Universidad Nacional de Rosario

Facultad de Ciencias Exactas, Ingeniería y Agrimensura

Desarrollo de técnicas de optimización via simulación aplicadas a sistemas de tráfico

Tesis presentada en la Facultad de Ciencias Exactas, Ingeniería y Agrimensura, en cumplimiento parcial de los requisitos para optar al título de

Doctor en Ingeniería

Autor:

Enrique Gabriel Baquela

Director: Dra. Ana Carolina Olivera
Co-Director: Mg. Marta Liliana Cerrano

Diciembre 2019

Certifico que el trabajo incluído en esta tesis es el resultado de tareas de investigación originales y que no ha sido presentado para optar a un título de postgrado en ninguna otra Universidad o Institución.

Enrique Gabriel Baquela

Resumen

El objetivo general de esta tesis doctoral fue desarrollar algoritmos basados en el concepto de Optimización vía Simulación (OvS) para el tratamiento de problemas de optimización asociados a los sistemas de tráfico urbano. OvS es una metodología general de optimización en la cual las funciones objetivo y/o las restricciones son evaluadas mediante el uso de un simulador.

Varios problemas han sido abordados en el marco de esta tesis doctoral. El primero fue el problema de optimización del rediseño de la red de tráfico ante incremento de la población. Se trata de un problema novedoso al cual denominamos *Problema de Asignación de Orígenes y Destinos de Tráfico* ("Origin-Destination Traffic Assignment Problem", ODTAP). El segundo problema fue la versión multiobjetivo del *Problema de inversión de carriles* ("Lane Reversal", LR) de la red de tráfico. El tercer problema tratado fue una variante del *Problema de ruteo de autobuses escolares* ("School Bus Routing Problem", SBRP), en el cual se permite que los pasajeros estén distribuidos aleatoriamente en el espacio.

Utilizar Optimización vía Simulación presenta un alto costo computacional, por ello se desarrolló un esquema general de optimización basado en metamodelos ("Kriging" en este caso), que permite reducir la cantidad de simulaciones a realizar. Asimismo, se desarrolló un segundo esquema donde se define en forma dinámica la longitud de cada simulación, a fin de reducir los tiempos de cálculo de cada una de ellas.

Por último, dada la naturaleza multiobjetivo y estocástica de algunos problemas tratados, se plantea una metodología de optimización que permite obtener Fronteras de Pareto que incrementa la probabilidad de que una solución a implementar sea óptima (o cercana al óptimo) en el mayor número de escenarios posibles.

Abstract

The main objective of this doctoral thesis was developing algorithms based on the concept of Optimization via Simulation (OvS) for the treatment of problems associated with traffic urban systems. OvS is a general methodology of optimization where the objective functions and/or the restrictions are evaluated through the use of a simulator.

Several problems have been addressed in the framework of this doctoral thesis. The first problem addressed was the problem of optimizing the redesign of the traffic network in the face of an increase in the population. This is a novel problem which we call *Problem of Assignment of Origins and Destinations of Traffic* (ODTAP). The second problem was the multiobjective version of the *Lane Reversal Problem* (LR) of the traffic network. The third problem addressed was the *School Bus Routing Problem* (SBRP). In the case studied, passengers are allowed to be randomly distributed in the space.

Utilize Optimization via Simulation presents a high computational cost, therefore a general optimization scheme based on metamodels was developed (Kriging in this case), which allows reducing the number of simulations to be performed. Likewise, a second optimization algorithm was developed based on dynamically defining the length of each simulation, in order to reduce the calculation times of each of them.

Finally, given the multiobjective and stochastic nature of some of the problems dealt with, an optimization methodology was developed to obtain Pareto fronts that increases the probability that a solution to be implemented will be optimal (or close to optimal) in as many different scenarios as possible.

Agradecimientos

Si bien se presenta como una obra individual, la elaboración de una tesis de doctorado es imposible sin una red de personas que funjan de guía y/o apoyo. Son muchas las personas que, directa o indirectamente, colaboraron para poder alcanzar el objetivo plasmado en estas páginas, pero no quiero dejar de nombrar a las siguientes:

- A mi directora de tesis, la Doctora Ana Carolina Olivera, que junto a su (bastante severa) guía y a su (aún mas grande) paciencia[1], me orientó y me dio aliento para transitar este sendero, a priori desconocido, y que tantas idas y vueltas tiene.
- A mi co-directora, la Magister Marta Liliana Cerrano, por el apoyo brindado cada vez que lo necesité.
- A mi esposa, Flavia, por el tiempo cedido voluntariamente para poder dedicarme a todas las tareas necesarias para la realización de la tesis, sin importar si fueran mañanas o noches, fines de semana o vacaciones. Y sobre todo por el apoyo psicológico, cada vez que algo salía mal[2].
- A mis hijos Clementina y Constantino, por el tiempo que les pedí prestado y que espero poder devolvérselos.

Enrique Gabriel Baquela

San Nicolás de los Arroyos, Buenos Aires, Argentina

Diciembre de 2019

[1] Severidad y paciencia totalmente necesarias debido a las múltiples falencias del autor de esta tesis.

[2] Algo bastante frecuente, vale decir.

'Ad Augusta Per Angusta.'

— Refrán romano

'Ph´nglui mglw´nafh Cthulhu R´lyeh wgah´nagl fhtagn.'

— Al Azif, Abdul Alhazred

'Aprender es la única cosa de la cual la mente nunca se cansa, nunca tiene miedo y nunca se arrepiente.'

— Leonardo Da Vinci

Abreviaturas y Símbolos

AEI Indicador Épsilon Aditivo (Additive Epsilon Indicator). 28, 107, 111, 118

CCO Operador de Comparación de Agrupamiento (Crowded-Comparison Operator). 24, 102

CD Distancia de agrupamiento (Crowding Distance). 23, 25, 141

CMSA Construir, Combinar, Resolver y Adaptar (Construct, Merge, Solve and Adapt). 78

COvS Problemas de OvS Continua (Continue OvS). 37, 39

DH Hipervolumen Dominado (Dominated Hypervolume). 28, 107, 108, 118, 124, 125, 132

DOvS Problemas de OvS Discreta (Discrete Ovs). 37, 39

GA Algoritmo Genético (Genetic Algorithm). 14, 17–19, 23, 24, 90

GD Distancia Generacional (Generational Distance). 27, 104, 107, 111, 118, 136–138, 141, 142, 145, 146, 149, 150

GS Uniformidad (Generalized Spread, Δ). 27, 107, 111, 118, 136, 137, 140, 142, 145, 146, 149–151

K-MOGA Algoritmo genetico multiobjetivo basado en metamodelos del tipo Kriging (Kriging based MultiObjective Genetic Algorithm)). 41, 103, 107, 108, 111, 117, 118

K-NSGA-II Algoritmo NSGA-II basado en metamodelos de Kriging (Kriging-based NSGA-II). En el Capítulo 7, se utiliza esta sigla para designar al algoritmo solo con sus Etapas I y II.. 91–93, 95, 103, 104, 107, 108, 111, 123, 129, 141, 142, 157

K-NSGA-II-S3 Algoritmo NSGA-II basado en metamodelos de Kriging (Kriging-based NSGA-II). En el Capítulo 7, se utiliza esta sigla para designar al algoritmo completo, con sus Etapas I, II y III.. 104, 107, 108, 110, 111, 117, 118

NSGA Algoritmo genético multiobjetivo de ordenamiento por no-dominancia (Non-Dominated Sorted Genetic Algorithm). 41, 103, 108

NSGA-II Algoritmo genético multiobjetivo elitista de ordenamiento por no-dominancia (elitist Non-Dominated Sorted Genetic Algorithm). 4, 5, 23–25, 65, 81, 82, 85, 86, 88, 89, 91, 92, 94, 95, 97–99, 101, 103, 104, 107, 108, 110, 111, 118, 125, 126, 130–133, 155

NTDP Problema de diseño de red de tráfico (Network Traffic Design Problem). 55, 56

ODTAP Problema de asignación de orígenes y destinos de tráfico (Origin-Destination Traffic Assignment Problem). 3, 4, 55, 56, 66, 69–71, 74, 133, 156–159

OvS Optimización via Simulación (Optimization via Simulation). 2–6, 13, 32–39, 54, 56, 60, 65, 66, 69, 74, 89, 91, 92, 97, 98, 123, 126, 142, 144, 145, 156–158

