CIMAT

Centro de Investigación en Matemáticas, A.C.

METAHEURÍSTICAS APLICADAS A LA RESOLUCIÓN DE PROBLEMAS DE RUTEO DE VEHÍCULOS CON CAPACIDAD

T E S I S

Que para obtener el grado de

Maestro en Ciencias

con Especialidad en

Computación y Matemáticas Industriales

Presenta

Oscar Miguel González Vázquez

Director de Tesis:

Dr. Carlos Segura González

Dr. Sergio Ivvan Valdez Peña

Autorización de la versión final

Guanajuato, Gto., 1 de Junio de 2018

Declaración de Autoría

Yo, Oscar M. GONZÁLEZ, declaro que esta tesis titulada "Metaheurísticas aplicadas a la Resolución de Problemas de Ruteo de Vehículos con Capacidad" y el trabajo presentado en ella son míos. Yo confirmo que:

- Este trabajo fue hecho totalmente o principalmente mientras estaba en candidatura para un título de investigación en este centro de investigación.
- Si alguna parte de esta tesis ha sido presentada anteriormente para un título o cualquier otra calificación en este Centro de Investigación o cualquier otra institución, ha sido claramente expresado.
- Cuando he consultado el trabajo publicado de otros, ha sido claramente atribuido.
- Donde he citado del trabajo de otros, la fuente siempre se da. Con la excepción de tales citas, esta tesis es completamente mi propio trabajo.
- Reconocí todas las principales fuentes de ayuda.
- Cuando la tesis se basa en el trabajo hecho por mí mismo junto con otros, he dejado en claro exactamente lo que otros hicieron y lo que yo mismo contribuí.

"Investigar es ver lo que todo el mundo ha visto, y pensar lo que nadie más ha pensado."

Albert Szent-Györgyi

CENTER FOR RESEARCH IN MATHEMATICS (CIMAT A.C.)

Abstract

Center for Research in Mathematics (CIMAT A.C.)
Computation Area

Master in Sciences with specialty in Computing and Industrial Mathematics

Metaheuristics applied to solve Capacitated Vehicle Routing Problems

by Oscar M. GONZÁLEZ

In this thesis, novel Evolutionary Algorithms (EAs) belonging to the family of Memetic Algorithms (MAs) are designed to solve some variants of the Vehicle Routing Problem. The first variant that is tackled is the Capacitated Vehicle Routing Problem with Time-Windows (CVRPTW). In this case, we introduce a new crossover operator which is compared with the classical Sequence-based Crossover (SBX) originally designed for the CVRPTW. This new crossover operator is called the Single Breaking-Point Sequence Based-Crossover (SBSBX) and its main advantage is to reduce the disruptive behavior of the original SBX operator. The reduction in the disruptive behavior facilitates the control of the diversity through the modification of the selection pressure in the parent selection phase. In fact, it is shown that with the SBX operator, the diversity cannot be controlled by modifying the parent selection phase. However, this is not the case with the novel SBSBX operator proposed. The proposed MA uses the Simulated Annealing (SA) as a trajectory-based method to enhance the individuals generated by the crossover operators. Additionally, we show that only operating on the parent selection phase is not enough to properly control the balance between exploration and exploitation and because of this reason some of the state-of-the-art methods face problems with the control of the diversity. Thus, an additional contribution of this thesis is to incorporate more recent strategies to control the diversity and to show that such proposals contribute significantly in the achievement of high-quality solutions. Specifically, the Replacement with Multiobjective based Dynamic Diversity Control strategy (RMDDC) is implemented to explicitly manage the diversity. The proposal is validated using the Solomon's Benchmark with 100 customers and the Homberger Benchmark with 200, 400 and 600 customers. New best-known solution could be generated in 14 instances. Taking into account that such instances have been tackled with a large amount of methods, this is an important achievement of this thesis.

In 2016, ORTEC[1] proposed a Capacitated Vehicle Routing Problem with Time-Windows, Unmatched pickups and deliveries, and Inventory restrictions (CVRPTWUI)

[1] ORTEC is a company that work in optimization software and analytic solutions.

based on a real life scenario. In this problem the days to serve the customers must be assigned — subject to some restrictions — and taking into account such assignment, a set of routes must be established for each day, i.e. a CVRP must be optimized for each day. This second problem is much more complex than the first one because it involves two interdependent components, which usually leads to deceptive, large-scale search spaces. This thesis analyzes the application of a method similar to the one developed for the CVRPTW in this much more complex problem. The way to solve it is using a MA with a local search based on hill-climbing and the RMDDC as survivor selection. In spite of the simplicity of the local search, the proposal reaches competitive results. In any case, in this second problem the current best-known solutions could not be attained, meaning that more actions are required to deal with the deceptiveness and larger search space of this problem. Thus, we can conclude that by using state-of-the-art operators with a proper management of diversity, academic problems can be properly solved. However, when dealing with complex large-scale and deceptive problems, using similar approaches is not enough to reach the best-known solutions. One of the reasons is that, while in the case of the CVRPTW a non-disruptive operator could be generated by extending some of the most well-known crossover operators, in the novel CVRPTWUI no specific crossover operators have been designed, so simple, non ad-hoc genetic operators, were applied.

CENTRO DE INVESTIGACIÓN EN MATEMÁTICAS A.C. (CIMAT A.C.)

Resumen

Centro de Investigación en Matemáticas
Area de Computación

Maestría en Ciencias con especialidad en Computación y Matemáticas Industriales

Metaheurísticas aplicadas a la Resolución de Problemas de Ruteo de Vehículos con Capacidad

por Oscar M. GONZÁLEZ

En esta tesis, se diseña un nuevo algoritmo evolutivo perteneciente a la familia de los algoritmos meméticos (Memetic Algorithms — MAs) para resolver algunas variantes del Problema del Ruteo de Vehículos. La primera variante que se aborda es el Problema del Ruteo de Vehículos con Capacidad y Ventanas de Tiempos (CVRPTW - Capacitated Vehicle Routing Problem with Time-Windows) en donde introducimos un nuevo operador de cruza que es comparado con el operador de cruza basado en secuencia (Sequence-Based Crossover - SBX) que fue diseñado para el CVRPTW. El operador de cruza propuesto es el operador de cruza de un punto en común de ruptura basado en secuencia (Single Breaking-Point Sequence Based-Crossover - SBSBX) y su principal ventaja es reducir el comportamiento destructivo que el operador SBX presentaba. La reducción del comportamiento destructivo facilita el control de la diversidad a través de la modificación de la presión de selección en la etapa de la selección de padres, de hecho, se muestra que con el operador SBX la diversidad no puede ser controlada mediante la modificación del operador de selección de padres. Sin embargo, este no es el caso con el operador SBSBX propuesto. El MA propuesto usa el recocido simulado (Simulated Annealing - SA) como método de trayectoria para mejorar los individuos generados por el operador de cruza. Se muestra que modificar exclusivamente la etapa de selección de padres no es suficiente para controlar de forma adecuada el balance entre la exploración y la explotación y, por esta razón, algunos de los métodos del estado del arte tienen problemáticas relacionadas con el control de diversidad. Debido a esto, una contribución adicional de esta tesis consistió en incorporar estrategias de control de diversidad más recientes y mostrar que las propuestas contribuyen significativamente al alcance de soluciones de alta calidad. Específicamente, la estrategia de control de diversidad dinámica basada en conceptos multi-objetivo (Replacement with Multi-objective based Dynamic Diversity Control strategy - RMDDC) es utilizada para administrar la diversidad de forma explícita. La propuesta es validada usando las pruebas de desempeño de

Solomon con 100 clientes y las pruebas de desempeño de Homberger con 200, 400 y 600 clientes pudiéndose generar nuevas mejores soluciones para 14 instancias. Teniendo en cuenta que estas instancias han sido abordadas mediante una gran cantidad de métodos, se considera que este es un logro importante de la tesis.

En 2016, ORTEC[2] propuso un Problema de Ruteo de Vehículos con Capacidad y Ventanas de Tiempo, entregas y recolectas no asignadas e inventario con restricciones (Capacitated Vehicle Routing Problem with Time-Windows, Unmatched pickups and deliveries, and Inventory restrictions - CVRPTWUI) basado en datos reales. En este problema, se debe elegir los días en que se atenderán a los clientes —sujeto a algunas restricciones— y tomando en cuenta dichas asignaciones, se deben establecer un conjunto de rutas para cada día, es decir, se debe optimizar un CVRP para cada día. Este segundo problema es mucho más complejo que el primero debido a que involucra dos componentes interdependientes, lo que generalmente conduce a espacios de búsqueda engañosos y a espacios de búsqueda muy grandes. Esta tesis analiza la aplicación de un método similar al desarrollado para el CVRPTW para este problema. La forma de resolverlo es usando un MA con una búsqueda local por escalada y el RMDDC como método de selección de sobrevivientes. A pesar de la simplicidad de la búsqueda local, la propuesta alcanza resultados competitivos. Sin embargo, en este segundo problema no se pudieron alcanzar los mejores resultados conocidos, por lo que se puede concluir que se requieren más acciones para tratar con la deceptividad y el espacio de búsqueda tan grande de este problema. En consecuencia, se puede concluir que mediante el uso de operadores del estado del arte junto con un manejo apropiado de la diversidad se pueden resolver problemas académicos e incluso superar los mejores resultados conocidos hasta la fecha. Sin embargo, cuando se trata de problemas complejos de alta dimensionalidad y engañosos, estos enfoques no son suficientes para alcanzar las mejores soluciones conocidas. Una posible razón es que, mientras en el caso del CVRPTW se usaron operadores a medida no destructivos, en el nuevo CVRPTWUI no se diseñaron operadores de cruza específicos, aplicándose exclusivamente operadores genéticos simples y no desarrollados a medida para el problema.

[2] ORTEC es una compañia que trabaja en software de optimización y soluciones analíticas.

Agradecimientos

Quisiera agradecer a todas las personas que hicieron posible la realización de esta tesis, a mi madre, padre y hermanos por su motivación constante para la culminación de la maestría.

A mis tutores, el Dr. Carlos Segura González y el Dr. S. Ivvan Valdez Peña por haberme acompañado a lo largo de toda la investigación enriqueciéndola con sus experiencias, por sus comentarios y sugerencias sobre el trabajo realizado y, principalmente, agradezco la perseverancia con la que me guiaron durante la realización de esta tesis.

Por último, agradezco los apoyos que han sido otorgados por parte del CIMAT y el CONACYT pues sin ellos no hubiera sido posible llevar a cabo esta tesis.

Contenido

Lista de Figuras

Lista de Tablas

Lista de Abreviaciones

BKS	Best-known Solution
CP	Constraint Programming
CVRP	Capacitated Vehicle Routing Problem
CVRPTW	Capacitated Vehicle Routing Problem with Time-Windows
CVRPTWUI	Capacitated Vehicle Routing Problem with Time-Windows, Unmatched pickups and deliveries, and Inventory restriction
EA	Evolutionary Algorithm
EAX	Edge Assembly Crossover
FAP	Frequency Assignment Problem
GA	Genetic Algorithm
ILS	Iterated Local Search
KP	Knapsack Problem
MA	Memetic Algorithm
MIP	Mixed Integer Programming
RBX	Route Based-Crossover
RMDDC	Replacement with Multi-objective based Dynamic Diversity Control strategy
SA	Simulated Annealing
SBX	Sequence Based-Crossover
SBSBX	Single Breaking-Point Sequence Based-Crossover
TS	Tabu Search
TSP	Traveling Salesman Problem
VeRoLog	the Working Group on Vehicle Routing and Logistics Optimization within EURO
VRP	Vehicle Routing Problem

Dedicado a todas las personas que creyeron en mí.

Capítulo 1

Introducción

Los procesos de optimización surgen en una gran cantidad de situaciones de la vida real y particularmente en el área de logística hay una gran cantidad de situaciones que se abordan modelandolas como problemas de optimización. Este capítulo ofrece algunos conceptos básicos de optimización relativos a la definición de problemas y métodos, introduce el problema del ruteo de vehículos e identifica algunas problemáticas de esta área definiendo la hipótesis de estudio de esta tesis, así como los objetivos y contribuciones de la misma.

1.1 Optimización

Los problemas de optimización surgen en múltiples situaciones, por lo que es un campo de gran importancia tanto desde el punto de vista teórico como práctico. A modo de ejemplo, en gran cantidad de empresas, de forma continua se intenta hacer eficiente sus procesos para reducir costos y/o tiempos, produciéndose así una optimización de dichos procesos. En el área de las ciencias de la computación, la optimización se define como el proceso de encontrar la mejor solución posible para un determinado problema, denominando a dicha solución la óptima del problema [40]. Para medir la calidad asociada a las soluciones candidatas, es necesario definir una medida de desempeño, la cual se llama función objetivo. Si consideramos un problema de minimización, cuyas soluciones candidatas válidas están dadas por el conjunto S, las soluciones óptimas del problema se definen como aquellas soluciones $x* \in S$ tal que cualquier otra solución $x \in S$ es peor o igual que $x*$ con respecto a la función objetivo:

$$x* / f(x*) \leq f(x) \, \forall \, x \in S \tag{1.1}$$

Para los casos en que la función del problema tiene un óptimo único ($x*$), éste se puede definir de la siguiente forma:

$$x* / f(x*) < f(x) \, \forall \, x \in S; x \neq x^* \tag{1.2}$$

Aunque en algunas ocasiones los problemas de optimización se modelan como minimización, en otros casos, se modelan como problemas de maximización. Dado

que muchos optimizadores están implementados considerando sólo minimización — posiblemente porque la mayor parte de conjuntos de pruebas estándar se definen como problemas de minimización — es importante recalcar que un problema de maximización puede verse como un problema de minimización si negamos el valor de la función objetivo, es decir:

$$\underset{x}{\text{maximizar}} \quad f(x) \quad = \quad \underset{x}{\text{minimizar}} \quad -f(x)$$

Un ejemplo de problema de minimización es reducir los costos de producción de piezas de una empresa, mientras que uno de maximización podría ser aumentar las ganancias de venta de algún producto.

Dado que el campo de optimización es muy grande, han surgido múltiples clasificaciones tanto para los problemas de optimización como para los métodos que se usan para tratar los mismos. Una clasificación muy aceptada diferencia entre problemas de optimización continua y problemas de optimización discretos o combinatorios. En [40] se refiere a los problemas discretos como aquellos problemas en que el conjunto de soluciones candidatas es finito. En contraste, los problemas continuos tienen un conjunto infinito de soluciones candidatas, típicamente modelado como vectores con componentes reales. A modo de ejemplo, un caso de optimización continua es el de la función esfera, en la cual hay que encontrar el vector x que minimice $f(x) = \sum_{i=1}^{d} x_i^2$ con $x_i \in [-5.12, 5.12]$ donde cada variable de la función está en el dominio de los números reales. En el caso de optimización discreta el problema de la mochila (Knapsack Problema - KP) es un ejemplo en el cual hay que encontrar el conjunto de objetos para cargar en la mochila que maximicen la ganancia, donde cada objeto tiene asociado un valor y un peso y la mochila tiene un peso máximo que puede soportar. De esta forma, la solución está dada por el conjunto de objetos que maximicen la ganancia y no excedan el peso que se puede cargar en la mochila.

Otra clasificación muy popular se basa en la incorporación de restricciones [40]. En las definiciones anteriores, se ha supuesto que se dispone de un espacio de búsqueda S en el que todas las soluciones son válidas. Sin embargo, en algunos casos, algunas soluciones del espacio de búsqueda S no cumplen algunas restricciones que son requeridas para disponer de una solución factible. En dichos casos, se define el conjunto $\Omega \subseteq S$ como el conjunto de soluciones que cumplen con un conjunto de restricciones. Con base a ello, se pueden clasificar los problemas de optimización de la siguiente forma:

- **Sin restricciones:** en estos casos no hay que cumplir ninguna restricción en particular, por lo que $\Omega = S$.
- **Con Restricciones:** para este caso se distinguen tres tipos.
  - Restricciones de caja: en este caso cada una de las variables tiene valores mínimos y máximos que pueden tomar, de forma que $l_i \leq x_i \leq u_i \; \forall i = 1, ..., d$. Dado que en este caso es trivial crear soluciones que cumplan las

restricciones, muchos autores los consideran como problemas sin restricciones.

- Restricciones lineales, las cuales establecen relaciones lineales de igualdad o desigualdad entre las variables de decisión.
- Restricciones no lineales: frente al caso anterior establecen relaciones entre las variables que no tienen que ser lineales. Son las restricciones más difíciles de manejar.

En lo referente a los métodos de optimización se pueden clasificar como deterministas o estocásticos. En los métodos deterministas dada una entrada del algoritmo (problema a optimizar), el esquema siempre genera la misma solución y los pasos intermedios que se realizan para llegar a dicha solución son siempre los mismos. Por su parte en los métodos estocásticos o no deterministas, cada ejecución del algoritmo puede llegar a diferentes soluciones. Este tipo de comportamiento se puede conseguir por múltiples medios, siendo el más habitual usar un generador de números aleatorios para, en ciertos pasos de la ejecución, elegir una opción entre varias alternativas posibles para continuar con el proceso de optimización. Otro caso, es aquel en el que el problema de optimización es estocástico. En este caso, dada una solución x es posible que al evaluarla múltiples veces el valor de $f(x)$ difiera. Esto se puede deber a múltiples razones como que la función objetivo sea una función con ruido o que se estime el valor de $f(x)$ mediante una simulación no determinista. En dichos casos, incluso si se usa un método de optimización determinista, cada ejecución podría llegar a diferentes soluciones finales.

Hasta ahora, el problema se definió considerando que existe una única función objetivo que se quiere optimizar. A este caso se le llama optimización mono-objetivo. Sin embargo, en muchos casos es posible que se quiera optimizar de forma simultánea varias funciones objetivo que podrían tener conflictos entre sí. A dichos casos se les llama problemas de optimización *multi-objetivo*. Además, es habitual que a aquellos problemas en que se usen 4 ó más funciones objetivo, se les denomine problemas de *muchos-objetivos* [21]. Esto se debe a que muchos optimizadores que ofrecen buen rendimiento con unos pocos objetivos, no funcionan de forma adecuada al tratar problemas con 4 ó más objetivos, por lo que se suele introducir esta diferenciación entre ellos. Nótese que esta clasificación aplica tanto a los problemas de optimización como a los métodos de optimización, pues, por ejemplo, existen métodos que sólo son aplicables para casos con un único objetivo.

En el caso de problemas mono-objetivo, en general los optimizadores buscan una única solución que sea lo más próxima posible a la solución óptima. En el caso de optimización multi-objetivo encontrar una única solución que sea óptima para todas las funciones objetivo puede ser imposible debido a que una solución podría ser mejor que otra en el primer objetivo pero peor en el segundo objetivo. Para este tipo de optimización es habitual aplicar conceptos como el de dominancia de Pareto para indicar si una solución es mejor que otra. *Dado un problema de optimización a*

minimizar con k funciones objetivo, una solución $x \in \Omega$ domina a otra solución $y \in \Omega$ ($x \preceq y$), si se cumple que:

- *La solución x no es peor que la solución y en ningún objetivo:* $\forall i \in I = \{1, 2, \ldots, k\} : f_i(x) \leq f_i(y)$.
- *La solución x es estrictamente mejor que la solución y en al menos un objetivo:* $\exists i \in I = \{1, 2, \ldots, k\} : f_i(x) < f_i(y)$.

Con el uso del concepto de dominancia, se puede definir lo que es una solución óptima de Pareto. *Una solución $x* \in \Omega$ es óptimo de Pareto si no existe alguna solución $x \in \Omega$ que la domine.* El conjunto óptimo de Pareto $P*$ está conformado por el conjunto de soluciones óptimas de Pareto $P* = \{x* \in \Omega\}$. Con esas definiciones, entonces el Frente de Pareto podría ser definido como: *Dado un problema multi-objetivo $\vec{f}(x)$ y un conjunto óptimo de Pareto $P*$, el Frente de Pareto ($FP*$) se define como:*

$$FP* := \{\vec{u} = \vec{f} = (f_1(x), ..., f_k(x)) | x \in P*\} \tag{1.3}$$

La mayor parte de algoritmos diseñados para problemas multi-objetivo intentan aproximar el conjunto óptimo de Pareto.

Otra clasificación muy aceptada, desde el punto de vista del método de optimización, es por medio de la calidad de la solución que se obtiene, distinguiendo entre los métodos exactos y los métodos aproximados o heurísticos. Los primeros garantizan la obtención del óptimo global, encontrando a la programación dinámica como uno de los métodos de este tipo. Sin embargo, existen problemas con espacio de búsqueda tan grandes y complejos que los métodos exactos no son capaces de generar la solución óptima en un tiempo aceptable. En este caso se usan los métodos aproximados que suelen generar soluciones de alta calidad en tiempos razonables pero no garantizan encontrar el óptimo global.

Por último, cabe destacar que muchos optimizadores actuales se basan en la noción de **vecindario**. De acuerdo a [1], *una **estructura de vecindad** es una función $N : \Omega \to \mathcal{P}(S^1)$ que asigna a cada $x \in \Omega$ un conjunto de vecinos $N(x) \subseteq S$. $N(x)$ es llamado el vecindario de x.* En algunos optimizadores se definen estructuras de vecindad de forma explícita y se especifican diferentes estrategias para explorar las vecindades, en otros casos, se definen de forma implícita. Por ejemplo, muchos autores ven a los operadores de mutación de los algoritmos evolutivos como una forma de crear vecinos de una solución dada. Algunos esquemas se basan en mantener una única solución candidata en cada instante de tiempo, e irla transformando aceptando a alguno de sus vecinos. La aplicación del operador de generación de vecinos y aceptación de uno de ellos es comúnmente llamada **movimiento**. Una vez definida la noción de vecindad, algunos métodos tratan de encontrar un óptimo global, mientras que otros tratan de detectar de forma eficiente óptimos locales. Por ello, los

[1] S denota al conjunto de soluciones candidatas en un problema de optimización.

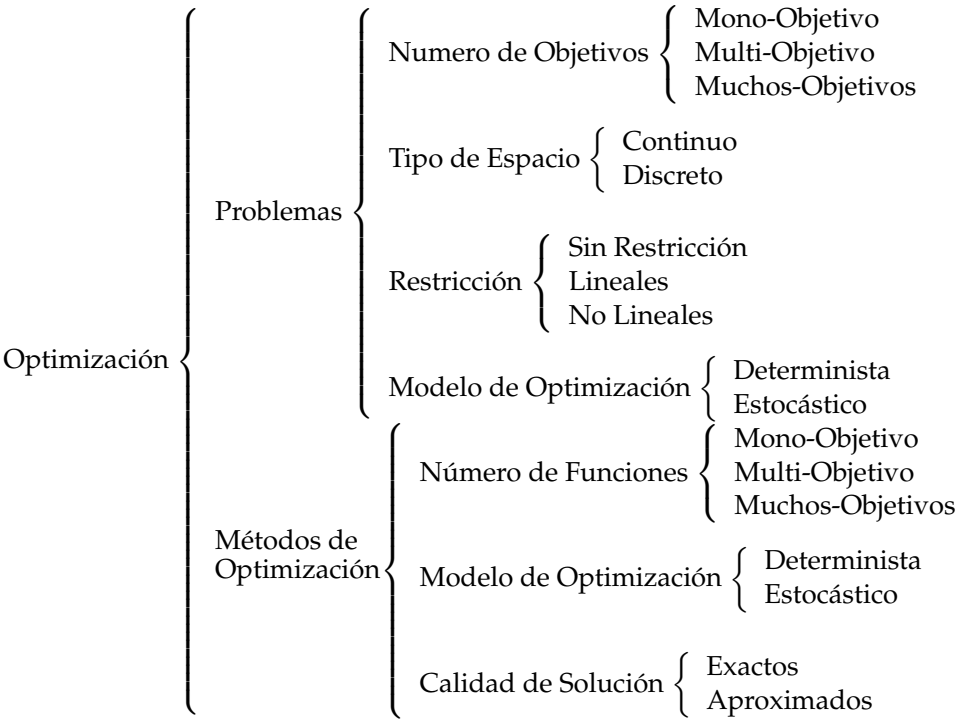


FIGURA 1.1: Clasificación de la optimización dependiendo del enfoque.

métodos pueden clasificarse como métodos de optimización globales o métodos de optimización locales.

En las descripciones anteriores, se han enumerado varias clasificaciones tomando en cuenta tanto los problemas de optimización como los esquemas de optimización. La figura 1.1 ofrece una panorámica de dichas clasificaciones.

El trabajo realizado en esta tesis se encuadra en el de los métodos de optimización aproximados con esquemas estocásticos de optimización. En lo referente al problema se tratan problemas con espacio de búsqueda discretos que tienen solo un objetivo, es decir, se trata un problema mono-objetivo discreto. Sin embargo, para el desarrollo del algoritmo de optimización se hace uso de algunos conceptos que surgen en el área de optimización multi-objetivo.

1.1.1 Algoritmos heurísticos

En el ámbito de algoritmos aproximados, los algoritmos heurísticos son de los más populares. Tal y como se explicó anteriormente, este tipo de algoritmos no aseguran un nivel de calidad en lo referente a las soluciones encontradas pero, incluso así, son los esquemas que han alcanzado las mejores soluciones conocidas para múltiples problemas de gran complejidad, en los que los esquemas exactos o de aproximación no se pueden aplicar en tiempos razonables. Los métodos heurísticos se clasifican en dos tipos. Por un lado, las heurísticas específicas son métodos que se diseñan

a medida para tratar un problema de optimización específico. Habitualmente, explotan alguna particularidad del problema, para tratar de generar de forma rápida soluciones que tengan una calidad aceptable. Por ejemplo, en el problema del viajante (TSP — Traveling Salesman Problem), un método heurístico podría elegir con mayor probabilidad aristas de bajo costo en cada paso, con el objetivo de minimizar el costo total de la ruta. Por otro lado, las metaheurísticas, el cual es el tipo de método en que se enfoca este trabajo, son métodos aproximados más generales aplicables a múltiples problemas de optimización. Para la aplicación de una metaheurística a un problema específico, se requiere diseñar e implementar un conjunto de métodos para el problema en cuestión. Por ejemplo, los algoritmos evolutivos que son una de las metaheurísticas más populares requieren que se definan métodos de cruce, que permitan a partir de dos soluciones crear otras nuevas soluciones que mezclen características de los padres, y a ser posible, que con mayor probabilidad compartan características buenas. Dado que estos componentes se definen de forma heurística, ya que no es posible normalmente encontrar la forma óptima de combinar dos soluciones, el esquema global recibe el nombre de metaheurística.

Tanto las heurísticas específicas como las metaheurísticas se pueden clasificar como métodos constructivos o métodos de mejora [18]. A continuación se describen estos tipos.

Heurísticas Constructivas

En [1], se definen a las heurísticas constructivas como algoritmos que generan nuevas soluciones candidatas desde cero, al ir agregando componentes a una solución parcial que inicialmente está vacía. El proceso de ir agregando componentes se repite hasta que la solución esté completa. Un ejemplo de este tipo de algoritmos es la heurística del vecino más cercano y la heurística del vecino más cercano de doble extremo, ambos usados con el TSP. La heurística del vecino más cercano (descrita en el Alg. 1) se caracteriza por ser un algoritmo sencillo y que se ejecuta de forma muy rápida por lo que es aplicable a casos grandes. La idea de este algoritmo se basa en ir uniendo los vértices más cercanos, para ello toman como referencia el último vértice visitado y lo unen al vértice más cercano y que no haya sido visitado aún. Sin embargo, su forma avariciosa de escoger la solución hace que el algoritmo no siempre obtenga valores de función objetivo adecuados.

Heurísticas de Mejora

En los métodos de mejora se comienza con una o varias soluciones candidatas completas y en cada paso se transforman las mismas, en muchos casos usando una definición de vecindad. El algoritmo más sencillo que es clasificado como método de mejora es la búsqueda local por escalada. En este caso se empieza con una solución completa — habitualmente generada de forma aleatoria — y sólo se aceptan movimientos hacia vecinos que sean mejores que la solución actual. Una vez que

Algoritmo 1: Algoritmo del vecino más cercano

1: Inicia en un vértice arbitrario como vértice actual V_c.
2: Marcar a V_c como visitado.
3: **while** Todos los vértices en el dominio no estén visitados **do**
4: Encontrar la arista más corta que conecte V_c con un vértice no visitado V.
5: Incluir la arista que conecta a V con V_c.
6: Asignar V como V_c.
7: Marcar a V como visitado.
8: **end while**
9: Incluir la arista que conecta el primer vertice visitado con el último vertice visitado.

se llega a un óptimo local el proceso de optimización finaliza. Cuando los algoritmos de búsqueda local hacen todos los movimientos definidos en la estructura de vecindad y la función objetivo ya no puede ser mejorada, se dice que el algoritmo se estancó en un mínimo local, surgiendo de ahí el nombre de búsqueda local. En general, los métodos de mejora son más lentos que los métodos constructivos, pero habitualmente llegan a soluciones de mejor calidad. Por ello, dependiendo del tiempo disponible para realizar la optimización, uno u otro tipo de método puede ser preferible. Algunas heurísticas de mejora más avanzadas consideran varias nociones de vecindad, realizan movimientos no basados en la noción de vecindad o aceptan movimientos que no tienen por qué ser de mejora. De hecho, la mayor parte de metaheurísticas de trayectoria se basan en incorporar mecanismos para evitar óptimos locales. Uno de los mayores inconvenientes que se les suele achacar a las heurísticas de mejora es que su efectividad suele depender en gran medida de la solución inicial.

1.1.2 Metaheurísticas

El término metaheurística está compuesto de dos términos griegos: heurística que viene del verbo *heuriskein* que significa "encontrar", y el prefijo *meta* que significa "más allá, en un nivel superior". Las metaheurísticas establecen un conjunto de mecanismos para guiar a un conjunto de heurísticas con el fin de llegar a soluciones de alta calidad. En [7], los autores dieron algunas de las propiedades fundamentales que caracterizan a las metaheurísticas:

- Las metaheurísticas son estrategias que "guían" el proceso de búsqueda.
- El objetivo es explorar eficientemente el espacio de búsqueda con el propósito de encontrar soluciones de alta calidad.
- Las técnicas que constituyen a las metaheurísticas van desde procedimientos simples de búsquedas locales hasta esquemas complejos con procesos de aprendizaje.

- Las metaheurísticas son algoritmos aproximados y, usualmente, no deterministas.
- Suelen incorporar mecanismos para evitar quedar atrapados en áreas pequeñas del espacio de búsqueda.
- Las metaheurísticas no son específicas para un problema.
- Las metaheurísticas pueden hacer uso específico del conocimiento del dominio en forma de heurísticas que son controladas por una estrategia de un nivel superior.
- Hoy en día, las metaheurísticas más avanzadas usan la experiencia adquirida durante el proceso de optimización (incorporadas en alguna forma de memoria) para adaptar los pasos posteriores de la búsqueda.

En el ámbito de las metaheurísticas surgen algunos conceptos que se usan ampliamente en esta tesis. Entre ellos cabe destacar los conceptos de *diversificación* e *intensificación* del espacio de búsqueda o región factible. La diversificación, también denominada exploración, se refiere a aquellas evaluaciones de soluciones candidatas que se realizan en regiones que son distantes a las de las soluciones que han sido evaluadas previamente por el algoritmo. Por su parte, los pasos de intensificación, también denominado explotación, se refiere a aquellas evaluaciones que se realizan en regiones del espacio de búsqueda que son cercanas a las de algunas soluciones que fueron evaluadas previamente. De esta forma la diversificación es útil para detectar regiones en el espacio de búsqueda con soluciones de alta calidad mientras que la explotación intenta buscar de forma más profunda en dichas regiones. Una de las claves para encontrar soluciones de alta calidad, es tener un balance apropiado entre las etapas de exploración y explotación. Debido a ello, algunos optimizadores como algunas variantes de la búsqueda Tabú, incluyen mecanismos explícitos para lanzar fases de exploración y de explotación, mientras que en otros optimizadores como los basados en algoritmos evolutivos, este proceso es implícito y depende en gran medida de los parámetros y componentes utilizados.

Finalmente, cabe destacar que en la actualidad existe una gran cantidad de metaheurísticas. Por nombrar algunas, cabe destacar la búsqueda tabú, el recocido simulado, la optimización por enjambres de partículas o los algoritmos evolutivos. A raíz de este gran crecimiento, han surgido numerosas clasificaciones que tratan de dar cierto orden a todos estos métodos. Entre estas clasificaciones una de las más importantes distingue entre algoritmos de trayectoria y algoritmos poblacionales, los cuales se introducen en las siguientes secciones.

Algoritmos de Trayectoria

Los algoritmos de trayectoria son una familia de algoritmos que surgieron con el afán de evitar la principal problemática que aparece en la búsqueda local, es decir, el estancamiento en óptimos locales. La principal característica de los algoritmos de trayectoria es que en cada instante mantienen una única solución candidata disponiendo de un conjunto de reglas para determinar a partir de la solución actual, y posiblemente de los movimientos realizados anteriormente, cuál será la nueva solución candidata. De esta forma, la solución actual realiza una especie de camino o trayectoria a través del espacio de búsqueda. Además, este tipo de esquemas almacenan la mejor solución encontrada durante todo el camino, siendo esta la solución que se reporta al final. Nótese que a diferencia de las búsquedas locales este tipo de esquemas no se quedan estancados en una solución, y por tanto, su criterio de parada no consiste en esperar a alcanzar un óptimo local, sino que habitualmente se establece en base al tiempo o número de soluciones evaluadas. También cabe destacar que en la mayor parte de casos los algoritmos de trayectoria hacen uso de una definición de vecindad, pero en muchos casos pueden aceptar nuevas soluciones que no pertenezcan a la vecindad o aceptar soluciones de la vecindad que sean peores que la solución actual.

