Solución del problema de ruteo de vehículos capacitado (CVRP) usando el algoritmo de aprendizaje reforzado Q-Learning

Pedro Elkin Rivera Jaimes, Orlando Stiven Jaramillo Piza

Trabajo de Grado para Optar el título de Ingeniero Industrial

Director
Daniel Orlando Martínez Quezada
MSc. Ingeniería Industrial

Codirector
Henry Lamos Díaz
PhD. Física-Matemática

Universidad Industrial de Santander
Facultad de Ingenierías Físico-Mecánicas
Escuela de Estudios Industriales y Empresariales
Bucaramanga
2020

Dedicatorias

Especialmente a mi madre Clara Inés que se ha esmerado a lo largo de su vida por darnos lo mejor a mi hermano y a mí; a mi hermano Carlos Andrés por su rol de padre y ejemplo a seguir durante mi formación académica.

A mi novia Adriana Cristina por su compañía en los momentos de ansiedad durante el proyecto y enseñarme que puedo hacer grandes cosas manteniendo la calma.

Al grupo de música y danzas afrocolombianas Macondo UIS y a su director Nicolás Maestre por abrirme sus puertas y formarme como profesional integral.

Pedro Elkin Rivera Jaimes

A mi madre Ana Stella y a mi padre Jesús Orlando por sus enseñanzas, consejos y apoyo a lo largo de mi academia; a mis hermanos Tatiana y Andrés por ser pilares de motivación en mi formación.

A mi novia Maira por su acompañamiento y apoyo durante la realización del proyecto.

Orlando Stiven Jaramillo Piza

Agradecimientos

Al profesor Daniel Martínez por brindarnos su confianza, esmero y dedicación para el desarrollo satisfactorio del proyecto. Ya que, sin su guía este proyecto no hubiera sido posible. Y al grupo de investigación OPALO, gracias.

Tabla de Contenido

Lista de Tablas

Lista de Figuras

Lista de Apéndices

Los apéndices están adjuntos en el CD y puede visualizarlos en base de datos de la biblioteca UIS:

Apéndice A. Análisis bibliométrico

Apéndice B. Artículo académico

RESUMEN

TÍTULO: SOLUCIÓN DEL PROBLEMA DE RUTEO DE VEHÍCULOS CAPACITADO (CVRP) USANDO EL ALGORITMO DE APRENDIZAJE REFORZADO Q-LEARNING. *

AUTORES: Orlando Stiven Jaramillo Piza, Pedro Elkin Rivera Jaimes **

PALABRAS CLAVE:
Optimización Combinatoria, Problema de Ruteo de Vehículos Capacitado, CVRP, Aprendizaje Reforzado, Q-Learning.

DESCRIPCIÓN:
El propósito de la siguiente investigación es dar solución a instancias de la literatura para el problema de ruteo de vehículos capacitado (CVRP) según formulación VRP4 de Toth & Vigo (2002) por medio de un algoritmo de aprendizaje reforzado conocido como Q-Learning. El desarrollo del algoritmo se realizó en Python y utiliza parámetros como tasa de aprendizaje, factor de descuento y factor de descomposición, cuya incidencia en la función objetivo del CVRP es evaluada a través de un diseño factorial 2^k.

Se realizó en una revisión de literatura donde se encontró que el VRP ha sido ampliamente estudiado desde 1959 a través métodos exactos, heurísticos y metaheurísticos, que ofrecen solución incluso a variantes con demanda estocástica haciendo uso de redes neuronales. Se evaluó la incidencia de cada parámetro en la minimización de la función objetivo, así como el desempeño del algoritmo haciendo uso de instancias de la literatura ya solucionadas de 55, 80 y 200 nodos tomadas del Set A (Augerat,1995) y el Set M (Christofides, Mingozzi and Toth, 1979) y se obtuvo como resultado que el algoritmo propuesto brinda soluciones cercanas al óptimo a instancias de mediano tamaño, y en intervalos cortos de tiempo, logrando una relación de al menos 1.27 para la solución propuesta sobre la solución óptima.

---

* Trabajo de grado
** Facultad de Ingenierías Fisicomecánicas. Escuela de Estudios Industriales y Empresariales. Director MSc. Daniel Orlando Martínez Quezada, Codirector: PhD. Henry Lamos Díaz

ABSTRACT

TITLE: SOLUTION TO CAPACITATED VEHICLE ROUTING PROBLEM (CVRP) USING THE REINFORCEMENT LEARNING ALGORITHM Q-LEARNING. *

AUTHORS: Orlando Stiven Jaramillo Piza, Pedro Elkin Rivera Jaimes **

KEYWORDS:

Combinatorial Optimization, Capacitated Vehicle Routing Problem, CVRP, Reinforcement Learning, Q-Learning.

DESCRIPTION:

This research has as purpose give a solution to capacitated vehicle routing problem (CVRP) literature instances following the VRP4 model formulation by Toth & Vigo (2002) through a reinforcement learning algorithm known as Q-Learning. The algorithm script was written and developed in Python and uses parameters like learning rate, discount factor and decay factor, whose effect in the CVRP objective function is assessed by a 2^k factorial design.

The literature review showed VRP has been widely approached since 1959 through exact, heuristic and metaheuristic methods, offering solutions even for stochastic demand variants utilizing neural networks. Also, the incidence in objective function for each parameter was evaluated, as well as the algorithm's performance using instances already solved of 55, 80 and 200 nodes taken from Set A (Augerat,1995) y el Set M (Christofides, Mingozzi and Toth, 1979). As a result, the applied algorithm for medium size instances, is capable of bring solutions near an optimum, achieving a relation of at least 1.27 for a proposed solution over and optimum solution.

* Bachelor Thesis

** Faculty of Physicomechanical Engineering. Industrial and Business School. Director MSc. Daniel Orlando Martínez Quezada, Co-director: PhD. Henry Lamos Díaz

Introducción

Con el fin de enfrentar los cambios en los costos, las organizaciones se encuentran en constante búsqueda de estrategias que les permitan solucionar problemas en sus procesos logísticos, reflejados en las variaciones del precio de la gasolina, el alza en los fletes, en los impuestos vehiculares, entre otros, que se traducen en el aumento de los costos, que pueden llegar a representar entre un 10% y un 20% del costo total de transporte de mercancía (L. S. Lee, Majid, Seow, & Broga, 2012).

Las investigaciones teóricas y las aplicaciones en el campo del ruteo de vehículos comenzaron en 1959 con el "truck dispatching problem" publicado por G. B. Dantzig & Ramser (1959), que después fue conocido como el Capacitated Vehicle Routing Problem (CVRP), el cual tenía como objetivo encontrar el ruteo óptimo de una flota de camiones de reparto de gasolina entre un terminal de graneles y varias estaciones abastecidas por dicho terminal. En casi sesenta años este campo se ha explotado y ha generado variantes basadas en el CVRP, lo que ha contribuido a que ésta sea la variante más estudiada y uno de los campos más fuertes en investigación de operaciones. Así, sin lugar a duda, las aplicaciones del problema de ruteo de vehículos con restricciones de capacidad (CVRP) han generado la necesidad de desarrollar técnicas y explorar nuevos métodos a fin de obtener soluciones factibles u óptimas que permitan a las empresas competir en el mercado.

De este modo, los primeros intentos de solución para este tipo de problemas con complejidad NP-Hard como el CVRP, fueron realizados a través de métodos exactos con tiempos computacionales altos para alcanzar una solución óptima. Lo anterior obligó a la comunidad científica a desarrollar métodos de solución que les permitiera disminuir el tiempo computacional, ya sea alcanzando una mejor solución mediante métodos exactos, o generando soluciones factibles

mediante métodos heurísticos, metaheurísticos, híbridos o utilizando enfoques de aprendizaje automático.

Por otro lado, el aprendizaje automático es una de las ramas de la inteligencia artificial, la cual estudia cómo construir sistemas computacionales que, a través de la experiencia, mejoren su desempeño, y que ha generado un alto impacto en el modo en que las organizaciones enfrentan sus retos en campos complejos como en la logística de la cadena de suministro; el Foro Económico Mundial confirmó durante este año que 20 de las 56 compañías pioneras en tecnología utilizan el aprendizaje automático en sus procesos (Venture Beat, 2019), siendo el aprendizaje reforzado, una de las variantes del campo del aprendizaje automático que más ha llamado la atención en la comunidad científica, dada su capacidad de aprender de manera rápida y dar solución a problemas complejos de optimización combinatoria (Nakajima, 2017).

De esta manera, en el presente trabajo se pretende hacer una revisión de literatura con el fin de identificar los principales autores y trabajos científicos, así como un análisis de los principales documentos que serán tomados en cuenta para la realización de este trabajo y de los métodos de aprendizaje reforzado utilizados para dar solución al Problema de Ruteo de Vehículos (CVRP), asimismo, se presentará la formulación del modelo matemático para el problema en cuestión, se propondrán unos objetivos a cumplir y los resultados esperados al finalizar el proyecto; posteriormente, dentro del marco de referencia de describirán algunos de los trabajos académicos similares y el marco teórico que servirá de guía para estudiar modelos de ruteo de vehículos; finalmente, se detallará la metodología y el cronograma a seguir para dar cumplimiento a los objetivos planteados y así, dar solución a instancias del CVRP caracterizadas por su alta complejidad computacional, utilizando una plataforma de programación, con el fin de diseñar un

algoritmo de aprendizaje reforzado capaz de tomar unos valores de entrada y entregar una solución al problema en cuestión.

La Tabla 1 muestra la evidencia del cumplimiento a los objetivos del proyecto.

Tabla 1.

Cumplimiento de objetivos del proyecto

Objetivo	***Cumplimiento***
Realizar una revisión bibliográfica de la literatura relacionada con métodos de aprendizaje reforzado para el VRP.	Capítulo 4
Definición del modelo matemático, su estructura y parámetros.	Capítulo 6
Diseñar un método de solución basado en el algoritmo de aprendizaje reforzado Q-Learning para la solución del CVRP.	Capítulo 7
Evaluar el desempeño del método de solución con instancias de prueba encontradas en la literatura.	Capítulo 7
Realizar un artículo académico de carácter publicable basado en los resultados de la investigación.	Apéndice B

1. Planteamiento del Problema

En una economía global competitiva es necesario generar buenas soluciones en corto tiempo para los problemas que afrontan las empresas en la vida real; del mismo modo, uno de los problemas más estudiados en el campo de la optimización combinatoria es el problema de ruteo de vehículos (VRP), el cual determina el conjunto óptimo de rutas para un parque automotor destinado a servir

un conjunto de clientes (Toth & Vigo, 2002). Hoy en día la importancia del VRP se debe a que reducir los costos de transporte y mejorar la calidad del servicio son factores altamente recomendados para satisfacer los clientes y las industrias; y, por otra parte, el fuerte impacto de aplicación en los campos del transporte, distribución y logística, como por ejemplo, la distribución de aceite, mensajería instantánea, cerveza, recolección de basura y el ruteo de vehículos con restricciones de capacidad (CVRP), es también determinante.

La formulación matemática del modelo CVRP para el presente trabajo se basará en la formulación VRP4 de Toth & Vigo (2002), la cual considera dos variables binarias, x_{ijk} que acumula el número de veces que el arco (i, j) es atravesado por el camión y y_{ijk} que toma el valor de 1 si el cliente i ha sido visitado por el camión k, también cuenta con los parámetros de c_{ij} para costo de atravesar el arco (i, j), C como capacidad de los camiones utilizados, d_i es la demanda del cliente i y n como el número de clientes a servir; de este modo a función objetivo se define como $MinZ = \sum_{i \in V} \sum_{j \in V} c_{ij} \sum_{k=1}^{K} x_{ijk}$

Cuyo propósito es disminuir los costos totales de transporte en los que se incurre al servir un número de clientes. Como se mencionó antes, las restricciones delimitan el espacio de solución: (i) asegurar que cada cliente sea visitado por un solo camión, (ii) garantizar que del depósito cada camión salga una sola vez, actualiza el valor de la variable cuando el arco anterior o posterior del cliente i es atravesado, (iii) limitar la capacidad de los camiones para que no excedan la suma de las demandas de los clientes que visitará cada camión, (iv) evitar que se generen ciclos en la solución, (v) especificar que las variables son de naturaleza binaria ya que sólo pueden tomar los valores de 0 ó 1. Todas las variantes posteriores se basan en el CVRP, lo que ha contribuido a que ésta sea la variante más estudiada en cuanto a métodos de solución. Sin embargo, dada la alta complejidad de este tipo de problema, se ha creado la necesidad en la comunidad científica de

proponer métodos con tiempos computacionales aceptables como el algoritmo Q-Learning evaluado en este proyecto.

2. Justificación del Problema

Una de las principales barreras que impiden a América Latina posicionarse en el sector logístico a nivel mundial, según la Asociación Latinoamericana de Logística (ALALOG), es el alto costo que representa el transporte para la economía de las naciones, siendo el 14% del PIB en países como México, Brasil, Perú y Colombia; esto debido a los elevados costos logísticos, donde cerca del 45% de estos costos son por concepto de actividades de transporte, lo que resalta la importancia de hacer una correcta gestión de estas actividades.

Haciendo referencia a la pertinencia práctica del proyecto, cabe resaltar que en Colombia se han realizado importantes avances en materia de desempeño logístico; razón que situó al país en 2018 en el puesto 58 de 160 países según el Índice de Desempeño Logístico (LPI) del Banco Mundial. Así las cosas, en Colombia son cada vez más comunes las investigaciones destinadas a apoyar la planeación de actividades logísticas que anteriormente se realizaban con base en la experiencia de personas directamente involucradas, como el problema del ruteo de buses escolares, el ruteo de ambulancias para la atención de desastres, la recolección de residuos sólidos, entre otros.

Por otro lado, considerando la pertinencia teórica del proyecto, se trae a colación el hecho de que en la Escuela de Estudios Industriales y Empresariales se han realizado numerosos proyectos de investigación en el área de Big data analytics y ruteo-transporte, brindando soluciones para problemas de optimización de procesos administrativos y logísticos con distintos enfoques. Por lo

tanto, se espera realizar un aporte a la literatura con la presente investigación y ser punto de referencia para posteriores trabajos en esta temática, donde se evalúen diferentes métodos de solución para el CVRP.