PL Programación Lineal (Linear Programming). 13

PLE Programación Lineal Entera-Mixta (Mixed-Integer Linear Programming). 13, 14, 60, 85, 86, 149

PSO Optimización por Enjambre de Partículas (Particle Swamp Optimization). 14, 19, 24, 25

RandB Problemas de rankeo y selección (Ranking and Selection). 37

RSUMO Paquete para el lenguaje de programación R que permite interactuar con SUMO. 5, 54, 135, 142

S-NSGA-II Algoritmo NSGA-II estocástico (Stochastic NSGA-II). 99, 128, 130–133, 155, 158, 159

SA Recocido Simulado (Simulated Annealing). 14–17, 78, 86, 88, 90

SBRP Problema de ruteo de autobuses escolares (School Bus Routing Problem). 3, 61, 157

SD Dinámica de Sistemas (System Dynamics). 128

SMPSO Optimización por Enjambre de Partículas Multiobjetivo con Velocidad Acotada (Speed-Constrained Multiobjective Particle Swamp Optimization). 4, 25, 70, 74, 77, 134, 136, 141, 142, 156, 157

SSBRP Problema de ruteo de autobuses escolares a una única escuela (Single-School Bus Routing Problem). 3, 5, 53, 61, 62, 64, 81, 86, 88, 147, 158

SUMO Simulador de movilidad urbana (Simulator of Urban MObility). 44, 53, 54, 65–70, 74, 135

TSP Problema del viajante de comercio (Travel Salesman Problem). 61, 82, 83, 86, 88

VRP Problema del de ruteo de vehículos (Vehicle Routing Problem). 61

VSL-NSGA-II Algoritmo NSGA-II con longitudes de simulaciones variables (Variable Simulation Length NSGA-II). 97, 98, 122, 124–126, 141, 142, 157, 159

Índice general

Índice de tablas

Índice de figuras

Índice de algorítmos

Índice de códigos

Capítulo 1

Introducción

En este capítulo se realiza una introducción a esta tesis doctoral con especial énfasis en las motivaciones, objetivos y contribuciones de la misma.

1.1. Motivaciones y Objetivos

La gestión de los sistemas de tráfico es una actividad compleja que, debido al dinamismo intrínseco de los sistemas con los que trata y la urgencia de los problemas a resolver, suele estar sujeta a un rendimiento deficiente. Se presentan dificultades tanto en el plano estratégico (por ejemplo, el rediseño de la red de tráfico) como en el operativo (por ejemplo, el corte temporal de arterias y desvío del tráfico) debido a que los problemas a resolver suelen tener muchas variables de decisión, múltiples objetivos y estar sujetos a incertidumbre.

Los problemas de optimización de sistemas de tráfico son un tipo de problema importante dentro de campo de la Investigación Operativa[1], pero que debido a la potencia de cómputo necesaria para resolverlos, hasta hace no muy poco tiempo su tratamiento era determinístico, enfocado a un solo objetivo y con un grado importante de sobresimplificación del sistema real. Debido a esto último, muchas soluciones, al momento de implementarlas en la práctica, no solo

[1]La Investigación Operativa es una rama de la matemática que trata sobre la toma de decisiones respecto del funcionamiento de sistemas complejos. No abarca solamente problemas de optimización y simulación, sino que se compone de un número muy amplio de herramientas, utilizables en un conjunto de dominios de aplicación también muy vasto. Se puede consultar mas información en `https://www.informs.org/Impact`

son subóptimas, sino que a veces ni siquiera son factibles [Gazis, 2002, Elefteriadou, 2014]. En la práctica diaria, el diseño y operación de sistemas de tráfico se realizan en base a criterios como los desarrollados en el Highway Capacity Manual [TRB, 2016]. Este manual, desarrollado por el *Transportation Research Board* de la *National Academies of Science* en EE.UU., contiene una serie de modelos analíticos, basados en relevamientos estadísticos previos en sistemas de tráficos reales, para definir la capacidad (el número de vehículos máximo que pueden circular sin embotellamiento) de redes de tráfico tanto urbanas como no-urbanas. Si bien extremadamente completo, cubriendo cada situación y configuración potencial, y siendo actualizado frecuentemente (la última versión es de 2016), no deja de ser un modelo estático extrapolado desde un ambiente particular, sin garantías de generalidad.

En los últimos años, los analistas comenzaron a tratar estos problemas utilizando simulaciones. Simular un sistema de tráfico permite escapar de las dificultades que presenta el modelado analítico, pudiendo representar en forma muy certera el desempeño del sistema. Sin embargo, este procedimiento está limitado a probar algunas pocas configuraciones y seleccionar de ellas la mejor. La solución sigue estando potencialmente muy lejos del óptimo, pero al menos es factible [Barceló, 2010].

Los avances de la última década en hardware hacen posible tratar estos problemas usando metodologías mas complejas. Por ejemplo, mediante *Optimización vía Simulación* (OvS). Esta es una metodología de optimización que consiste en la combinación de un algoritmo de optimización, por ejemplo una metaheurística, con un algoritmo de simulación. El primero genera y selecciona soluciones, mientras que el segundo evalúa el desempeño de las mismas. Se reemplaza entonces el uso de una función objetivo analítica por una simulación del sistema bajo estudio, la cual permite obtener estimaciones de las métricas utilizadas para evaluar la performance del sistema en forma realista. Mediante simulaciones se pueden modelar funciones objetivo estocásticas y/o multiobjetivo, con dinámicas no lineales y que cubran una porción del sistema estudiado mayor que una formulación analítica [Fu, 2015].

El enfoque de OvS permite obtener muy buenas soluciones para los problemas de decisión en sistemas de tráfico, pero tiene el inconveniente de requerir un elevado tiempo de cómputo y/o una gran cantidad de recursos de hardware. Simular un sistema de tráfico es computacionalmen-

te costoso y esta metodología necesita realizar muchas simulaciones para encontrar soluciones óptimas. Esta dificultad no es un problema de los sistemas de tráfico solamente, sino de la metodología de OvS en sí misma. Existe el mismo inconveniente en problemas de planificación de la producción, en el de diseño de redes eléctricas, en el diseño de aeronaves, etc. Debido a la necesidad del uso de simulaciones en problemas relativos a diseño de estructuras, componentes y sistemas complejos, es en estos donde se han desarrollado técnicas para reducir el costo computacional sin ocasionar una caída vertiginosa en la calidad de las soluciones encontradas.

De las muchas aplicaciones de OvS sobre problemas de sistemas de tráfico, en esta tesis se tratan tres:

- Rediseño de la red de tráfico para absorber el incremento poblacional proyectado.
- Inversión de carriles de tráfico como opción para el tratamiento de las olas de tráfico.
- Resolución del problema de transporte de personas en la red de tráfico.

La primera aplicación consiste en la optimización del rediseño urbano en una situación de incremento poblacional para que se minimize el efecto en el sistema de tráfico. Se debe definir donde agregar nuevos nodos generadores de demanda de tráfico, y realizar las modificaciones apropiadas a la red vial, teniendo en cuenta que dicha red es mayormente fija y resulta costoso modificarla. Se trata el problema en una forma original, formalizándolo como un problema de asignación. A este problema lo llamamos con el nombre de *Origin-Destination Traffic Assigment Problem* (ODTAP).

En el segundo problema, se trata la versión multiobjetivo del problema de inversión de carriles. Dicho problema consiste en seleccionar que carriles de una vía de tráfico invertir, en una determinada franja horaria, para disminuir el tiempo medio de circulación de los vehículos. El problema es extensamente tratado en la literatura, pero en general en forma monoobjetivo y/o en forma analítica, tomando en cuenta solamente el impacto local del problema. En esta tesis tratamos el problema en su faceta global y multiobjetivo, con y sin el efecto de los semáforos tomado en cuenta.

El tercer problema que se trata es el *Single-School Bus Routing Problem* (SSBRP), una variación del problema de optimización clásico denominado *School Bus Routing Problem* (SBRP). En el

mismo se trata de asignar pasajeros a autobuses, definir las paradas para los autobuses y las rutas de estos últimos, con el fin que todos los pasajeros arriben a un mismo destino. En la versión analizada, la ubicación inicial de los pasajeros se permite que sea incierta. Se tratan la formulación multiobjetivo de este problema, usando un esquema simple de OvS y se desarrolla un algoritmo para el caso general. Éste algoritmo se implementa mediante la metaheurística NSGA-II, y los dos anteriores mediante SMPSO, pero sus estructuras generales no presentan mayores inconvenientes para ser adaptadas al uso de otra metaheurística.