Algoritmo 2: Estructura General de los Métodos de Trayectoria

Require: Solución Inicial S_0
1: $t = 0$
2: **while** no suceda el criterio de Parada **do**
3: Generar soluciones candidatas S_c a partir de S_t.
4: Seleccionar una solución S_s de S_c para reemplazar la solución actual S_t.
5: $S_{t+1} = S_s$
6: $t = t + 1$
7: **end while**
8: **return** Mejor solución encontrada

En el Alg. 2 se da el algoritmo general de un método de trayectoria. Estos algoritmos parten de una solución inicial que podría ser generada con cualquier método, por ejemplo, soluciones ya generadas por otros algoritmos, soluciones aleatorias, entre otras maneras, para posteriormente iterar hasta que un criterio de parada sea alcanzado (Paso 2). En cada iteración generan una solución candidata que reemplazará a la solución actual por medio de algún criterio (Pasos 3, 4 y 5), al final, los algoritmos dan la mejor solución encontrada (paso 8).

Algunos de los algoritmos de trayectoria más usados son el recocido simulado (Simulated Annealing - SA), la búsqueda Tabu (Tabu Search - TS) y la búsqueda local iterada (Iterated Local Search - ILS). Cada uno de estos algoritmos ha contribuido en cierta forma y/o para diferentes problemas y por tanto todos ellos siguen considerándose como parte del estado del arte. Por ejemplo, en TS se manejan fases de diversificación y explotación de forma explícita, lo que ha llevado a encontrar

soluciones competitivas con los resultados de los métodos del estado del arte [22], aunque esto a costa de tener que especificar varios tipos de memorias que dependen del problema en cuestión a utilizar. En el caso del SA, estas dos fases se entrelazan y se controlan a través de una temperatura, controlándose a través de un conjunto de mecanismos que no dependen del problema, por lo que de forma sencilla se puede adaptar el esquema a diferentes problemas de optimización. Por su parte, en el caso de la ILS se intensifica con búsqueda local y se explora con un mecanismo de perturbación que hay que adaptar en función del problema a utilizar. Particularmente, la perturbación muchas veces se basa en realizar un muestreo sesgado que tome en cuenta el historial de búsqueda. En general, las metaheurísticas de trayectoria avanzadas encuentran soluciones que son mucho mejores que las encontradas por las búsquedas locales. Sin embargo, en muchos problemas se ha visto que la solución alcanzada sigue dependiendo en gran medida de la solución inicial, pues a pesar de evitar óptimos locales en muchos casos la búsqueda se centra exclusivamente en la región en que comienza la búsqueda [60]. Por ello, en muchos casos se recurre a combinar este tipo de métodos con estrategias de reinicio, o se recurre a la utilización de metaheurísticas poblacionales.

Algoritmos Poblacionales

Los algoritmos poblacionales (Ver Alg. 3) son métodos de mejora que trabajan de forma simultánea con un conjunto de soluciones candidatas, en lugar de operar con una única solución tal y como hacen los algoritmos de trayectoria. A grandes rasgos, se comienza con un conjunto inicial de soluciones candidatas — habitualmente generadas de forma aleatoria — y en cada paso se genera un nuevo conjunto, y a través de un paso de selección de reemplazamiento, se decide cuáles soluciones de entre las anteriores y las nuevas se mantienen activas. Este proceso continúa hasta que se cumple un criterio de paro.

Algoritmo 3: Estructura General de los Algoritmos Poblacionales

1: Generar una población inicial $P = P_0$.
2: $t = 0$.
3: **while** no suceda el criterio de Parada **do**
4: Generar una nueva población P'_t a partir de P_t
5: Seleccionar una nueva población $P_{t+1} = Seleccionar(P_t \cup P'_t)$
6: $t = t + 1$
7: **end while**
8: **return** Mejor solución encontrada

Existen múltiples metaheurísticas poblacionales, como los algoritmos evolutivos (Evolutionary Algorithms - EAs), la búsqueda dispersa o los algoritmos de estimación de distribución. Entre ellos, los EAs son posiblemente los métodos más usados y se trata de un conjunto de algoritmos bio-inspirados que en sus versiones iniciales trataban de imitar el proceso de evolución que surge en la naturaleza. En

la naturaleza existe una diversidad de seres vivos pero solo los mejores adaptados sobreviven y se preservan en el tiempo. Este comportamiento de selección, reproducción y supervivencia fue utilizado para inspirar el diseño de los primeros EAs. Con el paso del tiempo, los optimizadores más avanzados cada vez se han ido separando más del proceso de evolución y hoy en día suelen incluir muchos aspectos que a pesar de no estar relacionados con lo que ocurre en la naturaleza, se ha visto que aportan en lo referente al proceso de optimización.

Algoritmo 4: Estructura General de los Algoritmos Evolutivos

1: Se genera una población inicial $P = P_0$.
2: $t = 0$.
3: **while** no suceda el criterio de Parada **do**
4: Evaluar P_t
5: Seleccionar población de padres P'_t =Seleccionar(P_t)
6: Cruzar a la población de padres para generar población de hijos P''_t = Cruzar(P'_t)
7: Mutar o perturbar P''_t
8: Evaluar P''_t
9: Reemplazar a la población P_{t+1} = Reemplazar(P_t, P''_t)
10: $t = t + 1$
11: **end while**
12: **return** Mejor solución encontrada

El algoritmo 4 muestra el pseudocódigo de un EA básico, donde en cada generación o iteración se realizan operaciones de selección de padres, cruce, mutación y reemplazo. Teniendo en cuenta dicho pseudocódigo, a la hora de diseñar y aplicar un EA se deben especificar los siguientes elementos:

- Representación o codificación de los individuos
- Función de fitness para evaluar cuáles son los individuos más adecuados
- Mecanismo de selección de Padres
- Operadores de variación: habitualmente recombinación y mutación
- Mecanismo de selección de sobrevivientes (reemplazo)

La representación es la forma en que se va a codificar cada solución candidata. En los algoritmos genéticos iniciales — una variante de EA — siempre se codificaban a los individuos con cadenas binarias, con lo que se debía realizar una transformación entre el espacio de búsqueda del problema y las cadenas binarias. Realizar dichas transformaciones no siempre es sencillo, y en muchos casos se vio que no ofrecía beneficios, por lo que hoy en día, no se tiene la limitación de usar este tipo de codificaciones, y en muchos casos se usan codificaciones más cercanas al espacio de búsqueda real. Por ejemplo, para problemas basados en buscar permutaciones, la

codificación puede estar basada en arreglos que precisamente mantengan una permutación de los elementos en cuestión.

La función fitness mide la calidad de los individuos. En muchos casos la función de fitness es la misma que la función objetivo, aunque en otros casos es una extensión de la misma. Por ejemplo, en problemas de optimización con restricciones es habitual usar una función de fitness que tome en cuenta la función objetivo pero que a la vez penalice el valor de la función en base a las restricciones que no se cumplan.

El mecanismo de selección de padres es el proceso en el cual se escoge uno o más individuos de la población para aplicar posteriormente ciertas operaciones, como podría ser la cruza o la mutación. Habitualmente, en este paso se selecciona con mayor probabilidad a aquellos individuos que tienen mayor valor de fitness, surgiendo el término *presión de Selección*. Un algoritmo que selecciona con mucha mayor probabilidad a los mejores individuos se dice que tiene una presión de selección alta, mientras que conforme el tipo de muestreo realizado se acerca a una distribución uniforme, se dice que la presión de selección disminuye. Algunos de los operadores de selección de padres más conocidos son el torneo binario, la selección por ruleta y la selección aleatoria, cada una de ellas con diferente presión de selección. En específico, el torneo binario selecciona dos individuos de forma aleatoria, los compara y elige como padre el individuo con mejor fitness. Por su parte, el algoritmo de la ruleta asigna una probabilidad para ser seleccionado de acuerdo al fitness aportado, es decir, si tuviéramos un problema de maximización cuya función de fitness siempre toma valores positivos, entonces la probabilidad de seleccionar un individuo I_x de un conjunto de N individuo viene dada por la ecuación 1.4. Por último, para la selección aleatoria, todos los individuos tienen la misma probabilidad de ser seleccionados.

$$P_{I_x} = \frac{f(I_x)}{\sum_i^N f(I_i)} \tag{1.4}$$

El papel de los operadores de variación es el de crear nuevos individuos a partir de los individuos seleccionados en la selección de padres [18]. Para ello, se aplica un conjunto de operadores, entre los que los más habituales son los operadores de recombinación — también denominados operadores de cruce — y los operadores de mutación. En el operador de recombinación se generan nuevos individuos a partir de dos o más individuos generando nuevas soluciones candidatas que heredan características de los padres. Para diseñar buenos operadores de cruce, es primordial detectar qué tipos de características son las que se deberían heredar, y existen desde operadores con un alto grado de aleatoriedad, hasta esquemas que en el proceso de cruce implementan un algoritmo de optimización. Así, hoy en día existe un gran abanico de posibilidades a la hora de cruzar individuos [4]. Otro operador de variación muy habitual es el operador de mutación. Este operador hace una perturbación — habitualmente pequeña — a las soluciones generadas por el operador de cruce. Al igual que en los operadores de cruce, este proceso se puede hacer de forma

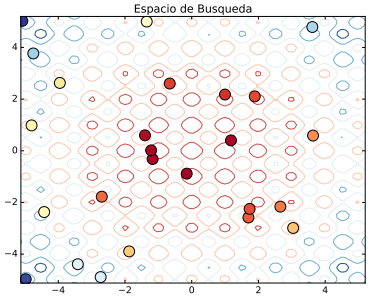
aleatoria — lo más típico en los algoritmos iniciales — o siguiendo reglas voraces para tratar de mejorar la solución generada.

El mecanismo de selección de sobrevivientes, también llamado fase de reemplazamiento, es la etapa donde se selecciona a los individuos que van a pasar a la siguiente generación. En [18], los autores discuten algunos de los operadores existentes como los basados en la edad o el reemplazo del peor (también llamado GENITOR). Cada uno trabaja de distinta forma, sin embargo, en el primero el mejor individuo — individuo élite — podría no ser preservado a lo largo de las generaciones. A través de la experiencia, se ha visto que mantener al individuo élite es muy importante, sin embargo, elegir sólo a los mejores individuos también crea inconvenientes importantes, por lo que se han definido esquemas más avanzados, algunos de los cuales serán usados en este tesis.

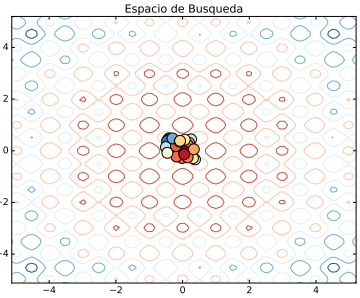
Los EAs han ofrecido muy buenos resultados en múltiples problemas de optimización. Sin embargo, también se han reportado múltiples inconvenientes al lidiar con algunos otros problemas. En esta tesis en concreto se ha lidiado con el problema conocido como *convergencia prematura* que se presenta cuando no hay un balance correcto entre la exploración y explotación, que en específico provoca que todos los individuos se localicen en una pequeña región del espacio de búsqueda mucho antes de alcanzar el criterio de paro. Al ocurrir esto, con una alta probabilidad, no se crearán nuevas soluciones fuera de dicha región, por lo que si no es una región de alta calidad, las soluciones finales encontradas no serán adecuadas. En la Fig. 1.2 se ejemplifica el problema de convergencia prematura. En la Fig. 1.2a se muestra la distribución inicial de los individuos en la que se visualiza que cubre diferentes regiones del espacio de búsqueda. En la Fig. 1.2b, que representa a las soluciones al 50% de la ejecución del algoritmo, se ve como todos los individuos se han enfocado en una pequeña región del espacio de búsqueda. Esto significa que con mucha probabilidad, durante el resto del 50% de la ejecución ya no se explorarán otras regiones, y en particular en este caso, no se detectaría el óptimo global. Finalmente, en la Fig. 1.2c, que representa a las soluciones presentes en la última generación se ve cómo los individuos se mantuvieron en esa zona.

Analizar cómo se distribuyen los individuos en el espacio de búsqueda, es decir, la diversidad de la población, podría no ser tan sencillo como parece. En el caso de optimización continua es habitual hacer uso de la distancia Euclidiana para analizar la diversidad, sin embargo, en problemas combinatorios hay que utilizar otras medidas de distancia, y no siempre es fácil definir medidas de distancia apropiadas.

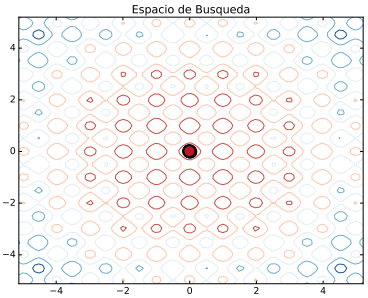
Finalmente, otro elemento que hay que especificar es el criterio de parada. Lo más habitual es usar criterios que tienen relación con la cantidad de cómputo utilizado como número de funciones de evaluación ejecutadas o el tiempo transcurrido. Sin embargo, dada la relación entre pérdida de diversidad y estancamiento, en muchos casos se toma en cuenta la diversidad poblacional como criterio de parada [48].



(A) Distribución inicial de los individuos en el espacio de búsqueda.



(B) Distribución de los individuos al 50% de la ejecución del algoritmo.



(C) Distribución Final de los individuos.

FIGURA 1.2: Distribución de los individuos con el problema de convergencia prematura.

También cabe destacar que existen muchos otros algoritmos poblacionales bio-inspirados que no están basados en la evolución, como los algoritmos de optimización basados en colonias de hormigas, así como algunos no bio-inspirados como la búsqueda dispersa. Se suele considerar a los algoritmos poblacionales como esquemas muy promisorios para detectar buenas regiones del espacio de búsqueda, pero no tan promisorios a la hora de encontrar muy buenas soluciones en dichas regiones. Dicho de otra forma, son buenos en explorar, pero no tan adecuados para intensificar. Por ello, es muy habitual diseñar algoritmos híbridos que mezclan métodos poblacionales con métodos de trayectoria. Entre ellos, los algoritmos meméticos integran un algoritmo de búsqueda local o una metaheurística de trayectoria — habitualmente llamado método de aprendizaje individual — en un algoritmo evolutivo. Particularmente, en sus versiones más sencillas incorporan dicho proceso tras la generación de nuevos individuos, es decir, cada individuo que se crea mediante cruce y mutación es mejorado utilizando el método de aprendizaje seleccionado. En esta tesis se trabaja con este tipo de algoritmos meméticos y su pseudocódigo básico se puede visualizar en el Algoritmo 5.

Algoritmo 5: Estructura General de un Algoritmo Memético Simple

1: Generar una población inicial $P = P_0$.
2: Evaluar a P_0
3: $t = 0$.
4: **while** no suceda el criterio de Parada **do**
5: Seleccionar Padres
6: Recombinar para producir descendencia
7: Mutar descendencia
8: Mejorar descendencia por medio de la búsqueda local o metaheurística de trayectoria
9: Seleccionar individuos para la siguiente generación P_{t+1}
10: $t = t + 1$
11: **end while**
12: **return** Mejor solución encontrada

1.1.3 Conjuntos de Desempeño

Un conjunto de desempeño es un conjunto de problemas de optimización — o instancias de un problema dado — que se utiliza para medir el comportamiento de un algoritmo. En el ámbito de optimización combinatoria es habitual crear conjuntos formados por instancias con diferentes características y de diferentes tamaños, que permitan llegar a conclusiones en lo referente al rendimiento de los optimizadores en diferentes ámbitos. Los conjuntos de desempeño más habituales consideran definiciones académicas de problemas de optimización, y están conformadas por problemas con características bien conocidas lo que permite realizar un mejor análisis de lo que está ocurriendo en el proceso de optimización. Sin embargo, para problemas más realistas podría no existir conjuntos de desempeño o se podría desconocer

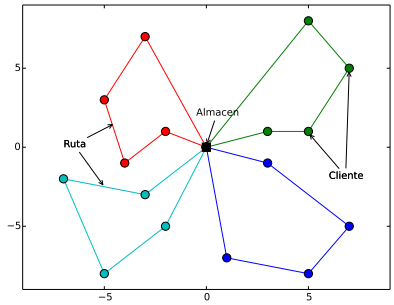


FIGURA 1.3: El VRP.

muchas de las características de los problemas incluidos en dichos conjuntos. A pesar de ello, utilizar dichos conjuntos es útil en el sentido que muchas veces incorporan problemas de gran complejidad. Por tanto, a la hora de validar algoritmos es bueno usar tanto conjuntos académicos, como problemas más realistas.

1.2 El Problema del Ruteo de Vehículos

El Problema de Ruteo de Vehículos (Vehicle Routing Problem - VRP) es un problema de optimización combinatoria que generaliza al Problema del Vendedor Viajero (Traveling Salesman Problem - TSP). El VRP está clasificado como problema NP-Hard, con lo que para instancias grandes, los enfoque basados en optimización heurística son los más habituales y los que han ofrecido mejores resultados. En especifico, el VRP en su versión más básica es un problema de minimización en el que se busca obtener un conjunto de rutas que inicien y finalicen en una localización llamada almacén, y que entre todas las rutas visiten todas las localizaciones dadas en un conjunto que son los clientes. El objetivo es minimizar la suma de los costos asociados a cada una de las rutas establecidas. Existen múltiples variantes de VRP donde se agregan o modifican el tipo de restricciones a considerar, como por ejemplo, establecer capacidades a los vehículos y pesos requeridos por los clientes, limitar el número máximo de rutas o el costo de las mismas, o establecer ventanas de tiempo para servir a los clientes, entre otras. Las instancias del VRP generalmente se modelan como un grafo (ver Fig. 1.3) donde los nodos son los clientes (existiendo además un nodo especial que es el almacén), y las aristas representan distancias o costos de ir de un cliente a otro, junto con información adicional como ventanas de tiempo o demandas. El objetivo es seleccionar un conjunto de aristas que determinarán las rutas a realizar. Dichas rutas deberán cumplir todas las restricciones establecidas, y se tratará de minimizar el costo asociado.

Las notaciones clásicas para este problema incluyen las siguientes:

- N es el número de clientes sin incluir el almacén.

- $V = \{v_0, v_1, \ldots, v_N\}$ es un conjunto de vértices donde v_0 es el almacén, y el resto son vértices asociados a cada uno de los clientes.
- $E = \{(v_i, v_j) | v_i, v_j \in V\}$ es un conjunto de aristas que une a los clientes entre sí, y a los clientes con el almacén.
- R_i es la ruta del vehículo i.
- C es una matriz de costos positivos o distancias entre los clientes y almacén. Con esta matriz determinamos el peso de cada arista.
- K es el máximo número de vehículos disponibles.

1.2.1 El Problema del Ruteo de Vehículos con Capacidad

El Problema del Ruteo de Vehículos con Capacidad (Capacitated Vehicle Routing Problem - CVRP) es una variante del VRP en la cual se debe de minimizar el sumatorio del costo de las rutas establecidas, pero satisfaciendo además una restricción de capacidad para los vehículos. En esta variante, cada cliente tiene una necesidad que involucra incluir un objeto de un determinado peso en el vehículo que visite al mismo. Debido a la limitación de capacidad máxima de los vehículos ciertas rutas podrían ser infactibles, por lo tanto, se tienen que diseñar un conjunto de rutas factibles que visiten a todos los clientes. Además, de entre todas las posibles soluciones que satisfacen dichas restricciones se intenta encontrar aquella que minimice el costo de los viajes.

1.2.2 El Problema del Ruteo de Vehículos con Capacidad y Ventanas de Tiempo

El Problema del Ruteo de Vehículos con Capacidad y Ventanas de Tiempo (Capacitated Vehicle Routing Problem with Time-Windows - CVRPTW) es otra variante del VRP en la cual, al igual que en el CVRP, se tiene la restricción de la capacidad en los vehículos pero adicionalmente los clientes tienen un lapso de tiempo en el cual están disponibles para recibir al vehículo. El lapso del tiempo del cliente es llamado ventana de tiempo y consta de dos variables que son el límite inferior y el límite superior del tiempo en que el cliente está disponible. Los vehículos deberían de llegar a cada cliente antes de ese lapso de tiempo. Además, si el vehículo llega antes del límite inferior, el vehículo se tendrá que mantener en dicha localización esperando a poder servir al cliente. Por el contrario, si el vehículo llega después del límite superior no hay forma de que el cliente vuelva a estar disponible con lo que se consideraría una solución no factible para el problema.

1.2.3 El Problema del Ruteo de Vehículos con Capacidad y Ventanas de Tiempo, Entregas y Recolectas no Asignadas e Inventario con Restricciones

El Problema del Ruteo de Vehículos con Capacidad y Ventanas de Tiempo, entregas y recolectas no asignadas e inventario con restricciones (Capacitated Vehicle Routing Problem with Time-Windows, Unmatched pickups and deliveries, and Inventory restrictions - CVRPTWUI) es una variante que extiende al CVRP donde, además de las restricciones del CVRP, se agregan restricciones de inventario y se tienen que establecer rutas en diferentes días, creándose para los clientes ventanas de tiempo pero a nivel de días. Con el cambio de la ventana de tiempos a días, se agrega un grado de complejidad al problema puesto que es necesario asignar al cliente al día que se va a atender, y una vez que se disponga del conjunto de clientes asignados a un día en específico se tienen que optimizar las rutas. Además en el problema se cuenta con un número limitado de productos, que en el caso abordado en la tesis son herramientas. Cada cliente debe ser visitado dos veces, un día para entregar la herramienta y después de m días para recoger la herramienta. Una vez recogida la herramienta se puede entregar el mismo día a otro cliente que sea visitado por el vehículo, u otro día posterior por parte de cualquier vehículo. Hay costos asociados por el uso de los vehículos por día, por la adquisición de los vehículos y herramientas, por unidad de viaje (además los vehículos no pueden recorrer más de una distancia límite por día) y el sumatorio de todos esos costos es lo que se debe minimizar. El CVRPTWUI fue planteado como problema de concurso junto a un conjunto de instancias con diferentes características y costos por la empresa ORTEC.

1.3 Hipótesis

En varios problemas de optimización combinatoria, los algoritmos meméticos que incorporan un control explícito de diversidad han permitido obtener los mejores resultados conocidos hasta la fecha, tal es el caso del problema de asignación de frecuencias — Frequency Assignment Problem (FAP) — [52] o del problema de particionado de grafos, entre otros. En estos casos, las ventajas ofrecidas por estos esquemas se debieron principalmente a que los otros algoritmos existentes convergían prematuramente, con lo que no se podía aprovechar los recursos de forma adecuada a largo plazo. También se ha visto que en problemas similares al VRP como es el caso del TSP, el uso de estas técnicas han ofrecido resultados muy competitivos [49]. En el caso del VRP muchos de los mejores resultados han sido obtenidos con algoritmos meméticos. Sin embargo, los algoritmos del estado del arte no utilizan mecanismos de control de diversidad, por lo que en esta tesis se parte de la hipótesis de que la incorporación de técnicas de control de diversidad en algoritmos meméticos para el VRP podría resultar en un mejor comportamiento de los mismos con lo que se podría

avanzar el estado del arte. Para validar esta hipótesis se decidió utilizarla tanto para una variante académica del VRP como para una versión más realista.

1.4 Objetivos

El principal objetivo de esta tesis es diseñar nuevos algoritmos meméticos que incorporen mecanismos de control de diversidad y que sean capaces de alcanzar o mejorar los mejores resultados conocidos hasta la fecha (Best-known Solution - BKS) de variantes académicas del VRP. Además, se pretende extender la estrategia desarrollada para tratar problemas más realistas con el fin de validar si este tipo de estrategias es suficiente para tratar dichos problemas de mayor complejidad. De forma más específica se utilizará el CVRPTW como variante académica, por ser un problema de gran popularidad que ha sido abordado con una gran cantidad de estrategias. Por su parte, como variante más realista se utilizará el CVRPTWUI por su novedad y alto grado de dificultad. Con el fin de validar la hipótesis de partida, se realizarán comparativas entre esquemas que incorporen control explícito de diversidad y esquemas que no los incorporen.

1.5 Contribuciones

Las contribuciones principales de esta tesis son las siguientes:

- Analizar la influencia que tiene la diversidad en el rendimiento de los algoritmos evolutivos en problemas de ruteo de vehículos como lo es el CVRPTW.
- Analizar la interacción entre el operador de cruce basado en secuencia (Sequence-based Crossover - SBX) con diferentes operadores de selección de padre, en específico, con selección aleatoria y selección por torneo binario. Además, con base al análisis anterior, se propone un nuevo operador de cruza con un comportamiento menos destructivo, el operador de cruce de un punto en común de ruptura basado en secuencia (Single Breaking-Point Sequence Based-Crossover - SBSBX).
- Diseñar e implementar un algoritmo evolutivo que administre la diversidad de forma explícita y que supere a los métodos del estado del arte en ejecuciones de largo plazo para el CVRPTW.
- Presentar la definición matemática del CVRPTWUI.
- Demostrar que la adaptación de un algoritmo que obtuvo resultados prometedores en problemas académicos puede obtener resultados cercanos a los mejores conocidos en problemas más realistas como lo es el CVRPTWUI.

1.6 Estructura de la tesis

Esta tesis está organizada de la siguiente manera. En el capítulo 2 se discuten los principales trabajos relacionados que tratan de resolver el CVRPTW y el CVRPTWUI. En este capítulo se puede encontrar una revisión de las codificaciones usadas, de los operadores de cruza, operadores de mutación, medidas de distancia y algoritmos más representativos para el CVRPTW y el CVRPTWUI. Además, se discuten algunos de los mecanismos más populares para controlar la diversidad en algoritmos poblacionales. En el capítulo 3 se presentan las novedades algorítmicas para el CVRPTW, explicando el algoritmo que se ha planteado para resolver el problema. Además se presenta una comparativa con otros algoritmos que han sido diseñados para este problema. En el capítulo 4 se aborda un problema más realista que el CVRPTW, el CVRPTWUI. Se presentan los métodos desarrollados y su validación experimental. Finalmente en el capítulo 5 se presentan las conclusiones y trabajos futuros.

Capítulo 2

Revisión de la Literatura

Este capítulo está dedicado a revisar algunos de los algoritmos que se han diseñado para tratar diferentes variantes del VRP, haciendo especial énfasis en aquellas variantes de problemas y algoritmos que son cercanos a las tratados en esta tesis. Dado que esta tesis implica el desarrollo de un MA, se revisan los diferentes componentes que se han usado en los MAs que han ofrecido mejor rendimiento para el VRP, así como otras estrategias que aunque no estén basadas en MAs son útiles a la hora de desarrollar nuevas propuestas. Particularmente, se discuten los siguientes tópicos: ventajas y desventajas de diferentes representaciones de soluciones candidatas, operadores de cruce, operadores de mutación y medidas de distancia. Como en esta tesis se incorporan mecanismos de control de diversidad de forma explícita en optimizadores para el VRP, se discuten algunas de las formas más populares de administrar la diversidad.

2.1 Optimizadores para el CVRP

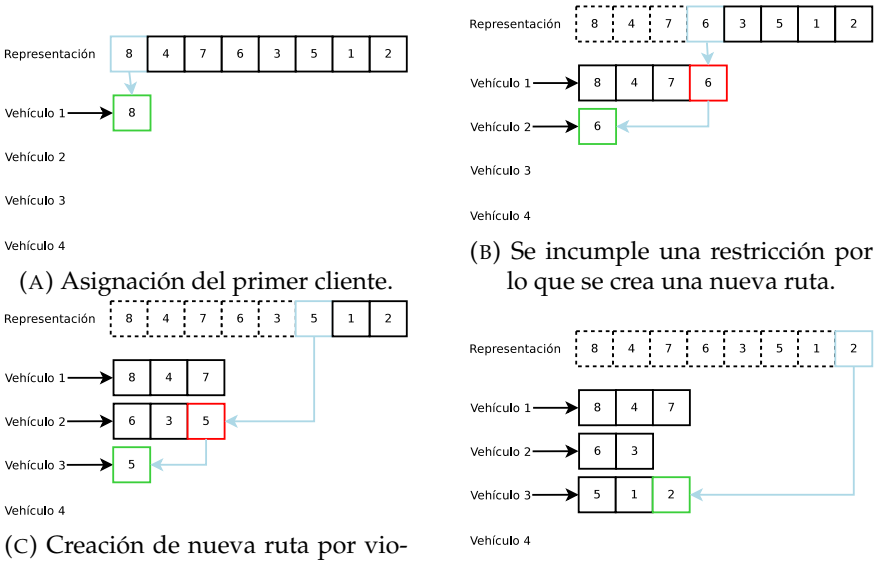
Dada la popularidad y la aplicabilidad del VRP, se han propuesto muchos algoritmos para abordar diferentes variantes del VRP, como podría ser el CVRPTW y el CVRPTWUI. Dado que ambas variantes utilizadas en esta tesis incluyen el uso de capacidades en los vehículos, sólo se han revisado métodos que sean aplicables a variantes del CVRP. Estos métodos van desde métodos exactos que solo pueden ser aplicados a casos relativamente pequeños a métodos basados en búsquedas locales, heurísticas y metaheurísticas. Las metaheurísticas son las más usadas a la hora de abordar instancias grandes, y son las que han reportado los mejores resultados conocidos en la mayor parte de instancias. Actualmente, algunas de las instancias del CVRPTW con hasta 100 clientes han podido ser resueltas con métodos de programación matemática. Esto se debe a que en algunas de ellas se ha podido explotar de mejor forma la localización de los clientes, y acortar el espacio de búsqueda. Sin embargo, la mayor parte de instancias de estos tamaños no han podido ser resueltas, con lo que a partir de esta cantidad de clientes es donde las metaheurísticas empiezan a ofrecer ventajas más claras. Una diferencia que es notoria a la hora de revisar los métodos exactos, es que los mismos a la hora de calcular las distancias entre

clientes usan un único dígito de precisión, mientras que en los esquemas aproximados se usan al menos seis dígitos o incluso la máxima precisión dada por el formato en coma flotante de doble precisión de la IEEE. Esto se debe a que en la mayor parte de esquemas exactos el costo computacional se incrementaría mucho si se usaran más dígitos de precisión, con lo que incluso las instancias pequeñas serían muy problemáticas. A nivel práctico, esto no tiene implicaciones fuertes y las mejores rutas alcanzadas de una u otra forma son similares. Sin embargo, a nivel numérico provoca que las mejores soluciones reportadas (Best-Known Solutions- BKS) difieran al comparar métodos exactos y métodos aproximados, mientras que al comparar las rutas sí que son exactamente las mismas, al menos en varias instancias [57], es decir, sólo cambia el valor numérico debido a la forma en que se calculan las distancias. Por ello, hay que tener especial cuidado a la hora de realizar comparativas, y verificar en los diferentes artículos cuántos dígitos de precisión se usaron, aunque en algunos casos esta información no se reporta.

Uno de los algoritmos más destacados en lo referente a métodos de programación matemática es el presentado en [16], el cual es aplicado al CVRPTW. En primer lugar, calculan el camino mínimo entre parajes de clientes evitando al almacén como punto intermedio utilizando para ello el algoritmo de *Floyd-Warshall*. A continuación usan un algoritmo de programación matemática, específicamente el método de los planos de corte para calcular cotas inferiores a las soluciones. Nótese que este algoritmo podría reportar soluciones no factibles, y por eso los valores se tienen que tomar como cotas inferiores, aunque en algunos casos, especialmente en las instancias más sencillas consiguen alcanzar soluciones factibles. En [44] los autores proponen un algoritmo híbrido, donde se combinan técnicas de programación matemáticas con búsquedas locales para tratar el CVRPTW usando las pruebas de desempeño de Homberger con hasta 1,000 clientes. Los autores no dan detalles en profundidad del algoritmo pero muestran las ventajas de usar vecindarios grandes. La búsqueda local la integran con métodos de programación de restricciones (Constraint Programming - CP) y programación mixta-entera (Mixed Integer Programming - MIP). Ofrecen algunas nuevos BKS, que se consideran a la hora de realizar las comparativas. Los métodos anteriores, por ser bastante diferentes al tipo de optimizador que se diseña en esta tesis, sólo se han usado para comparar las mejores soluciones alcanzadas, pero las componentes que usan no se pudieron aprovechar en los nuevos optimizadores desarrollados. En la siguiente sección, se hace una revisión de las metaheurísticas más exitosas, muchos de cuyos componentes sí fueron usados en esta tesis.

2.1.1 Metaheurísticas para el CVRP

En la mayoría de las instancias disponibles públicamente con 100 o más clientes, las metaheurísticas se consideran los enfoques más avanzados. Como el objetivo principal de esta tesis es avanzar en el diseño y desarrollo de metaheurísticas para los problemas del VRP, a continuación se discuten algunos de los enfoques más eficientes



(A) Asignación del primer cliente.

(B) Se incumple una restricción por lo que se crea una nueva ruta.

(C) Creación de nueva ruta por violación a una restricción.

(D) Asignación del último cliente.

FIGURA 2.1: Transformación de una representación basada en permutaciones a una solución del CVRPTW.

que se han ideado en la literatura. Sin embargo primero se discuten los componentes de las metaheurísticas que se han usado en la literatura antes de dar los detalles de los algoritmos mismos. Se hace especial énfasis en los componentes necesarios para EAs y MAs, pues este tipo de métodos han sido los más exitosos.

Representaciones

La representación o codificación de una solución candidata es una de las decisiones que los diseñadores de metaheurísticas tienen que enfrentar, dando como resultado el diseño de diferentes formas de representar a los individuos cada una con un conjunto de ventajas y desventajas. Dependiendo de la codificación, la aplicación de operadores puede variar, así que la eficiencia del optimizador no solo depende de la representación sino también de los operadores que son aplicados a dicha representación, de cualquier forma, algunas propiedades podrían mantenerse para alguna representación y no para otras, así que seleccionar la representación apropiada es un paso importante. En esta sección describimos dos de las representaciones más populares y damos la razones por las que seleccionamos una de ellas en nuestra propuesta.

