UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Problema de localización y distribución con criterios no convergentes

Informe de Proyecto de Grado presentado por

Ian Rolls, Sebastián Estevez

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisores

Pedro Piñeyro
Sandro Moscatelli

Montevideo, 27 de septiembre de 2023

Resumen

El presente trabajo aborda el Problema de Localización y Ruteo (LRP por sus siglas en inglés) con criterios no convergentes. Se realizó un relevamiento de la bibliografía existente sobre este problema y sus variantes con el fin de conocer los avances y el estado de las investigaciones sobre el tema y establecer una base sólida para la realización del trabajo.

Como resultado de la realización del estado del arte, se decidió extender un trabajo sobre LRP aplicado a la logística de salud en Países Bajos, con el objetivo de desarrollar un enfoque multiobjetivo que considere criterios no convergentes (costo, eficiencia y equidad) y abordar algunas características particulares que presenta el planteo.

De forma general, el problema consiste en determinar que lockers abrir de un conjunto potencial de lockers, determinar que pacientes asignar a los lockers y determinar las mejoras rutas desde un depósito central hacia estos lockers y pacientes. Como parte de la extensión, se consideran tres tipos de pacientes: bajo riesgo (deben ser asignados a un locker), alto riesgo (deben ser atendidos en su casa) e indiferentes (cualquiera de las dos opciones anteriores). Se ha clasificado este problema como LRP MultiObjetivo (MOLRP por sus siglas en inglés) porque contempla múltiples objetivos sobre un problema LRP.

Se desarrolló un modelo matemático para el MOLRP y se validó utilizando el software de programación lineal GLPK. Luego, se implementó un algoritmo de búsqueda de vecindad variable multiobjetivo (MO-VNS, por sus siglas en inglés) para resolver el problema. Además, se creó una aplicación web para facilitar el ingreso de datos y la visualización de soluciones.

Se realizó una experimentación numérica comparando el desempeño del algoritmo con el software de optimización CPLEX. Además, se analizaron distintas variantes del algoritmo. Se utilizaron distintas métricas de comparación multiobjetivo para evaluar las variantes de los algoritmos, entre ellas hipervolumen, R2, ϵ, C y promedio de cantidad de soluciones. Los resultados obtenidos demuestran la eficacia del procedimiento de resolución propuesto para la extensión del problema LRP abordado.

Palabras clave: Location Routing Problem, Variable Neighborhood Search, Multiobjective, Optimization, Metaheuristics

Agradecimientos

Queremos agradecer a todas aquellas personas que nos brindaron su apoyo y colaboración durante la realización de esta tesis de grado.

En primer lugar, queremos agradecer a Pedro Piñeyro y a Sandro Moscatelli por su compromiso, ayuda, y paciencia. Su experiencia y conocimiento fueron fundamentales para el éxito de este proyecto.

También queremos agradecer a nuestros amigos y compañeros que nos apoyaron en los momentos difíciles y nos animaron a seguir adelante.

También mencionar a nuestras familias, quienes siempre estuvieron a nuestro lado brindándonos su apoyo. Su comprensión y paciencia nos permitieron dedicar tiempo y energía a este proyecto.

Por último, pero no menos importante, queremos agradecer a todas las personas que de una u otra manera contribuyeron a nuestra formación académica y personal. Sin su apoyo, este logro no habría sido posible.

Gracias a todos por su valiosa contribución.

Índice general

Capítulo 1

Introducción

El presente documento describe el trabajo realizado en el marco de cumplimiento del Proyecto de Grado de Ingeniería en Computación. En el mismo se aborda el Problema de Localización y Ruteo (LRP, por sus siglas en inglés), que es un problema de optimización que combina los problemas de ubicaciones de instalaciones y la planificación de rutas de transporte (Nagy & Salhi, 2007). El LRP ha sido ampliamente investigado debido a su relevancia en la logística y la distribución, siendo de especial interés para la eficiencia operacional en la cadena de suministro (Melo et al., 2009). Este trabajo tiene como base el trabajo realizado por Veenstra et al. (2018a) y busca extender dicho estudio, transformando el problema de un único objetivo a tres objetivos no convergentes: costo, eficacia y equidad. Se entiende por objetivos no convergentes aquellos que no pueden ser optimizados simultáneamente sin sacrificar uno de ellos, y en estos casos, se debe buscar un equilibrio entre los objetivos. Además de extender el problema a uno multiobjetivo, se agrega límite en la capacidad de los vehículos, distintos tipos de pacientes y se unifican tipos de rutas.

La incorporación de estos tres objetivos no convergentes es, hasta donde sabemos, una contribución novedosa en el ámbito del LRP y responde a la creciente demanda por soluciones que equilibren aspectos económicos, operativos y sociales (Zitzler et al., 2003). Para resolver este problema multiobjetivo, se implementó un procedimiento de resolución basado en la metaheurística de Búsqueda de Vecindad Variable (VNS, por sus siglas en inglés) multiobjetivo, siguiendo el enfoque propuesto en Duarte et al. (2015). Esta técnica es reconocida por su capacidad para explorar de manera eficiente el espacio de soluciones, generando resultados que equilibren de manera adecuada los objetivos propuestos.

Con el fin de facilitar el manejo de los datos y la visualización de los resultados, se desarrolló un software especializado que permite ingresar la información del problema y analizar las soluciones en un entorno amigable e intuitivo. Dicho software fue de gran utilidad para llevar a cabo experimentaciones y evaluar el desempeño de las distintas variantes del algoritmo propuesto.

Para llevar a cabo una comparación rigurosa entre las diferentes soluciones

generadas por el algoritmo, se emplearon métricas de optimización multiobjetivo como el hipervolumen (Zitzler et al., 2003), ϵ (Laumanns et al., 2002), $R2$ (Hansen & Jaszkiewicz, 1994) y C (Deb et al., 2002). Estas métricas permiten cuantificar el desempeño del algoritmo en términos de convergencia y diversidad, proporcionando información valiosa para la toma de decisiones en situaciones prácticas.

Se realizó un análisis de los resultados obtenidos por la metaheurística multiobjetivo VNS aplicada en diferentes escenarios, variando tanto la topología como el tamaño de los mismos. Además, se realizan variantes del algoritmo que también son evaluadas para encontrar la mejor versión del mismo.

El resto de este documento se encuentra organizado de la siguiente manera: en el Capítulo 2, se brinda un marco teórico acerca de los problemas multiobjetivos, y de las principales técnicas de resolución existentes, tales como Algoritmos Evolutivos, VNS, VNS multiobjetivo y métricas de comparación multiobjetivo. En el Capítulo 3, se realiza un resumen del estado del arte del LRP. En el Capítulo 4, se presenta el problema base de este trabajo, describiendo su modelo matemático. Además, se describe el problema de este trabajo, brindando el modelo matemático, su validación y los test de esfuerzo realizados. En el Capítulo 5, se explica la solución propuesta por los autores del problema base. Posteriormente, se describe la solución propuesta para este trabajo, el diseño del algoritmo y se describen las distintas variantes implementadas y evaluadas. En el Capítulo 6, se describe la implementación del algoritmo propuesto en el Capítulo 5. Se describe la arquitectura de la solución, y se describe detalladamente cada uno de los componentes (Cliente Web, Backend y base de datos). Además, se explican las distintas funcionalidades de la solución. En el Capítulo 7 se describe la experimentación numérica llevada a cabo. Finalmente, en el Capítulo 8 se presentan las conclusiones y posibles líneas de trabajo a futuro.

Capítulo 2

Marco teórico

Esta sección introduce los conceptos más relevantes y que creemos necesarios para una mejor comprensión de la solución implementada. Comienza definiendo lo que es un problema de optimización multiobjetivo y la frontera de Pareto, para luego describir ciertas técnicas de resolución multiobjetivo, empezando por Variable Neighbourhood Search (VNS), fundamental para el entendimiento del algoritmo VNS multiobjetivo (MO-VNS por sus siglas en inglés) utilizado en la resolución del problema presentado en este trabajo. Por último, se describen ciertas métricas de comparación multiobjetivo.

2.1. Optimización multiobjetivo

En Gandibleux et al. (2012) se da una definición técnica de problema de optimización. En ese trabajo se define problema de optimización como un mapeo $f : R^n \to R^k$ donde R^n es el espacio de decisión o espacio de parámetros o de problema, y R^k es el espacio objetivo. Si $k = 1$ entonces el problema es mono-objetivo y si $k > 1$ es un problema multiobjetivo (MOP). Una solución a un problema de optimización consiste de un vector $(x_1, x_2, ..., x_n) \in X$ donde además se tiene una función $f : X \to Y$ que asigna un valor a una solución, el cual será también un vector $(y_1, y_2, ..., y_n) \in Y$.Un problema de optimización multiobjetivo se puede describir entonces de la siguiente forma:

$$\begin{aligned} \min_x/\max \quad & f(x) = (f_1(x), f_2(x), ..., f_k(x)) \\ \text{sujeto a:} \quad & x = (x_1, x_2, ..., x_n) \in X \\ & y = (y_1, y_2, ..., y_k) \in Y \end{aligned}$$

Donde x es un vector de decisión, X es el espacio de decisión, y es el vector objetivo, Y es el espacio objetivo y $f : X \to Y$ consiste de k funciones objetivos de valores reales.
Si $k = 1$ y si el problema consiste en minimizar ese objetivo. Entonces una solución $x^1 \in X$ es mejor que otra solución x^2 sí $f(x^2) > f(x^1)$. Pueden existir

varias soluciones óptimas en el espacio de decisión, pero solo un óptimo en el espacio objetivo.
Ahora, si $k > 1$ entonces comparar dos soluciones es más complejo. En estos casos se utiliza el concepto de dominancia de Pareto.
Sean $u = (u_1, u_2, ..., u_k), v = (v_1, v_2, ..., v_k) \in Y$ se dice que u domina a v sí $u_i \leq v_i, \forall i = 1, ..., k$ y $u \neq v$. Se dice que $x^* \in X$ es un óptimo de Pareto si no existe otro $x \in X$ tal que $f(x)$ domina a $f(x^*)$. El conjunto de puntos óptimos de Pareto se llama conjunto de Pareto (PS), mientras que el conjunto de todos los vectores objetivo de Pareto se conoce como frontera de Pareto, $PF = \{f(x) \in Y, x \in PS\}$. Esta frontera de Pareto es útil para los tomadores de decisiones, ya que les permite tomar la mejor solución de compromiso entre los objetivos.
En un problema multiobjetivo se trata de encontrar el conjunto de puntos no dominados en el espacio objetivo, el cual será una aproximación a la frontera de Pareto, buscando además que esta aproximación cumpla con tres objetivos:

- Minimizar la distancia de los puntos no dominados encontrados a la frontera de Pareto real.
- Buscar una buena distribución de la solución obtenida.
- Maximizar la extensión de las soluciones no dominadas obtenidas para cada objetivo.

2.2. Técnicas de resolución mono-objetivo

A continuación se enumeran algunas de las técnicas de resolución para problemas de optimización mono-objetivos.

- Técnicas exactas:
  - Programación lineal: El uso de la programación lineal para resolver problemas de optimización ha sido ampliamente estudiado. Un enfoque clásico es el método Simplex, introducido por Dantzig en 1947 (G. B. Dantzig, 1947).
  - Programación Entera: Para problemas donde las variables de decisión son discretas, se utilizan técnicas de programación entera. El algoritmo de ramificación y acotación (Branch and Bound) es una técnica comúnmente empleada para resolver problemas de programación entera (Crowder et al., 1983).
- Técnicas heurísticas:
  - Búsqueda Local: La búsqueda local es una técnica que explora el espacio de soluciones vecinas buscando una solución óptima local.

- Algoritmos Constructivos: Estos algoritmos construyen gradualmente una solución factible, mejorando iterativamente una solución parcial. Un ejemplo es el algoritmo de inserción y reemplazo utilizado en el problema del viajero de comercio (TSP, por sus siglas en inglés) (Lin & Kernighan, 1973).

- Algoritmos Greedy: Los algoritmos greedy toman decisiones en cada paso basándose en la elección óptima local en ese momento. Un ejemplo es el algoritmo de Kruskal para resolver el problema del árbol de expansión mínima (Kruskal, 1956).

- Técnicas metaheurísticas:

  - Algoritmos Genéticos: Los algoritmos genéticos son inspirados en la evolución biológica y utilizan operadores como mutación y cruzamiento para generar nuevas soluciones y explorar el espacio de búsqueda. Holland introdujo esta técnica en 1975 (Holland, 1975).

  - Simmulated Annealing: Basado en el proceso de enfriamiento de un material fundido, el recocido simulado busca soluciones aceptables a través de la exploración del espacio de búsqueda con una probabilidad de aceptación decreciente. Kirkpatrick et al. presentaron este enfoque en 1983 (Kirkpatrick et al., 1983).

  - Tabú Search: La búsqueda Tabú utiliza una lista de movimientos prohibidos para escapar de óptimos locales y explorar el espacio de soluciones. Glover propuso este método en 1986 (Glover, 1986).

  - VNS (Variable Neighbourhood Search):Variable Neighborhood Search (VNS) es una técnica de optimización metaheurística utilizada para resolver problemas de optimización combinatoria. Fue propuesta por Mladenović y Hansen (1997). La idea detrás de VNS es explorar soluciones en diferentes vecindarios y utilizar movimientos específicos para mejorar las soluciones actuales. El método comienza con una solución inicial y luego realiza una búsqueda local en un vecindario específico. Si la búsqueda local no mejora la solución, se cambia a otro vecindario y se realiza otra búsqueda local. Este proceso continúa hasta que se encuentre una solución que satisfaga ciertos criterios de parada o se alcance un número máximo de iteraciones. La clave de la eficacia de VNS radica en la exploración de múltiples vecindarios, lo que permite escapar de mínimos locales y encontrar soluciones óptimas o cercanas a ellas. Los vecindarios se definen mediante movimientos que modifican la solución actual de manera específica y controlada. Dado que este trabajo se basa en una versión multiobjetivo de VNS, esta metaheurística se explica en la Sección 2.2.1 de forma de brindar una introducción a la versión multiobjetivo.

2.2.1. Variable Neighbourhood Search

VNS es una metaheurística propuesta en Mladenović y Hansen (1997). Utiliza el concepto de vecindad $N(x)$ con $x \in X$ un conjunto dado, la cual se define normalmente por una métrica dada δ en X de forma que $N(x) = \{y \in X | \delta(x, y) \leq \alpha\}$ con α un número entero positivo. Normalmente, una vecindad se construye a partir de un operador aplicado a una solución dada, por ejemplo para el TSP es normal encontrar operadores que consisten en movimientos, eliminación o inserción de aristas en una solución dada.
VNS se basa en cambiar las estructuras de vecindades para realizar la búsqueda de una solución óptima basada en las siguientes observaciones:

- Un óptimo local relativo a una estructura de vecindad no es necesariamente un óptimo local para otras estructuras de vecindades.
- Un óptimo global es un óptimo local respecto a todas las estructuras de vecindades.
- Datos empíricos muestran que para muchos problemas los óptimos locales están relativamente cerca uno del otro.

La primera propiedad se explota usando movimientos complejos de forma incremental para encontrar óptimos locales con respecto a las estructuras de vecindades usadas. La segunda propiedad sugiere el uso de varias vecindades. Mientras que la tercera propiedad sugiere explotar la vecindad de la solución actual para intentar mejorar esa solución. De esta forma, en el VNS se definen tres procedimientos:

1. Shaking procedure (procedimiento de agitación): Utilizado para tratar de evitar mínimos locales. Dada la solución actual x, y una vecindad seleccionada por el Algoritmo 2 Cambio de vecindad secuencial, este procedimiento selecciona una nueva solución candidata en N. Una forma simple sería elegir de forma aleatoria una nueva solución en esa vecindad, como se muestra en el Algoritmo 3 Shaking procedure.

2. Improvement procedure (procedimiento de mejora): Intenta mejorar una solución dada. Existen dos tipos de procedimientos de mejora: Local Search (búsqueda local) y el Algoritmo 1 Secuencial VND best improvement. La búsqueda local explora una vecindad dada a partir de una solución candidata en búsqueda de una mejor solución. Existen dos estrategias: first improvement (primer mejora) y best improvement (mejor mejora). En la de "primer mejora.[el] procedimiento se detiene cuando encuentra la primer solución que es mejor a la actual, mientras que en la "mejor mejora", el procedimiento recorre toda la vecindad y retorna la que es mejor de toda la vecindad y la solución actual. Cualquier metaheurística puede funcionar como búsqueda local, desde algoritmos genéticos, GRASP, Tabú Search, y otros. Mientras que los procedimientos descendentes exploran más de una vecindad, ya sea de una manera secuencial o anidada, pueden usar la estrategia de primer mejora o mejor mejora. Estos explotan el hecho de que

un óptimo local con respecto a varias vecindades tiene más probabilidades de ser óptimo global que aquel óptimo local de una sola vecindad.

3. Neighborhood change step (paso de cambio de vecindad): El propósito de este procedimiento es el de guiar al VNS mientras se explora el espacio de soluciones. Toma la decisión de qué vecindad elegir para explorar, así como qué solución se aceptará como solución candidata. Hay distintos procedimientos para este paso: secuencial, cíclico, tubería. Por ejemplo, el secuencial se muestra en el Algoritmo 2 Cambio de vecindad secuencial, que cambia a la siguiente vecindad si no se mejora la solución y en caso de mejorarse se vuelve a la vecindad inicial. Es decir, al mejorar la solución, vuelve a repetir todo el procedimiento desde la vecindad inicial. La implementación cíclica establece un ciclo predefinido de vecindades y en cada iteración se selecciona la vecindad según el ciclo establecido. Esto permite explorar de manera equitativa todas las vecindades. Otra posible implementación es de forma de tubería o pipeline, que implica la utilización de varias vecindades de manera simultánea o en paralelo. Se generan soluciones candidatas en cada una de las vecindades y se comparan los resultados.

Algoritmo 1 Secuencial VND best improvement

procedure B-VND(x, l_{max}, N)
  repeat
    $stop \leftarrow false$
    $l \leftarrow 1$
    $x' \leftarrow x$
    repeat
      $x'' \leftarrow argmin_{y \in N_l(x)} f(y)$
      $NeighboorhoodChangeSequential(x, x'', l)$
    until $l = l_{max}$
    if $f(x') \leq f(x)$ **then**
      $stop \leftarrow true$
    end if
  until $stop = true$
  return x'
end procedure

Se definen entonces un conjunto de vecindades en un orden dado $N = \{N_1, N_2, ..., N_{k_{max}}\}$ donde N_k son operadores tal que $N_k(x)$ es una estructura de vecindad para x. Estas vecindades son procesadas por el Shaking procedure y el Neighborhood Change Step.
Para el Improvement Phase se utiliza otro conjunto de vecindades
$N' = \{N'_1, N'_2, ..., N'_{k'_{max}}\}$ donde N'_k son operadores tal que $N'_k(x)$ es una estructura de vecindad para x. A partir de una solución y vecindad N_k, el Shaking selecciona de alguna forma una nueva solución, la cual se pasa al procedimien-

to Improvement el cual, utilizando su conjunto de vecindades, intenta mejorar dicha solución. Luego, el Neighborhood Change Step cambia de alguna forma la vecindad a explorar por el Shake y decide si aceptar o no la nueva solución. Esto se repite hasta cumplir un criterio de parada.

Algoritmo 2 Cambio de vecindad secuencial

procedure NEIGHBOORHOODCHANGESEQUENTIAL(x, x', k)
  if $f(x') < f(x)$ **then**
    $x \leftarrow x'$
    $k \leftarrow 1$
  else
    $k \leftarrow k + 1$
  end if
end procedure

Algoritmo 3 Shaking procedure

procedure SHAKE($x, k, \mathcal{N}$)
  choose $x' \in \mathcal{N}_k(x)$ at random
  return x'
end procedure

Variantes VNS

A continuación se describen algunas de las variantes VNS mono-objetivo: Básica, Reducida y General.

- Básica: Esta variante utiliza un procedimiento Shaking, un Local Search y un Neighborhood Change Step, Algoritmo 4 Basic VNS.

- Reducida: No utiliza procedimiento de mejora, solo tiene procedimiento de Shaking y Neighborhood Change Step, Algoritmo 5 Reduced VNS.

- General: Utiliza como procedimiento de mejora un procedimiento descendente, además de utilizar un procedimiento de Shaking y Neighborhood Change Step, Algoritmo 6 General VNS.

Algoritmo 4 Basic VNS

procedure BASICVNS($x, k_{max}, \mathcal{N}, N$)
  repeat
    $k \leftarrow 1$
    while $k \leq k_{max}$ **do**
      $x' \leftarrow Shake(x, k, \mathcal{N})$
      $x'' \leftarrow LocalSearch(x', N)$
      $NeighboorhoodChangeSequential(x, x'', k)$
    end while
  until condicion de parada
  return x
end procedure

Algoritmo 5 Reduced VNS

procedure REDUCEDVNS($x, k_{max}, \mathcal{N}$)
  repeat
    $k \leftarrow 1$
    while $k \leq k_{max}$ **do**
      $x' \leftarrow Shake(x, k, \mathcal{N})$
      $NeighboorhoodChangeSequential(x, x', k)$
    end while
  until condicion de parada
  return x
end procedure

Algoritmo 6 General VNS

procedure GENERALVNS($x, k_{max}, l_{max}, \mathcal{N}, N$)
  repeat
    $k \leftarrow 1$
    while $k \leq k_{max}$ **do**
      $x' \leftarrow Shake(x, k, \mathcal{N})$
      $x'' \leftarrow VND(x', l_{max}, N)$
      $NeighboorhoodChangeSequential(x, x'', k)$
    end while
  until condicion de parada
  return x
end procedure

2.3. Técnicas de resolución multiobjetivo

Generar la frontera de Pareto es computacionalmente costoso y además no suele ser posible, ya que la complejidad del problema no permite aplicar métodos exactos para resolver este tipo de problemas. Es por esto que en la literatura se han propuesto muchos métodos metaheurísticos para resolver estos problemas, tales como Algoritmos Evolutivos, Tabú Search, Simulated Annealing, Ant Colony Optimization, entre otros.
Se han propuesto diferentes formas de aproximar los problemas de optimización multiobjetivo (MOP, por sus siglas en inglés), como son la agrupación de los objetivos en uno único, optimizar con base en un objetivo y el resto tratarlos como restricciones y otras variantes. Las metaheurísticas son otra técnica de resolución que se centran en obtener una aproximación al conjunto de Pareto.

Dada su importancia para este trabajo, a continuación se describe la variante multiobjetivo de la metaheurística VNS.

2.3.1. VNS multiobjetivo

La adaptación de VNS mono-objetivo a multiobjetivo en Duarte et al. (2015) se hace redefiniendo el concepto de solución. Una solución en este caso es el conjunto aproximado de puntos de eficiencia (aproximado al conjunto de Pareto) encontrados durante el proceso de búsqueda. Con esta nueva definición de solución se puede definir el concepto de mejora de una solución. Un movimiento mejora la solución actual cuando un nuevo punto es agregado al conjunto de Pareto o conjunto de puntos de eficiencia. El algoritmo VNS multiobjetivo presentado en el Algoritmo 7 VNS multiobjetivo general se basa en el Algoritmo 6 General VNS solo que, en vez de trabajar con una única solución, trabaja con un conjunto de soluciones.

Algoritmo 7 VNS multiobjetivo general

procedure MO-GVNS($E, k_{max}, r, k'_{max}, t_{max}$)
  repeat
    $k \leftarrow 1$
    repeat
      $E' \leftarrow MO - Shake(E, k)$
      $E'' \leftarrow MO - VND(E', k'_{max}, r)$
      $E \leftarrow MO - NeighborhoodChange(E, E'', k)$
    until $k = k_{max}$
    $t \leftarrow CPUTime()$
  until $t > t_{max}$
  return E
end procedure

En las siguientes subsecciones se explican en detalle los algoritmos MO-

Shake, MO-VND y MO-NeighborhoodChange.

2.3.2. Procedimiento MO-Shake

Se lleva a cabo implementando un procedimiento Shake mono-objetivo a cada punto de la solución multiobjetivo, como se muestra en el Algoritmo 8 Shake multiobjetivo.

Algoritmo 8 Shake multiobjetivo

procedure MO-SHAKE(E, k)
  $E' \leftarrow \varnothing$
  for each $x \in E$ **do**
    $x' \leftarrow Shake(x, k)$
    $E' \leftarrow E' \cup \{x'\}$
  end for
  return E'
end procedure

En este caso $Shake(x, k)$ es un procedimiento shake aleatorio en la vecindad k, es decir selecciona de forma aleatoria un x' dé $N_k(x)$. En el trabajo de Duarte et al. (2015) se proporcionan otros procedimientos shake con mayor énfasis en la intensificación que uno únicamente aleatorio, ya que puede no ser suficiente para explorar otras regiones del espacio de búsqueda. En este caso el shake perturba cada x de E según la vecindad, k pero no en una forma aleatoria, sino que lo hace según una de las funciones objetivo.

2.3.3. Procedimiento MO-VND

Para adaptar el VND a la versión multiobjetivo, se debe implementar el VND para mejorar cada objetivo por separado. Esto implica tener tantos VND como funciones objetivos. Cada VND intenta mejorar el valor de un objetivo sin importar el valor del resto de objetivos.
Cada VND retorna el conjunto aproximado de puntos de eficiencia encontrados. En el Algoritmo 10 VND-i se muestra un pseudocódigo de cada VND$_i$. Dado un punto x, este algoritmo explora ciertas vecindades y mejora x según el objetivo i. Es similar al VND secuencial presentado anteriormente, pero difiere en el procedimiento update del paso 6, el cual valida si x' califica para entrar a E o no, y también difiere en el paso 14, el cual devuelve el conjunto E en vez de una solución particular.
El procedimiento de mejora VND queda como se muestra en el Algoritmo 9 MO-VND:

Algoritmo 9 MO-VND

```
1:  procedure MO-VND(E, k'_max, r)
2:      S_1 ← ∅, S_2 ← ∅, ... , S_r ← ∅
3:      i ← 1
4:      repeat
5:          repeat
6:              x' ← Select(E \ S_i)
7:              E_i ← VND − i(x', k'_max)
8:              S_i ← S_i ∪ E_i
9:          until E \ S_i = ∅
10:         MO − ObjetiveChange(E, S_i, i)
11:     until i > r
12:     return E
13: end procedure
```

En el Algoritmo 9 MO-VND, E es la solución a mejorar, k'_{max} es el número máximo de vecindades considerado y r es el número de funciones objetivo. En el mismo se utilizan los conjuntos $S_i, \quad i = 1, ..., r$, los cuales llevan la cuenta de que soluciones ya han sido intensificadas para el objetivo i.
De esta forma, el algoritmo mejora cada solución de E para cada objetivo. Comienza con el primer objetivo, esto sugiere que hay que especificar un orden para los objetivos. Selecciona un punto al azar de E menos S_i, mejora ese punto según VND_i y luego agrega a S_i el conjunto de Pareto encontrado. Se marcan todos los puntos de este conjunto de Pareto como explotados para ese objetivo, ya que VND_i es un procedimiento determinista que hace converger a todos esos puntos al mismo conjunto de Pareto. Repite el procedimiento anterior hasta que todos los puntos de E hayan sido intensificados. Luego de esto (paso 10) se comprueba si hubo una mejora, esto es, si los S_i encontrados mejoran E (agregan al menos un punto de eficiencia al conjunto E). Si hubo una mejora se vuelve a probar el primer objetivo, si no, se sigue con el siguiente objetivo y se actualiza E con los puntos dé S_i. Notar que el algoritmo de la Línea 10 es análogo a el Algoritmo 11 Neighborhood Change Step multiobjetivo.

Algoritmo 10 VND-i

procedure VND-$i(x, k'_{max})$
  $k \leftarrow 1$
  $E \leftarrow \{x\}$
  repeat
    $x' \leftarrow argmin_{y \in N'_k(x)} z_i(y)$
    Update$(E, N'_k(x))$
    if $z_i(x') < z_i(x)$ **then**
      $x \leftarrow x'$
      $k \leftarrow 1$
    else
      $k \leftarrow k + 1$
    end if
  until $k = k_{max}$
  return E
end procedure

2.3.4. Procedimiento MO-NeighborhoodChange

El procedimiento MO-NeighbourhoodChange multiobjetivo es análogo al Algoritmo 2 Cambio de vecindad secuencial solo que se utiliza la nueva noción de solución. El algoritmo de cambio de vecindad multiobjetivo queda como se muestra en el Algoritmo 11 Neighborhood Change Step multiobjetivo.

Algoritmo 11 Neighborhood Change Step multiobjetivo

```
1: procedure MO-NeighborhoodChange(E, E', k)
2:     if MO − Improvement(E, E') then
3:         k ← 1
4:         E ← Update(E, E')
5:     else
6:         k ← k + 1
7:     end if
8: end procedure
```

El algoritmo de la Línea 4, genera una nueva solución a partir de E y E' insertando los puntos no dominados y eliminando los que quedan dominados.

Para definir si una solución es mejor que otra se utiliza el Algoritmo 12 Improvement multiobjetivo:

Algoritmo 12 Improvement multiobjetivo

```
procedure MO-Improvement(E, E')
    for each x ∈ E' do
        if (x ∉ E and not Dominated(x, E)) then
            return True
        end if
    end for
    return False
end procedure
```

Este algoritmo (Algoritmo 12 Improvement multiobjetivo) devuelve verdadero si hay al menos un punto en E' que no es dominado por ningún punto de E, y falso si todos los puntos de E' son dominados por puntos dé E.

2.4. Métricas de comparación multiobjetivo

En un problema multiobjetivo no existe una solución óptima, sino que existen varias soluciones al problema que representan las distintas negociaciones que pueda haber entre los objetivos. Es por esto, que comparar los conjuntos soluciones de dos problemas multiobjetivos es complejo. En el trabajo de Cheng et al. (2012) se analizan distintas métricas para comparar estas soluciones. En el mismo se presentan tres tipos de métricas: basadas en conjuntos, basadas en puntos de referencia, y las basadas en el conjunto de Pareto exacto o en la frontera de Pareto exacta.

2.4.1. Métricas basadas en conjuntos

Este tipo de métricas comparan dos conjuntos o dan una calificación a estos conjuntos para saber que tan buenos son con respecto a otros conjuntos (Cheng et al., 2012).

Relaciones de rendimiento superior: Dados dos conjuntos de vectores objetivos no dominados A y B, se definen las siguientes relaciones:

1. Desempeño superior débil: Se dice que A supera en desempeño débilmente a B, si todas las soluciones de B están contenidas en A y existe al menos una solución en A que no está en B.

2. Desempeño superior fuerte: Se dice que A supera en desempeño fuertemente a B, si todas las soluciones de B son iguales a o dominadas por soluciones de A y existe al menos una solución en B que es dominada por soluciones de A.

3. Desempeño superior completo: Se dice que A supera en desempeño completamente a B, si toda solución de B es dominada por soluciones de A.

Medida C Esta medida compara dos conjuntos de vectores de decisión (conjuntos de Pareto) y calcula la proporción de vectores en el segundo conjunto para los cuales hay soluciones al menos tan buenas en cada objetivo en el primer conjunto. La función C mapea un par de vectores de decisión (solución) a un valor en el intervalo $[0, 1]$.

$$C(A, B) := |\{b \in B | \exists a \in A : a \preceq b\}| \,/\, |B| \tag{2.1}$$

Entonces $C(A, B) = 1$ implica que todos los vectores de B son al menos débilmente dominados por A. Y además $C(A, B) = 0$ indica que ninguno de los puntos de B son débilmente dominados por A. Además no se cumple necesariamente que $C(A, B) = 1 - C(B, A)$. Esta medida tiene algunos inconvenientes:

1. No puede medir relación de subconjuntos. Si A es un subconjunto de B entonces $C(A, B) = C(B, A) = 0$.

2. Si las soluciones de A no son dominadas por las de B y viceversa, entonces $C(A, B) = C(B, A) = 0$.

3. No se consideran las magnitudes de las soluciones.

Función M_3 Mide la dispersión tanto de un conjunto de vectores de decisión o de un conjunto de vectores de solución no dominados en el espacio objetivo.

$$M_3(A) = \sqrt{\sum_{i=1}^{n} \text{máx} \|a_i - b_i\| \,|a, b \in A} \tag{2.2}$$

2.4.2. Métricas basadas en puntos de referencia

Son métricas que miden lo bueno que es una solución con un simple escalar, tomando un punto de referencia, por ejemplo, en el espacio objetivo.

Medida S **o hipervolumen** Determina que tanto del espacio objetivo es débilmente dominado por un conjunto de puntos no dominados. Mide el tamaño de la porción del espacio objetivo que es dominado por este conjunto de puntos a partir de un punto de referencia, como se muestra en la Figura 2.1, donde z_{ref} es el punto de referencia, y el área de la parte sombreada sería el hipervolumen (el conjunto de puntos no dominados por la solución encontrada a partir del punto de referencia). Captura la cercanía a la frontera de Pareto real y la dispersión de las soluciones a lo largo del espacio objetivo.

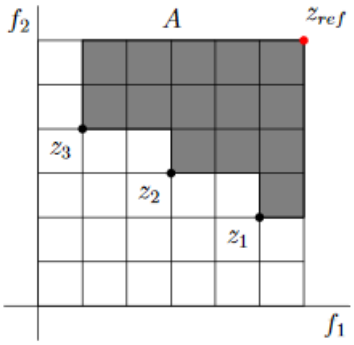

Figura 2.1: hipervolumen (Cheng et al., 2012)

Presenta algunos inconvenientes:

- Es sensible a la elección del punto de referencia.
- Los puntos extremos juegan un papel más importante que los del medio.
- Costoso de calcular.

Medida D Esta medida indica la diferencia de cubrimiento de dos conjuntos. Combina la medida C y el hipervolumen.

$$D(A, B) := S(A + B) - S(B) \quad A, B \subseteq X \tag{2.3}$$

Con X espacio de decisión. $D(A, B)$ indica el tamaño del espacio débilmente dominado por A, pero no débilmente dominado por B. Como se muestra en la Figura 2.2.

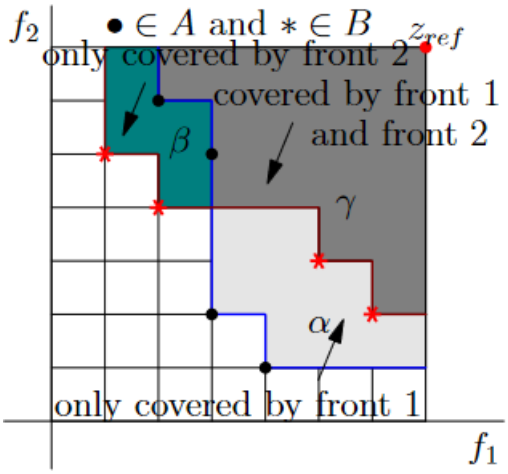

Figura 2.2: Medida D, $D(A,B) = \alpha$ y $D(B,A) = \beta$ (Cheng et al., 2012)

Si $D(A,B) = 0$ y $D(B,A) > 0$ entonces A es dominado por B.

Epsilon ϵ La medida épsilon se utiliza para evaluar el grado de dominancia entre dos soluciones en un espacio de objetivos multiobjetivo. La dominancia se refiere a la capacidad de una solución de superar o igualar a otra solución en todos los objetivos, sin ser peor en ninguno de ellos. La medida épsilon compara la distancia entre las soluciones óptimas conocidas y las soluciones obtenidas por un algoritmo. Si la distancia entre las soluciones obtenidas y las óptimas es menor o igual a un valor épsilon dado, se considera que el algoritmo ha encontrado soluciones de buena calidad.

R2 La medida R2 consiste en calcular el coeficiente de determinación R2 para cada objetivo individual y luego se promedian los valores para obtener una medida global. Cuanto más cercano a 1 sea el valor de R2, mejor será la calidad de las soluciones obtenidas por el algoritmo.

2.4.3. Métricas basadas en la frontera de Pareto real

Este tipo de métricas compara la distribución de la frontera de Pareto encontrada por el algoritmo con la frontera de Pareto real. Se utilizan para problemas de tipo benchmark, de los cuales se conoce su frontera de Pareto real. Las métricas que se analizan en dicho paper son: Distancia invertida generacional, diferencia de hipervolumen y métrica de espaciado.

Capítulo 3

Revisión de antecedentes del Location Routing Problem

En este capítulo se presenta un breve resumen del estado del arte, de forma de brindar una visión general acerca del LRP. Para ver la versión completa del estado del arte, remitirse al Anexo A: Estado del arte.
El Location Routing Problem (LRP) es un problema de optimización combinatoria que integra los problemas de localización y ruteo. El LRP busca determinar las ubicaciones óptimas para instalaciones de servicio (almacenes, centros de distribución, etc.) y al mismo tiempo optimizar las rutas de vehículos para satisfacer la demanda de los clientes. En este estado del arte, se explorarán variantes del LRP, áreas de aplicación, métodos utilizados, criterios no convergentes y su relación con la logística comercial y la logística humanitaria.

El LRP está compuesto por dos componentes fundamentales: el problema de localización de instalaciones y el problema de ruteo de vehículos. El problema de localización de instalaciones busca determinar la ubicación de ciertas instalaciones de forma de optimizar determinado objetivo, mientras que el problema de ruteo de vehículos se enfoca en optimizar las rutas de entrega para satisfacer la demanda de los clientes (Nagy & Salhi, 2007). La combinación de estos dos problemas permite abordar situaciones más complejas y realistas en la gestión de la cadena de suministro.

Existen varias variantes del LRP que abordan diferentes aspectos y restricciones, como el LRP Capacitado (CLRP), donde se consideran límites en la capacidad de los vehículos y las instalaciones (Prodhon & Prins, 2014a); el LRP Periódico (PLRP), que considera un horizonte de tiempo de múltiples períodos (Yu et al., 2009); y el LRP con Ventanas de Tiempo (LRPTW), en el que las entregas deben realizarse dentro de intervalos de tiempo específicos (Hemmel-

mayr et al., 2012).

El LRP es aplicable en diversas áreas, incluyendo la gestión de la cadena de suministro, logística humanitaria, distribución de productos y servicios, y la gestión de desastres. En la logística humanitaria, el LRP es útil para optimizar la distribución de ayuda en situaciones de emergencia (Balcik et al., 2010). En la gestión de desastres, el LRP puede ser utilizado para optimizar la ubicación de refugios y la distribución de recursos (Metropolis et al., 1953).

Diferentes enfoques han sido propuestos para resolver el LRP, incluyendo algoritmos exactos, heurísticos y metaheurísticos. Los algoritmos exactos, como la programación entera mixta (MIP), garantizan soluciones óptimas, pero pueden ser computacionalmente intensivos para problemas de gran escala (Contardo et al., 2014). Los enfoques heurísticos y metaheurísticos, como Tabú Search (Laporte et al., 2004) y los algoritmos genéticos (Prins et al., 2007), pueden proporcionan soluciones de calidad en tiempos de cómputo razonables.

En uno de los primeros trabajos que abordó el problema de LRP, se propuso una heurística basada en algoritmos de clusterización para la optimización de rutas de distribución de productos (Fisher et al., 1994). En otro trabajo, se utilizó la optimización combinatoria para resolver el problema de LRP en el contexto de la recolección de residuos (Gendreau et al., 1996).

Más recientemente, se han desarrollado nuevos enfoques basados en la optimización multiobjetivo para resolver el problema de LRP. En el trabajo de Guo et al. (2019), se presenta un enfoque basado en Teoría de Juegos para la asignación de flujos de trabajo en redes de sensores inalámbricos. En Yan et al. (2018), se propone un algoritmo genético multiobjetivo para el problema de LRP con múltiples objetivos.

Además, se han propuesto enfoques basados en técnicas de inteligencia artificial, como redes neuronales artificiales, para resolver el problema de LRP. En Li y Wen (2017) se propone una red neuronal basada en la técnica de aprendizaje profundo para resolver el problema de LRP en el contexto de la entrega de paquetes. En Zeng et al. (2020) se utiliza una red neuronal de tipo convolucional para resolver el problema de LRP en el contexto de la distribución de alimentos.

En el LRP, diferentes criterios no convergentes pueden ser considerados para la toma de decisiones, como la minimización de costos totales, la maximización de la calidad del servicio y la minimización del tiempo de tránsito. Estos criterios pueden tener diferentes prioridades en diferentes aplicaciones y pueden ser abordados mediante enfoques multiobjetivo (Deb et al., 2002).

El LRP es un problema crucial en la logística y la logística humanitaria, ya que la eficiencia en la ubicación de instalaciones y rutas de distribución tiene un impacto significativo en los costos operativos y la calidad del servicio. En

la logística humanitaria, la eficiencia en la entrega de ayuda a las poblaciones afectadas es esencial para minimizar el sufrimiento humano y garantizar una distribución justa y oportuna de los recursos (Tzeng et al., 2007).

Los objetivos de optimización en el LRP varían según las áreas de aplicación. En la gestión de la cadena de suministro, el objetivo principal suele ser minimizar los costos totales de transporte y almacenamiento (Barreto et al., 2007). En la logística humanitaria, los objetivos pueden incluir la minimización del tiempo de tránsito y la maximización de la calidad del servicio (Rawls & Turnquist, 2010). Estos objetivos pueden ser abordados mediante enfoques mono-objetivo o multiobjetivo, dependiendo de las prioridades y restricciones del problema específico.

A medida que el LRP evoluciona y se adapta a nuevos contextos, los investigadores continúan enfrentando desafíos y tendencias emergentes en la solución y aplicación de este problema. Algunas de estas tendencias incluyen:

a) Sostenibilidad: La creciente conciencia sobre el impacto ambiental de las actividades logísticas ha llevado a la inclusión de criterios de sostenibilidad en el LRP (Govindan et al., 2014). Esto implica considerar no solo los costos económicos, sino también los impactos ambientales y sociales en la optimización del problema.

b) Integración de Tecnologías de la Información: La incorporación de tecnologías de la información y comunicación en la resolución del LRP es esencial para mejorar la eficiencia y la calidad del servicio en tiempo real. Esto incluye el uso de sistemas de información geográfica (GIS), tecnologías de seguimiento y trazabilidad, y sistemas de soporte a la toma de decisiones (Tavana et al., 2017).

c) Modelos Estocásticos: La incertidumbre en la demanda y las condiciones de los clientes es un aspecto clave a tener en cuenta en la resolución del LRP. La incorporación de modelos estocásticos permite abordar esta incertidumbre y mejorar la robustez de las soluciones (Perboli et al., 2018).

d) Enfoques Híbridos y Colaborativos: La combinación de diferentes enfoques y algoritmos para resolver el LRP puede proporcionar soluciones más efectivas y robustas. Además, la colaboración entre diferentes actores en la cadena de suministro y la logística humanitaria puede mejorar la eficiencia y el impacto de las soluciones del LRP (Crainic et al., 2016).

En conclusión, el Location Routing Problem es un problema de optimización combinatoria que integra los problemas de localización y ruteo. Con numerosas variantes y aplicaciones en áreas como la logística, la logística humanitaria y la gestión de desastres, el LRP es un problema relevante y desafiante. Diferentes enfoques y criterios pueden ser considerados en función de las prioridades y restricciones específicas de cada aplicación. A medida que la investigación y

las aplicaciones del LRP evolucionan, los investigadores enfrentan desafíos y tendencias emergentes en la solución y aplicación de este problema, incluyendo la consideración de criterios de sostenibilidad, la integración de tecnologías de la información, la incorporación de modelos estocásticos y el uso de enfoques híbridos y colaborativos.