Además del tratamiento de estos tres problemas, se analizan en esta tesis formas de reducir el costo computacional del enfoque de Optimización vía Simulación. Por un lado, se diseña y evalúa una metodología para reducir la longitud del tiempo de simulación, apelando a técnicas de muestreo para definir cuan larga debe ser cada simulación. Adicionalmente, se desarrolla otra metodología basada en metamodelado para reducir la cantidad de simulaciones necesarias. Ambas son aplicables a cualquier tipo de problema que se resuelva con OvS, no solamente problemas de tráfico. Por último, se diseñó una tercer metodología tendiente a mejorar la calidad de los resultados en problemas de optimización con múltiples objetivos en los que además exista aleatoriedad. Esta última resulta aplicable a cualquier problema estocástico multiobjetivo, no solamente aquellos tratados con OvS. Los tres algoritmos están implementados sobre la base de la metaheurística multiobjetivo NSGA-II, pero las metodologías generales se pueden trasladar sin mayor dificultad a cualquier otro algoritmo basado en OvS (las dos primeras) o a cualquier algoritmo basado en estimación de Fronteras de Pareto en problemas estocásticos mediante criterios de no-dominancia.

1.2. Contribuciones

Las principales contribuciones realizadas durante la elaboración de esta tesis son:

- Definición, formalización y resolución mediante OvS del ODTAP, un novedoso problema en el ámbito del planeamiento del tráfico urbano.
- Diseño de un algoritmo basado en OvS para seleccionar los carriles a invertir de

manera tal de minimizar el tiempo medio de viaje y la cantidad de carriles invertidos.

- Ampliación del algoritmo anterior para resincronizar semáforos a fin de mejorar los resultados sobre los dos objetivos.
- Construcción de un novedoso algoritmo basado en OvS para la resolución del SSBRP en el caso de que la demanda de servicio esté distribuida aleatoriamente en el espacio.
- Construcción de una innovadora metaheurística de OvS multiobjetivo derivada del algoritmo NSGA-II que hace uso de metamodelos del tipo Kriging para reducir la cantidad de simulaciones necesarias para aproximar la Frontera de Pareto del problema y técnicas de muestreo para definir la validez del metamodelo.
- Diseño de una metaheurística basada en NSGA-II para estimar el tiempo mínimo de simulación necesario para poder evaluar comparativamente múltiples soluciones en un esquema de OvS.
- Elaboración de un criterio de evaluación novedoso e implementación mediante una metaheurística que adapta el NSGA-II para tratar el caso en que al menos un objetivo sea estocástico, sin necesidad de convertir el problema a uno determinístico mediante el uso de estadísticos como la esperanza.

Para evaluar los algoritmos creados se utilizaron conjuntos de problemas populares en la literatura (cuando se encontraban disponibles), se adaptaron problemas similares a las características de los tratados en esta tesis, o bien se desarrollaron procedimientos experimentales para la creación de datos de testeo.

Como producto adicional de esta tesis, se desarrolló el paquete de software RSUMO, el cual fue utilizado para gestionar y recolectar resultados de muchas de las simulaciones ejecutadas.

1.3. Organización de la tesis

Esta tesis se encuentra estructurada de la siguiente manera:

- En el presente capítulo se presenta una introducción general al tema de la tesis y se expone la motivación para su realización.
- En el Capítulo 2 se introducen los conceptos de optimización y optimización multiobjetivo, se explican los algoritmos de resolución y se presentan las métricas de análisis.
- En el Capítulo 3 se presenta la metodología de OvS. Se describe el enfoque general, se realiza una síntesis de la teoría de OvS y se trata el uso de metamodelos.
- La introducción de los sistemas de tráfico se realiza en el Capítulo 4. Se muestran los principales problemas de optimización relativos a este tipo de problemas, y se describe el funcionamiento de los simuladores de tráfico.
- Los métodos de resolución desarrollados para resolver los problemas asociados a sistemas de tráfico presentados en el Capítulo 4 se describen en el Capítulo 5.
- En el Capítulo 6 se describen los métodos desarrollados para la resolución de problemas de optimización mediante OvS y problemas estocásticos multiobjetivo.
- Los experimentos realizados, sus resultados y el análisis de los mismos son presentados en el Capítulo 7.
- Finalmente, el Capítulo 8 presenta las conclusiones de esta tesis.

Capítulo 2

Optimización

En este capítulo se realiza una introducción a los conceptos de optimización monoobjetivo y multiobjetivo, determinística y estocástica, y las técnicas utilizadas para su resolución. Dada la temática de esta tesis, el foco respecto a métodos de resolución estará puesto en técnicas metaheurísticas en vez de metodologías analíticas.

2.1. Introducción

Desde un punto de vista práctico, un problema de optimización consiste en la asignación de recursos escasos de manera tal de lograr el mejor efecto posible tras el uso de los mismos [Chinneck, 2015]. Pese a lo general de esta definición, encuentra aplicaciones en un gran conjunto de problemas, por ejemplo:

- ¿Cómo asignar prácticos de puerto a buques de manera tal que el tiempo de espera de todos los buques sea el menor posible?
- ¿Cómo asignar y secuenciar trabajos en equipos de mecanizados de manera tal que la producción mensual sea la mayor posible?
- ¿Cómo sincronizar un conjunto de semáforos para reducir los tiempos de viaje de los vehículos que circulan por esas arterias?

Un problema de optimización se puede definir formalmente mediante el Sistema de Ecuaciones 2.1, 2.2 y 2.3 [Luenberger and Ye, 2008, Fu, 2015] [1]:

$$Min\ \vec{f}(\vec{x}) \tag{2.1}$$

Sujeto a:

$$\vec{g}(\vec{x}) \leq \vec{r} \tag{2.2}$$

$$\vec{x} \in \Theta \tag{2.3}$$

En el cual $\vec{x}$ es una solución potencial, $\vec{f}$ es el vector de funciones a optimizar, $\vec{g}$ y $\vec{r}$ son los vectores de relaciones funcionales y cotas que restringen la viabilidad de las soluciones, y Θ es el dominio de $\vec{x}$ (definido usualmente, pero no necesariamente, como un conjunto de cotas inferiores y superiores para cada componente de $\vec{x}$). $\vec{x}$ es un vector de una o más componentes. Cada una de ellas se denomina *variable de decisión* y se suelen mostrar y agrupar en forma explicita, evitando el formalismo vectorial. Por convención, los problemas de optimización se plantean como problemas de minimización. Si queremos maximizar un objetivo o plantear relaciones de restricción del tipo "$\geq$" o ">" simplemente recurrimos a un cambio de signo, y si queremos plantear restricciones de igualdad, transformamos una restricción del tipo "=" en dos restricciones, una del tipo "$\leq$" y otra del tipo "$\geq$". Es importante aclarar que los operadores de comparación en las restricciones están vectorizados en esta definición, es decir, comparan las componentes homólogas de los dos vectores intervinientes.

Si una solución potencial $\vec{x}$ cumple con las Ecuaciones 2.2 y 2.3, la solución se denomina *factible*. Si, además, su imagen representa el mínimo valor posible de $\vec{f}(\vec{x})$ dentro de los valores que las ecuaciones anteriores permiten, la solución se denomina *óptima*. Una solución que no cumpla con las Ecuaciones 2.2 y 2.3 se denomina *no factible* o *infactible*. En la práctica, una solución

[1] La definición formal y la posterior taxonomía presentada en este capítulo es una entre varias posibles. Por otra taxonomía diferente, orientada al tipo de software de optimización utilizado, se puede consultar, por ejemplo, la siguiente url: `http://plato.asu.edu/sub/pns.html`

no factible es una solución que no se puede implementar.