3. Objetivos

3.1 Objetivo General

Resolver el Problema de Ruteo de Vehículos Capacitado (CVRP) usando un algoritmo de aprendizaje reforzado Q-Learning.

3.2 Objetivos Específicos

- Realizar una revisión bibliográfica de la literatura relacionada con métodos de aprendizaje reforzado para el VRP.
- Definir del modelo matemático, su estructura y parámetros.
- Diseñar un método de solución basado en el algoritmo de aprendizaje reforzado Q-Learning para la solución del CVRP.
- Evaluar el desempeño del método de solución con instancias de prueba encontradas en la literatura.
- Realizar un artículo académico de carácter publicable basado en los resultados de la investigación.

4. Revisión de la Literatura

4.1 Análisis de la Literatura

El Problema de Ruteo de Vehículos (VRP) en el campo empresarial sigue siendo uno de los desafíos más importantes a vencer para sectores como el transporte, la distribución y logística, el cual, por su alta complejidad computacional, siendo uno de los problemas de optimización

combinatoria discreta que más ha sido estudiado dentro de la comunidad científica, y que además ha generado que, aunque una gran variedad de métodos exactos y heurísticos hayan sido propuestos, hoy día sigue siendo todo un reto encontrar soluciones rápidas y factibles (Nazari, Oroojlooy, Snyder, & Takáč, 2018).

El primer registro en la literatura del VRP corresponde al trabajo realizado por Dantzig, Fulkerson & Johnson, quienes describen el Problema del Agente Viajero (TSP) como aquel vendedor que necesita encontrar la ruta más corta, partiendo de una ciudad inicial, para visitar un específico número de ciudades y regresando al mismo punto de salida (G. Dantzig, Fulkerson, & Johnson, 1954); posteriormente, Dantzig & Ramser proponen un modelo generalizado del TSP, presentando el modelo del "Truck Dispatching Problem", el cual representaba una flota de camiones con capacidad homogénea que deben satisfacer una demanda de aceite para cierto número de estaciones de gas y minimizando la distancia recorrida (G. B. Dantzig & Ramser, 1959). Cinco años más tarde, Clarke & Wright presentaron el modelo anterior como un problema generalizado de optimización lineal para el sector de logística y transporte, que además de contar con clientes o estaciones de gas geográficamente dispersos, utilizaban una flota de camiones de capacidad variable (Clarke & Wright, 1964).

4.1.1 Definición del Problema de Ruteo de Vehículos Capacitado. El VRP se define como el diseño de rutas de entrega al mínimo costo partiendo de un depósito hacia un grupo de clientes geográficamente dispersos y sujeto a restricciones que, en su variante clásica, se puede describir matemáticamente como: Sea $G = (V, A)$ un grafo dirigido donde $V = \{0, 1, 2, \dots, n\}$ son los vértices y $A = \{(i, j): i, j \in \mathrm{V}, \mathrm{i} \neq \mathrm{j}\}$ el grupo de arcos, el vértice cero es el depósito y los demás vértices corresponden a los clientes, el depósito cuenta con una flota de m vehículos homogéneos

de capacidad Q y cada cliente i tiene una demanda no negativa q_i (Kumar & Panneerselvam, 2012), que para el Problema del Ruteo de Vehículos Capacitado (CVRP) cuenta además con las siguientes restricciones: (i) cada cliente debe ser visitado una única vez por un único vehículo, (ii) cada ruta generada debe iniciar y finalizar en el depósito, (iii) para cada ruta, la demanda total no debe exceder la capacidad Q del vehículo y (iv) para cada ruta, la distancia total recorrida no debe exceder el límite L designado (L. S. Lee et al., 2012).

4.1.2 Otras variantes del VRP. En 1983 Solomon presenta un modelo de VRP con la restricción de ventanas de tiempo (VRPTW), que tomó en consideración el tiempo inicial y final de servicio para cada nodo, por lo cual, el vehículo salía del depósito a realizar las entregas a cada cliente y debía regresar en una ventana de tiempo definida previamente por el modelo (Ahn & Shin, 1991); del mismo modo la comunidad científica ha propuesto múltiples variantes del problema como: el VRP asimétrico, en el cual el costo de ir de un cliente i hacia otro j difiere de ir desde el cliente j hacia el cliente i, el VRP con múltiple depósito, que involucra la asignación de clientes y vehículos a cada depósitos, el VRP de dos escalones, donde el primer nivel es la entrega desde el depósito hacia los centros de distribución y el segundo nivel es la entrega desde los centros de distribución hacia los clientes, el VRP con flota heterogénea, que considera para cada vehículo una capacidad o costo asociado diferente, el Green VRP, que apunta a minimizar el costo ambiental y económico asociado al proceso de optimización, el VRP con devoluciones, en el cual los clientes pueden demandar o regresar algunos productos (Yadav, Sharma, & Routroy, 2018).

4.1.3 Métodos de solución del CVRP. Debido a la alta complejidad computacional de dar solución a estos problemas y al insuficiente desarrollo tecnológico de la época, solo se resolvieron problemas de configuración estática mientras que las versiones estocásticas, dinámicas y de carácter disperso no fueron estudiadas a profundidad; sin embargo, la investigación del VRP tomó un acelerón durante la década de los 90's, debido principalmente a la capacidad y disponibilidad de los microcomputadores que permitían incorporar mejores algoritmos de búsqueda de información en la red (Eksioglu, Vural, & Reisman, 2009); asimismo, el CVRP también se considera un problema de alta complejidad computacional o del tipo NP-hard, para el cual los métodos de solución exactos como el Branch and Bound acuñado por Little, Murty, Sweeney, & Karel (1963), el Branch and Cut desarrollado por Kalvelagen (2003), el Branch and Price presentado por G. B. Dantzig & Wolfe (1960) y el Robust Branch Cut and Pricing propuesto por Poggi & Uchoa (2003), solo lograron obtener resultados óptimos para pequeñas instancias, dando como resultado los métodos heurísticos y metaheurísticos, los cuales permiten encontrar buenas soluciones sin requerir un gran volumen de tiempo computacional, pero que no garantizan que la soluciones encontradas sean las óptimas (Szeto, Wu, & Ho, 2011).

Dentro de las heurísticas utilizadas para dar solución al CVRP se encuentra el algoritmo de ahorro desarrollado por Clarke & Wright en 1964, que inicia con una solución formada por un número n de rutas de ida y vuelta $(0, i, 0)$ $(i \in V\{0\})$; en cada iteración, se combina una ruta que finaliza en i con otra ruta que empieza en j, maximizando el ahorro $s_{ij} = c_{i0} + c_{0j} - c_{ij}$, siempre que la combinación sea factible; el proceso finaliza cuando no es posible combinar rutas (Clarke & Wright, 1964). El algoritmo de barrido propuesto por Gillett & Miller (1974), hace parte de las heurísticas clásicas de partición, en el cual un subgrupo de rutas prometedoras llamadas pétalos eran secuencialmente generadas al rotar una media línea enraizada en el depósito, siempre que las

restricciones de capacidad y de distancia para una sola ruta fueran cumplidas. Otro ejemplo es la heurística de "primero-agrupar, segundo-enrutar" propuesta por Fisher & Jaikumar (1981), consiste inicialmente en ubicar un número m de semillas y crear un grupo para cada semilla con el fin de minimizar la suma de la distancia del cliente respecto a la semilla, mientras satisface la restricción de capacidad.

Por otro lado, las metaheurísticas se pueden clasificar en búsqueda local, búsqueda global y mecanismos de aprendizaje y a continuación se describen cada una de ellas; las características principales de la búsqueda local son las reglas utilizadas para definir el vecindario de una solución y el modo de realizar la búsqueda, algunos ejemplos son la búsqueda tabú propuesta por Glover en 1986, el recocido simulado de Kirkpatrick, Gelatt & Vecchi en 1983, recocido determinístico presentado por Dueck en 1993, entre otros (Laporte, 2009); la búsqueda global trabaja con una población de soluciones, siendo los algoritmos genéticos uno de sus ejemplos más representativos, cuya solución es análoga a un cromosoma que inicia su ciclo generando una población inicial de individuos, luego evalúa, selecciona, recombina, genera mutaciones y establece una nueva población (Yadav et al., 2018), y finalmente se encuentran los mecanismos de aprendizaje que cuentan con la habilidad de aprender de la experiencia e incrementalmente ajustar sus pesos en cada iteración, la optimización de colonia de hormigas es uno de sus ejemplos y consiste en asemejarse al comportamiento de una hormiga que genera un rastro de feromonas mientras se desplaza para que, al entrar en contacto con otras hormigas, estas puedan optar por seguirlo o no; si las hormigas siguen el rastro, sus feromonas se añadirán al rastro actual y así la probabilidad de que la siguiente hormiga siga el mismo camino aumentará (Laporte, 2009).

4.1.4 Solución al CVRP mediante Aprendizaje Reforzado. Recientemente, se propuso un mecanismo de aprendizaje que combinaba métodos de solución del aprendizaje reforzado y la incrustación de grafos para solucionar el TSP, el cual obtenía mejores soluciones a medida que el algoritmo de aprendizaje Q-Learning tomaba acciones basado en su estado inicial, capturaba la solución anterior y, dependiendo de la función de recompensa, ajustaba los parámetros de la función de costo en la red de incrustación de grafos, tomaba una nueva decisión para pasar al siguiente estado y así obtenía una mejor solución (Dai, Khalil, Zhang, Dilkina, & Song, 2017).

Asimismo, Vinyals, Fortunato, & Jaitly, (2015) propusieron una infraestructura de Pointer Network que resuelve el mismo problema del TSP mediante un modelo Sequence-to-Sequence basado en Redes Neuronales Recurrentes (RNN), modelo tomado por Bello, Pham, Le, Norouzi, & Bengio, (2016) para desarrollar un marco de trabajo que hace uso de Aprendizaje Reforzado para mejorar la política modelada por un Pointer Network y la utiliza para dar solución a problemas de optimización combinatoria como el TSP y el problema de la mochila; ambos trabajos sirvieron como punto de referencia para el desarrollo del marco de trabajo propuesto por la Universidad de Lehigh que permitió dar solución al VRP con demanda estocástica, utilizando aprendizaje reforzado, en el cual, partiendo de una política de entrenamiento, se propuso una estructura basada en el Proceso de Decisión Markoviano (MDP) y en las Redes Neuronales Recurrentes (RNN) capaz de resolver, mediante una secuencia de decisiones, cualquier problema tomado de una distribución de entrenamiento asignada; lo que quiere decir que si se genera una nueva instancia del CVRP con el mismo número de clientes y capacidad del vehículo, y utilizando la misma distribución de ubicación y demanda del entrenamiento, la política entregará buenos resultados sin la necesidad de entrenar nuevamente el modelo para cada instancia; este modelo, presentado por Nazari et al., (2018) representó un avance significativo en la búsqueda de dar solución a

aplicaciones reales del VRP, ya que el método propuesto se logró ajustar bien a medida que el problema aumentaba y obtuvo un desempeño de tiempo de solución competitivo en comparación con las heurísticas clásicas utilizadas para el VRP con demanda estocástica.

5. Marco Teórico

A continuación, se presenta la conceptualización necesaria para entender cómo se desarrollan los problemas de optimización combinatoria como el CVRP y cómo se constituyen los algoritmos de aprendizaje automático como el Q-Learning.

5.1 Problemas de optimización combinatoria

El análisis y estudio de los problemas de optimización combinatoria son parte de uno de los campos más jóvenes y activos en la matemática discreta que expresa todas las posibles soluciones que puede llegar a tener un problema (Korte & Vygen, 2018). Al día de hoy, este campo de la optimización combinatoria ha llegado a ser considerado como fuerza motora de la matemática discreta siendo un tema tratado por casi 50 años y desde entonces, ha venido desarrollando estructuras discretas con fundamentos básicos de la teoría de grafos, la programación lineal y entera, y la teoría de complejidad (Cook, Cunningham, Pulleyblank, & Schrijver, 1998)

La optimización combinatoria tiene sus raíces en la teoría combinatoria donde se hablaba de combinaciones, agrupamientos, orden y la selección discreta de objetivos que usualmente era un número finito. Luego, con la investigación de operaciones su importancia pasó de “¿Existe la combinación?” o “¿Cuántas combinaciones existen para solucionar el problema?” al objetivo explícito de la optimización combinatoria que es el de dar respuesta a “¿Cuál es la mejor

combinación que soluciona el problema?". Por último, también podría decirse que esta rama le debe su existencia al advenimiento de la computación moderna ya que facilitó el abordaje de problemas de gran complejidad computacional que nadie hubiese podido llevar a cabo sin dicho advenimiento y teniendo en cuenta uno de los aspectos principales que es el buen desempeño de los algoritmos que solucionan los problemas en cuestión (Cook et al., 1998).

Es importante saber que el mayor estimulo lo genera el hecho de que miles de problemas de la vida real se pueden formular como problemas de optimización combinatoria abstractos, donde la gran mayoría pueden ser expresados fácilmente con teoría de grafos o programación lineal entera (Korte & Vygen, 2012)

5.2 Variantes del Problema de Ruteo de Vehículos (VRP)

El VRP es uno de los problemas de optimización combinatoria más importantes y por ende más estudiados; consiste en la determinación del conjunto óptimo de rutas que debe desempeñar una flota de vehículos para servir a un conjunto de clientes (Toth & Vigo, 2002). Ya van 60 años desde que Dantzig and Ramser lo introdujeron y desde entonces han surgido un número importante de variantes de dicho problema.

5.2.1 Vehicle Routing Problem Time Window (VRPTW). También conocido como el problema de ruteo de vehículos con restricciones de ventanas de tiempo es una extensión del VRP donde a cada cliente se le asocia el servicio en una ventana de tiempo y el vehículo que lo ejecuta debe permanecer en la ubicación de dicho cliente durante el servicio.

El VRPTW es un problema de complejidad NP-hard abordado por primera vez en 1983 por Marius M. Solomon, el cual permite que para ciertas ventanas de tiempo "suaves" se pueda romper

la restricción asumiendo un costo, pero para las denominadas ventanas de tiempo “duras” no se permite un vehículo llegue después del tiempo de iniciación de servicio. También establece que, si un vehículo llega antes de que el cliente esté listo para comenzar el servicio, el vehiculó debe esperar.