Capítulo 4

LRP multiobjetivo con criterios no convergentes

Este capítulo consta de dos secciones. En la primera sección, se presenta el trabajo de Veenstra et al. (2018a), se describe el problema, su formulación matemática y el método de resolución. En la segunda sección, se presenta una extensión del problema anterior, describiendo en que consiste dicha extensión, formulando el modelo matemático y se documenta la validación de dicho modelo.

4.1. Un problema simultáneo de localización y ruteo en la logística de salud en Países Bajos

A continuación se describe el trabajo presentado en Veenstra et al. (2018a). En términos generales, se resuelve de forma simultánea un problema de ubicación de lockers y ruteo de vehículos para la entrega de medicamentos desde una farmacia. Un locker es una palabra en inglés que se traduce al español como "armario", "casillero" o "taquilla". Se refiere a un compartimento de almacenamiento individual que generalmente se encuentra en lugares públicos, como escuelas, gimnasios, estaciones de tren, piscinas, vestuarios, entre otros. Los lockers son utilizados para que las personas puedan guardar sus pertenencias de forma segura mientras están en un lugar determinado. En este caso se utilizan para depositar medicamentos, los cuales serán retirados posteriormente por las personas. La entrega puede realizarse a lockers o a domicilio. Los lockers tienen un radio de cubrimiento, que permite a los pacientes dentro del radio tomar la medicación desde el locker.

El problema consiste en determinar qué lockers - de un conjunto de ubicaciones potenciales de lockers - se deben instalar y generar rutas que visiten esos lockers abiertos y rutas que visiten pacientes que no estén en el radio de cubri-

miento de ningún locker, a los cuales se les entregará la medicación a domicilio. Una ruta de pacientes es una secuencia ordenada de nodos que son visitados, que comienza en el depósito y terminan en el mismo. De forma análoga se definen rutas para lockers.

Características del problema:

- Lockers: se tiene un conjunto de posibles ubicaciones en donde se pueden instalar los lockers. Tienen un área de cubrimiento tal que todos los pacientes que se encuentren dentro de dicha área deben desplazarse hacia el locker para obtener los medicamentos. Los lockers pueden tener radios de cubrimientos distintos.
- Un locker atiende a un conjunto de pacientes y siempre puede abastecer a todos los pacientes que se encuentren en su radio de cubrimiento. Es decir, tienen capacidad infinita.
- Una sola farmacia como depósito: se tiene una única farmacia que actúa como depósito desde la cual se distribuyen los medicamentos a los pacientes y lockers.
- La farmacia no tiene problemas para cumplir con la demanda (tiene capacidad infinita).
- Cada cliente demanda una única unidad.
- Cada locker y cada paciente es visitado por un solo vehículo.
- Flota de vehículos: se encuentra disponible en la farmacia. Estos vehículos pueden visitar pacientes y lockers. Todos los vehículos tienen capacidad infinita.
- Las rutas para reposición de lockers y las rutas para entrega de medicamentos a domicilio son separadas. Esto se hace por las siguientes razones prácticas: es necesario garantizar que los lockers se reponen antes de una hora dada del día, lo cual es más difícil de alcanzar cuando se combinan pacientes en la misma ruta, ya que el tiempo de interacción con los pacientes es difícil de predecir. Por otro lado, el período de entrega ideal difiere en ambos: para los lockers es antes de que las personas regresen de trabajar, mientras que para la entrega a domicilio es luego de que lleguen. Por último, los conductores que realizan la entrega a domicilio deben tener mejores habilidades interpersonales.
- La duración de ruta de un vehículo consiste en la suma del tiempo de viaje y tiempo de servicio total de ese vehículo.
- Una ruta de un vehículo es factible si empieza y termina en el depósito y no excede la duración máxima de ruta.
- Cada vehículo puede realizar más de una ruta.

El objetivo del problema es determinar qué lockers instalar, generar rutas que visiten pacientes, y rutas que visiten lockers, minimizando los costos totales, que consisten en costos de ruteo y costos de instalar los lockers.

A continuación, se muestran imágenes que ejemplifican el problema y su solución. En la Figura 4.1 se muestran gráficamente los datos del problema: depósito o farmacia, pacientes, lockers y sus radios de cubrimiento. Mientras que en la Figura 4.2 se muestra gráficamente una solución al problema, se pueden apreciar las rutas que visitan solo lockers como las que visitan solo a los pacientes.

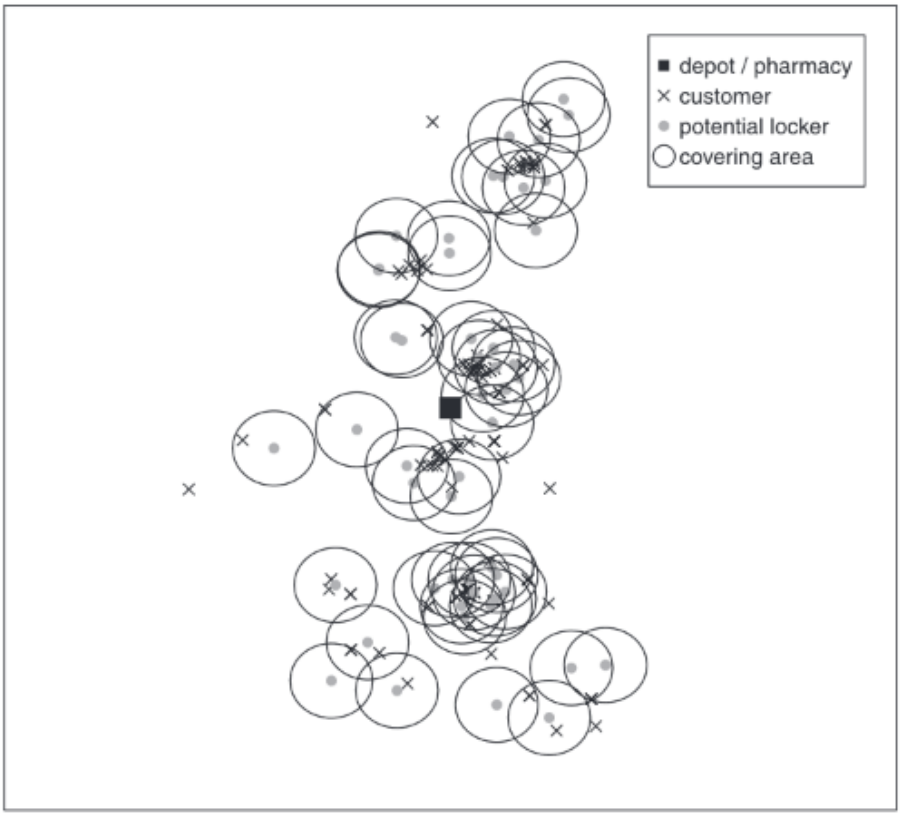

Figura 4.1: Ejemplo de datos del problema (Veenstra et al., 2018a)

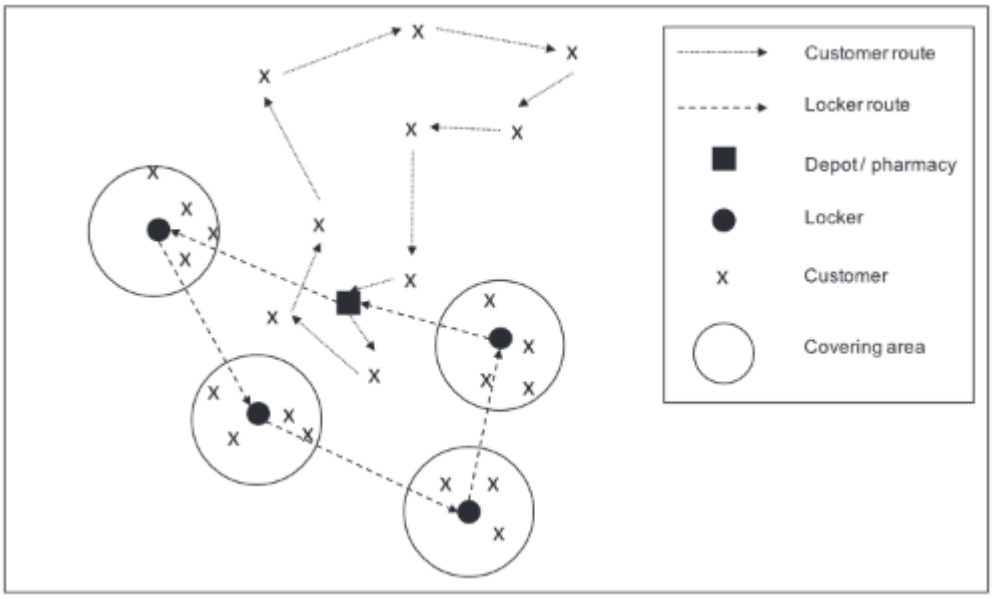

Figura 4.2: Ejemplo de solución del problema (Veenstra et al., 2018a)

4.1.1. Modelo

A continuación se describen los parámetros y variables utilizados en la formulación:

Parámetros:
$\mathcal{G} : (\mathcal{V}, \mathcal{A})$ grafo dirigido, $\mathcal{V}$ conjunto de nodos, $\mathcal{A}$ conjunto de arcos.
$\mathcal{P}$: conjunto de Pacientes.
$\mathcal{L}$: conjunto de posibles ubicaciones de lockers.
$\mathcal{V} : \{0\} \cup \mathcal{P} \cup L$ dónde 0 es la farmacia.
$\mathcal{K}$: conjunto de vehículos que visitan pacientes.
$\mathcal{M}$: conjunto de vehículos que visitan lockers.
r_j = distancia de cubrimiento del locker $j \in \mathcal{L}$.
F_j = costo fijo de instalar el locker $j \in \mathcal{L}$.
c_{ij} = costo del arco (i, j), con $i, j \in \mathcal{V}$.
d_{ij} = distancia del arco (i, j), con $i, j \in \mathcal{V}$.
t_{ij} = tiempo de recorrido del arco (i, j), con $i, j \in \mathcal{V}$.
Φ = factor de penalización de visitar a domicilio.
$T1$ = máxima duración de ruta para los vehículos $k \in \mathcal{K}$.
$T2$ = máxima duración de ruta para los vehículos $k \in \mathcal{M}$.
s_i = tiempo de servicio para $i \in \mathcal{V}$, $s_i > 0 \quad \forall i \in \mathcal{P} \cup \mathcal{L}$, $s_0 = 0$.

Variables:

$w_i : \{0,1\}$ indica si el paciente $i \in \mathcal{P}$ está asignado al depósito 0.
$z_i : \{0,1\}$ indica si el paciente $i \in \mathcal{P}$ está asignado a algún locker.
$u_{ij} : \{0,1\}$ indica si el paciente $i \in \mathcal{P}$ está asignado al locker $j \in \mathcal{L}$.
$v_i : \{0,1\}$ indica si el locker $i \in \mathcal{L}$ está abierto.
$x_{ij}^k : \{0,1\}$ indica si el vehículo $k \in \mathcal{K}$ recorre el arco (i,j), con $i,j \in P \cup \{0\}$.
$y_i^k : \{0,1\}$ indica si el vehículo $k \in \mathcal{K}$ visita el nodo $i \in \mathcal{P} \cup \{0\}$.
$h_{ij}^k \geq 0$ indica la carga del vehículo $k \in \mathcal{K}$ al atravesar el arco (i,j), con $i,j \in \mathcal{P} \cup \{0\}$.
$p_{ij}^k : \{0,1\}$ indica si el vehículo $k \in \mathcal{M}$ recorre el arco (i,j), con $i,j \in \mathcal{L} \cup \{0\}$.
$q_i^k : \{0,1\}$ indica si el vehículo $k \in \mathcal{M}$ visita el nodo $i \in \mathcal{L} \cup \{0\}$.
g_{ij}^k indica la carga del vehículo $k \in \mathcal{M}$ al atravesar el arco (i,j), con $i,j \in \mathcal{L} \cup \{0\}$.

A continuación se presenta la formulación matemática del problema propuesto en 4.2, la cual se corresponde con un modelo de programación lineal entera mixta (MILP por sus siglas en inglés).

Formulación matemática:

$$\min \quad \sum_{i\in\mathcal{P}\cup\{0\}}\sum_{j\in\mathcal{P}\cup\{0\}}\sum_{k\in\mathcal{K}}\phi c_{ij}x_{ij}^{k}+\sum_{i\in\mathcal{L}\cup\{0\}}\sum_{j\in\mathcal{L}\cup\{0\}}\sum_{k\in\mathcal{M}}c_{ij}p_{ij}^{\mathrm{k}}+\sum_{i\in\mathcal{L}}F_i\nu_i \tag{1}$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{L}}u_{ij}=z_i \qquad i\in\mathcal{P}, \tag{2}$$

$$w_i+z_i=1 \qquad i\in\mathcal{P}, \tag{3}$$

$$u_{ij}\leq v_j \qquad i\in\mathcal{P}, j\in\mathcal{L}, \tag{4}$$

$$u_{ij}d_{ij}\leq r_jv_j \qquad i\in\mathcal{P}, j\in\mathcal{L}, \tag{5}$$

$$r_jv_j\leq d_{ij}+z_iM \qquad i\in\mathcal{P}, j\in\mathcal{L}, \tag{6}$$

$$\sum_{j\in\mathcal{P}\cup\{0\}}x_{ij}^{k}+\sum_{j\in\mathcal{P}\cup\{0\}}x_{ji}^{k}=2y_i^{k} \qquad i\in\mathcal{P}, k\in\mathcal{K}, \tag{7}$$

$$\sum_{j\in\mathcal{P}}x_{0j}^{k}+\sum_{j\in\mathcal{P}}x_{j0}^{k}=2y_0^{k} \qquad k\in\mathcal{K}, \tag{8}$$

$$\sum_{k\in\mathcal{K}}y_i^{k}=w_i \qquad i\in\mathcal{P}, \tag{9}$$

$$\sum_{j\in\mathcal{P}\cup\{0\}}x_{ij}^{k}=\sum_{j\in\mathcal{P}\cup\{0\}}x_{ji}^{k} \qquad i\in\mathcal{P}, k\in\mathcal{K}, \tag{10}$$

$$\sum_{j\in\mathcal{P}}\sum_{k\in\mathcal{K}}h_{0j}^{k}=\sum_{i\in\mathcal{P}}w_i, \tag{11}$$

$$\sum_{i\in\mathcal{P}\cup\{0\}}h_{ij}^{k}-\sum_{i\in\mathcal{P}\cup\{0\}}h_{ji}^{k}=y_j^{k} \qquad j\in\mathcal{P}, k\in\mathcal{K}, \tag{12}$$

$$h_{ij}^{k}\leq(|\mathcal{P}|-1)\,x_{ij}^{k} \qquad i\in\mathcal{P}, j\in\mathcal{P}\cup\{0\}, k\in\mathcal{K}, \tag{13}$$

$$h_{0j}^{k}\leq|\mathcal{P}|\,x_{0j}^{k} \qquad j\in\mathcal{P}\cup\{0\}, k\in\mathcal{M}, \tag{14}$$

$$\sum_{i\in\mathcal{P}\cup\{0\}}\sum_{j\in\mathcal{P}\cup\{0\}}t_{ij}x_{ij}^{k}+\sum_{i\in\mathcal{P}\cup\{0\}}\sum_{j\in\mathcal{P}\cup\{0\}}s_jx_{ij}^{k}\leq T_1 \qquad k\in\mathcal{K}, \tag{15}$$

$$\sum_{j\in\mathcal{L}\cup\{0\}}p_{ij}^{k}+\sum_{j\in\mathcal{L}\cup\{0\}}p_{ji}^{k}=2q_i^{k} \qquad i\in\mathcal{L}, k\in\mathcal{M}, \tag{16}$$

$$\sum_{j\in\mathcal{L}}p_{0j}^{k}+\sum_{j\in L}p_{j0}^{k}=2q_0^{k} \qquad k\in\mathcal{M}, \tag{17}$$

$$\sum_{k\in\mathcal{M}}q_i^{k}=\nu_i \qquad i\in\mathcal{L}, \tag{18}$$

$$\sum_{j\in\mathcal{L}\cup\{0\}}p_{ij}^{k}=\sum_{j\in\mathcal{L}\cup\{0\}}p_{ji}^{k} \qquad i\in\mathcal{L}, k\in\mathcal{M}, \tag{19}$$

$$\sum_{j\in\mathcal{L}}\sum_{k\in\mathcal{M}}\mathrm{g}_{0j}^{k}=\sum_{i\in\mathcal{L}}\nu_i, \tag{20}$$

$$\sum_{i\in\mathcal{L}\cup\{0\}}\mathrm{g}_{ij}^{k}-\sum_{i\in\mathcal{L}\cup\{0\}}\mathrm{g}_{ji}^{k}=q_j^{k} \qquad j\in\mathcal{L}, k\in\mathcal{M}, \tag{21}$$

$$g_{ij}^{k}\leq(|\mathcal{L}|-1)\,p_{ij}^{k} \qquad i\in\mathcal{L}, j\in\mathcal{L}\cup\{0\}, k\in\mathcal{M}, \tag{22}$$

$$g^k_{0j} \leq |\mathcal{L}|\, p^k_{0j} \qquad j \in \mathcal{L} \cup \{0\}, k \in \mathcal{M}, \tag{23}$$

$$\sum_{i \in \mathcal{L} \cup \{0\}} \sum_{j \in \mathcal{L} \cup \{0\}} t_{ij} p^k_{ij} + \sum_{i \in \mathcal{L} \cup \{0\}} \sum_{j \in \mathcal{L} \cup \{0\}} s_j p^k_{ij} \leq T_2 \qquad k \in \mathcal{M}, \tag{24}$$

$$z_i, w_i \in \{0,1\} \qquad i \in \mathcal{P}, \tag{25}$$

$$u_{ij} \in \{0,1\} \qquad i \in \mathcal{P}, j \in \mathcal{L}, \tag{26}$$

$$v_i \in \{0,1\} \qquad i \in \mathcal{L}, \tag{27}$$

$$x^k_{ij} \in \{0,1\} \qquad i,j \in \mathcal{P} \cup \{0\}, k \in \mathcal{K}, \tag{28}$$

$$y^k_i \in \{0,1\} \qquad i \in \mathcal{P} \cup \{0\}, k \in \mathcal{K}, \tag{29}$$

$$h^k_{ij} \geq 0 \qquad i,j \in \mathcal{P} \cup \{0\}, k \in \mathcal{K}, \tag{30}$$

$$p^k_{ij} \in \{0,1\} \qquad i,j \in \mathcal{L} \cup \{0\}, k \in \mathcal{M}, \tag{31}$$

$$q^k_i \in \{0,1\} \qquad i \in \mathcal{L} \cup \{0\}, k \in \mathcal{M}, \tag{32}$$

$$g^k_{ij} \geq 0 \qquad i,j \in \mathcal{L} \cup \{0\}, k \in \mathcal{M}, \tag{33}$$

$$h^k_{i0} = 0 \qquad i \in \mathcal{P}, k \in \mathcal{K}, \tag{34}$$

$$h^k_{ii} = 0 \qquad i \in \mathcal{P} \cup \{0\}, k \in \mathcal{K}, \tag{35}$$

$$\sum_{k \in K} x^k_{i0} \leq \sum_{k \in \mathcal{K}} y^k_i \quad i \in \mathcal{P}, \tag{36}$$

$$\sum_{k \in K} x^k_{0i} \leq \sum_{k \in \mathcal{K}} y^k_i \quad i \in \mathcal{P}, \tag{37}$$

$$x^k_{ij} + x^k_{ji} \leq 1 \qquad i,j \in \mathcal{P}, k \in \mathcal{K}, \tag{38}$$

$$g^k_{i0} = 0 \qquad i \in \mathcal{L}, k \in \mathcal{K}, \tag{39}$$

$$g^k_{ii} = 0 \qquad i \in \mathcal{P} \cup \{0\}, k \in \mathcal{K}, \tag{40}$$

$$\sum_{k \in \mathcal{K}} p^k_{i0} \leq \sum_{k \in K} q^k_i \quad i \in \mathcal{L}, \tag{41}$$

$$\sum_{k \in \mathcal{K}} p^k_{0i} \leq \sum_{k \in \mathcal{K}} q^k_i \quad i \in \mathcal{L}, k \in \mathcal{M}, \tag{42}$$

$$p^k_{ij} + p^k_{ji} \leq 1 \qquad i,j \in \mathcal{L}, k \in \mathcal{M} \tag{43}$$

En la función objetivo de (1), el primer término representa el costo de las rutas a los pacientes, el segundo el costo de las rutas a los lockers, y el tercero representa el costo de instalar los lockers.
Las restricciones implican lo siguiente:

- (2): Un paciente debe estar asignado a un único locker.
- (3): Asegura que un paciente está asignado o bien a un locker o bien al depósito o farmacia.
- (4): Un paciente puede ser asignado a un locker solo si este está abierto.

- (5): Si un paciente está asignado a un locker entonces está dentro del área de cubrimiento de ese locker.
- (6): Si un paciente está dentro del área de cubrimiento de algunos lockers, debe estar asignado a alguno de esos lockers.
- (7): Si un vehículo visita a un paciente, lo hace una única vez.
- (8): Solo vehículos que se usan pueden visitar pacientes.
- (9): Si un paciente es asignado al depósito, entonces solamente un vehículo lo visita, de otra forma ningún vehículo lo visita.
- (10): Conservación de flujo. Estas restricciones se utilizan para garantizar que la cantidad de flujo que entra en un nodo sea igual a la cantidad de flujo que sale de ese mismo nodo.
- (11): Asegura que la carga total que sale del depósito es igual a la cantidad de la demanda de pacientes asignados a él.
- (12): Asegura que si un paciente es asignado a un vehículo dado, la diferencia de la carga del vehículo entrando y saliendo es exactamente uno.
- (13): Cota superior a la carga total que sale de un nodo distinto al depósito, como máximo puede ser la cantidad de pacientes menos uno, ya que se dejó una carga en ese nodo.
- (14): Cota superior a la carga total que sale del depósito, como máximo puede ser la cantidad de pacientes.
- (15): Asegura que se respeta la duración máxima para las rutas.
- (16)-(24): Similares a las restricciones (7) - (15) pero aplicando a los lockers.
- (23) - (33): Definen la naturaleza y rango de las variables.
- (34): Implica que la carga de un vehículo debe ser 0 cuando entra al depósito.
- (35): Implica que la carga de un vehículo es 0 para el arco (i, i) , con $i \in P \cup \{0\}$
- (36) y (37): Imponen que si un paciente no es visitado por un vehículo, el arco desde el paciente al depósito y el arco desde el depósito al paciente no son recorridos por un vehículo.
- (38): Excluye que un arco entre dos pacientes sea recorrido por un vehículo si el arco opuesto es atravesado por ese vehículo.
- (39) - (43): Análogas a las restricciones (34) - (38) pero aplican a las rutas de lockers.

Además, en el trabajo se define el parámetro M de la restricción (6), de forma de asegurar que se cumpla dicha restricción. Se define:

$$M = \begin{cases} \text{máx}_{j \in \mathcal{L}}\, r_j & \text{Si} \quad \exists i \in \mathcal{P}, j \in \mathcal{L} \quad s.t \quad d_{ij} < r_j, \\ \text{máx}_{j \in \mathcal{L}}\, r_j - \text{mín}_{i \in \mathcal{P}, j \in \mathcal{L} | d_{ij} < r_j}\, d_{ij} & \text{En otro caso} \end{cases}$$

4.2. LRP con criterios no convergentes de costo, eficiencia y equidad

El problema planteado a continuación está basado en el trabajo de Veenstra et al. (2018a) descrito en la sección anterior y es una extensión del mismo, la cual se describe a continuación.

La entrega a domicilio de medicamentos es una actividad en aumento que permite disminuir aglomeraciones en hospitales. Existen compañías que se dedican a esto (como Alliance Healthcare Netherlands (AHN)), y ofrecen su servicio a distintos hospitales.

En este contexto, se tiene una farmacia que cumple el papel de centro de distribución, desde donde se reparten los medicamentos por medio de un conjunto de vehículos homogéneos de capacidad Q. Cada vehículo transportará paquetes, que cuentan como una unidad; sin embargo, el tamaño de los paquetes es definido previamente y no influye en la resolución del problema. Entonces se entiende como paquete a una caja o espacio de tamaño estándar definido, que puede contener varios ítems. Los medicamentos pueden ser entregados en lockers, o ser entregados directamente en el domicilio del paciente. Las rutas de los vehículos incluyen tanto la distribución en lockers como a domicilio.

Se tienen 3 tipos de pacientes: los de alto riesgo, que necesariamente deben recibir los medicamentos en su domicilio. Los de bajo riesgo deben recibir la medicación en un locker. A los restantes se les puede asignar cualquiera de las opciones anteriores. La consideración de estos tres tipos de pacientes resulta de particular interés debido a la naturaleza de la entrega a domicilio en la realidad. Se observa que existen pacientes que, debido a diversas circunstancias, se ven imposibilitados de acceder a un locker, mientras que otros se ven en la necesidad imperativa de hacerlo, ya que, por ejemplo, pasan la mayor parte del día fuera de su domicilio. Un ejemplo esclarecedor de esta situación se encuentra en aquellos individuos que se encuentran temporalmente ausentes de su lugar de residencia. En contraste, existen pacientes a quienes les resulta indiferente recibir sus suministros médicos a través de un locker o en su propio domicilio.

Los lockers tienen un radio de cubrimiento. Por lo tanto, para que un paciente sea atendido en un locker, debe estar en el radio de cubrimiento de este.

Es importante resaltar que todos los pacientes deben ser atendidos, y que el costo de entrega a domicilio es mayor que la entrega en un locker.

Se debe determinar qué lockers instalar, los pacientes que serán atendidos en lockers y en cuál, y las rutas de cada vehículo para visitar tanto a los lockers

como a los pacientes.
Los objetivos son minimizar el costo total, constituido por el costo de las rutas más el costo de instalar los lockers, minimizar el tiempo de las rutas para cumplir con la entrega en el menor tiempo posible y maximizar la equidad de acuerdo a como se define a continuación.

A modo de resumen, se extiende el problema en las siguientes características:

- Vehículos con capacidad limitada Q. Además, cada paciente tiene una demanda de una unidad, esta unidad puede ser un único artículo o un paquete con varios artículos. Como precondición, la demanda de todos los pacientes debe poder ser abastecida con la cantidad de vehículos existentes. Es decir, siempre se puede cumplir con la demanda. Además, a cada vehículo se le asigna una única ruta.
- Las rutas generadas abarcan tanto la distribución a los pacientes como a los lockers. Esto es una mejora deseable para los objetivos de costos, eficacia y equidad, ya que si se tienen rutas separadas puede ocurrir que un vehículo se mueva de un locker a otro y entre medio existan pacientes por atender los cuales serán atendidos por otro vehículo, aumentando costos y empeorando la equidad en el tiempo de distribución. Además, notar que ejecutar dos instancias del problema, una únicamente con lockers y la otra con los pacientes que no pertenecen a lockers, se tendrían rutas separadas como en el problema original 4.1. En el caso de que se requiera cumplir con los motivos que se dan en 4.1 para separar rutas de lockers y pacientes, pero teniendo rutas mixtas habría que considerar como trabajo a futuro tener franjas horarias para lockers y pacientes.
- El problema ahora es tri-objetivo porque se busca además de minimizar costos, maximizar la equidad y minimizar del tiempo total.
- Se deben contemplar los distintos tipos de pacientes mencionados anteriormente.

El problema presenta criterios no convergentes:

- Eficacia: se busca minimizar el tiempo total del recorrido.
- Eficiencia: se busca minimizar costos.
- Equidad: Definimos la equidad de acuerdo al trabajo dé Liu et al. (2019a). La equidad se obtiene al minimizar el tiempo máximo de atención a los pacientes, es decir, que se busca atender a cada paciente en un tiempo similar.

4.2.1. Modelo

Parámetros

Para contemplar las extensiones al modelo original de Veenstra et al. (2018a) es necesario agregar y modificar variables y parámetros. A continuación se muestran los parámetros:

$\mathcal{G} : (\mathcal{V}, \mathcal{A})$ grafo dirigido, $\mathcal{V}$ conjunto de nodos, $\mathcal{A}$ de arcos.
$\mathcal{P}$: conjunto de Pacientes.
$\mathcal{L}$: conjunto de posibles ubicaciones de Lockers.
$\mathcal{V} : \mathcal{P} \cup \mathcal{L} \cup \{0\}$.
$\mathcal{K}$: conjunto de vehículos.
Q = capacidad de los vehículos.
r_j = distancia de cubrimiento del locker $j \in \mathcal{L}$.
C_j = costo fijo de abrir el locker $j \in \mathcal{L}$.
c_{ij} = costo del arco (i, j), con $i, j \in \mathcal{V}$.
d_{ij} = distancia del arco (i, j), con $i, j \in \mathcal{V}$.
t_{ij} = tiempo de recorrido del arco (i, j), con $i, j \in \mathcal{V}$.
s_i = tiempo de servicio para $i \in \mathcal{V}$.
VT_i = tiempo de llegada al nodo $i \in \mathcal{V}$ desde el depósito 0.
Φ_i = factor de penalización de visitar a domicilio para $i \in \mathcal{P}$, y 1 para $i \in \mathcal{L}$.
$tipo(p) \in \{1, 2, 3\}$ $p \in \mathcal{P}$ donde 1 = Alto riesgo, 2 = Indiferente y 3 = Bajo riesgo.

Variables

A continuación se muestran las variables:
$z_i \in \{0, 1\}$ indica si el paciente $i \in \mathcal{P}$ está asignado a algún locker.
$u_{ij} \in \{0, 1\}$ indica si el paciente $i \in \mathcal{P}$ está asignado al locker $j \in L$.
$v_i \in \{0, 1\}$ indica si el locker $i \in \mathcal{L}$ está abierto o no.
$w_i = \{0, 1\}$ indica si el paciente $i \in \mathcal{P}$ está asignado al depósito 0.
$x_{ij}^k \in \{0, 1\}$ indica si el vehículo $k \in \mathcal{K}$ recorre el arco (i, j), con $i, j \in V$.
$y_i^k \in \{0, 1\}$ indica si el vehículo $k \in \mathcal{K}$ visita el nodo $i \in \mathcal{V}$.
h_{ij}^k indica la carga del vehículo $k \in \mathcal{K}$ al atravesar el arco (i, j), con $i, j \in \mathcal{V}$.

Formulación

$$\min \quad F = (F_1, F_2, F_3) \tag{44}$$

$$\text{s.t.} \quad F_1 = \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}}\sum_{k\in\mathcal{K}} \Phi_j c_{ij} x_{ij}^k + \sum_{i\in\mathcal{L}} C_i v_i, \tag{45}$$

$$F_2 = \sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}} t_{ij} x_{ij}^k + \sum_{i\in\mathcal{L}} s_i v_i + \sum_{i\in\mathcal{P}} s_i \left(1 - z_i\right), \tag{46}$$

$$F_3 \geq VT_i \quad i \in \mathcal{V}, \tag{47}$$

$$VT_i \geq VT_j + t_{ji} + s_i - M\left(1 - \sum_{k\in\mathcal{K}} x_{ji}^k\right) \quad i \in \mathcal{V}, j \in \mathcal{V}, \tag{48}$$

$$VT_0 = 0, \tag{49}$$

$$\sum_{j\in\mathcal{L}} u_{ij} = z_i \quad i \in \mathcal{P}, \tag{50}$$

$$w_i + z_i = 1 \quad i \in \mathcal{P}, \tag{51}$$

$$u_{ij} \leq v_j \quad i \in \mathcal{P}, j \in \mathcal{L}, \tag{52}$$

$$u_{ij} d_{ij} \leq r_j v_j \quad i \in \mathcal{P}, j \in \mathcal{L}, \tag{53}$$

$$\sum_{j\in\mathcal{V}} x_{ij}^k + \sum_{j\in\mathcal{V}} x_{ji}^k = 2y_i^k \quad i \in \mathcal{P}\cup\mathcal{L}, k \in \mathcal{K}, \tag{54}$$

$$\sum_{j\in\mathcal{V}\setminus\{0\}} x_{0j}^k + \sum_{j\in\mathcal{V}\setminus\{0\}} x_{j0}^k = 2y_0^k \quad k \in \mathcal{K}, \tag{55}$$

$$\sum_{k\in\mathcal{K}} y_i^k = w_i \quad i \in \mathcal{P}, \tag{56}$$

$$\sum_{k\in\mathcal{K}} y_i^k = v_i \quad i \in \mathcal{L}, \tag{57}$$

$$\sum_{j\in\mathcal{V}} x_{ij}^k = \sum_{j\in\mathcal{V}} x_{ji}^k \quad k \in \mathcal{K}, i \in \mathcal{V}, \tag{58}$$

$$\sum_{j\in\mathcal{P}\cup\mathcal{L}}\sum_{k\in\mathcal{K}} h_{0j}^k = |\mathcal{P}|, \tag{59}$$

$$\sum_{i\in\mathcal{V}} h_{ij}^k - \sum_{i\in\mathcal{V}} h_{ji}^k = y_j^k \quad j \in \mathcal{P}, k \in \mathcal{K}, \tag{60}$$

$$\sum_{i\in\mathcal{V}} h_{ij}^k - \sum_{i\in\mathcal{V}} h_{ji}^k = \sum_{i\in\mathcal{P}} u_{ij} y_j^k \quad j \in \mathcal{L}, k \in \mathcal{K}, \tag{61}$$

$$h_{ij}^k \leq (|\mathcal{P}| - 1)\, x_{ij}^k \qquad i \in \mathcal{P}, j \in \mathcal{V}, k \in \mathcal{K}, \tag{62}$$

$$h_{ij}^k \leq |\mathcal{P}| - \sum_{m\in\mathcal{P}} u_{mi} \qquad i \in \mathcal{L}, j \in \mathcal{V}, k \in \mathcal{K}, \tag{63}$$

$$h_{ij}^k \leq N x_{ij}^k \qquad i \in \mathcal{L}, j \in \mathcal{V}, k \in \mathcal{K}, \tag{63b}$$

$$h_{0j}^k \leq |\mathcal{P}|\, x_{0j}^k \qquad j \in \mathcal{V}, k \in \mathcal{K}, \tag{64}$$

$$z_i = 1 \qquad i \in \mathcal{P}, tipo\,(i) = 3, \tag{65}$$

$$z_i = 0 \qquad i \in \mathcal{P}, tipo\,(i) = 1, \tag{66}$$

$$\sum_{k \in \mathcal{K}} y_i^k = 1 - z_i \qquad i \in \mathcal{P}, \tag{67}$$

$$\sum_{j \in \mathcal{V}} x_{0j}^k \leq 1 \qquad k \in \mathcal{K}, \tag{68}$$

$$\sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{L}} (x_{ij}^k * \sum_{p \in \mathcal{P}} u_{pj}) + \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{P}} x_{ij}^k \leq Q \qquad k \in \mathcal{K}, \tag{69}$$

$$\sum_{k \in \mathcal{K}} x_{i0}^k \leq \sum_{k \in \mathcal{K}} y_i^k \quad i \in \mathcal{V}, \tag{70}$$

$$\sum_{k \in \mathcal{K}} x_{0i}^k \leq \sum_{k \in \mathcal{K}} y_i^k \quad i \in \mathcal{V}, \tag{71}$$

$$x_{ij}^k + x_{ji}^k \leq 1 \qquad i, j \in \mathcal{V}, k \in \mathcal{K}, \tag{72}$$

$$\sum_{i \in \mathcal{P}} u_{ij} \leq Q \qquad j \in \mathcal{L}, \tag{73}$$

$$\sum_{i \in \mathcal{P}} u_{ij} \geq v_j \qquad j \in \mathcal{L}, \tag{74}$$

$$h_{i0}^k = 0 \qquad i \in \mathcal{V}, k \in \mathcal{K}, \tag{75}$$

$$h_{ii}^k = 0 \qquad i \in \mathcal{V} \cup \{0\}, k \in \mathcal{K}, \tag{76}$$

$$h_{ij}^k \geq 0 \qquad i, j \in \mathcal{V}, k \in \mathcal{K}, \tag{77}$$

$$u_{ij} \in \{0, 1\} \qquad i \in \mathcal{P}, j \in \mathcal{L}, \tag{78}$$

$$v_i \in \{0, 1\} \qquad i \in \mathcal{L}, \tag{79}$$

$$x_{ij}^k \in \{0, 1\} \qquad i, j \in \mathcal{P} \cup \{0\}, k \in \mathcal{K}, \tag{80}$$

$$y_i^k \in \{0, 1\} \qquad i \in \mathcal{V}, k \in \mathcal{K}, \tag{81}$$

$$z_i, w_i \in \{0, 1\} \qquad i \in \mathcal{P} \tag{82}$$

Donde M = la suma de todos los t_{ji} y N = $|\mathcal{P}|$.

Funciones objetivos:

- (45): Es el costo de las rutas más el costo de instalación de los lockers.
- (46): Es el tiempo total de recorrido más el de servicio. El primer sumando es el tiempo de desplazamiento entre nodos, el segundo sumando es el tiempo de servicio de los lockers y el tercer sumando es el tiempo de servicio a los pacientes.
- (47), (48), (49): Representan la función de equidad, la cual se define como el máximo tiempo de atención a un cliente. Esta función se implementa utilizando la variable VT_i la cual representa el tiempo de atención del

nodo i. Para formular el tiempo máximo de atención a los nodos de V se utiliza la restricción (48) la cual genera una restricción para cada par de nodos de V de forma que para los nodos que no están conectados se les asigna un valor muy bajo mediante el sumando $-M(1-\sum_{k\in\mathcal{K}} x_{ji}^k)$, dado que solo hay una arista para cada nodo visitado, VT_i va a ser mayor o igual al tiempo máximo de atención al nodo i y dado que $F3$ es mayor o igual a cada uno de los VT_i se cumple que es mayor o igual al tiempo máximo de atención de los nodos dé V. Luego al minimizar $F3$ se obtiene el mínimo del máximo tiempo de atención a los nodos dé V.

Las restricciones implican lo siguiente:

- (50): Si un paciente está asignado a un locker está asignado a un único locker. Esta restricción es igual a la restricción (2) del modelo original.
- (51): Un paciente está asignado a un locker o al depósito. Esta restricción es igual a la restricción (3) del modelo original.
- (52): Si un paciente está asignado a un locker, ese locker está abierto. Los lockers cerrados no pueden tener pacientes asignados. Esta restricción es igual a la restricción (4) del modelo original.
- (53): Si un paciente está asignado a un locker entonces está dentro del área de cubrimiento de ese locker. Esta restricción es igual a la restricción (5) del modelo original.
- (54): Si un vehículo visita un paciente o locker, entra y sale una única vez. Esta restricción contiene las restricciones (7) y (16) del modelo original.
- (55): Todo vehículo sale y comienza en el mismo depósito una sola vez. Esta restricción contiene las restricciones (8) y (17) del modelo original.
- (56): Si un paciente está asignado al depósito, entonces solo un vehículo lo visita. Esta restricción se corresponde con la restricción (9) del modelo original.
- (57): Si un locker está abierto, entonces solo un vehículo lo visita. Esta restricción es similar a la restricción (18) del modelo original.
- (58): Conservación de flujo. Se corresponden con las restricciones (10) y (19) del modelo original.
- (59): Asegura que la carga total que sale del depósito es igual a la cantidad de pacientes. Esta restricción es similar a la restricción (11) del modelo original.
- (60): Si un vehículo visita un paciente, debe dejar una carga en el mismo. Esta restricción se desprende de la restricción (12) del modelo original.

- (61): Si un vehículo visita un locker, debe dejar una carga igual a la cantidad de pacientes asignados al locker. Esta restricción es adaptada de la restricción (21) del modelo original.

- (62): Cota superior de la carga que lleva un vehículo después de visitar un paciente. Esta restricción es similar a la restricción (13) del modelo original.

- (63): Cota superior de la carga que lleva un vehículo después de visitar un locker. Esta restricción es similar a la restricción (22)

- (64): Cota superior de la carga que lleva un vehículo después de salir del depósito. Esta restricción corresponde a las restricciones (14) y (23) del modelo original.

- (65): Si un paciente es de tipo 3 debe estar asignado a un locker.

- (66): Si un paciente es de tipo 1 no debe estar asignado a ningún locker.

- (67): Si un vehículo visita a un paciente, el paciente no está asignado a ningún locker.

- (68): Solo se permite una ruta por vehículo.

- (69): Para cada vehículo la demanda de su ruta debe ser menor o igual a su capacidad.

- (70) y (71): Imponen que si un nodo no es visitado por un vehículo, el arco desde el nodo al depósito y el arco desde el depósito al nodo no son recorridos por un vehículo.

- (72): Excluye que un arco entre dos nodos sea recorrido por un vehículo si el arco opuesto es atravesado por ese vehículo. Esta restricción contiene las restricciones (38) y (43) del modelo original.

- (73): La cantidad de pacientes asignados a un locker no puede exceder la capacidad de los vehículos.

- (74): Si un locker no tiene pacientes asignados, entonces debe estar cerrado.

- (75): Implica que la carga de un vehículo debe ser 0 cuando entra al depósito. Esta restricción contiene las restricciones (34) y (39) del modelo original.

- (76): Implica que la carga de un vehículo debe ser 0 en cada arco $(i, i), \quad i \in \mathcal{V} \cup \{0\}$. Esta restricción contiene las restricciones (35) y (40) del modelo original.

- (77) a (81): Definen la naturaleza y el rango de las variables.

Como se puede observar en el modelo, existe una multiplicación de variables en las restricciones (61), (63) y (69), haciendo que el problema sea no lineal. Los problemas no lineales son en general más difíciles de resolver que los lineales, por lo que se decidió aplicar la siguiente linealización. Las restricciones (61), (63) y (69) son de la forma: $z[i,j] = x[i,j] * y[i,j]$. Dado que tanto x como y son binarias se puede aplicar la siguiente linealización:

$z[i,j] <= U[i,j] * y[i,j]$
$z[i,j] <= x[i,j]$
$z[i,j] >= x[i,j] - U[i,j] * (1 - y[i,j])$
$0 <= z[i,j] <= U[i,j]$

Donde: $0 <= x[i,j] <= U[i,j]$.

En este caso U es 1 dado que x es binaria. Por lo que z también es binaria. Para demostrar que la linealización es correcta se puede crear una tabla de verdad para x e y y ver que $z[i,j] = x[i,j] * y[i,j]$, dado que tanto x como y son binarias.

x	y	$x * y$	z
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

Tabla 4.1: Demostración correctitud linealización

A continuación se muestra como queda cada restricción no lineal luego de aplicada la linealización. Donde $PRODVAR$, $PRODVAR1$ y $PRODVAR2$ son la variable z mencionada en la linealización anterior.