La Ecuación 2.3 define el *espacio de decisión* o *de soluciones* (es decir, el conjunto de todas las soluciones potenciales al problema), mientras que la Ecuación 2.2 acota al espacio de soluciones, definiendo el subespacio factible $\vec{X} \subseteq \Theta$ (es decir, el conjunto de todos los valores que $\vec{x}$ puede efectivamente tomar). Al espacio formado por las imágenes de cada elemento del espacio de soluciones se lo conoce como *espacio de objetivos*. Si los posibles valores de cada componente de $\vec{x}$ pueden ser reales, estamos ante un problema de optimización continua. Si todas las componentes tienen que ser enteras, el problema es de optimización entera. Si algunas componentes tienen que ser enteras y otras pueden ser continuas, el problema es de optimización entera-mixta. El caso puntual de problemas de optimización entera con un espacio de soluciones finito se denomina optimización combinatoria [Cao and He, 2019].

De acuerdo al tipo de problema, la Ecuación 2.1 o la Ecuación 2.2 podrían no estar presentes. Si la Ecuación 2.1 no esta presente, estamos ante un problema de satisfacción de restricciones (*Constraints Programming*), en el cual no nos interesa optimizar un objetivo sino hallar soluciones factibles. Si la Ecuación 2.2 es la faltante, estamos ante un problema de optimización no restringida, en el cual todo el dominio de $\vec{f}$ es factible y el problema de optimización se reduce a buscar el mínimo global de $\vec{f}$ [Nocedal and Wright, 2006].

Si $\vec{f}$ tiene una sola componente, el problema es de optimización monoobjetivo, mientras que si tiene dos o más componentes el problema es multiobjetivo. Cada componente $i \in [1..n]$ representa un objetivo a minimizar mediante la función escalar f_i. Muchos problemas de interés práctico (dentro y fuera del ámbito de los sistemas de tráfico) tienen más de un objetivo.

Por último, $\vec{f}$ y $\vec{g}$ pueden ser determinísticas o estocásticas respecto de $\vec{x}$. En el primer caso, estamos ante un problema de optimización determinístico y, en el segundo, ante un problema de optimización estocástico [Nocedal and Wright, 2006, Bonnans, 2019]. En un problema estocástico, el objetivo a minimizar y las restricciones a satisfacer no hacen referencia a $\vec{f}$, $\vec{g}$ y $\vec{r}$ en forma directa, sino mediante la aplicación de estadísticos [Ruszczynski and Shapiro, 2009]. Por ejemplo, la Ecuación 2.1 podría verse transformada en la Ecuación 2.4, donde la componente

i del vector $\vec{E}$ es la esperanza matemática de $f_i(\vec{x})$.

$$Min\ \vec{E}(\vec{f}(\vec{x})) \tag{2.4}$$

2.2. Optimización monoobjetivo determinística

En un problema de optimización monoobjetivo, la función $\vec{f}$ tiene una sola componente, por lo cual se convierte en una función escalar f [Dennis and Schnabel, 1996, Vanderbei, 2008]. El objetivo a optimizar es aquí único, pudiendo compararse múltiples soluciones mediante un único valor. Una solución $\vec{x}$ es un óptimo [2] global si el valor asociado de f para todas las demás soluciones factibles es mayor o igual al de la solución en cuestión. Formalmente:

- Una solución $\vec{x}^*$ es un óptimo global débil para el problema monoobjetivo si y solo si no existe ningún $\vec{x} \in \vec{X} - \{\vec{x}^*\}$ tal que $f(\vec{x}) < f(\vec{x}^*)$.
- Una solución $\vec{x}^*$ es un óptimo global fuerte para el problema monoobjetivo si y solo si no existe ningún $\vec{x} \in \vec{X} - \{\vec{x}^*\}$ tal que $f(\vec{x}) \leq f(\vec{x}^*)$.

Este criterio de optimalidad se conoce como *Optimalidad Escalar*[3] [Caramia and Dell'Olmo, 2008].

Un dificultad que presentan los problemas de optimización monoobjetivo para su resolución es que muchas funciones f tienen óptimos locales, es decir, existen soluciones puntuales que cumplen el principio de Optimalidad Escalar en su entorno o vecindario, pero no para todo el conjunto de soluciones factibles. La presencia de óptimos locales dificulta la resolución de problemas de optimización porque, salvo que se conozcan previamente las propiedades matemáticas del problema a tratar, es difícil determinar si un óptimo es un óptimo solo en su entorno o en todo el conjunto factible.

[2] Un mínimo, según el formalismo elegido en esta tesis.

[3] Una definición equivalente, pero mas clara en términos de legibilidad sería: *una solución $\vec{x}^*$ es un óptimo global fuerte para el problema monoobjetivo si y solo si para todo $\vec{x} \in \vec{X} - \{\vec{x}^*\}$, $f(\vec{x}) > f(\vec{x}^*)$* (similar para el caso débil). Es decir, utilizando un cuantificador universal en lugar de la negación de un existencial. Sin embargo, la definición mediante negación es similar a la definición mas ampliamente utilizada de *Optimalidad de Pareto* para problemas multiobjetivo (al menos en la bibliografía consultada para el desarrollo de esta tesis). Esta definición se presentará mas adelante y será uno de los focos de esta tesis. Por lo tanto, se optó aquí por utilizar la versión de negación de un existencial a fin de facilitar la comparación entre definiciones de optimalidad en problemas monoobjetivo y multiobjetivo.

2.3. Optimización multiobjetivo determinística

En un problema de optimización multiobjetivo, la función $\vec{f}$ tiene más de una componente [Dennis and Schnabel, 1996], cada una de las cuales representan un objetivo diferente a minimizar (Figura 2.1). La principal dificultad de este tipo de problemas radica en que, en general, los objetivos son parcial o totalmente opuestos entre sí. En otras palabras, minimizar un objetivo suele lograrse en detrimento de los demás objetivos.

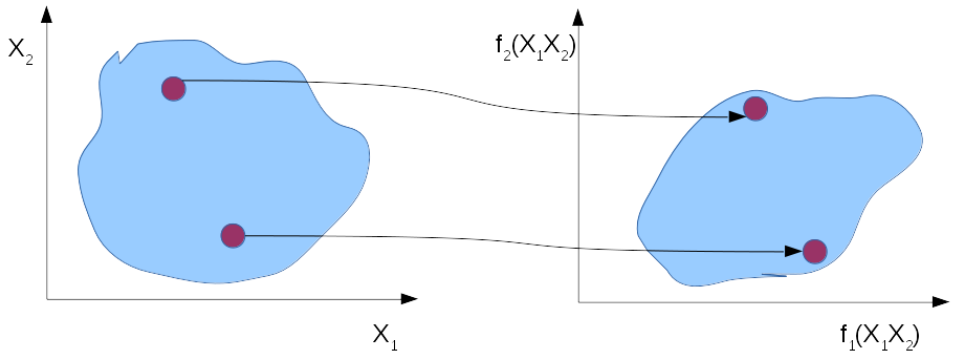


Figura 2.1: Espacio de decisión (izquierda) y de objetivos (derecha) de un problema multiobjetivo de dos variables y dos objetivos. Se observa como, en este caso, cada una de las dos soluciones resaltadas es mejor que la otra en uno de los objetivos y peor en el otro.

El concepto optimalidad escalar usado en problemas monoobjetivos no puede ser aplicado en forma directa a un problema multiobjetivo. Dicho concepto debe ser reemplazado por el criterio de *Optimalidad de Pareto* [Caramia and Dell'Olmo, 2008, Kaliszewski et al., 2016]. Una solución factible $\vec{x}$ es un *optimo de Pareto* si todas las demás soluciones factibles tienen asociado un valor superior para al menos uno de los objetivos, o bien tienen el mismo valor para todos los objetivos. Formalmente:

- Una solución $\vec{x}^*$ es un óptimo de Pareto débil para el problema multiobjetivo de m objetivos si y solo si no existe ningún $\vec{x} \in \vec{X} - \{\vec{x}^*\}$ tal que $f_i(\vec{x}) < f_i(\vec{x}^*), \forall i \in [1..m]$.
- Una solución $\vec{x}^*$ es un óptimo de Pareto fuerte para el problema multiobjetivo de m objetivos si y solo si no existe ningún $\vec{x} \in \vec{X} - \{\vec{x}^*\}$ tal que $f_i(\vec{x}) \leq f_i(\vec{x}^*), \forall i \in [1..m]$, con al menos una inequación cumpliéndose en forma estricta.