5.2.2 Vehicle Routing Problem with Backhauls (VRPB). Es otra extensión del CVRP en donde los clientes se subdividen en dos subconjuntos. El primer subconjunto representa los *linehaul customers* que son aquellos que requieren una entrega de una determinada cantidad de producto. El segundo subconjunto denominado los *backhaul customers* son aquellos que requieren de que se les recoja una cierta cantidad de producto. Esta situación es bastante práctica, y un ejemplo común es el de tomar a los supermercados como linehaul customers y a los proveedores como los backhaul customers; también se debe señalar que frecuentemente los linehaul customers necesitan ser priorizados en el servicio sobre los backhaul customers.

Además, al igual que el VRP clásico, son también abarcadas desde dos perspectivas diferentes; la primera es una versión *simétrica* del problema (VRPB) donde la distancia entre cada par de locaciones es la misma en ambas direcciones; y la segunda se refiere a la versión *asimétrica* del problema (AVRP) donde no se sostiene que la distancia entre dos locaciones sea la misma en ambas direcciones. Así mismo, ambas versiones generalizan al CVRP simétrico y asimétrico cuando $B = 0$, lo que las hace de complejidad NP-hard.

5.2.3 Vehicle Routing Problem with Pickup and Delivery (VRPPD). Este problema también hace parte de la familia de enrutamiento de vehículos con restricciones de capacidad en la flota donde dadas unas demandas de entrega y unas demandas de recogida se debe diseñar el sistema

de rutas que minimice el costo de la operación que bien puede transportar bienes o personas. Este problema se encuentra en la literatura comúnmente con algunas variaciones adicionales que lo hacen más complejo y que pueden incluir restricciones tales como ventanas de tiempo, restricción en la cantidad de pasajeros, flota heterogénea, entre otros.

5.3 Teoría de complejidad computacional

Esta teoría se centra en clasificar los problemas computacionales acorde a su dificultad (ver Figura *1*), y sus clases según Sipser (2013) son:

- Clase P: Básicamente comprende a todos los problemas que se pueden resolver por un programa razonablemente rápido, problemas como multiplicación, laberintos y ordenación. La P proviene de que el tiempo en que son resueltos estos problemas en una máquina de Turing determinista es de comportamiento polinomial (*Polinomial time*). Esta clase se conoce como una clase robusta que no es afectada por las particularidades del modelo computacional que se aborde y cuyos problemas de solucionan de forma realista en computadora.
- Clase NP: Esta clase es de mayor complejidad puesto que contiene todos los problemas de la clase P, pero también abarca muchos otros importantes como, por ejemplo: ruteo vehicular, planificación de trabajo, diseño de circuitos y plegamiento de proteínas. Estos problemas tienen la característica de que, a pesar de tomar una gran cantidad de tiempo para encontrar una solución al problema, ésta se puede corroborar en un tiempo razonable, es decir, estos problemas tienen verificadores de tiempo polinomial.

  NP proviene de que el tiempo en el que son resueltos estos problemas en una máquina de Turing no determinística es polinomial (*Nondeterministic Polinomial time*).

- Clase NP-Complete: Es un subgrupo de NP que abarca a todos los problemas esencialmente iguales, con algunas complicaciones en sus tiempos polinomiales lo cual los hace más difíciles que los NP. Algunos problemas ubicados acá son el sudoku, el plegamiento de las proteínas, el emparejamiento 3D, el knapsack problem y el satisfiability problem
- Clase NP-Hard: Son problemas al menos tan difíciles como los problemas en NP y su alta complejidad radica en que en los problemas NP-Hard, a diferencia de los NP, los tiempos de solución como los tiempos de verificación crecen exponencialmente con el tamaño del problema y también en que algunos de estos problemas son de complejidad desconocida. Algunos ejemplos de estos problemas son: el ajedrez, el TSP y el problema de la suma de subconjuntos.

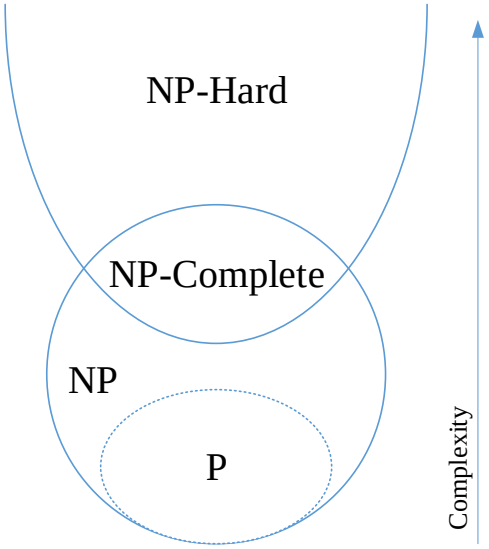


Figura 1. Diagrama de Euler para problemas P, NP, NP-Complete y NP-Hard.

5.4 Machine Learning

En 1958 Arthur Samuel dijo, “Machine Learning es el campo de estudio que le da a los computadores la habilidad de aprender sin ser explícitamente programados”, pero también, con una perspectiva desde la ingeniería; Tom Mitchell en 1997 la acuñó diciendo: “Se dice que un programa computacional es capaz de aprender de la experiencia E respecto a una tarea T con una medida de desempeño P, si el desempeño realizado en T, medido por P, mejora con la experiencia E”. Por lo tanto, hoy por hoy a grandes rasgos se entiende que Machine Learning es la ciencia o arte de programar computadores para que estos aprendan a hacer una detección automatizada de patrones significativos en grandes conjuntos de datos, es decir, que puedan aprender a partir de los datos (Géron, 2017).

Los modelos que surgen a partir de esta ciencia permiten construir un modelo que acepta variables de entrada, variables con las cuales se pueden obtener predicciones o resultados teniendo en cuenta que desde un principio al programa computacional no se le proveen todas las precondiciones puesto que el algoritmo está diseñado de tal manera que aprenda por sí mismo.

En algunas ocasiones Machine Learning e Inteligencia Artificial suelen ser campos que se confunden entre sí, sin embargo, estos dos campos son diferentes, ya que Machine Learning se centra en diseñar software que pueda aprender de experiencias en el pasado.

Algunas de las aplicaciones más conocidas de Machine Learning son: la detección de e-mail spam, publicidad enfocada (Google AdSense), motores de búsqueda en e-commerce, detección de actividad sospechosa a partir de CCTV’s, diagnósticos médicos para la detección de enfermedades, entre otros (Akhtar, 2017).

El Machine Learning se divide actualmente en tres partes, aprendizaje supervisado, aprendizaje no supervisado y aprendizaje reforzado.

En el aprendizaje supervisado los datos usados en el entrenamiento del algoritmo incluyen como información extra las soluciones deseadas, llamadas etiquetas, de tal manera que es el entorno quien hace las veces de maestro y supervisa al algoritmo proveyéndole de dichas etiquetas. Un ejemplo es el filtro de spam en correo electrónico y otro buen ejemplo es la predicción del precio de una casa con el paso del tiempo.

En el aprendizaje no supervisado los datos usados en el entrenamiento son similares entre sí, pero estos datos no contienen etiquetas, es decir que el sistema trata de aprender sin la necesidad de ser supervisado.

Por último, se encuentra un aprendizaje bastante diferente a los dos anteriores, ya que existe un sistema de aprendizaje llamado *agente* que puede observar el entorno, seleccionar y llevar a cabo *acciones*, para al final obtener una ganancia o pérdida según sea el caso. Este sistema debe encontrar por sí mismo la mejor estrategia, llamada *política* que define cuál acción debe tomar el agente cuando se da una determinada situación (Géron, 2017).

5.5 Reinforcement Learning

En el aprendizaje reforzado existe un agente que determina las acciones a tomar en un entorno especifico con el propósito de maximizar una recompensa numérica de modo que el aprendizaje se dé a partir de la interacción con el sistema. Se podría pensar en cómo un bebé aprende a caminar, la primera vez logra levantarse, trata de dar algunos pasos, se cae y vuelve a intentarlo. Después de un tiempo el bebé aprende a caminar y realmente en este escenario no hay quién le enseñe a caminar. Este aprendizaje se trata de prueba y error, donde hay situaciones humanas tanto como sistemas biológicos que no reciben instrucciones acerca de cómo realizar una tarea específica, sin

embargo, después su ejecución se hace una evaluación de esta con el propósito de mejorar en la siguiente ocasión (Géron, 2017).

El aprendizaje reforzado está basado en la psicología comportamental. Un ejemplo de esto se dio en el año 1890, donde un fisiólogo ruso llamado Ivan Pavlov llevó a cabo un experimento donde analizaba la salivación de los perros justo antes de ser alimentados. A los perros se les llamaba con un silbato cada que se les ofrecía comida y después de un tiempo los perros aprendían a salivar con el sonar del silbato independientemente de si habia comida o no. Allí, Pavlov precisó que la conexión estímulo-respuesta no necesitaba aprendizaje explícito, ya que un estímulo indefinido era causante de una respuesta indefinida, es decir, el aprendizaje de los caninos se generaba de la interacción con el entorno.

Una de las características de este aprendizaje es que utiliza enfoques de decisiones basados en cadenas de Markov y métodos basados en gradiente, en el cual, el agente solo puede recibir una recompensa después de ejecutar una acción, entonces el agente debe continuar tomando acciones e interactuar con el entorno para poder establecer cuál es la política óptima a través de prueba y error, tal como se muestra más adelante en el ejemplo presentado del algoritmo de aprendizaje reforzado Q-Learning. (Akhtar, 2017).

5.6 Modelos de Markov

Un modelo de Markov se refiere a un modelo estocástico usado para modelar sistemas aleatorios cambiantes. Dentro de estos se hace referencia a los procesos que satisfacen la propiedad Markoviana, procesos que no tienen memoria. Los modelos de Markov comúnmente más usados son: cadena de Markov, modelo oculto de Markov, proceso de decisión de Markov y proceso de decisión de Markov parcialmente observable.

Estos procesos de Markov son usados ampliamente en distintos campos como en los sistemas de comunicación, secuenciación de ADN, movilidad, redes sociales, epidemiología, ingeniería financiera, sistemas de colas, sistemas de decisión, entre otros (Gagniuc, 2017).

5.7 Q-Learning

Fue acuñado en 1989 por Chris Watkins y es un algoritmo de aprendizaje reforzado, que le permite a un agente aprender el valor óptimo (o muy cercano al óptimo) de una política. El algoritmo Q-Learning busca optimizar las soluciones de este proceso de decisión de Márkov. De manera que en cada estado $S_t \in$ S el agente se enfrenta a un vector de posibles acciones $A_t \in$ A. Ejecutar una acción A_t en el estado S_t le otorga una recompensa dada por R (S_t, A_t), tomada de la una matriz R de tamaño S x A, y posiciona al agente en el estado S_{t+1} que para este trabajo resulta ser $S_{t+1} = A_t$ = T (S_t, A_t).

Una política π es la encargada de establecer cuál es la acción A_t que debe tomar el agente en el estado S_t. Por lo tanto, el objetivo del agente es aprender la política π * que maximiza la recompensa total descontada esperada durante el tiempo de vida del agente en el entorno.

La memoria del agente es representada por una matriz de valores Q de tamaño S x A, que inicialmente es una matriz de ceros tamaño de la matriz R.

Cada valor Q (S_t, A_t) de la matriz Q, representa un estimado del valor esperado de la recompensa que puede ser obtenida a partir de ejecutar la acción A_t estando en el estado S_t, siguiendo la política π. Los valores Q, inicialmente todos ceros, son determinados y actualizados resolviendo la ecuación generalizada de Q-Learning (1) (Sutton & Barto, 2018).

(1)

$$Q(S_t, A_t) = \ Q(S_t, A_t) + \ \alpha \ \left[\mathrm{R}(S_t, A_t) + \gamma \max_{A} Q(S_{t+1}, A) - Q(S_t, A_t)\right]$$

Donde los valores $Q \rightarrow Q^*$ cuando el número de corridas $n \rightarrow \infty$. Esta función generalizada de Q-Learning demuestra un comportamiento temprano de convergencia y aunque la política π incide en el par estado-acción, lo único necesario para la correcta convergencia de los pares estado-acción en sus valores óptimos es que estos pares se sigan actualizando continuamente (Watkins, 1989).

Adicionalmente, uno de los temas principales en el aprendizaje reforzado es encontrar el balance adecuado entre la exploración y explotación del agente. Generalmente al inicio del aprendizaje la acción tomada por el agente es naturalmente aleatoria (Exploración), pero conforme se da este aprendizaje el agente toma ventaja de los valores Q que va conociendo (Explotación) y a esta relación entre exploración y explotación se le representa por medio del valor ε que toma los reales comprendidos entre 0 y 1. Por ejemplo, si $\varepsilon = 0.9$ significa que de las acciones a tomar por el agente 90% son aleatorias y 10% hacen uso de los valores Q conocidos (Atienza, 2018).

Así mismo, α se define como la taza de aprendizaje con la que el agente hará la actualización de su memoria en la matriz Q. Este parámetro α es crucial puesto que en su representación más sencilla un valor de α muy grande puede significar nunca poder encontrar un mínimo por estar siempre moviéndose entre valores muy adelante y muy atrás del mínimo. Pero por otro lado un valor muy pequeño de α puede significar tener que tomar un número muy grande de iteraciones antes de encontrar el mínimo (Atienza, 2018).

Por último, se hace énfasis en que el algoritmo permite encontrar una buena solución factible en relación estado-acción, para un problema que se pueda representar como un proceso de decisión de Márkov. El objetivo principal de este entrenamiento es encontrar la secuencia de acciones que

maximizan la suma de futuras recompensas, siendo así en la ruta más corta desde el estado inicial al estado final y su estructura se observa en la Figura 2:

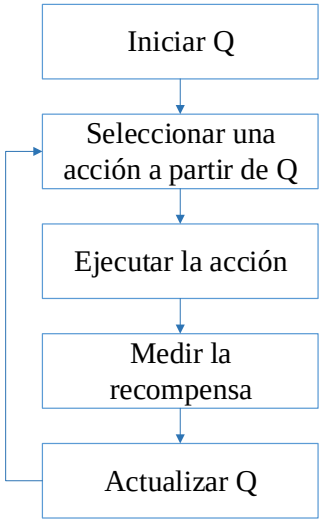


Figura 2. Estructura del algoritmo Q-Learning. Adaptado de Akhtar (2017).