Para el caso de la restricción (61):

$\sum_{i \in \mathcal{V}} h_{ij}^k - \sum_{i \in \mathcal{V}} h_{ji}^k = \sum_{i \in \mathcal{P}} u_{ij} y_j^k \quad j \in \mathcal{L}, k \in \mathcal{K}$

El producto $u_{ij} y_j^k$ es un producto de variables donde ambas variables son binarias. Sustituyendo $u_{ij} y_j^k$ por $PRODVAR_{ij}^k$ y aplicando la linealización mencionada anteriormente se cambia la restricción (61) por los siguientes conjuntos de restricciones:

$\sum_{i \in \mathcal{V}} h_{ij}^k - \sum_{i \in \mathcal{V}} h_{ji}^k = \sum_{i \in \mathcal{P}} PRODVAR_{ij}^k \quad j \in \mathcal{L}, k \in \mathcal{K}$
$PRODVAR_{ij}^k <= y_j^k \quad i \in \mathcal{P}, j \in \mathcal{L}, k \in \mathcal{K}$
$PRODVAR_{ij}^k <= u_{ij} \quad i \in \mathcal{P}, j \in \mathcal{L}, k \in \mathcal{K}$
$PRODVAR_{ij}^k >= u_{ij} - (1 - y_j^k) \quad i \in \mathcal{P}, j \in \mathcal{L}, k \in \mathcal{K}$

Para el caso de la restricción (63):

$$h_{ij}^k \leq \left(|\mathcal{P}| - \sum_{m\in\mathcal{P}} u_{mi}\right) x_{ij}^k \qquad i \in \mathcal{L}, j \in \mathcal{V}, k \in \mathcal{K}$$

Aplicando distributiva:

$$h_{ij}^k \leq \left(|\mathcal{P}|\, x_{ij}^k - \sum_{m\in\mathcal{P}} u_{mi} x_{ij}^k\right) \qquad i \in \mathcal{L}, j \in \mathcal{V}, k \in \mathcal{K}$$

El producto $u_{mi}x_{ij}^k$ es un producto de variables donde ambas variables son binarias. Sustituyendo $u_{mi}x_{ij}^k$ por $PRODVAR1_{mij}^k$ y aplicando la linealización mencionada anteriormente se cambia la restricción (63) por los siguientes conjuntos de restricciones:

$$h_{ij}^k \leq \left(|\mathcal{P}|\, x_{ij}^k - \sum_{m\in\mathcal{P}} PRODVAR1_{mij}^k\right) \qquad i \in \mathcal{L}, j \in \mathcal{V}, k \in \mathcal{K}$$
$$PRODVAR1_{mij}^k <= x_{ij}^k \quad m \in \mathcal{P}, i \in \mathcal{L}, j \in \mathcal{V}, k \in \mathcal{K}$$
$$PRODVAR1_{mij}^k <= u_{mi} \quad m \in \mathcal{P}, i \in \mathcal{L}, j \in \mathcal{V}, k \in \mathcal{K}$$
$$PRODVAR1_{mij}^k >= u_{mi} - (1 - x_{ij}^k) \quad m \in \mathcal{P}, i \in \mathcal{L}, j \in \mathcal{V}, k \in \mathcal{K}$$

Para el caso de la restricción (69):

$$\sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{L}} (x_{ij}^k * \sum_{p\in\mathcal{P}} u_{pj}) + \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{P}} x_{ij}^k \leq Q \qquad k \in \mathcal{K}$$

Aplicando distributiva:

$$\sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{L}} (\sum_{p\in\mathcal{P}} x_{ij}^k * u_{pj}) + \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{P}} x_{ij}^k \leq Q \qquad k \in \mathcal{K}$$

El producto $u_{pj}x_{ij}^k$ es un producto de variables donde ambas variables son binarias. Sustituyendo $u_{pj}x_{ij}^k$ por $PRODVAR2_{pij}^k$ y aplicando la linealización mencionada anteriormente se cambia la restricción (69) por los siguientes conjuntos de restricciones:

$$\sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{L}} \sum_{p\in\mathcal{P}} PRODVAR2_{pij}^k + \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{P}} x_{ij}^k \leq Q \qquad k \in \mathcal{K}$$
$$PRODVAR2_{pij}^k <= x_{ij}^k \quad m \in \mathcal{P}, i \in \mathcal{L}, j \in \mathcal{V}, k \in \mathcal{K}$$
$$PRODVAR2_{pij}^k <= u_{pi} \quad m \in \mathcal{P}, i \in \mathcal{L}, j \in \mathcal{V}, k \in \mathcal{K}$$
$$PRODVAR2_{pij}^k >= u_{pi} - (1 - x_{ij}^k) \quad m \in \mathcal{P}, i \in \mathcal{L}, j \in \mathcal{V}, k \in \mathcal{K}$$

4.2.2. Validación del modelo

Para validar el modelo se utiliza el software GLPK. Se implementa el modelo en el mismo y se resuelve para instancias pequeñas verificando de forma manual que la solución encontrada sea factible y que el valor encontrado de la función objetivo sea realmente el óptimo, esto es posible debido a que las instancias son chicas. Primero se evalúa cada función objetivo de forma individual, es decir se generan tres modelos mono-objetivos que minimizan cada una de las funciones objetivo por separado. Para analizar el resultado obtenido por el software GLPK, es fundamental poder visualizar los datos de manera amigable, y también ingresar los datos a través de una interfaz gráfica y no tener que definir variables y matrices en un archivo. Para eso se desarrolla una aplicación que consta de una API en .Net Core y un cliente en React. La versión final del prototipo se detalla en la Sección 6.2. El prototipo permite, entre otras funcionalidades, generar instancias, ver las instancias en el mapa, ejecutar algoritmos y ver las soluciones en el mapa, funcionalidades que se utilizan para realizar las validaciones al modelo que se describen en esta sección. Todos los casos de prueba detallados a continuación se ejecutaron en un equipo con 16 GB de RAM, un procesador Core i7-8650U 1.90 GHz, y sistema operativo Windows 10 de 64 bits.

Validaciones sobre cada función objetivo

El primer paso para validar el modelo fue probar individualmente cada función objetivo. Esto facilita las pruebas y permite ir aumentando progresivamente la complejidad de los casos.

Validación F_1

Para validar esta función se hicieron las siguientes pruebas:

- **Prueba 1**: El escenario consta de 3 pacientes tipo indiferente, 3 lockers, el depósito y un vehículo, como se muestra en la Figura 4.3.

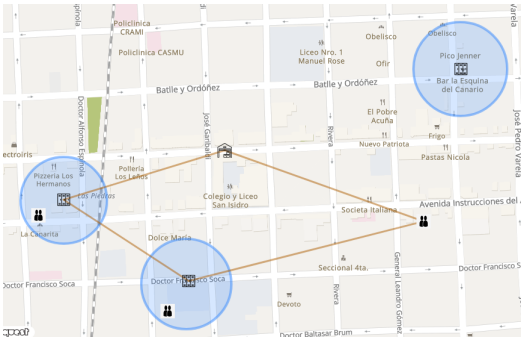

Figura 4.3: Instancia Prueba 1 de la Validación de F1.

- **Prueba 2**: Esta instancia es similar al anterior, pero se agrega otro vehículo. En este caso se detectan errores sobre las restricciones (62) que no admitía más de un vehículo. Para corregir se aplica linealización. Por otro lado, se corrige las restricciones (81), ya que no permitía rutas del tipo depósito - nodo - depósito. Luego de las correcciones se obtiene la solución esperada (Figura 4.4).

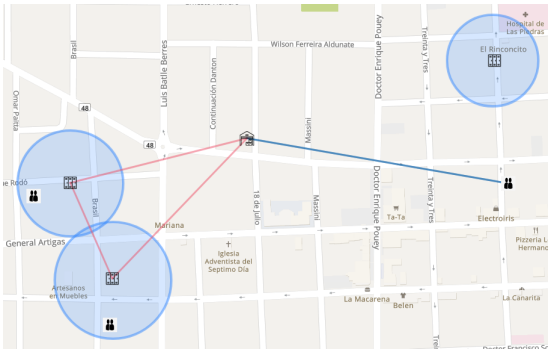

Figura 4.4: Solución obtenida para la prueba 2 de la Validación de F1 luego de correcciones.

- **Prueba 3**: Se prueba la variante de tener a un paciente de tipo alto riesgo y se verifica que no se asigne al locker, ya que necesariamente se debe atender en su hogar. La solución se muestra en la Figura 4.5

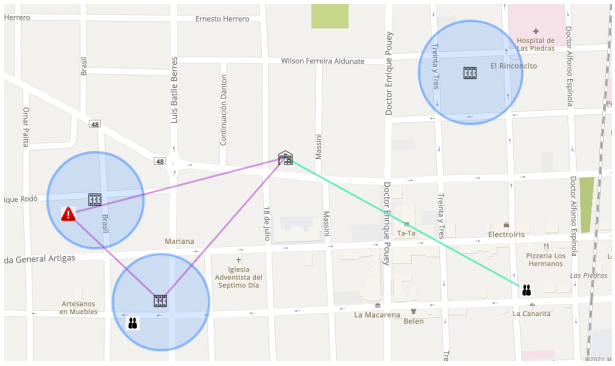

Figura 4.5: Solución obtenida para la prueba 3 de la Validación de F1.

- **Prueba 4**: Se tiene un paciente de tipo bajo riesgo, por lo tanto, necesariamente debe ser atendido en un locker. Como el paciente no está en el rango de cubrimiento de ningún locker, no se encuentra solución factible.

- **Prueba 5**: Se prueba que se asigne más de un paciente a un locker. La solución se muestra en la Figura 4.6

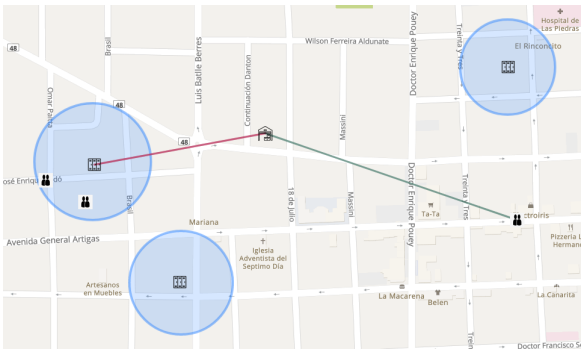

Figura 4.6: Solución obtenida para la prueba 5 de la Validación de F1.

- **Prueba 6**: En este caso se cuenta con 2 pacientes en el rango de cubrimiento de 2 lockers. Se verifica como se asignan ambos al mismo locker (al menos costoso) y esto es correcto porque se evita el costo de apertura del segundo locker. La solución se muestra en la Figura 4.7.

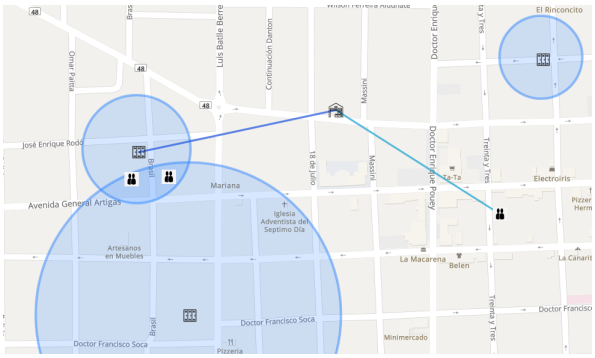

Figura 4.7: Solución obtenida para la prueba 6 de la Validación de F1.

- **Prueba 7**: Se prueba con 3 vehículos y capacidad 1. La solución se muestra en la Figura 4.8.

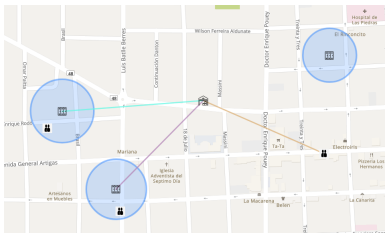

Figura 4.8: Solución obtenida para la prueba 7 de la Validación de F1.

- **Prueba 8**: Se tienen 3 vehículos y capacidad 1. Dos de los pacientes están en el rango de cubrimiento de un locker, se verifica que no se asignen 2 pacientes al locker, ya que la capacidad de los vehículos es 1. La solución se muestra en la Figura 4.9

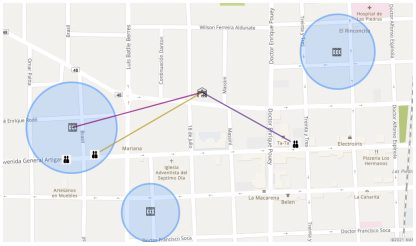

Figura 4.9: Solución obtenida para la prueba 8 de la Validación de F1.

Validación F_2

Dado que la función $F2$ es similar a la función F1, se utilizan los mismos casos de prueba que para la función $F1$. Los resultados fueron similares y no condujeron a arreglos en el modelo.

Validación F_3

$F3$ busca que cada paciente sea atendido en un tiempo similar.

- **Prueba 1**: Para esta prueba se utilizan 3 vehículos con capacidad 3. Además, el tiempo de ir de l0 a p0 es mayor que el tiempo de ir de l0 a p1. Y el tiempo de p1 a p2 y de p0 a p2 es mayor que el tiempo de 0 a p2, lo que justifica el uso de 2 vehículos. La solución se muestra en la Figura 4.10

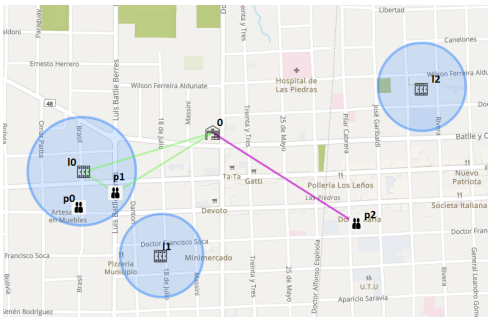

Figura 4.10: Solución obtenida para la prueba 1 de la validación de F3.

- **Prueba 2**: En esta prueba se utilizan 4 pacientes indiferentes y 2 vehículos con capacidad 3. Se obtiene la siguiente solución que se entiende que es correcta o al menos mejor que otras posibles soluciones. Otra solución puede que un vehículo visite p0 y p1, y otro visite p2 y p3. Pero esta segunda ruta es más larga en duración que cualquiera de las dos de esta solución.

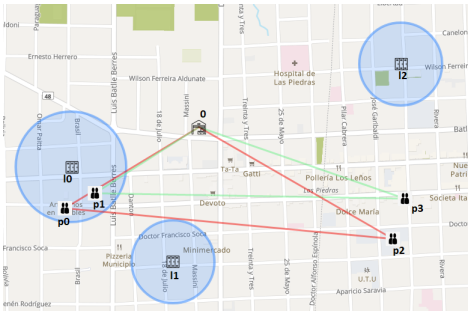

Figura 4.11: Solución obtenida para la prueba 2 de la validación de F3.

- **Prueba 3**: En este caso se le agrega un vehículo al caso anterior y se puede observar que es utilizado este otro vehículo para disminuir los tiempos de atención.

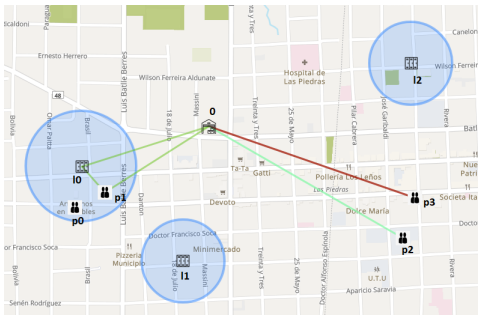

Figura 4.12: Solución obtenida para la prueba 3 de la validación de F3.

Validación de una única función: Luego de la validación de cada función objetivo de manera individual, se pretende validar el modelo para una única función. Recordando que GLPK es mono-objetivo, se debe de tener una única función que contemple los 3 objetivos.
Se utilizan dos maneras de construir esta única función objetivo.

- **Por pesos:** se expresa como $\alpha * F_1 + \beta * F_2 + (1 - \alpha - \beta) * F_3$. En este caso fijamos $\alpha = \beta = \frac{1}{3}$.

- **Por normalización:** se expresa como $(F_1 - \hat{F_1}) + (F_2 - \hat{F_2}) + (F_3 - \hat{F_3})$. Donde $\hat{F_1}, \hat{F_2}$ y $\hat{F_3}$ son las soluciones minimizando individualmente F_1, F_2 y F_3 respectivamente.

Los escenarios fueron resueltos utilizando ambas funciones objetivos (por pesos y por normalización), y siempre que se obtuvo la solución óptima el resultado fue el mismo.

Sobre los distintos escenarios a simular, se recurre al paper original para conocer cómo probaron su solución. Se evidencia que probaron sobre 3 conjuntos de pruebas: uno aleatorio que generaron, una adaptación de un conjunto de datos reales proporcionados por AHN, y benchmarks adaptados del LRP. Además, anuncian la falta de pruebas sobre escenarios con gran cantidad de lockers.

A continuación se muestran los diversos escenarios donde se prueba la solución:

- **Caso base 1**: Se sitúa un paciente de bajo riesgo en el radio de cubrimiento de un locker. Al ser de bajo riesgo, debe necesariamente ser asignado al locker. La solución encontrada por GLPK se muestra en la Figura 4.13 la cual es la solución esperada.

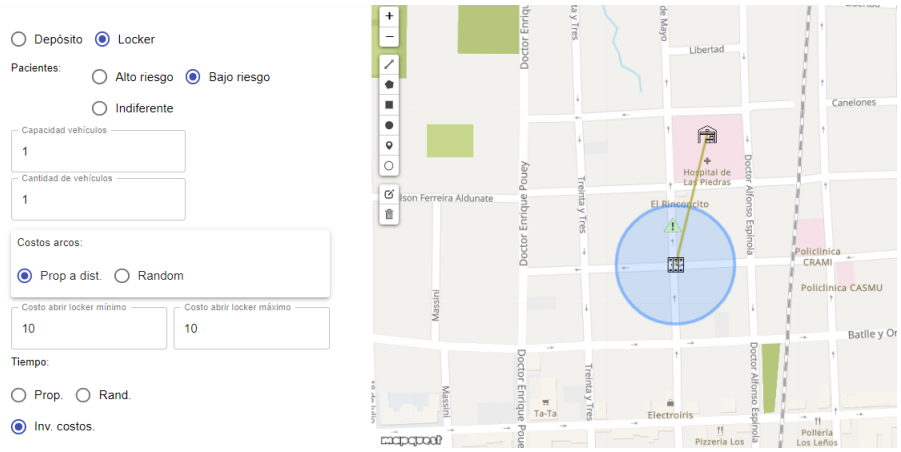

Figura 4.13: Solución al caso base 1 para la validación de la función objetivo por pesos y por normalización.

- **Caso base 2**: ahora el paciente es de alto riesgo, por lo que necesariamente se le debe entregar a domicilio y no puede ser asignado al locker. La solución encontrada por GLPK se muestra en la Figura 4.14 la cual es la solución esperada.

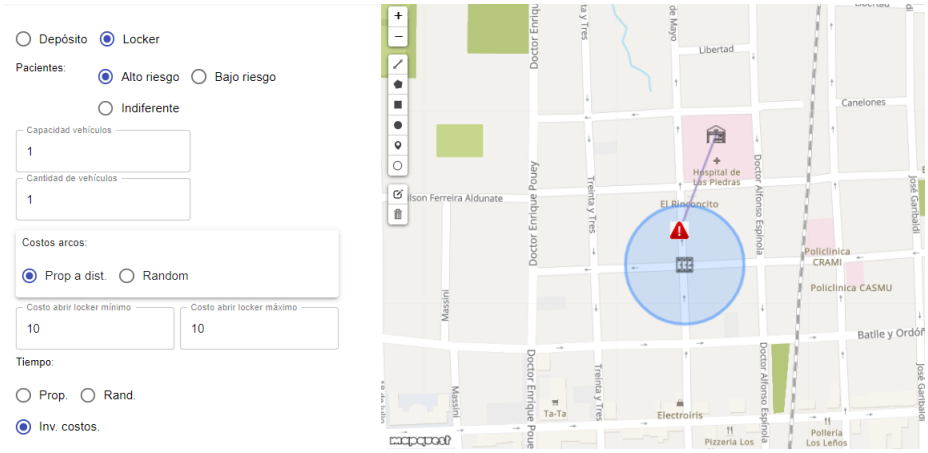

Figura 4.14: Solución de GLPK al caso base 2 para la validación de la función objetivo por pesos y por normalización. Se corresponde con la solución esperada.

- **Caso base 3**: el paciente de tipo indiferente se le entrega a domicilio. La solución encontrada por GLPK se muestra en la Figura 4.15 la cual es la solución esperada.

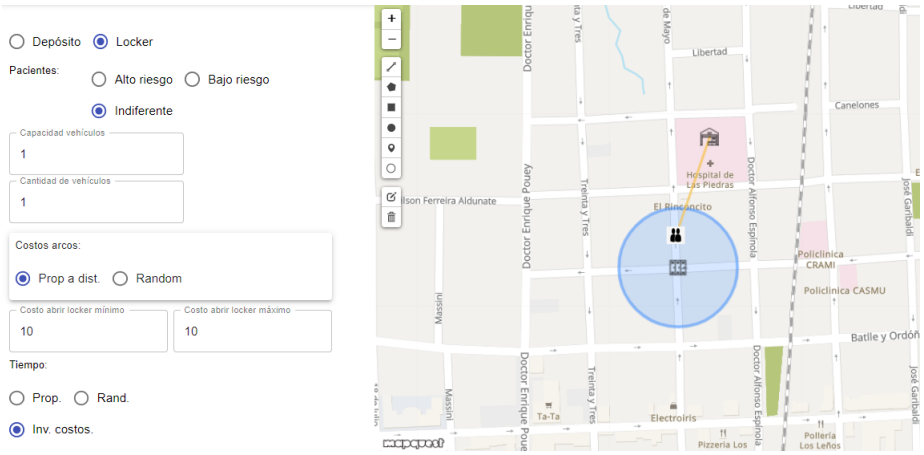

Figura 4.15: Solución de GLPK al caso base 3 para la validación de la función objetivo por pesos y por normalización. Se corresponde con la solución esperada.

- **Caso base 4**: mismo caso al anterior, pero se disminuye el costo y servicio de lockers, por lo que el paciente es asignado al locker. La solución encontrada por GLPK se muestra en la Figura 4.16 la cual es la solución esperada.

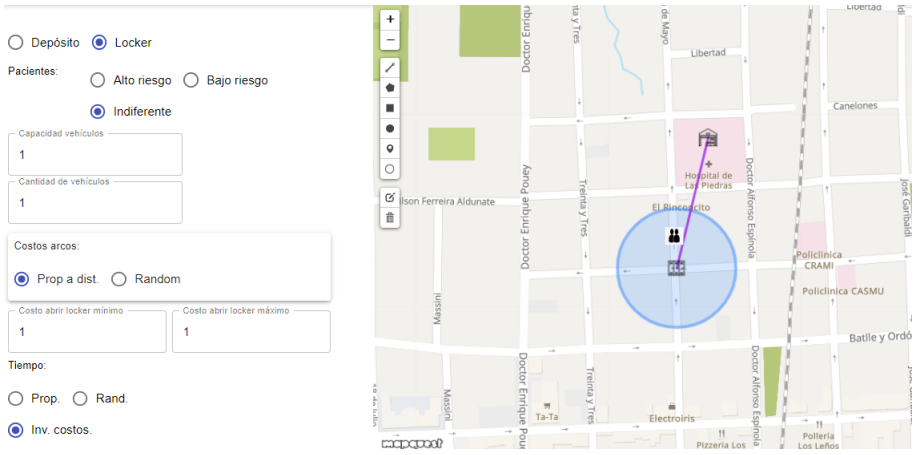

Figura 4.16: Solución de GLPK al caso base 4 para la validación de la función objetivo por pesos y por normalización. Se corresponde con la solución esperada.

- **Caso medio 1**: Se tienen 8 pacientes, 3 lockers y un vehículo. El comportamiento es correcto asignando a los pacientes de bajo riesgo a los

lockers, y yendo al domicilio de los pacientes de alto riesgo y de los que no se encuentran en el radio de cubrimiento de algún locker. La solución encontrada por GLPK se muestra en la Figura 4.17 la cual es la solución esperada.

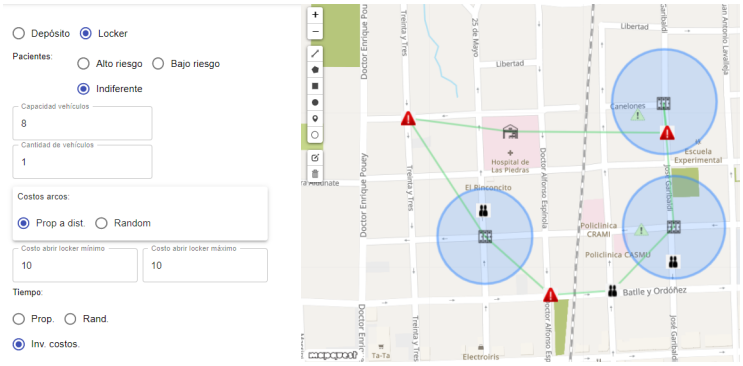

Figura 4.17: Solución de GLPK al caso medio 1 para la validación de la función objetivo por pesos y por normalización. Se corresponde con la solución esperada.

- **Caso medio 2**: Mismo caso anterior, pero ahora se tienen 4 vehículos de capacidad 2 cada uno. Se utilizan todos para cumplir con la demanda. La solución encontrada por GLPK se muestra en la Figura 4.18 la cual es la solución esperada.

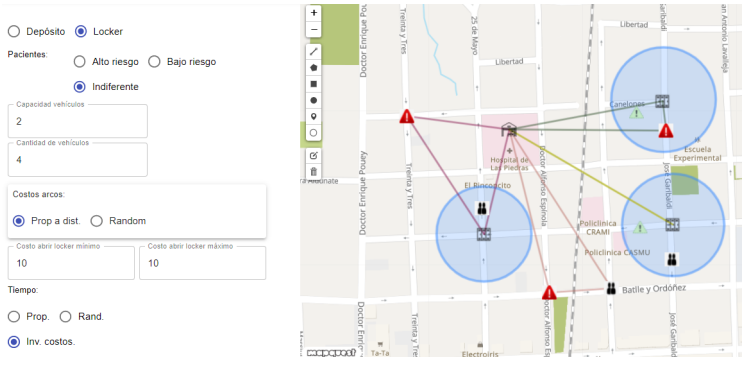

Figura 4.18: Solución de GLPK al caso medio 2 para la validación de la función objetivo por pesos y por normalización. Se corresponde con la solución esperada.

- **Caso medio 3**: Este escenario tiene más lockers que pacientes, 6 y 3 respectivamente. La solución encontrada por GLPK se muestra en la Figura 4.19 la cual es la solución esperada.

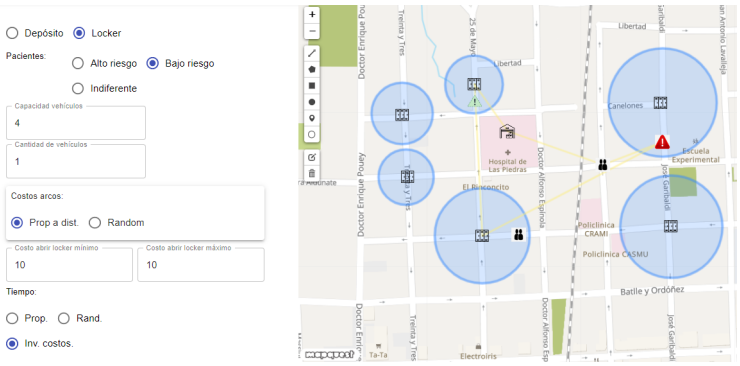

Figura 4.19: Solución de GLPK al caso medio 3 para la validación de la función objetivo por pesos y por normalización. Se corresponde con la solución esperada.

- **Caso medio 4**: Igual al caso anterior pero ahora con 2 vehículos. La solución encontrada por GLPK se muestra en la Figura 4.20 la cual es la solución esperada.

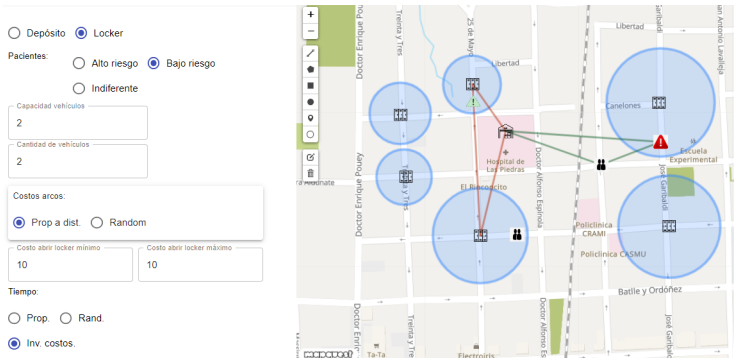

Figura 4.20: Solución de GLPK al caso medio 4 para la validación de la función objetivo por pesos y por normalización. Se corresponde con la solución esperada.

Se prueba el modelo en distintas situaciones: primero con los casos base, y luego variando la cantidad de vehículos en escenarios con alta densidad de pacientes y otros con alta densidad de lockers. En todos los casos el comportamiento es correcto.

4.2.3. Test de esfuerzo

Se pretende evaluar la capacidad de GLPK para resolver instancias grandes del problema. Se utiliza el modelo con la función por normalización especificada en la validación de una única función objetivo de la Sección 4.2.2

Se ejecutan los casos de prueba detallados a continuación en un equipo con 16 GB de RAM, un procesador Core i7-8650U 1.90 GHz, y sistema operativo Windows 10 de 64 bits.

Se comienza de un caso base con 3 pacientes, 3 lockers y 3 vehículos, y se analiza el tiempo al variar la cantidad de pacientes. Como se muestra en la Tabla 4.2 el tiempo de ejecución aumenta exponencialmente al aumentar la cantidad de pacientes.

#Pacientes	**Segundos**
3	0.1
6	4.6
9	+600

Tabla 4.2: Resultado ejecución caso aumento de cantidad de pacientes.

Para el mismo caso base, ahora se varía la cantidad de lockers. Como se muestra en la Tabla 4.3 la cantidad de lockers aumenta de seis a nueve (menos del doble) y el tiempo aumenta al doble para esos casos.

#Lockers	**Segundos**
3	0.1
6	0.2
9	0.4

Tabla 4.3: Resultado ejecución caso aumento cantidad de lockers.

A continuación se modifica la cantidad de vehículos a partir del caso base. Como se muestra en la Tabla 4.4 la cantidad de vehículos aumenta de seis a nueve (menos del doble) y el tiempo aumenta casi al triple para esos casos.

#Vehículos	**Segundos**
3	0.1
6	1.1
9	3.0

Tabla 4.4: Resultado ejecución caso aumento cantidad de vehículos

Además, se registran los tiempos de casos variados en el tamaño de la instancia, donde se encuentra el límite a partir de 7 nodos, y vehículos, donde el tiempo de procesamiento supera los 10 minutos, como se muestra en la Tabla 4.5.

#Lockers	#Pacientes	#Vehículos	Segundos
6	6	1	7.8
6	6	3	26.9
6	6	6	176.9
7	7	7	+600

Tabla 4.5: Resultado aumento de lockers, pacientes y vehículos de forma conjunta

En conclusión, el aumento de cantidad de pacientes, cantidad de lockers y cantidad de vehículos tanto de forma individual como de forma conjunta lleva a un aumento exponencial del tiempo de ejecución.

Capítulo 5

Solución del problema

En este capítulo se describe en primer lugar la solución al problema original descrito en la Sección 4.1 propuesta por Veenstra et al. (2018a). Posteriormente, se describe la solución al problema de este trabajo, describiendo la representación de la solución y los algoritmos desarrollados para la resolución del problema.

5.1. Solución problema original

La solución al problema original descrito en la Sección 4.1 es la implementación de un algoritmo VNS híbrido que se presenta en Algoritmo 13 Heurística híbrida.
La idea general de la heurística es, iterativamente, modificar el conjunto de lockers abiertos, actualizar las rutas adecuadamente y mejorarlas a través de un VNS. El criterio de aceptación de la nueva solución está basado en Simulated Annealing.

Algoritmo 13 Heurística híbrida

```
s ← construir solución inicial
repeat
    s' ← s
    modificar conjunto de lockers abiertos en s'
    actualizar rutas en s'
    aplicar VNS a rutas en s'
    if Accept(s', s) then
        s ← s'
    end if
until condición de parada
return s_best
```

La solución inicial es construida con un conjunto vacío de rutas de lockers,

y un conjunto vacío de rutas de pacientes. Cada locker es abierto con cierta probabilidad y se inserta en la ruta en la mejor posición, es decir, donde menos incrementa la función objetivo. Luego, cada paciente que no se encuentra en el rango de cubrimiento de ningún locker, se inserta en la ruta de pacientes en la mejor posición.

Se presentan 3 acciones que modifican el conjunto de lockers abiertos: abrir, cerrar o mover un locker.
Para abrir un locker se presentan 3 métodos: aleatorio, por reducción de costo, o basado en la cantidad de pacientes cubiertos.
La acción de cerrar un locker es análoga a la de abrir.
Mover un locker implica seleccionar un locker abierto que debe ser cerrado, y un locker cerrado que se debe abrir, que debe estar en el rango de cubrimiento del locker a cerrar. Si no existe un locker en el rango de cubrimiento, se selecciona un locker al azar.

Luego de modificarse el conjunto de lockers, las rutas son actualizadas. Esto es, si un locker es cerrado, se quita de la ruta de lockers, y además, todos los pacientes de su rango de cubrimiento, si no son cubiertos por otro locker, son agregados a la ruta de pacientes.
Si se abre un locker, se inserta en la ruta de lockers en la mejor posición, y todos los pacientes que estén en el rango de cubrimiento son quitados de la ruta de pacientes.

Luego se aplica el Algoritmo 14 VNS para mejorar las rutas:

Algoritmo 14 VNS

```
R : conjunto de rutas
improveOverall ← 1
while improveOverall = 1 do
    improveOverall ← 0
    for each k ∈ 1.,6 do
        improve ← 1
        while improve = 1 do
            improve ← 0
            R' ← R
            R' ← Operator_k(R')
            if Cost(R) > Cost(R') then
                improve ← 1
                improveOverall ← 1
                R ← R'
            end if
        end while
    end for
end while
```

El Algoritmo 14 VNS comienza con un conjunto de rutas que pueden ser las de lockers o de pacientes. Aplica un operador de búsqueda local hasta que no se mejora la solución, y se pasa al siguiente operador. Si se encuentra una mejora en algún operador, el procedimiento se reinicia, si no se termina.
Los 6 operadores de búsqueda local se usan en el siguiente orden: 1-0-Exchange, 1-1-Exchange, Intra-2-Opt, Inter-2-Opt, Intra-3-Opt, Inter-3-Opt.2.
El operador 1-0-Exchange elimina un nodo y lo reinserta en su mejor posición.
El operador 1-1-Exchange se selecciona un nodo aleatoriamente y es intercambiado por el nodo que produzca la mayor disminución del valor de la función objetivo.
El operador Intra-2-Opt se aplica a cada ruta individualmente. Para cada combinación de arcos de la ruta, se calcula el valor de la función objetivo al eliminar los arcos y reconectar los caminos resultantes. Si es posible disminuir el valor, se aplica al par de arcos que más disminuya el valor de la función objetivo.
El operador Inter-2-Opt es similar al Intra-2-Opt pero con arcos de diferentes rutas.
El operador Intra-3-Opt es similar al Intra-2-Opt pero con 3 arcos en lugar de 2.
El operador Inter-3-Opt.2 deriva del Intra-3-Opt y calcula el valor objetivo al remover una ruta parcial y reinsertarla en la mejor posición en otra ruta. Si se logra disminuir el valor objetivo, se aplica con los arcos que más logran disminuirlo.

En cada iteración, una nueva solución es aceptada si el valor objetivo disminuye respecto a la solución actual. Si el valor objetivo no disminuye, la aceptación de la solución se basa en el criterio de simulated annealing, por lo que es aceptada con cierta probabilidad.

Se aceptan soluciones no factibles en el caso de que se supere la duración máxima de ruta. En estos casos, se penaliza la solución de acuerdo al tiempo que se excede de la duración máxima permitida. Al finalizar la heurística, se verifica la factibilidad de la solución.

5.2. Solución a la extensión del problema

Se propone implementar un algoritmo VNS multiobjetivo, en particular el VNS multiobjetivo general que se presenta en Duarte et al. (2015) definido en este trabajo en la Sección 2.3.1.
La decisión de resolver el problema utilizando Algoritmo 7 VNS multiobjetivo general se basa en que la solución al problema original utiliza VNS para determinar las rutas óptimas, utilizando ciertos operadores conocidos fáciles de adaptar a los utilizados en el MO-VNS. Como se puede observar, el Algoritmo 11 Neighborhood Change Step multiobjetivo y el MO-ObjectiveChange presentado en la Línea 10 del Algoritmo 9 MO-VND son independientes del problema, solo es necesario implementar los Algoritmo 10 VND-i, y el Algoritmo 8 Shake

multiobjetivo. Para los cuales se necesita definir las vecindades. Una vecindad normalmente se construye a partir de un punto de eficiencia y un movimiento. Estas vecindades dependen de como se define una solución. Por ejemplo, una forma de implementar N_k para el Shake consiste en determinar un movimiento y aplicarlo a una solución k-veces de forma consecutiva. Mientras que para implementar las vecindades del VND-i se podría implementar cada vecindad con un movimiento distinto. Además de las vecindades, hay que implementar el algoritmo de inicialización de una solución, así como también decidir si permitir o no soluciones no factibles, y en caso de permitirlas, determinar el procedimiento para su manejo (penalización o corrección).

5.2.1. Diseño del algoritmo

A continuación se presenta el diseño de los distintos aspectos a considerar para implementar el VNS multiobjetivo definido anteriormente en la Sección 2.3.1 para el problema definido en la Sección 4.2.

Representación de la solución La representación de la solución queda determinada por 3 listas con la siguiente información:

1. Lista de rutas: contiene las rutas, es decir, una lista donde cada elemento es la ruta de un vehículo, representada como una lista de nodos (pacientes y lockers).

2. Lista de lockers: contiene a los lockers, indicando para cada uno si está abierto.

3. Lista de asignación de pacientes a lockers: indica a qué locker está asignado cada paciente si corresponde. Es una lista de pacientes, donde para cada uno se indica el locker al cual está asignado.
   Además, se tiene una lista de lockers con todos los pacientes asignados a él.

Movimientos Para definir las vecindades es necesario determinar los operadores o movimientos a utilizar.
Es importante notar que para cada operador se debe definir el método *All* (utilizado en el Algoritmo 10 VND-i) que retorna el resultado de aplicar dicho operador en todos los componentes de la solución, y el método $NextRandom$ (utilizado en el Algoritmo 8 Shake multiobjetivo) que retorna el resultado de aplicar el operador en un componente aleatorio.
A continuación se describen los movimientos implementados.

Cambio de conjunto de lockers abiertos (ChangeOpenLockers): Basados en los operadores definidos en el paper original presentados en la Sección 5.1, se propone cambiar el conjunto de lockers abiertos.

Algoritmo 15 Obtener todos los elementos de la vecindad del operador de cambio de conjunto de lockers abiertos

```
procedure ChangeOpenLockers_All(x : Solution)
    if AllClosedLockers(x) then
        return AllClosed_All(x)
    else if AllOpenedLockers(x) then
        return AllOpened_All(x)
    else
        return AllClosed_All(x) ∪ AllOpened_All(x)
    end if
end procedure
```

En el Algoritmo 15 Obtener todos los elementos de la vecindad del operador de cambio de conjunto de lockers abiertos se presenta el procedimiento para obtener todos los elementos de la vecindad del operador de cambio de conjunto de lockers abiertos. Se tienen diferentes implementaciones según el estado de los lockers: si están todos abiertos se ejecuta el Algoritmo 16 Cierra todos los lockers abiertos, si están todos cerrados se ejecuta el Algoritmo 17 Abre todos los lockers cerrados y si hay abiertos y cerrados se hace la unión de los algoritmos anteriores.

Algoritmo 16 Cierra todos los lockers abiertos

```
procedure AllOpened_All(x : Solution)
    result ← new Solution()
    for each locker ∈ x.GetOpenedLockers() do
        CloseLocker(x, result, locker)
    end for
    return result
end procedure
```

Algoritmo 17 Abre todos los lockers cerrados

```
procedure AllClosed_All(x : Solution)
    result ← new Solution()
    for each locker ∈ x.GetClosedLockers() do
        OpenLocker(x, result, locker)
    end for
    return result
end procedure
```

Si todos los lockers están cerrados, entonces se intenta abrir cada uno de ellos con el Algoritmo 18 Abre un locker dado. El algoritmo consulta si el locker se puede abrir. Un locker se puede abrir si: existe al menos un paciente en el rango de cubrimiento del locker que no sea de alto riesgo y que no esté asignado a

otro locker (o si está asignado, que dicho locker tenga más pacientes asignados). Además, este método retorna los pacientes que deben asignarse al locker y el paciente que debe desasignarse de su locker actual (si corresponde).
Si el locker se puede abrir, se crea la nueva solución a partir de la actual, y se le realizan las siguientes modificaciones: se abre el locker, se desasigna el paciente (si corresponde), se asignan los pacientes al locker, se remueven los pacientes (que se asignan al locker) de sus antiguas rutas, y se crea una nueva ruta desde el depósito al locker abierto.

Algoritmo 18 Abre un locker dado

```
procedure OpenLocker(x, result : Solution, locker : int)
    canOpen ← CanOpen(x, locker)
    if canOpen then
        newSol ← new Solution(x)
        newSol.OpenLocker(locker)
        newSol.UnassignPacient(canOpen.PatientsToUnassign)
        newSol.AssignPacient(locker, canOpen.PatientsToAssign)
        newSol.RemoveFromRoutes(canOpen.PatientsToAssign)
        newSol.AddNewRoute(Depot + locker + Depot)
        result.Add(newSol)
    end if
end procedure
```

Si todos los lockers están abiertos, entonces se intenta cerrar cada uno de ellos con el Algoritmo 19 Cierra un locker dado. El algoritmo consulta si el locker se puede cerrar. Un locker se puede cerrar si no hay tiene paciente de bajo riesgo asignado.
Si el locker se puede cerrar, se crea la nueva solución a partir de la actual, y se le realizan las siguientes modificaciones: se cierra el locker, se desasignan los pacientes del locker, se quita el locker de su ruta, y se agregan los pacientes desasignados a la ruta en la que estaba el locker, yendo al paciente más cercano.

Algoritmo 19 Cierra un locker dado

```
procedure CloseLocker(x, result : Solution, locker : int)
    canClose ← CanClose(x, locker)
    if canClose then
        newSol ← new Solution(x)
        newSol.CloseLocker(locker)
        newSol.UnassignPacients(locker)
        newSol.RemoveFromRoute(locker)
        newSol.AddToRoute(locker, locker.Patients)

        result.Add(newSol)
    end if
end procedure
```

A continuación se presenta el Algoritmo 20 Obtener un elemento de la vecindad del operador de cambiar conjunto de lockers abiertos. Se tienen diferentes implementaciones según el estado de los lockers: si están todos abiertos, o todos cerrados, o hay abiertos y cerrados.
El método $AllClosed\_NextRandom(x)$ de la línea 3 obtiene todos los lockers cerrados de la solución. Luego selecciona uno al azar e intenta abrirlo. Si lo logra, se devuelve esa nueva solución, y si no se intenta con un nuevo locker.
El método $AllOpened\_NextRandom(x)$ de la línea 5 es análogo pero intentando cerrar un locker.
Finalmente, si la solución tiene lockers abiertos y cerrados, se aplica uno de los dos métodos mencionados anteriormente con equiprobabilidad.