La *Frontera de Pareto* de un problema de optimización es el conjunto de todos los óptimos de Pareto, es decir, todas las soluciones factibles para las cuales es imposible mejorar un objetivo sin al menos empeorar otro. Cada una de estas soluciones se denominan *no-dominadas* y representan el mejor *trade-off* (concesión) posible entre objetivos. Por definición, cada solución no-dominada es equivalente al resto de las soluciones no-dominadas (presenta el mismo trade-off) y mejor a cualquiera de las soluciones *dominadas* (aquellas soluciones que presentan valores en sus objetivos que pueden ser mejorados sin empeorar ningún otro). La Figura 2.2 muestra un ejemplo de Frontera de Pareto para un problema de dos objetivos. Usando el criterio de Optimalidad de Pareto, la resolución de un problema de optimización multiobjetivo se convierte en la determinación de la Frontera de Pareto del mismo.

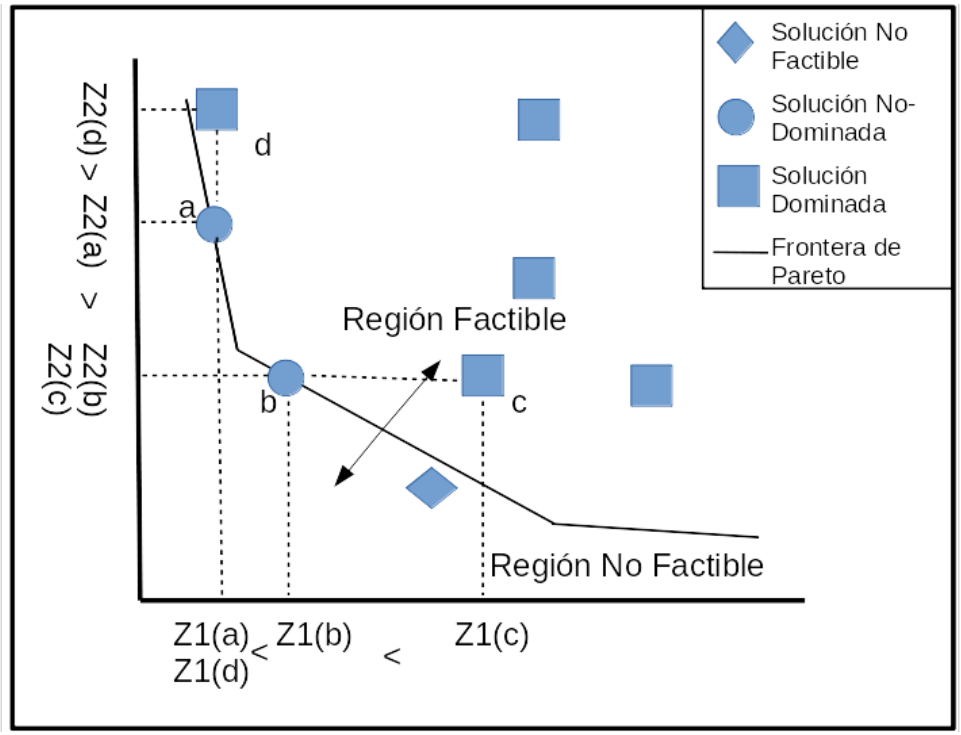


Figura 2.2: Frontera de Pareto para un problema de dos objetivos. Se observa como la Frontera separa el espacio de objetivos en un subespacio o región factible y otro no factible.

2.4. Resolución de problemas determinísticos

2.4.1. Problemas monoobjetivo

Existen muchas herramientas para la resolución de problemas monoobjetivo. De acuerdo a su naturaleza, se pueden agrupar en tres categorías[4]:

- Técnicas analíticas
- Técnicas heurísticas
- Técnicas metaheurísticas

Si el problema se encuentra formulado analíticamente, y su tamaño no es demasiado grande, se pueden aplicar técnicas matemáticas (basadas en Cálculo generalmente) para resolver el problema [Dennis and Schnabel, 1996]. Estas técnicas funcionan bien para problemas lineales o y algunos no-lineales, pero sus implementaciones numéricas pueden tener dificultades con problemas que posean muchos óptimos locales. Adicionalmente, varios de estos métodos necesitan que las funciones f y g sean diferenciables, lo cual puede suponer un impedimento en muchos casos. Existen algoritmos analíticos de carácter general, es decir, se pueden utilizar para resolver cualquier problema definido analíticamente, y algoritmos de carácter más específico, que permiten tratar solo ciertas estructuras matemáticas, tomando partido de sus características para volver más eficiente el cálculo de soluciones. Por ejemplo, si las relaciones entre todas las variables son lineales y el dominio de cada una de ellas es la recta real, el problema se puede resolver mediante herramientas de *Programación Lineal* (PL) [Dantzig, 1963]. Éstas son métodos algebraicos que garantizan la generación de soluciones óptimas. En el caso que una o más variables esté acotada al dominio entero, se pueden utilizar los algoritmos de resolución de problemas de *Programación Lineal Entera-Mixta* (PLE), los cuales también teóricamente permiten llegar al óptimo, pero a un costo muy superior que en el caso continuo [Schrijver, 1986, Vanderbei, 2008]. En efecto, mientras que la PL es un problema que pertenece a la clase de complejidad P, es decir para

[4]La descripción de las técnicas de optimización presentadas en este capítulo esta claramente sesgada a favor de las técnicas metaheurísticas, las cuales son usadas en esta tesis como bloque optimizador en el formalismo de OvS.

la cual existen algoritmos que pueden resolver una instancia del mismo en un tiempo que es polinomial en el tamaño de la instancia, la PLE es *NP-Hard*, es decir que no se conoce (y es improbable que exista) ningún algoritmo que lo resuelva en tiempo polinomial, llegando a ser intratable la resolución exacta de instancias de tamaño mediano o grandes.

Si los problemas no están definidos analíticamente, o bien lo están pero son del tipo *NP-Hard*, o bien se trata de instancias demasiado grandes, se puede intentar resolverlos mediante heurísticas o metaheurísticas. Una heurística es un procedimiento de resolución diseñado para un problema particular. No suelen ofrecer garantía de encontrar un óptimo global (aunque algunas si, y son muy eficientes en ello) pero, si están bien diseñadas, pueden encontrar soluciones cercanas a dicho óptimo global en una cantidad de tiempo razonable. Una buena heurística se basa en la estructura del problema que intenta resolver. El principal inconveniente con las mismas es que están diseñadas para el caso particular del problema que tratan, siendo complicado adaptarlas a variaciones del mismo. La principal ventaja es que suelen ser muy veloces [Rabadi, 2016].

Una metaheurística, por el contrario, es un procedimiento de resolución general, aplicable a varios tipos de problemas. No se basan en la utilización de conocimiento sobre el problema a resolver, sino que resuelven un problema genérico, debiendo el problema a resolver ser transformado a este problema genérico. Combinan tanto procedimientos de mejora local como estrategias de alto nivel para escapar de óptimos locales [Gendreau and Potvin, 2010, Luke, 2013] [5]. Tienen un costo computacional intermedio entre las heurísticas y los métodos analíticos [6]. Existen muchos algoritmos del este tipo, pero en esta tesis nos centramos fundamentalmente en tres algoritmos y sus variantes: el *Recocido Simulado* (SA, simulated annealing), los *Algoritmos Genéticos* (GA, genetic algorithm) y la *Optimización por Enjambre de Partículas* (PSO, por sus siglas en inglés)[7].

[5] Se denominan procedimientos de mejora local a las estrategias de optimización que, partiendo de una solución, tratan de encontrar un óptimo sin chequear la globalidad o localidad del mismo. El algoritmo *Hill Climbing* es un ejemplo de este tipo. La bibliografía referenciada aporta muchos ejemplos de estos tipos de algoritmos.

[6] En teoría, y solo dentro de cierto rango de tamaños. Para problemas muy pequeños, las técnicas analíticas pueden ser más rápidas, por ejemplo.

[7] La descripción de las metaheurísticas está también fuertemente sesgada a las familias utilizadas en los algoritmos diseñados en esta tesis.