En [3], los autores usan una representación completa la cual llamaremos *codificación basada en rutas*. Básicamente, esta representación guarda para cada vehículo una secuencia de clientes que deben ser servidos por dicho vehículo. Dicha secuencia podría ser vacía para algunos vehículos. Usando esta representación, todas las soluciones candidatas pueden ser representadas, lo cual es una característica deseable, sin embargo, dado que esta no es una codificación típica, se deben incorporar operadores diseñados de forma específica para esta representación. En

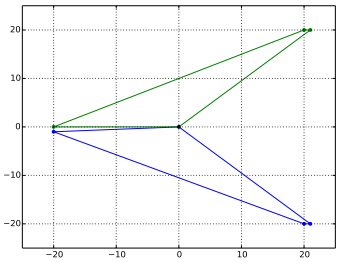
TABLA 2.1: Ejemplo de una instancia del CVRP con demandas, omitiendo su ubicación en el espacio geográfico.

No. Cliente	DEMANDA	Coord X	Coord Y
0	-	0	0
1	20	-20	0
2	20	-20	-1
3	20	20	20
4	20	21	20
5	20	20	-20
6	20	21	-20

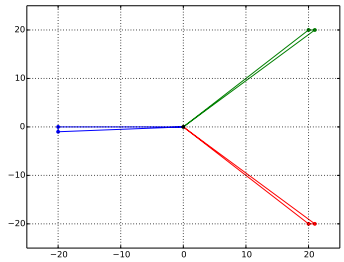
cualquier caso, ya se han diseñado algunos operadores de cruce para este tipo de representación, así que en esta tesis se analizan algunos de los operadores que han sido propuestos para esta representación y se selecciona uno de los que se ha usado más extensamente. Dado que se detectaron algunos inconvenientes de este operador, también se propone un nuevo operador extendiendo al mismo. Nótese que una de las desventajas que se le achaca a esta representación es la memoria adicional y las estructuras de datos que deben de ser usadas para implementar eficientemente los operadores genéticos. Sin embargo, tomando en cuenta la cantidad de memoria que se tiene usualmente disponible en los sistemas actuales, esto no fue un inconveniente importante.

En [9], los autores usaron una representación basada en una permutación de los clientes, por ejemplo, si existen ocho clientes una solución candidata podría venir dada por [8 4 7 6 3 5 1 2]. Para convertir esta representación en una solución específica, es decir, en un conjunto de rutas, se sigue un proceso donde los clientes son visitados en el orden dado por la permutación (Ver Fig. 2.1) y se dispone en cada momento de una ruta activa (la última que se está formando). Cada cliente se intenta agregar como último cliente de la ruta activa. Si al agregarlo se incumple alguna restricción, se crea una nueva ruta que contendrá exclusivamente al cliente que se está incorporando, y esta nueva ruta pasará a ser la activa (Ver Fig. 2.1b y Fig. 2.1c). Este proceso se repite hasta asignar todos los clientes a un vehículo (Ver Fig. 2.1d). Esta representación es bastante popular [2, 38, 34, 25], sin embargo, tiene el gran inconveniente de que no todas las soluciones candidatas factibles del CVRPTW pueden ser codificadas, por lo que dependiendo del caso puede que la solución óptima no tenga una representación con esta codificación.

Para ejemplificar mejor esta situación, en la Tabla 2.1 se define una instancia para el CVRP que se usará como ejemplo. En este caso el cliente 0 es el almacén, los vehículos tienen una capacidad de 60 y se tiene un máximo de 3 vehículos. En específico, para el caso de la representación basada en permutaciones, se probaron todas las permutaciones posibles de los clientes y se reconstruyeron las rutas siguiendo el procedimiento que se realiza cuando se usa este tipo de representación, para el caso de la representación basada en rutas, se realizaron todas las combinaciones posibles usando 1, 2 y 3 vehículos además de asignar diferentes cantidades de clientes en



(A) Mejor solución encontrada usando representación basada en permutaciones (Costo = 189.03).



(B) Mejor solución encontrada usando representación basada en rutas (Costo = 157.594).

FIGURA 2.2: Mejores soluciones encontradas usando diferentes representaciones de individuos.

cada vehículo.

En la fig. 2.2a se muestra la mejor solución alcanzada al usar la representación basada en permutaciones. Esta solución tiene un costo de 189.03 mientras que en la Fig. 2.2b se muestra la mejor solución alcanzada usando la representación basada en rutas obteniendo una solución con costo de 157.594. El mayor inconveniente de la representación basada en permutaciones es que para este caso siempre crea rutas en que se visitan a 3 clientes, aspecto que no adecuado para generar soluciones de alta calidad. Nótese que en este ejemplo una unidad de distancia es usada como una unidad en el costo por lo que el costo obtenido es la distancia recorrida por los vehículos para llegar a los clientes partiendo y regresando del almacén.

Como hemos visto, es claro que hay soluciones que no son representables con el método basado en permutaciones. De hecho, simplemente al contabilizar el número de soluciones representables con esta codificación es obvio que no todas las soluciones se pueden representar. Específicamente, el número de soluciones diferentes que pueden ser codificadas en este caso es dada por la ecuación 2.1.

$$\frac{6!}{2!} \tag{2.1}$$

Nótese que la cantidad total de soluciones para este problemas es mayor, particularmente, esta cantidad está dada por la ecuación 2.2. Por lo tanto, aunque podría ser tentador utilizar una representación basada en permutaciones con el objetivo de utilizar algunos de los operadores genéticos que se han diseñado para codificaciones basadas en permutación, dado que no hay garantía con respecto a la calidad de las soluciones que se pierden en la codificación, decidimos descartar dicha codificación a pesar de su popularidad.

$$\sum_{i=1}^{3} \binom{5}{i-1} \frac{6!}{i!} \tag{2.2}$$

Adicionalmente se hizo un análisis del tamaño del espacio de búsqueda del

CVRP. En primer lugar se hizo el esfuerzo de buscar esta fórmula en la literatura relacionada, sin embargo, no se pudo encontrar. Así que a continuación se explica la forma en que se obtuvo esta fórmula. La fórmula surgió tomando como base el problema de combinatoria de las pelotas y los recipientes, que se resuelve con la técnica conocida como las barras y las estrellas [66]. En el problema de las pelotas y los recipientes, se quiere colocar n pelotas dentro de k recipientes de forma que al menos haya una pelota en cada recipiente y se quiere contabilizar el número de formas diferentes en que se puede hacer. Para realizar este conteo podemos considerar que tenemos $k-1$ barras que van a actuar como separadores para las pelotas (o estrellas). En especifico, para este problema, las n estrellas se colocan en hilera y se colocan $k-1$ barras, cada barra entre dos estrellas con la finalidad de que el primer recipiente contenga desde la primera estrella hasta la primera barra, el segundo recipiente contenga desde la estrella después de la primera barra hasta la segunda barra y así sucesivamente, el último recipiente contiene las estrellas después de la última barra hasta la última estrella. Cada barra se puede colocar entre cualquier par de estrellas contiguas, pero no puede haber dos barras entre el mismo par de estrellas contiguas. Por tanto, para resolver este conteo, basta con contar el número de formas distintas en que podemos seleccionar $k-1$ elementos sobre un total de $n-1$ elementos, es decir, $\binom{n-1}{k-1}$. Nótese que las sumas de las pelotas en los recipientes da n, por tanto, la fórmula anterior también se puede usar para calcular el número de k-tuplas de enteros **positivos** cuya suma es n, que es lo que interesa para nuestro caso.

Para contabilizar el número de soluciones para el CVRP, vamos a considerar que tenemos un máximo de k vehículos. Al considerar que se van a crear i rutas, se tienen que asignar los clientes a las diferentes rutas, asignando al menos un cliente a cada ruta. Teniendo en cuenta todo lo anterior, podemos reemplazar las n estrellas a n clientes y k recipientes como i vehículos. De esta forma, si n es el número de clientes y k es el número máximo de vehículos que podemos usar, deseamos ver de cuántas maneras podemos asignar los clientes a 1 vehículo, 2 vehículos, ..., k vehículos, resultando en la fórmula 2.3

$$\sum_{i=1}^{k} \binom{n-1}{i-1} \tag{2.3}$$

Dado que el orden en que se asignan los clientes sí que importa en este caso, debemos multiplicar por $n!$:

$$\sum_{i=1}^{k} \binom{n-1}{i-1} n! \tag{2.4}$$

La fórmula 2.4 es válida si y sólo si, el orden de los vehículos importa. Sin embargo, en la mayor parte de variantes del CVRP los vehículos son indistiguibles. Por ello, se debe reducir por las permutaciones de los i vehículos, terminando en la fórmula 2.5.

TABLA 2.2: Cota superior del número de soluciones diferentes representables usando representación basada en permutación

No. de Clientes	Soluciones representables
1	1
2	2
3	6
4	24
5	120
6	720
7	5040
8	40320
9	362880
10	1814400

$$\sum_{i=1}^{k} \binom{n-1}{i-1} \frac{n!}{i!} \tag{2.5}$$

Nótese que en el caso de la codificación basada en permutaciones, una cota superior del número de soluciones que pueden ser codificadas es la cantidad de permutaciones (Ver Tabla 2.2) , es decir, $n!$. Esto aplica solo para cuando se llena un solo vehículo, para el caso cuando más de un vehículo se llena, este factor es dividido por $m!$ donde m es el número de vehículos llenos. Esto se debe a que existen permutaciones de soluciones donde lo único que cambia es el orden de los vehículos, es decir, suponiendo que tenemos 6 clientes y cada vehículo se llena con 2 clientes las soluciones $s_1 = 1, 2|3, 4|5, 6$, $s_2 = 1, 2|5, 6|3, 4$, $s_3 = 3, 4|1, 2|5, 6$, $s_4 = 3, 4|5, 6|1, 2$, $s_5 = 5, 6|1, 2|3, 4$ y $s_6 = 5, 6|3, 4|1, 2$ son idénticas ya que siempre la combinación $1, 2$ aparece en un vehículo, la combinación $3, 4$ en otro vehículo y finalmente la combinación $5, 6$ en el tercer vehículo. Sin embargo, en el caso de la codificación basada en rutas, el número de soluciones que pueden ser codificadas depende tanto del número de clientes como del número de vehículos (Ver Tabla 2.3). Esto se debe a que en algunos casos los vehículos no se llena hasta el punto en que la inserción de otro cliente implique la violación de alguna de las restricciones. Como se puede apreciar, incluso para caso pequeñas las diferencias entre el número de soluciones representables con ambas codificaciones es muy significativo.

Operadores de Cruza

Una de las características más importantes de los EAs es la capacidad de combinar la información de diferentes soluciones candidatas para crear nuevas soluciones. En el algoritmo genético original se definió una operación que es llamada recombinación o cruce para tal efecto. En su versión original se trataba de un operador en el que a partir de dos soluciones padre, se creaban dos nuevas soluciones, denominadas

TABLA 2.3: Número de soluciones usando representación basada en Rutas (filas = número de clientes, columnas = número de vehículos disponibles)

#	1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1	1
2	2	3	3	3	3	3	3	3	3	3
3	6	12	13	13	13	13	13	13	13	13
4	24	60	72	73	73	73	73	73	73	73
5	120	360	480	500	501	501	501	501	501	501
6	720	2520	3720	4020	4050	4051	4051	4051	4051	4051
7	5040	20160	32760	36960	37590	37632	37633	37633	37633	37633
8	40320	181440	322560	381360	393120	394296	394352	394353	394353	394353
9	362880	1814400	3507840	4354560	4566240	4594464	4596480	4596552	4596553	4596553
10	3628800	19958400	41731200	54432000	58242240	58877280	58937760	58941000	58941090	58941091

soluciones hijas. De esta forma, los hijos generados deberían de combinar las características de los padres. En los primeros algoritmos esto se hacía de forma independiente del problema, aunque hoy se sabe que es importante que se definan operadores que permitan heredar características importantes de los padres, aunque la definición de características dependen del problema en cuestión. Nótese que si al crear a los hijos se heredan muchas características de los padres, los hijos deberían estar cerca de sus padres en el espacio de búsqueda al considerar definiciones de medidas de distancia que tomen en cuenta dichas características.

El operador de cruce es usualmente considerado un operador dinámico en el sentido que cuando se usan individuos cercanos como entrada, el hijo generado está usualmente cerca de los padres en el espacio de búsqueda, sin embargo, cuando se toman en cuenta individuos lejanos crece la tendencia a crear hijos distantes a los padres. Esto último es una característica deseable de los operadores de cruza porque esto significa que, al menos en la fase inicial, donde los individuos son distantes, el operador de cruza ayuda en la exploración, mientras que en la fase final, donde la mayoría de los individuos tienden a estar localizados en regiones más pequeñas del espacio de búsqueda, se promueve la explotación de esas regiones. A continuación, se analizan algunos de los operadores de cruza que han sido integrados con las representaciones anteriormente descritas, revisando en primer lugar a los operadores de cruza que podrían ser usados con operadores basados en permutaciones y a continuación aquellos que se usan con representaciones basadas en rutas.

Para el caso basado en permutaciones, existe una gran variedad de operadores de cruza aplicables [18]. En [9] usan un operador basado en mapeo (Mapping-Based Crossover). Este operador es bastante sencillo de aplicar. En primer lugar, se generan dos números aleatorios e, $f \in [1, n]$ que representan un segmento del individuo con inicio y fin en las posiciones e y f. A continuación, todos los elementos en dicho rango, se intercambian entre los dos padres. Hay que notar que al hacer esto, podrían aparecer clientes duplicados y algunos clientes podrían no aparecer. Para corregir esto, los autores hacen notar que por cada cliente duplicado en el primer hijo existirá un cliente duplicado en el segundo hijo. Entonces, los clientes duplicados en el primer individuo, que no se encuentran en el segmento intercambiado anteriormente, se intercambian con los clientes duplicados en el segundo individuo. Los

autores no aclaran si los intercambios se hacen por orden, es decir, el primer número duplicado del primer individuo con el primer número duplicado del segundo individuo, o si los intercambios son aleatorios.

En [2] los autores usan un método de cruza que extiende al operador Two-Edge que se propuso originalmente en [23]. En esta cruza se construye una tabla con todas las aristas utilizadas en los padres, y se genera los hijos mediante un procedimiento que intenta utilizar una gran catidad de aristas presentes en dicha tabla.

El Operador de cruza optimizada (Optimized Crossover Operator), propuesto en [38] tiene la característica principal que a la hora de crear al hijo se tiene en cuenta la función de costo. En primer lugar se crea un grafo bipartito, en donde se incluyen nodos de clientes y nodos de posición. Se incluye una arista entre el cliente c y la posición p si el cliente c aparece en la posición p en alguno de los padres. A continuación se consideran todos los emparejamientos perfectos, y de entre todos ellos se selecciona el de menor costo. A continuación se incluye un segundo hijo que excluye las aristas ya incluidas en el primer hijo. Este segundo hijo tendrá peor valor de fitness, pero ayuda a mantener la diversidad en la población.

En el caso de los operadores que fueron diseñados para la representación basada en rutas, existen varios como el operador de cruza de montaje de aristas (Edge Assembly Crossover - EAX), la cruza basada en secuencia (Sequence-Based Crossover - SBX) y la cruza basada en rutas (Route-Based Crossover - RBX).

El operador EAX fue usado en [6] y fue diseñado originalmente para el TSP por [71], y adaptado para el CVRP por [68] y posteriormente para el CVRPTW por [69, 70]. El operador EAX trabaja eliminando y reemplazando aristas entre dos soluciones padres para crear una solución hija sin alterar la orientación de las aristas. En específico, este operador consta de cinco pasos donde en cada paso van construyendo al nuevo individuo tomando en cuenta las aristas de ambos padres. En el primer paso combinan todas las aristas de ambos padres en un solo individuo respetando la dirección de las aristas, excluyendo aquellas aristas que están presentes en ambos padres. Nótese que la arista (a, b) es diferente a la arista (b, a) ya que la dirección de ambas no es igual. Para el segundo paso se crean los AB-ciclos, que son ciclos los cuales se identifican por ser aristas en el primer y segundo padre que están orientadas en sentido opuesto entre ellas. En el tercer paso se crean E-conjuntos por la selección de AB-ciclos acorde a una regla dada, un E-conjunto es un conjunto de aristas que se crean por la combinación de AB-ciclos. Durante el cuarto paso se crea una solución intermedia mediante la aplicación de los E-conjuntos en el primer padre. Finalmente las sub-rutas son unidas dentro de rutas en la solución intermedia (las sub-rutas son rutas que no inician ni terminan en el almacén). Este operador trabaja en un tiempo de $O(m^2)$ donde m es el número de clientes en la instancia por lo que la aplicación en instancias grandes da como resultado un operador lento.

Los operadores SBX y RBX fueron propuestos en [42]. En ambos casos, a partir de dos padres se genera únicamente un hijo. En el operador SBX inicialmente se procede a combinar dos rutas tomando una de cada uno de los padres. En específico,

selecciona una ruta de forma aleatoria de cada padre, y para cada ruta selecciona un cliente de forma aleatoria denominándolos puntos de ruptura. A continuación genera una nueva ruta uniendo la primera parte de la ruta del primer padre con la segunda parte de la ruta del segundo padre. Esta nueva ruta es unida al resto de rutas que existían en el primer padre, salvo la ruta que se eligió de forma aleatoria inicialmente, conformando así al hijo. Debido a su modo de funcionamiento, pueden aparecer clientes duplicados y pueden no aparecer algunos clientes, por lo que el operador SBX necesita una etapa de reparación en la que clientes no enrutados se asignan a una ruta y los duplicados se eliminan de tal forma que se asegure que cada cliente aparezca una única vez. La Fig. 2.3 ilustra un ejemplo de aplicación del SBX. Las rutas con flechas sólidas son las rutas ya sean seleccionadas para la cruza o resultante de la cruza, mientras que la rutas con flechas punteadas significa que son las rutas externas. Adicionalmente, el cuadrado es el almacén, mientras que los círculos son clientes, siendo los círculos negros y blancos en las soluciones padres 1 y 2 los clientes de la primer y segunda mitad respectivamente que conformarán a la ruta resultante de la cruza. Hay que notar que la solución creada antes de la fase de reparación presenta clientes que no se encuentran en rutas, mientras que hay clientes que se encuentran duplicados en diferentes rutas. Esto último se puede observar porque llegan más de dos flechas al círculo o parten más de dos flechas del mismo, indicando así que existen por lo menos dos rutas que llegan o parten del cliente. Los detalles específicos sobre el algoritmo de reparación se encuentran en el capítulo 3.

El operador RBX (ver Fig. 2.4) trabaja de una forma un poco diferente. En este operador se selecciona una ruta del primer padre y, para generar al hijo, se reemplaza una ruta del segundo padre por esta nueva ruta. Ambas rutas son seleccionadas de forma aleatoria y se aplica la misma etapa de reparación que en el SBX, pues también pueden aparecer clientes duplicados o no enrutados. Hay que notar que ambos operadores son bastantes destructivos ya que incluso aunque los padres sean idénticos, el hijo generado podría diferir de los padres. Esta característica no se consideró deseable para el algoritmo que se va a proponer, puesto que sería muy difícil controlar el balanceo entre exploración e intensificación. En particular, incluso al combinar individuos muy parecidos, podrían aparecer nuevas características (aristas o rutas) que no estén presentes en ninguno de los padres. Por ello, se proponen algunas modificaciones para reducir la capacidad disruptiva de este operador. De acuerdo a la literatura especializada, el operador SBX ha obtenido los resultados más prometedores y debido a esto, este operador ha sido el seleccionado para realizar nuestros estudios y extensiones.

Operadores de Mutación

La mutación se refiere a un conjunto de operadores que modifican a un individuo para crear una nueva solución candidata. La utilización de operadores de mutación ha sido habitual desde los primeros EAs, sin embargo su función no siempre ha

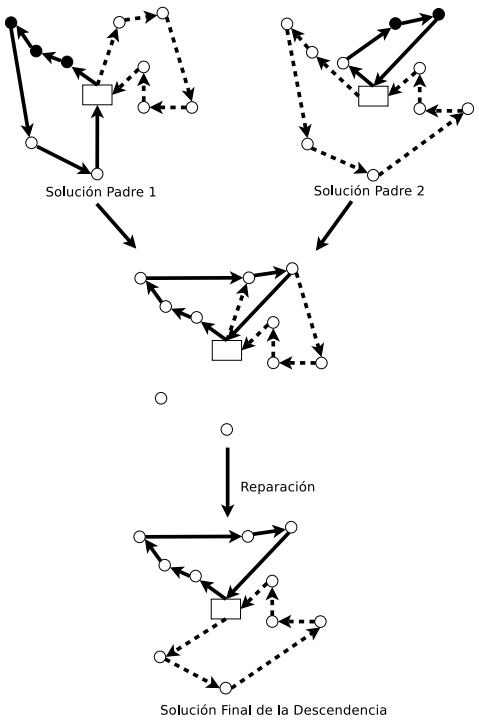


FIGURA 2.3: SBX con todas sus etapas.

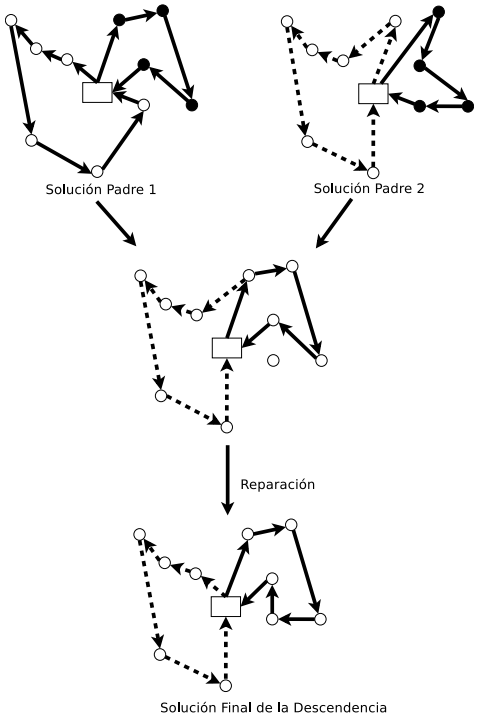
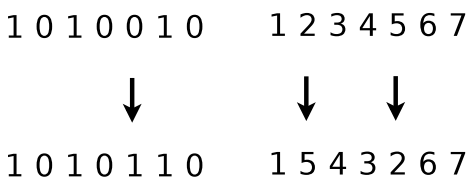


FIGURA 2.4: RBX con todas sus etapas.



(A) Mutación de cambio de bit (B) Mutación de intercambio

FIGURA 2.5: Mutaciones en diferentes representaciones

sido clara [58]. Por ejemplo, mientras que en algunas propuestas como en la programación evolutiva, la mutación fue el operador principal, en otros casos como en los GAs, este no fue tan importante e incluso se diseñaron muchas propuestas que carecían de operador de mutación. Inicialmente, la mutación usualmente fue vista como una forma para evitar convergencia prematura pero hoy en día es usual verlo como un operador que induce explotación porque las modificaciones que hace usualmente son pequeñas, en consecuencia, en la mayoría de los EAs, los operadores de mutación realizan pequeñas perturbaciones para mover ligeramente al individuo en el espacio de búsqueda, permitiendo una mejor intensificación. Nótese que con el objetivo de realizar tales pequeños movimientos, la definición del problema debe ser tomada en cuenta. Por ejemplo, en el vendedor viajante simétrico, invertir una parte de la permutación mantiene la mayoría de los aristas por lo que se considera esto como un pequeño cambio. Sin embargo, en la versión asimétrica, la misma modificación podría ser muy destructiva. De esta forma, el diseño apropiado de una operación de mutación depende tanto de la representación como de la definición del problema.

Existe un gran número de operadores de mutación aplicables a las representaciones más usadas. Por ejemplo, en el caso de la representación binaria, la mutación de cambio de bit (ver Fig. 2.5a) es una de los más comunes. Por su parte, en el caso de la representación basada en permutaciones, la mutación de intercambio (Ver Fig. 2.5b), la cual intercambia con cierta probabilidad valores de dos posiciones diferentes, es bastante popular.

En el caso de los algoritmos meméticos, los métodos de trayectoria están a cargo de realizar la fase de explotación. Para ciertos casos podría tener sentido incorporar una mutación que complemente el efecto de la metaheurística de trayectoria, sin embargo, en muchos MAs donde se aplican métodos de trayectoria más avanzados se deja de hacer uso de mutaciones. De hecho, algunas de las operaciones llevadas a cabo por los métodos de trayectoria son similares a la mutación.

En el caso del CVRPTW, el método [3] genera vecinos tomando en cuenta 10 tipos diferentes de transformaciones donde cada transformación puede ser vista como una mutación. Estas transformaciones son las siguientes:

- **Relocalización Aleatoria del Cliente:** se selecciona un cliente de forma aleatoria de una ruta aleatoria y es colocado en una posición aleatoria de su ruta

original.

- **Mejor Relocalización del Cliente:** se selecciona un cliente de forma aleatoria de una ruta aleatoria y es colocado en la mejor posición de su ruta original.
- **Migración Aleatoria del Cliente:** se selecciona un cliente aleatorio de una ruta aleatoria y este cliente es colocado en una posición aleatoria de una ruta no vacía.
- **Mejor Migración del Cliente:** se selecciona un cliente aleatorio de una ruta aleatoria, y a continuación este cliente es colocado en la mejor posición posible de una ruta no vacía.
- **Intercambio Aleatorio de Clientes:** se seleccionan dos clientes aleatorios c_1 y c_2 de dos rutas aleatorias R_1 y R_2, respectivamente, y son intercambiados.
- **Mejor Intercambio de Clientes:** se seleccionan dos rutas aleatorias R_1 y R_2 y el par de clientes c_1 y c_2 que provee la mayor mejora ($c_1 \in R_1$, $c_2 \in R_2$) son intercambiados. En caso que no haya intercambios de mejora, se hace el intercambio que menos empeore.
- **Intercambio de Clientes con Ventana Coincidente de Tiempo:** se selecciona un cliente de una ruta aleatoria, y a continuación este es cambiado con un cliente con una ventana de tiempo similar de una ruta aleatoria no vacía. Con el objetivo de seleccionar la ventana de tiempo similar, se toma en cuenta el límite superior de la ventana de tiempo, es decir, se minimiza la diferencia entre los dos clientes involucrados.
- **Partición de la Ruta:** primero se selecciona un cliente aleatorio de una ruta aleatoria. A continuación esta ruta se rompe en dos partes, la primera parte va desde el inicio de la ruta a este cliente, mientras que la segunda parte va desde el siguiente cliente al final de la ruta. La primera parte es conservada en la ruta original y los clientes en la segunda parte son asignados de forma aleatoria a rutas vacías, si no existen rutas vacías disponibles este operador no hace nada, en otro caso, cada cliente es asignado uno por uno a cualquiera de este conjunto de rutas vacías de forma aleatoria.
- **Nueva Ruta:** se selecciona un cliente aleatorio de una ruta aleatoria, este cliente es asignado a una ruta vacía.
- **Eliminación de Ruta:** aleatoriamente se selecciona una ruta, entonces, todos los clientes de esta ruta son asignados en la mejor posición disponible de otras rutas.

En varias de las descripciones anteriores, se indica que un cliente se mueve a la mejor posición posible. Esto quiere decir a aquella posición que minimice la función

de fitness. Las mutaciones anteriormente descritas son, probablemente, las más aplicadas en general, sin embargo, existen algunos operadores adicionales que pueden ser visto como generalizaciones de los anteriores. En [20], los autores emplean tres operadores de mutación diferentes: relocalización, intercambio y reposición. La operación de relocalización toma un número de clientes de una ruta dada y los coloca en otra ruta, la operación de intercambio cambia secuencias de clientes entre dos rutas, finalmente, la operación de reposición selecciona un cliente de una ruta específica y lo reinserta en otra posición de la misma ruta. Estos operadores también son usados en [43] y hay que notar que los principios tras estos operadores son básicamente los mismos que los usados en [3], sin embargo, en este último caso se realizan varias modificaciones de las vistas de forma simultánea. La razón es que en [3] las modificaciones son hechas dentro de un enfoque del recocido simulado, así que se podrían encadenar varias modificaciones.

En esta tesis el trabajo se enfoca en desarrollar algoritmos meméticos para variantes del CVRP. En la variante CVRPTW se integró el algoritmo memético con SA así que el paso de la mutación no fue requerido. Por su parte en el caso del CVRPTWUI se usó una búsqueda local simple así que en este segundo caso sí se incluyó un operador de mutación.

Algoritmos más exitosos

En esta última sección, se describen algunos de los algoritmos más exitosos, especificando cuáles han sido los componentes que utilizan cada uno. En primer lugar, cabe destacar que para la gran mayoría de variantes del VRP las metaheurísticas son los algoritmos que predominan en los enfoques del estado del arte, existiendo desde enfoques basados en trayectoria hasta los esquemas poblacionales [28]. En el caso de los métodos de trayectoria, en [12], los autores propusieron una versión paralelizada de la búsqueda tabu (Tabu Search - TS) iterada para abordar diferentes versiones del VRP. En la versión propuesta del TS, los autores primero obtienen una solución inicial ejecutando una búsqueda local, partiendo del resultado obtenido ejecutan el TS. Como perturbación, escogen un cliente y se identifican los π clientes más cercanos, donde π es seleccionado aleatoriamente del rango $[0, \lceil\sqrt{n}\rceil]$, posteriormente se remueven los $\pi + 1$ clientes de sus rutas para ser reinsertados aleatoriamente. Para paralelizar ejecutan varios TS con diferente semilla inicial y cada ciertas iteraciones comparten la mejor solución encontrada entre procesos. En general, solo reportan el promedio de número de rutas y el promedio del costo agrupando conjuntos de instancias, por lo que hacer una comparativa instancia por instancia no es posible. En [17], proponen una variante de GRASP (Greedy Randomized Adaptive Search Procedure) combinada con una búsqueda local evolutiva (Evolutionary Local Search - ELS) para resolver el CVRP de carga bidimensional (Two-dimensional Loading Capacitated Vehicle Routing Problem - 2L-CVRP). Básicamente, usan el ELS para reemplazar la búsqueda local del GRASP y aplicarlo al 2L-CVRP obteniendo resultados prometedores. En [65], aplican un recocido simulado (Simulated Annealing

- SA) al 2L-CVRP logrando mejorar el BKS de 3 instancias y en las demás instancias llegan a resultados iguales o cercanos a los BKS.

De entre los esquemas de trayectoria, el recocido simulado (Simulated Annealing - SA) está entre los que ha obtenido resultados más prometedores. En [3], los autores propusieron una variante del SA basado en población, principalmente, la novedad en este algoritmo es que se evolucionan varias soluciones candidatas de forma simultánea usando múltiples valores de temperatura inicial. No existe interacción entre las soluciones candidatas, por lo que este puede ser visto como un método de trayectoria con reinicio. Adicionalmente, los autores introducen una versión paralela de su algoritmo usando OpenMP y MPI para resolver las pruebas de desempeño de Solomon. Los resultados reportados en [3] son bastante competitivos con las instancias tipo Cxxx, sin embargo, el algoritmo no alcanza resultados de alta calidad cuando trata de resolver las instancias Rxxx y RCxxx. En el capítulo 3 se da mayor información sobre la forma en que se crean los diferentes tipos de instancias. Tenemos la hipótesis que al extender este enfoque para permitir la interacción de las diferentes soluciones candidatas se obtendrían resultados mejores, por lo que este algoritmo es uno de los enfoques que se ha integrado en nuestro MA.

En el caso de metaheurísticas poblaciones más tradicionales — con interacciones entre las soluciones candidatas — se han propuesto diferentes enfoques para abordar el VRP. En [2] se plantea un Algoritmo Genético (Genetic Algorithm - GA) que aborda el VRP con múltiples almacenes (Multi-Depot Vehicle Routing Problem - MDVRP). Esta propuesta usa un representación basada en permutaciones y usa el mecanismo de cruza basada en Edge-2 como operador de cruza para generar hijos. Los autores usan una instancia no estándar del MDVRP que contiene 100 clientes y 4 almacenes, sin embargo, no hacen comparaciones con algún otro algoritmo que esté planteado para este tipo de problema, por lo que no es sencillo contrastar la calidad del método sin realizar reimplementaciones. Evolución Diferencial (ED) también ha sido usada como alternativa para resolver problemas de ruteo, por ejemplo, en [37] se usa para resolver el problema del VRP con entregas y recogidas simultáneas además de ventanas de tiempo. Para adaptar la ED a un problema combinatorio como lo es el VRP usan un vector de número reales que se evoluciona usando las operaciones continuas de ED, y ese vector es usado para generar una permutación. En específico usan la regla de ordenación por el valor más grande (LOV – largest-order-value) para convertir este vector en una permutación. La regla de LOV (Eq. 2.6) es ilustrada con una instancia de ejemplo de $n = 6$, donde el individuo es $X_i = [1.36, 3.83, 2.58, 0.66, 2.63, 0.85]$. El valor más grande es $x_{i,2}$ entonces es seleccionado y se le asigna el rank 1, el segundo valor más grande es $X_{i,5}$ asignándole el rank 2, de la misma forma con cada valor para obtener la secuencia $\phi_i = [4, 1, 3, 6, 2, 5]$. Acorde a la regla de LOV, si $j = 1$, entonces $\phi_{i,1} = 4$ y $\pi_{i,\phi_{i,1}} = \pi_{i,4} = 1$, si $j = 5$, entonces $\phi_{i,5} = 2$ y $\pi_{i,\phi_{i,5}} = \pi_{i,2} = 5$ y así sucesivamente formando la permutación $\pi_i = [2, 5, 3, 1, 6, 4]$. De esta forma se convierte de forma simple el vector con valores continuos a una permutación.

$$\pi_{i,\phi_{i,j}} = j \tag{2.6}$$

Adicionalmente, ED se ha combinado con algoritmos genéticos para tratar otras variantes del VRP [19] y se han usado otros algoritmos que inicialmente fueron diseñados para entornos continuos como la optimización de enjambre de partículas (Particle Swarm Optimization - PSO) [26]. En [74], usan una optimización por colonia de hormigas (Ant Colony Optimization - ACO) híbrida con un algoritmo genético (Genetic Algorithm - GA) para resolver el CVRP con ventana de tiempo, tiempo de viaje y demandas difusas. Hacen una comparativa entre su propuesta, ACO y GAs en una instancia nueva, obteniendo los mejores resultados. Sin embargo, al ser una nueva instancia propuesta no existen otros algoritmos con los que contrastar los resultados. En [73] plantean un ACO híbrido, tomando las ventajas del SA y del ACO para generar su propuesta y aplicarlo a problemas del CVRP. En específico, el SA es usado para generar el estado inicial del ACO tomando la mejor solución encontrada por el SA. En otras palabras, corren primero el SA y después el ACO. El rendimiento del algoritmo frente a otros algoritmos más simples, como el SA, el ACO o TS es bastante bueno pero comparándolo con los BKS se quedan a cierta distancia de los mejores valores reportados.