Algoritmo 20 Obtener un elemento de la vecindad del operador de cambiar conjunto de lockers abiertos

```
1:  procedure ChangeOpenLocker_NextRandom(x : Solution)
2:      if AllClosedLockers(x) then
3:          return AllClosed_NextRandom(x)
4:      else if AllOpenedLockers(x) then
5:          return AllOpened_NextRandom(x)
6:      else
7:          if Random() < 0,5 then
8:              return AllClosed_NextRandom(x)
9:          else
10:             return AllOpened_NextRandom(x)
11:         end if
12:     end if
13: end procedure
```

Cambio de asignación de pacientes a lockers (ChangePatientAssigna-

tion): Este movimiento pretende seleccionar pacientes de la solución, y tratar de desasignarlos o asignarlos a un locker.

A continuación se presenta el Algoritmo 21 Obtener todos los elementos de la vecindad del operador de cambio de asignación de pacientes. Notar que los únicos pacientes que pueden sufrir cambio de asignación a locker son los de tipo indiferente. Entonces, si el paciente está asignado a un locker, se asigna al depósito, esto es: se desasigna del locker, se asigna al depósito y se crea la ruta desde el depósito al paciente. También se agrega a la solución el resultado del método *AppendToLockerRoute* de la línea 7 que desasigna al paciente del locker, pero en vez de crear una nueva ruta desde el depósito, se ingresa en la misma ruta que está el locker.
En cambio, si el paciente no está asignado a un locker, se intenta asignarlo como se muestra en el Algoritmo 22 Asigna un paciente dado al primer locker posible.

El Algoritmo 22 Asigna un paciente dado al primer locker posible. Primero obtiene todos los lockers que incluyen en su rango de cubrimiento al paciente. Si el locker elegido está abierto, el método *CanAssignPatientToLocker* en la línea 6 verifica si se puede asignar el paciente teniendo en cuenta la demanda de la ruta a la que pertenece el locker. Si es posible, entonces se confirma la asignación del paciente al locker.
Si el locker está cerrado, entonces se genera una nueva ruta desde el paciente al depósito.

Algoritmo 21 Obtener todos los elementos de la vecindad del operador de cambio de asignación de pacientes

```
1:  procedure ChangePatientAssignation_All(x : Solution)
2:      result ← new Set <Solution>()
3:      patients ← GetIndiferentPatients()
4:      for each p ∈ patients do
5:          if p.AssignedToLocker(x) then
6:              result.Add(AssignToDepot(p,  x))
7:              result.Add(AppendToLockerRoute(p,  x))
8:          else
9:              result.Add(AssignToLocker(x,  p))
10:         end if
11:     end for
12:
13:     return result
14: end procedure
```

Algoritmo 22 Asigna un paciente dado al primer locker posible

```
1:  procedure AssignToLocker(x, Solution, p : Patient)
2:      result ← new Set <Solution>()
3:      patientLockers ← x.GetLockersWithPatientsInRange(p)
4:      for each l ∈ patientLockers do
5:          if x.IsLockerOpened(l) then
6:              if x.CanAssignPacientToLocker(p, l) then
7:                  newSol ← new Solution(x)
8:                  newSol.AssignPatient(p, l))
9:                  result.Add(newSol)
10:             end if
11:         else
12:             result.Add(AssignToDepot(p, x))
13:         end if
14:     end for
15:
16:     return result
17: end procedure
```

Luego, recordemos que el algoritmo *NextRandom* debe definirse para cada movimiento (5.2.1). En este caso es análogo al Algoritmo 21 con las siguientes salvedades:

- Se selecciona un paciente al azar (en lugar de todos)
- Con equiprobabilidad, se ejecuta *AssignToDepot* o *AssignToLockerRoute* (ver línea 6)

Cambio de rutas (ChangeRoute): Este es el tercer movimiento definido para el algoritmo VNS multiobjetivo implementado. Se propone modificar las rutas sin cambiar la asignación de lockers y pacientes. Para eso se pretende separar y unir diferentes rutas como muestra el Algoritmo 23 Obtener todos los elementos de la vecindad para el operador de cambio de rutas:
Primero se obtienen todas las soluciones resultantes de dividir las rutas de la solución actual, y luego se busca recombinar rutas para cada una de esas soluciones obtenidas.
Para separar rutas se ejecuta el Algoritmo 24 Separar rutas el cual toma los lockers abiertos y los pacientes que no están asignados a un locker. Para cada uno, se remueve de la ruta que tiene asignada y se crea una nueva ruta desde el depósito. La combinación de rutas se realiza ejecutando el Algoritmo 25 Combina rutas el cual toma todos los pares de rutas posibles. Si la suma de la demanda de ambas rutas no supera a la capacidad de los vehículos, entonces se pueden unir. El método *LinkRoutes* Línea 7 es el encargado de unir ambas rutas. Se generan dos soluciones: la que resulta de unir la primer ruta al final de la segunda, y la que resulta de unir la segunda ruta al final de la primera.

Algoritmo 23 Obtener todos los elementos de la vecindad para el operador de cambio de rutas

```
1: procedure ChangeRoute(x : Solution)
2:     result ← new Set <Solution>()
3:     separatedRoutesSol ← OpSeparateRoutes(x)
4:     for each sol ∈ separatedRoutesSol do
5:         result.Add(OpCombineRoutes(sol))
6:     end for
7:
8:     result.Add(OpCombineRoutes(x))
9:     return result
10: end procedure
```

Algoritmo 24 Separar rutas

```
1: procedure OpSeparateRoutes(x, Solution)
2:     result ← new Set <Solution>()
3:     nodes ← x.GetOpenedLockers() ∪ x.GetPatientAssignedToDepot()
4:     for each node ∈ nodes do
5:         sol ← new Solution(x)
6:         sol.RemoveFromItsRoute(node)
7:         sol.AddRoute(Depot + node + Depot)
8:         result.Add(sol)
9:     end for
10:
11:    return result
12: end procedure
```

Algoritmo 25 Combina rutas

```
1: procedure OpCombineRoutes(x, Solution)
2:     result ← new Set <Solution>()
3:     routes ← x.GetRoutes()
4:     for each route1 ∈ routes do
5:         for each route2 ∈ routes do
6:             if x.GetRouteDemand(route1) + x.GetRouteDemand(route2) ≤
   Capacity then
7:                 result.Add(LinkRoutes(route1, route2))
8:                 result.Add(LinkRoutes(route2, route1))
9:             end if
10:        end for
11:    end for
12:
13:    return result
14: end procedure
```

A continuación el Algoritmo 26 Obtener un elemento de la vecindad (NextRandom):

Algoritmo 26 Obtener un elemento de la vecindad

```
procedure NextRandom(x : Solution)
    return OpCombineRoutes(OpSeparateRoutes(x))
end procedure
```

Es muy similar al método Algoritmo 23 Obtener todos los elementos de la vecindad para el operador de cambio de rutas pero con las siguientes modificaciones para devolver una solución aleatoria:

- El método Algoritmo 24 Separar rutas aplica sobre un nodo aleatorio en lugar de todos (ver línea 3).
- El método Algoritmo 25 Combina rutas selecciona un par de rutas al azar, y con equiprobabilidad concatena la primera al final de la segunda, o al revés (ver línea 5).

Vecindades para el VND Se tienen tres vecindades. Cada una es generada por cada movimiento mencionado en la sección anterior.
$N1(x) = \{changeOpenedLockers(x)\}$ Resultado de aplicar Algoritmo 15 Obtener todos los elementos de la vecindad del operador de cambio de conjunto de lockers abiertos con x en el espacio de dominio.
$N2(x) = \{changePatientAssignation(x)\}$ Resultado de aplicar Algoritmo 21 Obtener todos los elementos de la vecindad del operador de cambio de asignación de pacientes con x en el espacio de dominio.
$N3(x) = \{changeRoute(x)\}$ Resultado de aplicar Algoritmo 23 Obtener todos los elementos de la vecindad para el operador de cambio de rutas con x en el espacio de dominio.

Vecindades para el Shake Las vecindades para el Shake resultan de aplicar en composición el Algoritmo 15 Obtener todos los elementos de la vecindad del operador de cambio de conjunto de lockers abiertos.
N_k consiste en aplicar k veces $changeOpenLockers(x)$ en composición. Por ejemplo: $N_2 = changeOpenLockers(changeOpenLockers(x))$.

Inicialización de la solución Se divide la inicialización de la solución en 3, donde cada tercio es una buena solución para cada uno de los objetivos.

Algoritmo 27 Procedimiento de inicialización Greedy de la solución.

```
1:  procedure Initialize solutionSize
2:      result ← new Set <Solution>()
3:      for i ← 0 to solutionSize do
4:          sol ← new Solution(x)
5:          if i < solutionSize / 3 then
6:              sol.RandomInitialize(INITIALIZATION_TYPE.LESS_COST)
7:          else if i < 2 * ( solutionSize / 3) then
8:              sol.RandomInitialize(INITIALIZATION_TYPE.LESS_TIME)
9:          else if i < 3 * ( solutionSize / 3) then
10:             sol.RandomInitialize(INITIALIZATION_TYPE.MORE_EQUITY)
11:         end if
12:         result.Add(sol)
13:     end for
14:     return result
15: end procedure
```

Como se puede observar, cada tercio de soluciones se genera utilizando el procedimiento Algoritmo 28 Inicializa de forma aleatoria una solución la cual inicializa la solución según un enumerado que indica que la solución debe ser la de menor costo, menor tiempo o mayor equidad. Para asegurar la equidad lo que se hace es considerar que cada nodo está en una ruta que contiene a ese nodo, se calculan los tiempos de atención de esos nodos, se toma el máximo y luego al armar las rutas se asegura que el tiempo de atención no supere ese máximo.

Algoritmo 28 Inicializa de forma aleatoria una solución

```
1: procedure RandomInitialize(input initializationType)
2:     AssignLowRiskPatientsToSomeLocker()
3:     AssignIndiferentPatientsToLockers()
4:     CreateRoutes(initializationType)
5: end procedure
```

El Algoritmo 28 Inicializa de forma aleatoria una solución. Primero asigna los pacientes de bajo riesgo, luego asigna los pacientes indiferentes a lockers, y por último crea rutas centrándose en el objetivo que se pasa por parámetro.

El Algoritmo 29 Asigna los pacientes de bajo riesgo a algún locker disponible. El algoritmo comienza obteniendo la lista de pacientes de bajo riesgo y la lista de lockers disponibles. Luego, para cada paciente de bajo riesgo, el algoritmo busca un locker que incluya al paciente dentro de su rango de cubrimiento y que aún tenga capacidad para admitir más pacientes. El primer locker que cumple estas condiciones se abre, se asigna el paciente al locker y se asigna el locker al paciente.

Algoritmo 29 Asigna los pacientes de bajo riesgo a algún locker

```
procedure AssignLowRiskPatientsToSomeLocker
    lowRiskPatients ← GetPatients(LOW_RISK)
    lockers ← GetLockers()
    for each patient ∈ lowRiskPatients do
        for each locker ∈ lockers do
            if patient.IsInLockerRange(locker) then
                if locker.PatientsInLocker < vehiclesCapacity then
                    locker.Open()
                    patient.Assign(locker)
                    locker.Assign(patient)
                    break
                end if
            end if
        end for
    end for
end procedure
```

El Algoritmo 30 Asigna los pacientes indiferentes a lockers es similar al anterior pero para pacientes indiferentes.

Algoritmo 30 Asigna los pacientes indiferentes a lockers

```
1:  procedure AssignIndiferentPatientsToLockers
2:      indiferentPatients ← GetPatients(INDIFERENT)
3:      lockers ← GetLockers()
4:      for each patient ∈ indiferentPatients do
5:          for each locker ∈ lockers do
6:              if patient.IsInLockerRange(locker) and random.GetRandom()
    and locker.PatientsInLocker < vehiclesCapacity then
7:                  locker.Open()
8:                  patient.Assign(locker)
9:                  locker.Assign(patient)
10:                 break
11:             end if
12:         end for
13:     end for
14: end procedure
```

El Algoritmo 31 Crea rutas de forma Greedy según el tipo pasado por parámetro inicializa una estructura de datos $nodesToVisit$, la cual indica para cada nodo el costo de agregar dicho nodo a una ruta, si es un paciente, su costo es uno y si es un locker, su costo es la cantidad de pacientes asignados al locker. Además, esta variable sirve para saber que nodos faltan por visitar, aquellos cuyo costo sea 0 es que ya fue visitado. Luego delega la función de crear las

rutas al procedimiento adecuado según el parámetro *initType*, si *initType* es $LESS\_TIME$, se crearán rutas teniendo en cuenta el objetivo de tiempo, de forma análoga para $LESS\_COST$ y $MORE\_EQUITY$.

Algoritmo 31 Crea rutas de forma Greedy según el tipo pasado por parámetro

```
1:  procedure CreateRoutes(initType)
2:      nodesToVisit   ▷ nodos que quedan por visitar y el costo en capacidad de ruta
3:      for each locker ∈ GetLockers() do
4:          nodesToVisit(locker) ← locker.AssignatedPatientsCount()
5:      end for
6:      for each patient ∈ GetPatients() do
7:          nodesToVisit(patient) ← 1
8:      end for
9:      if initType == LESS_TIME or initType == LESS_COST then
10:         LessTimeOrLessCostInit(initType, nodesToVisit)
11:     else
12:         MoreEquityInit(initType, nodesToVisit)
13:     end if
14: end procedure
```

Como se mencionó anteriormente, el Algoritmo 32 Inicializa las rutas de forma Greedy tomando en cuenta el tiempo o el costo, según se indique en el parámetro genera rutas según el objetivo de tiempo o de costo, según el parámetro initType.

Algoritmo 32 Inicializa las rutas de forma Greedy tomando en cuenta el tiempo o el costo, según se indique en el parámetro

```
1:  procedure LessTimeOrLessCostInit(initType, nodesToVisit)
2:      while AreNodesToVisit(nodesToVisit) do
3:          route ← empty list of integers
4:          currentNode ← Depot
5:          vehicleCapacityAvailable ← VehiclesCapacity
6:          auxList ← nodesToVisit.GetNodesThatFitInTheRoute()
7:          while auxList.HasElements() do
8:              nextNode ← GetNearestNode(initType, currentNode, auxList)
9:              UpdateVariables()
10:             auxList ← nodesToVisit.GetNodesThatFitInTheRoute()
11:         end while
12:         Routes.Add(route)
13:     end while
14: end procedure
```

Algoritmo 33 Update Variables

```
1: procedure UpdateVariables
2:     vehicleCapacityAvailable ← vehicleCapacityAvailable - nodesToVisit[nextNode]
3:     nodesToVisit[nextNode] ← 0
4:     route.Add(nextNode)
5:     currentNode ← nextNode
6: end procedure
```

El Algoritmo 32 Inicializa las rutas de forma Greedy tomando en cuenta el tiempo o el costo, según se indique en el parámetro se basa en crear rutas de forma greedy obteniendo el nodo más cercano al anterior agregado en la ruta según la función $GetNearestNode(initType, currentNode, auxList)$ de la Línea 8 la cual obtiene el nodo más cercano de auxList al currentNode según el tipo de inicialización.

La función UpdateVariables (Línea 9) se define en el Algoritmo 33 Update Variables.

Por otro lado, nodesToVisit.GetNodesThatFitInTheRoute() Línea 10 devuelve los nodos que se pueden insertar en la ruta, para esto se fija en que el valor de nodeToVisit sea menor a la capacidad disponible de vehículo actual.

Algoritmo 34 Inicializa las rutas de forma Greedy según la equidad

```
1: procedure MoreEquityInit(initType, nodesToVisit)
2:     maxAttendanceTime ← GetMaximumAttendanceTimeFromDepotAtOneStep(nodesToVisit)
3:     while AreNodesToVisit(nodesToVisit) do
4:         route ← empty list
5:         currentNode ← DepotId
6:         vehicleCapacityAvailable ← VehiclesCapacity
7:         routeAttendanceTime ← 0
8:         auxList ← nodesToVisit.GetNodesThatFitInTheRoute()
9:         while auxList.HasElements() do
10:            nextNode ← GetNearestNode(initType, currentNode, auxList)
11:            if route.AttendanceTime(nextNode) ≤ maxAttendanceTime then
12:                UpdateVariables()
13:                auxList ← nodesToVisit.GetNodesThatFitInTheRoute()
14:            else
15:                break
16:            end if
17:        end while
18:        Routes.Add(route)
19:    end while
20: end procedure
```

El Algoritmo 34 Inicializa las rutas de forma Greedy según la equidad es análogo al Algoritmo 32 Inicializa las rutas de forma Greedy tomando en cuenta el tiempo o el costo, según se indique en el parámetro dónde se tiene en cuenta que el tiempo de atención del último nodo de una ruta no supera el tiempo máximo de atención obtenido por la función GetMaximumAttendanceTimeFromDepotAtOneStep (Línea 2) la cual calcula el tiempo de atención desde el depósito hasta cada nodo y se queda con el máximo.

La función UpdateVariables (Línea 12) se define en el Algoritmo 33 Update Variables.

Capítulo 6

Implementación

En esta sección se describe la implementación de la solución.

El principal objetivo es construir una herramienta que facilite la creación de instancias del problema propuesto en la Sección 4.2 y ejecutar el algoritmo MO-VNS propuesto en la Sección 5.2 desarrollado sobre dichas instancias.

La aplicación desarrollada sigue un patrón de arquitectura Cliente-Servidor, que es un enfoque común en el diseño de sistemas informáticos distribuidos (Tanenbaum & Van Steen, 2007). En este enfoque, la aplicación cliente (en este caso, la aplicación web desarrollada en ReactJS («React. The library for web and native user interfaces», s.f.)) envía solicitudes a un servidor (una Web API implementada en C# .NET 5 («.NET Home», s.f.)) para obtener acceso a recursos o servicios. La decisión del stack tecnológico fue tomada teniendo en cuenta que son las tecnologías utilizadas diariamente en nuestro ámbito laboral, por lo que aprovechamos dicho expertise y evitamos incertidumbre de aprender una nueva tecnología.

Además, la aplicación también utiliza el patrón de arquitectura en capas, que divide la aplicación en diferentes capas o niveles lógicos para separar responsabilidades y mejorar la modularidad. En este caso, la capa de presentación (la aplicación web en ReactJS) se encarga de la interfaz de usuario y la interacción del usuario, la capa de servicios (la API implementada en Web API C# .NET 5) proporciona servicios para procesar y recuperar datos, y la capa de acceso a datos (base de datos SQL Server («SQL Server Overview», s.f.) con acceso a datos con Entity Framework («Entity Framework Documentation Hub», s.f.) se encarga del almacenamiento y recuperación de los datos de la aplicación. Además, se cuenta con una capa que tiene el core para resolución de problemas multiobjetivo. La elección de estas tecnologías se realizó con base en poseer experiencia en dichas tecnologías.

Este enfoque de arquitectura en capas y Cliente-Servidor mejora la mantenibilidad, escalabilidad y la eficiencia del desarrollo de la aplicación, permitiendo que cada capa de la aplicación se enfoque en su tarea específica y facilitando la integración de nuevas funcionalidades en el sistema.

Este capítulo se estructura de la siguiente forma: comienza describiendo la

arquitectura general de la solución donde se muestran los distintos componentes de la solución y cómo interactúan entre ellos. Posteriormente, se describe cada componente en particular, detallando las funcionalidades implementadas, tecnologías utilizadas y detalles de implementación.

6.1. Arquitectura general

Como se muestra en la Figura 6.1 la solución consta de distintos componentes: aplicación web, backend y base de datos. La aplicación web se comunica con una web API del backend mediante solicitudes HTTP. El backend se comunica con la base de datos mediante Entity Framework. El backend, a su vez, está constituido por varios componentes: controladores, repositorios, capa de acceso a datos y capa de lógica de negocios.

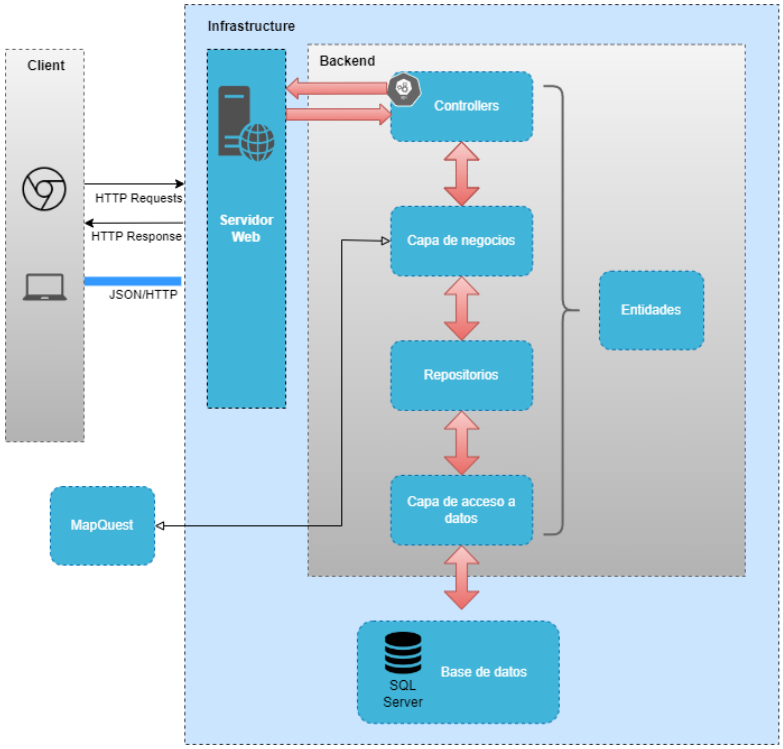

Figura 6.1: Arquitectura de la solución.

6.2. Aplicación web

La aplicación web en la Figura 6.2 se desarrolla en ReactJS («React. The library for web and native user interfaces», s.f.) utilizando Material UI para la

interfaz gráfica (**mui**). Además, se hace uso de la librería MapQuest («MapQuest Developer Documentation», s.f.) con Leaflet JS para la creación y manipulación del mapa. Para las tablas se utilizó los DataGrid de («React Data Grid Component Overview», s.f.) el cual permite filtrar y ordenar por columnas, además de paginación para los datos. Estas tecnologías fueron seleccionadas debido a que ya se tenía conocimiento previo y no se necesitó tiempo de capacitación. Como se muestra en la Figura 6.3 la aplicación se divide en paneles los cuales contienen las distintas funcionalidades.

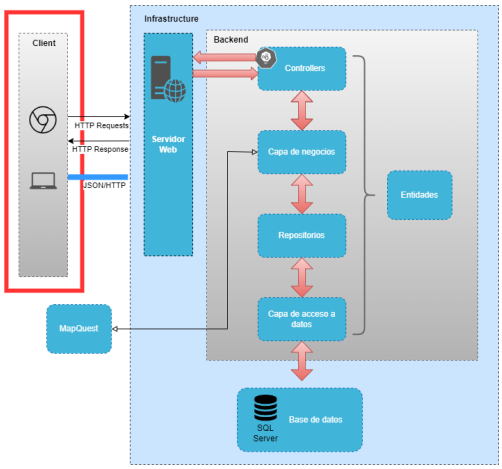

Figura 6.2: Aplicación web.

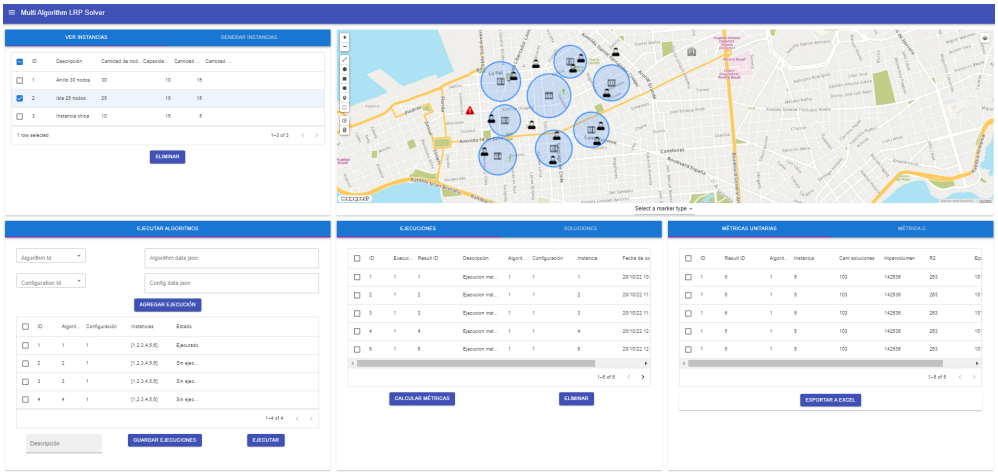

Figura 6.3: Pantalla completa con todos los paneles.

- Panel de mapa (Figura 6.5): Permite visualizar las instancias en el mapa,

así como agregar los distintos tipos de nodos al mapa (lockers y sus áreas de cubrimiento, depósito y pacientes). También se permite ver las rutas entre nodos de una solución en particular, siguiendo las calles o una línea directa entre nodos dependiendo la distancia que se haya utilizado. Se utiliza la siguiente notación para los iconos de los nodos:

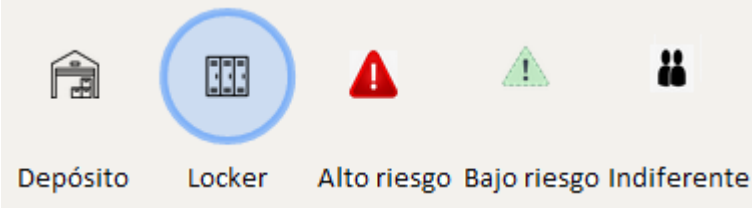

Figura 6.4: Notación iconos para los nodos

- Panel de instancias (Figura 6.5): Este panel está separado en dos pestañas, en la primera se permite visualizar y eliminar instancias (Figura 6.5), en la segunda se permite generar instancias aleatorias (Figura 6.6). La funcionalidad de visualizar instancias muestra en una tabla todas las instancias creadas en el sistema, la tabla muestra el ID de la instancia en base de datos, la descripción, cantidad de nodos, capacidad de vehículos, cantidad de lockers y cantidad de pacientes. Se permite borrar las instancias, seleccionándolas y presionando el botón Eliminar. La funcionalidad de generar instancias permite generar instancias aleatorias especificando los siguientes datos:

  - Vehículos: Se permite ingresar cantidad de vehículos y capacidad de vehículos.
  - Tipo de distancia entre nodos: Euclidiana o siguiendo las calles utilizando MapQuest.
  - Costos de arcos: Igual al valor de la distancia entre los nodos o aleatorio, en cuyo caso se debe especificar un valor mínimo y un valor máximo entre los cuales se seleccionará de forma aleatoria uniforme el costo del arco.
  - Costos de abrir lockers: se debe especificar un valor mínimo y un valor máximo entre los cuales se seleccionará de forma aleatoria uniforme el costo del locker.
  - Tiempo entre nodos: Se puede seleccionar para que sea igual al valor de los costos de los arcos, inversamente proporcional al costo de arcos o aleatorio entre un rango de valores.
  - Servicio: Se permite ingresar el valor de servicio por tipo de nodo (locker o paciente) o aleatorio entre un rango de valores.
  - Penalización: De forma análoga al servicio.

- Descripción: Permite agregar una descripción a la instancia a crear.

Los nodos, posición y distancia entre nodos se especifican en el panel del mapa. Al crear una instancia se permite utilizar distancia entre nodos por calles (utilizando MapQuest) o distancia euclidiana.

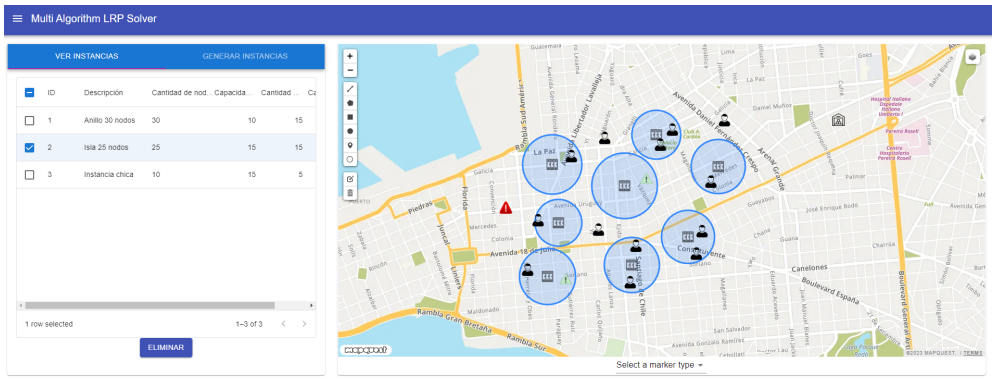

Figura 6.5: Panel de mapas e instancias

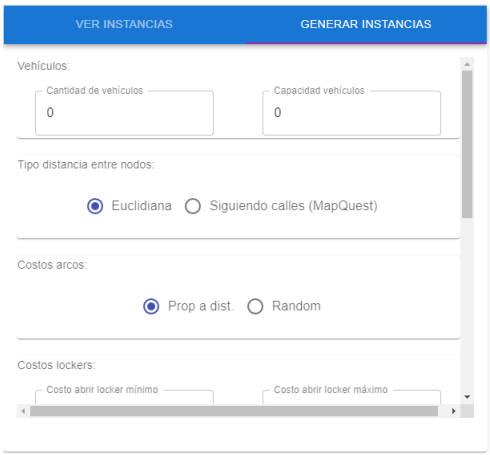

Figura 6.6: Pestaña generar instancias

- Panel ejecutar algoritmos (Figura 6.7): Este panel permite la ejecución de algoritmos, con una determinada configuración, sobre un conjunto de instancias. Para esto, se selecciona un algoritmo y una configuración desde las listas desplegables correspondientes, se seleccionan en el panel de instancias las instancias deseadas y se hace clic en el botón Agregar ejecución. Se pueden planificar varias ejecuciones, las cuales se muestran en una tabla que muestra el algoritmo, la configuración utilizada, el conjunto de

instancias sobre las cuales se ejecutará el algoritmo y el estado (ejecutado, sin ejecutar, guardada). Para llevar a cabo estas ejecuciones se debe seleccionar las filas deseadas y presionar el botón Ejecutar. Además, se permite guardar las ejecuciones en estado ejecutado, seleccionándolas de la tabla, agregando una descripción y presionando el botón Guardar ejecuciones. Al seleccionar una ejecución en estado guardada se mostrarán las mismas en el panel de Ejecuciones y soluciones en la pestaña de ejecuciones (Figura 6.8).

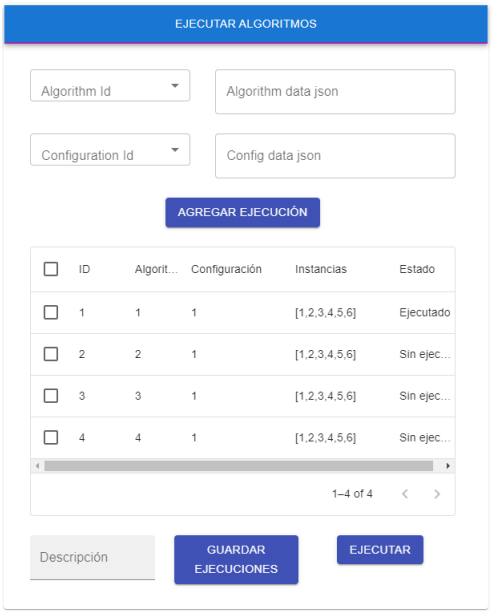

Figura 6.7: Panel de ejecutar algoritmos

- Panel de Ejecuciones y soluciones (Figura 6.8): Al ejecutar un algoritmo sobre una instancia con una configuración dada, se genera una ejecución, la cual está formada por un conjunto de soluciones, donde cada solución consta de la asignación de pacientes a lockers, el estado de los lockers (abierto, cerrado), las rutas entre los distintos nodos y el valor de las tres funciones objetivo. Este panel consta de dos pestañas, una pestaña de ejecuciones (Figura 6.8) y una de soluciones (Figura 6.9). En la de ejecuciones se muestran las distintas ejecuciones realizadas, las cuales constan de un ID de ejecución que identifica el grupo de ejecuciones que se mandó a ejecutar en lote (esto es, cuando en el panel de Ejecutar algoritmos se ejecuta un algoritmo sobre distintas instancias todas estas ejecuciones forman parte de un grupo), un ID de resultado que identifica cada ejecución en particular, una descripción, el ID del algoritmo utilizado, el ID de la configuración utilizada, el ID de la instancia sobre la que se realizó

la ejecución, la fecha de comienzo y el tiempo de ejecución. Además, esta pestaña tiene un botón para calcular las métricas unitarias de las ejecuciones seleccionadas, seleccionando las ejecuciones en la tabla, el resultado de las métricas se muestra en el panel de métricas en la pestaña de Métricas unitarias (Figura 6.11). Se permite también eliminar ejecuciones, seleccionándolas desde la tabla y presionando el botón eliminar, al eliminar las ejecuciones también se eliminan las soluciones asociadas a esas ejecuciones. La pestaña de soluciones muestra todas las soluciones del sistema o las soluciones asociadas a las ejecuciones seleccionadas en la pestaña de ejecuciones. Se muestra una tabla que indica el ID de ejecución a la que pertenece la solución, el ID de la solución en sí, el valor de las funciones objetivos y si la misma es factible o no. Además, brinda la posibilidad de ver las rutas entre nodos en el mapa, seleccionando la solución y haciendo clic en el botón Ver en mapa. La solución se muestra en el mapa como se ve en la Figura 6.10, cada ruta se muestra de un color diferente, los pacientes que no están en ninguna ruta es porque están asignados al locker más próximo. Los lockers abiertos son aquellos que están en una ruta, es decir, hay una línea en el mapa que pasa por ellos.

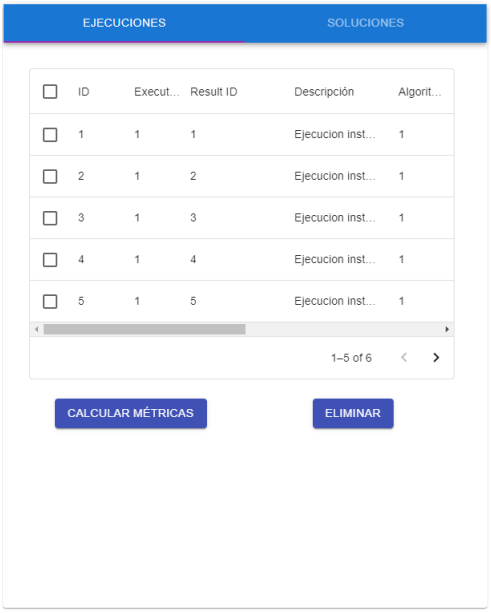

Figura 6.8: Pestaña de ejecuciones

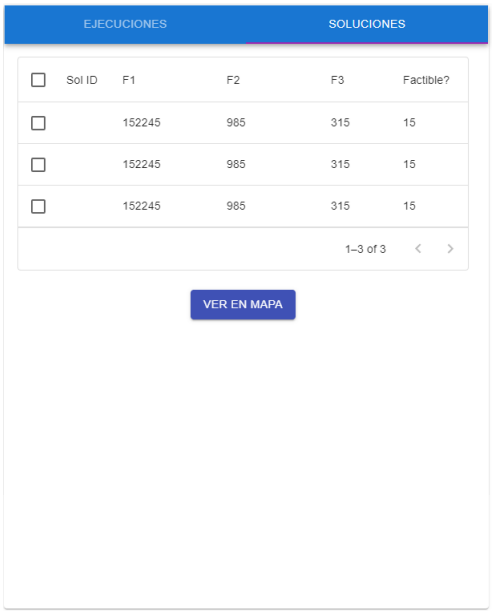

Figura 6.9: Pestaña de soluciones

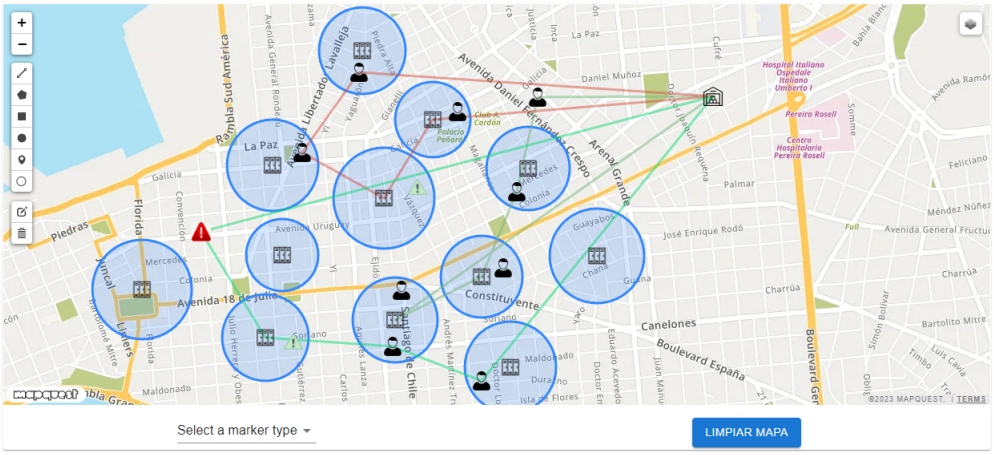

Figura 6.10: Ver solución euclidiana

- Panel de métricas (Figura 6.11): Este panel consta de dos pestañas, una pestaña de métricas unitarias y una pestaña para la métrica C. La primera pestaña muestra el resultado de las métricas para las ejecuciones seleccionadas en el panel de Ejecuciones y soluciones. Se muestra una tabla que consta de un ID de ejecución, el ID del algoritmo de la ejecución, el ID de la instancia sobre la cual se ejecutó el algoritmo, la cantidad de soluciones

de esa ejecución, el valor del hipervolumen, el valor de R2 y el valor de la medida épsilon. Además, se permite exportar a Excel. La pestaña de métrica C es simplemente un botón de exportar a Excel, donde se calculará la métrica C para cada par de algoritmos, calculando por instancia para las ejecuciones seleccionadas en el panel de Ejecuciones y soluciones.

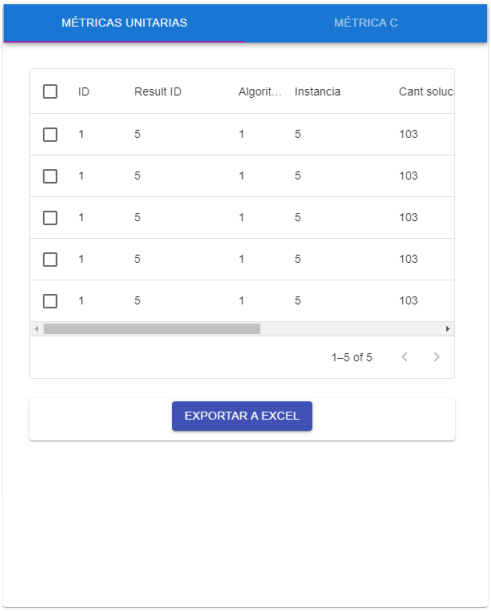

Figura 6.11: Pestaña de métricas unitarias

6.3. Backend

En esta sección se describe el backend (Figura 6.12) de la aplicación. Describiendo la funcionalidad de cada uno de sus componentes.

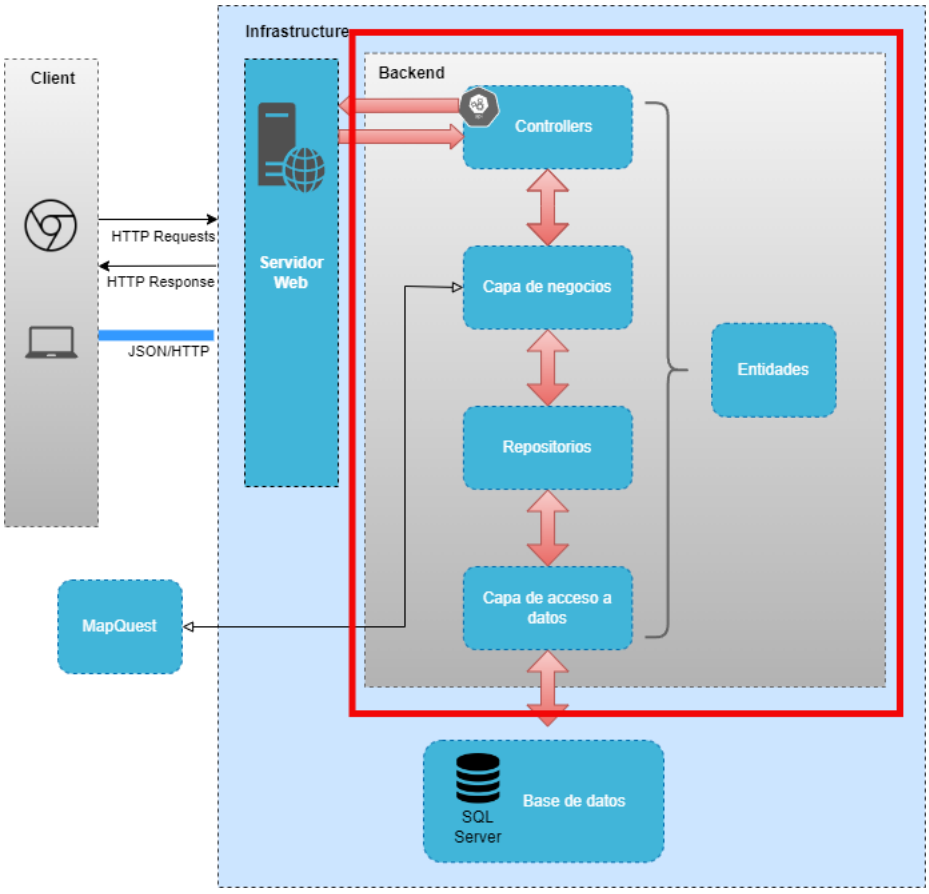

Figura 6.12: Backend.

El backend (Figura 6.12) está constituido por los siguientes módulos:

Controladores (Figura 6.13): Se encargan de manejar las solicitudes HTTP entrantes y de devolver una respuesta apropiada al cliente. Interpretan los datos de la solicitud, realizan cualquier operación necesaria en el modelo de datos y devuelven una respuesta al cliente. Diseñados para manejar una determinada ruta o conjunto de rutas dentro de la API. Para esta solución se tienen los siguientes controladores:

- ExecutionsController: Endpoints para el manejo de ejecuciones. Permite ejecutar algoritmos, ver las ejecuciones hechas, etc.

- InstancesController: Endpoints para el manejo de instancias. Alta, baja y mantenimiento de instancias.

- AlgorithmsController: Endpoints para el manejo de algoritmos. Alta, baja y modificación de los mismos.

- SolutionsController: Endpoints para el manejo de soluciones. Obtener soluciones, generar reportes, calcular métricas, etc.

- ConfigurationsController: Endpoints para el alta, baja y modificación de configuraciones.