Recocido Simulado

El SA es un algoritmo de búsqueda local (o de explotación [8]), que incluye estrategias de escape de óptimos locales, usado para resolver, principalmente, problemas de optimización combinatoria, aunque tiene aplicaciones en el ámbito de los problemas continuos. Basado en el proceso de recocido metalúrgico, el algoritmo busca imitar el reacomodamiento de las moléculas de un sólido metálico cuando es sometido a un enfriamiento lento. En esta situación, las moléculas tienden a agruparse en la configuración de mínima energía. En base a este concepto, el SA realiza una búsqueda local, en el cual a partir de una solución se busca en su vecindario (es decir, soluciones que se obtienen mediante pequeñas variaciones de la solución actual) por mejores soluciones. Si se encuentra alguna, esta es la nueva solución actual. Pero, debido a que este método puede llevar a una convergencia hacia un óptimo local, se incluye un proceso aleatorio (basado en un parámetro denominado *temperatura*) que permite seleccionar soluciones peores que la solución original, permitiendo escapar de la trampa de los óptimos locales.

En su forma básica, el SA es un algoritmo general, que no se basa en la estructura del problema a resolver. Por lo tanto, dicho problema debe ser codificado a los términos del SA. Esto implica que:

- Las variables deben ser codificadas en forma de un vector (generalmente de componentes enteras).
- Las función objetivo y las restricciones deben ser codificadas en una única función (Ecuación 2.5), en la cual se penalice las soluciones no factibles mediante una tasa α_i (es decir, las que no cumplen las restricciones, Ecuación 2.6).
- Se debe definir un método para construir vecinos para cualquier solución dada.

$$f_{sa}(\vec{x}) = f(\vec{x}) + \sum_{i=1}^{n} P_i(\vec{x}) \tag{2.5}$$

[8] Debido a que buscan óptimos realizando pequeñas variaciones a la solución actual, en vez de saltar de una zona del dominio a otra.

$$P_i(\vec{x}) = \begin{cases} 0, & g_i(\vec{x}) \leq r_i \\ \alpha_i \cdot (g_i(\vec{x}) - r_i), & g_i(\vec{x}) > r_i \end{cases} \quad \forall i \in [1..n] \tag{2.6}$$

El pseudocódigo de un SA básico [Brownlee, 2012] se muestra en el Algoritmo 1.

Algoritmo 1 Recocido Simulado

```
1: procedure RECOCIDOSIMULADO(maxTemp, maxIter, numVar)
2:     S_actual ← crearSolucionInicial(numVar)
3:     S_mejor ← S_actual
4:     i ← maxTemp
5:     while i ≠ 0 do
6:         temp_actual ← calcularTemp(i, maxTemp, maxIter)
7:         S_i ← crearVecino(S_actual)
8:         if costo(S_i) < costo(S_actual) then
9:             S_actual ← S_i
10:            if costo(S_i) < costo(S_mejor) then
11:                S_mejor ← S_i
12:            end if
13:        else
14:            if Exp((costo(S_i) − costo(S_mejor)) / temp_actual) > Aleatorio(0, 1) then
15:                S_actual ← S_i
16:            end if
17:        end if
18:    end while
19:    return S_mejor
20: end procedure
```

El SA toma como entradas el tamaño del vector de soluciones, la temperatura máxima y la cantidad máxima de iteraciones. En las líneas 2-4 se inicializa el algoritmo, mientras que la búsqueda se realiza en las líneas 5-18. En la línea 6 se calcula la temperatura de la iteración actual, en la 7 se crea un vecino de la solución actual y en las líneas 8-17 se evalúa dicho vecino. Si el vecino es mejor que la solución actual, el vecino es la nueva solución actual (lineas 8-9). Si además es mejor que la mejor solución encontrada hasta el momento, el vecino es la nueva mejor solución (líneas 10-12). En caso que el vecino sea peor que la solución actual, este tiene una probabilidad decreciente con la temperatura para ser seleccionado como nueva solución actual (lineas 14-16). La función *Aleatorio(a,b)* que es utilizada en la línea 14 es una función que genera un número aleatorio con distribución uniforme en el intervalo (a, b). El número aleatorio generado es distinto cada vez que la función es llamada.

En cierta forma, se podría describir a este algoritmo no como un algoritmo de búsqueda local, sino como uno que converge a la búsqueda local. En sus primeras iteraciones, debido a que la temperatura es alta, la posibilidad de seleccionar una solución desfavorable respecto a la actual es alta. Pero a medida que las iteraciones avanzan, este comportamiento va desapareciendo y la probabilidad de seleccionar soluciones que generen mejoras incrementales es cada vez mayor.

Al igual que en una búsqueda local (y en métodos basados en búsquedas locales), la estructura que tiene el vecindario de cada solución juego un rol fundamental en esta metaheurística. Dada una solución, una vecina de ésta es otra solución que se construye a partir de la primera mediante alguna operación, usualmente la aplicación de una pequeña modificación. Al diseñar un vecindario, es deseable que la operación involucrada sea veloz (desde el punto de vista computacional) y, de ser posible, que cualquier elemento del espacio de soluciones pueda ser alcanzado por cualquier otro mediante la sucesiva aplicación de esta operación.

Existen múltiples variantes respecto del algoritmo base de SA, las cuales están enfocadas a resolver distintos tipos de problemas [Henderson et al., 2003].

Algoritmo Genético

Los GA, a diferencia del SA, son algoritmos poblacionales de exploración global. Es decir, no tratan de mejorar una solución inicial, sino que buscan en todo el dominio analizando múltiples soluciones en simultáneo.

Los GA están basados en el proceso de la selección natural y la herencia. Comenzando con un conjunto de individuos (codificados por sus cromosomas), los mismos son sometidos a su entorno, sobreviviendo solo los más aptos. Estos individuos luego se reproducen entre sí, generando una nueva población que los sobrevive, con el aliciente que algunos individuos de esta población poseen características alteradas aleatoriamente mediante mutación. En su implementación algorítmica, los cromosomas se codifican como vectores binarios o reales, las poblaciones como un conjunto de cromosomas, el proceso de selección se basa en la evaluación de las soluciones mediante un función como la indicada en la Ecuación 2.5, la generación de descendientes se realiza mediante truncamiento, reordenado y/o combinación de los cromosomas, y las mutaciones

mediante un proceso de alteración aleatorio.

El Algoritmo 2 muestra el pseudocódigo del GA canónico [Brownlee, 2012]. Un GA no devuelve una única solución, sino un conjunto de soluciones, la última población generada. En esta población final, la mayoría de las soluciones deberían converger a la mejor solución hallada.

Algoritmo 2 Algoritmo Genético

```
1: procedure ALGORITMOGENETICO(numIndiv, numVar, maxGen)
2:     P_actual ← crearPoblacionInicial(numIndiv, numVar)
3:     i ← 0
4:     while i < maxGen do
5:         P_mejores ← seleccionarMejores(P_actual)
6:         P_descendientes ← cruzamiento(P_mejores)
7:         P_actual ← mutacion(P_descendientes)
8:     end while
9:     return P_actual
10: end procedure
```

El GA toma como entradas la cantidad de soluciones que componen a una población, el tamaño del vector que codifica a cada solución y el número de generaciones máximas (o iteraciones). Las líneas 2-3 inicializan el algoritmo, mientras que la exploración se realiza en las líneas 4-8. En la línea 5 se evalúan y seleccionan las mejores soluciones de la población actual, en la línea 6 se genera una nueva población mediante el proceso de cruzamiento realizado a la selección anterior, y en la línea 7 se modifican aleatoriamente algunas soluciones de esta nueva población.

Existen muchas variantes de los GA [Shukla et al., 2015, Shrestha and Mahmood, 2016], en las cuales se modifica el tipo de componente de los cromosomas (binario o real), la forma de selección (simple, por torneo, etc), la forma de cruzamiento (1 punto, 2 puntos, etc), la forma de mutación y la posibilidad de usar elitismo (esto es, permitir que elementos de la población actual permanezcan en la población de descendientes)[Kumar et al., 2010, Umbarkar and Sheth, 2015].

Si bien el GA es principalmente un algoritmo de exploración, a medida que las iteraciones avanzan, adquiere características de explotación, ya que la diversidad dentro de la población tiende a disminuir por efectos de la presión selectiva [9].

[9]Es decir, cuando en una iteración determinada la población actual esta compuesta de muchas soluciones buenas, es poco probable que el algoritmo pueda generar soluciones malas para la próxima iteración.

Optimización por Enjambre de Partículas

Los PSO, al igual que los GA, son algoritmos poblacionales, que hacen evolucionar a un conjunto de soluciones (llamadas *partículas* aquí) en simultáneo en vez de una única solución por iteración. Se basan en el movimiento de bandadas de aves [Brownlee, 2012].