Un ejemplo puede ser imaginarse una búsqueda del tesoro en un laberinto. La persona (agente) empezaría a explorar desde una posición dada (estado inicial), desde cualquier estado puedes ir hacia la izquierda, la derecha, arriba o abajo, o quedarte en el mismo lugar. Cada acción te llevará a una diferente posición (estado), donde en uno de esos estados se encuentra el tesoro (estado objetivo). Además, el laberinto cuenta con serpientes en ciertas posiciones. El objetivo es atravesar desde el estado inicial hasta el estado objetivo siguiendo el camino que no tiene serpientes, tal como se muestra en la malla mostrada en la Figura *3*.

INICIO		SERPIENTES	
●	SERPIENTES	SERPIENTES	
●	SERPIENTES		
●	●	●	TESORO

Figura 3. Representación de la malla para ejemplo del laberinto

Cuando se ubique el agente en la malla empezará primero a explorar; él no sabe dónde se encuentran las serpientes ni el tesoro, así que, para ayudarle en su búsqueda, se le asignarán

recompensas luego de tomar cada acción. Para cada pozo de serpientes obtendrá una recompensa de -10, mientras que cada vez que llegue al tesoro obtendrá una recompensa de +10; también se quiere que el agente llegue al estado objetivo lo más pronto posible, por lo que los demás estados tendrán una recompensa de -1. Luego, se le dirá que debe maximizar la recompensa, por lo cual el agente explorará, aprenderá que no debe cruzarse con las serpientes y encontrará el camino más corto para llegar al tesoro, tal como lo indican los puntos marcados en la malla.

5.8 Métodos basados en gradientes

Los métodos de gradiente son algoritmos que resuelven los problemas de la forma, $min_{x\in\mathbb{R}^n}\ f(x)$, con f: $\mathbb{R}^n$ continuamente diferenciable.

Dentro de los cuales destacan varios como, el método del gradiente descendiente, el método del gradiente descendente estocástico y el método del gradiente conjugado, entre muchos otros. El método de gradiente descendiente es de primer orden y localiza el punto óptimo de la función a partir del valor de pendiente en dirección del mínimo o máximo local según sea el caso. El método de gradiente descendente estocástico también es un método iterativo para la optimización de problemas con mínimos o máximos locales, y a diferencia del anterior se le llama estocástico porque selecciona de forma aleatoria las muestras a las cuales se les evalúa el valor del gradiente Por último, el método de gradiente conjugado puede ser considerado un punto intermedio entre el método de gradiente con mayor descenso y el método de Newton, puesto que la idea es acelerar la lenta convergencia del primero y no usar toda la información que requiere el segundo para evaluar, almacenar e invertir el Hessiano (Polak, 1997).

Con respecto al aprendizaje reforzado, que su objetivo es maximizar la ganancia “esperada” J cuando se sigue una política π, se define un conjunto de parámetros θ para parametrizar esta

política π e igual que en cualquier otro problema de machine learning, si se puede encontrar el valor de esos parámetros θ los cuales maximizan J, entonces se da por resuelto el problema. Una manera práctica para resolver este problema de maximización en la literatura de machine learning es el método de gradiente ascendiente (o descendiente) en donde se sigue una regla de actualización de la forma: $\Theta_{t+1} = \Theta_t + \alpha \Delta J(\Theta_t)$.

Y es acá donde se presenta el mayor reto porque computar este método de gradiente significa que se debe calcular qué tanto cambia J incluso cuando el cambio en θ sea muy pequeño en cualquier dimensión dada. Razón por la cual se han abordado otros métodos de gradiente que buscan facilitar esta manera de parametrizar los valores de θ como lo es el método de gradiente descendiente estocástico ya mencionado (Géron, 2017).

5.9 Diseños Factoriales

Permiten identificar el efecto que dos o más factores tienen sobre un estudio concreto y así determinar el cambio en la respuesta producido por un cambio en el nivel de cada factor. Uno de los casos especiales en los diseños factoriales es el de k factores donde cada uno tiene sólo dos niveles caracterizados usualmente por ser “alto” y “bajo” y, dado que una réplica completa de este tipo requiere 2x2x2x ... x2 = 2^k, se conoce como diseño factorial 2^k.

6. Formulación del Modelo para el CVRP

En el siguiente apartado se presenta la descripción del modelo y notación matemática del problema en cuestión.

6.1 Descripción del Modelo

El modelo propuesto en el presente trabajo aborda la primera variante del VRP, conocida como el problema de ruteo de vehículos con restricciones de capacidad o CVRP; el cual, consiste en obtener el conjunto de rutas de un grupo de vehículos que tienen un límite de capacidad (peso, volumen, unidades, etc.) para transportar desde un depósito hacia un número determinado de clientes. En el CVRP, a cada uno de los clientes les corresponde entregas únicas y las demandas son determinísticas, previamente definidas y no pueden ser fraccionadas; los vehículos son homogéneos, solo cuentan con la restricción de capacidad y están a disposición de un único depósito. El CVRP es formulado como un modelo de programación lineal entera mixta cuyo objetivo es minimizar el costo total (en distancia o tiempo) de atender la demanda de todos los clientes y a continuación se presenta la formulación matemática propuesta para este proyecto basada en la formulación VRP4 de Toth & Vigo (2002).

6.1.1 Índices

$V = (h, i, j = 0,1, \dots, N)$

$k = (k = 1, \dots, K)$.

6.1.2 Parámetros de Entrada

$c_{ij} =$ Costo de ir desde el cliente i hasta el cliente j.

$C =$ Capacidad de los vehículos utilizados.

$d_i =$ Demanda del cliente i.

6.1.3 Variables de Decisión

$X_{ijk} =$ Variable binaria que toma el valor de 1 si el arco (i, j) es una ruta factible para ser atravesada por el vehículo k, y toma 0 de lo contrario.

$Y_{ik} =$ Variable binaria que toma el valor de 1 si el cliente i ha sido visitado por el vehículo k, y toma 0 de lo contrario.

6.1.4 Función Objetivo. Para el problema en cuestión, la función objetivo (2) tiene como propósito disminuir los costos totales de transporte c_{ij} en los que se incurre al atender la demanda de los clientes X_{ijk}.

$$Min\ Z = \sum_{i \in V} \sum_{j \in V} c_{ij} \sum_{k=1}^{K} X_{ijk} \tag{2}$$

6.1.5 Restricciones. En aras de delimitar el espacio de solución y relajar el problema modelado, se presentan las siguientes restricciones:

$\sum_{k=1}^{K} Y_{ik} = 1 \qquad \forall i \in V \setminus \{0\}$ (3)

$\sum_{k=1}^{K} Y_{0k} = K$ (4)

$\sum_{j \in V} X_{ijk} = \sum_{j \in V} X_{jik} = Y_{ik} \qquad \forall i \in V, k = 1, \dots, K$ (5)

$\sum_{i=1}^{I} d_i Y_{ik} \leq C$ (6)

$\sum_{i \in S} \sum_{j \notin S} X_{ijk} \geq Y_{hk} \qquad \forall S \subseteq V \setminus \{0\}, h \in S, k = 1, \dots, K$ (7)

$Y_{ik} \in \{0,1\} \qquad \forall i \in V, k = 1, \dots, K$ (8)

$X_{ijk} \in \{0,1\} \qquad \forall i, j \in V, k = 1, \dots, K$ (9)

La restricción (3) asegura que cada cliente sea visitado por solo un vehículo, mientras que la (4) garantiza que cada vehículo salga del depósito solo una vez. La restricción (5) actualiza el valor

de la variable Y_{ik} cuando el arco anterior o posterior del cliente i es atravesado, la (6) limita la capacidad de los vehículos de modo que no se exceda la suma de las demandas de los clientes que visitará cada vehículo y la (7) se encarga de la eliminación de subtours o ciclos hamiltonianos, de modo que no se generen ciclos en la solución. Finalmente, las restricciones (8) y (9) especifican que las variables X_{jik} y Y_{ik} son de naturaleza binaria por lo que sus valores solo pueden ser 1 o 0.

7. Algoritmo de Q-Learning para el CVRP

Este trabajo tiene como propósito solucionar el CVRP bajo el enfoque del Reinforcement Learning como un proceso de decisión de Márkov (MDP) asumiendo que el entorno, por simplicidad, es determinístico, en el que, a partir de una determinada acción ejecutada por un agente en un estado dado, resulta en un próximo estado y en un valor de recompensa conocidos. El proceso de decisión de Márkov se basa en una tupla (S, A, R, T, γ) que define la interacción del agente con el entorno, donde S representa el conjunto de estados, A el conjunto de acciones, R la función de recompensa R (S_t, A_t), T la función de transición T (S_t, A_t) = S_{t+1} y, el factor de descuento es denotado por la letra γ que representa el descuento que el agente le hace a las recompensas futuras en γ^t veces (Sutton & Barto, 2018).

7.1 Definición del Algoritmo

En el presente trabajo, la solución propuesta para el CVRP con el algoritmo Q-Learning realizado en Python define el entorno como:

- S: Conjunto de posibles estados S_t en los que se puede encontrar el agente. Es conjunto está conformado por los clientes de la instancia a evaluar y por dos estados adicionales que

son: salir del depósito y retornar al depósito. Es decir, el conjunto S es de tamaño C + 2, donde C es la cantidad de clientes de la instancia.

- A: Conjunto de acciones A_t en el entorno del agente. La primera acción es partir del depósito, la última acción es retornar al depósito y todas las demás representan el cliente a visitar. La primera acción es omitida durante el criterio de posibles acciones del agente puesto que parte del depósito una única vez y no puede volver al estado "salir del depósito". A su vez, la última acción que denota "retornar al depósito" ocurre una vez el agente ha agotado la capacidad de su vehículo y no puede saldar más clientes. El conjunto A por lo tanto posee un tamaño de C + 2.
- R: La función de recompensa R (S_t, A_t) está definida como una *matriz R* de tamaño S x A, donde los valores R (S_t, A_t) contenidos en la matriz R indican la distancia euclidiana entre el cliente en el que está el agente S_t y el cliente que visitará el agente A_t, señalando así el costo de la trayectoria $\overline{S_t\, A_t}$. En esta matriz, a todos los valores R (S_t, A_t) igual a cero se les omite como posibles acciones, siendo que estos representan una transición del estado S_t al mismo estado S_t o la imposibilidad de hacer uso del estado, como lo es el estado "salir del depósito" en el cual una vez el agente ha salido, no puede volver a "salir del depósito". Lo mismo con el estado "retornar al depósito", ya que ocurre una única vez cuando el agente al no contar con la capacidad vehicular para saldar al menos un cliente no le queda otra opción más que retornar al depósito.
- T: La función de transición T (S_t, A_t) se puede describir como "estando en un determinado nodo S_t, visitar el nodo A_t", es decir que $S_{t+1} = A_t$.
- γ: Es el factor de descuento y como hiperparámetro en este problema simboliza qué tanto queremos que el agente descuente los valores futuros de recompensa (sea costo o ganancia)

al visitar un cliente o retornar al depósito. Dicho de otra manera, este hiperparámetro le añade incertidumbre al agente en la predicción del valor de la recompensa entre más pasos t en el futuro traté de visualizar.

Además, se dispone de una matriz Q inicializada en ceros, del tamaño S x A, para ir conservando la memoria del agente conforme ejecuta una acción A_t y cambia de estado S_t.

Por último, la política de decisión del agente se ε-greedy, donde la exploración está definida como visitar aleatoriamente cualquiera de los clientes de la instancia y la explotación es visitar al cliente con mayor valor Q.

Como ejemplo se podría pensar en un problema CVRP de 2 vehículos de capacidad CAP = 15 y 3 clientes con demandas DEM (1) = 10, DEM (2) = 5 y DEM (3) = 15, cuya configuración de parámetros se observa en la Tabla *2*. Se definen las matrices de recompensa y valor Q que harán parte del ejercicio (ver Figura 4) y se procede a realizar los siguientes pasos:

Tabla 2.

Parámetros de ejemplo Q-Learning

Parámetro	***Configuración inicial***
Tasa de Aprendizaje	0.7
Factor de Descuento	0.8
Factor de Exploración	1
Factor de Descomposición	0.1

Matriz R

	Salir del depósito	1	2	3	Retornar al depósito
Salir del depósito		-40	-20	-10	
1			-50	-30	60
2		-50		-60	80
3		-30	-60		90
Retornar al depósito					

Matriz Q en t=0

	Salir del depósito	1	2	3	Retornar al depósito
Salir del depósito	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
Retornar al depósito	0	0	0	0	0

Figura 4. Matriz de recompensa (R) y de valor Q

1. Se define el estado inicial S_0 = salir, es decir, salir del depósito.
2. Se establecen las posibles acciones del agente en S_0 , $A(S_0) = 1, 2, 3$, según matriz R
3. Se elige una acción a partir de $A(S_0)$ acorde con la política ε-greedy, $A_0 = 2$
4. Se salda el cliente 2 dado que CAP = 15 es mayor que DEM (2) = 5. Quedando como CAP = 10 Y DEM (2) = 0
5. $A_0 = 2$ determina el siguiente estado S_1 = 2
6. Posibles acciones a partir de S_1 son $A(S_1 = 2) = 1, 3$, retornar
7. Actualizar Q (S_0, A_0) como se observa en la Figura 5, producto de (10).

$$Q(S_t, A_t) = \ Q(S_t, A_t) + \ \alpha \ \left[R(S_t, A_t) + \gamma \max_A Q(S_{t+1}, A) - Q(S_t, A_t)\right]$$

$$Q(salir, 2) = \ Q(salir, 2) + \ \alpha \ \left[R(salir, 2), + \gamma \max_A Q(2, [1, 3, retornar]) - Q(salir, 2)\right]$$

$$Q(salir, 2) = \ 0 + \ 0.7 * [-20 + 0.8 * 0 - 0] \qquad (10)$$

$$Q(salir, 2) = -14$$

Matriz Q en t=1

	Salir del depósito	1	2	3	Retornar al depósito
Salir del depósito	0	0	-14	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
Retornar al depósito	0	0	0	0	0

Figura 5. Matriz Q en el estado S (1)