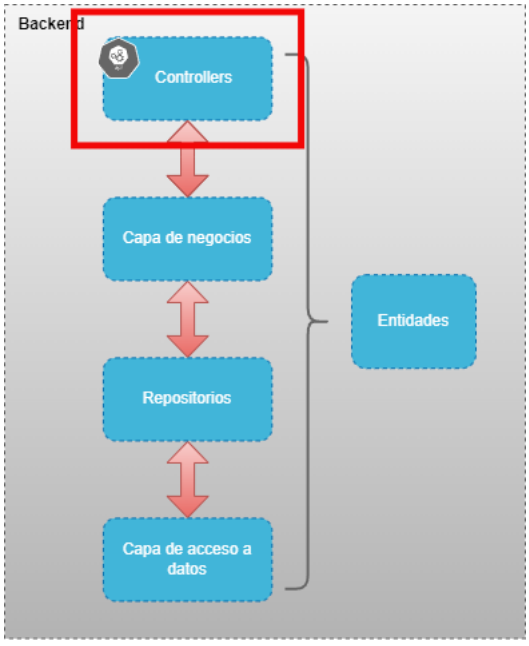

Figura 6.13: Controladores.

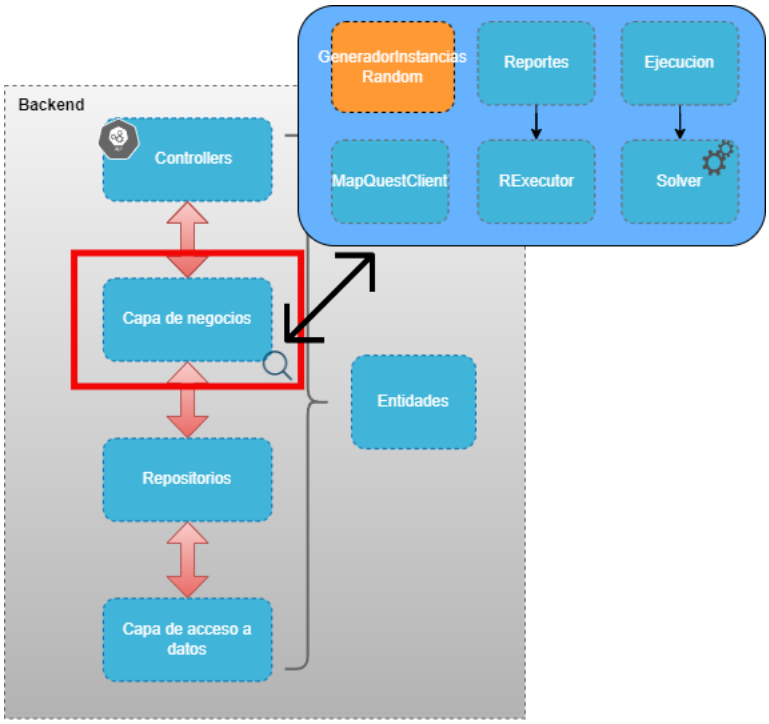

Figura 6.14: Diagrama capa de negocios.

Lógica de negocios (Figura 6.14): La capa de negocio se divide en 5 módulos principales:

- GeneradorInstanciasRandom: Se encarga de generar instancias aleatorias a partir de rangos de parámetros (costo de servicio, costo de abrir, locker, matriz de costos, matriz de tiempo, penalización, entre otros).
- MapQuestClient: Este módulo contiene la lógica para la comunicación con la API de MapQuest.
- Reportes: Este módulo provee la lógica necesaria para generar reportes que son útiles para comparar o visualizar los resultados.
- RExecutor: Módulo que tiene la lógica encargada de invocar las funciones en R necesarias, como por ejemplo el hipervolumen, épsilon, r2.
- Ejecución: Módulo encargado de obtener y preparar los datos (instancias, algoritmo a ejecutar, configuración, etc.) y enviárselo al solver para ejecutar el algoritmo de forma apropiada sobre las instancias correspondientes.
- Solver: Es el que ejecuta cada algoritmo en particular.

Solver:

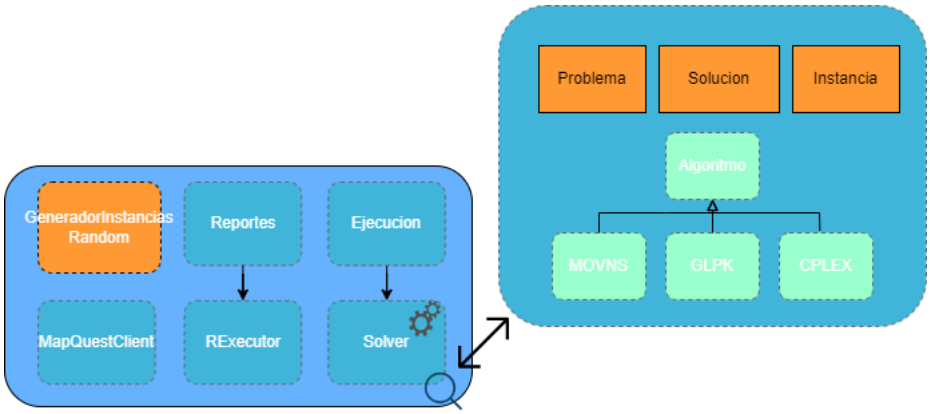

Figura 6.15: Componentes principales módulo solver.

Cómo se muestra en la Figura 6.15, este módulo tiene 4 clases principales:

- Problema: Clase que define el problema de optimización. Permite definir las funciones objetivo, las restricciones, parámetros y variables utilizadas.

- Solución: Esta clase representa una solución al problema. En este caso, una solución consiste en un conjunto de rutas, asignación de pacientes a lockers y estado de los lockers (abierto, cerrado). Además, contiene la lógica para manipulación de soluciones, como obtener el nodo más cerca a otro en cuanto a tiempo, costo o distancia, obtener los lockers que están abiertos, etc.

- Instancia: Esta clase contiene los datos del problema con los cuales ejecutar el algoritmo. Así como lógica para manipular esas instancias.

- Algoritmo: Esta clase representa el algoritmo en sí, de la cual derivan 3 clases: MO-VNS, GLPK y CPLEX. En el caso de MO-VNS se implementa el algoritmo en sí, mientras que para GLPK y CPLEX se encargan de preparar los datos, archivos, y todo lo necesario para invocar GLPK y ampl respectivamente.

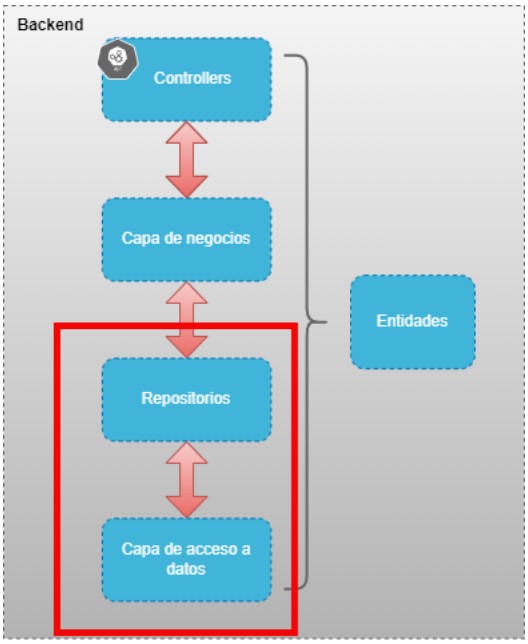

Figura 6.16: Repositorios y capa de acceso a datos

Repositorios (Figura 6.16): Los repositorios se utilizan para encapsular la lógica de acceso a datos y proporcionar una interfaz para interactuar con la capa de persistencia de datos de la aplicación. Esto significa que los repositorios se encargan de realizar operaciones de lectura y escritura en la base de datos o en cualquier otro sistema de almacenamiento de datos que se esté utilizando en la aplicación.

Capa de acceso a datos (Figura 6.16): Capa de acceso a datos que implementa el acceso a los datos mediante Entity Framework.

6.4. Base de datos

En esta sección se describe el componente de base de datos (Figura 6.17) de la aplicación. Detallando el modelo entidad-relación propuesto para la persistencia de datos.

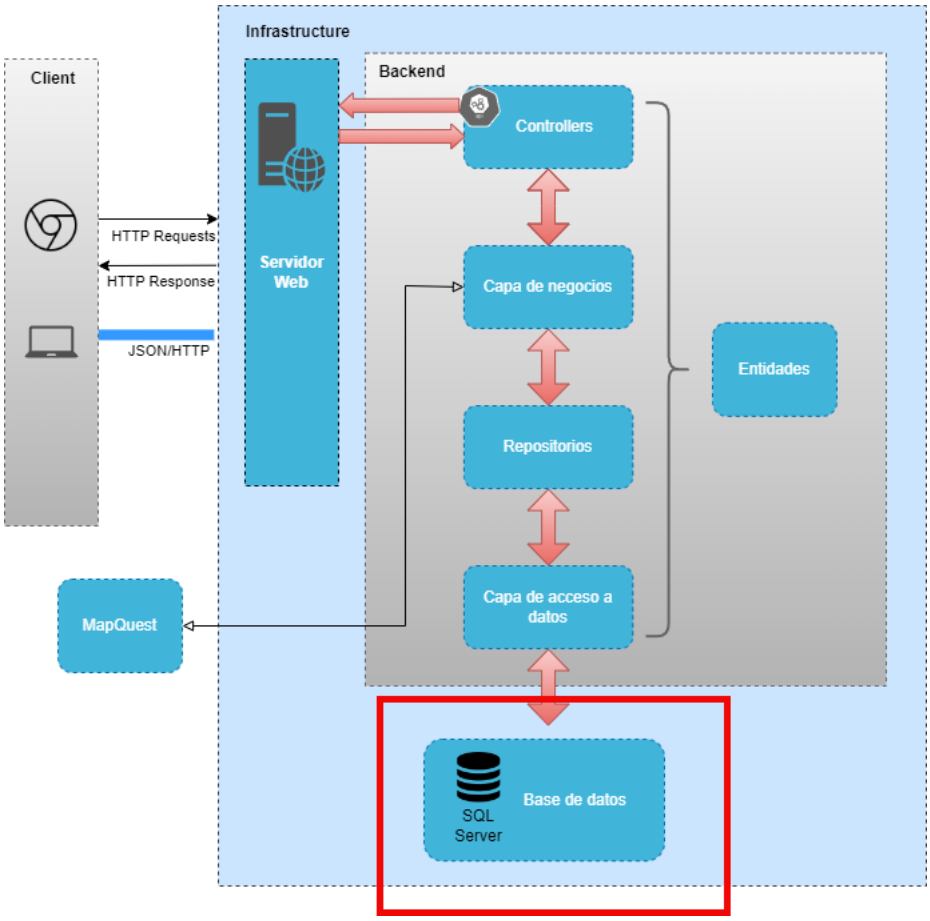

Figura 6.17: Base de datos

Se utiliza Microsoft SQL Server 2019 Developer Edition («SQL Server Downloads», s.f.) como manejador de base de datos, debido a que es gratuito para la cantidad de datos que necesitamos procesar, además de contar con experiencia en su uso, por lo que no se necesitó de tiempo de capacitación. En la Figura 6.18 se presenta el modelo entidad relación, para cumplir con los requerimientos de la solución.

A continuación se explican las distintas partes del modelo:

La relación Ejecuta representa la ejecución de un algoritmo con una configuración, sobre una instancia dada y un resultado de ejecución. De las configuraciones se conoce su nombre y una descripción para la misma. De los algoritmos se tiene su nombre y descripción, además de haber distintos tipos como GLPK, CPLEX y MO-VNS. De las instancias se conoce la capacidad de los vehículos,

la matriz de distancias entre nodos, la matriz de costo entre nodos y la matriz de tiempo entre nodos, estas matrices son un JSON que representan la matriz considerando los ID de los nodos. Además, cada instancia tiene un conjunto de nodos asociados. De los nodos se conoce la latitud, la longitud, el tiempo de servicio y el factor de penalización. Hay tres tipos de nodos: locker, paciente y depósito, de los lockers se conoce además su radio de cubrimiento y el costo de apertura, mientras que de los pacientes se conoce su tipo (alto riesgo, bajo riesgo o indiferente). De cada resultado de ejecución se conoce su tiempo de ejecución y la fecha de comienzo de la ejecución. Además, cada ejecución tiene un conjunto de soluciones de las cuales se sabe los valores de las funciones objetivo. Por otro lado, las soluciones están formadas por un conjunto de rutas, las cuales tienen 2 o más nodos representado por la relación NodosRuta donde el tipo asociativo OrdenRuta especifica el orden del nodo en la ruta. Además, las soluciones tienen un conjunto de lockers asociados representado por la asociación ResultadoLocker y por el tipo asociativo ResultadoLockerDatos, el cual especifica si el nodo de esa solución está abierto o no, además de los pacientes asignados a ese locker de solución.

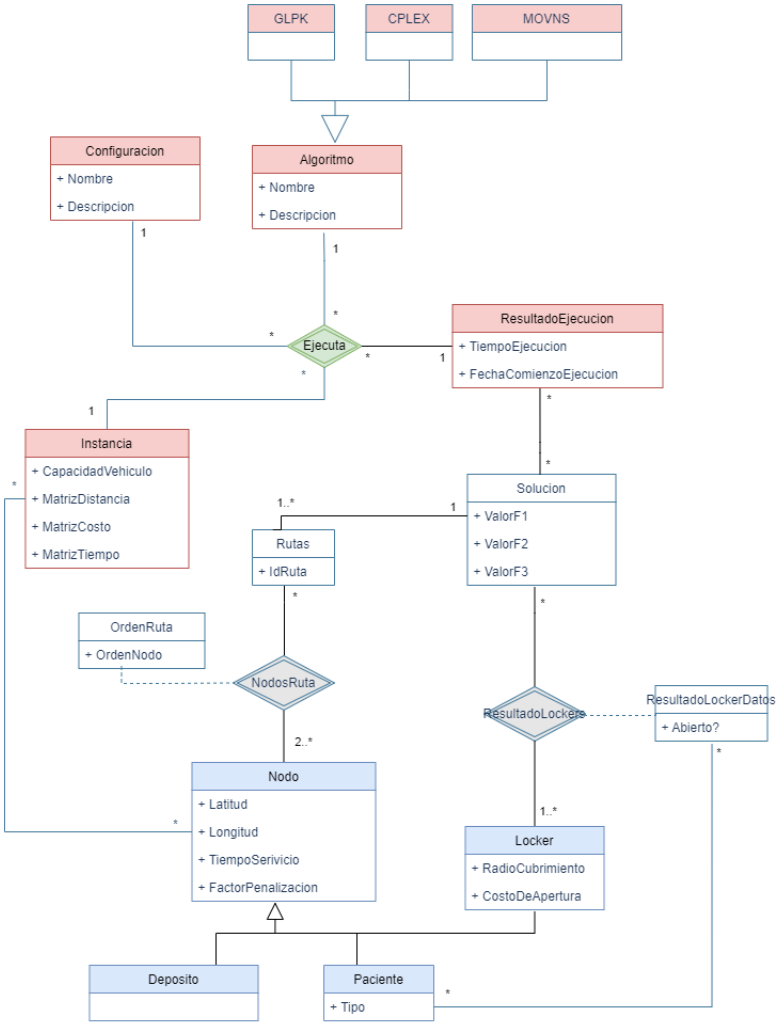

Figura 6.18: Modelo entidad relación.

Capítulo 7

Experimentación numérica

En este capítulo se describen los experimentos computacionales realizados para analizar la efectividad del MO-VNS desarrollado. La experimentación numérica se llevó a cabo en dos etapas. La primera etapa consistió en probar el MO-VNS contra un solver exacto (CPLEX en AMPL). Dado que CPLEX resuelve problemas mono-objetivo y MO-VNS es multiobjetivo, para realizar esta comparación se sigue un procedimiento específico, el cual se explica en la Sección 7.4. Posteriormente, se realiza una segunda etapa de experimentación Sección 7.5, consistente en la ejecución de distintas variantes de MO-VNS con instancias grandes de forma de determinar cuál de ellas es la mejor variante. Los experimentos se realizaron en una máquina con las siguientes especificaciones: CPU AMD Ryzen 5 5600X 6-Core Processor 3.70 GHz con 16 GB de RAM. En la Sección 7.1 se describen las instancias de prueba generadas para la realización de esta experimentación. Luego en la Sección 7.2 se describen las métricas utilizadas en esta experimentación para la comparación de algoritmos multiobjetivos. Posteriormente, en la Sección 7.3, se describen las variantes del algoritmo MO-VNS desarrollado a probar en la segunda etapa de esta experimentación. Luego, en la Sección 7.4 y en la Sección 7.5 se detallan las etapas de experimentación realizadas.

7.1. Generación de instancias

En esta sección, se presenta la generación de dos conjuntos de instancias de prueba. Un conjunto de instancias chicas con 25 nodos y un conjunto de instancias grandes con 100 nodos. La razón detrás de esta división es para facilitar la evaluación del rendimiento contra CPLEX (utilizando instancias chicas) y también para determinar cómo se comporta MO-VNS en instancias de mayor tamaño. Mientras que las instancias grandes se utilizaron para comparar las distintas variantes del algoritmo MO-VNS desarrollado, ya que CPLEX no puede resolver instancias grandes en un tiempo razonable.

Los dos conjuntos se crearon siguiendo las siguientes topologías con el obje-

tivo de tener distintas disposiciones de los nodos de forma genérica:

- Isla (Figura 7.1): Consiste en un conjunto de nodos dispuestos de forma uniforme dentro de un círculo alrededor del depósito.

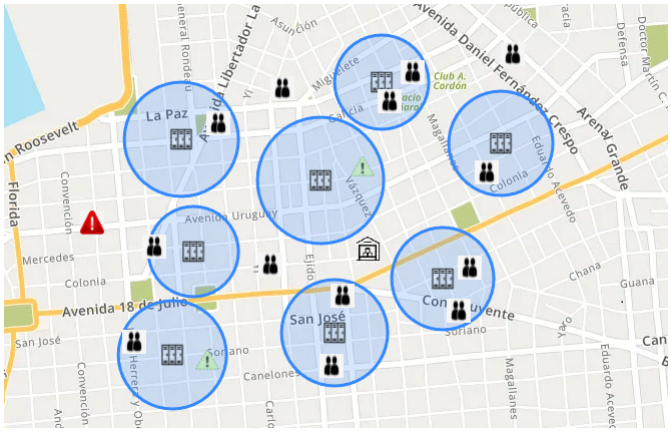

Figura 7.1: Topología una isla.

- Dos Islas (Figura 7.2): Consiste en dos islas con el depósito entremedio de ambas.

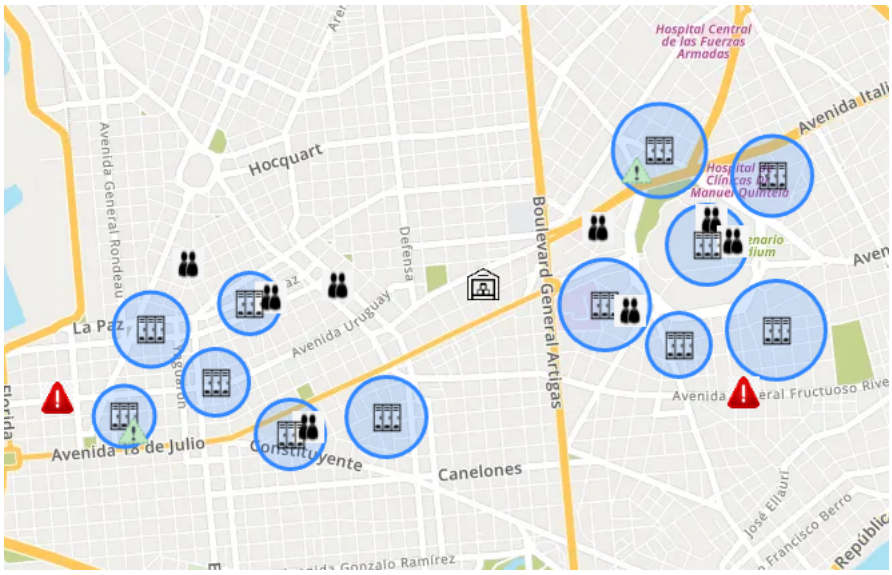

Figura 7.2: Topología dos islas.

- Árbol (Figura 7.3): Los nodos dispuestos en forma de árbol con el depósito

como raíz.

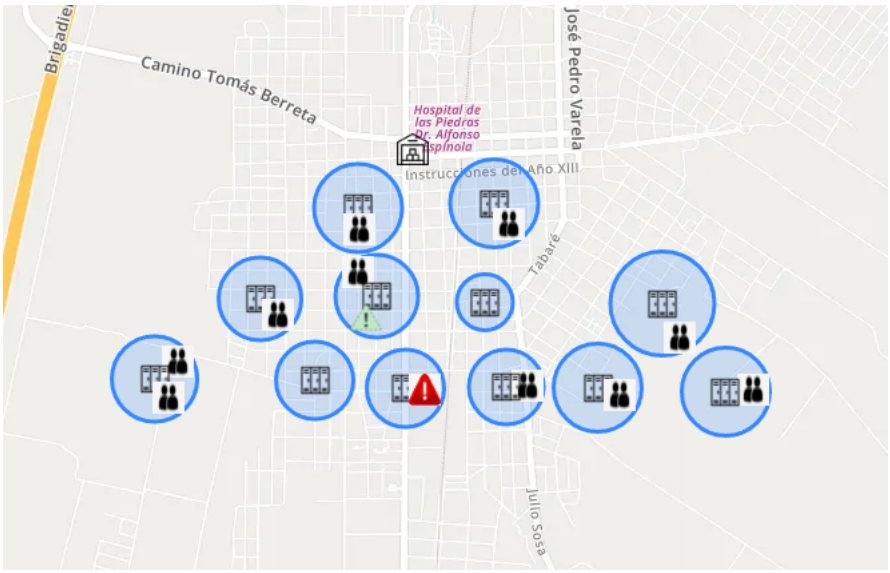

Figura 7.3: Topología árbol.

- Anillo (Figura 7.4): Los nodos forman un anillo. El depósito se encuentra en dicho anillo en una posición aleatoria.

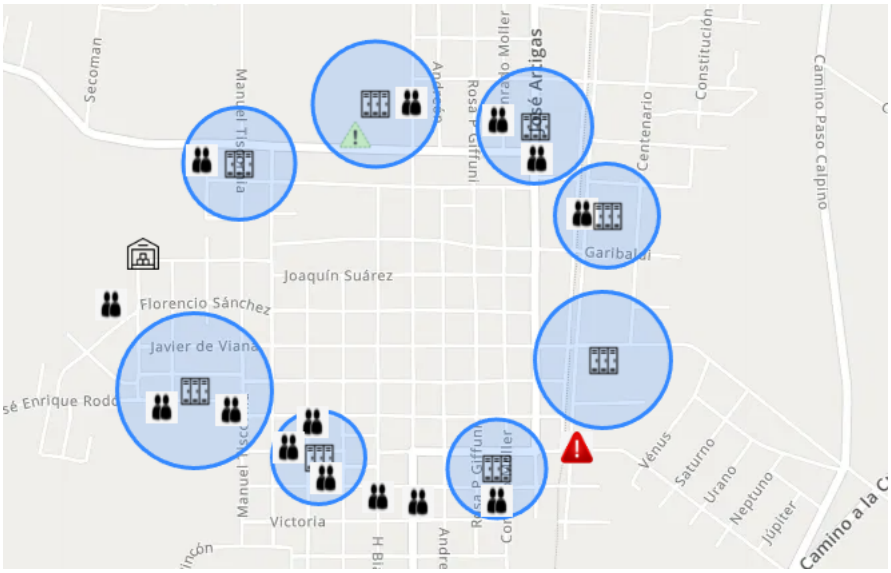

Figura 7.4: Topología anillo

- Línea (Figura 7.5): Los nodos están dispuestos a lo largo de una línea. El depósito se encuentra en una posición aleatoria en dicha línea.

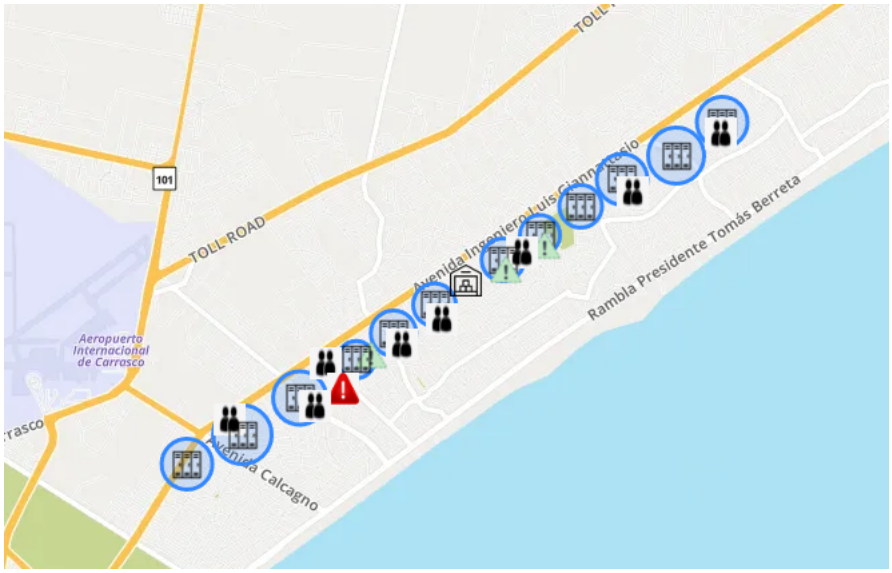

Figura 7.5: Topología línea.

- Estrella (Figura 7.6): Similar al anillo, pero el depósito ahora se encuentra en el centro del anillo.

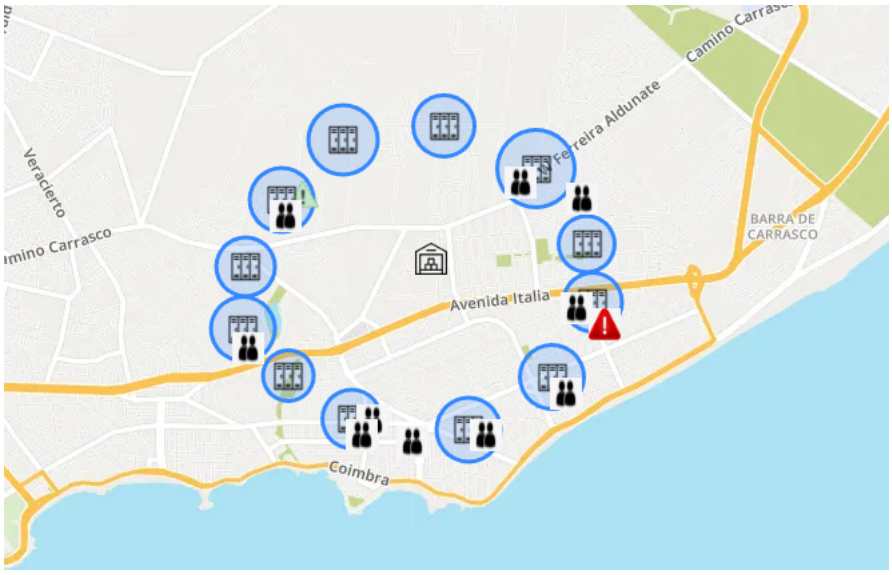

Figura 7.6: Topología estrella.

A continuación, se procede a describir los dos conjuntos de instancias.

Instancias chicas Se generaron 36 instancias chicas (cada instancia consta de 25 nodos, incluyendo el depósito), 6 instancias por topología. Las 6 instancias de cada topología se crearon variando la cantidad de lockers y la capacidad de cada vehículo, de la siguiente forma: se crean 3 instancias donde 2/3 de los nodos son pacientes y 1/3 lockers, las otras 3 instancias se crean con 1/2 de pacientes y lockers. Cada instancia de estos dos grupos de 3 instancias se crean variando la capacidad del vehículo en cantidad de pacientes / 2, cantidad de pacientes / 3 y cantidad de pacientes / 5. El costo entre arcos se elige proporcional a la distancia, es decir, tiene el mismo valor que la distancia entre arcos. El tiempo entre nodos se elige como la distancia entre nodos dividido 45 (simulando una velocidad de 45 km/h). El resto de parámetros se elige aleatoriamente uniforme entre un par de valores, como sigue: tiempo de servicio (entre 100 y 1000), costo de abrir lockers (entre 100 y 1000) y penalización (entre 100 y 200). Los tipos de pacientes se eligen aleatoriamente uniforme, asegurando mayor cantidad de pacientes indiferentes que el resto de tipos de pacientes. En la Tabla 7.1 se enumeran las instancias generadas, incluyendo la cantidad de cada tipo de paciente.

Instancias grandes Se generó 1 instancia por topología (6 en total) con los siguientes valores de parámetros: 66 pacientes ubicados aleatoriamente uniforme y seleccionando el tipo de paciente (11 bajo riesgo, 22 alto riesgo, y 33 indiferentes) también de forma aleatoria y uniforme, 33 lockers (radio variable

		Cantidad				Tipo paciente	
Topologia	Instance ID	Pacientes	Lockers	Cap.vehiculo	Ind	Alto riesgo	Bajo riesgo
Isla	1	16	8	8	13	1	2
Isla	2	16	8	5	13	1	2
Isla	3	16	8	3	13	1	2
Isla	4	12	12	6	9	1	2
Isla	5	12	12	4	9	1	2
Isla	6	12	12	2	9	1	2
2 isla	7	16	8	8	12	2	2
2 isla	8	16	8	5	12	2	2
2 isla	9	16	8	3	12	2	2
2 isla	10	12	12	6	8	2	2
2 isla	11	12	12	4	8	2	2
2 isla	12	12	12	2	8	2	2
Anillo	13	16	8	8	12	2	2
Anillo	14	16	8	5	12	2	2
Anillo	15	16	8	3	12	2	2
Anillo	16	12	12	6	10	1	1
Anillo	17	12	12	4	10	1	1
Anillo	18	12	12	2	10	1	1
Arbol	19	16	8	8	14	1	1
Arbol	20	16	8	5	14	1	1
Arbol	21	16	8	3	14	1	1
Arbol	22	12	12	6	10	1	1
Arbol	23	12	12	4	10	1	1
Arbol	24	12	12	2	10	1	1
Estrella	25	16	8	8	12	1	3
Estrella	26	16	8	5	12	1	3
Estrella	27	16	8	3	12	1	3
Estrella	28	12	12	6	8	1	3
Estrella	29	12	12	4	8	1	3
Estrella	30	12	12	2	8	1	3
Linea	31	16	8	8	13	1	2
Linea	32	16	8	5	13	1	2
Linea	33	16	8	3	13	1	2
Linea	34	12	12	6	10	1	1
Linea	35	12	12	4	10	1	1
Linea	36	12	12	2	10	1	1

Tabla 7.1: Instancias chicas 25 nodos.

seleccionado aleatoriamente uniforme entre 100 metros y 500 metros), capacidad vehículo de 15. El resto de parámetros (tiempo entre nodos, costo de arcos, tiempo de servicio, costo de abrir, locker y penalización) se eligen de la misma forma que para las instancias chicas. En la Tabla 7.2 se enumeran las instancias grandes, incluido la cantidad de cada tipo de paciente.

		Cantidad				Tipo paciente		
Topologia	Instance ID	Pacientes	Lockers	Cap.vehiculo	Ind	Alto riesgo	Bajo riesgo	
Isla	37	100	66	33	15	33	22	11
2 isla	38	100	66	33	15	33	22	11
Anillo	40	100	66	33	15	33	22	11
Arbol	41	100	66	33	15	33	22	11
Estrella	42	100	66	33	15	33	22	11
Linea	43	100	66	33	15	33	22	11

Tabla 7.2: Instancias grandes 100 nodos.

7.2. Métricas utilizadas

Para determinar que tan bueno es un algoritmo multiobjetivo con respecto a otro es necesario utilizar métricas que trabajen sobre las fronteras de Pareto, como se explica en la Sección 2.4. Para ello se utiliza la librería emoa en R (Mersmann, 2023b). La implementación se realiza en C# .NET utilizando la librería RdotNet para invocar las funciones en R de la librería emoa. En estos experimentos se utilizan las siguientes métricas, donde la implementación de emoa del hipervolumen, ϵ y r2 se obtienen del trabajo de Zitzler et al. (2003) y la especificación emoa de cada una de esas métricas en Mersmann (2023a):

- AVG(E): Promedio de cantidad de soluciones de la frontera.
- hipervolumen: Se utiliza la medida hypervolume_indicator(E, R) de la librería emoa. Esta métrica precisa del conjunto de soluciones E y un conjunto de referencia R. Calcula la diferencia del hipervolumen de E y el de R, por lo que a menor medida de este hipervolumen más cerca está E de R por lo que deriva en mejor calidad de la solución.
- Épsilon: Medida epsilon_indicator de emoa. En esta métrica, si se elige la versión multiplicativa, dada una frontera A y una frontera de referencia R, eps(A, R) es el mínimo factor por el cual se puede multiplicar cada punto de R de forma que el resultado de la transformación sea débilmente dominado por A. De esta forma, a menor medida, mejor es la calidad del método (Duarte et al., 2015).
- R2: Medida r2_indicator de emoa. Utilizando la función de utilidad Augmented Tchebycheff la cual combina una función lineal con una función no lineal, evitando valores idénticos para distintos puntos de la solución (Zitzler et al., 2003).
- C: Medida C, como se explica en la Sección 2.4.

Las características de cada métrica se describen en la Sección 2.4. Varias de estas medidas necesitan un conjunto de referencia y un punto de referencia. Como se menciona en (Duarte et al., 2015), el conjunto de puntos de referencia se puede crear generando un conjunto de puntos de eficiencia a partir de la unión de todas las soluciones generadas por las distintas versiones del algoritmo. El punto de referencia es el ideal point (Knowles et al., 2006) el cual es el punto que domina débilmente al resto de ambos conjuntos, a lo que domina al conjunto de referencia es el mismo para todas las comparaciones.

Dado que cada versión del algoritmo se ejecuta sobre varias instancias, se toma el promedio en cada medida. Por ejemplo, la medida de hipervolumen generada es el promedio del hipervolumen para todas las instancias.

Para la medida C se realiza la comparación para todas las combinaciones posibles de las distintas variantes del algoritmo.

7.3. Variantes del algoritmo a probar

En esta sección se describen las variantes del algoritmo a probar. La selección de estas variantes está basada en el trabajo dé Duarte et al. (2015). La idea de estas variantes es ver como se comporta el algoritmo si solo ejecutase cada una de sus partes por separado (Algoritmo 1 a 3), como se comporta al cambiar uno de los operadores utilizados para el Shake (Algoritmo 4) y además compararlo contra el algoritmo de inicialización, para ver cuanto mejora con respecto a la inicialización.

- Algoritmo 1: Algoritmo descrito en la Sección 5.2
- Algoritmo 2: Algoritmo 1 pero sin el procedimiento VND. Este algoritmo solo ejecuta el procedimiento MO - Shake 8 y MO - NeighborhoodChange 11.
- Algoritmo 3: Ejecuta solo el procedimiento MO - VND 9 del Algoritmo 1.
- Algoritmo 4: Algoritmo 1, pero se modifica las vecindades del procedimiento MO - Shake, en el Algoritmo 1 la vecindad $N_k(x)$ consiste en aplicar el operador ShakeOperator(x) en composición k veces, donde ShakeOperator(x) = ChangeOpenLocker(x) definido en la Sección 5.2, por ejemplo para k = 2, $N_2(x) = ChangeOpenLocker(ChangeOpenLocker(x))$. En el Algoritmo 4, ShakeOperator(x) = $ChangeRoute(ChangePatientAssignation(ChangeOpenLocker(x)))$, por ejemplo $N_2(x) =$ $ChangeRoute(ChangePatientAssignation(ChangeOpenLocker$ $(ChangeRoute(ChangePatientAssignation(ChangeOpenLocker(x)))))$.
- Algoritmo 5: Ejecuta solo la inicialización greedy del algoritmo. Con esto se busca probar que tanto mejora el algoritmo luego de la inicialización.

7.4. Primera etapa

Esta etapa consiste en comparar el MO-VNS contra CPLEX. Dado que CPLEX resuelve problemas mono-objetivo y MO-VNS resuelve problemas multiobjetivo la comparación se realiza como sigue.

Se parte ejecutando el algoritmo MO-VNS un total de 30 veces para cada instancia por un tiempo máximo de 30 minutos cada ejecución y tiempo sin mejora de 5 minutos, para cada una de esas ejecuciones se aplica el siguiente procedimiento para ejecutar CPLEX, con un tiempo máximo de ejecución de 30 minutos. Para cada solución del MO-VNS, se genera un modelo en el que se pretende minimizar una función objetivo con CPLEX, y los otros dos objetivos se tratan como restricciones. Esto genera 3 modelos por solución (uno para cada objetivo). Ejemplo, para la función objetivo 1 (F1): se genera un modelo matemático igual al original, pero que optimiza F1 y con las restricciones de igualdad en F2 y F3 con los valores encontrados por MO-VNS, como se muestra en la Figura 7.7.
Se seleccionan 12 de las instancias chicas (seleccionadas de forma aleatoria uniforme), resultando en las instancias con los IDs 1, 5, 7, 11, 13, 17, 19, 23, 25, 29, 31 y 35 en la Tabla 7.1. Luego se calculan las distintas métricas.

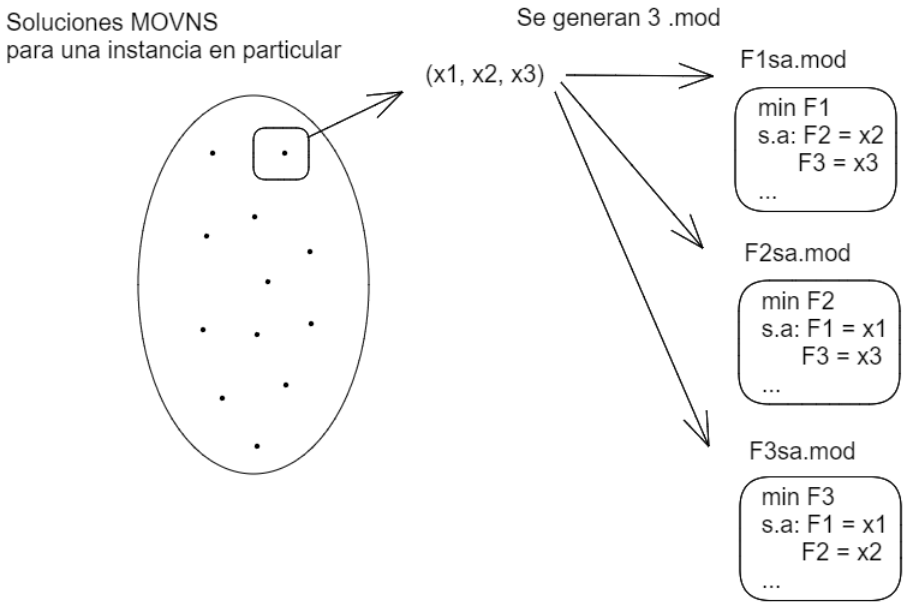

Figura 7.7: Generación de modelos para comparación con CPLEX.

La comparación se realiza de dos maneras distintas. La primera se utiliza para ver qué tan buena es una solución de forma independiente. Para esto, se compara cada solución del MO-VNS con sus respectivas soluciones en CPLEX.

Si el valor encontrado por CPLEX de la función objetivo minimizada es el mismo que el valor de esa función de MO-VNS, significa que no hay otra solución con el mismo valor en los otros objetivos y menor valor en la función minimizada, siempre y cuando el CPLEX encuentre el óptimo.
Para esta parte, siempre que CPLEX encontró el óptimo, el valor óptimo fue igual al de la solución con MO-VNS en esa función objetivo.
Por otro lado, en los casos donde CPLEX no encontró el óptimo (menor al 10 % de los casos), el valor encontrado por CPLEX se encuentra con un absmipgap (porcentaje de diferencia entre la solución encontrada por CPLEX y el valor óptimo del problema relajado («Options for CPLEX Solver in AMPL», s.f.)) menor al 8 % en F1, menor a 19 % en F2 y menor al 25 % en F3. En estos casos el valor del MO-VNS resultó menor que el valor encontrado por CPLEX, por lo que la distancia al óptimo es aún menor.

La segunda manera de comparar consiste en generar una frontera de Pareto generada por CPLEX, conformada por todas las soluciones brindadas por CPLEX a los distintos modelos descritos anteriormente. Luego se convierte ese conjunto en un conjunto de puntos de eficiencia (conjunto en el que no existe un punto que domine completamente a otro de ese conjunto, es decir, los puntos del conjunto no se dominan entre sí). Con dicha frontera, se aplica las métricas C, hipervolumen, épsilon y r2. El conjunto de referencia que utilizan las métricas hipervolumen y épsilon, se obtiene de unir las soluciones obtenidas por ambos algoritmos y se aplica la función para convertir el conjunto en un conjunto de puntos de eficiencia.
Con el primer paso nos aseguramos que cada una de las soluciones del MO-VNS es buena de forma independiente y con el segundo paso se compara a nivel de frontera.

La Tabla 7.3 presenta los resultados de cada métrica para estas 12 instancias. Como se mencionó anteriormente, se ejecutó 30 veces el algoritmo para cada instancia, por lo que cada fila de la tabla muestra el valor promedio de cada métrica de esas 30 ejecuciones para una instancia dada. La primera columna identifica la instancia. La segunda y la tercera son la métrica C, ya que no es suficiente con calcular C(A, B), también hay que calcular C(B, A) (Duarte et al., 2015). El resto de columnas representan el hipervolumen, medida épsilon ϵ y medida r2, respectivamente para ambos algoritmos.
En cuanto a la medida C, en todos los casos el algoritmo MO-VNS es mejor que CPLEX. Para el resto de medidas, la calidad de las fronteras obtenidas varía. Según Knowles et al. (2006) si al menos dos de las cuatro métricas se contradicen, quiere decir que no se puede determinar que una frontera es mejor que otra. En la Tabla 7.3 se muestra como el MO-VNS tiene mejores valores que CPLEX para las instancias 5, 11, 17, 23, 29, 31, y 35, mientras que para las instancias 1, 7 y 13 CPLEX brinda mejores fronteras. Por otro lado, para las instancias 19 y 25 las fronteras son incomparables. Por lo que para la mayoría de las instancias MO-VNS resultó tener mejor desempeño.

Id	C(A, B)	C(B, A)	Hyp(A)	Hyp(B)	Eps(A)	Eps(B)	r2(A)	r2(B)
1	0,000	0,333	0,754	1,000	1,000	0,168	0,264	0,494
5	0,000	0,000	0,251	0,028	0,195	0,054	0,827	0,137
7	0,000	0,375	0,158	0,652	0,343	0,180	0,409	0,510
11	0,100	0,217	0,051	0,005	0,216	0,009	0,779	0,030
13	0,000	0,000	0,097	0,117	0,305	0,027	0,209	0,350
17	0,100	0,150	0,027	0,002	0,156	0,000	0,261	0,027
19	0,000	0,308	0,034	0,034	0,096	0,021	0,304	0,189
23	0,100	0,125	0,060	0,005	0,156	0,009	0,354	0,018
25	0,000	0,250	0,131	0,212	0,545	0,012	0,360	0,254
29	0,000	0,375	0,075	0,018	0,030	0,009	1,000	0,067
31	0,100	0,200	0,164	0,051	0,177	0,039	0,469	0,138
35	0,200	0,375	0,023	0,000	0,018	0,003	0,174	0,000

Tabla 7.3: Resultados comparación MO-VNS vs CPLEX (A = CPLEX, B = MO-VNS). Valores normalizados entre 0 y 1. En hipervolumen, ϵ y R2 a menor valor mejor la frontera. En la medida C a mayor valor mejor el algoritmo.