De entre la gran cantidad de enfoques poblacionales, los algoritmos meméticos son uno de los más populares [72, 55, 67, 25, 39] y más efectivos, así que se discuten con más detalle a continuación. En [43], se propuso un algoritmo memético multi-objetivo para resolver el CVRPTW. Esta propuesta minimiza simultáneamente la distancia de viaje y el número de rutas. Como motor de búsqueda local se usó un enfoque de escalada (hill-climbing) y usan las pruebas de desempeño de Solomon para probar su propuesta. Es considerado uno de los enfoques más prometedores, de hecho, reportaron algunos de los BKS para las instancias tipo Cxxx. Sin embargo, los resultados en instancias de otros tipos no son tan prometedores. Dado que una de las consecuencias de los enfoques multi-objetivo es alterar la forma en la que la diversidad se mantiene [54], la alta calidad de los resultados obtenidos frente a otros enfoques mono-objetivo pueden indicar que esquemas de un solo objetivo podrían estar teniendo problemas en lo referente al control de la diversidad. En [25] se usa un algoritmo memético integrado en un modelo de islas para resolver un CVRP con ventanas de tiempo y tiempos de viaje y de servicios estocásticos. En este multi-MA, cada P generaciones —P es el tamaño de la población— se elige un individuo aleatorio con fitness menor a la mediana de la población y se migra a otra isla, existiendo mecanismos para decidir a qué individuo reemplaza. Nótese que los modelos de islas también se han usado habitualmente para evitar converger de forma excesivamente rápida, por lo que la existencia de este esquema es otro indicativo de los problemas que aparecen relacionados con la convergencia rápida. Como representación usan la representación basada en permutaciones y aplican el operador de cruce basado en orden (Ordered Crossover - OX). Usan las pruebas de desempeño

de Solomon creadas para el CVRPTW. Sin embargo, hacen una serie de modificaciones al conjunto de pruebas de desempeño para adaptarlo al problema que se está resolviendo. Dado que los BKS para este problema y conjunto de pruebas de desempeño no se encuentran disponibles, es difícil poder llegar a conclusiones relativas al rendimiento del mismo.

En [6], los autores propusieron un Algoritmo Memético Paralelo (Parallel Memetic Algorithm - PMA), para abordar el CVRPTW, consistente en dos etapas. En la primera etapa se establece el número mínimo m de rutas a utilizar en las soluciones del CVRPTW. Además se genera un conjunto de soluciones con m rutas. En la segunda etapa se enfoca en minimizar la distancia viajada iniciando con el conjunto de soluciones creadas en la primera etapa. La paralelización consiste en ejecutar el algoritmo para que ambas etapas cooperen entre ellas, es decir, cada hilo se encarga de generar un individuo de la población inicial con m rutas y después minimizar la distancia viajada con el procedimiento de búsqueda local, perturbar, y reparar. Hacen uso del operador EAX como operador de cruce (ver sección 2.1.1). El procedimiento de mejora se encarga de optimizar el hijo generado de la cruza y funciona de la siguiente forma. En primer lugar, se elimina una ruta aleatoria de la solución colocando a los clientes de esta ruta en un conjunto denominado clientes no atendidos. A continuación, se intentan reinsertar los cliente no atendidos en las rutas restantes del individuo contando las veces que este procedimiento falló, definiendo un fallo como un intento de insertar en una posición que provoca que se viole algunas de las restricciones. Esto se hace hasta que se inserta todos los clientes o se alcanza un tiempo máximo. Si existen algunos clientes que no se pudieron reinsertar entonces todos los clientes que se encontraban en el conjunto de clientes no atendidos nuevamente se sacan de las rutas, agregando valor a la función de penalización y el procedimiento se vuelve a realizar hasta que se insertan todos los clientes. Finalmente, se van aceptando a todos los clientes que tienen el menor número en la función de penalización además se va guardando el mejor individuo encontrado por cada hilo. El algoritmo es aplicado al conjunto de pruebas de desempeño de Homberger con 200, 400, 600, 800 y 1000 clientes obteniendo resultados bastantes competitivos en casi todas las instancias.

En el caso del problema CVRPTWUI se usarán las pruebas de desempeño de VeRoLog-ORTEC. Como las pruebas de desempeño de VeRoLog-ORTEC han sido propuestas recientemente, no existen aún artículos que aborden esta variante específica del problema. Sin embargo, existen algunas otras variantes del VRP que son muy similares. En [24], los autores abordan un VRP que comparte varias restricciones con la versión del VeRoLog-ORTEC, sin embargo, el problema solo considera planeación diaria, es decir, no combina dos problemas NP-Hard, así que los operadores ideados en tal enfoque no pueden ser aplicados a la nueva variante del VRP. Los autores proponen una heurística de búsqueda en vecindario grande (Large Neighborhood Search - LNS) para resolver esta variante del VRP y para construir, y posteriormente

reconstruir las soluciones, usan una heurística de inserción del más barato en paralelo, donde para cada cliente intentan todos los puntos posibles de inserción en las rutas posibles y se integra en la localización con el menor costo de inserción en términos de distancia viajada. Específicamente, el algoritmo propuesto consta de 4 etapas, la primera etapa es una fase de destrucción donde implementa tres mecanismos de destrucción donde cada mecanismo se escoge aleatoriamente y se destruye un porcentaje de 5, 10 o 20% de clientes de la solución. En la segunda etapa, se aplica una fase de reconstrucción la cual es la misma que se aplica para construir los clientes por primera vez. En la tercera etapa se aplica una búsqueda local limitada, esta búsqueda local se constituye por operadores heurísticos de la literatura como lo son el 2-opt, el intercambio cruzado y la relocalización ejecutándose en secuencia, es decir, primero el 2-opt, luego intercambio cruzado y por último la relocalización. Las tres etapas anteriores se repiten iterativamente y al final, se ejecuta una cuarta etapa que consiste en una búsqueda local intensiva la cual consta de las mismas etapas que la búsqueda local limitada pero agregando a la secuencia el intercambio, el 2-opt y el intercambio cruzado al final de la secuencia. Es necesario recalcar que para esta extensión del CVRP se usan mayoritariamente esquemas de búsquedas locales más simples en lugar de usar esquemas de trayectoria o meméticos que se usan en otro tipo de problemas con menos restricciones. La razón es que la incorporación de restricciones adicionales dificulta (o imposibilita) realizar evaluaciones incrementales para calcular en tiempo constante el costo de los vecinos. Como veremos en los resultados computacionales, en nuestro algoritmo también aparecieron problemáticas de este tipo al tratar el CVRPTWUI por lo que se tuvo que restringir el algoritmo memético a que usara simplemente búsquedas locales en lugar de técnicas más complejas como SA.

2.2 Diversidad en Algoritmos Poblacionales

Los EAs son, probablemente, las metaheurísticas poblacionales más usadas en la actualidad, habiendo sido aplicadas con éxito en muchos problemas [18] y en diferentes campos de optimización, tales como optimización multi-objetivo [10], optimización combinatoria [13] y optimización continua [59]. A pesar de la gran cantidad de áreas donde se han aplicado, diseñar un EA usualmente involucra varias decisiones de diseño difíciles de tomar[8], por lo que se han realizado muchos esfuerzos para automatizar la creación de componentes y la selección de parámetros de los EAs con el objetivo de facilitar su diseño [31]. De entre todas las dificultades que los diseñadores de EAs pueden encontrar, la aparición de convergencia prematura es, probablemente, uno de los inconvenientes más frecuentes [61]. La convergencia prematura aparece cuando todos los miembros de la población están localizados en una pequeña región del espacio de búsqueda que difiere de la región que contiene a la solución óptima y los operadores genéticos y otros componentes incluidos en los EAs no les permite escapar de dicha región. Esta problemática está estrechamente

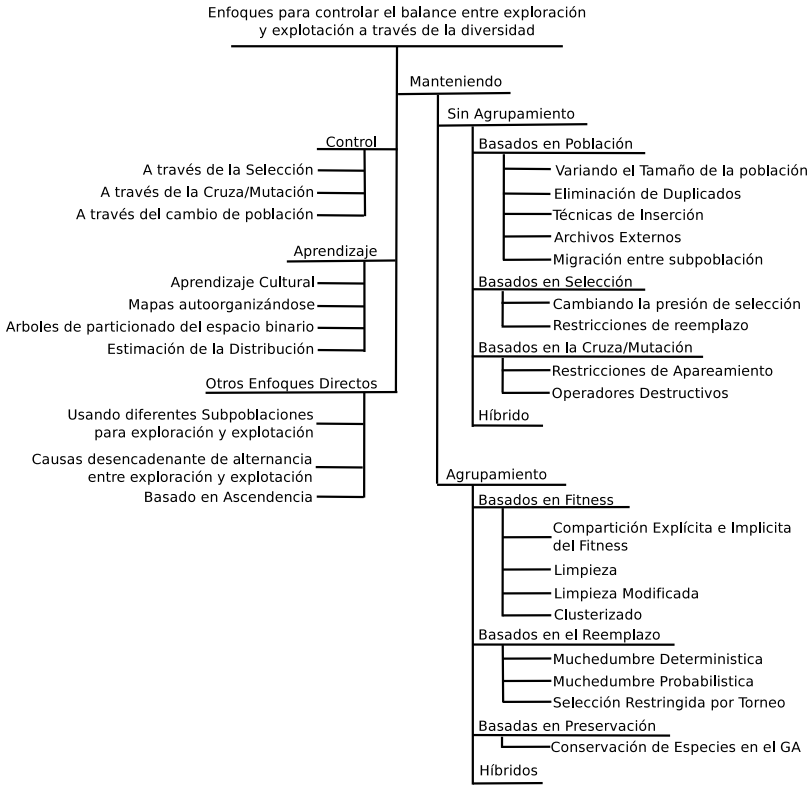


FIGURA 2.6: Enfoques actuales para balancear la exploración y la explotación (Imagen tomada de [62]).

relacionada con la forma en que se administra la diversidad en la población así que incrementar la diversidad de la población es una forma de aliviar los problemas relacionados con la convergencia prematura [61]. Sin embargo, mantener una población muy diversa puede prevenir la intensificación del espacio de búsqueda, resultando por tanto en soluciones de baja calidad. Por esta razón, Mahfoud acuñó el concepto *diversidad útil* [33] para referirse a la cantidad de diversidad que da como resultado soluciones adecuadas.

Los MAs inducen grandes niveles de intensificación debido a la incorporación de métodos de trayectoria, así que un problema frecuente en la aplicación de los MAs es precisamente la aparición de convergencia prematura. En esta tesis, hemos desarrollado un conjunto de MAs con el objetivo de ejecutarlos a largo plazo ya que lo que se pretendía era igualar o mejorar los BKS al menos en problemas académicos. A modo ilustrativo, en algunas ejecuciones el criterio de parada se fijó a un día, por lo que los algoritmos deben tener la capacidad de aprovechar correctamente los recursos durante todo el periodo. Por ello, seleccionar apropiadamente una forma de evitar la convergencia prematura es un paso realmente importante. Se han propuesto muchas formas para lidiar con la convergencia prematura [41]. En particular, muchos esquemas hacen especial énfasis en conseguir un balance apropiado entre exploración y explotación [62]. En la Fig. 2.6 se muestran algunos de los enfoques más populares

estando clasificados en esquemas basados en mantenimiento, control, aprendizaje y otros esquemas. Los esquemas basados en mantenimiento se basan en técnicas que en lugar de medir la diversidad y utilizar esta medida como retroalimentación para mejorar el equilibrio entre la exploración y la explotación, simplemente se asume que las técnicas propuestas mantendrán la diversidad y, por lo tanto, se logrará un mejor equilibrio entre exploración y explotación. Por otro lado, el control de la diversidad mide la diversidad de la población, el fitness del individuo y/o las mejoras de fitness para tomarlas como retroalimentación y dirigir el proceso evolutivo hacia la exploración o explotación. Los esquemas basados en aprendizaje automático, a diferencia del control de diversidad, usan técnicas de aprendizaje automático y la historia a largo plazo de la diversidad para modificar los pasos que da el algoritmo.

Las técnicas basadas en mantenimiento y con agrupamiento han ofrecido muy buenos resultados y son de las más estudiadas, por lo que se decidió trabajar en esta tesis con este tipo de técnicas. Además, de forma más específica, la técnica desarrollada quedaría clasificada como basado en reemplazo, pues es la fase de reemplazamiento en la que se realizan las modificaciones.

Nótese que las implicaciones de la convergencia prematura dependen del criterio de parada asignado, por ejemplo, el número de generaciones o el tiempo total asociado a la ejecución del algoritmo. Dependiendo del problema y del optimizador, la convergencia podría aparecer tras un cierto tiempo de ejecución del algoritmo, así si el algoritmo para antes o cercano a ese punto, la convergencia prematura no es un problema, sin embargo, si el algoritmo corre más allá por un largo periodo, los recursos no serían usados apropiadamente. De esta forma, parece lógico relacionar las decisiones tomadas por el optimizador con el criterio de parada y tiempo transcurrido, sin embargo, muchos de los métodos que han sido ideados no toman en cuenta el criterio de parada para evitar la convergencia prematura. La forma en que la mayor parte de métodos operan es definiendo un conjunto de parámetros que deben ser afinados adecuadamente por un experto con el objetivo de converger en un tiempo adecuado, sin embargo, el afinado de esos parámetros es, usualmente, una tarea complicada.

A continuación se dan algunos detalles adicionales sobre algunos métodos que han sido usados muy frecuentemente para lidiar con la convergencia prematura. Algunos de los métodos más antiguos estaban basados en controlar la presión de selección inducida por la etapa de selección de padres. Sin embargo, se ha mostrado que evitar la convergencia prematura controlando exclusivamente la selección de padres no es posible siempre [5]. En los últimos años, se han ideado varios enfoques que alteran la fase de reemplazo (Ver Fig. 2.6) [32, 36, 64]. El principio básico tras este tipo de enfoques es que al diversificar los sobrevivientes se puede inducir más exploración, pues como se ha discutido anteriormente los operadores de cruza tienden a crear soluciones más diversas cuando las entradas son individuos distantes. Un ejemplo de esta metodología es la Selección por Torneo Restringido [27] (Restricted Tournament Selection - RTS), la cual es uno de los métodos más populares para

retrasar la convergencia prematura en los EAs. En específico, el RTS selecciona de forma aleatoria a dos individuos A y B para cruzarlos y mutarlos generando A' y B'. Posteriormente, se busca al individuo más cercano a A' y al más cercano a B' a partir de de un muestreo de la población con w elementos, identificando así a A'' y B'', respectivamente. A continuación, se pone a competir a A' contra A'' por un lugar en la población y una competición similar es mantenida para B' contra B''. En este método el parámetro w puede ser usado para alterar el balance entre la exploración y la intensificación.

En [64], los autores propusieron un método basado en utilizar una función de fitness que mezcla calidad y contribución a la diversidad. En el método propuesto, todos los individuos se ordenan por calidad y por contribución a la diversidad. Para cada individuo su contribución a la diversidad es calculada como la media de las distancia a los N_{Close} individuos más cercanos. A continuación se calcula una función de fitness que mezcla la información de en qué posición quedó ordenado en lo referente a fitness con la posición en que quedó en lo referente a calidad y el peor individuo es eliminado. Una vez eliminado un individuo, se repite el proceso hasta que sólo queden N individuos, que serán los supervivientes.

Finalmente, en [50] se propone una estrategia de reemplazo diseñada para aliviar el inconveniente de la convergencia prematura que toma en cuenta el criterio de parada y el tiempo o generaciones transcurridas para seleccionar los sobrevivientes de la población. Este método se llama reemplazo con una estrategia de control de diversidad dinámica basada en conceptos multi-objetivo (Replacement with Multi-objective based Dynamic Diversity Control strategy - RMDDC) y en el mismo la diversidad es tomada en cuenta como objetivo adicional y se usan conceptos que surgen en el campo de optimización multi-objetivo para realizar el reemplazo. El método usa el principio de balancear entre exploración y explotación de forma que en las etapas iniciales se inducen grandes niveles de exploración, mientras que al final se inducen mayores niveles de intensificación. El RMDDC ha sido aplicado a diferentes problemas de optimización combinatoria como lo son el Sudoku [53], el problema de empaquetado bidimensional [51] (two-Dimensional Packing Problem - 2DPP),el problema de asignación de frecuencias [52] (Frequency Assignment Problem - FAP) y el problema del vendedor Viajante [49] (Traveling Salesman Problem - TSP). Además, ha sido comparado con un gran conjunto de métodos que controlan la diversidad en diferentes formas y los resultados han sido bastantes competitivos, de hecho, el RMDDC logró nuevos BKS en un conjunto de instancias populares del FAP. El principio básico del RMDDC es penalizar a todos aquellos individuos cuya contribución a la diversidad esté por debajo de un nivel deseado, específicamente, todos esos individuos que no alcanzan una contribución de diversidad deseada son descartados en la etapa de selección de sobrevivientes. Dado que el interés es seleccionar soluciones de alta calidad y soluciones diversas, se calcula un conjunto de soluciones no dominadas (teniendo en cuenta la definición de dominancia de Pareto) entre los

individuos no penalizados que toman en cuenta dichos objetivos: contribución a diversidad y calidad. Dada la calidad general que se ha obtenido con este método, se hace uso del mismo en la propuesta de esta tesis. Los detalles específicos de este método y la forma de adaptarlo al VRP se dan en los siguientes capítulos.

2.2.1 Medidas de Distancia para el CVRP

Una clave para medir adecuadamente la diversidad de una población es disponer de medidas de distancia acorde al problema en cuestión. Las medidas de distancia miden qué tan lejos o cerca está un individuo de otro individuo, ocurriendo que cuando la distancia es pequeña entre dos individuos implica que ellos están localizados en una región similar en el espacio de búsqueda y por tanto comparten muchas características. Con el objetivo de evitar la convergencia prematura, los EAs deberían de buscar en varias regiones simultáneamente, sin embargo, después de detectar regiones prometedoras, es importante ser capaz de intensificar la búsqueda en tales regiones. En consecuencia, con el objetivo de analizar adecuadamente la forma en que está operando un determinado EA, es importante definir una medida de distancia. Estas medidas no sólo se usan para hacer análisis, sino que también algunos métodos como el RMDDC hacen uso de ellas para tomar decisiones durante el proceso de optimización. En general se ha visto que la efectividad de algoritmos como el RMDDC dependen de la definición de medida usada. En algunos casos la distancia podría ser calculada a nivel de fenotipo, es decir, usando los valores que le dan solución al problema de forma directa pero si se usa algún tipo de codificación de los valores para resolver el problema, también es posible usar medidas ligadas al genotipo. En algunos casos en que se dificulta usar una medida ligada al genotipo o fenotipo se ha propuesto usar medidas ligadas al espacio objetivo.

En optimización continua, la medida comúnmente más aplicada es la distancia Euclidiana [46], sin embargo, en problemas combinatorios, necesitamos una medida adecuada que capture la naturaleza del problema. Existen varias medidas de distancia que intentan capturar la naturaleza del problema del CVRP, por ejemplo, en [47] los autores discuten una distancia de emparejamiento exacto para codificaciones basadas en permutaciones. Esta medida se basa en contar el número de valores que están en la misma posición en ambos individuos y restar este valor al máximo número de genes obteniendo así un valor, que cuando es pequeño significa que ambos individuos se encuentran cercanos en el espacio de búsqueda. En la tabla 2.4, la distancia entre el individuo I_1 y el individuo I_2 usando esta medida sería 2. Esta medida tiene algunos inconvenientes importantes. Específicamente, dado que en el CVRPTW se trabaja con rutas, si se trata de aplicar esta medida a una representación basada en rutas podría no captar características importantes. Por ejemplo, en la tabla 2.5 se ejemplifican dos individuos, el I_1 contiene únicamente una ruta mientras que el I_2 contiene dos rutas, ambas delimitadas por el índice del almacén el cual es cero. Si se eliminan los índices de los almacenes y se usa la medida de emparejamiento exacto nos daría un valor de cero, por lo que no alcanza a captar las propiedades de

TABLA 2.4: Ejemplo de dos configuraciones de posibles soluciones con una sola ruta.

I_1	0	1	2	3	4	5	6	7	0
I_2	0	1	2	3	5	4	6	7	0

TABLA 2.5: Ejemplo de dos configuraciones de posibles soluciones, la primera con una ruta y la segunda con dos rutas.

I_1	0	1	2	3	4	5	6	7	0	*	*
I_2	0	1	0	0	2	3	4	5	6	7	0

los individuos usando una representación basada en rutas. Además, lo importante en este problema no es en qué posición queda un cliente en la permutación, sino que hay otros aspectos más importantes, como el cliente/almacén que se visita justo antes y después, o con qué otros clientes quedó agrupado. Todo esto no es tomado en cuenta en la medida anterior.

Un distancia diferente, llamada distancia de tipo R, fue presentada en [35]. En este caso la distancia entre dos individuos se define como el número total de aristas en el primer individuo menos el número de aristas en común, es decir, la distancia de tipo R entre los individuos I_1 e I_2 es el número de aristas en I_1 menos el número de aristas en común, donde un número alto significa que ambos individuos están en diferentes regiones del espacio de búsqueda. En el ejemplo definido en la tabla 2.4, el conjunto de aristas de I_1 son $e_{I_1} = \{(0,1),(1,2),(2,3),(3,4),(4,5),(5,6),(6,7),(7,0)\}$ mientras que el conjunto de aristas en I_2 son $e_{I_2} = \{(0,1),(1,2),(2,3),(3,5),(5,4),(4,6),(6,7),(7,0)\}$ resultando en una distancia de 2. Nótese que en este ejemplo, la aristas $(4,5)$ y $(5,4)$ se consideran como la misma arista ya que el grafo que consideramos no es dirigido, pero en grafos dirigidos esta arista no debería contar como la misma. En cambio, tomando el ejemplo definido en la tabla 2.5 donde el conjunto de aristas de I_1 son $e_{I_1} = \{(0,1),(1,2),(2,3),(3,4),(4,5),(5,6),(6,7),(7,0)\}$ mientras que el conjunto de aristas en I_2 son $e_{I_2} = \{(0,1),(1,0),(0,2),(2,3),(3,4),(4,5),(5,6),(6,7),(7,0)\}$ la distancia obtenida sería de 1 por lo que la distancia de tipo R capta mejor las propiedades compartidas. Hay que notar que la distancia de tipo R no es conmutativa, es decir, puede ocurrir que $d(I_1,I_2) \neq d(I_2,I_1)$. Dado que esta medida está mucho más aceptada en la literatura, habiendo sido usada en diversos esquemas, esta fue la medida que utilizamos en esta tesis.

Capítulo 3

Ruteo de Vehículos con Capacidad y Ventanas de Tiempo

El Problema del Ruteo de Vehículos con Capacidad y Ventanas de Tiempo (Capacitated Vehicle Routing Problem with Time-Windows - CVRPTW) es un problema de optimización combinatoria. En este capítulo se incluye la definición matemática utilizada para el CVRPTW, y se describe el algoritmo que se diseñó para abordar el CVRPTW, el cual es un algoritmo memético con manejo explícito de diversidad que incluye el uso de recocido simulado para intensificar las zonas promisorias detectadas por el algoritmo poblacional. Se presenta además la validación experimental del optimizador desarrollado.

3.1 Descripción del Problema

El CVRPTW es un problema muy popular de optimización combinatoria en el que se busca satisfacer las demandas de producto de un conjunto de clientes, generando para ello un conjunto de rutas que comiencen y terminen en un almacén para un máximo de K vehículos, de forma que se minimice el costo de visitar a cada uno de los clientes. Además, las rutas generadas deben respetar las restricciones de capacidad de los vehículos teniendo en cuenta para ello los pesos asociados a las demandas de los clientes que sirve cada vehículo, así como las ventanas de tiempo en las que los clientes están disponibles.

3.1.1 Definición Matemática

El problema CVRPTW se formaliza de la siguiente forma. Una instancia del problema viene dada por:

- Un grafo $G = (V, E)$ completo no dirigido, donde los vértices $V = \{v_0, \ldots, v_N\}$ corresponden a un almacén y a los clientes, y las aristas $e \in E = \{(v_i, v_j) : v_i, v_j \in V\}$ representan el tiempo para viajar entre ellos. El costo asociado a la arista (v_i, v_j) es C_{ij}.

- Un vector $d = \{d_0, \ldots, d_N\}$, donde d_i es el espacio requerido para cumplir con la demanda del cliente i con $d_0 = 0$.

- Un valor K que representa el número máximo de vehículos a utilizar.
- Una ventana de tiempo para cada cliente que representa los instantes de tiempo en que están disponible para ser servidos, siendo a_j el inicio de la ventana del cliente j y b_j el fin de la ventana de tiempo.

Una solución candidata viene dada por un conjunto de K rutas que inician y terminan en el almacén, pudiendo ocurrir que algunas de esas rutas están vacías lo que representaría a vehículos que no se usarían en la solución. Cada ruta k es un ciclo simple así que se puede representar indicando si un vehículo viaja del nodo i al nodo j en dicha ruta. De esta forma, esas rutas pueden ser definidas matemáticamente con un conjunto de variables de decisión X_{ij}^k que establecen un conjunto de rutas, como sigue:

$$X_{ij}^k = \begin{cases} 1 & \text{si el vehículo } k \text{ viaja del nodo } v_i \text{ al nodo } v_j \\ 0 & \text{En otro caso} \end{cases}$$

Teniendo en cuenta las restricciones y objetivos, el CVRPTW se puede modelar de la siguiente forma:

$$\min TD = \sum_{k=1}^{K} \sum_{i=0}^{N} \sum_{j=0}^{N} X_{ij}^k C_{ij} \tag{3.1}$$

Sujeto a:

$$X_{ii}^k = 0 \ \forall i \in \{0, \ldots, N\}, \ \forall k \in \{1, \ldots, K\} \tag{3.2}$$

$$X_{ij}^k \in \{0, 1\} \ \forall i, j \in \{0, \ldots, N\}, \ \forall k \in \{1, \ldots, K\} \tag{3.3}$$

$$\sum_{k=1}^{K} \sum_{i=0}^{N} X_{ij}^k = 1 \ \forall j \in \{1, \ldots, N\} \tag{3.4}$$

$$\sum_{i=0}^{N} \sum_{j=0}^{N} X_{ij}^k d_j \leq Q^k \ \forall k \in \{1, \ldots, K\} \tag{3.5}$$

$$\sum_{j=1}^{N} X_{0j}^k \leq 1 \ \forall k \in \{1, \ldots, K\} \tag{3.6}$$

$$\sum_{j=1}^{N} X_{0j}^k - \sum_{j=1}^{N} X_{j0}^k = 0 \ \forall k \in \{1, \ldots, K\} \tag{3.7}$$

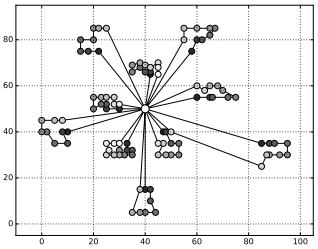
$$a_j \leq s_j \leq b_j \ \forall j \in \{1, \ldots, N\} \tag{3.8}$$

La función objetivo se define en la ecuación 3.1 donde C_{ij} es el costo de viajar del nodo v_i al nodo v_j. En este caso C_{ij} representa la cantidad de tiempo requerida para viajar del nodo v_i al nodo v_j, K es el número máximo de vehículos que pueden ser usados, N es el número de clientes (el almacén es v_0, y los clientes son $v_1, ..., v_N$). La ecuación 3.2 fuerza que un vehículo no pueda viajar de un nodo a sí mismo. La ecuación 3.3 restringe al valor de x_{ij}^k a 0 o 1; cuando este es asignado a uno, implica que, el vehículo k viaja del nodo v_i al nodo v_j, mientras que en caso contrario es asignado a 0. La ecuación 3.4 asegura que cada cliente es visitado exactamente una vez. La ecuación 3.5 garantiza que la suma de las demandas de los clientes visitados por el vehículo k no exceda su capacidad. En las instancias utilizadas se ha considerado la misma capacidad Q para todos los vehículos, y d_j es la demanda del cliente j. La ecuación 3.6 asegura que cada vehículo realiza a lo sumo una ruta, por lo que el número máximo de rutas está limitado a K. La ecuación 3.7 garantiza que todas las rutas inician y terminan en el almacén. Finalmente, la ecuación 3.8 asegura que las ventanas de tiempo son cumplidas, donde a_j es el inicio de la ventana del cliente j para permitir el servicio, b_j es el término de la ventana del cliente j para permitir el servicio, s_j es el instante de tiempo en que se sirve al cliente j definido en la ecuación 3.9 con L como un escalar muy grande ($L = \sum_{i=1}^{N} \sum_{j=1}^{N} C_{ij}$).

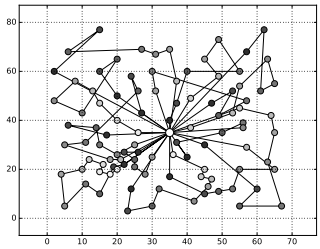
$$s_j = \begin{cases} 0 & \text{si } j = 0 \\ max(s_i + C_{ij} - L(1 - X_{ij}^k), a_j) & \text{en otro caso} \\ \forall i, j \in \{1, \ldots, N\}, \forall k \in \{1, \ldots, K\} & \end{cases} \tag{3.9}$$

3.2 Pruebas de Desempeño

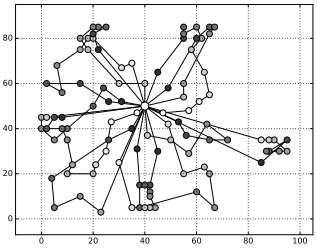
Las pruebas de desempeño son conjuntos de instancias con diferentes características que se utilizan para realizar comparativas entre diferentes algoritmos. En el caso del CVRPTW existen diferentes pruebas de desempeño, eligiéndose en esta tesis las instancias de Solomon [56] ya que son las instancias que más se han usado para el CVRPTW. Estas pruebas de desempeño, originalmente, se diseñaron con instancias de 25, 50 y 100 clientes. Las instancias se separaron en tres grupos distinguiéndose entre sí por la forma en que se distribuyen los clientes. En las instancias tipo Cxxx los clientes se distribuyen en el espacio geográfico formando grupos (véase la Fig. 3.1a). Las instancias de tipo Rxxx usan un generador aleatorio uniforme para distribuir a los clientes (véase la Fig. 3.1b). Finalmente, las instancias de tipo RCxxx son una combinación de los dos grupos de instancias anteriores (véase la Fig. 3.1c), generando algunos clientes agrupados y otros distribuidos aleatoriamente. Posteriormente, y ante el avance de los algoritmos en esta área, Homberger amplió las pruebas de desempeño de Solomon a 200, 400, 600, 800 y 1000 clientes [6], manteniendo los mismos grupos de instancias.



(A) Localización geográfica de los clientes en instancias clusterizadas.



(B) Localización geográfica de los clientes en instancias aleatorias.



(C) Localización geográfica de los clientes en instancias mixtas.

FIGURA 3.1: Ejemplos de localización y configuración de posibles soluciones en las instancias C103, R103 y RC103 con 100 clientes (Soluciones tomadas de [14]).

En la Fig. 3.1 se ejemplifica cada tipo de instancia, donde se toman como ejemplos las instancias C103, R103 y RC103 que pertenecen a tipo clusterizado, aleatorio y mixta. Se puede apreciar que en las de tipo clusterizadas se forman grupos de clientes que suelen facilitar el proceso de optimización, mientras que en las aleatorias los clientes están más dispersos por todo el espacio geográfico. En cualquier caso, es importante hacer notar que aunque dos clientes estén muy cerca, las ventana de tiempo que poseen podrían no concordar para colocarlos en la misma ruta, lo cual añade dificultades adicionales al proceso de optimización.

En nuestros análisis, se eligieron inicialmente las pruebas de desempeño con 100 clientes porque tienen espacios de búsqueda lo suficientemente grandes como para ser bastante desafiantes y han sido utilizadas en muchos artículos. Los análisis se realizaron con instancias de los 3 tipos con el fin de explorar el rendimiento de las propuestas en diferentes condiciones de uso. Posteriormente, dado el buen rendimiento que se obtuvo con el algoritmo memético que integra el manejo explícito de diversidad, se utilizaron adicionalmente instancias de cada grupo con 200 y 400 clientes. En total para la validación experimental de las propuestas se utilizaron 36 instancias, siendo capaces de alcanzar (o superar) los resultados actuales en 27 instancias y de mejorarlos en 14 instancias.

Algoritmo 6: Esquema General del Algoritmo Memético

```
Data: Tamaño de la Población, Criterio de Parada, Temperatura Inicial,
      Operador de Cruza, Operador de Selección de Padres, Mecanismo de
      Selección de Sobrevivientes
Result: El mejor individuo encontrado
Inicialización de la población;
while Criterio de Parada no sea alcanzado do
    while descendencia <tamaño de la descendencia deseado do
        Seleccionar dos padres con el operador de selección de padres;
        Cruza de los padres usando el operador de cruza;
        Insertar nuevo individuo a descendencia;
    end
    aplicar recocido simulado a cada hijo en descendencia;
    Seleccionar la nueva generación con el mecanismo de selección de
     sobrevivientes;
end
```

3.3 Método Propuesto

Esta sección está dedicada a explicar el algoritmo propuesto para resolver el problema del CVRPTW y a presentar la validación experimental del mismo con varias de las instancias de Solomon. En específico, se propone un algoritmo memético que combina a un algoritmo evolutivo con un método de trayectoria.