8. Se elige una acción a partir de $A(S_1)$ acorde con la política ε-greedy, $A_1 = 1$
9. Se salda el cliente 1 dado que CAP = 10 es igual que DEM (1) = 10. Es decir que la capacidad del vehículo se actualiza al valor CAP = 0 y la demanda del cliente a DEM (1) = 0. Siendo que su capacidad no le permite saldar más clientes, pasa directamente al depósito
10. $A_1 = 1$ determina el siguiente estado S_2 = 1
11. Posibles acciones a partir de S_2 son $A(S_2 = 1) = 2, 3$, retornar
12. Actualizar Q (S_1, A_1) como se observa en la Figura 6, producto de (11).

$$Q(S_t, A_t) = \ Q(S_t, A_t) + \ \alpha \ \left[R(S_t, A_t) + \gamma \max_A Q(S_{t+1}, A) - Q(S_t, A_t)\right]$$

$$Q(2,1) = \ Q(2,1) + \ \alpha \ \left[R(2,1), +\gamma \max_A Q(1, [2, 3, retornar]) - Q(2,1)\right]$$

$$Q(2,1) = \ 0 + \ 0.7 * [-50 + 0.8 * 0 - 0] \qquad (11)$$

Matriz Q en t=2

	Salir del depósito	1	2	3	Retornar al depósito
Salir del depósito	0	0	-14	0	0
1	0	0	0	0	0
2	0	-35	0	0	0
3	0	0	0	0	0
Retornar al depósito	0	0	0	0	0

$$Q(salir, 2) = -35$$

Figura 6. Matriz Q en el estado S (2)

13. Se retorna al depósito, $A_2 = retornar$

14. $A_2 = retornar$ determina el siguiente estado S_2 = retornar

15. No hay posibles acciones a partir de S_2.

16. Actualizar Q (S_2, A_2) como se observa en la Figura 7, producto de (12).

$$Q(S_t, A_t) = \ Q(S_t, A_t) + \ \alpha \ \left[R(S_t, A_t) + \gamma \max_A Q(S_{t+1}, A) - Q(S_t, A_t)\right]$$

$$Q(1, retornar) = \ Q(1, retornar) + \ \alpha \ [R(1, retornar), + \gamma \ (0) - Q(1, retornar)]$$

$$Q(1, retornar) = \ 0 + \ 0.7 * [60 + 0.8 * 0 - 0] \qquad (12)$$

$$Q(1, retornar) = 42$$

Matriz Q en t=3

	Salir del depósito	1	2	3	Retornar al depósito
Salir del depósito	0	0	-14	0	0
1	0	0	0	0	42
2	0	-35	0	0	0
3	0	0	0	0	0
Retornar al depósito	0	0	0	0	0

Figura 7. Matriz Q en el estado S (3)

Terminando así una corrida de se denota en la Figura 8 y Figura 9. También, se puede encontrar la programación del algoritmo propuesto en el siguiente repositorio: https://github.com/Stiven-Jaramillo/Q-Learning-for-CVRP.git.

1: **inicio de entrenamiento**

2: iniciar el agente con los parámetros de la instancia

3: definir los parámetros Q-Learning del agente

4: **para** corridas = 0,1,2, ..., N **hacer**

5: reiniciar el agente en estado inicial (salir del depósito)

6: reiniciar capacidad total del vehículo

7: reiniciar demanda para todos los clientes

8: **mientras** haya capacidad del vehículo suficiente para saldar al menos un cliente, **hacer**

9: seleccionar cliente a visitar por el agente usando política ε-greedy a partir del estado actual

10: actualizar matriz Q

11: **si** la capacidad es mayor que la demanda del cliente visitado, **hacer**

12: actualizar capacidad del vehículo restando la demanda del cliente visitado

13: hacer cero la demanda del cliente visitado

14: **fin si**

15: actualizar el estado del agente tomando como nuevo estado actual el cliente visitado

16: **fin mientras**

17: retornar agente al depósito

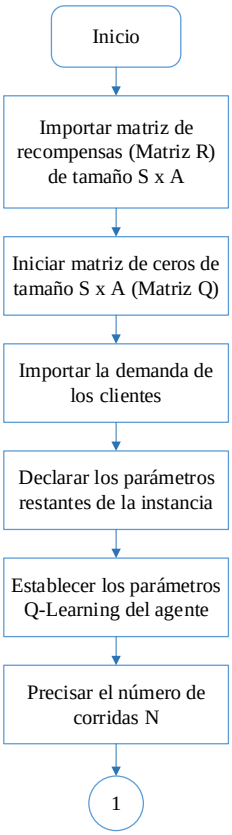
18: actualizar matriz Q

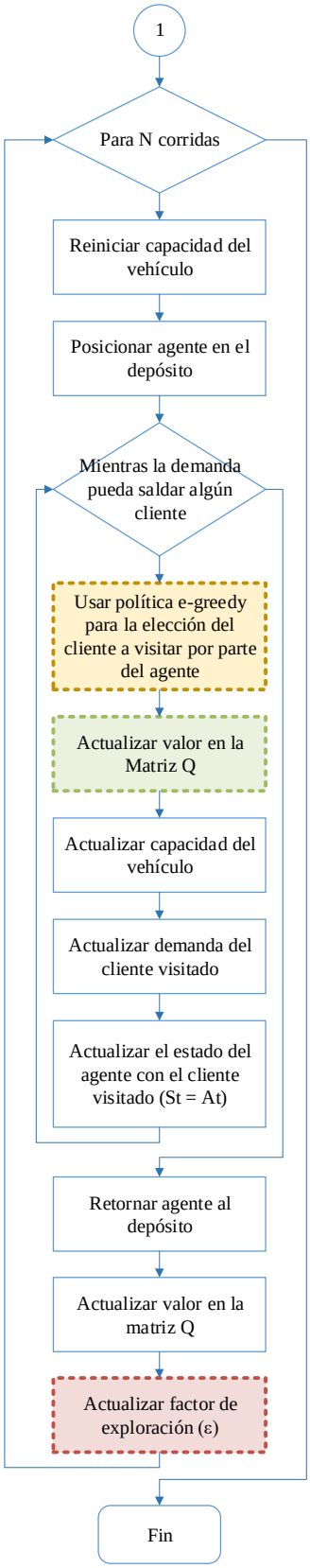
19: actualizar épsilon

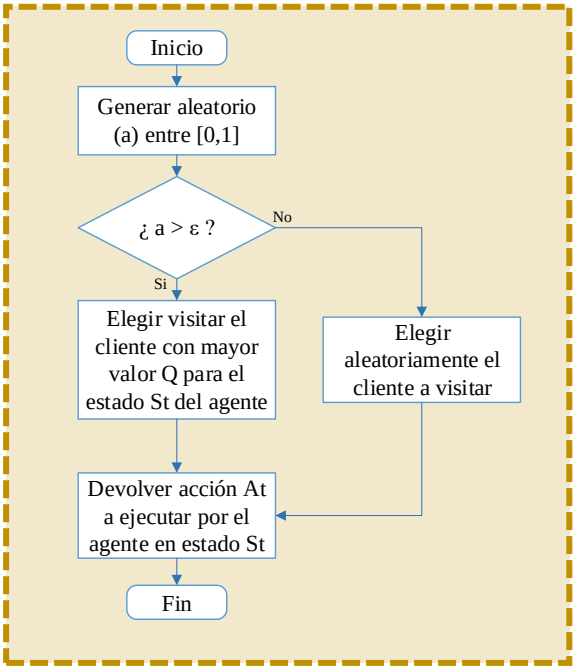
20: **fin para**

21: **fin de entrenamiento**

Figura 8. Pseudocódigo para entrenamiento del agente Q-Learning


Inicio
Importar matriz de recompensas (Matriz R) de tamaño S x A
Iniciar matriz de ceros de tamaño S x A (Matriz Q)
Importar la demanda de los clientes
Declarar los parámetros restantes de la instancia
Establecer los parámetros Q-Learning del agente
Precisar el número de corridas N
1


1
Para N corridas
Reiniciar capacidad del vehículo
Posicionar agente en el depósito
Mientras la demanda pueda saldar algún cliente
Usar política e-greedy para la elección del cliente a visitar por parte del agente
Actualizar valor en la Matriz Q
Actualizar capacidad del vehículo
Actualizar demanda del cliente visitado
Actualizar el estado del agente con el cliente visitado (St = At)
Retornar agente al depósito
Actualizar valor en la matriz Q
Actualizar factor de exploración (ε)
Fin


Inicio
Generar aleatorio (a) entre [0,1]
¿ a > ε ?
No
Si
Elegir visitar el cliente con mayor valor Q para el estado St del agente
Elegir aleatoriamente el cliente a visitar
Devolver acción At a ejecutar por el agente en estado St
Fin

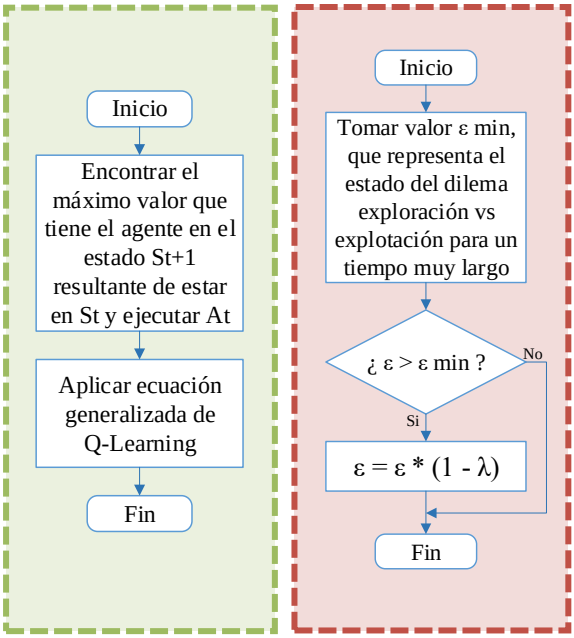


Figura 9. Diagrama de flujo del algoritmo Q-Learning.

7.2 Validación del Algoritmo

Ya desarrollado el algoritmo de aprendizaje, se procede a analizar si el método propuesto brinda soluciones a la formulación del CVRP. Para ello se procede en un primer momento a realizar un análisis estadístico de su desempeño y segundo, a solucionar un par de instancias de prueba tomadas del Set A (Augerat,1995) y Set M (Christofides, Mingozzi and Toth, 1979).

7.2.1 Primera etapa: análisis estadístico de parámetros. Para la validación del algoritmo se insistió en que el algoritmo aprendiera a reconocer las rutas que minimizan el costo en la función objetivo; sin embargo, otro objetivo es alcanzar este costo mínimo en el menor número de corridas posible. Por lo tanto, mediante el software estadístico Minitab 18 se realiza un diseño factorial 2^k, con el propósito de lograr mayor comprensión de la influencia de los parámetros en el desempeño del algoritmo, tomando como variable respuesta la relación entre costo mínimo obtenido y el costo

óptimo del problema (Score/Óptimo), y utilizando como factores los parámetros del algoritmo: tasa de aprendizaje (α), factor de descuento (γ) y factor de descomposición (λ); obteniendo así un diseño factorial completo de 2^3 y cinco (5) réplicas.

La instancia utilizada para el diseño se tomó del Set A (Augerat,1995), para 54 clientes, 9 vehículos, 100 de capacidad y un costo óptimo de 1073. Luego, se incluyen los niveles de cada factor (ver Tabla *3*) en el orden mencionado anteriormente, se ejecuta el código para 20000 corridas y en la Tabla *4* se consignan los resultados de la variable respuesta para cada combinación generada por el algoritmo Q-Learning.

Tabla 3.

Niveles de factor para el diseño factorial

Factor	***Nivel +1***	***Nivel -1***
Tasa de Aprendizaje	0.01	0.7
Factor de Descuento	0.01	0.8
Factor de Descomposición	0.001	0.3

Tabla 4.

Combinaciones en el diseño factorial

α	γ	λ	*OrdenEst*	*Score/Óptimo*
1	-1	-1	18	1.272
-1	-1	1	29	2.29
1	1	1	40	1.368
-1	-1	-1	25	2.286
-1	-1	1	37	2.299
-1	1	1	15	2.28
-1	-1	-1	17	2.32
1	1	1	24	1.381
1	-1	1	38	1.274
-1	-1	-1	1	2.377
1	1	-1	4	1.381
1	1	-1	36	1.338
1	1	-1	12	1.381
1	-1	1	30	1.312
1	-1	1	14	1.288
1	-1	-1	10	1.294
1	1	1	32	1.326
1	-1	1	22	1.271
-1	1	1	39	2.311
-1	1	1	23	2.264
1	-1	1	6	1.291
-1	-1	-1	33	2.334
1	-1	-1	34	1.28
-1	1	-1	27	2.286
-1	-1	-1	9	2.28
1	-1	-1	26	1.272
-1	-1	1	21	2.286
-1	1	1	31	2.29
1	1	-1	20	1.381
-1	1	1	7	2.393
-1	1	-1	35	2.326
-1	-1	1	13	2.333
1	1	1	8	1.381
1	1	-1	28	1.322
-1	1	-1	11	2.361
-1	1	-1	19	2.32
1	1	1	16	1.381
-1	-1	1	5	2.275
-1	1	-1	3	2.318
1	-1	-1	2	1.282

Ya consignados los valores de la variable respuesta, se realiza un análisis del diseño factorial y en la Figura 10 se observa que la distribución de los residuos para todas las observaciones

correspondientes a la variable respuesta se comporta aparentemente como una distribución normal, que son simétricos e independientes entre sí.