En el PSO canónico, todas las partículas quedan definidas en base a su posición p y velocidad v en un espacio $n - dimensional$. Iteración tras iteración, cada una evoluciona en base a la información histórica individual que recolectó y en base a la información histórica global de toda la bandada, actualizando su posición y velocidad. Esta evolución se regula según las ecuaciones 2.7 y 2.8:

$$v_{i,j}(t+1) = w \times v_{i,j}(t) + c_1 \times r_1 \times (p_{i,j}(t) - x_{i,j}(t)) + c_2 \times r_2 \times (p_{g,j}(t) - x_{i,j}(t)) \tag{2.7}$$

$$x_{i,j}(t+1) = x_{i,j}(t) + v_{i,j}(t+1) \tag{2.8}$$

Donde w es el factor de inercia, $v_{i,j}$ es la componente de la velocidad de la partícula i en la dimensión j, c_1 y c_2 es la ponderación de la información individual y global en cada actualización, r_1 y $r_2 \sim U(0,1)$ (donde $U(a,b)$ se refiere a la distribución uniforme de valores en el intervalo (a,b)) representan la aceleración diferencial de cada partícula, p_i es el mejor valor encontrado por la partícula i ($pbest$) y p_g es la mejor solución encontrada en toda la bandada ($gbest$), la cual es conocida como la solución *lider*. En cada iteración se actualiza el valor de la velocidad y luego el valor de la posición para cada partícula. El pseudocódigo asociado a la versión canónica de esta metaheurística se muestra en el Algorítmo 3. En las líneas 7-8 se representa el cálculo de las Ecuaciones 2.7 y 2.8. Como en todas las metaheurísticas, existe múltiples variantes de este esquema básico [García-Gonzalo and Fernández-Martínez, 2012].

2.4.2. Problemas multiobjetivo

La resolución de problemas multiobjetivo parte de la base de adaptar métodos monoobjetivo. Hay muchos formas de tratar problemas con múltiples objetivos, pero según su finali-

Algoritmo 3 Optimización por Enjambre de Partículas

```
1: procedure ALGORITMOENJAMBRE(numParticulas, numVar, maxIter)
2:     Evaluaciones ← ()
3:     P_actual ← crearPoblacionInicial(numParticulas, numVar)
4:     i ← 0
5:     while i < maxIter do
6:         Evaluaciones ← evaluarSoluciones(P_actual)
7:         P_siguiente ← actualizarVelocidades(P_actual, Evaluaciones)
8:         P_siguiente ← actualizarPosiciones(P_actual, Evaluaciones)
9:         P_actual ← P_siguiente
10:    end while
11:    return P_actual
12: end procedure
```

dad se podrían clasificar en dos grupos [Caramia and Dell'Olmo, 2008, Antunes et al., 2016, Elarbi et al., 2017]:

1. Búsqueda de un óptimo
2. Búsqueda de la Frontera de Pareto

El primer grupo engloba a un conjunto de métodos de resolución que buscan transformar el problema multiobjetivo en uno monoobjetivo y resolverlo por cualquiera de los métodos descriptos en la Sección 2.4.1. Dentro de ellos encontramos:

1. Optimización por pesos
2. Optimización por metas
3. Asignación de prioridades

La *optimización por pesos* consiste en definir una nueva función objetivo f' que consiste en la suma pesada de cada una de las componentes de $\vec{f}$ (Ecuación 2.9). w_i es el peso que el tomador de decisión le asigna a cada uno de los i objetivos.

$$f' = \sum_{i=1}^{|\vec{f}|} w_i \cdot f_i \tag{2.9}$$

Este método transforma un problema multiobjetivo en uno monoobjetivo, el cual se resuelve aplicando el criterio de optimización escalar al problema. Si bien es simple y poco costoso computacionalmente hablando[10], es muy sensible a la selección de pesos. Si los objetivos se miden en la misma unidad y existe una forma no-subjetiva de definir los pesos de cada objetivo, esta alternativa es muy buena, pero en caso contrario, el método se vuelve muy arbitrario.

El método de *optimización por metas* es similar al anterior, pero en vez de pesar los objetivos y sumarlos se define una meta para cada objetivo y se miden los desvíos respecto de dichas metas. Los desvíos luego se pesan y se consolidan en una única función, que trata entonces de minimizar la suma de los desvíos absolutos. La Ecuación 2.10 muestra como se calcula la nueva función objetivo f', en donde m_i es la meta deseada para el objetivo i.

$$f' = \sum_{i=1}^{|\vec{f}|} w_i \cdot |f_i - m_i| \tag{2.10}$$

Nuevamente, el método presenta mucha subjetividad, ya que no solo hay que determinar los pesos w_i sino que hay que definir el nivel deseable para cada objetivo.

El método de *asignación por prioridades* consiste en definir un orden de prioridad entre los objetivos (siendo el mas prioritario el objetivo $i = 1$, el segundo mas prioritario el objetivo $i = 2$ y así sucesivamente), optimizando solamente el objetivo de mayor prioridad y fijando cotas superiores para cada uno de los otros objetivos. Los objetivos acotados se convierten en restricciones y se busca entonces minimizar uno de los objetivos asegurando un valor límite para el resto.

$$f' = f_1(\vec{x}) \tag{2.11}$$

$$f_i(\vec{x}) \leq m_i, \forall i = 2..|\vec{f}| \tag{2.12}$$

Las Ecuaciones 2.11 y 2.12 muestran la transformación necesaria, en donde m_i es la cota superior del objetivo i.

[10] Comparado con otras técnicas multiobjetivo

Todos los métodos anteriores tienen el defecto de añadir subjetividad al proceso de optimización. El tomador de decisiones debe definir pesos y/o metas y el resultado obtenido depende fuertemente de esa definición inicial. Los métodos de optimización basados en calcular la *Frontera de Pareto* eliminan esta subjetividad del proceso de optimización, devolviéndole al tomador de decisiones un conjunto de soluciones equivalentes, sin pesar ninguna de ella, siendo solamente subjetiva la selección de cual alternativa elegir.

Existen tres formas principales de generar la Frontera de Pareto de un problema [Elarbi et al., 2017]:

- Pesando objetivos iterativamente.
- Buscando soluciones no-dominadas.
- Optimizando métricas de calidad.

Pesado iterativo de objetivos

Mediante este método, se normalizan las diferentes funciones objetivo y se resuelven tantos problemas de optimización como combinaciones de pesos se puedan evaluar, con la condición que todos los pesos deben sumar 1. Es decir, se plantea el siguiente generador de problemas y se busca el óptimo de cada uno de ellos (Ecuación 2.13).

$$f' = \sum_{i=1}^{|\vec{f}|} w_i \cdot f_i, \forall W = (w_1, ..., w_{|\vec{f}|}) \in \mathbb{R}_+^{|\vec{f}|} \setminus \sum_{i=1}^{|\vec{f}|} w_i = 1 \tag{2.13}$$

Para cada problema concreto, es decir, para un determinado W, su solución óptima se corresponde con un punto de la Frontera de Pareto (pudiendo varios W tener asociado el mismo punto de la frontera). Obviamente no se pueden generar y optimizar los infinitos diferentes problemas que arroja esta formulación, por lo que en la práctica se utiliza alguna metodología de selección de valores de W que aseguren una distancia mínima entre ellos. Es decir, la implementación práctica de esta metodología no garantiza construir la Frontera de Pareto completamente, pero si que al menos se van a encontrar puntos que correspondan a ella. Nótese que, de acuerdo a esto, de los métodos que tratan la optimización multiobjetivo mediante el criterio de Optima-

lidad Escalar (es decir, transforman el problema a uno monoobjetivo), el método de pesado es el único que garantiza que la solución hallada es un óptimo de Pareto.

Un ejemplo de ese tipo de algoritmos es el *Adaptive Weighted Sum Method* [Kim and Weck, 2006, Ryu et al., 2009], en el cual los pesos son recalculados en cada iteración mediante un algoritmo de búsqueda hacia adelante y atrás.