3.3.1 Algoritmo Evolutivo

Uno de los objetivos principales de la tesis, tal y como se ha descrito, fue diseñar un algoritmo que fuese capaz de alcanzar o mejorar las mejores soluciones conocidas para el CVRPTW. En base a los resultados reportados por diferentes estrategias, se pudo detectar que en general los métodos de trayectoria habían alcanzado los mejores resultados conocidos para instancias pequeñas e incluso para varias instancias grandes de tipo Cxxx. Sin embargo, no ofrecían resultados tan buenos ante instancias más complejas como las de tipo Rxxx y RCxxx. Por su parte, los algoritmos poblaciones — especialmente los meméticos — ofrecían mejores resultados en instancias grandes o complejas, por lo que se propuso diseñar un nuevo algoritmo memético (Memetic Algorithm - MA) que integrase el uso de recocido simulado, ya que entre las variantes basadas en trayectoria, recocido simulado había ofrecido buenos resultados. Las novedades principales en el método propuesto son:

- Se diseña un nuevo operador de cruce que en nuestros experimentos demuestra ser menos destructivo que el cruce basado en secuencia (Sequence Based-Crossover - SBX), un operador clásico de la literatura diseñado específicamente para el CVRPTW. En específico, el operador propuesto se llama cruce de un punto en común de ruptura basado en secuencia (Single Breaking-point Sequence-Based Operator - SBSBX).

- Se incorporan un mecanismo de control de diversidad que no se había usado para problemas de ruteos de vehículos.

La propuesta desarrollada en esta tesis se puede visualizar en el algoritmo 6, cuya estructura general es la del algoritmo memético estándar. Como es habitual, en los pasos de selección de padres y selección de reemplazamiento se hace uso de una función de fitness que evalúa la calidad de los individuos. Con el objetivo de evaluar a los individuos, se toman en cuenta los costos de las rutas y las restricciones. En los casos donde se violan algunas restricciones, el costo del individuo — a minimizar — es igual al costo de la ruta más la constante L definida en la sección 3.1.1. En otro caso, el costo del individuo es integrado sólo por el costo de las rutas. Nótese que esto asegura que cualquier solución factible es preferible a cualquier solución no factible. Hay que notar además que para las instancias donde el número de clientes es igual a 100, acorde a [15], el costo se calcula truncando el costo de cada arista al primer dígito decimal, mientras que en el caso donde los clientes son más de 100 no se trunca y se usa el valor completo almacenándolo en un tipo *double*. Esta decisión se tomó en base a que la mayor parte de trabajos que operan con las instancias de Solomon y sus extensiones a mayor número de clientes operan de esta forma. El primer paso de la propuesta (línea 1) consiste en realizar la inicialización de la población, en la cual se usan las tres estrategias propuestas en [3]. Específicamente, para cada individuo se selecciona aleatoriamente una de las tres estrategias. En todas las estrategias, a la hora de asignar un cliente se elige de forma aleatoria el vehículo que lo va a servir y dicho cliente es visitado después de todos los clientes que ya fueron asignados previamente a dicha ruta. Lo que varía entre las estrategias es el orden en que se seleccionan los clientes. En la primera estrategia el orden en que se eligen los clientes es aleatorio. En la segunda estrategia los clientes son insertados a rutas aleatorias de acuerdo a su identificador, es decir, el cliente 1 se asigna primero, luego el cliente dos y así sucesivamente. Finalmente, para la tercera estrategia, los clientes se asignan en orden de acuerdo a su ventana de tiempo, es decir, en primer lugar se ordenan los clientes en base al fin de su ventana de tiempo y se van asignando considerando dicho orden. Para aquellos casos en los que el final de la ventana de tiempo coincide, se ordena en base a su identificador. Posteriormente, hasta que el criterio de parada sea alcanzado (línea 2), que en nuestro caso es asignado a un tiempo transcurrido, se ejecutan las líneas desde la 3 a la 9. La descendencia ($N - 1$ individuos para el evolutivo generacional con elitismo y N individuos para el evolutivo con el RMDDC) se genera en las líneas 3 a 7. Particularmente, en cada iteración (línea 3) se genera un hijo. Para ello, primero se seleccionan dos padres (línea 4). En este trabajo se consideraron dos tipos de selección: selección aleatoria y selección por torneo binario, la primera de baja presión de selección y la segunda con una alta presión de selección. En el paso 5 se utiliza un operador de cruce para generar una nueva solución candidata. En este trabajo se consideraron dos operadores diferentes, el SBX y el SBSBX, siendo este último una de

las contribuciones de la tesis. Finalmente, se inserta el nuevo individuo en el conjunto de descendientes (paso 6). Tal y como es habitual en los meméticos, se aplica una metaheurística de trayectoria para mejorar a los descendientes (paso 8), usándose en este caso el recocido simulado. Finalmente, en el paso 9 se procede con la selección de reemplazamiento, para seleccionar a los individuos que conformarán la nueva población. En este trabajo se consideraron dos posibilidades. La primera de ellas conforma el evolutivo generacional con elitismo. Específicamente, la siguiente población está formada por los $N - 1$ descendientes junto al mejor individuo de la población anterior. La segunda de ellas hace uso del RMDDC con el fin de incorporar un mejor control de la diversidad poblacional en la fase de reemplazamiento.

Las siguientes secciones están dedicadas a ofrecer detalles adicionales sobre las diferentes componentes que conforman el algoritmo memético desarrollado.

Operador de Cruza

El operador SBX es uno de los operadores de cruce más utilizados en el ámbito del CVRPTW, por lo que fue uno de los operadores que se seleccionó para realizar los análisis. Este operador funciona de la siguiente forma: dados dos padres P_1 y P_2 se genera un nuevo hijo donde, de cada padre, se selecciona una ruta no vacía de forma aleatoria – R_1 y R_2 — y a continuación se selecciona un cliente de forma aleatoria de cada ruta, el cual será el punto de quiebre del padre correspondiente. Como resultado de lo anterior y teniendo en cuenta los puntos de quiebre se crean dos subrutas en cada padre $R_1 = (R_{11}, R_{12})$ y $R_2 = (R_{21}, R_{22})$. Para cada ruta la primera subruta incluye los clientes hasta el punto de ruptura, y la segunda incluye los restantes. Con estas dos subrutas se genera una ruta nueva por la unión de la primer parte de la ruta R_1 con la segunda parte de la ruta R_2, es decir, $R_{new} = (R_{11}, R_{22})$. Finalmente, se genera un nuevo hijo reemplazando R_1 por R_{new} en P_1. Tras realizar algunas pruebas experimentales con el SBX, se pudo detectar que uno de los inconvenientes del mismo es que podía llegar a ser un operador muy destructivo, razón por la que se extendió el mismo generando el SBSBX. En la mayor parte de operadores de cruce clásicos, cuando los individuos de entrada son muy parecidos entre sí, el nuevo hijo creado también es muy parecido a los padres. En concreto, en la mayor parte de casos, si los individuos padre son clones entre sí, el nuevo individuo generado también será un clon de los padres. Esto tiene el efecto de que al mezclar individuos consigo mismos y debido al efecto de la mutación, se produce una intensificación en la región en que se encuentra dicho individuo. Sin embargo, el SBX no garantiza este comportamiento, dándose que incluso aunque los padres sean clones entre sí, el nuevo hijo podría diferir significativamente de los padres. Este efecto se ilustra en la Fig. 3.2, donde se muestra cómo a pesar de que los dos padres son idénticos, el hijo generado difiere. Esto se debe a que la forma de seleccionar el punto de ruptura en ambos padres para generar una nueva ruta por la unión de las subrutas de los padres es independiente en cada padre, es decir, se seleccionan las dos rutas de forma aleatoria y el punto de quiebra en ambas rutas también es aleatorio.

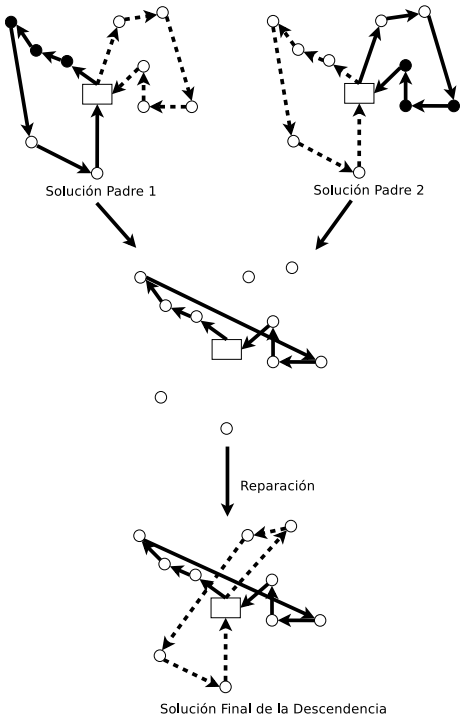


FIGURA 3.2: Ejemplo de lo destructivo que puede ser el SBX.

Se pudo verificar que el SBX clásico es bastante destructivo al aplicar la medida de distancia tipo R, verificando que ante padres con distancias pequeñas entre sí, se generaban hijos distantes, es decir, el SBX modifica fuertemente la región donde se realiza la búsqueda. Esto puede tener un efecto deseado para algunos evolutivos, en el sentido de que incluso si se llega a perder la diversidad en el proceso de optimización, se puede recuperar por acción del operador de cruce, pero es a la vez un comportamiento no clásico, con lo que se extendió el operador SBX para reducir su capacidad destructiva.

El inconveniente anterior se trata de corregir con la propuesta SBSBX (véase Algoritmo 7). En este nuevo operador se fuerza a que el punto de ruptura en ambos padres sea el mismo. Para ello se selecciona un cliente de forma aleatoria y dicho cliente actúa como punto de ruptura en ambos padres (Ver Fig. 3.3) en lugar de realizar selecciones independientes como se hacía con el SBX. De esta forma, el funcionamiento del SBSBX se basa en seleccionar primero un cliente de forma aleatoria y, posteriormente, buscar la ruta donde se encuentra este cliente en ambos padres. Nótese que en esta variante del SBSBX si los padres son clones, el hijo también será un clon, pues se dará que $R_{11} = R_{21}$ y que $R_{12} = R_{22}$

Tanto al aplicar el SBX como el SBSBX se podría generar hijos no válidos en el sentido que se creen soluciones con clientes no enrutados (ver diamante negro en Fig. 3.3) o con clientes duplicados (ver cuadrado azul en Fig. 3.3). Para resolver esto se aplica una etapa de reparación que asegura que el nuevo hijo siempre será válido,

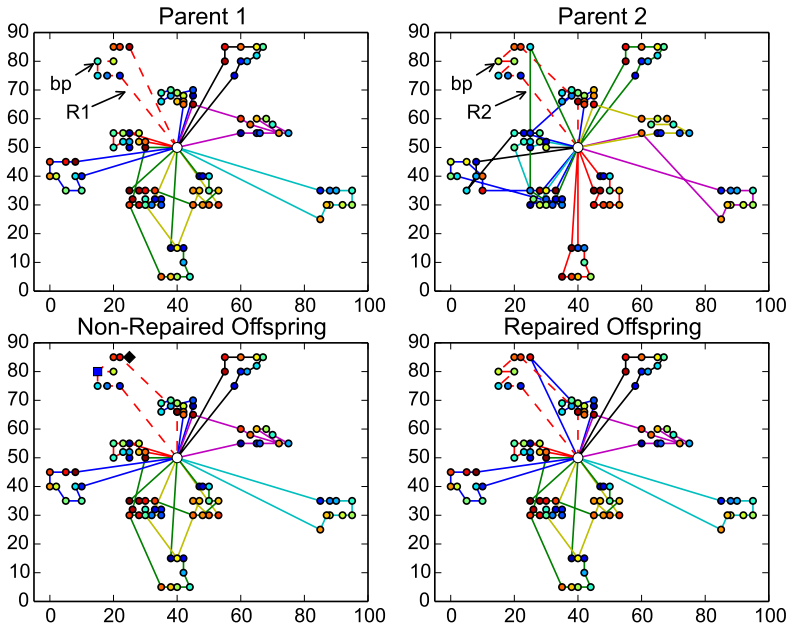


FIGURA 3.3: Ejemplo de aplicación del operador SBSBX

Algoritmo 7: Operador de Cruza de un Punto en Común Basada en Secuencia

Data: $Padre_1$ seleccionado, $Padre_2$ seleccionado
Result: descendencia generada
1 Copiar $padre_1$ a hijo generado;
2 Seleccionar $Ruta_1$ y $cliente_1$ en $padre_1$;
3 Buscar $cliente_1$ en $padre_2$ y seleccionar su ruta como $Ruta_2$;
4 Unir $Ruta_1$ y $Ruta_2$ para generar $Ruta_{new}$ con $cliente_1$ como punto de ruptura ;
5 Reemplazar $Ruta_1$ por $Ruta_{new}$ en el hijo generado ;
6 Aplicar la etapa de reparación al hijo generado ;

es decir, visitará cada cliente una única vez, aunque no necesariamente factible. La etapa de reparación consta de tres procedimientos que son aplicados dependiendo de la razón por lo que hay que reparar. Estos procedimientos son los siguientes:

- **Cliente duplicado en R_{new}:** Uno de los clientes duplicados se selecciona aleatoriamente y se elimina.
- **Clientes duplicados en diferentes rutas:** Como se quiere conservar R_{new}, se elimina el cliente duplicado que no se encuentra en R_{new}.
- **Cliente no enrutado:** Este cliente es insertado en la posición factible que minimice el costo. Si no existe posición factible de inserción, entonces se inserta de forma aleatoria.

Si tras aplicar lo anterior se generan hijos no factibles, por ejemplo que no satisfacen todas las restricciones, en la versión original del SBX [42] se repite todo el proceso hasta generar hijos factibles. En dicho algoritmo no se utilizaba búsqueda local ni recocido simulado. En nuestro algoritmo pudimos verificar que al menos para las instancias con que se validó el esquema, usualmente la aplicación de SA convierte cualquier solución no factible a una factible en muy poco tiempo, por lo que en nuestra implementación no se repite el proceso, sino que SA comenzaría a operar con una solución no factible.

Método de Trayectoria

El algoritmo memético propuesto aplica un SA [30] como método de trayectoria para mejorar la descendencia creada. Se seleccionó SA porque otros algoritmos poblacionales que aplicaban SA han reportado un buen rendimiento. Una de las problemáticas de hacer uso de SA es que se incrementa el número de parámetros a fijar. En concreto, es importante seleccionar la temperatura inicial. Por un lado, utilizar una temperatura alta podría cambiar la región de exploración de los hijos generados de forma drástica, pues muchos vecinos podrían ser aceptados casi independientemente de su calidad realizando prácticamente una caminata aleatoria en las primeras fases. Por otro lado, una temperatura demasiado baja podría convertir el SA en una búsqueda local perdiendo la capacidad de escapar de mínimos locales. Por ello hay que buscar un valor intermedio para la temperatura inicial.

Algoritmo 8: Algoritmo del Recocido Simulado.

Require: Individuo, $T_{cooling}$, T_{max}, T_{min}
1: I_{best} = Individuo
2: $I_{current}$ = Individuo
3: $T = T_{max}$
4: **while** no suceda el criterio de Parada **do**
5: I_{mutado} = mutar($I_{current}$)
6: **if** aceptado_con_metropolis **then**
7: $I_{current} = I_{mutado}$
8: **if** $I_{current} < I_{best}$ **then**
9: $I_{best} = I_{current}$
10: **end if**
11: **end if**
12: $T = T * T_{cooling}$
13: **if** $T < T_{min}$ **then**
14: $T = T_{max}$
15: **end if**
16: **end while**
17: **return** I_{best}

En el Alg. 8 se detalla el procedimiento usado para implementar el SA. En las líneas 1 y 2 se copia el individuo al que se le aplicará el SA en variables temporales que representan al mejor individuo evaluado en el proceso de búsqueda del SA y

TABLA 3.1: Probabilidad de las Operaciones de Transformación

Operación de Transformación	Probabilidad
Realocación Aleatoria del Cliente	15%
Mejor Realocación del Cliente	5%
Migración Aleatoria del Cliente	15%
Mejor Migración del Cliente	5%
Intercambio Aleatorio del Cliente	10%
Mejor Intercambio del Cliente	5%
Intercambio del Cliente con Ventana de Tiempo Coincidente	10%
Partición de Ruta	15%
Nueva Ruta	15%
Eliminación de la Ruta	5%

al individuo actual. En la línea 3 se inicializa la temperatura al valor T_{max}. Posteriormente, mientras no se alcance el criterio de parada, que en nuestro caso es en base a tiempo transcurrido, se ejecutan las líneas 5 a la 15. En la línea 5 se aplica una transformación al individuo actual ($I_{current}$) y se guarda en I_{mutado}. Nótese que habitualmente en SA se tiene una noción de vecindad, y en este paso se genera cualquiera de los vecinos de manera equiprobable. En nuestra implementación, en su lugar se tienen un conjunto de procedimientos de mutación que permiten realizar cambios en el individuo. En la línea 6 se aplica el criterio de metrópolis para decidir si el nuevo individuo es aceptado o no. En caso de ser aceptado, se convierte en la solución candidata actual (línea 7) y en la línea 8 se verifica que si el individuo generado es mejor que el mejor individuo encontrado en el SA hasta el momento para almacenarlo como el mejor individuo evaluado (línea 9). En la línea 12 se aplica el enfriamiento haciendo uso del planificador exponencial, es decir, se multiplica la temperatura actual por un factor $T_{cooling}$. Finalmente, en la línea 13 si la T actual se encuentra por debajo de una temperatura mínima T_{min}, se reinicia la temperatura a su inicial en la línea 14. Finalmente, se devuelve el mejor individuo encontrado durante todo el proceso (línea 17).

Para la generación de las nuevas soluciones candidatas, la variante implementada de SA se basa en 10 transformaciones diferentes. Estas transformaciones son: realocación aleatoria del cliente, mejor realocación del cliente, migración aleatoria del cliente, mejor migración del cliente, intercambio aleatorio del cliente, mejor intercambio del cliente, intercambio del cliente con ventana de tiempo coincidente, partición de ruta, nueva ruta y eliminación de la ruta, donde en cada iteración del SA se aplica cada uno de los procesos con la probabilidad que está dada en la tabla 3.1. Nótese que cada uno de estos procesos fue explicado anteriormente en el capítulo 2.

3.3.2 Control de Diversidad

La diversidad juega un papel muy importante en los algoritmos poblacionales especialmente para aquellos casos en que se quieren realizar ejecuciones a largo plazo. En los algoritmos en que no se implementa un mecanismo de control de diversidad explícito, la cantidad de diversidad que se mantiene en la población depende en gran medida de la presión de selección inducida por la selección de padres, pero intervienen muchas otras propiedades como la capacidad destructiva de los operadores genéticos. Si se escogen operadores que no tienen la suficiente presión de selección el algoritmo podría no converger y, por tanto, no intensificar bien zonas prometedores detectadas. Sin embargo, si los operadores presentan una presión de selección demasiado alta — lo que es mucho más habitual — el algoritmo podría presentar convergencia prematura y no tener una exploración suficiente para identificar zonas prometedores. Esta segunda situación ocurre frecuentemente, por lo que se suelen introducir técnicas explícitas de control de diversidad para retrasar la convergencia.

El MA que se propone implementa un mecanismo de control de diversidad, en el cual se toma en cuenta el criterio de parada y el tiempo transcurrido para seleccionar los sobrevivientes de la población. La estrategia de control de diversidad dinámica basada en conceptos multi-objetivo [50] (Replacement with Multi-objective based Dynamic Diversity Control strategy - RMDDC) fue la selecciona para tal propósito. RMDDC considera la contribución a la diversidad de cada individuo como un objetivo adicional a maximizar y considera algunos conceptos que surgen en el campo de optimización multi-objetivo para realizar el reemplazo.

El algoritmo RMDDC (vea Algoritmo 9) funciona de la siguiente manera. En primer lugar, se unen la población actual y la población de hijos (línea 1) en una población temporal ($IndividuosActuales$) de tamaño $2N$, donde N es el tamaño de la población para cada generación, que representa a los individuos que podrían sobrevivir. A continuación, se calcula la función de fitness de dichos individuos (línea 2 a 4), identificando al individuo con mejor función de fitness (línea 5) para seleccionarlo como superviviente (línea 6), suprimiéndolo de los individuos pendientes de ser seleccionados (línea 7).

Uno de los principios del algoritmo es tratar de evitar que se seleccionen individuos demasiado cercanos entre sí, es decir, individuos que contribuyan poco a la diversidad. En el pseudocódigo, D representa precisamente esa distancia mínima deseada, la cual se va disminuyendo de forma lineal con respecto al tiempo transcurrido (T_e), a partir de una distancia inicial (D_i), de forma que al llegar al tiempo final (T_e) se alcance el valor 0, tal y como se expresa en la línea 8 y en la siguiente ecuación:

$$D = (D_i*) - (D_i*) * \frac{T_e}{T_t} \tag{3.10}$$

Posteriormente, se entra en un proceso iterativo (líneas 9 a 20) en que en cada iteración se selecciona un nuevo sobreviviente hasta que se tengan N individuos

seleccionados. Para esta selección se considera la contribución a la diversidad y la función fitness. La forma en que se mide la contribución a la diversidad para cada individuo pendiente es calculando la distancia al vecino más cercano (Distance Closest Neighbor - DCN) que se encuentra en la población de sobrevivientes. En otras palabras, para cada individuo de la población de individuos actuales se calcula la DCN en la población de sobrevivientes (líneas 10 a la 15). Nótese que, en nuestro caso, la medida usada para calcular la distancia entre dos individuos es la medida de tipo R. Si un individuo tiene una contribución a la diversidad menor a la deseada, se resetea su coste fijándose a infinito, con el objetivo de que el individuo no sea seleccionado (sólo se podría seleccionar si no existe ningún individuo que cumpla el criterio de contribución a diversidad). Para escoger a cada sobreviviente es donde se usa un concepto de optimización multi-objetivo. Dado que se quieren individuos que contribuyan de forma significativa a la diversidad (la maximicen) pero que reduzcan la función de fitness, se calcula el frente de Pareto en base a esos dos criterios (línea 16). Posteriormente, se selecciona un individuo aleatoriamente de entre los que que se encuentran en el frente de Pareto (línea 17), agregando este individuo a la población de sobrevivientes (línea 18) y eliminandolo de los individuos pendientes de ser seleccionados (línea 19). Nótese que cada vez que se agrega un individuo a la población de sobrevivientes se necesita actualizar la DCN para la población de individuos pendientes.

La Fig. 3.4 ilustra el proceso de selección de un individuo. En la misma cada círculo representa un individuo con su valor de contribución de diversidad (DCN) y su función fitness. La D representa la cantidad mínima de contribución a la diversidad deseada. Los individuos que están a la izquierda de D se penalizan, aumentando artificialmente su función de fitness. A continuación, se calcula el frente de Pareto (individuos marcados con $R1$), en este caso sólo los individuos A y B, y se seleccionaría uno de ellos de manera aleatoria. Nótese que en caso que todos los individuos tengan una contribución menor a D, los individuos que formarían el frente no dominado serían los que más contribuyeron a la diversidad, con lo que se realizaría una selección basándose exclusivamente en maximizar la diversidad.

El único aspecto que queda por determinar es el valor inicial de diversidad deseada (D_I). Este valor se calcula como la diversidad poblacional —la diversidad poblacional es calculada como la media de la distancia del individuo más cercano para cada individuo en la población— obtenida en la primera generación (tras la aplicación del recocido simulado) multiplicado por un valor α. La razón para hacer uso de un multiplicador α es que se podrían presentar algunos casos en que la diversidad inicial sea muy alta o muy baja presentando una no adecuada etapa de exploración. Para evitar este problema se introduce el parámetro α el cual se encarga de multiplicar la diversidad inicial. Un valor por encima de 1.0 indicaría que la diversidad deseada estaría por encima de la diversidad inicial y un valor por debajo de 1.0 que la diversidad deseada sea menor a la diversidad inicial.

Algoritmo 9: Algoritmo RMDDC.

Require: Población, Descendencia
IndividuosActuales = Población ∪ Descendencia
for I ∈ IndividuosActuales **do**
  I.cost = costo asociado al individuo I
end for
Best = Individuo con el menor costo en IndividuosActuales
NewPop = Best
IndividuosActuales = IndividuosActuales \ { Best }
Actualizar D tomando en cuenta el tiempo transcurrido (T_e), criterio de parada (T_s) y valor inicial de la diversidad (D_I). Por ejemplo, en el caso de decremento lineal, implementar $D = D_I - D_I * \frac{T_e}{Ts}$
while |NewPop| < N **do**
  for I ∈ IndividuosActuales **do**
    I.DCN = distancia al individuo más cercano de I en NewPop
    if I.DCN < D **then**
      I.cost = Infinito
    end if
  end for
  ND = Individuos no dominados de IndividuosActuales (sin repeticiones) tomando en cuenta I.dcn y I.cost como los dos objetivos
  Seleccionado = Aleatoriamente seleccionar un individuo de ND
  NewPop = NewPop ∪ Seleccionado
  IndividuosActuales = IndividuosActuales \ { Seleccionado }
end while
Población = NewPop

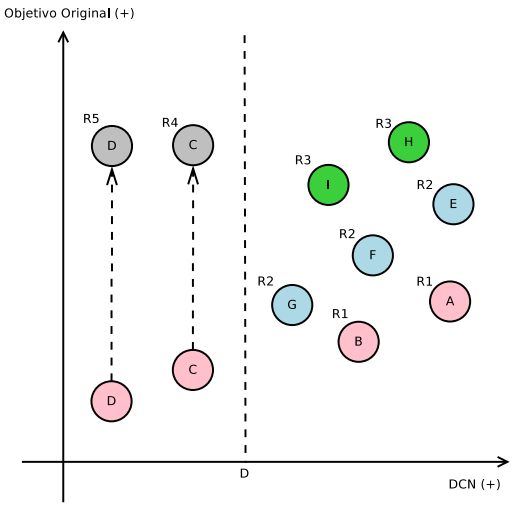


FIGURA 3.4: Sistema de penalización del RMDDC.

3.3.3 Validación Experimental

Esta sección está dedicada a presentar las validaciones experimentales de la propuesta algorítmica anterior. Dado que se usa un enfoque estocástico en nuestra propuesta, cada ejecución se repitió 30 veces con semillas diferentes en el generador aleatorio. Los algoritmos fueron codificados en c++, compilados con g++ 4.8.2 para Ubuntu 14.04.1 y las pruebas se realizaron en una Intel(R) Xeon(R) CPU E5-2620 v2 (2.10 GHz - 32 GB de RAM). Adicionalmente, con el objetivo de comparar apropiadamente el rendimiento de los algoritmos, se realizaron un conjunto de pruebas estadísticas siguiendo los lineamientos presentados en [50]. Primero, se verifican los resultados para ver si siguen una distribución Gaussiana con la prueba de Shapiro-Wilk. Si siguen la distribución Gaussiana, se aplica las pruebas de Levene para verificar la homogeneidad de las varianzas. En caso de verificarse la homogeneidad, se realizan las pruebas de ANOVA; en otro caso, se ejecutan las pruebas de Welch. Cuando las pruebas de Shapiro-wilk muestra que no están siguiendo una distribución Gaussiana, se usan las pruebas no paramétricas de Kruskal-Wallis para verificar si las muestras se extraen de la misma distribución. En todos los casos anteriores se consideró un nivel de significancia de 5%.

Los resultados que se discuten a continuación se han dividido en cuatro bloques. El primer bloque presenta un conjunto de experimentos que se realizaron para determinar una temperatura inicial adecuada para el SA. El segundo bloque presenta análisis de la diversidad inducida por el uso de diferentes operadores de selección de padres y de cruce. En el tercer bloque se analiza el rendimiento obtenido al incluir el uso de control explícito de diversidad. Finalmente, en el cuarto bloque se discute los rendimientos obtenidos por los diferentes enfoques. Con el objetivo de validar nuestra propuesta, los resultados se comparan con las mejores soluciones conocidas (Best-Known Solution - BKS) alcanzados por cualquier optimizador hasta la fecha.

Instanc	Temp	BKS	Media	Mediana	std	Mejor	Peor
C103	10	826.3	828.06	828.06	0.0	828.06	828.06
C103	100	826.3	874.25	875.80	16.45	847.30	903.00
R103	10	1208.7	1215.07	1215.14	0.82	1213.62	1216.72
R103	100	1208.7	1386.11	1387.10	8.75	1366.29	1399.96
RC103	10	1258	1262.56	1262.98	0.72	1261.67	1264.59
RC103	100	1258	1370.73	1371.42	14.23	1332.18	1400.64

TABLA 3.2: Resultados obtenidos usando el SBX y reemplazo generacional para diferentes temperaturas iniciales

Determinación de Temperatura Inicial

La determinación de la temperatura inicial a usar en SA es importante, pues tal y como se explicó anteriormente tiene un efecto importante sobre si el proceso se enfoca más a intensificación o exploración. Dado que es un parámetro que influye mucho en el rendimiento, es necesario determinar una temperatura que trabaje de forma adecuada con el MA. Para fijar un valor promisorio, se tuvo en cuenta algunos resultados encontrados en estudios previos y se usó el MA con SBX y reemplazo generacional con elitismo. En concreto, se usaron dos temperaturas: 10 y 100, de forma que al usar un valor pequeño el esquema se enfocará más en la intensificación. Como tamaño de población, se usaron 50 individuos teniendo una probabilidad del 100% como probabilidad de cruce. El SA se ejecutaba 5 segundos para mejorar cada individuo, reiniciando la temperatura a su estado inicial cuando caía por debajo de una temperatura determinada, este límite inferior se asignó a 0.01. Adicionalmente, se usó como criterio de parada del MA el tiempo, asignando 24 horas como límite de ejecución.

La tabla 3.2 muestra un resumen de los resultados obtenidos con 3 instancias diferentes. Es claro que en todos los casos se obtienen mejores resultados al hacer uso de temperaturas bajas, de hecho en las 3 instancias la media obtenida al usar el valor 10 fue menor (mejor) que la mejor solución encontrada por las configuraciones que consideraron una temperatura inicial igual a 100. De hecho, se ejecutaron los tests estadísticos previamente mencionados, pudiendo confirmar que en todos los casos las diferencias son significativas con media y mediana más baja en el caso de usar menores temperaturas. Esto se debe a que cuando se ocupa intensificar, la temperatura alta llega a ser muy destructiva moviendo incluso al hijo generado de la región, provocando que el algoritmo memético no pueda intensificar. Esto queda claro por ejemplo en la instancia C103 donde a pesar de ser considerada una de las instancias fáciles en la que se pueden lograr buenos resultados simplemente con intensificación, no se logra los resultados que se obtienen con una temperatura baja. Dado que con esta variante ya se obtenían resultados similares o superiores a los de otras variantes de SA, se dio por buena esta parametrización y se enfocaron los esfuerzos en mejorar otras partes del algoritmo.

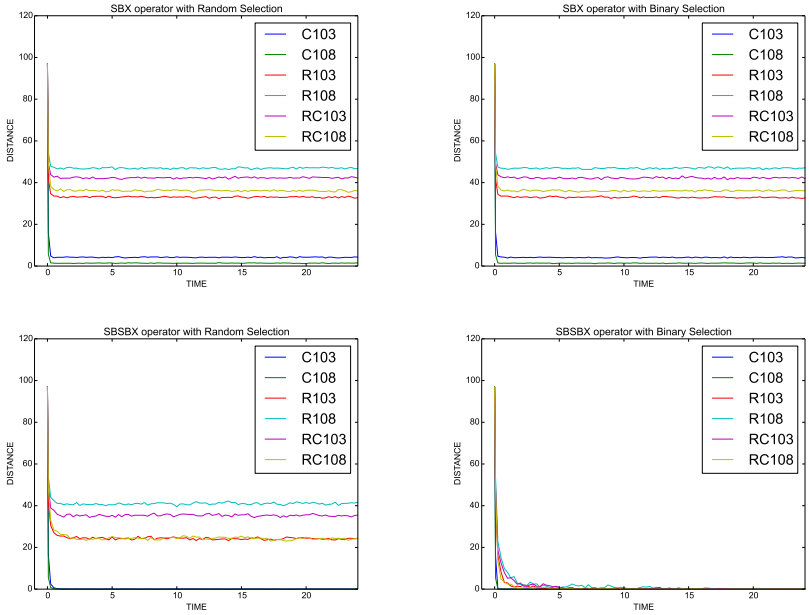


FIGURA 3.5: Diversidad mantenida con los diferentes operadores de cruce y selección de padres.

Análisis de Diversidad

Dado que una de las hipótesis de este trabajo fue que los algoritmos actuales podrían estar teniendo inconvenientes importantes en lo relativo al manejo de diversidad, y que la selección de padres y el operador de cruce tienen efectos importantes sobre la diversidad, se realizaron experimentos haciendo variar el operador de selección de padres y el operador de cruce y se exploró los efectos sobre la diversidad y sobre la calidad de los resultados. En lo referente al operador de selección de padres se probaron el torneo binario y selección aleatoria. La razón es que la selección aleatoria es un operador de muy baja presión de selección, en contraparte, el torneo binario tiene una presión de selección alta. En lo referente al operador de cruce, se probaron el SBX que es más destructivo y por tanto tiende a ofrecer mayor diversidad y el SBSBX, que por ser menos destructivo debería tender a perder la diversidad de forma más rápida ambos con una probabilidad de cruce del 100%. En cada caso, se asignó a 50 el tamaño de la población y se escogió para el SA una temperatura de 10. En estos experimentos se trabajó con 6 instancias diferentes y como medida de diversidad se calculó en primer lugar para cada individuo la distancia a su individuo más cercano utilizando la medida tipo R y se reporta la media de dichos valores.