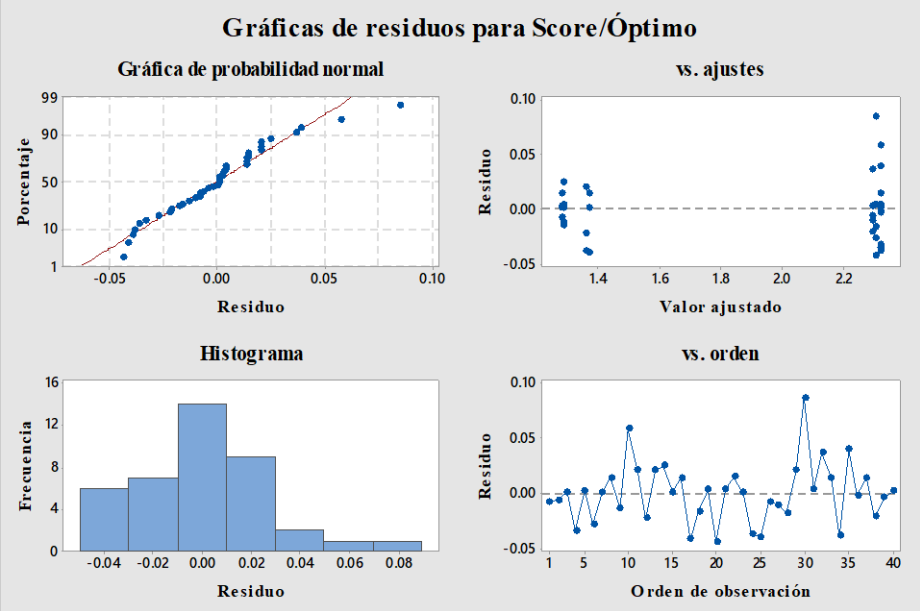


Figura 10. Gráficas de residuos para la variable respuesta. Adaptado de Minitab 18

Sin embargo, dada la incertidumbre en la gráfica de residuos vs ajustes, se procede a realizar la prueba de Levene con nivel de significancia de 0.05 para comprobar la premisa de varianza constante; en la cual se obtiene un valor p de 0.671, asegurando que los residuos sí cumplen con la premisa (ver Figura 11).

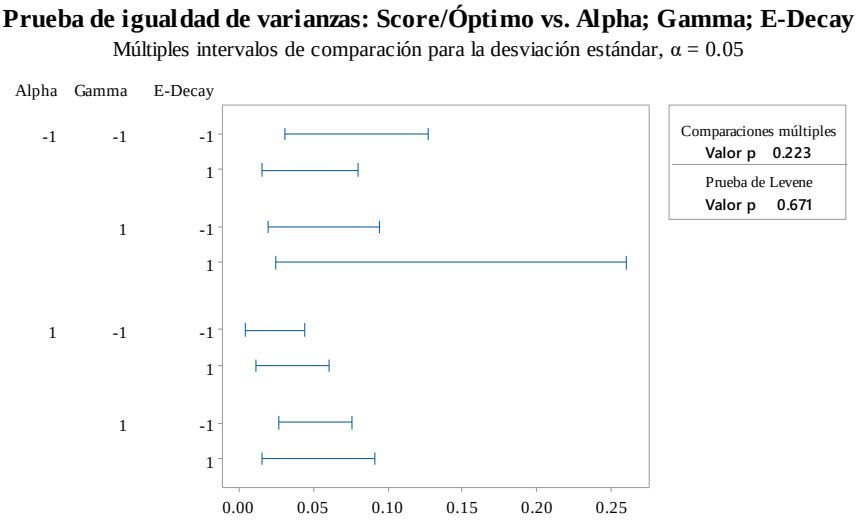


Figura 11. Prueba de igualdad de varianzas. Adaptado de Minitab 18

Además, con el análisis se determina que la tasa de aprendizaje es el parámetro que tiene mayor incidencia sobre la función objetivo y que la incidencia del factor de descomposición para el algoritmo Q-Learning no es significativa (ver Figura 12).

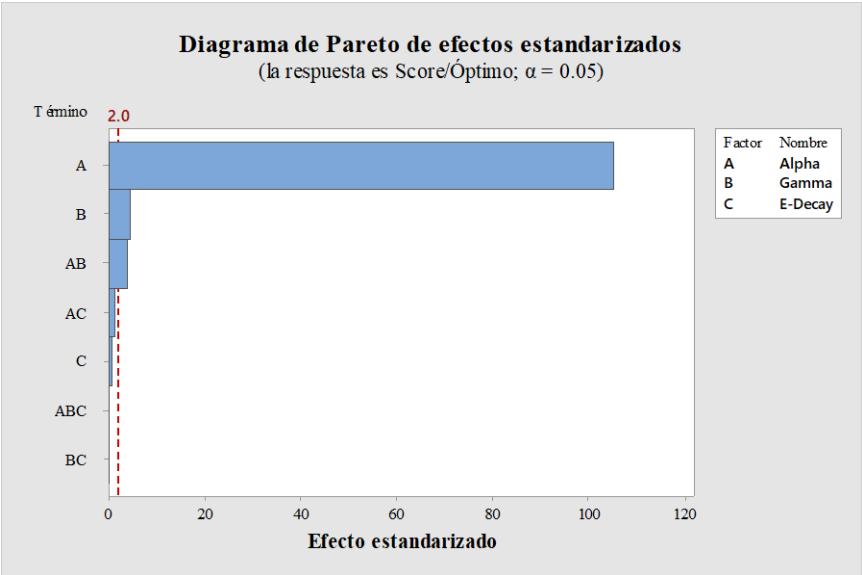


Figura 12. Pareto para efectos de factor en la variable respuesta. Adaptado de Minitab 18

Por lo tanto, se toma el ajuste de parámetros correspondiente a la combinación 1, -1, 1 para los factores tasa de aprendizaje (α), factor de descuento (γ) y factor de descomposición (λ) respectivamente, del cual el algoritmo obtuvo un valor de respuesta (Score/Óptimo) de 1.271 veces el óptimo, es decir, un costo mínimo de 1364 respecto a 1073 de costo óptimo para la instancia tomada; y alcanzó el costo mínimo luego de la corrida 192, tal como se muestra en la Figura 13 y

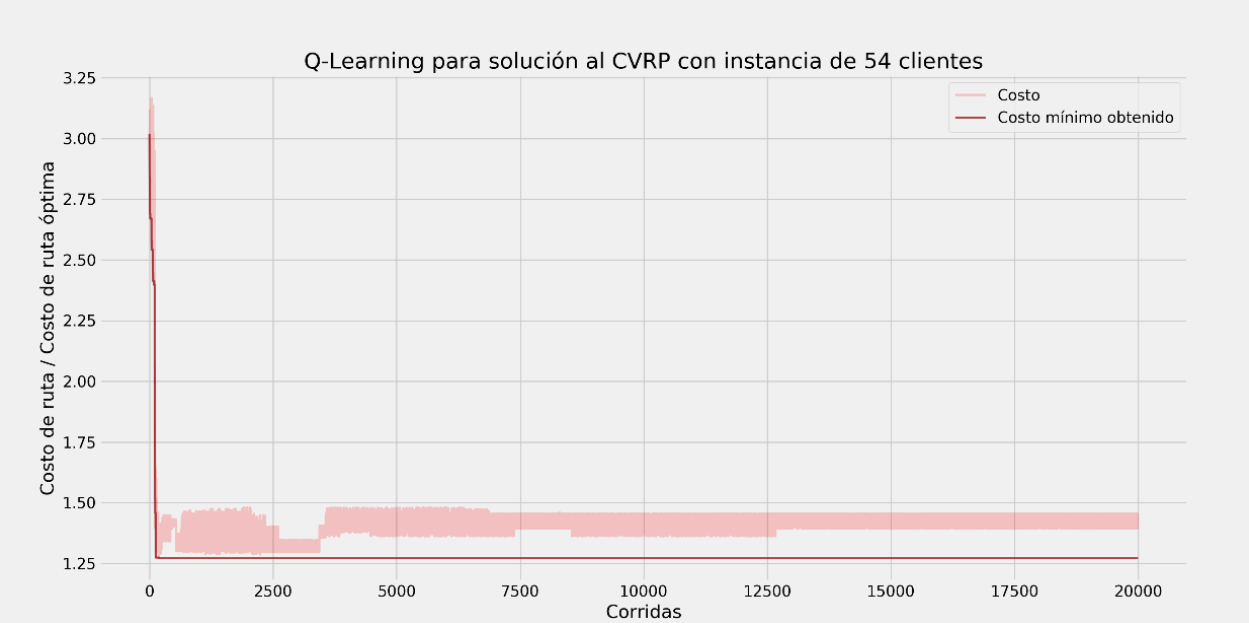


Figura 14.

Figura 13. Mejor resultado obtenido del diseño factorial.

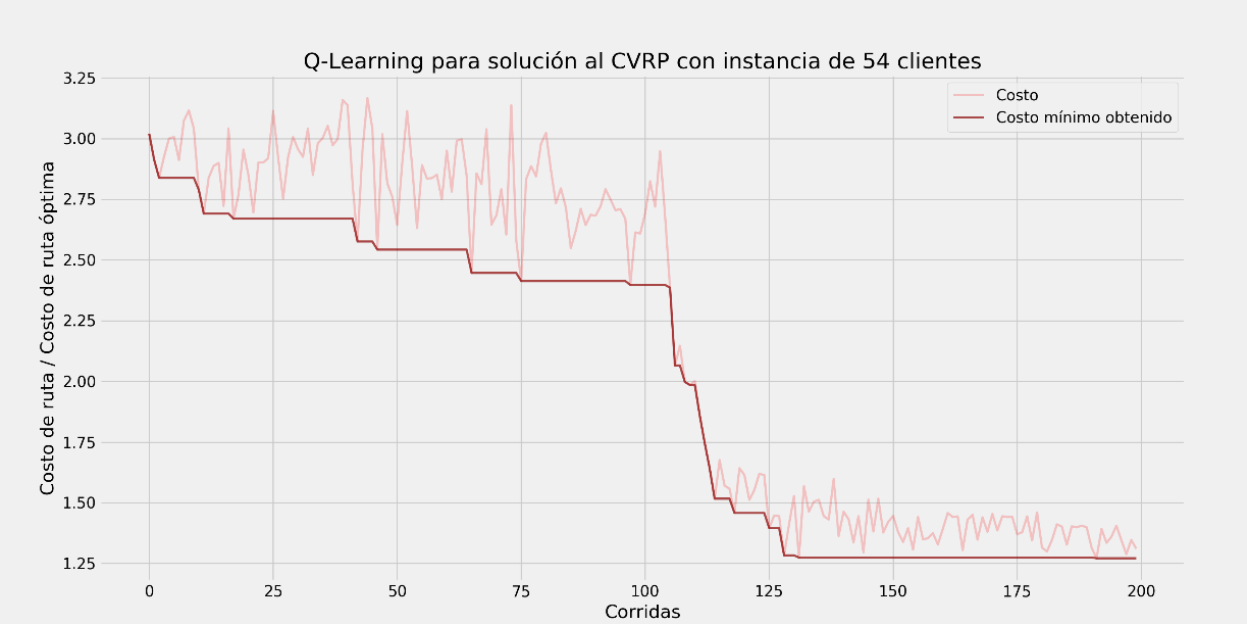


Figura 14. Mejor resultado obtenido del diseño factorial (primeras 200 corridas).

Siendo así, que de los resultados anteriores se obtuvieron las rutas mostradas en la Figura 15, incluyendo la demanda saldada por ruta y la demanda total satisfecha.

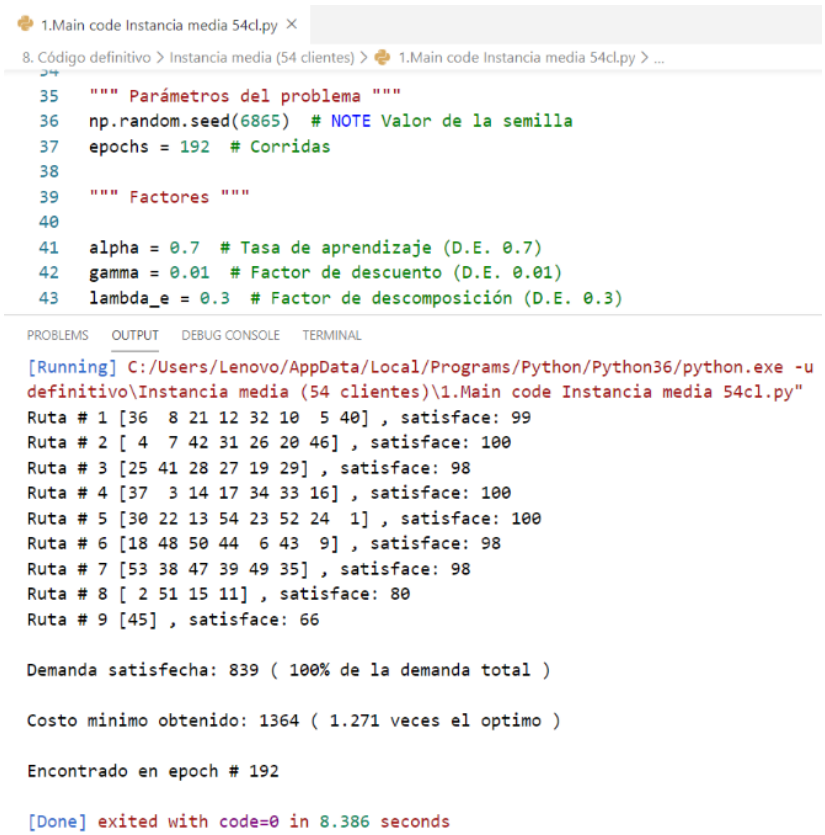


Figura 15. Tabla de resultados de instancia de 54 clientes

7.2.2 Segunda etapa: desempeño del algoritmo. Para comprender el comportamiento del algoritmo respecto a los factores empleados en el diseño anterior, se realiza una prueba que muestra cómo cambia la curva de aprendizaje del algoritmo a medida que varían sus parámetros. En las pruebas presentadas a continuación los valores tomados para evaluar los parámetros son tomados del diseño factorial y un punto central entre los valores alto y bajo del mismo.

Se inicia la prueba con la tasa de aprendizaje (α) para conocer su relación con el desempeño del algoritmo, en la Figura 16 se observa los resultados utilizando los valores de 0.7, 0.35 y 0.01 tomados del diseño de experimentos.

