de clientes. Este sistema se ha programado en lenguaje Python y se han realizado diversos experimentos con instancias de hasta 50 clientes. Se han propuesto además algunas mejoras sobre el algoritmo original.

REFERENCIAS

[1] BBVA, «The COVID-19 impact on Consumption in Real Time and High Definition A Big Data BBVA Research Project,» 2020.

[2] D. L. Applegate, R. E. Bixby, V. Chvátal y W. J. Cook, The Traveling Salesman Problem, 2006, p. 12.

[3] G.B.Dantzig y J.H. Ramser, «The truck dispatching problem,» *Managment Science,* vol. 6, nº 1.

[4] B. Golden, S. Raghavan y E. Wasil, de *The Vehicle Routing Problelm: Latest Advances and New Challengues*, Springer, 2008, pp. 3-6.

[5] R. Cordone y R. W. Calvo, «A Heuristic for the Vehicle Routing Problem,» *Journal of Heuristics,* pp. 107-129, 2001.

[6] J. M. Wooldridge, An introduction to multiagent systems, 2 ed., John Wiley & Sons Ltd, 2009.

[7] D. Barbucha y P. Jedrzejowicz, An Agent-Based Approach to Vehicle Routing, vol. 1, 2007.

[8] D. Barbucha, I. Czarnowski, P. Jedrzejowicz, E. Ratajczak y I. Wierzbowska, JADE-based A-Team as a tool for implementig population-based algorithms, vol. 3, 2006.

[9] L. B. Rocha, E. C. González y J. A. Orjuela, Una revisión al estado del arte del problema de ruteo de vehículos: Evolución histórica y métodos de solución., vol. 16, 2011, pp. 35-55.

[10] E. M. Orozco, *Metaheurísticas para el problema de ruteo de vehículos con ventanas de tiempo,* 2017.

[11] R. Marti, *Procedimientos Metaheurísticos en Optimización Combinatoria.*

[12] M. M. Solomon, «Algorithms for the Vehicle Routing and Scheduling Problems with Time Windows Constraints,» 1984.

[13] G. Clarke y J. Wright, «Scheduling of vehicles from a central depot to a number of delivery,» *Operations Research,* vol. 12, nº 4, p. 568–581, 1964.

[14] S.Lin y B. W. Kernighan, «An Effective Heuristic Algorithm for the Traveling-Salesman Problem,» *Operations Research,* vol. 21, nº 2, pp. 498-516, 1973.

[15] I. Or, *Traveling Salesman-Type Combinatorial Problems and their relation,* 1976.

[16] M. Gendreau, A. Hertz y G. Laporte, «New Insertion and Postoptimization Procedures for the Traveling Salesman Problem,» *Operations Research;,* vol. 40, nº 6, p. 1086, 1992.

[17] P. M. Thompson y H. N. Psaraftis, «Cyclic Transfer Algorithms for Multivehicle Routing and Scheduling

Problems,» *Operations Research,* vol. 41, nº 5, pp. 935-946, 1993.

[18] A. Olivera, *Heurísticas para Problemas de Ruteo de Vehículos,* Montevideo, 2004.

[19] V. Breedam, «Improvement heuristics for the vehicle routing problem,» *European Journal of Operational Research,* vol. 86, nº 3, pp. 480-490, 1995.

[20] U. P. d. Valencia. [En línea]. Available: https://optimizacionheuristica.blogs.upv.es/2015/02/22/que-son-las-metaheuristicas/.

[21] I. Osman y J. Kelly, «Meta-Heuristics: Theory & Applications,» Kluwer Academic Publishers, 1996.

[22] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, 1989.

[23] S. Kirkpatrick, C. D. Gelatt y M. P. Vecchi, «Optimization by Simulated Annealing,» vol. 280, pp. 671-680, 1983.

[24] M. Dorigo, *Optimization, Learning and Natural Algorithms,* Ph.D. Thesis, Politecnico di Milano, Italian, 1992.

[25] F. Glover, «Future paths for integer programming and links to artificial intelligence,» *Computers & Operations Research,* vol. 13, nº 5, pp. 533-549, 1986.

[26] G. v. Rossum, Interviewee, *The Making of Python.* [Entrevista]. 2003.

[27] «Spade,» [En línea]. Available: https://spade-mas.readthedocs.io/en/latest/readme.html.

[28] «Asyncio,» [En línea]. Available: https://docs.python.org/3/library/asyncio.html.

[29] «Aima Python Code,» [En línea]. Available: http://aima.cs.berkeley.edu/python/readme.html.

[30] «Mesa: Agent-based modeling in Python 3+,» [En línea]. Available: https://mesa.readthedocs.io/en/master/.

[31] «VRP-REP,» [En línea]. Available: http://www.vrp-rep.org/variants/item/vrptw.html.

[32] O. Bräysy y M. Gendreau, «Vehicle Routing Problem with Time Windows,,» *TRANSPORTATION SCIENCE,* vol. 39, nº 1, pp. 104-118, 2005.

[33] B. Kallehauge, J. Larsen y O. Madsen, «Lagrangean Duality Applied on Vehicle Routing with Time Windows Experimental Results,» 2001.