En la Fig. 3.5 se muestra la evolución de la diversidad a lo largo del tiempo para ejecuciones de 24 horas, mostrándose grandes diferencias al hacer variar el operador de selección de padres y el operador de cruza. En el caso de aplicar el operador SBX, es claro que la diversidad mantenida en la población no se ve afectada por el operador de selección de padres. De hecho, independientemente de la selección de padres, en algunas instancias la diversidad que se mantiene en la población es

TABLA 3.3: Comparativa entre el SBX y el SBSBX con diferentes selecciones de padre (BKS = Mejores Valores Conocidos).

Instancia	BKS	SBX				SBSBX			
		Aleatorio		T. Binario		Aleatorio		T. Binario	
		Mejor	Media	Mejor	Media	Mejor	Media	Mejor	Media
C103	826.3[16]	826.3	**826.3**	826.3	**826.3**	826.3	**826.3**	826.3	**826.3**
C108	827.3[16]	827.3	**827.3**	827.3	**827.3**	827.3	**827.3**	827.3	**827.3**
R103	1208.7[11]	1208.7	1210.36	1208.7	1210.46	1208.7	**1208.88**	1208.7	**1208.91**
R108	932.1[45]	933.7	**936.63**	933.7	**936.74**	933.7	**935.91**	934.3	938.09
RC103	1258[11]	1258	1258.65	1258	1258.72	1258	**1258.18**	1258	**1259.3**
RC108	1114.2[16]	1114.2	1114.79	1114.2	1114.57	1114.2	**1114.22**	1114.2	1123.01

bastante grande durante la ejecución completa. En consecuencia, cuando se usa el SBX, intentar controlar la diversidad por la inducción de diferentes niveles de presión en la etapa de la selección de padres no es posible. La razón parece ser que el SBX es muy destructivo e induce una gran diversidad independientemente de la acción de componentes adicionales del algoritmo memético. Como resultado este operador no se enfoca en las regiones prometedoras encontradas (o las abandona), así que podría decirse que los MAs con el SBX evitan en cierto modo la etapa de explotación.

El operador SBSBX trabaja bastante diferente en términos de la diversidad mantenida. Podemos apreciar que la diversidad decrementa a cero cuando se usa torneo binario de manera muy rápida, significando que inicialmente el MA explora en regiones diferentes y entonces se intensifica en una determinada región seleccionada, lo que es un comportamiento mucho más típico de los meméticos. Dado que el torneo binario induce una presión de selección bastante grande, lo anterior es un comportamiento esperado. Además, el tiempo de convergencia depende fuertemente de la instancia, lo cual no es una característica deseable. Por ejemplo, en el caso de las instancias Cxxx, se obtiene una convergencia bastante rápida, así que este esquema podría presentar una convergencia prematura. Finalmente, en el caso de la selección aleatoria, existen algunas instancias donde la diversidad decrece muy rápido, mientras que en otros casos, se mantiene una diversidad bastante grande. En consecuencia, en algunas instancias el operador tiene el mismo problema que el SBX, es decir, que nunca tiene una fase clara de explotación. Sin embargo, la cantidad de diversidad que se mantiene es menor que en el caso del SBX, lo que implica que al menos tiene un comportamiento relativamente más intensificador.

En la Tabla 3.3 se resumen los resultados obtenidos por las 4 diferentes enfoques. Particularmente, para cada uno, se muestran el mejor y la media de los resultados obtenidos y adicionalmente, se muestra la mejor solución conocida en la columna BKS. En cada instancia, se muestra la media más baja alcanzada por cualquiera de los métodos en **negrita**, además, se resaltan los métodos que obtuvieron resultados cuyas diferencias con respecto al mejor no son estadísticamente significativas. La superioridad del SBSBX con selección aleatoria es bastante clara. De hecho, se resaltan sus resultados en cada instancia.

TABLA 3.4: Comparativa estadística entre el SBX y el SBSBX con las diferentes selecciones de padre(V=Victorias, ND=No Diferente, D=Derrota)

Instancia	V	ND	D	Total
SBSBX_Aleatorio	**8**	**10**	**0**	**8**
SBX_Aleatorio	2	12	4	-2
SBX_TBinario	2	12	4	-2
SBSBX_TBinario	2	10	6	-4

El impacto de la diversidad en la población en los resultados finales depende del tipo de instancia. En las instancias Cxxx, las cuales son consideradas como las más fáciles, cualquier operador de cruza y cualquier selección de padre opera de forma adecuada, alcanzado los mismos resultados, siendo estos los mejores conocidos hasta la fecha. Sin embargo, en instancias más difíciles el impacto sobre la calidad de los resultados es claro. La combinación del torneo binario y el SBSBX no alcanza resultados prometedores pues dado que se realizan ejecuciones a largo plazo, la reducción tan rápida de la diversidad implica un uso inapropiado de los recursos computacionales, pues el algoritmo se centra muy rápidamente en una región reducida del espacio de búsqueda. Sin embargo, no es adecuado tampoco mantener una diversidad excesivamente alta; en consecuencia, los esquemas que obtienen los mejores resultados son los que combinan el SBSBX con la selección aleatoria, es decir, un operador no tan destructivo con una selección de padres que induce una baja presión de selección.

Con el objetivo de confirmar los hallazgos anteriores, se realizaron pruebas estadísticas por pares, es decir, en cada instancia todos los pares de algoritmos fueron comparados estadísticamente. Esto significa que, considerando las seis instancias, se realizaron 18 pruebas estadísticas para cada esquema. La Tabla 3.4 muestra los resultados de este procedimiento: la columna con 'V' indica el número de victorias correspondiente al algoritmo listado en cada fila, el número de casos donde las diferencias no son estadísticamente significativas se muestran en la columna 'ND', finalmente, se muestra la cantidad de derrotas en la columna con la letra 'D'. La columna final muestra la puntuación que es calculada como el número de victorias menos el número de derrotas. Esta tabla confirma que el algoritmo más adecuado entre los anteriores es aquel que combina el SBSBX con la selección aleatoria, así como la importancia que tiene la presión de selección sobre los resultados.

Finalmente, se ejecutaron pruebas adicionales con más instancias con el objetivo de mostrar el comportamiento prometedor del SBSBX con selección aleatoria. En la tabla 3.5 se resumen los resultados obtenidos donde, para cada caso, se obtienen resultados bastantes competitivos. De hecho, de las 12 instancias que se probaron en un caso se pudo obtener un nuevo BKS y, solo en un caso, no se pudo obtener el BKS alcanzado por cualquier otro método desarrollado hasta la fecha. Hay que notar

TABLA 3.5: SBSBX con selección aleatoria (BKS = Mejor Valor Conocido)

Instancia	BKS	Media	Desv. Est.	Mejor	Peor
C203	**588.7**[29]	**588.7**	0.0	**588.7**	**588.7**
C208	**585.8**[29]	**585.8**	0.0	**585.8**	**585.8**
R203	**870.8**[45]	871.04	0.5144	**870.8**	872.5
R208	701.2[45]	702.83	0.9007	**701.0**	704.3
RC203	**923.7**[45]	**923.7**	0.0	**923.7**	**923.7**
RC208	**776.1**[45]	776.45	0.2285	**776.1**	777.3

TABLA 3.6: Pruebas estadísticas usando diferente α para resolver el CVRPTW con 100, 200 y 400 clientes(V = Victoria, ND = No Diferente, D = Derrota)

	Clustered				**Random**				**Rand Clust**				**Total**			
α	**V**	**ND**	**D**	**T**	**V**	**ND**	**D**	**T**	**V**	**ND**	**D**	**T**	**V**	**ND**	**D**	**T**
0.0	0	12	24	**-24**	0	0	36	**-36**	0	0	36	**-36**	0	12	96	**-96**
0.2	4	16	16	**-12**	6	13	17	**-11**	6	6	24	**-18**	16	35	57	**-41**
0.4	11	25	0	**11**	13	20	3	**10**	14	11	11	**3**	38	56	14	**24**
0.6	9	27	0	**9**	15	20	1	**14**	17	19	0	**17**	41	66	1	**40**
0.8	9	27	0	**9**	13	21	2	**11**	18	16	2	**16**	40	64	4	**36**
1.0	8	28	0	**8**	10	23	3	**7**	12	20	4	**8**	30	71	7	**23**
1.2	7	23	6	**1**	9	23	4	**5**	12	19	5	**7**	28	65	15	**13**

que, en esta tabla, para cada instancia los valores mínimos se muestran en **negritas**.

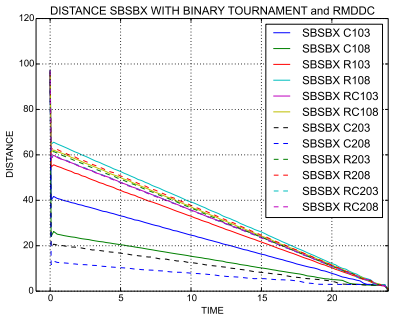
Análisis del Control Explícito de Diversidad

Uno de los objetivos de esta tesis fue diseñar un algoritmo lo suficientemente eficaz para resolver el problema del CVRPTW en comparación a los algoritmos existentes. Si bien el MA con el SBSBX y selección aleatoria presenta resultados bastantes competitivos, en algunas instancias no alcanza un ratio de éxito del 100% e incluso en una instancia no alcanza el BKS conocido. Este problema se podría corregir con un control apropiado de diversidad ya que los recursos computacionales podrían ser utilizados más eficientemente al alternar desde fases más exploratorias a fases más intensificadoras a lo largo del proceso de optimización. Un inconveniente del MA con el SBSBX y selección aleatoria es que al final de la ejecución con algunas instancias no llega a intensificar en las zonas prometedoras que ha encontrando. Por su parte, al SBSBX con torneo binario se converge demasiado rápido, por lo que en ese sentido ninguna de las dos variantes tiene el comportamiento deseado. Para solventar esta problemática, se propone un MA que incorpore el RMDDC como método de control de diversidad, el SBSBX como operador de cruza y el torneo binario como selección de padres. Si bien el SBSBX con torneo binario presenta problemas de convergencia prematura y peores resultados que el SBSBX con selección aleatoria, esta combinación presenta una mejor explotación de zonas prometedoras y el problema de convergencia prematura se corregiría con el uso del RMDDC.

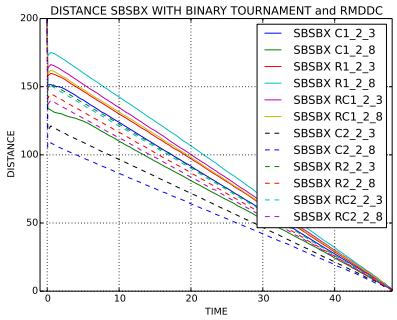
Para la utilización del RMDDC se hizo un análisis para determinar un valor de α adecuado. Para este primer análisis se escogieron 7 valores para α, 2 instancias con distribución clusterizada, 2 con aleatoria y 2 con aleatoria y clusterizada de 100, 200 y 400 clientes realizando pruebas estadísticas por pares, por lo que se realizaron un total de 36 pruebas por cada α. En la Tabla 3.6 se resumen los resultados obtenidos, donde se dividen los resultados por instancias de tipo clusterizada, de tipo aleatoria y mixtas entre clusterizadas/aleatorias. El número de victorias obtenidas por cada α se muestra en la columna 'V', la cantidad de comparativas con diferencias no significativas se enumeran en la columna 'ND', la cantidad de derrotas se incluye en la columna 'D', mientras que el balance entre victorias y derrotas se detalla en la columna 'T'. Nótese que originalmente se realizaron pruebas con diferentes α e instancias con 100 clientes, pero fue imposible determinar el valor de α a utilizar debido a la gran cantidad de empates que hubo (muchos valores llegaban a los BKS), y es por ello que en este experimento se incluyeron casos de 200 y 400 clientes. El valor de 0.6 para α muestra los resultados más prometedores, donde un $\alpha = 0.0$ significa que RMDDC no aplica penalizaciones por lo que pierde en casi todas las instancias y una diversidad alta $\alpha = 1.2$ muestra también un pobre rendimiento por centrarse demasiado en la exploración. Además, es reseñable que el valor $\alpha = 0.6$ fue adecuado para todos los tipos de instancias.

Con el objetivo de confirmar los beneficios ofrecidos por el MA con RMDDC, éste se comparó con el MA sin control de diversidad, es decir, aplicando reemplazo generacional con elitismo. En la Tabla 3.7 se hace un resumen de los resultados obtenidos. Para un total de 36 instancias, se muestra la media y mejor resultado obtenido por el MA con RMDDC y con reemplazo generacional con elitismo. Además, se muestra también el BKS publicado para cada instancia. Los mejores valores encontrados se ponen en **negritas**, mientras que en las columnas de las 'Medias' se pone el signo '=' para aquellos casos donde las diferencias entre ambos tipo de reemplazo no fueron significativas y con '*' se marcan aquellos que los test estadísticos confirmaron la superioridad del mismo. En la mayoría de las instancias el MA con RMDDC alcanza o supera los BKS, mientras el MA sin control de diversidad sólo alcanza los BKS en instancias con 100 clientes por lo que es claro el mejor rendimiento del algoritmo con control de diversidad. De hecho, en 32 casos los test estadísticos confirmaron la superioridad del RMDDC, mientras que en las otras 4 instancias los resultados obtenidos por ambos esquemas fueron similares.

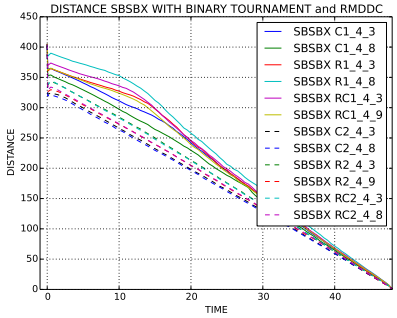
En la Fig. 3.6 se muestra el comportamiento de la diversidad a lo largo de la ejecución, agrupando los resultados en base al número de clientes. Para este caso en cada individuo de la población se encuentra al más cercano y se muestra el promedio de todos los valores. Hay que observar que en instancias con pocos clientes la diversidad decrece linealmente sin ninguna deformación significativa (Ver Subfig. 3.6a). Sin embargo, cuando la cantidad de clientes aumenta, la diversidad muestra un decremento no lineal (Ver Subfig. 3.6c). Hipotetizamos que evitando ese tipo de efectos se podrían alcanzar mejores resultados, pero el desarrollo de alternativas



(A) Diversidad mantenida con el RMDDC en instancias con 100 clientes.



(B) Diversidad mantenida con el RMDDC en instancias con 200 clientes.



(C) Diversidad mantenida con el RMDDC en instancias con 400 clientes.

FIGURA 3.6: Diversidad mantenida por el algoritmo con el RMDDC en instancias con 100, 200 y 400 clientes.

para tratar este aspecto se deja como trabajo futuro.

Las Tablas 3.8 y 3.9 muestran información adicional sobre resultados obtenidos, incluyendo la media, mediana, mejor, peor y desviación estándar.

3.3.4 Discusión

En este capítulo se discutió acerca del procedimiento para diseñar un algoritmo memético que ha ofrecido resultados mejores que los reportados hasta la fecha por cualquier otro algoritmo. En específico, el MA consta de un SA como método de trayectoria para mejorar los individuos obtenidos por un nuevo operador de cruce, el SBSBX. El SBSBX se diseñó con el objetivo de disponer de un un operador de cruce con un comportamiento menos destructivo que el SBX planteado originalmente para el CVRPTW. Además, se incorporó el método de control de diversidad conocido como RMDDC para disponer de un mejor control de la convergencia del esquema. Adicionalmente, se hizo un análisis de un parámetro necesario para el control de diversidad que integra el RMDDC obteniendo, de esta forma, un mejor comportamiento en instancias complejas.

Este MA logra obtener resultados muy prometedores, de hecho, se generaron 14 nuevas mejores soluciones que las conocidas hasta la fecha en las instancias con las

que se realizó la validación experimental. Adicionalmente, el MA con control de diversidad muestra un comportamiento muy estable logrando, por ejemplo, un ratio de éxito del 100% en 10 de las 12 instancias con 100 clientes, mientras que en las dos restantes se logró alcanzar e incluso en una superar el BKS al menos en una ejecución. Por otra parte, los experimentos muestran un comportamiento menos destructivo al usar el SBSBX lo que implica que de esta manera se tiene un comportamiento más similar a los operadores clásicos de cruza al generar hijos cercanos a los padres cuando éstos son muy similares entre sí. Por su parte, en el caso del SBX se muestra que la diversidad que se mantiene a lo largo de la ejecución es invariante a la presión de selección inducida por la selección de padres, lo cual se considera un comportamiento no deseado.

TABLA 3.7: Comparación de resultados entre un MA con control de diversidad (tercera y cuarta columna) y un MA sin control de diversidad (quinta y sexta columna) (BKS = Mejor Valor Conocido)

Inst	BKS	RMDDC		GENELITE	
		Mejor	Media	Mejor	Media
C103	**826.30**	**826.30**	**826.30=**	**826.30**	**826.30=**
C108	**827.30**	**827.30**	**827.30=**	**827.30**	**827.30=**
C203	**588.70**	**588.70**	**588.70=**	**588.70**	**588.70=**
C208	**585.80**	**585.80**	**585.80=**	**585.80**	**585.80=**
R103	**1208.70**	**1208.70**	**1208.70***	**1208.70**	1216.57
R108	**932.10**	**932.10**	936.37*	942.70	949.82
R203	**870.80**	**870.80**	**870.80***	**870.80**	874.64
R208	701.20	**701.00**	701.14*	702.40	707.66
RC103	**1258.00**	**1258.00**	**1258.00***	1260.70	1287.56
RC108	**1114.20**	**1114.20**	**1114.20***	1130.10	1140.99
RC203	**923.70**	**923.70**	**923.70***	**923.70**	938.35
RC208	**776.10**	**776.10**	**776.10***	779.40	787.79
C1_2_3	2707.35	**2681.94**	2681.96*	2681.94	2705.70
C1_2_8	2775.48	**2690.27**	**2690.27***	2690.27	2699.25
C2_2_3	1775.08	**1770.57**	1776.87*	1787.69	1817.40
C2_2_8	**1820.53**	**1820.53**	1823.15*	1823.88	1833.08
R1_2_3	**3381.96**	3392.10	3411.45*	3458.92	3498.74
R1_2_8	**2951.99**	2969.49	2981.69*	3017.49	3083.84
R2_2_3	2880.62	**2558.03**	2565.47*	2582.70	2631.07
R2_2_8	**1849.87**	**1849.87**	1853.04*	1849.87	1918.80
RC1_2_3	**3008.33**	3027.75	3050.60*	3051.21	3123.79
RC1_2_8	**3083.93**	3110.09	3132.65*	3184.91	3239.11
RC2_2_3	2601.87	**2236.99**	2244.97*	2257.51	2309.79
RC2_2_8	2292.53	**2157.54**	2168.42*	2192.46	2230.95
C1_4_3	7060.67	**6968.35**	7081.68*	7543.42	7924.59
C1_4_8	7347.23	**7120.39**	7129.83*	7302.85	7832.33
C2_4_3	4018.02	**3776.29**	3788.46*	3920.26	4087.03
C2_4_8	4301.75	**3797.98**	3809.34*	3900.53	4102.90
R1_4_3	**7819.09**	8133.71	8265.68*	8493.56	8751.55
R1_4_8	**7263.07**	7586.77	7677.16*	7845.80	8016.14
R2_4_3	5911.07	**5501.04**	5571.81*	5761.62	5962.59
R2_4_8	**4015.60**	4055.95	4127.24*	4309.63	4417.03
RC1_4_3	**7534.43**	7878.77	7969.38*	8225.98	8377.99
RC1_4_8	**7760.23**	8197.56	8286.17*	8422.49	8662.95
RC2_4_3	4930.84	**4635.13**	4701.28*	4839.73	5004.99
RC2_4_8	4792.75	**4764.58**	4831.60*	4952.61	5129.38

TABLA 3.8: Resultados usando el SBXBX, RMDDC, torneo binario y con un α = 0.6 para diferentes instancias

Inst	BKS	Media	Mediana	std	Mejor	Peor
C103	826.30	826.30	826.30	0.000000	826.30	826.30
C108	827.30	827.30	827.30	0.000000	827.30	827.30
C203	588.70	588.70	588.70	0.000000	588.70	588.70
C208	585.80	585.80	585.80	0.000000	585.80	585.80
R103	1208.70	1208.70	1208.70	0.000000	1208.70	1208.70
R108	932.10	936.37	938.80	3.647313	932.10	943.10
R203	870.80	870.80	870.80	0.000000	870.80	870.80
R208	701.20	701.14	701.00	0.420000	701.00	702.40
RC103	1258.00	1258.00	1258.00	0.000000	1258.00	1258.00
RC108	1114.20	1114.20	1114.20	0.000000	1114.20	1114.20
RC203	923.70	923.70	923.70	0.000000	923.70	923.70
RC208	776.10	776.10	776.10	0.000000	776.10	776.10
C1_2_3	2707.35	2681.96	2681.94	0.059611	2681.94	2682.18
C1_2_8	2775.48	2690.27	2690.27	0.000000	2690.27	2690.27
C2_2_3	1775.08	1776.87	1770.57	8.269346	1770.57	1788.13
C2_2_8	1820.53	1823.15	1823.88	1.845087	1820.53	1827.24
R1_2_3	3381.96	3411.45	3408.16	11.425023	3392.10	3440.72
R1_2_8	2951.99	2981.69	2978.94	10.095999	2969.49	3010.42
R2_2_3	2880.62	2565.47	2564.07	7.993266	2558.03	2588.49
R2_2_8	1849.87	1853.04	1851.43	6.451816	1849.87	1882.57
RC1_2_3	3008.33	3050.60	3053.89	15.619007	3027.75	3078.92
RC1_2_8	3083.93	3132.65	3132.21	10.712371	3110.09	3156.76
RC2_2_3	2601.87	2244.97	2244.87	7.613328	2236.99	2276.79
RC2_2_8	2292.53	2168.42	2165.25	8.277934	2157.54	2185.37
C1_4_3	7060.67	7081.68	7060.36	66.856998	6968.35	7235.19
C1_4_8	7347.23	7129.83	7125.08	13.720879	7120.39	7180.31
C2_4_3	4018.02	3788.46	3787.59	6.477049	3776.29	3803.87
C2_4_8	4301.75	3809.34	3806.99	11.092843	3797.98	3839.46
R1_4_3	7819.09	8265.68	8261.22	65.729818	8133.71	8389.91
R1_4_8	7263.07	7677.16	7686.39	47.444609	7586.77	7761.67
R2_4_3	5911.07	5571.81	5565.48	48.735219	5501.04	5734.54
R2_4_8	4015.60	4127.24	4124.95	42.427798	4055.95	4223.16
RC1_4_3	7534.43	7969.38	7961.36	50.631227	7878.77	8045.23
RC1_4_8	7760.23	8286.17	8287.96	53.208099	8197.56	8382.32
RC2_4_3	4930.84	4701.28	4706.20	24.304871	4635.13	4736.93
RC2_4_8	4792.75	4831.60	4833.36	28.956622	4764.58	4895.74

TABLA 3.9: Resultados usando el SBSBX, reemplazo generacional con elitismo y torneo binario para diferentes instancias

Inst	BKS	Media	Mediana	std	Mejor	Peor
C103	826.30	826.30	826.30	0.000000	826.30	826.30
C108	827.30	827.30	827.30	0.000000	827.30	827.30
C203	588.70	588.70	588.70	0.000000	588.70	588.70
C208	585.80	585.80	585.80	0.000000	585.80	585.80
R103	1208.70	1216.57	1216.90	4.774875	1208.70	1229.50
R108	932.10	949.82	949.70	4.178571	942.70	957.80
R203	870.80	874.64	874.00	2.742821	870.80	880.70
R208	701.20	707.66	708.40	3.944404	702.40	718.50
RC103	1258.00	1287.56	1284.25	15.618357	1260.70	1324.10
RC108	1114.20	1140.99	1141.85	7.764352	1130.10	1160.50
RC203	923.70	938.35	937.80	6.197674	923.70	951.60
RC208	776.10	787.79	787.65	5.874847	779.40	801.20
C1_2_3	2707.35	2705.70	2691.07	34.385424	2681.94	2797.54
C1_2_8	2775.48	2699.25	2690.27	33.549188	2690.27	2867.84
C2_2_3	1775.08	1817.40	1818.99	21.171315	1787.69	1889.89
C2_2_8	1820.53	1833.08	1830.69	9.190692	1823.88	1859.20
R1_2_3	3381.96	3498.74	3495.56	26.729770	3458.92	3542.75
R1_2_8	2951.99	3083.84	3088.06	33.717766	3017.49	3166.20
R2_2_3	2880.62	2631.07	2620.12	35.531271	2582.70	2725.34
R2_2_8	1849.87	1918.80	1915.72	35.208431	1849.87	1974.40
RC1_2_3	3008.33	3123.79	3114.78	39.865098	3051.21	3246.75
RC1_2_8	3083.93	3239.11	3237.83	30.321932	3184.91	3288.41
RC2_2_3	2601.87	2309.79	2305.58	30.052967	2257.51	2385.39
RC2_2_8	2292.53	2230.95	2223.45	28.318057	2192.46	2318.54
C1_4_3	7060.67	7924.59	7864.94	234.665048	7543.42	8528.88
C1_4_8	7347.23	7832.33	7787.75	223.605127	7302.85	8453.10
C2_4_3	4018.02	4087.03	4067.50	111.204806	3920.26	4286.52
C2_4_8	4301.75	4102.90	4126.62	105.275022	3900.53	4308.45
R1_4_3	7819.09	8751.55	8756.52	121.182214	8493.56	8945.31
R1_4_8	7263.07	8016.14	8017.18	95.043991	7845.80	8184.60
R2_4_3	5911.07	5962.59	5968.03	83.321912	5761.62	6153.73
R2_4_8	4015.60	4417.03	4405.78	70.463270	4309.63	4586.12
RC1_4_3	7534.43	8377.99	8355.93	104.488084	8225.98	8752.31
RC1_4_8	7760.23	8662.95	8645.61	117.823193	8422.49	8910.83
RC2_4_3	4930.84	5004.99	4998.66	87.915831	4839.73	5225.12
RC2_4_8	4792.75	5129.38	5129.01	101.686062	4952.61	5411.23

Capítulo 4

Problemas con Entregas y Recolectas no Asignadas e Inventario con Restricciones

El Problema del Ruteo de Vehículos con Capacidad y Ventanas de Tiempo, entregas y recolectas no asignadas e inventario con restricciones (Capacitated Vehicle Routing Problem with Time-Windows, Unmatched pickups and deliveries, and Inventory restrictions - CVRPTWUI) es una variante del problema del Ruteo de Vehículos con Capacidad (Capacitated Vehicle Routing Problem - CVRP) propuesto por VeRoLog-ORTEC[1]. El problema propuesto por VeRoLog-ORTEC lidia con la logística de una compañía de renta de herramientas que dispone de diversos clientes dispersos en diferentes ubicaciones. En este problema combinatorio, al igual que en el CVRP, se trata de satisfacer las demandas de los clientes mediante la identificación de un conjunto de rutas que serán cubiertas por vehículos que llevan las herramientas que demandan los clientes teniendo una capacidad máxima en cada vehículo. Adicionalmente a la restricción de capacidad, en este problema se agrega una ventana de tiempo que, a diferencia del problema discutido en el capítulo anterior, está dada a nivel de días por lo que se tiene que asignar el cliente al día en el que será visitado para posteriormente crear las rutas que cubran los clientes por día teniendo un nivel mayor de complejidad al combinar dos problemas como son el de asignación y el de optimización de rutas. Además, en el CVRPTWUI se hace uso de herramientas que tienen un costo de adquisición por lo que se intentan organizar las rutas para tratar de minimizar el número de herramientas a usar de forma simultánea. Para resolver el problema se debe determinar el número de herramientas de cada tipo que se va a disponer en el almacén, estando limitado el número de herramientas que se puede adquirir de cada tipo. A la hora de hacer el reparto se debe asegurar que no se usen más herramientas de las adquiridas de forma simultánea. A los clientes se les ofrece las herramientas en préstamo, siendo por tanto necesario programar dos visitas del vehículo: una para entregar en calidad de préstamo la herramienta y otra para recogerla. El número de días que debe pasar entre la entrega y la recogida viene determinado por la petición del cliente. Este capítulo está dedicado a presentar la

[1] VeRoLog-ORTEC es una compañía que trabaja en software de optimización y soluciones analíticas.

formulación matemática de este problema, una adaptación del algoritmo que se presentó anteriormente para tratar el CVRPTW y la validación experimental del esquema anterior.

4.1 Descripción del Problema

En el problema CVRPTWUI se trata de optimizar toda la logística relacionada con la entrega y recogida de herramientas rentadas incluyendo el costo de los viajes, de la adquisición de los vehículos para hacer los viajes, del uso diario de los vehículos y de la adquisición de las herramientas. En esta variante los clientes generan peticiones que especifican las herramientas que necesitan, una ventana de tiempo que especifica los días en la que la herramienta puede ser recibida y el número de días que la herramienta estará ocupada por el cliente. Una de las características del CVRPTWUI es que incorpora varios tipos de restricciones, considerando más restricciones que los CVRP de tipo académico, lo que dificulta bastante el diseño e implementación de los optimizadores. El CVRPTWUI involucra restricciones de los siguientes tipos:

- **Inventario:** hay que decidir cuántas herramientas de cada tipo se adquiere, teniendo un límite máximo para cada tipo, y hay que establecer rutas de forma que no se usen de forma simultánea más herramientas de las adquiridias.
- **Distancia:** se fija la distancia máxima que un vehículo puede recorrer por día.
- **Ventanas de tiempo:** la entrega a cada cliente se debe realizar en un rango de días específico.

Además, en este problema, a diferencia de en las versiones académicas del CVRP, las rutas pueden estar conformadas por varios regresos al almacén antes de finalizar la ruta con el fin de recoger herramientas adicionales. Esto provoca que el problema sea más realista pero hace crecer el espacio de búsqueda dificultando aún más el proceso de optimización.

El objetivo del problema es minimizar los costos asociados con toda la operación de la logística para satisfacer las peticiones de cada cliente tomando en cuenta los costos de los viajes, adquisición de vehículos, uso diario de los vehículos y adquisición de las herramientas. En específico, para cada cliente hay que realizar dos visitas; una para realizar una entrega de herramienta y la otra para recoger la herramienta ya liberada por el cliente con la condición de que la recolección de la herramienta debe de ser exacta el día en que el cliente deja de ocuparla. Adicionalmente, hay que prestar atención al uso de las herramientas ya que podría ser posible que en algún punto de los días todas las herramientas se encuentren en préstamo teniendo que recoger herramientas que se liberen antes de seguir prestando más herramientas. Sin embargo, antes de asignarla a otro cliente, hay que tomar las siguientes condiciones en cuenta:

- Sólo el vehículo que recoge una herramienta en un determinado día puede entregarla a otro cliente.
- El vehículo puede dejar la herramienta en el almacén para liberar espacio ocupado y posteriormente volverla a cargar.
- No existe intercambio de herramientas entre vehículos.
- El vehículo solo puede recoger la herramienta si el cliente ya la liberó y además cuenta con espacio suficiente en el vehículo para la herramienta.
- Una herramienta que no se encuentre en uso al inicio del día puede ser asignada a cualquier vehículo.

VeRoLog-ORTEC proporcionó un evaluador en la página web del concurso [63]. Sin embargo, no ofreció una definición matemática formal del mismo. Con el objetivo de facilitar la comprensión del mismo, en esta tesis se formaliza el mismo.

4.1.1 Definición Matemática

En el CVRPTWUI una instancia del problema viene dada por:

- Un valor N que representa el número de peticiones que contiene el problema.
- Un vector $h = \{h_1, \ldots, h_N\}$, donde h_i es el tipo de herramienta solicitada en la petición i.
- Un grafo $G = (V, E)$ completo no dirigido, donde los vértices $V = \{v_0, \ldots, v_N\}$ corresponden a un almacén y a los clientes asociados a cada petición, y las aristas $e \in E = \{(v_i, v_j) : v_i, v_j \in V\}$ representan el tiempo para viajar entre ellos.
- Una matriz C de valores mayores o iguales a cero que indica el costo de viajar entre los clientes y almacén. Con esta matriz determinamos el peso de cada arista.
- Un valor M que representa el número máximo de días en el que el problema se desarrolla.
- Un valor Q que representa la capacidad máxima de los vehículos (nótese que la capacidad máxima de todos los vehículos es la misma).
- Un valor T que representa la cantidad de tipos de herramientas que existen.
- Un vector $d = \{d_1, \ldots, d_T\}$, donde d_i es el espacio requerido por la herramienta de tipo i.
- Un vector $p = \{p_1, \ldots, p_T\}$, donde p_i es la cantidad máxima de herramientas del tipo i que se pueden adquirir.

- Un vector $a = \{a_1, \ldots, a_T\}$, donde a_i es el costo por adquirir una herramienta del tipo i para tenerla disponible para los préstamos.
- Un valor D que representa la distancia máxima que puede recorrer cada vehículo por día.
- Un valor b_1 que representa el costo por la adquisición de cada vehículo.
- Un valor b_2 que representa el costo por el uso diario de cada vehículo.
- Un valor b_3 que representa el costo por unidad de distancia viajada con cada vehículo.
- Un vector g, donde g_i representa el tiempo que una herramienta estará ocupada por el cliente i.
- Un vector aa, donde aa_i representa el día de inicio de la ventana de tiempo para el cliente i.
- Un vector ee, donde ee_i representa el día de término de la ventana de tiempo para el cliente i.