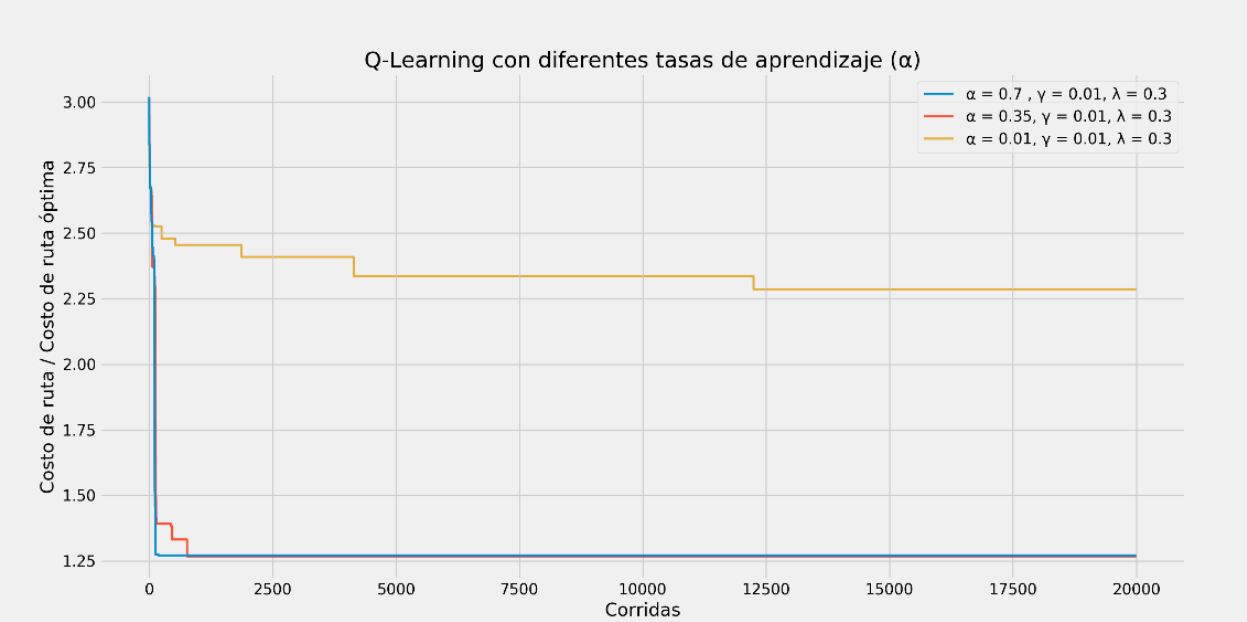


Figura 16. Resultados de Q-Learning usando diferentes tasas de aprendizaje

De la prueba se observa que utilizar valores muy pequeños en la tasa de aprendizaje ($\alpha = 0.001$; línea amarilla) no genera un decrecimiento rápido en el costo mínimo obtenido por el algoritmo. Por el contrario, entre más altos los valores α de la prueba mejor era el desempeño de la relación costo de ruta / costo óptimo.

Se realiza la misma prueba para el factor de descuento (ver Figura 17) con el fin de ampliar el entendimiento sobre el efecto en el desempeño de la variable respuesta, donde los valores a evaluar fueron γ = 0.01, 0.4 y 0.8.

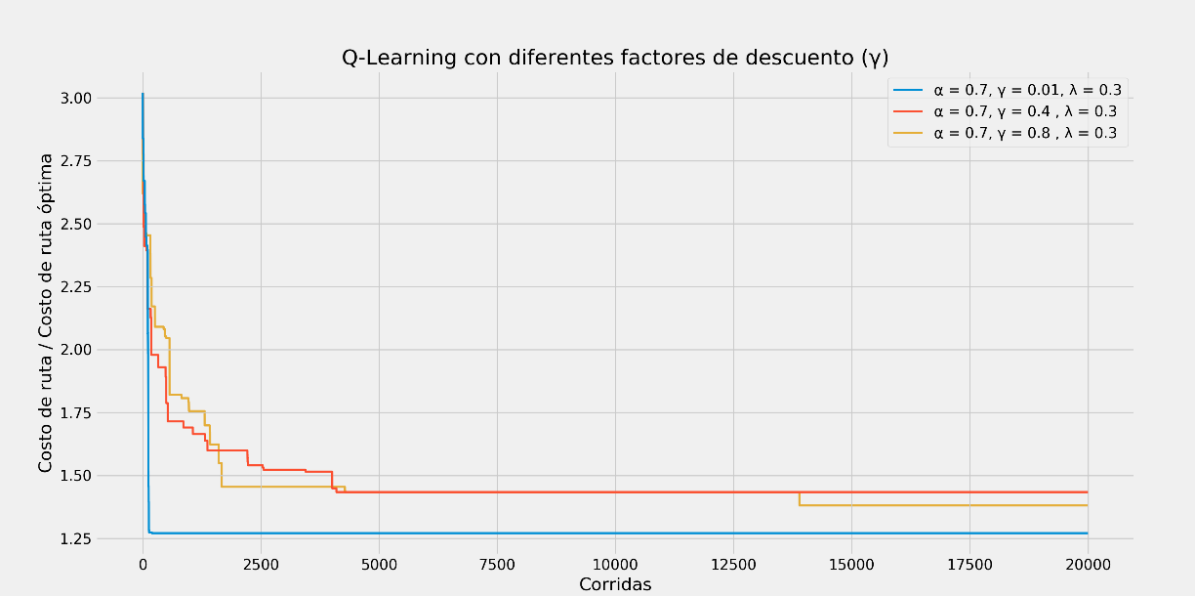


Figura 17. Resultados de Q-Learning usando diferentes factores de descuento

La gráfica anterior sugiere un comportamiento opuesto al visto en la tasa de aprendizaje; lo que indica que para valores γ pequeños, el desempeño del algoritmo es mejor y obtiene más rápido el costo mínimo.

Finalmente, se evalúa el efecto de variar la probabilidad de que el algoritmo Q-Learning explore el ambiente. Para lo cual se toma el factor de descomposición con los valores de λ = 0.3, 0.15 y 0.001, tal como se observa en la Figura 18.

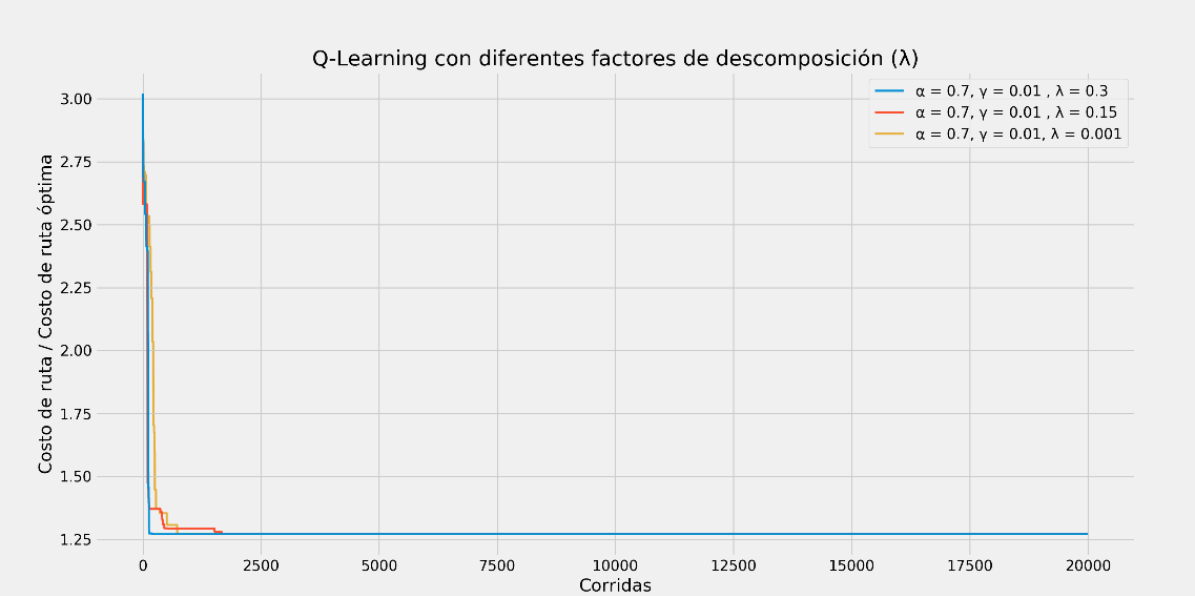


Figura 18. Resultados de Q-Learning usando diferentes factores de descomposición

Para esta prueba inicialmente se probaron otros valores de λ, que sumados a los resultados obtenidos de la gráfica anterior indican que este parámetro debe contar con valores pequeños que le permitan al agente explorar en mayor medida posibles opciones dentro del problema en cuestión. Asimismo, se obtuvo como resultado que para $\lambda = 0.3$ la relación costo obtenido / óptimo decae de manera más pronunciada que para los otros valores.

Además, con un ajuste de parámetros propuesto (ver Tabla *5*), el algoritmo Q-Learning logra encontrar soluciones factibles a la instancia de literatura seleccionada del Set A (Augerat,1995), para 79 clientes, 10 vehículos, 100 de capacidad y un costo óptimo de 1763; y en un ejercicio realizado para 20000 corridas, el algoritmo aprende la ruta más cercana a la óptima a partir de la corrida 19076, donde alcanza un costo mínimo de 2344, es decir, 1.237 veces el costo óptimo.

Tabla 5.

Ajuste de parámetros propuesto

Parámetros	***Valor***
Tasa de Aprendizaje	0.7
Factor de Descuento	0.8
Factor de Exploración	1
Factor de Descomposición	0.0005

Como resultado de la prueba, en la Figura 19 y Figura 20 se muestra que a medida que actualiza su valor Q en cada corrida, el algoritmo va aprendiendo cuál es la ruta que lo conduce cada vez más cerca del costo mínimo.

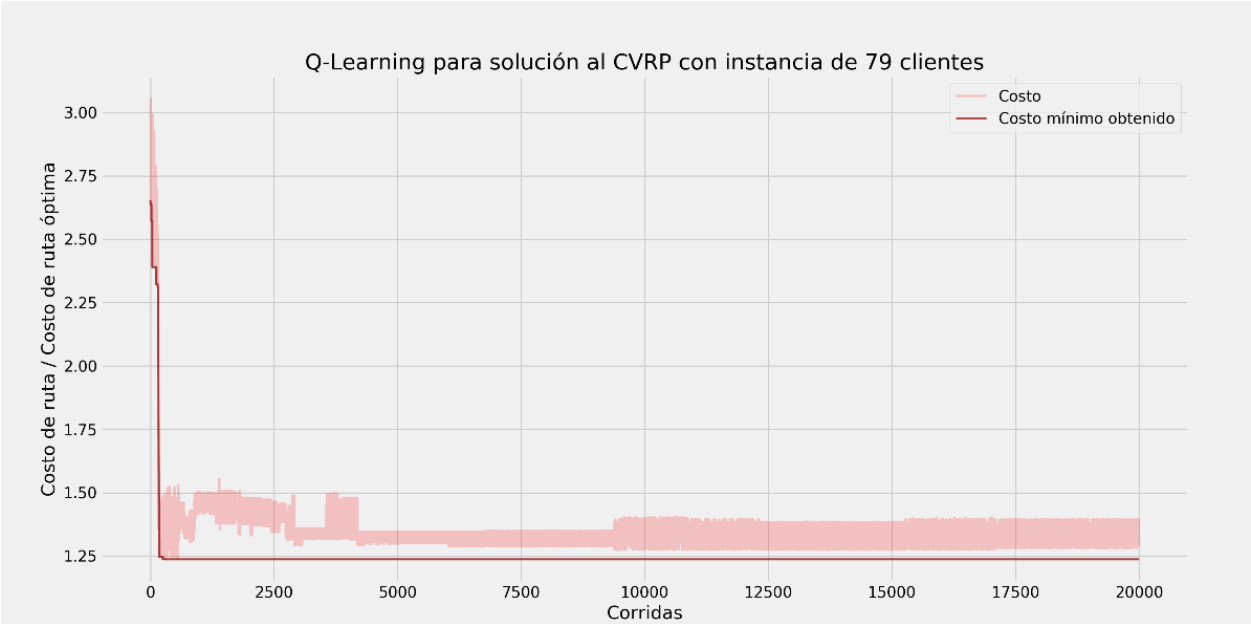


Figura 19. Costo obtenido/óptimo para instancia de 79 clientes

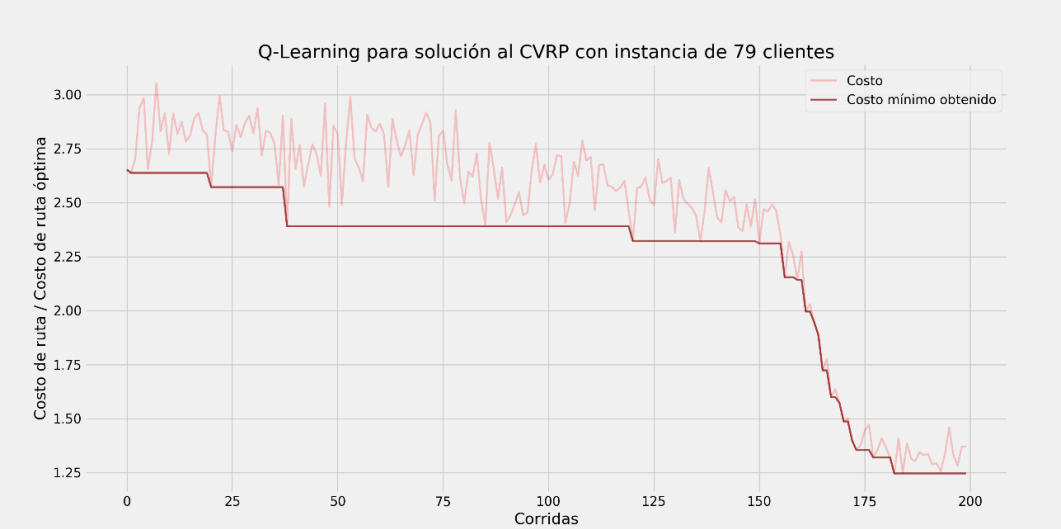


Figura 20. Costo obtenido/óptimo para instancia de 79 clientes (primeras 200 corridas)

Luego, para el mismo ajuste propuesto de la instancia anterior se realiza una prueba adicional para una instancia de 199 clientes, tomada del Set M (Christofides, Mingozzi and Toth, 1979), con parámetros de capacidad vehicular de 200 unidades, 17 vehículos disponibles y valor óptimo 1275, para determinar el comportamiento del algoritmo en una instancia mayor a 100 clientes (ver Figura 21); en el cual, alcanza un costo mínimo de 1817 en la iteración 19068, siendo 1.425 veces el óptimo.