En conclusión, MO-VNS resultó ser mejor que CPLEX tanto en la comparación de las soluciones de forma independiente como en la comparación a nivel de fronteras.

7.5. Segunda etapa

Como se mencionó anteriormente, en esta etapa se prueban distintas versiones del algoritmo MO-VNS, en particular las descritas en la Sección 7.3. Se ejecuta cada algoritmo 10 veces para cada instancia grande, y se toma el valor promedio en las distintas métricas debido a las distintas secciones con aleatoriedad existentes en los algoritmos. Luego se calculan las distintas métricas mencionadas en la Sección 7.2 y se determina qué variante del algoritmo es mejor, teniendo en cuenta que según Knowles et al. (2006) si al menos dos de las cuatro métricas se contradicen, quiere decir que no se puede determinar que una frontera es mejor que otra. En caso contrario, una frontera es mejor que la otra. En la Tabla 7.4 se presentan los resultados de las métricas para los distintos algoritmos en diferentes instancias. La primera columna indica el algoritmo. En la segunda columna, se muestra el promedio de soluciones por instancia encontradas por cada algoritmo sobre todas las instancias. Cuanto mayor sea este número, mejor será el algoritmo, ya que encontró un conjunto de soluciones no dominadas más grande. Las siguientes columnas son: hipervolumen, épsilon y r2, las cuales representan los valores promedio por instancia calculados sobre todas las instancias, normalizados entre 0 y 1 utilizando la normalización mínima-máxima $z_i = (x_i – minimo(x))/(maximo(x) – minimo(x))$, dónde z_i es el nuevo valor normalizado, x_i es el valor no normalizado, $minimo(x)$ es el menor valor del conjunto, $maximo(x)$ es el máximo valor del conjunto. Como se mencionó en la Sección 7.2, a menor estas medidas, mejor solución.
Para ver las tablas de las ejecuciones de los algoritmos, por instancia remitirse

al Apéndice B.1. Como se puede observar en la Tabla 7.4, el Algoritmo 4 es el mejor algoritmo con respecto a las métricas hipervolumen, épsilon y r2. Seguido por el Algoritmo 1, quien tiene valores similares en esas métricas, pero genera mayor cantidad de soluciones proporcionando mayor diversidad de soluciones. Luego sigue el Algoritmo 2, generando muchas más soluciones pero con peor calidad. Y por último el Algoritmo 3, quien genera muchas más soluciones, pero tiene la peor calidad de los cuatro. Que el Algoritmo 2 genere muchas soluciones es esperable, ya que este no realiza el procedimiento VND, solo se centra en el Shake, procedimiento que explora el espacio de búsqueda. El Algoritmo 3 solo realiza el procedimiento VND, el cual se centra en mejorar los valores de las soluciones. Por último, se puede ver que el Algoritmo 5 posee el peor valor en las 3 métricas, menos en el average de soluciones que por defecto el algoritmo de inicialización crea 200 soluciones.

Algoritmo	AVG(E)	Hyp	ϵ	R2
1	101,167	0,014	0,227	0,000
2	175,333	0,523	0,605	0,945
3	192,000	0,820	0,920	0,812
4	62,500	0,000	0,000	0,004
5	200,000	1,000	1,000	1,000

Tabla 7.4: Resultados comparación variantes MO-VNS, E = conjunto de soluciones. Valores normalizados entre 0 y 1. A menor valor de hipervolumen, ϵ y R2 mejor el algoritmo. A mayor valor de AVG(E) mejor el algoritmo

A continuación, se muestran los resultados de la métrica C. Notar que se calcula C(A, B) y C(B, A).

Algoritmo/Algoritmo	1	2	3	4	5
1	100	65	94	32	100
2	7	100	96	5	100
3	1	2	100	1	100
4	36	66	90	100	100
5	0	0	0	0	100

Tabla 7.5: Medida C en porcentaje de dominancia, para cada par de variante del algoritmo. Por filas, a mayor valor, mejor el algoritmo, por columnas se considera de forma opuesta.

En la Tabla 7.5, cada columna indica qué tan dominado es el algoritmo, pues a mayor porcentaje es más dominado por los otros algoritmos. Las filas indican cuánto domina ese algoritmo a los demás.
Si se ve por filas, que indica cuanto domina un algoritmo al resto, se ve que el Algoritmo 1 domina un 65 % de soluciones del Algoritmo 2, un 94 % de soluciones del Algoritmo 3 y un 32 % del Algoritmo 4. Mientras que el Algoritmo 4 domina un 36 % de soluciones del algoritmo 2, un 66 % de soluciones del Algoritmo 2

y un 90 % del Algoritmo 4. Por lo que con respecto a Algoritmos 2 y 3, los Algoritmos 1 y 4 son superiores, mientras que el 1 domina 32 % del 4 y el 4 un 36 % del 1, o sea que son muy similares. Luego seguiría el Algoritmo 2 quien domina bastante al Algoritmo 3 y poco al 1 y 4. Por último, el Algoritmo 3 quien domina muy poco al resto de algoritmos y es muy dominado por estos. Además, como se puede ver, el Algoritmo 5 es dominado completamente por el resto de algoritmos.

En conclusión, los Algoritmos 1 y 4 presentan los mejores valores. Si la cantidad de soluciones deseadas no es algo crítico, entonces el algoritmo 4 es mejor, de lo contrario el algoritmo 1 sería el ideal.

Capítulo 8

Conclusiones y Trabajo Futuro

El presente trabajo abordó el Location Routing Problem extendiéndolo con criterios no convergentes. Se trabajó partiendo del trabajo de Veenstra et al. (2018a) al cual se le agregaron características que se entiende que son de interés al problema por su aplicabilidad: vehículos con capacidad limitada, rutas que incluyan tanto a lockers como pacientes y distintos tipos de pacientes. Además, se consideran dos objetivos más aparte de la minimización de costos: maximizar equidad y minimizar el tiempo total.

Se realizó una revisión del estado del arte relacionado con el LRP donde se recopilaron varios trabajos relacionados. A través de este análisis, se logró obtener una comprensión del problema, su definición, variantes, modelado del problema, las áreas de aplicación, los métodos de resolución y las técnicas para evaluar el rendimiento de los algoritmos desarrollados.

A partir del análisis de la literatura, se seleccionó un trabajo que abordaba la resolución de un LRP específico relacionado con la entrega de medicamentos desde una farmacia hacia lockers y pacientes, con el objetivo de minimizar los costos. Se decidió trabajar sobre este trabajo y se discutió el alcance para determinar las funcionalidades a implementar. Se extendió para abarcar una versión multiobjetivo, considerando objetivos no convergentes como el costo, la eficiencia y la equidad, objetivos que son ampliamente considerados en la logística humanitaria (Deb et al., 2002). De esta forma se cumple con el objetivo de la tesis, el cual era resolver un LRP con criterios no convergentes aplicados a la logística humanitaria. Además, se tuvieron en cuenta diferentes tipos de pacientes, capacidad en los vehículos y se permitió que las rutas incluyeran tanto a los pacientes como a los lockers. Posteriormente, se procedió a modelar el nuevo problema y se llevó a cabo una etapa de validación del modelo. Para ello, se utilizó el solver GLPK en instancias tamaño pequeño y se implementó una primera versión de un prototipo web, que permitió validar el modelo, realizando correcciones y agregado de restricciones relevantes. Esta primera versión del

prototipo web disponía de funcionalidades como las de crear instancias y verlas en el mapa, y mostrar las soluciones en el mapa. A continuación, se evaluó el rendimiento del modelo utilizando instancias más grandes, empleando el solver CPLEX. Los resultados demostraron que el problema presenta una alta complejidad de resolución, ya que en muchos de los casos no se consiguió resolver el problema en un tiempo razonable.

Con el fin de seguir avanzando en el desarrollo de la solución, se continuó trabajando en el prototipo, añadiendo funcionalidades que facilitaron la realización de pruebas a los algoritmos. El prototipo final es una web con distintos paneles, los cuales incluyen las siguientes funcionalidades:

- Instancias: Creación de instancias con valores aleatorios, permitiendo el uso de un mapa para especificar posición de los nodos y los radios de cubrimiento de los lockers, tomando la distancia de los nodos según como se especifican los nodos en el mapa. Visualización de instancias mediante una tabla y en el mapa. Y eliminación de instancias.

- Ejecuciones: Permite planificar la ejecución de algoritmos con diferentes configuraciones y sobre distintas instancias y guardar el resultado de esas ejecuciones. Se permite además visualizar las ejecuciones que se han realizado, así como las distintas soluciones tanto por tabla como por mapa, mostrando las rutas en el mapa.

- Métricas: Se permite la ejecución de las métricas sobre distintas ejecuciones, además de permitir exportarlas a Excel.

Luego se pasó a crear dos conjuntos de instancias siguiendo seis tipos de topologías distintas, de forma de abarcar distintos tipos de distribuciones de los nodos.

Posteriormente, se desarrolló el algoritmo MO-VNS y se realizó una evaluación comparativa utilizando el prototipo web con las instancias de menor tamaño, contrastándolo con el software CPLEX. Dado que CPLEX resuelve problemas mono-objetivo y el algoritmo desarrollado es multiobjetivo, se propuso y llevo a cabo una forma de compararlos, ejecutando CPLEX tres veces para cada solución del MO-VNS, donde se fijan los valores de dos objetivos y se optimiza el restante.

Finalmente, se llevó a cabo una fase de experimentación en la cual se compararon distintas variantes del algoritmo MO-VNS. Para comparar estas variantes del algoritmo, se utilizaron métricas multiobjetivo tales como hipervolumen, medida R2, medida ϵ y medida C sobre varias ejecuciones para aumentar la validez de las pruebas.

Los resultados demostraron que el algoritmo MO-VNS desarrollado ofrece mejores soluciones en comparación con CPLEX para el mismo tiempo de ejecución y para el conjunto de instancias propuesto. Por otro lado, los resultados de la segunda etapa de experimentación mostraron que las diferentes variantes brindan distintas características tanto en cantidad como calidad de las soluciones, encontrando que las variantes que mejores resultados brindan fueron las

que realizan todos los pasos del MO-VNS.

Trabajo a futuro

Con base en los hallazgos y limitaciones de este trabajo, se identifican algunas direcciones para la investigación futura:

- Múltiples farmacias: En vez de tener una única farmacia desde la cual repartir los medicamentos, se pueden considerar varias farmacias, con la posibilidad de que las rutas comiencen en una farmacia y terminen en otra.
- Reposición de medicamentos de los vehículos: Permitir que un vehículo pueda reabastecerse de medicamentos en una farmacia para seguir con la distribución.
- Capacidad en los lockers y farmacia/s: Que los lockers y farmacias tengan una capacidad limitada y cada uno su propia capacidad.
- Entregas con ventanas de tiempo: Cada locker o paciente tenga su fecha de entrega o tiempo máximo de entrega preestablecido.
- Tipos de medicamentos: Considerar distintos tipos de medicamentos a repartir, con distinto volumen y peso.
- Más de una unidad de demanda por paciente: Que los pacientes puedan demandar más de una unidad.
- Que los pacientes y lockers puedan ser visitados por más de un vehículo, esto puede servir para abastecer los lockers entre más de un vehículo.
- Considerar distintos tipos de vehículos y que cada tipo de vehículo tenga su propia capacidad.
- Disponer de distintas franjas horarias para las distintas entregas.
- Investigar la aplicabilidad de otros algoritmos de optimización para resolver problemas similares y comparar su desempeño con el algoritmo propuesto en este trabajo.
- Considerar la inclusión de otros criterios relevantes en la resolución del LRP, como la sostenibilidad ambiental o la satisfacción del cliente.
- Continuar mejorando la interfaz de la aplicación desarrollada para que sea más amigable para el usuario o que incluya más funcionalidades. Mejoras en la visualización de soluciones en el mapa, como por ejemplo: indicar que pacientes quedaron asignados a que lockers.
- Mejoras de performance en el algoritmo desarrollado.

En resumen, se entiende que en el trabajo realizado y descrito en este informe, se proporcionan las bases para la resolución del LRP.

Referencias

.NET Home. (s.f.). Consultado el 1 de agosto de 2023, desde https://dotnet.microsoft.com/

Balcik, B., Beamon, B. M., Krejci, C. C., Muramatsu, K. M., & Ramirez, M. (2010). Coordination in humanitarian relief chains: Practices, challenges and opportunities. *International Journal of production economics*, *126*(1), 22-34.

Barreto, S., Ferreira, C., Paixao, J., & Santos, B. S. (2007). Using clustering analysis in a capacitated location-routing problem. *European Journal of Operational Research*, *179*(3), 968-977.

Cheng, S., Shi, Y., & Qin, Q. (2012). On the performance metrics of multiobjective optimization. *International Conference in Swarm Intelligence*, 504-512.

Contardo, C., Cordeau, J.-F., & Gendron, B. (2014). An exact algorithm based on cut-and-column generation for the capacitated location-routing problem. *Journal on Computing*, *26*(1), 88-102.

Crainic, T. G., Perboli, G., & Tadei, R. (2016). Collaborative strategies for city distribution. *Transportation Research Procedia*, *12*, 709-722.

Crowder, H., Johnson, E. L., & Padberg, M. (1983). Solving large-scale zero-one linear programming problems. *Operations Research*, *31*(5), 803-834.

Dantzig, G. B. (1947). Maximization of a Linear Function of Variables Subject to Linear Inequalities. *Activity Analysis of Production and Allocation*, 339-347.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, *6*(2), 182-197.

Duarte, A., Pantrigo, J. J., Pardo, E. G., & Mladenovic, N. (2015). Multi-objective variable neighborhood search: an application to combinatorial optimization problems. *Journal of Global Optimization*, *63*, 515-536.

Entity Framework Documentation Hub. (s.f.). Consultado el 1 de agosto de 2023, desde https://learn.microsoft.com/en-us/ef

Fisher, M. L., Jaikumar, R., & Van Wassenhove, L. N. (1994). A heuristic for the location of depots in a capacitated vehicle routing problem. *Operations Research*, *42*(1), 109-118.

Gandibleux, X., Sevaux, M., Sörensen, K., & T'kindt, V. (2012). *Metaheuristics for multiobjective optimisation* (Vol. 535). Springer Science & Business Media.

Gendreau, M., Laporte, G., & Seguin, R. (1996). VRP with time windows: exact algorithms and heuristics. *European Journal of Operational Research*, *88*(1), 9-23.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, *13*(5), 533-549.

Govindan, K., Jafarian, A., Khodaverdi, R., & Devika, K. (2014). Two-echelon multiple-vehicle location-routing problem with time windows for optimization of sustainable supply chain network of perishable food. *International Journal of Production Economics*, *152*, 9-28.

Guo, Y., Chen, J., Yang, J., & Chen, J. (2019). A game-theoretic approach for flow assignment in wireless sensor networks. *IEEE Transactions on Mobile Computing*, *19*(5), 1044-1056.

Hansen, M. P., & Jaszkiewicz, A. (1994). *Evaluating the quality of approximations to the non-dominated set*. IMM, Department of Mathematical Modelling, Technical Universityof Denmark.

Hemmelmayr, V. C., Doerner, K. F., & Hartl, R. F. (2012). A variable neighborhood search for the multi-period collection of recyclable materials. *Transportation Research Part B: Methodological*, *46*(10), 1596-1616.

Holland, J. H. (1975). Adaptation in natural and artificial systems.

Kirkpatrick, S., Gelatt Jr, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671-680.

Knowles, J. D., Thiele, L., & Zitzler, E. (2006). A tutorial on the performance assessment of stochastic multiobjective optimizers. *TIK-report*, *214*.

Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*.

Laporte, G., Semet, F., & Taillard, É. D. (2004). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research*, *52*(3), 461-471.

Laumanns, M., Thiele, L., Zitzler, E., Welzl, E., & Deb, K. (2002). Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem. *Parallel Problem Solving from Nature—PPSN VII: 7th International Conference Granada, Spain, September 7–11, 2002 Proceedings 7*, 44-53.

Li, H., & Wen, M. (2017). An end-to-end deep neural network for solving location-routing problems. *IEEE Transactions on Intelligent Transportation Systems*, *18*(8), 2179-2191.

Lin, S., & Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, *21*(2), 498-516.

Liu, C., Kou, G., Peng, Y., & Alsaadi, F. E. (2019a). Location-routing problem for relief distribution in the early post-earthquake stage from the perspective of fairness. *Sustainability*, *11*(12), 3420.

MapQuest Developer Documentation. (s.f.). Consultado el 1 de agosto de 2023, desde https://developer.mapquest.com/

Melo, M. T., Nickel, S., & Saldanha-Da-Gama, F. (2009). Facility location and supply chain management–A review. *European journal of operational research*, *196*(2), 401-412.

Mersmann, O. (2023a). *Binary quality indicators in emoa*. Consultado el 1 de agosto de 2023, desde https://rdrr.io/cran/emoa/man/binary_indicator.html

Mersmann, O. (2023b). *Emoa: Evolutionary multiobjective optimization algorithms version 0.5-0.2 from cran*. Consultado el 1 de agosto de 2023, desde https://rdrr.io/cran/emoa/

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, *21*(6), 1087-1092.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & operations research*, *24*(11), 1097-1100.

MUI: A popular React UI framework. (s.f.). Consultado el 1 de agosto de 2023, desde https://mui.com/

Nagy, G., & Salhi, S. (2007). Location-routing: Issues, models and methods. *European journal of operational research*, *177*(2), 649-672.

Options for CPLEX Solver in AMPL. (s.f.). Consultado el 1 de agosto de 2023, desde https://dev.ampl.com/solvers/cplex/options.html

Perboli, G., Rei, W., & Tadei, R. (2018). The two-echelon capacitated vehicle routing problem: Models and math-based heuristics. *Transportation Science*, *52*(3), 730-747.

Prins, C., Prodhon, C., Calvo, R. W., & Soriano, P. (2007). A memetic algorithm with population management (MA—PM) for the capacitated location-routing problem. En *Metaheuristics: Computer Decision-Making* (pp. 513-528). Springer.

Prodhon, C., & Prins, C. (2014a). A survey of recent research on location-routing problems. *European Journal of Operational Research*, *238*(1), 1-17.

Rawls, C. G., & Turnquist, M. A. (2010). Pre-positioning of emergency supplies for disaster response. *Transportation Research Part B: Methodological*, *44*(4), 521-534.

React Data Grid Component Overview. (s.f.). Consultado el 1 de agosto de 2023, desde https://mui.com/x/react-data-grid/

React. The library for web and native user interfaces. (s.f.). Consultado el 1 de agosto de 2023, desde https://react.dev/

SQL Server Downloads. (s.f.). Consultado el 1 de agosto de 2023, desde https://www.microsoft.com/en-us/sql-server/sql-server-downloads

SQL Server Overview. (s.f.). Consultado el 1 de agosto de 2023, desde https://www.microsoft.com/en-us/sql-server

Tanenbaum, A. S., & Van Steen, M. (2007). *Distributed Systems: Principles and Paradigms*. Pearson Prentice Hall.

Tavana, M., Zandi, F., & Di Caprio, D. (2017). A hybrid intelligent algorithm for solving a multi-objective location-routing problem with time windows in a distribution network. *Expert Systems with Applications*, *78*, 319-331.

Tzeng, G.-H., Cheng, H.-J., & Huang, T.-D. (2007). Multi-objective optimal planning for designing relief delivery systems. *Transportation Research Part E: Logistics and Transportation Review*, *43*(6), 673-686.

Veenstra, M., Roodbergen, K. J., Coelho, L. C., & Zhu, S. X. (2018a). A simultaneous facility location and vehicle routing problem arising in health care logistics in the Netherlands. *European Journal of Operational Research*, *268*(2), 703-715.

Yan, J., Li, X., Li, X., Li, K., & Li, X. (2018). Multi-objective location-routing problem based on a hybrid genetic algorithm. *Applied Soft Computing*, *65*, 136-148.

Yu, V. F., Lin, S.-W., & Yang, T.-H. (2009). A simulated annealing heuristic for the capacitated location routing problem. *Computers & Industrial Engineering*, *56*(1), 408-419.

Zeng, B., Jiang, S., Zhang, W., & Zhou, Y. (2020). A deep convolutional neural network approach to the location routing problem. *Applied Soft Computing*, *93*, 106394. https://doi.org/10.1016/j.asoc.2020.106394

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., & Da Fonseca, V. G. (2003). Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on evolutionary computation*, *7*(2), 117-132.

Anexos

Anexo A

Estado del arte

Índice estado del arte

A.1. Introducción

El objetivo del documento es introducir el problema del LRP (Location-Routing Problem) el cual consiste a grandes rasgos en ubicar o abrir depósitos y encontrar las rutas óptimas desde dichos depósitos a nodos de demanda para la distribución de recursos, de forma de cumplir con ciertos objetivos. Particularmente se estudiará el LRP en el área de las logísticas humanitarias con criterios no convergentes, esto es, cuando se tienen varios objetivos que entran en conflicto, por ejemplo, reducción de costos y minimización de tiempo requerido para la entrega de recursos.

El LRP se puede descomponer en dos problemas de base altamente estudiados, estos son el VRP (Vehicle Routing Problem) y el FLP (Facility Location Problem) o problemas de localización en general. Por lo que para abordar al LRP, se sigue una estrategia incremental donde se introducen de manera separada cada uno de estos problemas para facilitar la comprensión y posteriormente tratarlos de forma conjunta.

La resolución del LRP es altamente aplicable en el área de las logísticas, y en particular en el área de gestión de riesgos. La gestión de riesgos es la disciplina que trata con el riesgo y la evitación del riesgo (George et al., 2011). Uno de los principales causantes de riesgos son los desastres. Los desastres son problemas complejos que prueban la habilidad de comunidades y naciones a efectivamente proteger su población e infraestructura, para reducir pérdidas humanas y de propiedad, y a recuperarse rápidamente (Altay & Green III, 2006). Se pueden definir como eventos impactantes que interrumpen seriamente el funcionamiento de una comunidad o sociedad, causando daños materiales, humanos, económicos o ambientales que no pueden ser manejados por agencias locales a través de procedimientos estándar (Jiang & Yuan, 2019).
Los desastres pueden estar causados por:

- Errores humanos y fallas tecnológicas.
- Malevolencia intencional.
- Causas naturales.
- Combinaciones de algunas o todas las anteriores.

Además, se consideran de baja prioridad y alto impacto. Baja prioridad de ocurrencia y alto impacto en la comunidad o entorno (Moscatelli et al., 2009).
Por otro lado, del conjunto de desastres, se pueden separar los desastres de gran escala o catástrofes. Un desastre a gran escala o catástrofe se define como eventos de altas consecuencias que provocan impactos generalizados y paralizantes, donde la habilidad de la sociedad perjudicada para responder al desastre se ve altamente comprometida (Holguín-Veras et al., 2012).
Un desastre a gran escala presenta entonces las siguientes características: impacto de gran escala (pudiendo afectar áreas geográficas amplias y grandes grupos de población), consecuencias severas (causando grandes cantidades de víctimas

y daños a propiedades), participación de múltiples agencias (involucrando muchas partes como voluntarios, equipos de rescate, etc), presiones de tiempo y riesgo (donde el tiempo es crítico para salvar vidas, así como presiones de tiempo para la toma de decisiones y acción), surgimiento de demanda y escasez de recursos, gran incertidumbre y daños en infraestructuras (Jiang & Yuan, 2019).

La gestión de riesgos es comúnmente descrita en cuatro fases: mitigación, preparación, respuesta, y recuperación. La fase de mitigación es la aplicación de medidas que previenen la aparición de un desastre o reducen el impacto que deberían ocasionar. Preparación son actividades que preparan a la comunidad para responder cuando un desastre ocurra. Respuesta es el empleo de recursos y procedimientos de gestión de riesgos guiados por planes para preservar vida, propiedades, ambiente y la estructura social, económica y política de una comunidad. Recuperación incluye las acciones tomadas a largo plazo luego del impacto de un desastre para estabilizar la comunidad y restaurarla a algo parecido a la normalidad (Jiang & Yuan, 2019).

Una situación de riesgo típica tiene las características como "gran incertidumbre, eventos repentinos e inesperados, riesgo de posibles víctimas en masa, gran cantidad de presión de tiempo y urgencia, gran escasez de recursos, impactos y daños de gran escala, ruptura de infraestructura de soporte necesaria para la coordinación como electricidad, telecomunicaciones y transporte. Lo que es complicado por factores como interdependencia entre infraestructuras, autoridad múltiple, participación de personal masivo, conflictos de intereses, y alta demanda por información en tiempo real" (Jiang & Yuan, 2019).
Para hacer frente a las situaciones de riesgo surgen las distintas logísticas, como son las logísticas de emergencias, que son la función de soporte que asegura la entrega a tiempo de recursos de emergencia y servicios de rescate en las regiones afectadas para asistir en actividades de rescate (Jiang & Yuan, 2019). Las logísticas humanitarias difiere de las logísticas de emergencias en que se concentran más en ayudar a las personas en su supervivencia durante y después de un desastre (Jiang & Yuan, 2019).
El término logística humanitaria abarca un gran rango de operaciones, incluyendo la distribución de suministros médicos para prevención de enfermedades, suministro de alimentos, y suministros críticos como consecuencia de un desastre (Holguín-Veras et al., 2012).
Las logísticas humanitarias presentan ciertos desafíos que en otras logísticas no ocurren. Por ejemplo, las logísticas comerciales tienen las siguientes características:

- El objetivo principal de las logísticas comerciales es el de minimizar el costo de transporte o de las logísticas.
- La carga a ser transportada se origina desde las compañías suministradoras hacia las compañías clientes.
- Se conoce la demanda.

- Hay procedimientos de toma de decisiones establecidos, que involucran un conjunto pequeño de tomadores de decisiones.
- Se transportan grandes cantidades de carga.
- Las redes sociales a cargo de las operaciones logísticas están intactas y funcionando.
- Los sistemas de soporte, como los de transporte, están relativamente estables y funcionales.

Mientras que las logísticas humanitarias abarcan una gran cantidad de actividades que ocurren en cualquier fase de la gestión de riesgos. En las fases de mitigación y preparación se realizan actividades para mejorar la seguridad y reducir los daños potenciales en personas e infraestructura si ocurre un desastre. En la fase de respuesta se realizan actividades de transporte de suministros y equipos para búsqueda y rescate, y de equipos y materiales para reparación de infraestructuras. La etapa de recuperación se puede dividir en dos fases: recuperación a corto plazo y a largo plazo. En la de recuperación a corto plazo se realizan actividades como gestión de donaciones y voluntarios, evaluación de daños, aseguración de viviendas temporales. Mientras que la recuperación a largo plazo puede durar años y se realizan actividades para volver a la normalidad o mejorar la calidad de vida de los afectados. Sin embargo, estas dos fases son diferentes, la primera ocurre en circunstancias caóticas y desafiantes, mientras que la segunda en circunstancias más estables (Holguín-Veras et al., 2012).
Todas estas características de las situaciones de emergencias presentan desafíos para las logísticas como son: la escala y complejidad del problema, criterios de decisión y objetivos diferentes, problemas de colaboración entre muchas partes, requerimientos de tiempo y toma de decisiones en tiempo real, asignación de recursos escasos, modelado estocástico y basado en escenarios para manejar la incertidumbre, además de que se debe tener en cuenta el daño en las infraestructuras (Jiang & Yuan, 2019).
Los problemas de localización surgen normalmente en la etapa de mitigación y preparación. El pre-posicionamiento de ayuda cerca de áreas donde hay posibilidad de ocurrencia de un desastre es una técnica que se usa para mejorar la capacidad y tiempos de respuesta a estos eventos, reduciendo tiempos y costos de las operaciones. Además, el proceso de evaluación de ubicaciones para centros de distribución, así como su cantidad, involucra muchos atributos y usualmente es necesario hacer compromisos entre factores tangibles e intangibles, como son aspectos ambientales, costos, calidad de vida, incentivos locales, proximidad a proveedores, proximidad a clientes, características laborales, capacidad de los centros de distribución, accesibilidad, seguridad (S. Roh et al., 2015).
En cambio, los problemas de ruteo son más comunes en la fase de respuesta en lo que se llama la distribución de última milla, donde es necesario transportar bienes desde uno o varios depósitos hacia zonas de demanda. El transporte de bienes en la distribución de última milla requiere múltiples criterios de performance que usualmente entran en conflicto como son tiempo, costos,

cubrimiento, equidad, seguridad (Awasthi et al., 2011). Lo que se suele buscar, dadas las características de las situaciones de emergencia, son las mejores rutas para transportar los bienes hacia las áreas afectadas, cumpliendo con ciertos criterios. Además, la gran incertidumbre luego de un desastre complica aún más estas decisiones de ruteo, dado que luego de un desastre es posible que ciertos caminos queden parcial o totalmente inutilizables (Alem et al., 2016).
Por otro lado, las decisiones de ruteo están íntimamente ligadas a las decisiones de ubicación de los depósitos desde donde parten los vehículos. Por lo que los problemas de localización y ruteo de forma conjunta también son comunes en la fase de respuesta. En estos problemas, normalmente se tienen un conjunto de posibles centros de distribución de los cuales algunos de ellos se abrirán o utilizarán, calculándose las mejores rutas por las que viajarán los vehículos para entregar los recursos. Además, se suele combinar la ubicación o apertura de centros de distribución con la asignación de demanda a dichos centros, luego los vehículos de un centro de distribución se encargan de suministrar los recursos a los nodos de demanda asignados al centro. Buscando optimizar ciertos objetivos como asignación y distribución justa de la ayuda y a tiempo (Liu et al., 2019b), maximización de demanda cubierta y en menor medida, minimización de costos (Rath & Gutjahr, 2014).

Como se puede ver, los criterios que definen una buena solución son distintos e inclusive más desafiantes en el área de gestión de riesgos que en otras áreas donde pierde relevancia minimizar costos, y se busca cumplir con principios de equidad (distribución justa de recursos), eficiencia (entrega a tiempo de los recursos) y eficacia (entrega de los recursos necesarios a las personas afectadas), los cuales entran en conflicto.

El LRP pertenece a la familia de problemas NP-hard (Marinakis, 2009), por lo que en la literatura aparecen métodos meta heurísticos de forma frecuente utilizados para obtener buenas soluciones.

El presente estado del arte está estructurado como sigue: la Sección Problemas de base se presentan por separado los problemas de localización y los problemas de ruteo, definiendo los problemas, enumerando variantes y métodos de resolución de trabajos que aplican la resolución de estos problemas al área de gestión de riesgos. La Sección Problemas de localización y ruteo aborda en profundidad al LRP, donde se define el problema, se muestran sus variantes y clasificaciones, así como casos de estudio aplicados al área de gestión de riesgos. Finalmente, en la Sección Conclusiones se muestran las principales lecciones del trabajo.

A.2. Problemas de base

Dado que el foco de estudio de este documento son los problemas de localización y ruteo de forma conjunta, se realiza solo un análisis superficial de los problemas de base, brindando una definición de cada problema, así como variantes, métodos de resolución y ejemplos de aplicación.

A.2.1. Problemas de localización

El problema de localización, en su versión más simple, consiste en determinar la ubicación de un depósito de forma de minimizar la suma de las distancias a un conjunto de clientes distribuidos geográficamente.
Otras variantes más complejas del problema abarcan múltiples depósitos a ubicar simultáneamente, y restricciones sobre la ubicación de las mismas (Tuy, 2009).

Los problemas de localización se pueden ver como problemas de ubicación de instalaciones (Facility Location) en los cuales se ubican un conjunto de instalaciones (o recursos) para minimizar el costo de satisfacer cierta demanda (de los clientes) respecto a un conjunto de restricciones.

Hay cuatro componentes que describen los problemas de ubicación: clientes o puntos de demanda (los cuales están previamente ubicados), instalaciones a ubicar, un espacio donde los clientes e instalaciones van a ser ubicados y una métrica que indica distancia o tiempo entre clientes e instalaciones.

Los modelos de ubicación de instalaciones difieren en la función objetivo, la métrica de distancia aplicada, el número y tamaño de las instalaciones, y ciertos índices o factores que llevan a diferentes tipos de modelos.

Algunos de los modelos clásicos de ubicación de instalaciones son: SFLP (Single Facility Location Problem), MFLP (Multiple Facility Location Problem), median problem, center problem, and covering problem. Otros más contemporáneos son HFLP (Hierarchical Facility Location Problem), HLP (Hub Location Problem), CLP (Competitive Location Problem).
A continuación se definen y mencionan algunos ejemplos de estos problemas. Por un recorrido sobre más variantes de problemas de localización puede encontrarse en el artículo de Krarup (2009), y en Farahani y Hekmatfar (2009).

Single Facility Location Problem El single facility location problem, consiste en ubicar un objeto (máquina en una tienda, ítem dentro de un almacén, un almacén en un lugar). Normalmente, se quiere minimizar una cierta función objetivo que toma en cuenta distancias entre la nueva instalación y el conjunto de instalaciones o ubicaciones existentes. En la práctica hay muchos factores que se tienen en cuenta para ubicar una nueva instalación, siendo difícil encontrar una ubicación única que cumpla con todos. Además, las soluciones varían según si el espacio donde se puede ubicar la instalación es continuo o discreto. Un ejemplo puede ser el que se menciona en el trabajo de S. Roh et al. (2015) en el cual se resuelve el problema de determinar un ranking de ubicaciones pa-

ra la localización de un depósito de ayuda humanitaria. Se pretende elegir la mejor ubicación para el depósito. Para comparar las ubicaciones hay que determinar los criterios a seguir, por lo que se utilizan reuniones de expertos para evaluar qué criterios utilizar. Ejemplos de criterios pueden ser costo, distancia y seguridad. Se utiliza una metodología de Analytic Hierarchy Process (AHP) y Technique for Order Preference by Similarity to Ideal Solution under fuzzy environment (fuzzy TOPSIS) para resolver el problema. Se aplica a dos casos: un caso a nivel regional (ubicar un depósito en alguna región que puede involucrar varios países), y otro caso a nivel local en Dubai. AHP se utiliza solo para ponderar los criterios seleccionados y fuzzy TOPSIS para evaluar las alternativas de ubicación con base en esa ponderación de criterios. Luego de calculado el ranking, para cada caso se hace un análisis de sensibilidad para determinar si cambios en los pesos de los criterios provocan cambios en el ranking, ya que la evaluación de criterios está sujeta a la subjetividad de los expertos.
En S. Y. Roh et al. (2018) se propone una solución al problema de selección de la ubicación para un depósito de suministros a utilizar luego de un desastre natural, de forma de almacenar suministros y poder entregarlos desde dicho depósito. Dado un conjunto de ubicaciones a evaluar como potenciales ubicaciones para el depósito, se utiliza una metodología consistente en aplicar Delphi modificado, fuzzy AHP y fuzzy TOPSIS para establecer un ranking de esas ubicaciones, donde el método Delphi modificado es utilizado para evaluar los criterios a utilizar a la hora de evaluar cada ubicación, y fuzzy AHP y fuzzy TOPSIS para determinar el ranking final.
Otro ejemplo de SFLP se resuelve en Awasthi et al. (2011) dónde se aborda la problemática de ubicar un centro de distribución urbano en un escenario de incertidumbre. Un escenario de incertidumbre queda definido cuando no se tiene el conocimiento o es muy complejo calcular algunas variables del modelo. Por ejemplo, en este caso, es difícil saber cómo afectará la contaminación al colocar el centro de distribución en uno u otro lugar. Para solventar este problema, y siguiendo el ejemplo, se valora con palabras y no con números, diciendo que el impacto puede ser alto, medio o bajo. Al trabajar con estas incertidumbres se aplica la teoría difusa que permite elegir la mejor decisión entre un conjunto de tomadores de decisión.

En particular, el trabajo sigue los siguientes pasos:

- Se definen los criterios que importan para tomar la decisión (seguridad, costo, impacto ambiental, entre otros).
- Se definen los lugares candidatos a colocar el centro de distribución (acá se apela a la experiencia de los tomadores de decisión)
- Se aplican las herramientas de la teoría difusa para evaluar los lugares candidatos de acuerdo a los criterios
- Se realiza un análisis de sensibilidad para evaluar la influencia de los pesos de cada criterio

Multifacility Location Problem En el MLP (Multifacility Location Problem) se ubican de forma óptima más de una instalación con respecto a la ubicación de instalaciones ya conocidas.

Location-Allocation Problem Otro tipo de problema de ubicación es el location-allocation problem. El cual consiste en ubicar un conjunto de nuevas instalaciones de forma de minimizar el coste de transporte desde las instalaciones a clientes, asignando clientes o instalaciones a las nuevas instalaciones. Dónde desde las nuevas instalaciones hacia los clientes o instalaciones asignadas solo puede haber viajes directos (se va a dicha instalación, cliente y se vuelve a la instalación de partida). Un ejemplo de este tipo de problema es el que se menciona en el trabajo de Noyan et al. (2016). En el mismo se aborda el problema de la distribución de recursos luego de un desastre natural, en particular de un terremoto. Para esto se debe determinar la ubicación y capacidad de puntos de distribución, desde los cuales las personas obtendrán los recursos, y asignar ubicaciones de demanda a dichos puntos. Además, se asignan recursos desde un centro de distribución local hacia los puntos de distribución, teniendo en cuenta la incertidumbre en la demanda y condiciones de la red de transporte. Se utiliza un modelo en dos etapas que busca conseguir niveles altos de accesibilidad y equidad, dónde en la primera etapa se ubican los puntos de distribución y en la segunda se asignan suministros y ubicaciones de demanda a los puntos de distribución previamente ubicados.

Hierarchical Location Problem En el Hierarchical Location Problem se consideran diferentes tipos de instalaciones a ubicar, las cuales no pueden ser ubicadas de forma independiente para cada nivel. Estas instalaciones son normalmente jerárquicas según el tipo de servicio que brindan. Estos sistemas jerárquicos consisten de m niveles, donde el nivel 1 es el menor nivel y el m el mayor, el nivel 0 son los nodos de demanda. Un ejemplo de este tipo de problema se resuelve en Chi et al. (2011). En este trabajo se plantea un GA (Genetic Algorithm) que determina un modelo genérico de ayuda humanitaria. El modelo definido es una red de estructura arborescente con jerarquías donde cada nodo puede ser -de mayor a menor jerarquía- hub, almacén o consumidor final de la ayuda. La raíz de cada árbol es un nodo que puede ser del tipo hub o almacén, y entre todas las raíces de todos los árboles se forma un grafo completo. Las hojas son los consumidores finales de la ayuda humanitaria, por lo tanto, no hay conectividad entre hojas. La definición formal del modelo contempla costos de abrir un hub o almacén, flujo y distancia entre nodos, costo de transporte y capacidad de almacenes. Se espera que la definición del modelo sea lo suficientemente genérica para adaptarse a distintas realidades de ciudades y espacios geográficos, aunque no es claro que sea el mejor para todos los tipos de desastres naturales.
El modelo resultante queda determinado por un algoritmo genético que de manera heurística construye el "mejor" modelo. En este caso, el GA es multiobjetivo para poder priorizar distintos intereses.

A partir de los datos de entrada necesarios para el GA (nodos, distancias, costos, que se corresponden a nuestra realidad), se retorna el grafo que indica dónde deben ser ubicados los hubs y almacenes.

A.2.2. Problemas de ruteo

Problema de ruteo de vehículos

Definición del problema Se puede definir el VRP Vehicle Routing Problem como el problema de definir rutas de costo mínimo desde un depósito a un conjunto de clientes dispersos geográficamente. Se han definido muchas variantes del VRP debido a las diferentes realidades que se presentan al aplicarlo en escenarios de la vida real. Por eso es interesante ver al VRP como una familia de problemas (Laporte, 2009).

En este contexto, definimos formalmente el problema de VRP clásico (Laporte, 2009):
Sea $G = (V, E)$ un grafo dirigido con $V = \{0, ..., n\}$ conjunto de vértices y $E = \{(i, j) : i, j \in V, i \neq j\}$ conjunto de aristas.

- El vértice 0 representa el depósito. El resto son los clientes
- El depósito cuenta con una flota de m vehículos idénticos con capacidad Q
- Cada cliente i tiene una demanda no negativa q_i
- Se define la matriz de costo A con entradas c_{ij}, que indica el costo de traslado desde el vértice i al vértice j

El VRP consiste en determinar las m rutas de los vehículos tal que comiencen y terminen en el depósito, cada cliente es visitado una única vez por un vehículo, la demanda de cada ruta no supera a Q, y el largo de la ruta no supera el límite predefinido L (Laporte, 2009).

Se aborda el caso simétrico, es decir, que $c_{ij} = c_{ji} \; \forall \; (i, j) \in E$, por lo que se trabaja con el conjunto de aristas $E = \{(i, j) : i, j \in V, \; i < j\}$.
Un análisis más detallado de algoritmos exactos para el VRP asimétrico puede encontrarse en el artículo de Toth y Vigo (2002a).
Para más información sobre las variantes más comunes del VRP, el lector puede referirse al siguiente libro de Toth y Vigo (2002b), o al artículo de Golden et al. (2008).

El VRP es una generalización del conocido Traveling Salesman Problem (TSP), pero es más difícil de resolver en la práctica. Mientras que existen algoritmos exactos capaces de resolver instancias del TSP de cientos o miles de vértices, no ocurre lo mismo para el VRP, donde los mejores algoritmos exactos solo pueden resolver instancias con aproximadamente cien vértices.

Algoritmos exactos

A continuación se presentan algunos algoritmos utilizados que resuelven de forma exacta el VRP.

Branch and Bound Una de las técnicas clásicas para resolver problemas de programación entera es branch and bound. Esta técnica se basa en particionar iterativamente un conjunto S (branching) para generar subproblemas del problema original. El algoritmo detecta si la ramificación actual no lleva a una solución óptima, y deja de explorarla ((bound)) para concentrarse en la siguiente rama.

Branch and bound fue utilizado varias veces para resolver el problema de VRP. Uno de los primeros trabajos al respecto se presenta en el artículo de Christofides y Eilon (1969).

Con el tiempo, diferentes formulaciones de la técnica branch and bound fueron surgiendo y mejorando los resultados. Por ejemplo, una mejora de este algoritmo se presenta en el artículo (Laporte et al., 1986).

Sin embargo, ninguno de estos dos algoritmos tuvieron éxito para resolver instancias que no sean pequeñas o sencillas (Laporte, 2009).

La implementación actual de branch and bound se basa en árboles de búsqueda, donde la raíz es el problema original a resolver. Con esta formulación, en el artículo de Christofides et al. (1981a) se resuelven con éxito instancias entre 10 y 25 vértices. Una mejora del algoritmo presente en el artículo de Hadjiconstantinou et al. (1995) logró resolver instancias de hasta 50 vértices.

Programación dinámica Un método poderoso para la optimización de un sistema que se puede dividir en etapas es la programación dinámica desarrollada en el artículo de Bellman (1966). El concepto principal de esta técnica se basa en el principio de optimización que se define como sigue (Luus, 2009):

> Dada una secuencia óptima de decisiones, toda sub secuencia de ella es, a su vez, óptima.

En el artículo de Christofides et al. (1981b) se resuelven instancias de entre 10 y 25 vértices utilizando esta técnica, aunque no se ha profundizado en su desarrollo.