Los objetivos y restricciones del CVRPTWUI pueden ser modelados de la siguiente forma. Dadas las definiciones anteriores, una solución candidata es representada por un conjunto de $K \times M$ secuencias, correspondientes a rutas para M días, usándose un máximo de K vehículos en cada día. Cada ruta inicia y termina en el almacén y puede visitar el almacén varias veces. Nótese que alguna de esas rutas podrían estar vacías por lo que los vehículos podrían no visitar a ningún cliente, así que K representa el número máximo de vehículos que se pueden usar en cada día. Entre todas las rutas para cada petición se realiza una única vez la entrega de herramienta y una única vez la recogida de la misma. Para especificar si se está entregando o recogiendo, las secuencias que especifican la ruta a seguir, además del 0 que representa al almacén, contendrán valores en el conjunto $\{1, \ldots, N\}$ para especificar entrega de herramientas, y en el conjunto $\{-1, \ldots, -N\}$ para especificar recogida de herramientas. De esta forma, una solución candidata viene dada por $K \times M$ secuencias de la siguiente forma:

$$\begin{aligned}\sigma^{k,m} =&(\sigma^{k,m}(1), \sigma^{k,m}(2), \ldots, \sigma^{k,m}(|\sigma^{k,m}|))(\forall k \in \{1, \ldots, K\},\ \forall m \in \{1, \ldots, M\})\\ &\text{s.t. } \sigma^{k,m}(i) \in \{1, \ldots, N\} \cup \{-1, \ldots, -N\} \cup \{0\}, (\sigma^{k,m}(1) = 0 \wedge\\ &\sigma^{km}(|\sigma^{k,m}|) = 0)\end{aligned}$$

$$\begin{aligned}&((\sigma^{k1,m1}(i) = 0) \vee (\sigma^{k1,m1}(i) \neq \sigma^{k2,m2}(j)))(\forall (k1, m1, i) \neq (k2, m2, j),\\ &\forall k1, k2 \in \{1, \ldots, K\}, \forall m1, m2 \in \{1, \ldots, M\}, \forall i \in \{1, \ldots, |\sigma^{k1,m1}|\},\\ &\forall j \in \{1, \ldots, |\sigma^{k2,m2}|\})\end{aligned} \tag{4.1}$$

Nótese que para este problema el almacén puede ser visitado intraruta para hacer carga o descarga de nuevas herramientas, por lo que en una ruta el almacén no solamente podría ser visitado al inicio y fin de la ruta, sino ser visitado en más de una ocasión, esta condición se ejemplifica en la ecuación 4.1. Este hecho es lo que se expresa en la tercera restricción.

Con el propósito de definir de forma más sencilla la función de costo y las restricciones, se definen las variables de decisión $X_{i,j}^{k,m}$ que expresan si en la ruta k del día m, los número i y j aparecen de forma consecutiva en la secuencia, es decir, si en la ruta se va de i a j, cumpliéndose que $i, j \in \{1, \ldots, N\} \cup \{-1, \ldots, -N\} \cup \{0\}$ con el significado ya detallado.

$$X_{i,j}^{k,m} = \begin{cases} 1 & \text{si } \exists\, h \in \{1, \ldots, |\sigma^{k,m}| - 2\} / \\ & \sigma^{k,m}(h) = i \wedge \sigma^{k,m}(h+1) = j \\ 0 & \text{En otro caso} \end{cases} \begin{array}{l} \forall k \in \{1, \ldots, K\}, \\ \forall m \in \{1, \ldots, M\}, \\ \forall i, j \in \{-N, \ldots, N\}) \end{array}$$

También se definen las siguientes funciones para facilitar la notación:

$$gt(i,j) = \begin{cases} 1 & \text{si } i > j \\ 0 & \text{En otro caso} \end{cases}$$

$$eq(i,j) = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{En otro caso} \end{cases}$$

Adicionalmente, a partir de las anteriores se definen las siguientes variables:

$$V_k^m = \max(X_{i,j}^{k,m})\ (\forall i, j \in \{-N, \ldots, N\}, \quad \forall k \in \{1, \ldots, K\}, \forall m \in \{1, \ldots, M\}) \tag{4.2}$$

$$NV^m = \sum_{k=1}^{K} V_k^m\ (\forall m \in \{1, \ldots, M\}) \tag{4.3}$$

$$NV_{\max} = \max(NV^m)(\forall m \in \{1, \ldots, M\}) \tag{4.4}$$

$$Match^{k,m}(i,j) = \begin{cases} 1 & \text{si } (h_{|\sigma^{k,m}(i)|} = h_{|\sigma^{k,m}(j)|}) \wedge (\sigma^{k,m}(i) < 0 \wedge \sigma^{k,m}(j) > 0) \\ & (\forall i, j \in \{1, \ldots, |\sigma^{k,m}|\}) \\ 0 & \text{En otro caso} \end{cases} \tag{4.5}$$

De manera ordenada, es decir, evaluando en el orden $\{0,\ldots,N\}$ definimos:

$$nextZero^{k,m}(i) = \begin{cases} min(j)/eq(\sigma^{k,m}(j),0)=1\wedge & \text{si } \exists j/eq(\sigma^{k,m}(j),0)=1\wedge \\ (j>i) & (j>i)(\forall k\in\{1,\ldots,K\}, \\ & \forall m\in\{1,\ldots,M\}, \\ & \forall j\in\{1,\ldots,|\sigma^{k,m}|\}) \\ -1 & \text{en otro caso} \end{cases} \tag{4.6}$$

$$prevZero^{k,m}(i) = \begin{cases} max(j)/eq(\sigma^{k,m}(j),0)=1\wedge & \text{si } \exists j/eq(\sigma^{k,m}(j),0)=1\wedge \\ (j<i) & (j<i)(\forall k\in\{1,\ldots,K\}, \\ & \forall m\in\{1,\ldots,M\}, \\ & \forall j\in\{1,\ldots,|\sigma^{k,m}|\}) \\ -1 & \text{en otro caso} \end{cases} \tag{4.7}$$

$$prev^{k,m}(0) = -1$$

$$prev^{k,m}(i) = \begin{cases} min(j)/Match^{k,m}(i,j)=1\wedge & \text{si } \exists j/Match^{k,m}(i,j)=1\wedge(j<i)\wedge \\ (j<i)\wedge(j>prev^{k,m}(l))/ & (j>prev^{k,m}(l)) \\ h_{|\sigma^{k,m}(l)|}=h_{|\sigma^{k,m}(i)|} & (\forall k\in\{1\ldots,K\}, m\in\{1,\ldots,M\}, \\ & l\in\{1,\ldots,|\sigma^{k,m}|\}/h_{|\sigma^{k,m}(l)|}=h_{|\sigma^{k,m}(i)|}, \\ & j\in\{1,\ldots,|\sigma^{k,m}|\}) \\ -1 & \text{en otro caso} \end{cases} \tag{4.8}$$

$$prev2^{k,m}(0) = -1$$

$$prev2^{k,m}(i) = \begin{cases} min(j)/Match^{k,m}(i,j)=1\wedge & \text{si } \exists j/Match^{k,m}(i,j)=1\wedge(j<i)\wedge \\ (j<i)\wedge(j>prev2^{k,m}(l))/ & (j>prev2^{k,m}(l)) \\ h_{|\sigma^{k,m}(l)|}=h_{|\sigma^{k,m}(i)|} & (\forall k\in\{1\ldots,K\}, m\in\{1,\ldots,M\}, \\ & l\in\{1,\ldots,|\sigma^{k,m}|\}/h_{|\sigma^{k,m}(l)|}=h_{|\sigma^{k,m}(i)|}, \\ & j\in\{prevZero(i),\ldots,i\} \\ -1 & \text{en otro caso} \end{cases} \tag{4.9}$$

$$weight^{k,m}(i) = \begin{cases} \sum_{j=i}^{nextZero^{k,m}(i)} d_{|\sigma^{k,m}(j)|} gt(\sigma^{k,m}(j), 0) - & \text{si } \sigma^{k,m}(i) = 0 \\ \sum_{j=i}^{nextZero^{k,m}(i)} gt(prev2^{k,m}(j), -1) d_{|\sigma^{k,m}(j)|} & \\ weight^{k,m}(i-1) - \frac{\sigma^{k,m}(i)}{|\sigma^{k,m}(i)|} d_{|\sigma^{k,m}(i)|} & \text{en otro caso} \\ & (\forall i \in \{1, \ldots, |\sigma^{k,m}(i)|\}, \\ & \forall k \in \{1, \ldots, K\}, \\ & \forall m \in \{1, \ldots, M\}) \end{cases} \tag{4.10}$$

$$DR^{k,m} = \sum_{i=-N}^{N} \sum_{j=-N}^{N} X_{i,j}^{k,m} C_{|i|,|j|} (\forall k \in \{1, \ldots, K\}, \forall m \in \{1, \ldots, M\}) \tag{4.11}$$

$$\begin{aligned} UD_t(0) =& 0 \\ UD_t(m) =& (\sum_{k=1}^{K} \sum_{i=1}^{|\sigma^{k,m}|} gt(\sigma^{k,m}(i), 0) eq(t, h_{|\sigma^{k,m}(i)|}) - \sum_{k=1}^{K} \sum_{i=1}^{|\sigma^{k,m}|} gt(prev^{k,m}(i), -1) \\ & eq(t, h_{|\sigma^{k,m}(i)|})) (\forall m \in \{1, \ldots, M\}, \forall t \in \{1, \ldots, T\}) \end{aligned} \tag{4.12}$$

$$\begin{aligned} LD_t(0) =& 0 \\ LD_t(m) =& (\sum_{k=1}^{K} \sum_{i=1}^{|\sigma^{k,m}|} gt(0, \sigma^{k,m}(i)) eq(t, h_{|\sigma^{k,m}(i)|}) - \sum_{k=1}^{K} \sum_{i=1}^{|\sigma^{k,m}|} gt(prev^{k,m}(i), -1) \\ & eq(t, h_{|\sigma^{k,m}(i)|})) (\forall m \in \{1, \ldots, M\}, \forall t \in \{1, \ldots, T\}) \end{aligned} \tag{4.13}$$

$$\begin{aligned} HUD_t(0) =& 0 \\ HUD_t(m) =& HUD_t(m-1) - LD(m-1) + UD(m) \\ & (\forall m \in \{1, \ldots, M\}, \forall t \in \{1, \ldots, T\}) \end{aligned} \tag{4.14}$$

$$H_t = max(HUD_t) (\forall t \in \{1, \ldots, T\}) \tag{4.15}$$

$$de_i = m \in \{1, \ldots, M\} / \left(\sum_{k=1}^{K} \sum_{j=1}^{|\sigma^{k,m}|} eq(\sigma^{k,m}(j), i) \right) = 1 \; \forall i \in \{1, \ldots, N\} \tag{4.16}$$

$$pi_{|i|} = m \in \{1, \ldots, M\} / \left(\sum_{k=1}^{K} \sum_{j=1}^{|\sigma^{k,m}|} eq(\sigma^{k,m}(j), i) \right) = 1 \; \forall i \in \{-1, \ldots, -N\} \quad (4.17)$$

La ecuación 4.2 indica si el vehículo k en el día m está siendo usado o no. La ecuación 4.3 indica el número de vehículos que se usa en cada día. La ecuación 4.4 indica la cantidad máxima de vehículos que se necesitan para satisfacer la demanda diaria de vehículos, es decir, la cantidad de vehículos que se necesitan adquirir. La ecuación 4.5 indica todas las coincidencias donde se realiza una entrega de herramienta por un vehículo k en el día m, posterior a una recogida de una herramienta del mismo tipo por el mismo vehículo. La ecuación 4.8 realiza todas las asignaciones de forma óptima de las herramientas que se pueden reutilizar en la misma ruta. La ecuación 4.10 calcula el peso llevado en los vehículos en cada instante de visita a cada cliente en la ruta, sin embargo, si $\sigma^{k,m}(i) = 0$ calcula el peso de aquellos elementos que se es necesario cargar desde el almacén. La ecuacion 4.11 calcula la distancia que recorre cada vehículo en su ruta para satisfacer las demandas de los clientes. La ecuación 4.12 calcula la cantidad de herramientas necesarias para satisfacer las demandas del día mientras que la ecuación 4.13 calcula la cantidad de herramientas que se liberan por día. Las ecuaciones 4.14 calcula las herramientas que están en uso en cada día. La ecuación 4.15 obtiene el máximo de herramientas utilizadas. La ecuación 4.16 obtiene el día que se hace la entrega de la herramienta. La ecuación 4.17 obtiene el día que se hace la recogida de la herramienta. Con las definiciones anteriores podemos definir la función costo del CVRPTWUI:

$$f(x) = \sum_{k=1}^{K} \sum_{m=1}^{M} \sum_{i=-N}^{N} \sum_{j=-N}^{N} b_3 X_{i,j}^{k,m} C_{i,j}^{k,m} + \sum_{m=1}^{M} NV^m b_2 + NV_{max} b_1 + \sum_{t=1}^{T} a_t H_t \quad (4.18)$$

Sujeto a:

$$de_i \in \{aa_i, \ldots, ee_i\} (\forall i \in \{1, \ldots, N\}) \quad (4.19)$$

$$(pi_i - de_i) = g_i (\forall i \in \{1, \ldots, N\}) \quad (4.20)$$

$$H_i \leq p_i (\forall i \in \{1, \ldots, T\}) \quad (4.21)$$

$$DR^{k,m} \leq D (\forall k \in \{1, \ldots, K\}, \forall m \in \{1, \ldots, M\}) \quad (4.22)$$

$$max(weight^{k,m}) \leq Q (\forall k \in \{1, \ldots, K\}, \forall m \in \{1, \ldots, M\}) \quad (4.23)$$

TI	CV	CVD	CH	CD
1	2	3	1	4
2	3	2	1	4
3	1	3	2	4
4	3	1	2	4
5	4	3	1	2

TABLA 4.1: Importancias relativas entre los costos para distintos tipos de instancia (TI = Tipo de Instancia, CV = Costo del Vehículo, CVD = Costo del uso del Vehículo por Día, CH = Costo de las Herramientas y CD = Costo de la Distancia).

La función objetivo se define en la ecuación 4.18 donde $C_{|i||j|}$ es la distancia de viaje del cliente i al cliente j. Nótese que, en el caso de los clientes, cuando $i < 0$ indica que se está realizando una recogida de herramienta, cuando $i = 0$ indica que el vehículo se encuentra en el almacén y cuando $i > 0$ indica que se está entregando una herramienta. La ecuación 4.19 restringe el valor de de_i dentro del rango de la ventana de tiempo para las entregas de herramienta. La ecuación 4.20 garantiza que la diferencia de tiempo entre la entrega y recogida de la herramienta sea exactamente g_i, donde g_i indica el total de tiempo que la herramienta es ocupada por el cliente. La ecuación 4.21 garantiza que el límite de herramientas de cada tipo no sea excedido. La ecuación 4.22 garantiza que el límite de distancia recorrida permitida para cada vehículo no sea excedido. La ecuación 4.23 garantiza que la capacidad no sea excedida para cada vehículo.

4.2 Pruebas de Desempeño

VeRoLog-ORTEC publicó un conjunto de pruebas de desempeño junto a la descripción del problema. El nombre de estas instancias llevan una nomenclatura que especifica el número de peticiones de los clientes que contienen, además del total de días en el que la instancia se desarrolla. Así, la instancia VeRoLog_r100d5_1 implica que la instancia está conformada por 100 peticiones y se extiende por un total de 5 días. El último valor es el tipo de instancia, tal y como se describe a continuación. Las pruebas de desempeño cuentan con 100, 500 y 1000 peticiones y abarcan desde 5 días hasta 30, incluyendo casos con 10, 15 y 25 días. Adicionalmente, estas pruebas pueden ser clasificadas en 5 tipos diferentes dependiendo del tipo de costo más significativo. En otras palabras, existen grupos de instancias donde el costo más importante es la adquisición de vehículos, otras donde el costo mayor es el asociado a la adquisición de herramientas, etc.

En la Tabla 4.1 se especifica el orden relativo entre los costos para los distintos tipos de instancia. Aunque este orden puede ser importante, no es la única característica requerida para clasificar correctamente las instancias. Por ejemplo, el costo asociado a la distancia es por unidad de distancia recorrida por lo que si los clientes se encuentran muy espaciados en el espacio geográfico el costo podría superar a los

demás costos asociados. En nuestros experimentos se han seleccionado las instancias con 100 peticiones y 5 días ya que se consideró que tienen una complejidad lo suficientemente desafiante para el algoritmo que se estaba proponiendo. Para lidiar con instancias mayores se deberían realizar implementaciones más eficientes y hacer uso de estructuras de datos más avanzadas para reducir el costo computacional de las implementaciones que se tienen hasta el momento.

4.3 Métodos Propuestos

El problema del CVRPTWUI planteado por VeRoLog-ORTEC se caracteriza por integrar dos problemas en uno: un problema de asignación y un problema de optimización de rutas. En esta sección se discuten los métodos planteados para resolver el problema que incluyen desde una búsqueda local iterada hasta un Algoritmo Memético con manejo explícito de diversidad y modelo basado en islas. Además, de los métodos se discute la función fitness usada para los algoritmos usados en este capítulo, la cual es común para todos los métodos.

4.3.1 Función de Fitness

La función Fitness es importante en los algoritmos evolutivos y otros métodos de optimización debido a que ofrecen una medida de que tan prometedora es una solución y da la pauta para comparar dos soluciones diferentes. En esta sección describimos la función fitness usada en los diferentes algoritmos propuestos. En concreto, en este caso se está usando una función de fitness $ff(x)$ que tiene dos componentes $p(x)$ y $f(x)$ tal y como se aprecia en la ecuación 4.24 y se usa el método lexicográfico de comparación. Nótese que al utilizar el método lexicográfico, la forma de determinar si una solución x es mejor que una solución y, lo que se expresa como $x < y$ considera la expresión dada en la ecuación 4.25.

$$ff(x) = (p(x), f(x)) \tag{4.24}$$

La primera componente (ver ecuación 4.26) es una medida relacionada con las restricciones no cumplidas, de forma que este valor se hace 0 cuando se cumplen todas las restricciones. La primera sumatoria se refiere a la penalización por el peso excedido en cada vehículo, la segunda sumatoria se refiere a la penalización por exceder el límite de herramientas que se puede adquirir y, finalmente, la tercera sumatoria se refiere a la distancia que se excede por los vehículos. Por otro lado, la segunda componente (ver ecuación 4.27) es la función objetivo del problema, es decir, el costo de la solución candidata. La primera sumatoria cuantifica el costo relativo a los viajes, es decir, la suma de los costos asociados a las aristas recorridas, la segunda sumatoria está relacionada con el uso de vehículos diarios, la tercera sumatoria se refiere a la adquisición de vehículos para cumplir con la demanda del

uso diario de los mismos, mientras que la última sumatoria se refiere al costo por la adquisición de las herramientas para satisfacer las demandas de los clientes.

$$x \leq y \Leftrightarrow (p(x) < p(y)) \vee ((p(x) == p(y)) \wedge (f(x) < f(y))) \tag{4.25}$$

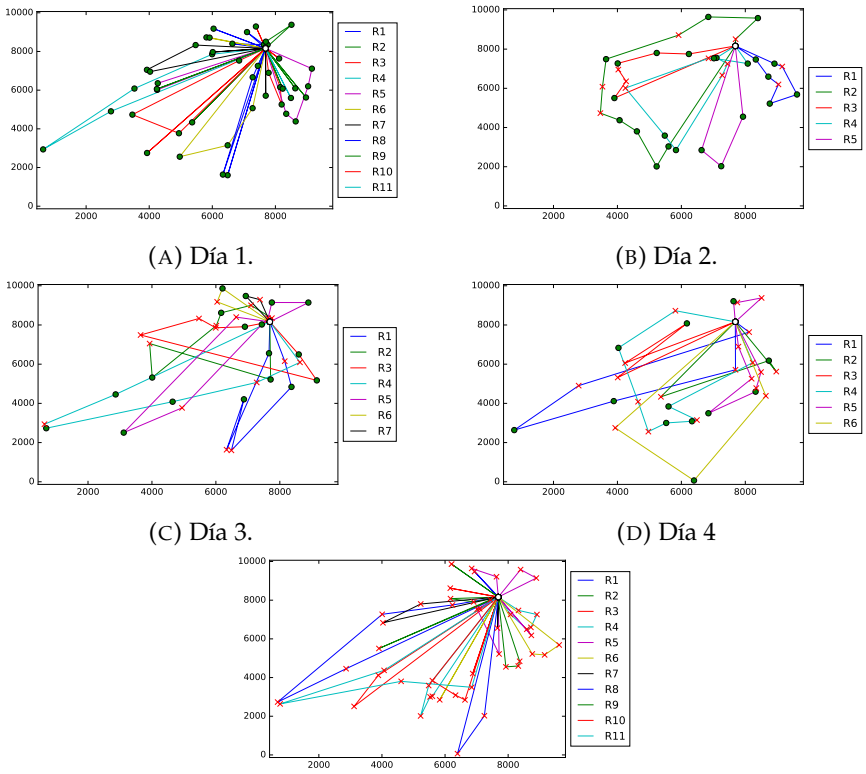
$$\begin{aligned} p(x) = & \sum_{m=1}^{M} \sum_{k=1}^{K} gt(max(weight^{km}), Q)(max(weight^{km}) - Q) \\ & + \sum_{t=1}^{T} gt(HU_t, p_t)(HU_t - p_t) \\ & + \sum_{m=1}^{M} \sum_{k=1}^{K} gt(DR^{km}, D)(DR^{km} - D) \end{aligned} \tag{4.26}$$

$$f(x) = \sum_{k=1}^{K} \sum_{m=1}^{M} \sum_{i=-N}^{N} \sum_{j=-N}^{N} b_3 X_{ij}^{km} C_{ij}^{km} + \sum_{m=0}^{M} NV^m b_2 + NV_{max} b_1 + \sum_{t=1}^{T} a_t H_t \tag{4.27}$$

4.3.2 Método de Búsqueda Local Iterada

La búsqueda local iterada (Iterated Local Search - ILS) es unos de los primeros métodos de trayectoria y, por ende, uno de los más sencillos. La implementación de una ILS se basa en definir un vecindario, recorrerlo habitualmente con una búsqueda local por escalada, y realizar un reinicio parcial — también llamado perturbación — al llegar a un óptimo local. Posteriormente, se inicia de nuevo la búsqueda local y este proceso se repite hasta que una condición de paro es alcanzada. Una de las ventajas de la ILS es que es fácil de implementar y suele obtener resultados bastante mejores que la búsqueda local simple. Dado que casi no existen trabajos anteriores con estas instancias, se decidió implementar la ILS para tener un punto de partida con el que comparar.

El método de ILS que proponemos para el problema cuenta con una serie de transformaciones, que definen el vecindario local, además de una estrategia para realizar el reinicio parcial. En específico, la búsqueda local considera tres niveles de vecindad las cuales son utilizados de una forma parecida al algoritmo de descenso de vecindarios variables (Variable Neighborhood Descent - VND) pero con una pequeña diferencia. En el VND, siempre se selecciona el mejor individuo de un nivel dado. Sin embargo, en nuestro caso, el vecindario es explorado de una forma aleatoria y se acepta el primer vecino que mejore la solución actual. En la ILS propuesta se usa cada nivel sólo cuando la solución actual es un óptimo local con respecto a los niveles más bajos. Esto quiere decir que, si se consigue una mejora en cualquier nivel, se tiene que retornar al primer nivel para conseguir llevar la nueva solución a nuevos óptimos locales de las vecindades ya exploradas.



(A) Día 1.

(B) Día 2.

(C) Día 3.

(D) Día 4

(E) Día 5.

FIGURA 4.1: Solución obtenida usando el enfoque del 2-OPT

Las definiciones de las vecindades en cada nivel van enfocadas a optimizar partes en específico de la solución. La primera etapa está enfocada a optimizar las rutas actuales considerando la vecindad 2-opt. Para los tamaños de ruta consideradas en las instancias, las vecindades basadas en 2-opt suelen llegar a óptimos globales, por lo que no se requerían definiciones más complejas. De hecho, para varias de las soluciones arrojadas por el optimizador se verificó que en realidad las rutas eran óptimas realizando una búsqueda completa por enumeración. Hay que destacar que cuando hablamos del óptimo global de cada ruta, nos referimos únicamente a la configuración actual de las rutas sin realizar intercambios entre rutas diferentes, es decir, si un día tiene 5 rutas las 5 rutas podrían estar en sus óptimos globales pero el día no necesariamente estaría en el óptimo global ya que se podría necesitar migrar alguna petición de una ruta a otra.

La segunda etapa considera dos tipos de vecindades, la primera considera un vecindario con los movimientos de una petición de una ruta a cualquier otra ruta, mientras que la segunda considera los intercambios de peticiones entre rutas. Nótese que al realizar un intercambio entre vehículos de diferentes días, se tiene que realizar una realocación del complemento de las peticiones. Por ejemplo, cuando una entrega se mueve de día entonces la recuperación de la herramienta también debe de moverse de día. Con el objetivo de evitar movimientos no prometedores, en un

primer paso, se verifica si el movimiento sin el complemento obtiene algún tipo de ganancia, si es así, se localiza la mejor posición para el complemento y entonces el movimiento es aceptado si se obtiene algún tipo de ganancia. En otro caso no se acepta el movimiento. Finalmente, para la tercera etapa se considera el uso de vehículos adicionales, es decir, al contrario de la etapa anterior que solo considera movimientos entre vehículos que tienen asignado al menos una petición, en esta etapa se consideran movimientos de cualquier petición a vehículos vacíos. Originalmente, las tres etapas estaban unidas en un vecindario grande, sin embargo, esto conducía a un pobre desempeño porque tendía a generar soluciones con muchos vehículos, lo que era una forma fácil de generar soluciones factibles pero con un costo grande. En consecuencia, cuando se considera un vecindario grande, la deceptividad de la función fitness que viene dirigida por movimientos que reducen de forma rápida las penalizaciones provoca una importante degradación en el comportamiento en lo relativo a la función de fitness. En la Fig. 4.1 se ejemplifica una solución obtenida usando la optimización por etapas. Hay que notar que usar el 2-OPT como optimizador para las rutas es suficiente, por ejemplo, en los días que no hay combinación entre entregas y recogidas (Ver SubFig. 4.1a y SubFig. 4.1e) la solución obtenida no presenta cruces típicos de soluciones no óptimas. Sin embargo, en el día 4 (Ver SubFig. 4.1d) la ruta 3 presenta un cruce en la ruta. Aunque en principio podría parecer que se debe a un mal proceso de optimización, este movimiento más largo se debe a que primero realiza una recogida de herramienta y posteriormente una entrega, por lo que la herramienta recogida con anterioridad es reutilizada implicando un menor uso de herramientas y, por lo tanto, un menor costo de la ruta. Una de las problemáticas es que al disponer de varios costos dependientes es difícil poder visualizar mediante gráficas sencillas todos los costos asociados.

En el Alg. 10 se describe el procedimiento seguido por la ILS propuesta. Los pasos 2 al 21 se repiten hasta alcanzar un criterio de paro. Primero, en el paso 4 se inicializan los tres vecindarios que serán usados por cada etapa anteriormente descrita. Del paso 5 al 16 se recorre vecindario por vecindario, comenzando en el vecindario 1, continuando con el vecindario 2 y terminando con el vecindario 3. Del paso 6 al 15 se realiza la búsqueda local por escalada estocástica hasta que el vecindario no tenga movimientos por realizar. En el paso 9 si se realiza un movimiento que mejore al individuo actual, se reemplaza al individuo local (paso 10) y se reinician los vecindarios para comenzar con el vecindario 1 (paso 11). En los paso 17 al 19 se intercambia el individuo que resulta de la búsqueda local si este individuo es mejor que el encontrado en otras iteraciones de búsquedas locales. En el paso 20 se realiza una perturbación parcial, que en nuestro caso consistía en eliminar un vehículo no vacío e insertar las peticiones que satisfacían este vehículo en otros vehículos escogidos de forma aleatoria.

Algoritmo 10: Esquema General de la búsqueda local iterada

```
   Data: Criterio de Parada
   Result: El mejor individuo encontrado
1  Inicialización del individuo;
2  Asignar IndividuoActual a individuo;
3  while Criterio de Parada no sea alcanzado do
4      Inicializar el Vecindario1, Vecindario2, Vecindario3;
5      foreach Vecindario ∈ { Vecindario1, Vecindario2, Vecindario3 } do
6          while Vecindario tenga movimientos do
7              Escoger un movimiento de forma aleatoria del Vecindario ;
8              Realizar el movimiento a IndividuoActual para crear
                IndividuoTemporal ;
9              if IndividuoTemporal es mejor que IndividuoActual then
10                 Asignar a IndividuoTemporal como Individuo ;
11                 Reiniciar el vecindario y comenzar en vecindario1 ;
12             else
13                 Eliminar movimiento del vecindario;
14             end
15         end
16     end
17     if IndividuoActual es mejor que individuo then
18         asignar a IndividuoActual como individuo
19     end
20     perturbar parcialmente el individuo y asignarlo a IndividuoActual ;
21 end
```

4.3.3 Algoritmo Evolutivo

Los resultados obtenidos por el ILS no son lo suficientemente buenos para quedar cerca del BKS de las instancias con las que se realizaron pruebas. Con el propósito de mejorar los resultados anteriores, se propone el uso de un MA (Ver Alg. 11). Este MA consta de los elementos clásicos de un algoritmo evolutivo como lo podrían ser la selección de padres (paso 4) o la utilización del operador de cruce (paso 5), entre otros. Además, usa una estrategia para mejorar la descendencia generada. En específico se integró con una búsqueda por escalada simple y para la definición del vecindario se tomó el vecindario definido para la ILS propuesta anteriormente. Nótese que muchos meméticos mejoran considerablemente al utilizar estrategias más complejas que la búsqueda local simple y que anteriormente en el caso del CVRPTW se había usado recocido simulado. Sin embargo, para este caso debido al costo computacional asociado a la evaluación de los individuos no fue posible integrarlo con algoritmos más complejos. Además, se hace uso de la selección por torneo binario como selección de padres, RMDDC como operador de selección de sobrevivientes, y se usa un operador de cruza muy simple basado en intercambiar rutas completas de días, al que se denomina Day-Based Crossover - DBX. Adicionalmente, tras realizar una primera búsqueda local (paso 10), se realiza una perturbación (paso 11) para posteriormente realizar otra búsqueda local (paso 12). Del individuo resultante

Algoritmo 11: Esquema General del Algoritmo Memético

Data: Tamaño de la Población, Criterio de Parada, Operador de Cruza, Operador de Selección de Padres, Mecanismo de Selección de Sobrevivientes
Result: El mejor individuo encontrado

```
1  Inicialización de la población;
2  while Criterio de Parada no sea alcanzado do
3      while |descendencia| <tamaño de la descendencia deseado do
4          Seleccionar dos padres con el operador de selección de padres;
5          Cruza de los padres usando el operador de cruza;
6          Insertar nuevo individuo a descendencia;
7      end
8      DescendenciaFinal = ∅ ;
9      foreach hijoCurrent en descendencia do
10         hijo1 = resultado de búsqueda local a hijoCurrent;
11         hijo2 = perturbación a hijo1;
12         hijo2 = resultado de búsqueda local a hijo2;
13         DescendenciaFinal = DescendenciaFinal ∪ { mejor(hijo1, hijo2) } ;
14     end
15     DescendenciaFinal = DescendenciaFinal ∪ padres ;
16     Seleccionar la nueva generación con el mecanismo de selección de
         sobrevivientes;
17 end
```

Algoritmo 12: Operador de cruza basado en días

Data: Individuo1, Individuo2
Result: El hijo generado

```
1  Seleccionar un día 'x' del Individuo2;
2  Reemplazar el día 'x' del Individuo1 con el día del Individuo2;
3  Aplicar fase de reparación para evitar peticiones duplicadas o faltantes y
     distancias no válidas entre entregas y recogidas;
```

de la primera búsqueda local y la segunda búsqueda local se mantiene el que mejor aptitud haya obtenido y ese es considerado el descendiente (paso 13). En el paso 16 se usa el RMDDC para hacer una selección de los individuos que van sobreviven a la siguiente generación. El DBX toma en cuenta los días para hacer la cruza, en otras palabras, cuando dos individuos seleccionados se usan para generar descendencia la forma en que se realiza es intercambiando un día seleccionado del segundo individuo y reemplazandolo en el primer individuo. Realizar esta operación podría generar soluciones no válidas, por ejemplo soluciones con peticiones duplicadas o peticiones faltantes y con distancias no válidas entre las entregas y las recogidas. Por ello es necesario ejecutar una fase de reparación después del operador de cruza, la cual está descrita en el Alg. 12.

Nótese que el RMDDC requiere la definición de una medida de distancia. La medida de distancia está basada exclusivamente en la solución dada al problema de asignación. Así, la distancia entre dos individuos es el número de peticiones que

Algoritmo 13: Esquema General del Algoritmo Memético con enfoque en islas

```
Data: Tamaño de la Población, Criterio de Parada, Operador de Cruza,
      Operador de Selección de Padres, Mecanismo de Selección de
      Sobrevivientes
Result: El mejor individuo encontrado
Inicialización de la población;
while Criterio de Parada no sea alcanzado do
    while descendencia <tamaño de la descendencia deseado do
        Seleccionar dos padres con el operador de selección de padres;
        Cruza de los padres usando el operador de cruza;
        Insertar nuevo individuo a descendencia;
    end
    DescendenciaFinal = ∅ ;
    foreach hijoCurrent en descendencia do
        hijo1 = resultado de búsqueda local a hijoCurrent;
        hijo2 = perturbación a hijo1;
        hijo2 = resultado de búsqueda local a hijo2;
        DescendenciaFinal = DescendenciaFinal ∪ { mejor(hijo1, hijo2) } ;
    end
    DescendenciaFinal = DescendenciaFinal ∪ padres ;
    aplicar el mecanimos de selección a cada region de 1/4 de psicina de
     seleccion;
end
```

son colocados en diferentes días. Nótese que dada la naturaleza del RMDDC se evitará la entrada de clones, identificándose como clones aquellas soluciones que no tienen asignaciones de clientes a diferentes días independientemente de las rutas finales que se generen. Esto en cierta forma podría reducir la capacidad de intensificación del esquema, pero se deja como trabajo futuro definir medidas de distancia que tomen en cuenta tanto los días seleccionados para cada cliente, como las propias rutas en sí.