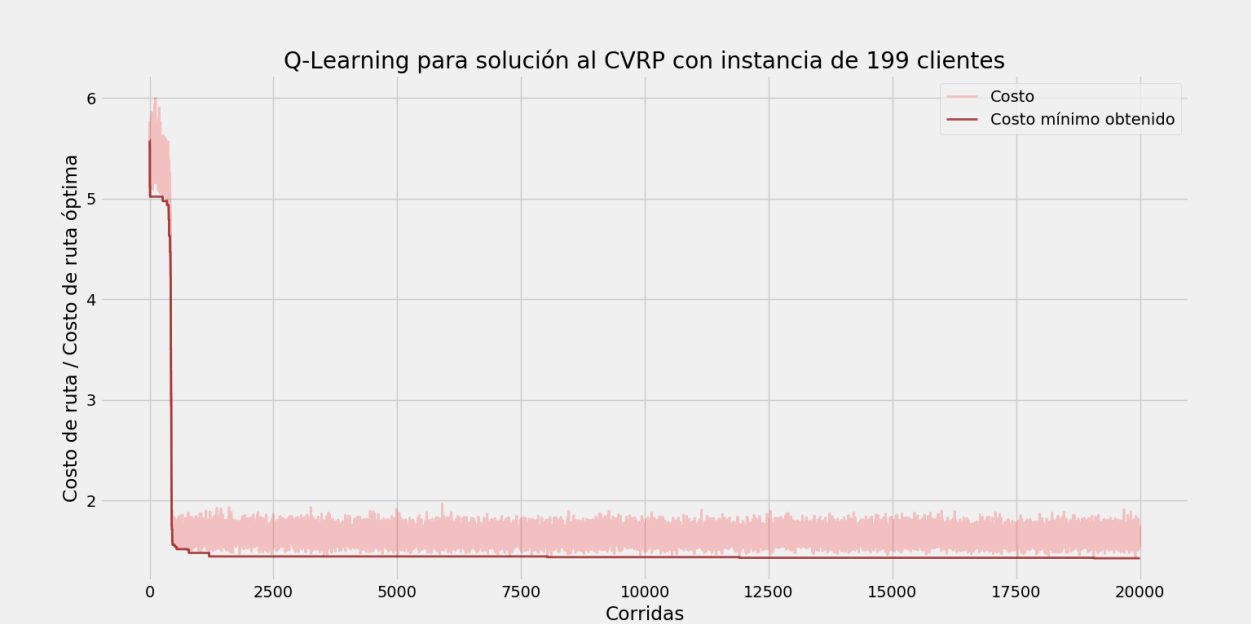


Figura 21. Costo obtenido/óptimo para instancia de 199 clientes

Finalmente, se presenta una comparación para las primeras 1000 corridas entre el costo obtenido/óptimo contra el número de corridas en las tres instancias anteriores (ver Figura 22) con el ánimo de presentar una relación entre la complejidad de la instancia y el valor obtenido de la solución propuesta. Comparación en la que se identifica que el descenso más pronunciado en cada instancia se da a mayor número de corridas conforme aumenta el tamaño de la instancia.

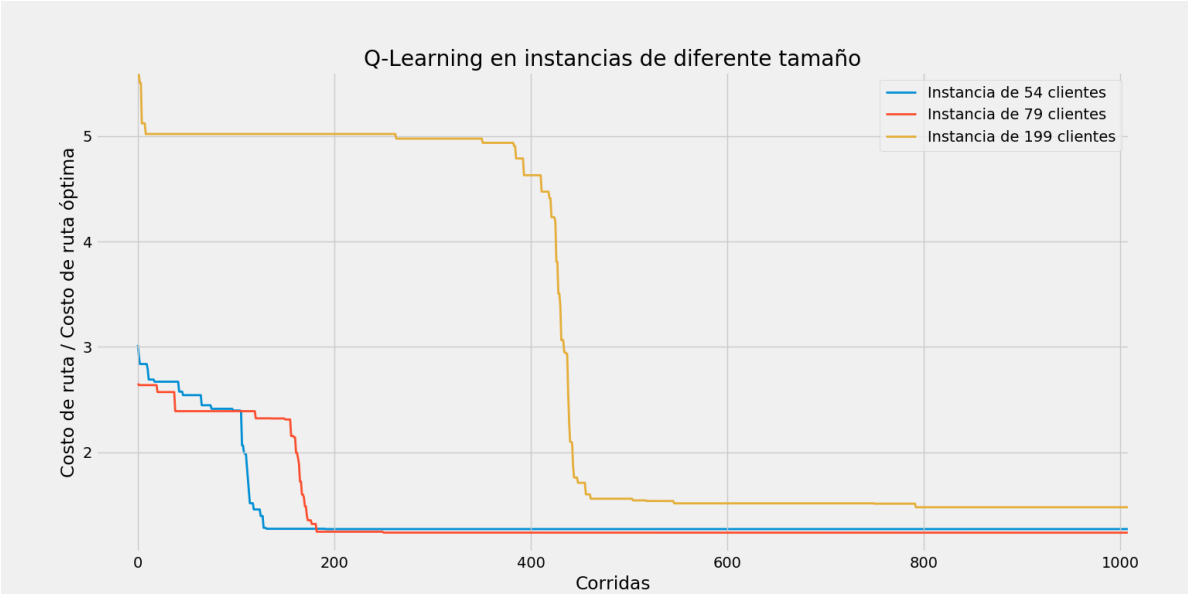


Figura 22. Comparación de instancias con 54, 79 y 199 clientes

8. Conclusiones

Con base en la revisión de la literatura se logra identificar el interés de la comunidad científica en encontrar soluciones ágiles a problemas de alta complejidad computacional como el CVRP; sin embargo, se evidencia que son pocos los autores que proponen modelos de aprendizaje reforzado a problemas de ruteo de vehículos, por lo que este proyecto genera la oportunidad de desarrollar futuras investigaciones en este campo.

El algoritmo de aprendizaje reforzado Q-Learning desarrollado en lenguaje Python encuentra soluciones factibles para instancias en la literatura de 54, 79 y 199 clientes cumpliendo todas las restricciones del Problema de Ruteo de Vehículos Capacitado (CVRP) según formulación VRP4 de Toth & Vigo (2002). Sin embargo, conforme aumenta la complejidad del problema, el desempeño del algoritmo se ve reducido puesto que aumenta significativamente el número de corridas alcanzar valores cercanos al óptimo.

Del análisis estadístico de parámetros en la instancia de tamaño mediano, se encontró que el ajuste de parámetros del algoritmo varía según la instancia del problema; una combinación que minimiza el costo en una instancia determinada para el CVRP no minimiza necesariamente la función objetivo de una instancia distinta. Además, del diagrama de Pareto para efecto de factores en la variable respuesta, es posible concluir que la tasa de aprendizaje (α) es el parámetro con mayor efecto significativo en la minimización de la función objetivo.

La mejor combinación de parámetros hallada para la instancia de 54 clientes logra que el algoritmo aprenda la mejor ruta en la corrida 192 con un costo mínimo de 27.1% por encima del óptimo.

9. Recomendaciones

Para futuras investigaciones se sugiere:

Evaluar escenarios con instancias mayores (>200 clientes) para el mismo modelo matemático, que permita retar aún más el desempeño del algoritmo Q-Learning como método de solución de problemas de alta complejidad computacional.

Considerar otras restricciones asociadas al Problema de Ruteo de Vehículos que permitan un acercamiento mayor del modelo matemático a la realidad. Por ejemplo, intentar con la restricción de ventanas de tiempo o VRPTW (Zhang, Li, Zhang, & Lin, 2020) que evalúe intervalos de servicio y tiempos de espera del vehículo para visitas anticipadas a los clientes.

Realizar una nueva revisión para métodos de solución emergentes de aprendizaje reforzado como el aprendizaje profundo o Deep Learning que, a través de métodos de aproximación y corrección, permitan abarcar problemas de mayor complejidad y obtener mejores respuestas en un menor número de corridas, cómo es el caso de las redes neuronales profundas para la aproximación de la matriz Q (DQN).

Referencias Bibliográficas

Ahn, B.-H., & Shin, J.-Y. (1991). Vehicle-routeing with Time Windows and Time-varying Congestion. *Journal of the Operational Research Society*, *42*(2), 393–400. Recuperado de http://www.jstor.org/stable/3009290

Akhtar, F. (2017). *Practical Reinforcement Learning: Develop self-evolving, intelligent agents with OpenAI Gym, Python, and Java*. Packt Publishing.

Atienza, R. (2018). *Advanced Deep Learning with Keras*. *Packt Publishing*.

Bellman, R. (1954). The Theory of Dynamic Programming. *The Rand Corporation*, 27.

Bello, I., Pham, H., Le, Q. V., Norouzi, M., & Bengio, S. (2016). Neural Combinatorial Optimization with Reinforcement Learning, 1–15. Recuperado de http://arxiv.org/abs/1611.09940

Clarke, G., & Wright, J. W. (1964). Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, *12*(4), 568–581. https://doi.org/10.1287/opre.12.4.568

Cook, W., Cunningham, W., Pulleyblank, W., & Schrijver, A. (1998). Combinaotial Optimization. New York: John Wiley & Sons, Inc.

Dai, H., Khalil, E. B., Zhang, Y., Dilkina, B., & Song, L. (2017). Learning Combinatorial Optimization Algorithms over Graphs, (Nips). Recuperado de http://arxiv.org/abs/1704.01665

Dantzig, G. B., & Ramser, J. (1959). The Truck Dispatching Problem. *Management*, *6*(1), 80–91.

Dantzig, G. B., & Wolfe, P. (1960). Decomposition Principle for Linear Programs. *Operations Research*. https://doi.org/10.1287/opre.8.1.101

Dantzig, G., Fulkerson, R., & Johnson, S. (1954). Solution of a Large-Scale Traveling-Salesman

Problem. *Operations Research*, *2*(4), 393–410. https://doi.org/10.1109/ICET.2009.5353167

Eksioglu, B., Vural, A. V., & Reisman, A. (2009). The vehicle routing problem: A taxonomic review. *Computers and Industrial Engineering*, *57*(4), 1472–1483. https://doi.org/10.1016/j.cie.2009.05.009

Fisher, M. L., & Jaikumar, R. (1981). Generalized Assignment Heuristic for Vehicle Routing, *11*, 109–124.

Gagniuc, P. A. (2017). *Markov Chains: From theory to implementation and experimentation*. New Jersey: John Wiley & Sons, Inc.

Géron, A. (2017). *Hands-On Machine Learning with Scikit-Learn & TensorFlow* (1st ed.). O'Reilly Media.

Gillett, B. E., & Miller, L. R. (1974). A Heuristic Algorithm for the Vehicle-Dispatch Problem, (August 2015).

Kalvelagen, E. (2003). Branch-and-Bound Methods for an Minlp Model With Semi-Continuous Variables, 1–11.

Korte, B., & Vygen, J. (2012). *Algorithms and Combinatorics Volume 21* (5th Editio). Heidelberg: Springer.

Korte, B., & Vygen, J. (2018). *Combinatorial Optimization: Theory and Algorithms*. Bonn. https://doi.org/https://doi.org/10.1007/978-3-662-56039-6

Kumar, S. N., & Panneerselvam, R. (2012). A Survey on the Vehicle Routing Problem and Its Variants. *Intelligent Information Management*, *04*(03), 66–74. https://doi.org/10.4236/iim.2012.43010

Laporte, G. (2009). Fifty Years of Vehicle Routing. *Transportation Science*, *43*(4), 408–416. https://doi.org/10.1287/trsc.1090.0301

Lee, L. S., Majid, Z. A., Seow, H. V, & Broga, J. (2012). Ant Colony Optimization for Container Loading Problem Department of Mathematics, Faculty of Science , Nottingham University Business School , Faculty of Arts and Social Sciences , *8*(2), 169–175.

Little, J., Murty, K., Sweeney, D., & Karel, C. (1963). An Algorithm for the Traveling Salesman Problem. *Operations Research [online]*, *11*, 972–989.

Nakajima, H. (2017). The Solution of Combinatorial Optimization Problems Based on Reinforcement Learning, 78–82.

Nazari, M., Oroojlooy, A., Snyder, L. V., & Takáč, M. (2018). Reinforcement Learning for Solving the Vehicle Routing Problem, (NeurIPS). Recuperado de http://arxiv.org/abs/1802.04240

Poggi, M., & Uchoa, E. (2003). Robust Branch Cut and Price Algoritms. *Proceedings of the Conference Mathematical Program in Rio: A Conference in Honour of Nelson Maculan*, 10. Recuperado de http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.4.1278&rep=rep1&type=pdf

Polak, E. (1997). Optimization: Algorithms and consistent approximations.

Python, W. (2019). BeginnersGuide/Overview - Python Wiki. Recuperado el 21 de enero de 2020, de https://wiki.python.org/moin/BeginnersGuide/Overview

Sipser, M. (2013). *Introduction to the Theory of Computation*. (M. Lee, Ed.) (Third Edit). Canada: Cengage Learning.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd Editio). London: The MIT Press.

Szeto, W. Y., Wu, Y., & Ho, S. C. (2011). An artificial bee colony algorithm for the capacitated vehicle routing problem. *European Journal of Operational Research*, *215*(1), 126–135.

https://doi.org/10.1016/j.ejor.2011.06.006

Toth, P., & Vigo, D. (2002). *The Vehicle Routing Problem*. *Animal Genetics* (Vol. 39). Philadelphia.

Venture Beat. (2019). AI and machine learning dominate World Economic Forum's list of 2019 Technology Pioneers | VentureBeat. Recuperado el 9 de julio de 2019, de https://venturebeat.com/2019/07/01/ai-and-machine-learning-dominate-world-economic-forums-list-of-2019-technology-pioneers/

Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer Networks. *Advances in Neural Information Processing Systems*, *61*, 85–117. https://doi.org/10.1016/j.neunet.2014.09.003

Watkins, C. (1989). *Learning from delayed rewards*. https://doi.org/10.1016/0921-8890(95)00026-C

Yadav, U., Sharma, S. K., & Routroy, S. (2018). Vehicle routing problem: recent literature review of its variants. *International Journal of Operational Research*, *33*(1), 1. https://doi.org/10.1504/ijor.2018.10015367