Formulación de flujo de vehículos La formulación de flujo de vehículos de dos índices para el VRP propuesta en el artículo de Laporte y Nobert (1983) es una extensión de la formulación clásica del TSP propuesta por G. Dantzig et al. (1954).
Sea x_{ij} una variable que toma los valores $0, 1$ y 2 que indica la cantidad de veces que un vehículo recorre la arista (i, j). Entonces el problema es:

$$\min \quad \sum_{(i,j)\in E} c_{ij}x_{ij} \tag{A.1a}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{oj} = 2m, \tag{A.1b}$$

$$\sum_{i<j} x_{ik} + \sum_{j>k} x_{kj} = 2 \qquad k \in V \setminus \{0\}, \tag{A.1c}$$

$$\sum_{i,j\in S} x_{ij} \leq |S| - v(S) \qquad S \subseteq V \setminus \{0\}, \tag{A.1d}$$

$$x_{0j} \in \{0,1,2\} \qquad j \in V \setminus \{0\}, \tag{A.1e}$$

$$x_{ij} \in \{0,1\} \qquad i,j \in V \setminus \{0\} \tag{A.1f}$$

$v(S)$ es una cota inferior del número de vehículos necesarios para visitar todos los vértices de S, que es el conjunto de rutas determinado, m es la cantidad de vehículos disponibles.
Las restricciones (A.1b) y (A.1c) son restricciones sobre el grado del grafo, y la restricción (A.1d) elimina subtours e impone las restricciones de capacidad y largo de la ruta.

Heurísticas
Los algoritmos heurísticos buscan obtener una solución lo más cercana al óptimo posible en un tiempo computacional razonable.
Se presentan a continuación algunos algoritmos heurísticos conocidos para el VRP.

Algoritmo de Ahorros El Algoritmo de ahorros propuesto por Clarke y Wright en 1964 es fácil de describir e implementar, y obtiene soluciones razonablemente buenas. Esto explica su popularidad para resolver el VRP.
Se comienza con una solución inicial donde a cada cliente i se le asigna la ruta $(0, i, 0)$. Se define el ahorro como $s_{ij} = c_{i0} + c_{0j} - c_{ij}$ donde c es la función de costo entre nodos. Se calcula para cada par de clientes i y j.
En cada iteración se combina una ruta que finalice en el cliente i con otra que comience en el cliente j, maximizando el ahorro s_{ij}, siempre que la combinación de las rutas sea factible. El ahorro máximo utilizado no es considerado en la siguiente iteración. Finaliza cuando no es posible combinar rutas (Laporte, 2009).

Set Partitioning Heuristics Una heurística para abordar el VRP consiste en formular el problema como un Set Partitioning Problem (SPP). Se tiene un conjunto de rutas factibles R y cada cliente es visitado por varias rutas del conjunto. Entonces interesa hallar el subconjunto de R de costo mínimo que visite exactamente una vez a cada cliente.

Uno de los primeros ejemplos de esta metodología es el sweet algorithm propuesto en el artículo de Gillett y Miller (1974).
Por otros artículos más sofisticados referirse a Foster y Ryan (1976), Ryan et al. (1993) y Renaud et al. (1996).

Clúster-First, Route-Second Heuristics Las heurísticas del tipo Clúster-First, Route-Second constan de dos fases: en la primera se generan los clústeres, que en el contexto del VRP son grupos de clientes que estarán en la misma ruta en la solución; en la segunda fase, se calcula la ruta que visite a todos los clientes.
En Fisher y Jaikumar (1981) se propone una heurística de este tipo, donde primero se fijan m clientes semillas y se construye un clúster a partir de cada uno, sin sobrepasar la capacidad del vehículo. Esto se logra resolviendo un Generalized Assignment Problem (GAP). Entonces, la ruta de cada clúster se determina al resolver un TSP (Laporte, 2009).

Metaheurísticas

En esta sección se presentan los métodos de resolución metaheurísticos, que pueden ser aplicados para la resolución del VRP.

Una metaheurística es un método de alto nivel que guía heurísticas subyacentes que resuelven un problema dado. La familia de metaheurísticas es extensa y existen varias maneras de clasificarlas, una de ellas es la clasificación entre heurísticas poblacionales y no poblaciones (o de trayectoria). Las primeras trabajan con una población de soluciones, considerando múltiples puntos de búsqueda que evolucionan en cada iteración al combinar individuos de la población (Laporte, 2009). Las segundas constan de un algoritmo de búsqueda local que explora una trayectoria en el espacio de búsqueda.
Como ejemplos de metaheurísticas poblacionales tenemos algoritmos genéticos, algoritmos miméticos, colonia de hormigas.
Algunas metaheurísticas no poblacionales son Tabú Search, VNS, Greedy Randomized Adaptive Search Procedure (GRASP).

Una metaheurística GRASP es un procedimiento iterativo donde en cada paso se aplican dos fases: construcción y mejora.
En la fase de construcción se crea una solución factible del problema, añadiendo de a un elemento a la vez. En cada iteración, aplicando la función greedy, se define una lista de candidatos a formar parte de la solución. Aleatoriamente, se selecciona a uno de los candidatos y se lo incluye en la solución. El proceso se repite hasta obtener una solución factible.
La fase de mejora u optimización consta de una búsqueda local en las soluciones vecinas.

VNS es una metaheurística que tiene como característica el concepto de vecindad para cada individuo de la población. La vecindad debe definirse teniendo en cuenta las características del problema a resolver. VNS aplica una búsque-

da local en el vecindario, y a su vez varía los vecindarios para diversificar la búsqueda.
Si bien existen varias implementaciones del VNS, se presenta un pseudocódigo general:

Algoritmo 35 VNS

```
1: s_0 ← generarSolInicial()
2: while no se cumpla condición de parada do
3:     k ← 1
4:     while k ≤ k_max do
5:         s ← s_0
6:         s ← s.busquedaLocal(k)
7:     end while
8:     if f(s) ≥ f(s_0) then
9:         s_0 ← s
10:        k ← 1
11:    else
12:        k ← k + 1
13:    end if
14: end while
15: Return mejor solución encontrada
```

Como ejemplo de resolución del VRP aplicando una metaheurística, referimos al artículo Ferrer et al. (2018) donde se resuelve el problema de encontrar las rutas óptimas para el transporte de una cantidad fija de suministros desde nodos de suministros (aeropuertos, puertos, depósitos) hasta nodos de demanda. El modelo calcula las mejores rutas para transportar los suministros y calcula el tamaño requerido de la flota de vehículos.
El modelo toma seis criterios en cuenta: tiempo, costo, equidad, prioridad, seguridad y confiabilidad. Se trata de minimizar el rendimiento de estos criterios siguiendo una metodología de programación de compromiso. Una característica es que la seguridad es conseguida forzando a que los vehículos viajen en convoy, brindando la planificación de cada vehículo de forma individual.
Se aplica el modelo a un caso real de desastres de inundaciones en Pakistán en el año 2010. Dado que esta instancia del modelo es de dimensión alta (41 nodos y 106 aristas) se utiliza una metaheurística GRASP para su implementación.

Problema de ruteo e inventario

Definición del problema En el artículo Campbell et al. (1998) se define el Inventory Routing Problem (IRP) como la distribución repetida de un único producto, desde una única instalación, a un conjunto de clientes sobre un horizonte de planificación dado. Cada cliente consume el producto a una determinada tasa y tiene la capacidad de mantener un inventario local de ese producto hasta una

capacidad determinada. Luego, una flota de vehículos con determinada capacidad, está disponible en la instalación para distribuir el producto a los clientes. Se busca minimizar el costo promedio de distribución durante el periodo de planificación sin que los clientes se queden sin producto en ningún momento.
Hay que tomar tres decisiones: ¿cuándo servir a un cliente?, ¿cuánto entregar a un cliente cuando es servido? Y ¿qué rutas de entrega usar?

Métodos de resolución En el artículo de Sakiani et al. (2020) se resuelve un problema variante del Inventory Routing Problem (IRP), donde además se toma en cuenta el transporte de suministros de distintas características (durables y consumibles), donde se manejan los cambios dinámicos luego de un desastre siguiendo un enfoque de rolling horizon, esto es que el modelo se debe ejecutar al inicio de cada periodo, considerando las cosas que cambiaron luego de cada periodo, combina VRP con Pickup and Delivery (VRPPD) y Network Flow Problem (NFP) para obtener las ventajas de cada uno. Consiste en un Mixed Integer Linear Programming (MILP). Para resolver este problema se utiliza el algoritmo Specialized Simulated Annealing Algorithm (SSAA) el cual trata de resolver rutas factibles para cada flota en cada iteración. Para evaluar las rutas encontradas en cada iteración se utiliza el software CPLEX. Se propone un método de 3 fases para tunear los parámetros del SSAA, los métodos locales y además de los operadores de cruzamiento que proponen para el SSAA.
Se aplica sobre muestras del problema generadas aleatoriamente y sobre un caso real de estudio en de un terremoto en el este de Azerbaijan del 2012

En el artículo Alem et al. (2016) se desarrolla un modelo de flujo estocástico de dos etapas para ayudar a decidir de forma rápida como entregar ayuda humanitaria a víctimas de desastre en un conjunto de periodos de tiempo. Se seleccionan las rutas más adecuadas, qué vehículos usar, qué cantidad, en que períodos. También toma en cuenta asignación de presupuesto, dimensionado de flota de vehículos de múltiples tipos, obtención de suministros durante una respuesta, y tiempos de entrega variables. Se permite el pre posicionamiento de ayuda de emergencia antes de que el desastre ocurra. El modelo soporta también definir niveles de inventario apropiados, y se permite la acumulación de suministros a un precio alto. La incertidumbre se modela vía una programación estocástica de dos etapas basadas en escenarios. En la primera etapa se planifica el pre posicionamiento de ayuda de emergencia, así como la capacidad total de cada tipo de vehículo. En la segunda etapa se hacen decisiones operacionales de transporte (incluyendo decisiones relacionadas con la asignación de vehículos a rutas), también decisiones de inventario, escasez y obtención de suministros. La función objetivo es flexible, siendo capaz de minimizar el costo total esperado o priorizar la satisfacción de la demanda, mientras evita el inventario excesivo. Además, se muestra como el modelo se puede extender para considerar diferentes preferencias aversas al riesgo de forma de producir soluciones menos riesgosas o confiables y/o para mejorar la justicia. El modelo sigue un tipo de red genérico de

k niveles, es multiperiodo, multiproducto, multideposito, se considera el tiempo de respuesta (incluido el tiempo de transporte). Se toman decisiones con respecto al presupuesto, flota, inventario, pre posicionamiento de suministros, escasez, transporte, entre otros. El procedimiento de solución es una heurística basada en Mixed Integer Linear Programming (MILP). Las variables estocásticas son la demanda, suministros, red de transporte, presupuesto, riesgo/robustez. Hay dos tipos de nodos, nodos con demanda (puntos centrales de alivio) y nodos sin demanda (depósitos donde irán los suministros). El pre posicionamiento de suministros es permitido solo en el primer periodo. La ayuda de emergencia no utilizada en el primer periodo es utilizado en los siguientes. La obtención de ayuda de emergencia es permitida en todos los periodos, pero solo en los puntos centrales de alivio. El presupuesto se usa para transporte y obtención de ayuda de emergencia, todo el presupuesto que no se usa en un periodo se utiliza en el siguiente.

Antes de que el desastre ocurra se define la cantidad y ubicación de la ayuda de emergencia pre posicionada, así como también la capacidad total de cada tipo de vehículo, dado que estas cosas pueden llevar tiempo, ambas decisiones se consideran en la primera etapa del modelo. Luego de que el desastre ocurre, se determinan las variables de decisión de la segunda etapa, basado en la información actualizada sobre la calidad de las rutas, suministros, demanda y donaciones. Estas decisiones incluyen el flujo de ayuda de emergencia entre arcos, el número y tipo de vehículos necesarios para realizar la distribución, problemas de obtención, inventario y backlogging. Se asume que los depósitos y centros de alivio están bien ubicados de antemano. Por lo que el modelo se concentra en pre posicionamiento de ayuda de emergencia, distribución, decisiones de flota.

Se desarrolla una heurística en dos fases para resolver el problema, la primera fase resuelve una versión simplificada del modelo donde se sobre estima el número total de vehículos de la primera etapa. La segunda fase resuelve las decisiones tomadas en la primera fase correspondientes al flujo de distribución operacional y resuelve el modelo original para determinar las variables de decisión restantes.

Se presentan algunas extensiones del modelo considerando medidas de riesgo encontradas en el estado del arte para mejorar la política de cumplimiento de la demanda. Se investigó el enfoque minimax-regret, y dos medidas utilizadas en el estado del arte: semideviation, y conditional value-at-risk. Se prueba en casos de prueba de Río de Janeiro, además de probarlo en instancias random.

A.3. Problemas de localización y ruteo

A.3.1. Definición del problema

El Location Routing Problem (LRP) fue introducido como un problema combinatorio en los artículos de Or y Pierskalla (1979), Jacobsen y Madsen (1978) y Laporte y Nobert (1981).
Se tiene un conjunto de clientes con ubicación y demanda de un producto, y un

conjunto de depósitos a ubicar con diferentes capacidades. El LRP debe determinar la cantidad de depósitos a utilizar, y para cada una definir su ubicación y capacidad. Además, debe hallar las rutas óptimas desde los depósitos a los clientes para satisfacer su demanda. Siempre buscando minimizar el costo total del sistema.
La dificultad del problema viene dada por la cantidad de decisiones que se deben tomar:

- Cuántos depósitos deben ser ubicados.
- Dónde se deben ubicar los depósitos
- Qué clientes asignar a cada depósito
- Qué clientes asignar a cada ruta
- En qué orden se atienden los clientes en una ruta

La principal diferencia entre el LRP y el VRP, es que en el primero, además de definir las rutas, se debe también determinar simultáneamente la ubicación óptima de los depósitos.
También es importante destacar que en el LRP, varios clientes son atendidos en una misma ruta. No se requiere que cuando un vehículo atiende a un cliente, deba regresar al depósito para volver a atender a otro cliente. Si esto sucede, estamos ante otro tipo de problema conocido como Location Allocation Problem (LAP) (Marinakis, 2009).

A.3.2. Variantes y extensiones

Un LRP estándar se define como un problema determinista (todos los datos del problema son conocidos de antemano), estático (solo un período de planificación), discreto (las potenciales ubicaciones para las instalaciones son dadas como un subconjunto de los vértices de un grafo), de un solo escalón (los bienes son servidos desde un único nivel de instalaciones), y de un solo objetivo donde cada cliente debe ser visitado exactamente una vez para la entrega de un recurso desde una instalación, y dónde no son relevantes las decisiones de inventario (Drexl & Schneider, 2015).
Algunas variantes y extensiones al LRP son las siguientes:

- WLRP (Warehouse Location Routing Problem): Consiste en transportar bienes desde plantas a uno o más depósitos y desde estos depósitos hacia clientes. El número, tamaño, y la ubicación de los depósitos, así como la asignación de clientes a depósitos y las rutas de entrega, deben ser determinadas con el objetivo de minimizar el costo total. El costo total consiste en el costo de almacenamiento y el costo de transporte. El costo de almacenamiento se compone del costo de apertura del depósito (costo fijo) y el costo de rendimiento que depende de la cantidad de bienes transportados hasta un depósito en particular.

- OLRP (Open Location Routing Problem): El problema denominado Open Location Routing Problem tiene como variante el comportamiento de los vehículos: comienzan en un depósito, atienden a los clientes, y luego no necesitan regresar al depósito, sino que desde el lugar en el que se encuentran continúan atendiendo a otro grupo de clientes.
- CLRP (Covering Tour Location Routing Problem with Replenishment): Es una extensión del LRP que introduce restricciones de tiempo en los servicios, reposición en depósitos intermedios, y movilidad de los clientes en una distancia acotada (se recorre a pie).

Por una revisión más completa de variantes del LRP se pueden ver Drexl y Schneider (2015) y Prodhon y Prins (2014b).

Clasificación del LRP según la perspectiva del problema

Una posible clasificación según Hassanzadeh et al. (2009) que toma en cuenta la perspectiva del problema es la siguiente:

- Nivel jerárquico: El cual puede ser de una etapa o dos etapas. De una etapa implica un tipo de instalación, mientras que de dos etapas implica dos tipos de instalaciones.
- Naturaleza de la demanda: Sí, es determinista o estocástica.
- Número de instalaciones: Sí, se trata de una sola instalación o múltiples.
- Tamaño de la flota de vehículos: Un solo vehículo o múltiples.
- Capacidad de los vehículos: Con o sin restricción en capacidad en los vehículos.
- Capacidad de las instalaciones: Con o sin restricción en la capacidad de las instalaciones.
- Capa de instalaciones: Instalaciones primarias o secundarias. Las primarias corresponden a fábricas o plantas, mientras que las secundarias corresponden a depósitos o almacenes. Por lo general, ya se conoce la ubicación de las instalaciones primarias y están fijas, mientras que para las secundarias, hay que determinar su ubicación, número y las rutas desde esas instalaciones.
- Horizonte de planificación: Si se considera un solo periodo o múltiples en los que se resuelve el problema.
- Restricciones de tiempo: En este caso se tienen restricciones no especificadas sin fechas límite, restricciones de tiempo suaves con fechas límites y holgura en las fechas límite y restricciones de tiempo fuertes con fechas límites estrictas.

- Función objetivo: Sí, se considera un solo objetivo o múltiples.
- Tipo del modelo de datos: Sí, se utilizan datos de prueba reales o hipotéticos.

En la Tabla A.1 se muestra esta clasificación para algunos de los puntos anteriores para los distintos trabajos abordados en este documento.

Trabajo	Demanda	Cap.vehículos	Cap.Instalacion	Restr. tiempo
(Rath & Gutjahr, 2014)	Determinista	Limitado	Limitado	Fuertes
(Veenstra et al., 2018b)	Determinista	No Limitado	No Limitado	Fuertes
(Nedjati et al., 2017)	Estocástica	Limitado	Limitado	Flexibles
(Liu et al., 2019b)	Determinista	Limitado	Limitado	Fuertes
(Wang et al., 2014)	Estocástica	Limitado	Limitado	Sin restricción
(Liang et al., 2019)	Estocástica	No Limitado	Limitado	Sin restricción
(Ahmadi et al., 2015)	Determinista	Limitado	Limitado	Flexibles
(Mingang et al., 2009)	Determinista	No limitado	Limitado	Fuertes

Tabla A.1: Clasificación respecto a la perspectiva del problema.

Todos los trabajos consideran múltiples instalaciones y vehículos, son de un único periodo, múltiples objetivos menos (Liang et al., 2019) que es de un solo objetivo, y resuelven el problema aplicado a datos del mundo real y/o hipotéticos. En cuanto al nivel de jerarquía y tipo de capa de instalación se analizará en la siguiente sección.

Clasificación del LRP según diagrama de capas

Otra posible clasificación propuesta en Hassanzadeh et al. (2009) consiste en considerar el diagrama de capas de las instalaciones y usuarios o clientes junto al modo de distribución usado. Es decir, si se puede diferenciar las instalaciones en capas y como es la distribución entre estas capas. Se puede tener por ejemplo instalaciones primarias, secundarias y usuarios como se mencionó en la clasificación anterior, en este caso se tienen tres capas.
El modo de distribución refiere a como se distribuyen los recursos desde una capa a la siguiente. Se distinguen dos formas:

- $M = R$ implica que todos los viajes de la capa deben ser del tipo retorno, esto es que los vehículos van hacia un depósito o usuario de la siguiente capa y vuelven.
- $M = T$ implica que los viajes de los vehículos de una capa visitan varias instalaciones o usuarios de la capa siguiente y vuelven a la instalación de la que partieron.

Para representar esta clasificación se utiliza la siguiente expresión:
$\lambda/M1/M2/M3/.../M_{\lambda-1}$ donde λ es el número de capas y los M_i representan

los modos de distribución mencionados anteriormente.

Para el caso del LRP, las decisiones de ubicación deben ser para al menos una capa, ya que de lo contrario se obtiene un VRP. Además, para los viajes, al menos uno de ellos debe ser de tipo T, de otra forma sería un problema puramente de ubicación de instalaciones. En la Tabla A.2 se muestra esta clasificación para los trabajos abordados en este documento.

Trabajo	Tipo	Capa 1	Capa 2	Capa 3	Ubicación
(Rath & Gutjahr, 2014)	3/R/T	Plantas	Depósitos	Clientes	Capa 2
(Veenstra et al., 2018b)	3/T/T	Farmacia (1)	Lockers	Pacientes	Capa 2
(Nedjati et al., 2017)	2/T	Depósitos	Víctimas de desastres		Capa 1
(Liu et al., 2019b)	2/T	Centros de distr.	Áreas de desastres		Capa 1
(Wang et al., 2014)	2/T	Centros de distr.	Áreas de desastres		Capa 1
(Ahmadi et al., 2015)	2/T	Centros de distr.	Áreas de desastres		Capa 1
(Mingang et al., 2009)	2/T	Depósitos	Áreas de desastres		Capa 1

Tabla A.2: Clasificación respecto al diagrama de capas.

A.3.3. Modelado

A continuación presentamos un ejemplo de una formulación de un LRP. En particular corresponde al problema explicado en la Sección A.4.6.
Algunas consideraciones:

- La duración de ruta de un vehículo consiste en la suma del tiempo de viaje y tiempo de servicio total.
- Una ruta de un vehículo es factible si empieza y termina en el depósito y si no excede la duración máxima de ruta.

$G = (V, A)$ grafo dirigido, V conjunto de nodos, A conjunto de arcos
P = conjunto de Pacientes
L = conjunto de posibles ubicaciones de lockers
$V = 0 \cup P \cup L$ dónde 0 es la farmacia.
K = conjunto de vehículos que visitan pacientes.
M = conjunto de vehículos que visitan lockers.
r_j = distancia de cubrimiento del casillero j de L
F_j = costo fijo de abrir el casillero j de L
c_{ij} = costo del arco (i, j)
d_{ij} = distancia del arco (i, j)
t_{ij} = tiempo de recorrido del arco (i, j)
Φ = factor de penalización de visitar a domicilio, por las razones que se mencionaron anteriormente.
$T1$ = máxima duración de ruta para los vehículos de K, o sea la suma de las duraciones de ruta de los vehículos de K no puede superar este parámetro.

$T2$ = máxima duración de ruta para los vehículos de M, o sea la suma de las duraciones de ruta de los vehículos de K no puede superar este parámetro.
s_i = tiempo de servicio para $i \in V$, $s_i > 0$ para todo $i \in P \cup L$ y $s_0 = 0$
$w_i = \{0, 1\}$ indica si el paciente $i \in P$ está asignado al depósito 0
$z_i = \{0, 1\}$ indica si el paciente $i \in P$ está asignado a algún locker
$u_{ij} = \{0, 1\}$ indica si el paciente $i \in P$ está asignado al locker j
$v_i = \{0, 1\}$ indica si el locker $i \in L$ está abierto o no
$x_{ij}^k = \{0, 1\}$ indica si el vehículo k de K recorre el arco $(i, j) \quad i, j \in P \cup 0$
$y_i^k = \{0, 1\}$ indica si el vehículo k de K visita el nodo $i \in P \cup 0$
$h_{ij}^k \in \mathbb{N}$ indica la carga del vehículo $k \in K$ al atravesar el arco $(i, j) \quad i, j \in P \cup 0$
$p_{ij}^k = \{0, 1\}$ indica si el vehículo k de M recorre el arco $(i, j) \quad i, j \in L \cup 0$
$q_i^k = \{0, 1\}$ indica si el vehículo k de M visita el nodo $i \in L \cup 0$
$g_{ij}^k \in \mathbb{N}$ indica la carga del vehículo $k \in M$ al atravesar el arco $(i, j) \quad i, j \in L \cup 0$

Formulación matemática:

$$\min \quad \sum_{i\in P\cup\{0\}}\sum_{j\in P\cup\{0\}}\sum_{k\in\mathcal{K}}\phi c_{ij}x_{ij}^{\mathrm{k}} + \sum_{i\in\mathcal{L}\cup\{0\}}\sum_{j\in L\cup\{0\}\mathrm{l}}\sum_{k\in\mathcal{M}} c_{ij}p_{ij}^{\mathrm{k}} + \sum_{i\in\mathcal{L}} F_i\nu_i \quad (1)$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{L}} u_{ij} = z_i \qquad i\in\mathcal{P}, \quad (2)$$

$$w_i + z_i = 1 \qquad i\in\mathcal{P}, \quad (3)$$

$$u_{ij} \le v_j \qquad i\in\mathcal{P}, j\in\mathcal{L}, \quad (4)$$

$$u_{ij}d_{ij} \le r_j v_j \qquad i\in\mathcal{P}, j\in\mathcal{L}, \quad (5)$$

$$r_j v_j \le d_{ij} + z_i M \qquad i\in\mathcal{P}, j\in\mathcal{L}, \quad (6)$$

$$\sum_{j\in P\cup\{0\}} x_{ij}^k + \sum_{j\in P\cup\{0\}} x_{ji}^k = 2y_i^k \qquad i\in\mathcal{P}, k\in\mathcal{K}, \quad (7)$$

$$\sum_{j\in P} x_{0j}^k + \sum_{j\in P} x_{j0}^k = 2y_0^k \qquad k\in\mathcal{K}, \quad (8)$$

$$\sum_{k\in\mathcal{K}} y_i^k = w_i \qquad i\in\mathcal{P}, \quad (9)$$

$$\sum_{j\in P\cup\{0\}} x_{ij}^k = \sum_{j\in P\cup\{0\}} x_{ji}^k \qquad i\in\mathcal{P}, k\in\mathcal{K}, \quad (10)$$

$$\sum_{j\in P}\sum_{k\in\mathcal{K}} h_{0j}^k = \sum_{i\in P} w_i, \quad (11)$$

$$\sum_{i\in P\cup\{0\}} h_{ij}^k - \sum_{i\in P\cup\{0\}} h_{ji}^k = y_j^k \qquad j\in\mathcal{P}, k\in\mathcal{K}, \quad (12)$$

$$h_{ij}^k \le (|P|-1)\, x_{ij}^k \qquad i\in P, j\in P\cup\{0\}, k\in K, \quad (13)$$

$$h_{0j}^k \le |P|\, x_{0j}^k \qquad j\in P\cup\{0\}, k\in K, \quad (14)$$

$$\sum_{i\in P\cup\{0\}}\sum_{j\in P\cup\{0\}} t_{ij}x_{ij}^k + \sum_{i\in P\cup\{0\}}\sum_{j\in P\cup\{0\}} s_j x_{ij}^k \le T_1 \qquad k\in\mathcal{K}, \quad (15)$$

$$\sum_{j\in\mathcal{L}\cup\{0\}} p_{ij}^k + \sum_{j\in\mathcal{L}\cup\{0\}} p_{ji}^k = 2q_i^k \qquad i\in\mathcal{L}, k\in\mathcal{M}, \quad (16)$$

$$\sum_{j\in L} p_{0j}^k + \sum_{j\in L} p_{j0}^k = 2q_0^k \qquad k\in\mathcal{M}, \quad (17)$$

$$\sum_{k\in\mathcal{M}} q_i^k = \nu_i \qquad i\in\mathcal{L}, \quad (18)$$

$$\sum_{j\in L\cup\{0\}} p_{ij}^k = \sum_{j\in\mathcal{L}\cup\{0\}} p_{ji}^k \qquad i\in\mathcal{L}, k\in\Lambda 4, \quad (19)$$

$$\sum_{j\in L}\sum_{k\in\mathcal{M}} \mathrm{g}_{0j}^k = \sum_{i\in L}\nu_i, \quad (20)$$

$$\sum_{i \in \mathcal{L} \cup \{0\}} g_{ij}^k - \sum_{i \in \mathcal{L} \cup \{0\}} g_{ji}^k = q_j^k \qquad j \in \mathcal{L}, k \in \mathcal{M}, \tag{21}$$

$$g_{ij}^k \leq (|L| - 1)\, p_{ij}^k \qquad i \in L, j \in L \cup \{0\}, k \in M, \tag{22}$$

$$g_{0j}^k \leq |L|\, p_{0j}^k \qquad j \in L \cup \{0\}, k \in M, \tag{23}$$

$$\sum_{i \in \mathcal{L} \cup \{0\}} \sum_{j \in \mathcal{L} \cup \{0\}} t_{ij} p_{ij}^k + \sum_{i \in \mathcal{L} \cup \{0\}} \sum_{j \in \mathcal{L} \cup \{0\}} s_j p_{ij}^k \leq T_2 \qquad k \in \mathcal{M}, \tag{24}$$

$$z_i, w_i \in \{0, 1\} \qquad i \in \mathcal{P}, \tag{25}$$

$$u_{ij} \in \{0, 1\} \qquad i \in \mathcal{P}, j \in \mathcal{L}, \tag{26}$$

$$v_i \in \{0, 1\} \qquad i \in \mathcal{L}, \tag{27}$$

$$x_{ij}^k \in \{0, 1\} \qquad i, j \in \mathcal{P} \cup \{0\}, k \in \mathcal{K}, \tag{28}$$

$$y_i^k \in \{0, 1\} \qquad i \in \mathcal{P} \cup \{0\}, k \in \mathcal{K}, \tag{29}$$

$$h_{ij}^k \geq 0 \qquad i, j \in \mathcal{P} \cup \{0\}, k \in \mathcal{K}, \tag{30}$$

$$p_{ij}^k \in \{0, 1\} \qquad i, j \in \mathcal{L} \cup \{0\}, k \in \mathcal{M}, \tag{31}$$

$$q_i^k \in \{0, 1\} \qquad i \in \mathcal{L} \cup \{0\}, k \in \mathcal{M}, \tag{32}$$

$$g_{ij}^k \geq 0 \qquad i, j \in \mathcal{L} \cup \{0\}, k \in \mathcal{M} \tag{33}$$

Las restricciones implican lo siguiente:

- (1): El primer sumando representa el costo de las rutas a los pacientes, el segundo el costo de las rutas a los lockers, y el tercero representa el costo de abrir los lockers.
- (2): Si un paciente está asignado a un locker, está asignado a un único locker.
- (3): Asegura que un paciente este asignado o bien a un locker o bien al depósito.
- (4): Un paciente puede ser asignado a un locker si está abierto.
- (5): Si un paciente está asignado a un locker entonces está dentro del área de cubrimiento de ese locker.
- (6): Si un paciente está dentro del área de cubrimiento de algunos lockers, debe estar asignado a alguno de esos lockers.
- (7): Si un vehículo visita un paciente, entra y sale una única vez. Y si un vehículo no visita a un paciente, su nodo no es visitado.
- (8): Solo vehículos que se usan pueden visitar pacientes.
- (9): Si un paciente es asignado al depósito, entonces solamente un vehículo visita su nodo, de otra forma ningún vehículo lo visita.
- (10): Conservación de flujo.

- (11): Asegura que la carga total que sale del depósito es igual a la cantidad de pacientes asignados a el.
- (12): Asegura que si un paciente es asignado a un vehículo dado, la diferencia de la carga del vehículo entrando al nodo y la de saliendo es exactamente uno.
- (13) y (14): Junto a 11 y 12 aseguran que la solución esté conectada.
- (15): Asegura que se respeta la duración máxima para las rutas.
- (16)-(24): Similares a las restricciones (7) - (15) pero aplicando a los lockers.
- (23) - (33): Definen la naturaleza y rango de las variables

A.3.4. Métodos de resolución

Metodos exactos

- AECA (Adaptive epsilon-constraint algorithm): AECA se utiliza para resolver problemas multiobjetivo, requiere resolver una secuencia de problemas de optimización de objetivo único. Se elige una de las funciones objetivo para ser la función objetivo del problema de objetivo único. Las otras funciones objetivos se restringen con límites apropiados, los cuales son gradualmente determinados por el AECA. AECA determina la frontera de Pareto exacta si resuelve el subproblema de objetivo único de forma exacta, de lo contrario se obtiene una aproximación de dicha frontera. Como es el caso de Rath y Gutjahr (2014).
- Hierarchical sequence method: También se utiliza para resolver problemas multiobjetivo. El método ordena los objetivos en orden de importancia y obtiene la solución óptima para el objetivo más importante. Luego se obtiene la solución óptima según el siguiente objetivo, pero garantizando que la solución sea óptima con respecto al primer objetivo (Liu et al., 2019b).
- CPLEX: Es un solver para problemas de programación lineal, MILP, y programación cuadrática. Muchos trabajos usan este software para calcular la solución exacta y compararla con la solución de la heurística o metaheuristica implementada. Por ejemplo en Veenstra et al. (2018b).

Heurísticas

- Algoritmos de dos etapas: Varios trabajos utilizan algoritmos de dos etapas para resolver el problema, donde en la primer etapa se resuelve la ubicación de las instalaciones y en la segunda etapa se resuelve el ruteo de recursos. Como por ejemplo en Mingang et al. (2009)

- Algoritmo greedy: Un algoritmo greedy es un algoritmo simple e intuitivo para resolver problemas de optimización. El algoritmo toma la decisión óptima en cada paso e intenta encontrar el óptimo global para resolver el problema. En el caso de Liu et al. (2019b) se utilizan algoritmos greedy como parte de una heurística híbrida para una asignación de ayuda de manera justa y para asignar nodos de demanda a los centros de distribución.

Metaheurísticas

- Ant Colony Optimization Algorithm: Es una técnica probabilística para solucionar problemas computacionales que pueden reducirse a buscar los mejores caminos o rutas en grafos. Basado en el comportamiento de las hormigas para encontrar el camino más corto entre el hormiguero y una fuente de comida. En Mingang et al. (2009) se utiliza este algoritmo para resolver la función heurística que calcula las rutas con ciertos objetivos. En Liu et al. (2019b) se utiliza este algoritmo para planificar la entrega de ayuda a los nodos de demanda.

- VNS (Variable Neighborhood Search): Definido en la Sección A.2.2. En Veenstra et al. (2018b) se utiliza el VNS para mejorar las rutas que se van calculando luego de ubicar los lockers. En Rath y Gutjahr (2014) se utiliza el VNS para encontrar una solución heurística al subproblema MDMT-CTOP (Multi-depot, multiple trip capacitated team orienteering problem). En Ahmadi et al. (2015) se utiliza el VNS para resolver el LRP para instancias grandes.

- NSGAII (Nondominated Sorting Genetic Algorithm II): Es un algoritmo evolutivo para resolución de problemas de optimización multi objetivo. Un algoritmo evolutivo es un método meta heurístico basado en población para hallar soluciones de calidad a problemas de optimización complejos.

  Partiendo de un conjunto de soluciones (*población*) generadas en el proceso de Inicialización, el algoritmo itera sobre las generaciones mientras no se cumpla el criterio de parada definido. En cada generación, algunos individuos (según la Selección) de la población se modifican de acuerdo a los operadores evolutivos de Cruzamiento y Mutación. La forma de Reemplazo definida indica los individuos que son reemplazados.
  Cada individuo tiene un valor de *fitness*, que depende del problema y determina la aptitud del individuo para resolverlo.
  Así la población evoluciona hasta cumplirse el criterio de parada. Entonces se devuelve la mejor solución.

  Usos de este algoritmo se pueden encontrar en Wang et al. (2014) y en Nedjati et al. (2017).

- Non-dominated Sorting Differential Evolution (NSDE): Es también un algoritmo para resolución de problemas de optimización multiobjetivos (Wang et al., 2014).

- Simulated Annealing: Es un algoritmo de búsqueda metaheurística para problemas de optimización global; el objetivo general de este tipo de algoritmos es encontrar una buena aproximación al valor óptimo de una función en un espacio de búsqueda grande. El nombre e inspiración viene del proceso de recocido del acero y cerámicas, una técnica que consiste en calentar y luego enfriar lentamente el material para variar sus propiedades físicas. El calor causa que los átomos aumenten su energía y que puedan así desplazarse de sus posiciones iniciales (un mínimo local de energía); el enfriamiento lento les da mayores probabilidades de recristalizar en configuraciones con menor energía que la inicial (mínimo global) (Gutiérrez et al., 1998). En cada iteración, el método de recocido simulado evalúa algunos vecinos del estado actual s y probabilísticamente decide entre efectuar una transición a un nuevo estado s' o quedarse en el estado s. En Veenstra et al. (2018b) se utiliza Simulated Annealing para establecer un criterio para aceptar o no una solución construida con el VNS explicado a continuación.

- Otras metaheuristicas utilizadas comúnmente para la resolución del LRP son Tabú Search y GRASP (Marinakis, 2009).

Por una revisión en mayor profundidad acerca de métodos de resolución exactos, heurísticos y metaheurísticos sobre el LRP se puede consultar Marinakis (2009) y Hassanzadeh et al. (2009).

A continuación se pretende mostrar la aplicación de un método de resolución. En particular, continuamos con el caso presentado en la Sección A.4.6 y veremos la resolución al modelo presentado en la Sección A.3.3.

Se plantea una metaheurística que iterativamente modifica la solución actual cambiando el conjunto de lockers abiertos, modifica las rutas en consecuencia, y se aplica un procedimiento VNS para mejorar dichas rutas. La estructura de la metaheurística es la siguiente:

Algoritmo 36 Metaheuristic

$s \leftarrow generarSolInicial()$
while no se cumpla condición de parada **do**
  $s' \leftarrow s$
  Cambiar el conjunto de lockers abiertos en s'
  Actualizar las rutas en s'
  Aplicar VNS a las rutas en s'
  if $Acepta(s', s)$ **then**
    $s \leftarrow s'$
  end if
end while
return mejor s

En este punto, interesa centrarnos en el procedimiento VNS de la metaheurística. Los detalles del resto de las operaciones se pueden ver en el artículo.

El VNS propuesto tiene el siguiente pseudocódigo:

Algoritmo 37 VNS

```
R : conjuntoderutas
improveOverall ← 1
while improveOverall = 1 do
    improveOverall ← 0
    for each k ∈ {1, ..., 6} do
        improve ← 1
        while improve = 1 do
            improve ← 0 R' ← R R' ← Operator_k(R')
            if cost(R) > cost(R') then
                improve ← 1
                improveOverall ← 1 R ← R'
            end if
        end while
    end for
end while
```

Se tiene un conjunto de rutas (de lockers o pacientes). Se aplica una búsqueda local con un operador hasta que no sea posible mejorar la solución, en este punto se cambia de operador.
Si se aplican todos los operadores y se mejora la solución, entonces se vuelve a repetir el procedimiento, si no se termina.
Los seis operadores son los siguientes y se aplican en orden: $1-0-Exchange$, $1-1-Exchange$, $Intra-2-Opt$, $Inter-2-Opt$, $Intra-3-Opt$ y $Intra-3-Opt'$.
$1-0-Exchange$ selecciona aleatoriamente una ubicación de la ruta y lo ubica en la mejor posición (donde menos aumenta el valor objetivo).
$1-1-Exchange$ selecciona aleatoriamente dos ubicaciones y las intercambia si mejora la solución.
$Intra-2-Opt$ se aplica de forma separada en cada ruta. Para cada combinación de 2 arcos, se eliminan los mismos y se trata de reconectar la ruta resultante. Si se obtiene una mejor solución, entonces esos arcos son removidos.
$Inter-2-Opt$ es análogo a $Intra-2-Opt$ pero con arcos provenientes de diferentes rutas.
$Intra-3-Opt$ es análogo a $Intra-2-Opt$ pero eliminando 3 arcos.
$Intra-3-Opt'$ generaliza este concepto y elimina una sección de una ruta y la reubica en el mejor lugar posible en otra ruta.

A.4. Casos de estudio

En el desarrollo del presente documento fueron analizados varios artículos en los que se presentan diferentes escenarios del LRP. Algunos son introducidos en esta sección.

A.4.1. LRP para distribución de ayuda luego de un tifón

En el trabajo de Mingang et al. (2009) se divide el LRP en dos subproblemas: problema de ubicación de instalaciones de emergencia y problema de ruteo de recursos de emergencia, se establece un modelo para el LRP con el objetivo de minimizar el costo total, el cual incluye el costo fijo, el costo de transporte y el costo de pérdidas por desastres. Se utiliza un algoritmo heurístico de dos etapas, donde la primera etapa utiliza el método de Clustering para resolver el primer problema con restricciones de tiempo, demanda y peso de las distancias de las instalaciones a los puntos de demanda, de forma de seleccionar las ubicaciones para las instalaciones, luego en la segunda etapa se resuelve el segundo problema usando Ant Colony Algorithm considerando las restricciones de tiempo de transporte, probabilidad de pasaje seguro en las rutas y distancias de transporte.

A.4.2. Transporte de bienes, extensión WLRP, modelo $3/R/T$

En el artículo de Rath y Gutjahr (2014) se resuelve un problema basado en el Warehousing Location Routing Problem (WLRP).

El problema se puede tratar como un problema de un solo producto. A diferencia del WLRP estándar, el suministro total puede ser menor que la demanda total.

En este problema cada depósito tiene una flota de vehículos. El costo de adquisición de los vehículos, así como el costo de operación (combustible, choferes y otros) se incluyen en el costo de apertura del depósito. Por otro lado, los vehículos tienen una determinada capacidad, pero está permitido que regresen al depósito para recargarse de bienes una cantidad indefinida de veces, siempre y cuando se respete un tiempo de manejo total máximo. Por lo que para el transporte de bienes desde los depósitos existe un tiempo total máximo de viaje para cada depósito para toda su flota. En cuanto al transporte de bienes desde las plantas a los depósitos, se asume que hay vehículos y choferes suficientes. El costo de transporte depende de la cantidad de bienes a entregar.

La flota de vehículos en cada depósito se considera homogénea. Se utilizan para entregar suministros a los clientes. Tienen una capacidad límite. Cuando el vehículo termina el viaje regresa al depósito. Pueden cargar suministros para

muchos clientes. Puede retornar al depósito para reabastecerse y seguir entregando, siempre y cuando no se supere el tiempo límite de viaje. Solo puede visitar una vez a cada cliente, los cuales se sirven completamente, solo si no hay suministros suficientes el último queda servido parcialmente.

Se utiliza un enfoque multiobjetivo. El primer objetivo minimiza los costos fijos de depósitos y vehículos (costos de apertura de depósitos, costo de obtención y operación de vehículos). El segundo objetivo minimiza el presupuesto asignado para el costo operativo (costo de transporte de las plantas a los depósitos más el costo de almacenamiento de los bienes en los depósitos). El tercer objetivo maximiza la demanda cubierta, este último objetivo entra fuertemente en conflicto con los otros dos.

Para modelar el problema se propone una formulación MILP y una relajación del mismo. Se proponen dos métodos de solución, uno exacto y una técnica heurística-matemática a la cual llaman constraint pool heuristic, la cual usa una formulación MILP de base y un algoritmo VNS para agregar iterativamente restricciones generadas heurísticamente.

Para tratar con la parte multiobjetivo del problema se utiliza el AECA (adaptive epsilon-constraint algorithm) tanto para el método de solución exacta como para el método heurístico. En este problema se limita la primera y segunda función objetivo y se considera el subproblema de objetivo único con respecto a la tercera función objetivo. Obteniéndose una nueva formulación del problema, de un solo objetivo.

Como subproblema se resuelve el MDMT-CTOP (multi depot, multi trip capacitated team orienteering problem) mediante un procedimiento VNS.