Diversidad adicional con enfoque basado en islas

Las instancias propuestas por VeRoLog-ORTEC parecen presentar deceptividad, en el sentido que dada una solución candidata no factible, la forma más sencilla de convertirla en factible es mediante la inclusión de más coches, presentando así un camino de descenso en la función de fitness que no lleva a soluciones de alta calidad. En nuestros experimentos con el MA que integra el RMDDC como método de control de diversidad detectamos que en algunas instancias no se alcanzaba el BKS debido a que era necesario usar un vehículo menos. Sin embargo, conseguir eliminar un vehículo haciendo uso de las vecindades es complicado, pues al tratar de ir pasando peticiones de un vehículo a otro, aumentaba el costo, y sólo hasta que se elimina el vehículo por completo se conseguía la reducción del costo asociado a usar un vehículo menos.

Por ello se trató de abordar el problema con un enfoque diferente. En este enfoque se intenta mantener soluciones con diferente número máximo de vehículos. En el algoritmo básico se dispone de un límite superior de vehículos el cual se usaba para toda la población. En este enfoque se cambia este detalle dividiendo la población en grupos (similar al modelo de islas) y cada grupo contiene un número máximo de vehículos que puede usar. El RMDDC se aplica a cada agrupamiento para tratar de mantener la diversidad para diferentes números de vehículos pero para la cruza se usa toda la población, es decir, dos individuos de diferente agrupamiento pueden cruzarse, lo cual difiere del modelo de islas estándar. Tras cruzar se podría generar un individuo con una cantidad de vehículos que supere al límite establecido en la isla, es por ello que es necesario agregar una nueva operación a la etapa de reparación. Esta nueva operación se encarga de dejar a los descendientes en cada agrupamiento con una cantidad de vehículos que no exceda el límite superior establecido, para ello, cuando se excede la cantidad, en el día o los días que excedan este límite se escoge un vehículo aleatoriamente y se reasignan todas las peticiones que atiende el mismo, tratando de colocar las peticiones en diferentes días al día usado para realizar la cruza —al realizar la cruza se escoge un día del segundo padre y se reemplaza el día en el primer padre—. Nótese que en algunos casos las peticiones no pueden ser movidas de día ya que su ventana de tiempo solo abarca un día, mientras que para los demás casos es también necesario mover la contraparte para no tener distancias no válidas entre recogidas y entregas. El esquema general de este algoritmo se encuentra detallado en el Alg. 13.

4.3.4 Validación Experimental

Con el objetivo de validar los algoritmos, se usaron las instancias propuestas por VeRoLog-ORTEC con un máximo de 5 días en su ventana de tiempo y para reducir el espacio de búsqueda en todas las propuestas se aplicó un límite en el número de rutas que se pueden crear. En el caso del ILS no se requiere ningún parámetro salvo el criterio de paro que se estableció a 24 horas. Para el caso de los algoritmos poblacionales, el único parámetro adicional que tiene que ser especificado es el tamaño de la población, el cual fue asignado a 50 individuos, fijando el criterio de paro también a 24 horas. Además, debido a la naturaleza estocástica de las propuestas, se realizaron 30 ejecuciones independientes. Los algoritmos fueron implementados en c++ y compilados con g++ 4.8.2 y todas las pruebas fueron ejecutadas en un Intel(R) Xeon(R) CPU E5-2620 v2 (2.10 GHz - 32 GB de RAM).

En la table 4.2 se resumen los resultados obtenidos con ILS. Al comparar frente a la mejor solución conocida se puede concluir que es necesario reducir la adquisición total o uso de vehículos diarios para acercarse al BKS. Por ejemplo, en la instancia VeRoLog_r100d5_1 es necesario reducir la adquisición de 3 vehículos mientras que en la instancia VeRoLog_r100d5_2 es necesario reducir el uso de 10 vehículos, reduciéndolo en promedio 2 vehículos por día. Para la instancia VeRoLog_r100d5_3 con la reducción de la adquisición de un vehículo se obtendrían resultados similares

Instancia	BKS	Mejor	Media
VeRoLog_r100d5_1	1,552,435,049	1,552,947,881	1,558,042,576
VeRoLog_r100d5_2	996,709,544	1,000,396,810	1,005,062,364
VeRoLog_r100d5_3	119,957,689	142,135,963	160,061,627.9
VeRoLog_r100d5_4	1,359,088,350	1,628,532,682	1,650,297,280

TABLA 4.2: Resultados obtenidos usando la búsqueda local iterada (BKS = Mejores soluciones conocidas [Best-Known Solutions]).

Instancia	BKS	Mejor	Media
VeRoLog_r100d5_1	1,552,435,049	1,562,976,044	1,570,753,699
VeRoLog_r100d5_2	996,709,544	1,000,389,253	1,009,470,042
VeRoLog_r100d5_3	119,957,689	160,207,676	161,034,523.8
VeRoLog_r100d5_4	1,359,088,350	1,659,348,437	1,678,246,228

TABLA 4.3: Resultados obtenidos usando el MA con manejo de diversidad (BKS = mejores soluciones conocidas[Best-Known Solutions]).

al BKS y, finalmente, en la instancia VeRoLog_r100d5_4 la diferencia entre el mejor resultado alcanzado con la ILS y el BKS radica en que se usan en torno a 2 vehículos por día adicionales en nuestra propuesta.

En general, a la ILS le es muy difícil reducir los costos relacionados con los vehículos, ya que para eliminar un vehículo se tiene que realizar más de un movimiento en el que se vayan eliminando asignaciones de pedidos a vehículos poco cargados pero hasta que no se eliminen todas las asignaciones no se consigue el beneficio de reducir el vehículo, por lo que se debe desarrollar un proceso de intensificación más potente para este problema. El MA con el RMDDC que proponemos intenta solucionar este problema y los resultados obtenidos por el mismo se detallan en la Tabla 4.3.

Los resultados alcanzados por el MA con RMDDC son peores que los alcanzados con la ILS. La principal razón es que la búsqueda local implementada tarda mucho tiempo para lograr llevar a una solución candidata a un mínimo local y, por este motivo, el número de generaciones que desarrolla el MA es muy reducido, siendo el MA incapaz de alcanzar soluciones de alta calidad con ese tiempo. En consecuencia, decidimos paralelizar el MA para incrementar el número de generaciones alcanzadas. Esta paralelización —la cual se realizó en MPI— envía a cada proceso un hijo generado, y el proceso se encarga de realizar la búsqueda local. Con este esquema se realizaron las 50 búsquedas locales de forma simultánea, consiguiendo así más iteraciones del MA. La Tabla 4.4 detalla los resultados obtenidos, hay que notar que en algunas instancias como la VeRoLog_r100d5_1 la diferencia del costo entre la mejor solución lograda por el MA paralelizado y la mejor encontrada, es de $\approx 200,000$ correspondiendo al costo de la adquisición de un vehículo. Esto quiere decir que, a diferencia de los algoritmos anteriores, logró obtener resultados con diferencia de solo un vehículo. Sin embargo, se mantuvo una búsqueda local demasiado simple por lo que los resultados aunque ya se acercan a los mejores conocidos, siguen sin

Instancia	BKS	Mejor	Media
VeRoLog_r100d5_1	1,552,435,049	1,552,665,590	1,552,680,349.2
VeRoLog_r100d5_2	996,709,544	1,008,769,218	1,012,180,312.9
VeRoLog_r100d5_3	119,957,689	140,722,230	154,482,308.3
VeRoLog_r100d5_4	1,359,088,350	1,508,918,443	1,571,783,577.3

TABLA 4.4: Resultados obtenidos usando el MA con manejo de diversidad y paralelizado (BKS = mejores soluciones conocidas[Best-Known Solutions]).

Instancia	BKS	Mejor	Media
VeRoLog_r100d5_1	1,552,435,049	1,553,151,741	1,568,658,481.5
VeRoLog_r100d5_2	996,709,544	1,001,605,979	1,010,266,091.7
VeRoLog_r100d5_3	119,957,689	141,934,390	160,116,396.8
VeRoLog_r100d5_4	1,359,088,350	1,691,457,263	1,732,700,667.6

TABLA 4.5: Resultados obtenidos usando el MA con manejo de diversidad usando un enfoque basado en islas (BKS = mejores soluciones conocidas[Best-Known Solutions]).

alcanzarlo.

Finalmente, con el fin de tratar de lidiar con el problema del número de vehículos se usó el esquema basado en islas descrito anteriormente, y cuyos resultados se muestran en la tabla 4.5. Este esquema presenta el mismo problema que el MA con el RMDDC: se evolucionan muy pocas generación debido a la larga duración de las búsquedas locales por lo que lo resultados alcanzados con este esquema son similares al esquema que no contiene el enfoque en islas.

En base a estos resultados se concluye que para tratar con este tipo de problemas más complejos es necesario desarrollar algoritmos más complejos, pues la extensión natural que se realizó de las vecindades para lidiar con este tipo de espacios de búsqueda más grandes provocó que la mayor parte del tiempo se dedique a realizar búsquedas locales que en muchos casos no dirigen la solución hacia el tipo de solución que sabemos que son las mejores conocidas en el momento actual. Por tanto, los algoritmos desarrollados son capaces de iguales y mejorar en muchos casos los problemas académicos, pero al lidiar con problemas más grandes surgen inconvenientes importantes para los que se considera que se tienen que desarrollar esquemas más complejos haciendo uso de estructuras de datos avanzadas para reducir el costo computacional.

4.3.5 Discusión

En este capítulo se adaptó un algoritmo que había obtenido un comportamiento muy eficiente con problemas académicos a un problema más realista como es el CVRPTWUI. En primer lugar se desarrolló un método de trayectoria que sirvió para realizar validaciones de resultados, pero que no pudo ser integrado con un memético debido a su alto costo computacional. Posteriormente, se adaptó el algoritmo y se

reemplazó el SA que usaba con anterioridad por un método más sencillo, como lo es la búsqueda local por escalada (hill-Climbing Search). El problema del costo computacional asociado a la búsqueda local fue el principal problema que se detectó que impidió obtener buenos resultados. Mientras que en muchos problemas académicos es posible reevaluar soluciones de forma incremental, al lidiar con problemas más complejos esto no es sencillo, lo que provoca que estos procesos sean realmente costosos y que el número de generaciones se reduzca drásticamente. Con el fin de lidiar con esta problemática se propusieron varias propuestas como paralelizar o usar un modelo basado en islas que aunque mejoraron los resultados y permitieron estar en el top 10 de entre 22 equipos, no permitieron alcanzar los mejores resultados conocidos.

En el futuro se quieren explorar diferentes líneas de trabajo. Entre ellas, reducir el tiempo que consume la búsqueda local parece la opción más viable. Para ello se podría redefinir el vecindario o usar estructuras de datos que permitan explorar de mejor forma el vecindario. Una vez mejorado este aspecto se podría incluir el uso de esquemas más complejos como SA u otras métodos de trayectoria que creemos que es primordial para alcanzar las mejores soluciones conocidas.

Capítulo 5

Conclusiones y Trabajos Futuros

En esta tesis se propusieron varios algoritmos para abordar dos variantes del CVRP: el CVRPTW y el CVRPTWUI. El primero de ellos es considerado un problema académico mientras que el segundo está basado en una situación real y añade una mayor complejidad en forma de varias restricciones. Los algoritmos propuestos abarcan desde métodos de trayectoria como lo son la ILS y un SA, hasta un método poblacional, que fue un MA con control explícito de diversidad. Primero, para el CVRPTW, se implementó una variante de SA con el que se abordaron instancias de 100 clientes. Este algoritmo presentó resultados bastantes competitivos en instancias fáciles como las instancias clusterizadas, sin embargo, en instancias más complejas, el resultado fue inferior a los BKS obtenidos con otros optimizadores. En consecuencia, se propuso un MA que integraba el SA como método de trayectoria para mejorar la descendencia de cada generación y como operador de cruza se usó el SBX, que es uno de los operadores más populares en el área del CVRPTW. El SBX obtenía resultados muy prometedores en la literatura, sin embargo, presentaba un comportamiento no muy usual en operadores de cruza, pues este operador era muy destructivo por lo que regiones identificadas como prometedoras no eran intensificadas adecuadamente. Para resolver este problema, se propuso una extensión del mismo resultando en el operador de cruza SBSBX. Se realizaron análisis de calidad y diversidad comparando al SBX con el SBSBX y usando dos métodos diferentes de selección de padres. A partir de la validación, se pudo concluir que al utilizar el SBX, independientemente de la presión inducida por la fase de selección de padres, se mantenía un grado de diversidad similar a lo largo de la ejecución del algoritmo. Sin embargo, con el SBSBX se presentaba convergencia prematura cuando se usaba torneo binario y una diversidad alta cuando se usaba selección aleatoria. La combinación del SBSBX con la selección aleatoria fue la que ofreció los mejores resultados, llegando a generarse una nueva mejor solución que las conocidas en una instancia. A pesar del buen comportamiento del SBSBX con selección aleatoria, la misma presentaba algunas deficiencias en las instancias más complejas, no alcanzando el BKS en las mismas. Dado que se pudo comprobar que el balance entre exploración e intensificación con el esquema anterior no era adecuado, se integró el MA con el RMDDC para controlar de forma explícita la diversidad. Para este esquema se utilizó el torneo binario como

operador de selección de padres, pues a pesar de que presentaba convergencia prematura al usar el esquema generacional, en el caso de usar el RMDDC se evitaba esta problemática. Al integrar el RMDDC con el MA fue necesario realizar el análisis de un parámetro adicional (α), el cual escala la diversidad inicial con la que opera el algoritmo. Para la determinación del α idóneo fue necesario ampliar las pruebas de desempeño a 200 y 400 clientes, debido a que determinar el α con instancias que incluían sólo 100 clientes no era posible debido a que muchos valores de α obtenían resultados muy buenos, consiguiendo superar al estado del arte en varios casos. Con las pruebas en instancias que incluían más de 100 clientes, los análisis del α arrojaron que el valor 0.6 era muy estable para tratar diferentes tipos de instancias. Ya con la determinación del α en 0.6, al algoritmo fue capaz de alcanzar (o superar) los mejores resultados actuales en 27 instancias de 36 instancias utilizadas, siendo capaz de mejorar en 14 de ellas.

En la segunda parte, con respecto al CVRPTWUI, se adaptó el algoritmo propuesto para el CVRPTW que obtenía mejor comportamiento. Al tratarse de un problema con instancias recién planteadas, se inició diseñando un método de trayectoria relativamente simple como lo es la ILS. Esta ILS tiene la característica de ser rápida de implementar además de ser un algoritmo muy simple. Posteriormente, se usó el MA integrándose una búsqueda por escalada simple en lugar del SA o ILS pues debido al coste computacional mayor asociado a este problema, fue imposible realizar integraciones entre el MA y metaheurísticas de trayectoria. Aunque esto es una debilidad muy importante de la propuesta, no fue posible desarrollar esquemas que explorarán la vecindad definida de forma eficiente, por lo que se tuvo que recurrir a una búsqueda por escalada simple. Este MA obtiene resultados que quedarían en el top 10 de 22 participantes en el concurso de VeRoLog-ORTEC, teniendo como diferencia solo un vehículo en algunas instancias. Se propusieron nuevos enfoques basados en paralelización y en un modelo de islas. Sin embargo, aunque se consiguieron ciertos avances en algunos casos, los resultados no fueron muy superiores, dándose que con las implementaciones realizadas el número de generaciones evolucionadas es muy bajo, por lo que el algoritmo memético no puede converger hacia regiones de alta calidad.

A modo de conclusión cabe destacar que con los métodos desarrollados se pudo obtener mejores resultados que el estado del arte en el caso de problemas académicos, y resultados cercanos a los del estado del arte en problemas de mayor complejidad. En este segundo problema, no se desarrollaron operadores genéticos a medida, tal y como se hizo en el caso del problema académico y sólo se usa una búsqueda local simple en lugar de una metaheurística de trayectoria para la intensificación, aspectos que en el futuro se deben mejorar para alcanzar los mejores resultados conocidos.

Hay varias líneas que se quieren explorar en el futuro. Primeramente, con respecto al CVRPTW se podría hacer un análisis más completo para determinar una mejor forma de fijar la temperatura inicial para el SA, ya que en esta tesis no se dedicó

demasiado esfuerzo computacional a ese aspecto. Adicionalmente, se podrían utilizar instancias mayores e incrementar el tiempo de cómputo o utilizar paralelismo, para explorar los límites de los métodos desarrollados en lo referente a la escalabilidad. Otra línea que se podría explorar es integrar otros métodos de trayectoria en el MA. Con respecto al algoritmo propuesto para el CVRPTWUI, el tiempo que tarda la búsqueda local en el MA es demasiado alto, por lo que las versiones actuales evolucionan muy pocas generaciones. Por tanto, se debe estudiar cómo mejorar ese aspecto para explorar de forma más eficiente la vecindad o redefinir la vecindad. Otro aspecto a estudiar consiste en el diseño de operadores genéticos mejor adaptados a este problema.

Pawel

Bibliografía

[1] Enrique Alba. *Parallel Metaheuristics. A New Class of Algorithms*. A John Wiley & Sons, INC., Publication, 2005.

[2] Suk-Tae Bae et al. "Integrated GA-VRP solver for multi-depot system". In: *Computers & Industrial Engineering* 53.2 (2007). Selected Papers from The 27th. International Conference on Computers & Industrial Engineering - Part 2, pp. 233 –240. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2007.06.014. URL: http://www.sciencedirect.com/science/article/pii/S0360835207001040.

[3] Raul Baños et al. "Analysis of OpenMP and MPI implementations of meta-heuristics for vehicle routing problems". In: *Applied Soft Computing* 43 (2016), pp. 262–275.

[4] Pawel Blazej, Malgorzata Wnetrzak, and Pawel Mackiewicz. "The role of crossover operator in evolutionary-based approach to the problem of genetic code optimization". In: *Biosystems* 150 (2016), pp. 61 –72. ISSN: 0303-2647. DOI: https://doi.org/10.1016/j.biosystems.2016.08.008. URL: http://www.sciencedirect.com/science/article/pii/S0303264716301848.

[5] T. Blickle and L Thiele. "A comparison of selection schemes used in evolutionary algorithms". In: *Evol. Comput.* 4.4 (1996), pp. 361–394.

[6] Miroslab Blocho and Zbigniew J. Czech. "A Parallel Memetic Algorithm for the Vehicle Routing Problem with Time Windows". In: *3PGCIC 2013, 8th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing* (2013).

[7] Christian Blum and Andrea Roli. "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison". In: *ACM Comput. Surv.* 35.3 (Sept. 2003), pp. 268–308. ISSN: 0360-0300. DOI: 10.1145/937503.937505. URL: http://doi.acm.org/10.1145/937503.937505.

[8] Y. Borenstein and A. Moraglio. *Theory and Principled Methods for the Design of Metaheuristics*. Springer, 2014.

[9] Erbao Cao, Ruotian Gao, and Mingyong Lai. "Research on the vehicle routing problem with interval demands". In: *Applied Mathematical Modeling* 54 (2018), pp. 332–346.

[10] C.A.C. Coello, G.B. Lamont, and D.A.V. Veldhuizend. *Evolutionary Algorithms for Solving Multi-objective Problems*. springer, 2006.

[11] William Cook and Jennifer L. Ritch. "A parallel cutting plane algorithm for the vehicle routing problem with time windows". In: *Computational and Applied Mathematics* (1999).

[12] Jean-François Cordeau and Mirko Maischberger. "A parallel iterated tabu search heuristic for vehicle routing problems". In: *Computers & Operations Research* 39.9 (2012), pp. 2033 –2050. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2011.09.021. URL: http://www.sciencedirect.com/science/article/pii/S0305054811002826.

[13] C. Cotta and J.I. van Hemert. *Recent advances in evolutionary computation for combinatorial optimization*. Vol. 153. springer, 2008.

[14] Zbigniew J. Czech. *Best solutions found by the parallel simulated annealing algorithm for Solomon's vehicle routing problem with time windows (VRPTW) benchmark instances*. 2014. URL: http://sun.aei.polsl.pl/~zjc/best-solutions-solomon.html.

[15] Martin Desrochers, Jacques Desrosiers, and Marius Solomon. "A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows". In: *Oper. Research* 40.2 (1992), pp. 342–354.

[16] Jacques Desrosiers, Oli B. G. Madsen, and Francois Soumis. "2-Path Cuts for the Vehicle Routing Problem with Time Windows". In: *Transportation Science* 1 (12 1999), pp. 147–167.

[17] Christophe Duhamel et al. "A multi-start evolutionary local search for the two-dimensional loading capacitated vehicle routing problem". In: *Computers & Operations Research* 38.3 (2011), pp. 617 –640. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2010.08.017. URL: http://www.sciencedirect.com/science/article/pii/S0305054810001875.

[18] Agoston E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. SpringerVerlag, 2003. ISBN: 3540401849.

[19] Cao Erbao, Lai Mingyong, and Nie Kai. "A Differential Evolution & Genetic Algorithm for Vehicle Routing Problem with Simultaneous Delivery and Pick-up and Time Windows". In: *IFAC Proceedings Volumes* 41.2 (2008). 17th IFAC World Congress, pp. 10576 –10581. ISSN: 1474-6670. DOI: https://doi.org/10.3182/20080706-5-KR-1001.01791. URL: http://www.sciencedirect.com/science/article/pii/S1474667016406634.

[20] A. Garcia-Najera and JA Bullinaria. "An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows". In: *Computers & Operations Research* 35.1 (2011), pp. 287–300.

[21] Abel García-Nájera and Antonio López-Jaimes. "An investigation into many-objective optimization on combinatorial problems: Analyzing the pickup and delivery problem". In: *Swarm and Evolutionary Computation* 38 (2018), pp. 218 –230. ISSN: 2210-6502. DOI: https://doi.org/10.1016/j.swevo.2017.

08.001. URL: http://www.sciencedirect.com/science/article/pii/S2210650216303583.

[22] Fred Glover and Manuel Laguna. *Tabu Search*. Norwell, MA, USA: Kluwer Academic Publishers, 1997. ISBN: 079239965X.

[23] John J. Grefenstette et al. "Genetic Algorithms for the Traveling Salesman Problem". In: *Proceedings of the 1st International Conference on Genetic Algorithms*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1985, pp. 160–168. ISBN: 0-8058-0426-9. URL: http://dl.acm.org/citation.cfm?id=645511.657094.

[24] Joaquim Gromicho, Sameh Haneyah, and Leendert Kok. "Solving a Real-Life VRP with Inter-Route and Intra-Route Callenges". In: *SSRN* (2015).

[25] A. Gutierrez et al. "A multi population memetic algorithm for the vehicle routing problem with time windows and stochastic travel and service times". In: *IFAC-PapersOnLine* 49.12 (2016). 8th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2016, pp. 1204 –1209. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2016.07.673. URL: http://www.sciencedirect.com/science/article/pii/S2405896316309491.

[26] M.A. Hannan et al. "Capacitated vehicle-routing problem model for scheduled solid waste collection and route optimization using PSO algorithm". In: *Waste Management* 71 (2018), pp. 31 –41. ISSN: 0956-053X. DOI: https://doi.org/10.1016/j.wasman.2017.10.019. URL: http://www.sciencedirect.com/science/article/pii/S0956053X17307675.

[27] Georges R. Harik. "Finding Multimodal Solutions Using Restricted Tournament Selection". In: *Proceedings of the 6th International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, pp. 24–31. ISBN: 1-55860-370-0. URL: http://dl.acm.org/citation.cfm?id=645514.658086.

[28] Angel A. Juan et al. "A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems". In: *Operations Research Perspectives* 2 (2015), pp. 62–72.

[29] Brian Kallehauge, Jesper Larsen, and Oli B.G. Madsen. "Lagrangean duality and non-differentiable optimization applied on routing with time windows - experimental results". In: *Internal report IMM-REP-2000-8, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark*. 2000.

[30] A. Khachaturyan, S. Semenovskaya, and B. Vainshtein. "Statistical - Thermodynamic Approach to Determination of Structure Amplitude Phases". In: *Sov. Phys. Crystallography* 24 (1979), pp. 519–524.

[31] Fernando G. Lobo, Cludio F. Lima, and Zbigniew Michalewicz. *Parameter Setting in Evolutionary Algorithms*. 1st. Springer Publishing Company, Incorporated, 2007. ISBN: 3540694315, 9783540694311.

[32] Manuel Lozano, Francisco Herrera, and José Ramón Cano. "Replacement strategies to preserve useful diversity in steady-state genetic algorithms". In: *Information Sciences* 178.23 (2008). Including Special Section: Genetic and Evolutionary Computing, pp. 4421 –4433. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2008.07.031. URL: http://www.sciencedirect.com/science/article/pii/S0020025508002867.

[33] Samir W. Mahfoud. "Niching Methods for Genetic Algorithms". UMI Order No. GAX95-43663. PhD thesis. Champaign, IL, USA, 1995.

[34] K. L. Mak and Z. G. Guo. "A genetic algorithm for vehicle routing problems with stochastic demand and soft time windows". In: *Proceedings of the 2004 IEEE Systems and Information Engineering Design Symposium, 2004.* Apr. 2004, pp. 183–190. DOI: 10.1109/SIEDS.2004.239880.

[35] R. Martí, M. Laguna, and V. Campos. *Scatter Search vs. Genetic Algorithms: An Experimental Evaluation with Permutation Problems*. Kluwer Academic Publishers, 2005, pp. 263–282.

[36] Ole J. Mengshoel, Severino F. Galán, and Antonio de Dios. "Adaptive generalized crowding for genetic algorithms". In: *Information Sciences* 258.Supplement C (2014), pp. 140 –159. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2013.08.056. URL: http://www.sciencedirect.com/science/article/pii/S0020025513006300.

[37] Lai Mingyong and Cao Erbao. "An improved differential evolution algorithm for vehicle routing problem with simultaneous pickups and deliveries and time windows". In: *Engineering Applications of Artificial Intelligence* 23.2 (2010), pp. 188 –195. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2009.09.001. URL: http://www.sciencedirect.com/science/article/pii/S0952197609001213.

[38] Habibeh Nazif and Lai Soon Lee. "Optimised crossover genetic algorithm for capacitated vehicle routing problem". In: *Applied Mathematical Modelling* 36 (2012), pp. 2110–2117.

[39] Sandra Ulrich Ngueveu, Christian Prins, and Roberto Wolfler Calvo. "An effective memetic algorithm for the cumulative capacitated vehicle routing problem". In: *Computers & Operations Research* 37.11 (2010). Metaheuristics for Logistics and Vehicle Routing, pp. 1877 –1885. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2009.06.014. URL: http://www.sciencedirect.com/science/article/pii/S0305054809001725.

[40] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. 2. ed. Springer series in operations research and financial engineering. New York, NY: Springer, 2006.

[41] H.M Pandey, A. Chaudhary, and D Mehrotra. "A comparative review of approaches to prevent premature convergence in GA". In: *Appl Soft. Comput.* 24 (2014), pp. 1047–1077.

[42] Jean-Yves Potvin and Samy Bengio. "The Vehicle Routing Problem with Time Windows - Part II: Genetic Search". In: *Informs J Comput* 8(2) (1996), pp. 165–172.

[43] Yutao Qi et al. "A decomposition based memetic algorithm for multi-objective vehicle routing problem with time windows". In: *Computers & Operations Research* 62 (2015), pp. 61–77.

[44] Quintiq. *VRPTW World Records*. 2017. URL: http://www.quintiq.com/optimization/vrptw-world-records.html.

[45] Roberto Roberti. "Exact Algorithms for Different Classes of Vehicle Routing Problems". PhD thesis. University of Bologna, 2012.

[46] Simon Ronald. "Distance Functions for Order-Based Encodings". In: *Program and Data Optimization Group* 1 (1997).

[47] Simon Ronald. "More Distance Functions for Order-Based Encodings". In: *Program and Data Optimization Group* 1 (1998), pp. 558–563.

[48] Oliver Schuetze et al. *NEO 2015 - Numerical and Evolutionary Optimization*. Studies in Computational Intelligence. Tijuana, Mexico: Springer, 2016. URL: https://hal.inria.fr/hal-01389071.

[49] Carlos Segura et al. "A Novel Diversity-based Evolutionary Algorithm for the Traveling Salesman Problem". In: *GECCO'15* (2015), pp. 489–496.

[50] Carlos Segura et al. "A Novel Diversity-Based Replacement Strategy for Evolutionary Algorithms". In: *IEEE Trans. on Cybernetics* 46.12 (2016), pp. 3233–3246.

[51] Carlos Segura et al. "A Novel Diversity-Based Replacement Strategy for Evolutionary Algorithms". In: *IEEE Transactions on Cybernetics* 46.12 (2016), pp. 3233–3246.

[52] Carlos Segura et al. "Improving Diversity in Evolutionary Algorithms: New Best Solutions for Frequency Assignment". In: *IEEE Transactions on Evolutionary Computation* 21.4 (2017), pp. 539–553. ISSN: 1089-778X. DOI: 10.1109/TEVC.2016.2641477.

[53] Carlos Segura et al. "The importance of Diversity in the Application of Evolutionary Algorithms to the Sudoku Problem". In: *2016 IEEE Congress on Evolutionary computation (CEC)* (2016), pp. 919–926.

[54] Carlos Segura et al. "Using multi-objective evolutionary algorithms for single-objective constrained and unconstrained optimization". In: *Annals of Operations Research* 240.1 (2016), pp. 217–250. ISSN: 1572-9338.

[55] Martin Dario Arango Serna et al. "Vehicle Routing to Multiple Warehouses Using a Memetic Algorithm". In: *Procedia - Social and Behavioral Sciences* 160 (2014). XI Congreso de Ingenieria del Transporte (CIT 2014), pp. 587 –596. ISSN: 1877-0428. DOI: https://doi.org/10.1016/j.sbspro.2014.12.172. URL: http://www.sciencedirect.com/science/article/pii/S1877042814062739.

[56] Marius M. Solomon. "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints". In: *Operations Research* 35 (Apr. 1987), pp. 254–265.

[57] Marius M. Solomon. *Benchmarks Problem and Solution*. 2005. URL: http://web.cba.neu.edu/~msolomon/problems.htm.

[58] William M. Spears. "The Role of Mutation and Recombination in Evolutionary Algorithms". AAI9829073. PhD thesis. Fairfax, VA, USA, 1998. ISBN: 0-591-81732-2.

[59] R Storn and K Price. "Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces". In: *J. Glob. Optim.* 11.4 (1997), pp. 341–359.

[60] E.G. Talbi. *Metaheuristics: From Design to Implementation*. Wiley Series on Parallel and Distributed Computing. Wiley, 2009. ISBN: 9780470496909. URL: https://books.google.com.mx/books?id=SIsa6zi5XV8C.

[61] Matej Črepinšek, Shih-Hsi Liu, and Marjan Mernik. "Exploration and Exploitation in Evolutionary Algorithms: A Survey". In: *ACM Comput. Surv.* 45.3 (July 2013), 35:1–35:33. ISSN: 0360-0300. DOI: 10.1145/2480741.2480752. URL: http://doi.acm.org/10.1145/2480741.2480752.

[62] Matej Črepinšek, Shih-Hsi Liu, and Marjan Mernik. "Exploration and Exploitation in Evolutionary Algorithms: A Survey". In: *ACM Comput. Surv.* 45.3 (July 2013), 35:1–35:33. ISSN: 0360-0300. DOI: 10.1145/2480741.2480752. URL: http://doi.acm.org/10.1145/2480741.2480752.

[63] VeRoLog and ORTEC. *VeRoLog solver challenger*. 2017. URL: https://verolog.ortec.com/.

[64] Thibaut Vidal et al. "A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows". In: *Computers & Operations Research* 40.1 (2013), pp. 475 –489. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2012.07.018. URL: http://www.sciencedirect.com/science/article/pii/S0305054812001645.

[65] Lijun Wei et al. "A simulated annealing algorithm for the capacitated vehicle routing problem with two-dimensional loading constraints". In: *European Journal of Operational Research* 265.3 (2018), pp. 843 –859. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2017.08.035. URL: http://www.sciencedirect.com/science/article/pii/S037722171730766X.

[66] Wikipedia. *Stars and bars (combinatorics)*. 2017. URL: https://en.wikipedia.org/wiki/Stars_and_bars_(combinatorics).

[67] David A. Wood. "Evolutionary memetic algorithms supported by metaheuristic profiling effectively applied to the optimization of discrete routing problems". In: *Journal of Natural Gas Science and Engineering* 35 (2016), pp. 997 – 1014. ISSN: 1875-5100. DOI: https://doi.org/10.1016/j.jngse.2016.09.031. URL: http://www.sciencedirect.com/science/article/pii/S1875510016306710.

[68] Nagata Y. "Edge assembly crossover for the capacitated vehicle routing problem." In: *Proceedings of the 7th European Conference on evolutionary computation in combinatorial optimization* (2007).

[69] Nagata Y and Braysy O. "A powerful route minimization heuristic for the vehicle routing problem with time windows." In: *Oper. Res. Lett* 37.5 (2009).

[70] Nagata Y., Braysy O, and Dullaert W. "A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows." In: *Comp. & OR* 37 (2010).

[71] Nagata Y and Kobayashi S. "Edge assembly crossover: a high-power genetic algorithm for the travelling salesman problem". In: *Proceedings of the 7th international conference on genetic* (1997).

[72] Yuzhou Zhang et al. "Memetic algorithm with route decomposing for periodic capacitated arc routing problem". In: *Applied Soft Computing* 52 (2017), pp. 1130 –1142. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2016.09.017. URL: http://www.sciencedirect.com/science/article/pii/S1568494616304768.

[73] Tong Zhen, Yuhua Zhu, and Qiuwen Zhang. "A hybrid ant colony algorithm for the capacitated vehicle routing problem". In: *2008 IEEE International Symposium on IT in Medicine and Education*. Dec. 2008, pp. 935–939. DOI: 10.1109/ITME.2008.4744004.

[74] F. E. Zulvia, R. J. Kuo, and Tung-Lai Hu. "Solving CVRP with time window, fuzzy travel time and demand via a hybrid ant colony optimization and genetic algortihm". In: *2012 IEEE Congress on Evolutionary Computation*. June 2012, pp. 1–8. DOI: 10.1109/CEC.2012.6252922.