A.4.3. OLRP para distribución de ayuda luego de un terremoto

En Wang et al. (2014) se considera la variante OLRP. El trabajo plantea el problema de determinar una red de distribución de ayuda en un escenario posterior a un terremoto con las siguientes particularidades:

- Vehículos heterogéneos con diferente capacidad, velocidad y costo de movilidad
- Para cada camino entre los nodos de la zona, se define la confianza como la probabilidad de atravesar ese camino
- Cada cliente puede ser atendido más de una vez si su demanda lo requiere
- Los vehículos no regresan al depósito luego de atender a los clientes, sino que desde donde terminaron comienzan a atender a otro grupo

Se deben determinar los centros de distribución que se deben abrir desde un conjunto de candidatos, asignar los vehículos a los centros de distribución y definir la planificación de las rutas.
El problema se plantea como multiobjetivo, ya que interesa minimizar el tiempo de los viajes, minimizar el costo total, y maximizar la mínima confianza de las rutas de los vehículos.
Para resolver este modelo se presentan los algoritmos metaheurísticos de NSGAII y Non-dominated Sorting Differential Evolution (NSDE).

A.4.4. Multi-depot LRP para distribución de ayuda en dos etapas considerando fallas en la red y tiempo de ayuda estándar

En Ahmadi et al. (2015) se propone un modelo de logística humanitaria con las siguientes consideraciones:

1. Se considera posible la destrucción de la red vial en tiempo real (común en terremotos, por ejemplo), a través de un sistema de información geográfico.

2. Se aplica la restricción standard relief time (SRT), que son las primeras 12hs desde el desastre y se consideran críticas.

3. Se considera estándar la cantidad y tipo de ayuda necesaria para cada persona, y se penaliza si no se cumple la demanda.

Se divide el problema en dos niveles: el primero es llamado estratégico porque aplica antes que ocurra el desastre, y tiene como fin determinar cuántos y dónde deben ser ubicados los centros de distribución. El segundo nivel, operacional, aplica en la fase de respuesta al desastre y se encarga de definir la cantidad y ubicación de depósitos locales, cantidad de ayuda a enviar a las áreas afectadas, ruteo de vehículos y su asignación a los depósitos locales.
Para la resolución del modelo en una instancia real, en este caso un terremoto en San Francisco, se desarrolla un algoritmo VNS.

A.4.5. CLRPR de distribución de ayuda luego de un terremoto

En Nedjati et al. (2017) se presenta el (CLRPR), este artículo se centra en el sistema de distribución de ayuda en un escenario posterior a un terremoto.
El modelo del problema toma en cuenta un límite de tiempo en el que se presta el servicio de distribución. En ese período, los vehículos tienen el siguiente comportamiento: comienzan su ruta desde un determinado depósito y atienden a los clientes asignados, reponen inventario en cualquier otro depósito y atienden a otro grupo de cliente, y así hasta terminar el recorrido en cualquier depósito.
Se deben determinar los depósitos a abrir de un conjunto de candidatos.
Además, se diferencian tres tipos de clientes a atender: los que están ubicados

en las rutas de los vehículos, los que no están en las rutas, pero se encuentran a una distancia que pueden recorrer a pie, y finalmente los nodos perdidos son el resto de los clientes. Se busca minimizar los clientes no atendidos y el tiempo de espera de los clientes atendidos, cumpliendo con las restricciones de tiempo establecidas.
Como solución se presenta el algoritmo metaheurístico Nondominated Sorting Genetic Algorithm II (NSGAII) utilizado para resolver problemas multiobjetivo.

A.4.6. Entrega de medicación a pacientes

Otro problema que involucra la resolución de un problema de localización y ruteo es el que se menciona en el artículo de Veenstra et al. (2018b). El mismo trata sobre la entrega de medicación a pacientes, en particular cuando estos deben repetir medicación.
Se tiene un depósito central (por ejemplo una farmacia local) desde la cual se envían los medicamentos a los pacientes. Hay dos tipos de entrega: de forma directa a los pacientes y entrega a lockers ubicados en determinadas ubicaciones. Por temas prácticos, las entregas a los pacientes y a los lockers se consideran de forma separada. Estos lockers tienen un área de cubrimiento, los pacientes que estén dentro de dicha área se servirán los medicamentos desde ese locker.
Se tiene un conjunto potencial de ubicaciones de lockers a abrir, mientras que los pacientes que no queden en el área de ningún locker se les entrega la medicación a domicilio. El problema consiste en determinar qué lockers abrir y determinar qué rutas seguir de forma de minimizar los costos.

El problema se modela como un Integer Linear Programming (ILP) donde la función objetivo es la suma de los costos de las rutas y los costos de apertura de lockers. Para resolver el problema se propone una heurística híbrida con Variable Neighborhood Search (VNS) y Simulated Annealing. En cada iteración de la heurística: se cambia el conjunto de lockers a abrir, se actualizan las rutas, y se aplica VNS para mejorar dichas rutas. Luego la solución se acepta si es mejor a la de la iteración anterior. Si no lo es, se utiliza Simulated Annealing para determinar si la solución es aceptada de todas formas. Para determinar qué lockers abrir se utilizan distintos procedimientos. Estos procedimientos son los siguientes: abrir un locker (se abre uno de los lockers cerrados), cerrar un locker (se cierra uno de los lockers abiertos), estos dos procedimientos son con base en tres métodos: random, basado en reducción de costos y basado en el número de pacientes cubiertos. El tercer procedimiento consiste en mover un locker, con determinada probabilidad p se toma un locker para cerrar y otro que esté en el rango de cubrimiento del primero para abrir con la misma probabilidad, con probabilidad $1 - p$ se toma uno para abrir y otro para cerrar, donde el locker para cerrar tiene que estar dentro del rango de cubrimiento del que se abre. En ambos casos, si no existe un locker dentro del rango de cubrimiento del otro, se selecciona el segundo de forma aleatoria.
Se muestra mediante un análisis de sensibilidad que la utilización de estos procedimientos es mejor que utilizar una forma aleatoria. Se proponen seis operadores para la búsqueda local a la hora de mejorar las rutas en el VNS.

Además, se realiza un análisis sobre el impacto del valor de ciertos parámetros en la solución final, como son el área de cubrimiento de los lockers, costo de apertura de cada locker y penalización por entrega a domicilio.
De esta forma se está resolviendo un problema de localización (qué lockers abrir) y ruteo (determinación de rutas de entrega) de forma simultánea.

A.4.7. Distribución de ayuda de manera justa luego de un terremoto

Por otro lado, en el siguiente trabajo de Liu et al. (2019b) se resuelve el LRP para el caso de escasez de ayuda en etapas tempranas luego de un terremoto, llevando a cabo una asignación y distribución justa de la ayuda.
Se formula una función de pérdida para cada nodo de demanda, la cual considera la severidad del desastre junto a la vulnerabilidad de cada nodo de demanda, esta vulnerabilidad depende de la ubicación, composición del personal, extensión del desastre, capacidad de reducción del desastre, y la sensibilidad ante desastres del nodo de demanda. Se propone un modelo multiobjetivo para el problema utilizando el método lexicográfico para ordenación óptima de objetos donde se consideran restricciones de tiempo, daños parciales a las carreteras, entrega de ayuda multimodal, severidad del desastre, y vulnerabilidad de los nodos de demanda cuando la demanda no se satisface. Los objetivos del modelo son:

- Minimizar la pérdida máxima en los nodos de demanda, con esto se mide si la asignación de ayuda es justa.
- Minimizar la pérdida total de todos los nodos, lo cual es un índice de la utilidad de la ayuda. Se utiliza la misma función de pérdida.
- Minimizar el tiempo máximo requerido por los nodos de demanda para recibir la ayuda.

Algunas características del problema incluyen suponer que hay escasez de ayuda luego del terremoto, si el nodo de demanda es una isla entonces se entrega ayuda con helicópteros, en otro caso se entrega con vehículos, para los nodos de demanda grandes, se utiliza la segmentación de demanda simultáneamente para entrega directa y entrega común, los nodos de demanda pueden ser servidos muchas veces, el equipamiento de distribución de ayuda es limitado y este equipamiento empieza en los centros de distribución y vuelven al mismo nodo luego.
Se tienen entonces un conjunto de centros de distribución candidatos, un conjunto de nodos de demanda, vehículos y helicópteros con capacidad, de los cuales también se conoce su velocidad, y las rutas entre los nodos junto a su distancia. Se sabe también si las rutas entre los nodos están utilizable o no, velocidad máxima en las rutas entre cada nodo, tiempo límite para la distribución, entre otros. Se quiere determinar que centros de distribución abrir, qué nodos de demanda asignar a cada centro, y las rutas que siguen los vehículos y helicópteros

para entregar la ayuda. De forma de cumplir con los objetivos mencionados anteriormente.
Para resolver el problema se propone un algoritmo heurístico híbrido el cual comprende un algoritmo greedy combinado con un ant colony algorithm de forma jerárquico secuencial. Los procedimientos del algoritmo son, cada uno, incluyendo ciertos pasos que no detallaremos:

1. Algoritmo de inicialización.
2. Asignación de ayuda de forma justa, utilizando un algoritmo greedy.
3. Asignación de nodos de demanda a centros de distribución con un algoritmo greedy.
4. Determinar el método de entrega hacia los nodos de demanda (si por vehículo o helicóptero)
5. Uso de ant colony algorithm para planificar la entrega de ayuda.
6. Regla de terminación del algoritmo.

A.4.8. Ubicación de refugios y evacuación de victimas

En el trabajo de Liang et al. (2019) se resuelve el problema de ubicar refugios y evacuar personas de zonas afectadas hacia esos refugios, determinando las rutas de evacuación, minimizando el tiempo total de evacuación en todos los tramos de carreteras.
Se propone un modelo de programación estocástico (estocástico en la demanda de evacuación) averso al riesgo que incorpora la medida de riesgo CVaR (Conditional Value-at-Risk) para balancear las expectativas relacionadas con el tiempo de evacuación total y el riesgo. Se introducen restricciones de oportunidad para asegurar que los niveles de utilización de los refugios no sean menores que un determinado límite con un cierto grado de confiabilidad. El modelo es de dos etapas, en la primera etapa se hacen las decisiones de ubicación de refugios. Mientras que en la segunda etapa se considera el ruteo de evacuados hacia los refugios. Para manejar la no linealidad de la función objetivo del flujo de tráfico se propone un enfoque SOCP (second-order cone programming).

A.5. Conclusiones

El LRP es analizado con un relevamiento de artículos recientes sobre diferentes implementaciones y métodos de resolución. Se verifica que el estudio del LRP es un tema actual de investigación, ya que han surgido trabajos recientes con distintas aplicaciones y métodos de resolución.
En particular, este documento se enfoca en las aplicaciones del LRP en un contexto de desastre natural. En estos casos vemos que es crucial el tiempo de resolución del problema.
A raíz de los casos de estudio, surge como método de resolución las metaheurísticas. Se presentan varios ejemplos donde las metaheurísticas resuelven diferentes instancias del LRP, lo que permite entender las características y particularidades de resolver este tipo de problemas con estos algoritmos.

Las metaheurísticas tienen como ventaja que son métodos muy generales que incorporan datos del problema en la definición, además existe una gran variedad, por lo que en muchas situaciones se pueden aplicar para la resolución de un problema combinatorio.

Se observa que en los trabajos relevados, en general, es posible agregar complejidad al problema que aporte valor. Por ejemplo, en Veenstra et al. (2018b) se plantea el problema con un único depósito central, pero en un escenario real interesa y tiene sentido tener varios depósitos.
Con los avances en capacidad de procesamiento, es interesante aumentar la complejidad de los problemas para que se asemejen lo más posible a un escenario real.

Referencias del estado del arte

Ahmadi, M., Seifi, A., & Tootooni, B. (2015). A humanitarian logistics model for disaster relief operation considering network failure and standard relief time: A case study on San Francisco district. *Transportation Research Part E: Logistics and Transportation Review*, *75*, 145-163.

Alem, D., Clark, A., & Moreno, A. (2016). Stochastic network models for logistics planning in disaster relief. *European Journal of Operational Research*, *255*, 187-206.

Altay, N., & Green III, W. G. (2006). OR/MS research in disaster operations management. *European journal of operational research*, *175*(1), 475-493.

Awasthi, A., Chauhan, S. S., & Goyal, S. K. (2011). A multi-criteria decision making approach for location planning for urban distribution centers under uncertainty. *Mathematical and Computer Modelling*, *53*(1-2), 98-109.

Bellman, R. (1966). Dynamic programming. *Science*, *153*(3731), 34-37.

Campbell, A., Clarke, L., Kleywegt, A., & Savelsbergh, M. (1998). The inventory routing problem. En *Fleet management and logistics* (pp. 95-113). Springer.

Chi, T.-H., Yang, H., & Hsiao, H.-M. (2011). A new hierarchical facility location model and genetic algorithm for humanitarian relief. *The 5th International Conference on New Trends in Information Science and Service Science*, *2*, 367-374.

Christofides, N., & Eilon, S. (1969). An algorithm for the vehicle-dispatching problem. *Journal of the Operational Research Society*, *20*(3), 309-318.

Christofides, N., Mingozzi, A., & Toth, P. (1981a). Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical programming*, *20*(1), 255-282.

Christofides, N., Mingozzi, A., & Toth, P. (1981b). State-space relaxation procedures for the computation of bounds to routing problems. *Networks*, *11*(2), 145-164.

Dantzig, G., Fulkerson, R., & Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, *2*(4), 393-410.

Drexl, M., & Schneider, M. (2015). A survey of variants and extensions of the location-routing problem. *European Journal of Operational Research*, *241*(2), 283-308.

Farahani, R. Z., & Hekmatfar, M. (2009). *Facility location: concepts, models, algorithms and case studies*. Springer.

Ferrer, J. M., Martín-Campo, F. J., Ortuño, M. T., Pedraza-Martínez, A. J., Tirado, G., & Vitoriano, B. (2018). Multi-criteria optimization for last mile distribution of disaster relief aid: Test cases and applications. *European Journal of Operational Research*, *269*(2), 501-515.

Fisher, M. L., & Jaikumar, R. (1981). A generalized assignment heuristic for vehicle routing. *Networks*, *11*(2), 109-124.

Foster, B. A., & Ryan, D. M. (1976). An integer programming approach to the vehicle scheduling problem. *Journal of the Operational Research Society*, *27*(2), 367-384.

George, H., Jane, B., Damon, P., et al. (2011). Introduction to emergency management.

Gillett, B. E., & Miller, L. R. (1974). A heuristic algorithm for the vehicle-dispatch problem. *Operations research*, *22*(2), 340-349.

Golden, B. L., Raghavan, S., & Wasil, E. A. (2008). *The vehicle routing problem: latest advances and new challenges* (Vol. 43). Springer Science & Business Media.

Gutiérrez, M. Á., de los Cobos, S., & Pérez, B. (1998). Op timización con recocido simulado para el problema de conjunto inde pendiente. *Universidad Autónoma Metropolitana, México, disponible en: http://www. azc. uam. mx/publicaciones/enlinea2/3 2rec. htm,[fecha de consulta: 12 de enero de 2007]*.

Hadjiconstantinou, E., Christofides, N., & Mingozzi, A. (1995). A new exact algorithm for the vehicle routing problem based onq-paths andk-shortest paths relaxations. *Annals of Operations Research*, *61*(1), 21-43.

Hassanzadeh, A., Mohseninezhad, L., Tirdad, A., Dadgostari, F., & Zolfagharinia, H. (2009). Location-routing problem. En *Facility location* (pp. 395-417). Springer.

Holguín-Veras, J., Jaller, M., Van Wassenhove, L. N., Pérez, N., & Wachtendorf, T. (2012). On the unique features of post-disaster humanitarian logistics. *Journal of Operations Management*, *30*(7-8), 494-506.

Jacobsen, S., & Madsen, O. (1978). On the location of transfer points in a two-level newspaper delivery system–a case study. *international symposium on locational decisions*, 24-28.

Jiang, Y., & Yuan, Y. (2019). Emergency logistics in a large-scale disaster context: Achievements and challenges. *International journal of environmental research and public health*, *16*(5), 779.

Krarup, J. (2009). Warehouse location problemWarehouse Location Problem. En C. A. Floudas & P. M. Pardalos (Eds.), *Encyclopedia of Optimization* (pp. 4050-4055). Springer US.

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation science*, *43*(4), 408-416.

Laporte, G., Mercure, H., & Nobert, Y. (1986). An exact algorithm for the asymmetrical capacitated vehicle routing problem. *Networks*, *16*(1), 33-46.

Laporte, G., & Nobert, Y. (1981). An exact algorithm for minimizing routing and operating costs in depot location. *European Journal of Operational Research*, *6*(2), 224-226.

Laporte, G., & Nobert, Y. (1983). A branch and bound algorithm for the capacitated vehicle routing problem. *Operations-Research-Spektrum*, *5*(2), 77-85.

Liang, B., Yang, D., Qin, X., & Tinta, T. (2019). A risk-averse shelter location and evacuation routing assignment problem in an uncertain environment. *International journal of environmental research and public health*, *16*(20), 4007.

Liu, C., Kou, G., Peng, Y., & Alsaadi, F. E. (2019b). Location-routing problem for relief distribution in the early post-earthquake stage from the perspective of fairness. *Sustainability*, *11*(12), 3420.

Luus, R. (2009). Dynamic programming: optimal control applicationsDynamic Programming: Optimal Control Applications. En C. A. Floudas & P. M. Pardalos (Eds.), *Encyclopedia of Optimization* (pp. 862-869). Springer US. https://doi.org/10.1007/978-0-387-74759-0_151

Marinakis, Y. (2009). Location routing problemLocation Routing Problem. En C. A. Floudas & P. M. Pardalos (Eds.), *Encyclopedia of Optimization* (pp. 1919-1925). Springer US.

Mingang, Z., Zengshou, C., & Xiaoyan, W. (2009). Research on location-routing problem of relief system based on emergency logistics. *2009 16th International Conference on Industrial Engineering and Engineering Management*, 228-232.

Moscatelli, S., Tansini, L., & Viera, O. (2009). Disaster management and operation research in Uruguay. *Reportes Técnicos 09-08*.

Nedjati, A., Izbirak, G., & Arkat, J. (2017). Bi-objective covering tour location routing problem with replenishment at intermediate depots: Formulation and meta-heuristics. *Computers & Industrial Engineering*, *110*, 191-206.

Noyan, N., Balcik, B., & Atakan, S. (2016). A stochastic optimization model for designing last mile relief networks. *Transportation Science*, *50*(3), 1092-1113.

Or, I., & Pierskalla, W. P. (1979). A transportation location-allocation model for regional blood banking. *AIIE transactions*, *11*(2), 86-95.

Prodhon, C., & Prins, C. (2014b). A survey of recent research on location-routing problems. *European Journal of Operational Research*, *238*(1), 1-17.

Rath, S., & Gutjahr, W. J. (2014). A math-heuristic for the warehouse location–routing problem in disaster relief. *Computers & Operations Research*, *42*, 25-39. https://doi.org/10.1016/j.cor.2011.07.016

Renaud, J., Boctor, F. F., & Laporte, G. (1996). An improved petal heuristic for the vehicle routeing problem. *Journal of the operational Research Society*, *47*(2), 329-336.

Roh, S. Y., Shin, Y. R., & Seo, Y. J. (2018). The Pre-positioned Warehouse Location Selection for International Humanitarian Relief Logistics. *The Asian Journal of Shipping and Logistics*, *34*(4), 297-307.

Roh, S., Pettit, S., Harris, I., & Beresford, A. (2015). The pre-positioning of warehouses at regional and local levels for a humanitarian relief organisation. *International Journal of Production Economics*, *170*, 616-628.

Ryan, D. M., Hjorring, C., & Glover, F. (1993). Extensions of the petal method for vehicle routeing. *Journal of the Operational Research Society*, *44*(3), 289-296.

Sakiani, R., Seifi, A., & Khorshiddoust, R. R. (2020). Inventory routing and dynamic redistribution of relief goods in post-disaster operations. *Computers & Industrial Engineering*, *140*, 106219.

Toth, P., & Vigo, D. (2002a). Branch-and-bound algorithms for the capacitated VRP. En *The vehicle routing problem* (pp. 29-51). SIAM.

Toth, P., & Vigo, D. (2002b). *The vehicle routing problem*. SIAM.

Tuy, H. (2009). Global optimization in location problemsGlobal Optimization in Location Problems. En C. A. Floudas & P. M. Pardalos (Eds.), *Encyclopedia of Optimization* (pp. 1354-1359). Springer US.

Veenstra, M., Roodbergen, K. J., Coelho, L. C., & Zhu, S. X. (2018b). A simultaneous facility location and vehicle routing problem arising in health care logistics in the Netherlands. *European Journal of Operational Research*, *268*(2), 703-715.

Wang, H., Du, L., & Ma, S. (2014). Multi-objective open location-routing model with split delivery for optimized relief distribution in post-earthquake. *Transportation Research Part E: Logistics and Transportation Review*, *69*, 160-179.

Anexo B

Experimentación

B.1. Tablas segunda etapa

A continuación se muestran las tablas de la segunda etapa de la experimentación separadas por algoritmo. En las tablas siguientes la primera columna indica la instancia. En la segunda columna, se muestra la cantidad de soluciones. Cuanto mayor sea este número, mejor será el algoritmo, ya que encontró un conjunto de soluciones no dominadas más grande. Las siguientes columnas son: hipervolumen, épsilon y r2 normalizados entre 0 y 1 utilizando la normalización mínima-máxima $z_i = (x_i – minimo(x))/(maximo(x) – minimo(x))$, dónde z_i es el nuevo valor normalizado, x_i es el valor no normalizado, $minimo(x)$ es el menor valor del conjunto, $maximo(x)$ es el máximo valor del conjunto. Como se mencionó en la Sección 7.2, a menor estas medidas, mejor solución.

Instance ID	—E—	Hyp	ϵ	R2
37	72	0,031	0,031	0,242
38	84	0,007	0,000	0,043
40	150	0,135	1,000	0,281
41	100	0,000	0,004	0,000
42	106	0,216	0,047	0,152
43	95	1,000	0,096	1,000

Tabla B.1: Resultados comparación variantes MO-VNS, E = conjunto de soluciones, Algoritmo 1

Instance ID	—E—	Hyp	ϵ	R2
37	121	0,000	0,288	0,412
38	133	0,048	0,487	0,394
40	379	0,129	0,074	0,116
41	267	0,023	0,349	0,000
42	124	0,308	0,000	0,085
43	28	1,000	1,000	1,000

Tabla B.2: Resultados comparación variantes MO-VNS, E = conjunto de soluciones, Algoritmo 2

Instance ID	—E—	Hyp	ϵ	R2
37	183	0,000	0,006	0,345
38	200	0,099	0,297	0,300
40	200	0,332	0,379	0,251
41	190	0,091	0,167	0,078
42	191	0,455	0,000	0,000
43	188	1,000	1,000	1,000

Tabla B.3: Resultados comparación variantes MO-VNS, E = conjunto de soluciones, Algoritmo 3

Instance ID	—E—	Hyp	ϵ	R2
37	54	0,026	0,036	0,057
38	81	0,174	0,608	0,577
40	49	1,000	1,000	1,000
41	67	0,000	0,159	0,000
42	60	0,298	0,165	0,073
43	64	0,013	0,000	0,004

Tabla B.4: Resultados comparación variantes MO-VNS, E = conjunto de soluciones, Algoritmo 4

Instance ID	—E—	Hyp	ϵ	R2
37	200	0,001	0,000	0,000
38	200	0,000	0,001	0,001
40	200	0,051	0,526	0,504
41	200	0,252	0,789	0,333
42	200	0,010	0,015	0,001
43	200	1,000	1,000	1,000

Tabla B.5: Resultados comparación variantes MO-VNS, E = conjunto de soluciones, Algoritmo 5

Anexo C

Manual de usuario

A continuación se explican en profundidad las diferentes acciones que permite realizar la aplicación web desarrollada.

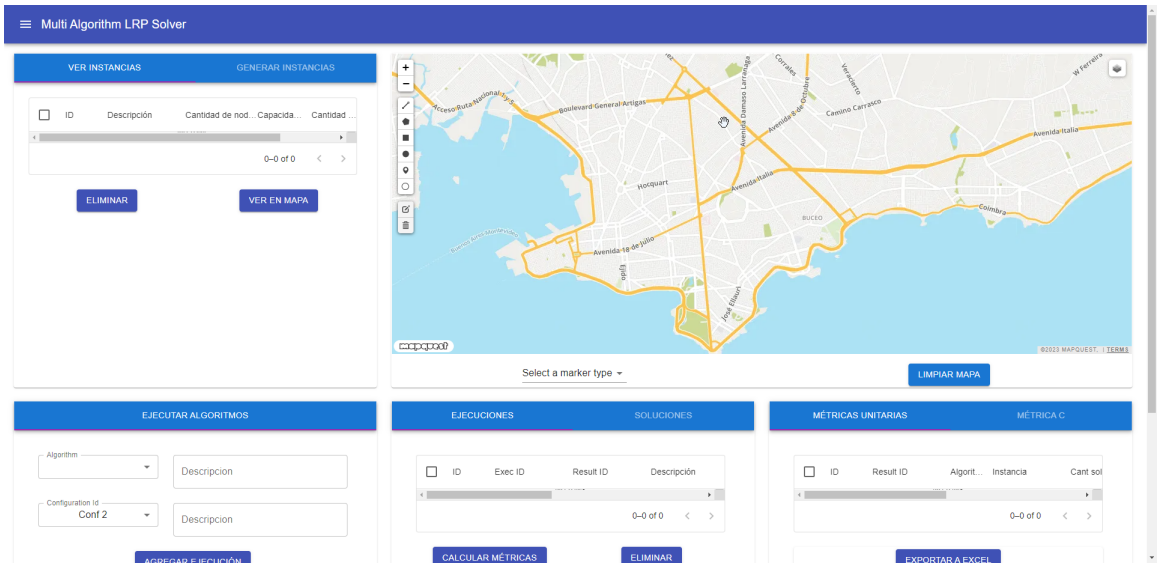

Figura C.1: Pantalla inicial.

Como se muestra en la Figura C.1, siempre en la sección derecha de la pantalla se presenta un mapa interactivo que permite generar y visualizar instancias de solución.

Generar Instancias: El menú de Generar Instancias permite instanciar un problema desde la pantalla. El primer paso es definir en el mapa el depósito, los clientes, las ubicaciones potenciales de los lockers y su radio de cubrimiento.

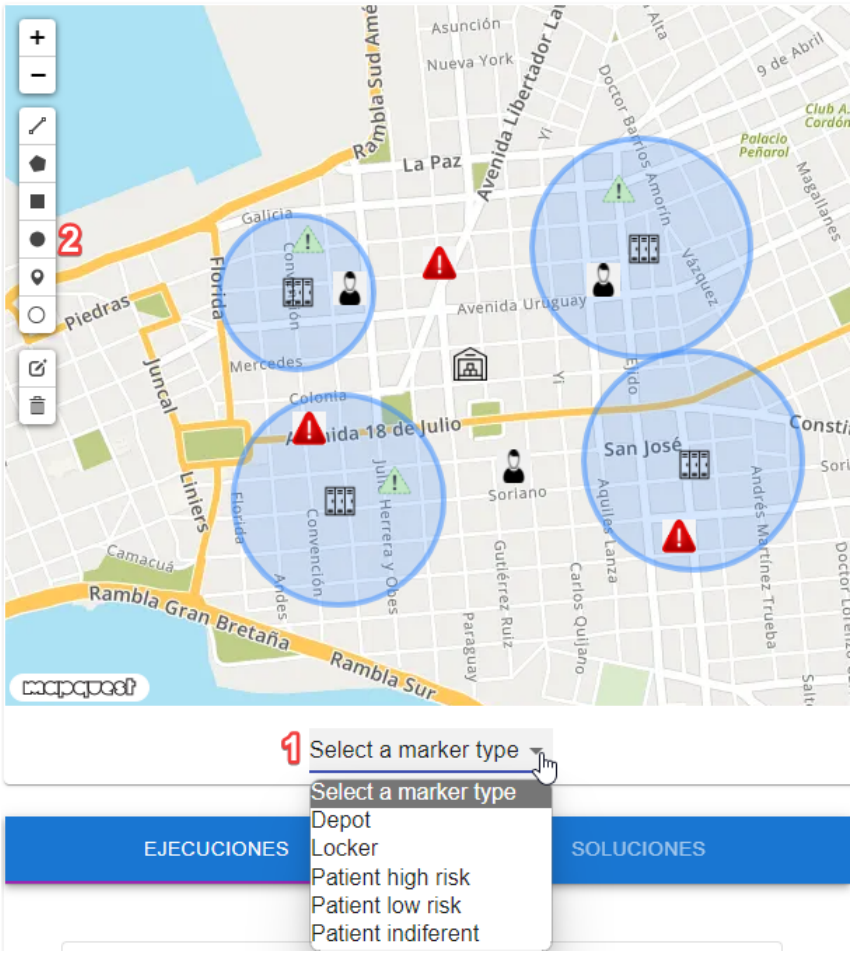

Figura C.2: Detalles de mapa.

Para ubicar los nodos, se debe seleccionar el tipo de nodo (depósito, locker o paciente) desde la lista desplegable como se muestra en el punto 1 de la Figura C.2 y ubicarlo en el mapa. Para ingresar el radio de cubrimiento de un locker, se selecciona la herramienta indicada en el punto 2 de la Figura C.2 y se dibuja el círculo con centro en el locker deseado.

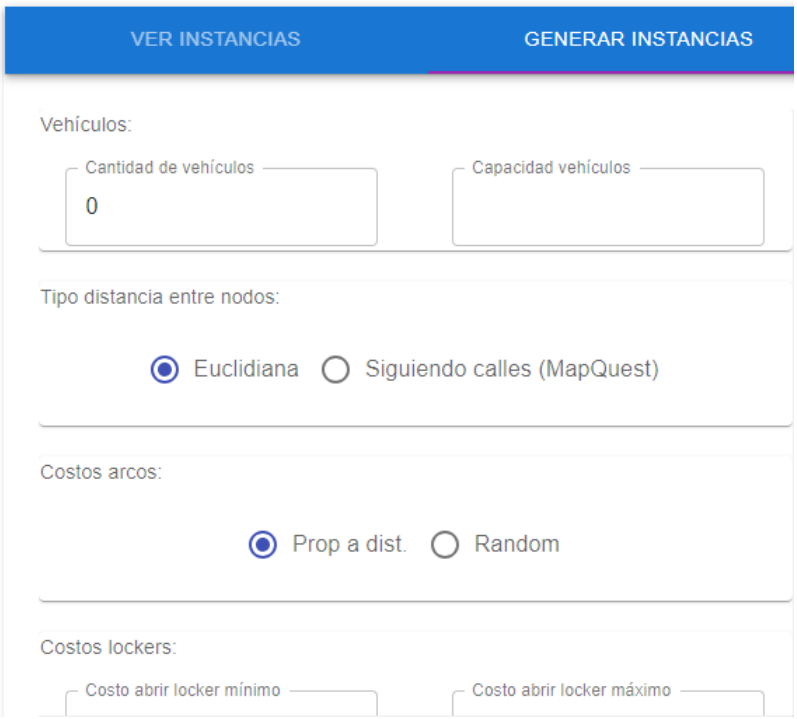

Figura C.3: Datos de instancia.

Luego hay que ingresar desde el menú de Generar Instancia los datos restantes como se indica en la Figura C.3. A continuación los datos a informar:

- Vehículos: se debe indicar la cantidad y la capacidad máxima de los vehículos (recordar que todos tienen la misma capacidad). La capacidad total (cantidad de vehículos por capacidad) debe ser al menos igual a la cantidad de pacientes a atender.

- Tipo de distancia entre nodos: define la forma de calcular la distancia entre nodos. Se puede optar por la distancia Euclidiana o por la brindada por la librería MapQuest. Tener en cuenta que MapQuest indica la distancia a través de rutas por las calles y teniendo en cuenta el sentido de las mismas.

- Costos de arcos: define la matriz de costos entre los nodos. Los mismos pueden definirse como proporcionales a la distancia (mayor distancia implica mayor costo) o aleatorios dentro un rango definido.

- Costo de lockers: define el costo que implica abrir un locker. Se define aleatorio dentro de un rango definido.

- Tiempo: define la matriz de tiempo entre 3 opciones: Proporcional a una velocidad indicada, aleatorio dentro de un rango dado, o inversamente proporcional a la matriz de costo (útil para tener objetivos no convergentes).
- Servicio: define el costo de los servicios que puede indicarse por tipo (para pacientes y lockers) o aleatorio dentro de un rango dado.
- Penalización: define la penalización por atender a los pacientes en su domicilio y puede fijarse un valor o indicar aleatorio dentro de un rango dado.
- Descripción: define un texto para identificar a la instancia.

Con todos los datos anteriores definidos, el botón de Generar Instancia crea el modelo de la instancia y lo persiste en base de datos.

Ver Instancias:

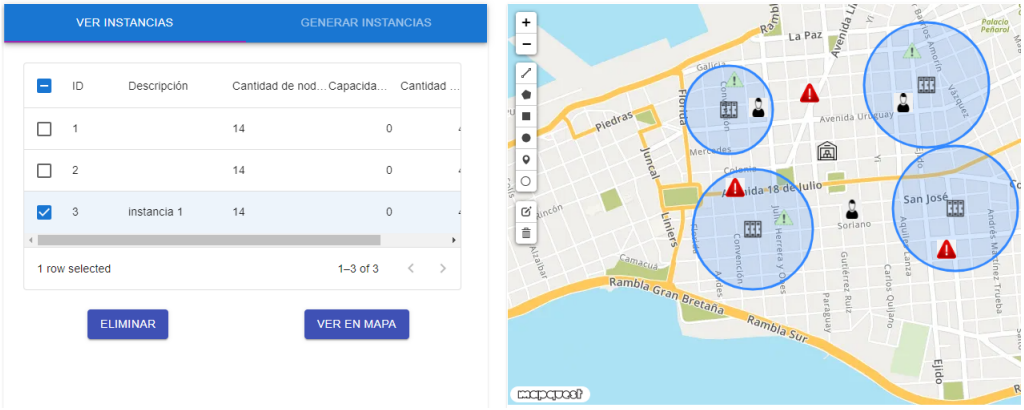

Figura C.4: Ver instancias.

El menú de Ver instancias (Figura C.4) permite seleccionar de entre las instancias generadas y mostrarlas en el mapa o eliminarlas.

Ejecutar Algoritmos:

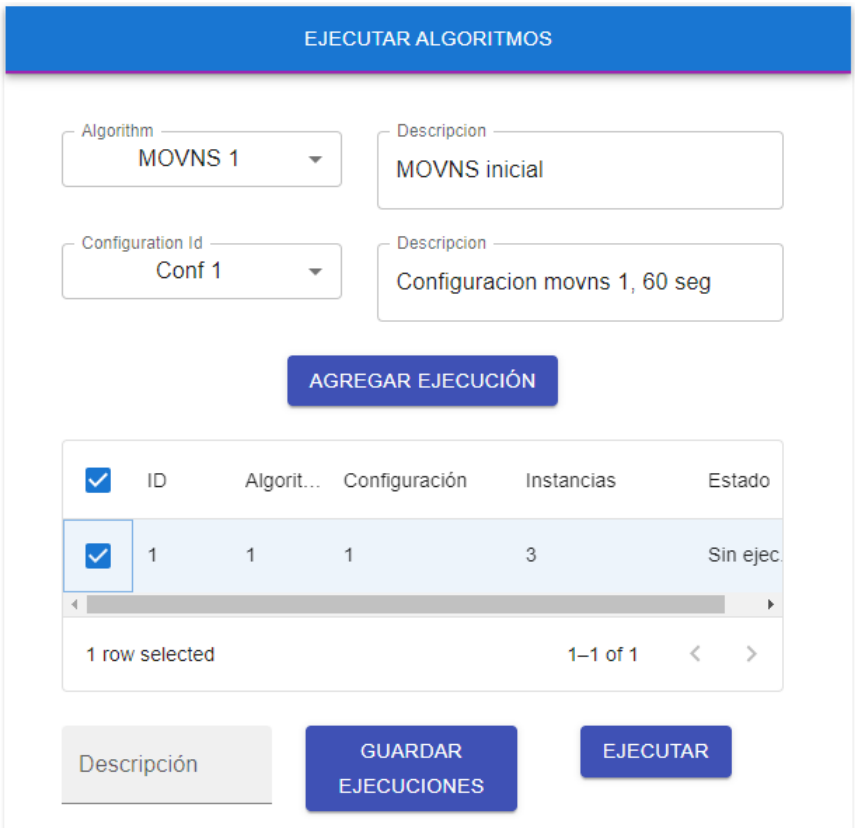

Figura C.5: Ejecutar Algoritmos.

La sección de Ejecutar Algoritmos (Figura C.5) permite seleccionar un algoritmo con una configuración dada (previamente guardada en base de datos) para que resuelva las instancias marcadas en el menú de Ver Instancias.
El botón de Ejecutar resuelve la instancia con el Algoritmo seleccionado, y el Guardar Ejecuciones persiste los resultados en base de datos.

Ejecuciones:

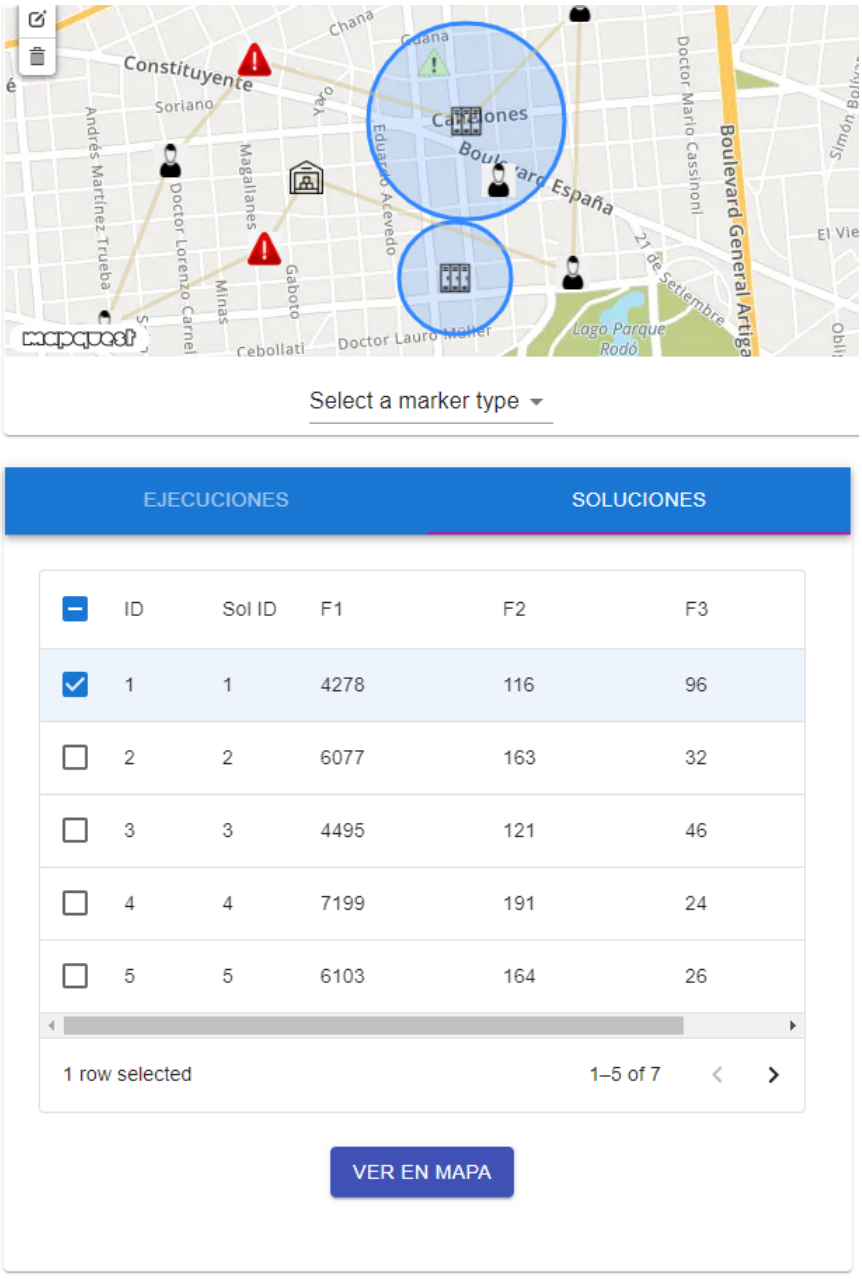

Figura C.6: Resultados en mapa.

La sección de Ejecuciones, dentro del tab de Soluciones, lista las soluciones

generadas y permite seleccionarlas para visualizarlas en el mapa, como se muestra en la Figura C.6.
El tab de Ejecuciones, permite seleccionar las ejecuciones para luego calcular las métricas como se muestra en la Figura C.7.

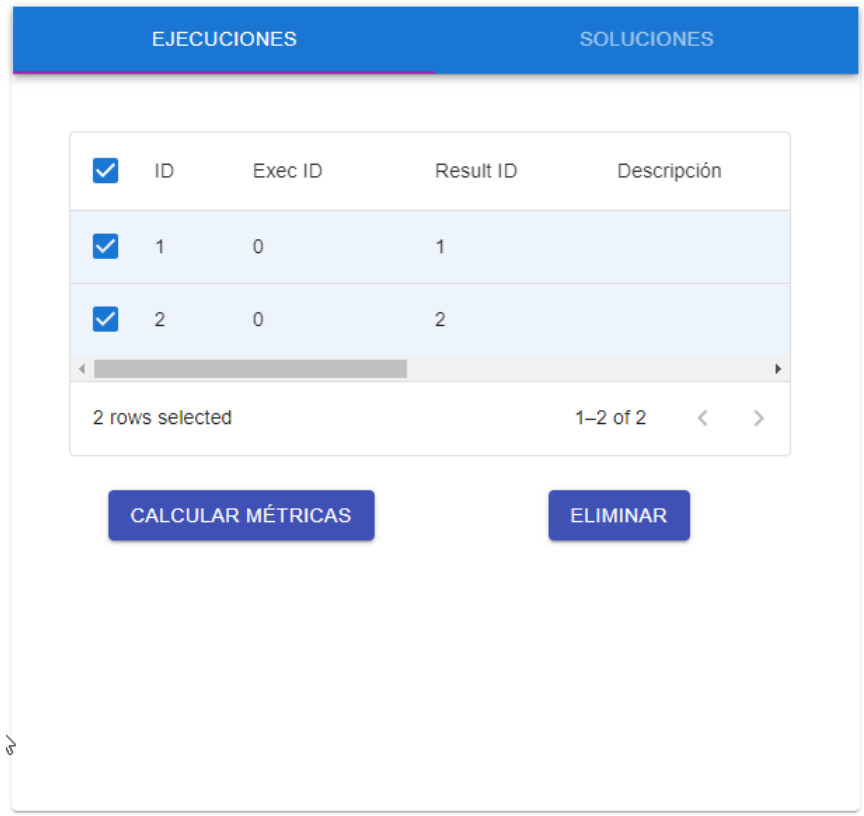

Figura C.7: Calcular métricas.

Métricas: Finalmente, se pueden visualizar los resultados de las métricas para las ejecuciones seleccionadas en el paso anterior en la sección de Métricas, como se muestra en la Figura C.8. En la tabla se muestra los valores calculados para las diferentes métricas, así como información de la ejecución.

Figura C.8: Métricas unitarias.

El botón de Exportar a Excel genera el archivo solicitado con la información de la tabla.