Búsqueda de no-dominancia

La construcción de la Frontera de Pareto mediante la búsqueda de soluciones no-dominadas utiliza un enfoque totalmente diferente. Partiendo de un conjunto de soluciones, trata de encontrar variaciones de las mismas que dominen a las soluciones originales. El proceso se repite iterativamente, con el objetivo que el conjunto de soluciones no-dominadas de cada iteración tienda a converger con la Frontera de Pareto [Deb, 2001, Durillo et al., 2009, Carrasqueira et al., 2015]. Este método no garantiza convergencia con la Frontera de Pareto, pero empíricamente arroja buenos resultados siendo, en general, más eficiente en su implementación que el método anterior.

Un ejemplo de este tipo de algoritmos es el NSGA-II [Deb et al., 2002], el cual es usado como base en muchos de los desarrollos presentados en esta tesis. Éste es un algoritmo perteneciente a la familia de los GA adaptado a la optimización multiobjetivo [Deb et al., 2002]. A diferencia de un GA convencional, en el cual la evaluación de cada individuo coincide con el valor de la función objetivo, aquí se usa el criterio de no-dominancia y la distancia de agrupamiento (CD) para determinar cuando una solución es mejor que otra. Cada individuo es calificado en base a la posición que ocupa en la jerarquía de no-dominancia y, para elementos del mismo nivel, en base a la cercanía a otras soluciones.

En cada iteración del algoritmo, el valor de todos los objetivos de cada individuo es calculado y en base a estos valores se calcula el ranking de no-dominancia o de Pareto entre todos los individuos. Para ello, se compara cada individuo con el resto de la población para determinar si es dominado por algún otro individuo. Si no es dominado por ninguno, se le asigna un valor de ranking 1. Cuando todas los individuos fueron comparados, todos los no-dominados (aquellos con ranking 1) se retiran temporalmente de la población y el proceso es repetido con

los individuos restantes. A los individuos no-dominados de esta iteración se les asigna el valor 2 en el ranking y son retirados temporalmente de la población. El proceso se repite hasta que todos los individuos tengan asignado un valor en el ranking. Cada conjunto de individuos con el mismo valor de ranking representan soluciones equivalentes, que dominan a todas las de valor de ranking mayor y son dominadas por todas las de valor de ranking menor. A fin de tener un criterio de orden entre las soluciones con el mismo valor de ranking, se calcula la distancia de agrupamiento de cada solución. Para ello se ordenan las soluciones en forma creciente para cada objetivo y, para cada solución, se calcula la distancia promedio de las soluciones inmediatamente anterior y posterior.

Con cada individuo rankeado y con una distancia de agrupamiento asignada, se puede construir un operador de comparación $\prec_n$denominado Operador de Comparación de Agrupamiento (CCO) para determinar cuando una solución es mejor que otra. Asumiendo que el problema es de minimización, $\prec_n$ se define según la Ecuación 2.14, donde i_{rank} y j_{rank} son los valores del ranking de Pareto de los individuos i y j mientras que $i_{crownDist}$ y $j_{crownDist}$ las distancias de agrupamiento de dichos individuos.

$$i \prec_n j \; si \; (i_{rank} < j_{rank}) \; o \; [(i_{rank} = j_{rank}) \; y \; (i_{crownDist} > j_{crownDist})] \tag{2.14}$$

El algoritmo básico de NSGA-II es similar al de un GA con torneo y elitismo [Deb et al., 2002, Luke, 2013] [11] [12]. En cualquier generación m, se comienza con una población P_m de tamaño n y se evalúa cada individuo, calculándose los valores de ranking de Pareto y distancia de agrupamiento. Luego, se crea la población de descendientes Q_m seleccionando individuos mediante torneo binario, cruzamiento y mutación. En base a estas dos poblaciones se define una nueva población $R_m = P_m \cup Q_m$ y se seleccionan los mejores n elementos de R_m usando el operador $\prec_n$. Estos individuos conforman la población P_{m+1}. El proceso se repite hasta que se alcanza el número de generaciones máximas u otro criterio de terminación definido.

PSO también dispone de adaptaciones al mundo multiobjetivo. En esta tesis se utilizó también

[11] Elitismo consiste en permitir que elementos de la población actual persistan en la población de descendientes.

[12] Selección por torneo es un método de selección en el cual, en vez de elegir los mejores en forma global, se realiza un proceso de selección jerárquico de a pares. Por ejemplo, dado los pares (A,B) y (C,D), se eligen los mejores de cada uno, digamos A y C, y estos compiten nuevamente entre sí, para ver quien es el mejor.

el *Speed-constrained Multi-objective PSO* (SMPSO) [Durillo et al., 2009]. Ha demostrado tener muy buena performance comparado con otros algoritmos multiobjetivo, rivalizando incluso con el NSGA-II [Carrasqueira et al., 2015]. El problema principal en cualquier adaptación multiobjetivo basada en Fronteras de Pareto para PSO consiste en cómo seleccionar el líder de cada iteración (el *gbest*). Aquí se encara la selección de líderes por medio de una lista de tamaño fijo de soluciones líderes, seleccionadas del conjunto de soluciones no-dominadas. En cada iteración se agregan y quitan soluciones de la lista siguiendo el criterio de no-dominancia y la distancia de agrupamiento (CD). El cálculo de la no-dominancia sigue un criterio de dominancia no estricta o $\epsilon - dominancia$. Con este criterio, una solución q se considera dominante respecto de otra solución s si para todos los objetivos $f(q)/(1+\epsilon) \leq f(s)$ y para al menos un objetivo $f(q)/(1+\epsilon) < f(s)$. También, a efecto de evitar sesgos, los parámetros c_1 y c_2 se seleccionan aleatoriamente, al igual que r_1 y r_2. Por último, se incorpora un proceso de mutación en las soluciones. SMPSO agrega una restricción al cálculo de velocidades, con el fin de impedir que la posición de las partículas oscile entre sus extremos factibles sin evaluar puntos intermedios. Mas precisamente, para el caso que $|v_{i,j}(t)| \geq delta$, se reasigna $v_{i,j}(t) \leftarrow signo(v_{i,j}(t)) \cdot delta$, donde *delta* se calcula según la Ecuación 2.15.

$$delta_j = \frac{upperLimit_j - lowerLimit_j}{2} \tag{2.15}$$

Donde $upperLimit_j$ y $lowerLimit_j$ son, respectivamente, los límites superiores e inferiores para cada dimensión j.

Optimización de métricas de calidad

Esta familia de métodos se basa en el uso de técnicas monoobjetivo para hacer evolucionar una población de soluciones con el fin de optimizar alguna de las métricas de calidad que serán vistas en la Sección 2.5. Es decir, se resuelve un problema de optimización derivado del original, en el cual las variables de decisión permiten definir un conjunto de soluciones (mediante, por ejemplo, la unión de las representaciones de todas las soluciones del conjunto) y en el cual la función a optimizar es un indicador que define el nivel de calidad de la estimación realizada

de la Frontera de Pareto. El método, si bien bueno, tiene el problema que el costo de computo de las métricas es muy alto (costo al que se debe sumar la evaluación de cada solución de la población). Ejemplos son el *Indicator-Based Evolutionary Algorithm* [Elarbi et al., 2017] y el *Weighted Stress Function Method* [Denysiuk and Gaspar-Cunha, 2017].

2.5. Métricas de calidad para optimización determinística

Un aspecto crucial en el diseño y la selección de algoritmos de optimización es la evaluación de su desempeño. Dejando de lado que tan general es el algoritmo, a fin de resolver un problema puntual existen dos características que definen que tan bueno es dicho procedimiento: la calidad de las soluciones halladas y el costo computacional. Si bien no es una relación exacta, en general los algoritmos que arrojan mejores soluciones son más costosos que los algoritmos que arrojan peores. Por lo tanto, el tomador de decisiones debe realizar un trade-off entre estas dos propiedades.

El costo computacional es simple de medir empíricamente y sencillo de entender en forma intuitiva. Básicamente hace referencia a cuantos recursos computacionales se consumen para resolver el problema. La memoria necesaria puede ser un factor importante en muchos problemas, pero en general el costo está asociado al tiempo medio de procesamiento en un microprocesador determinado. Este costo tiende a aumentar a medida que aumenta el tamaño de la instancia a resolver.

La calidad de un algoritmo se refiere a que tan buenas soluciones arroja en promedio. En otras palabras, que tan cercanas están del óptimo global. En el caso monoobjetivo, suponiendo que conocemos el óptimo global, es fácil medir la calidad del algoritmo, basta con comparar los resultados. Para el caso multiobjetivo, dado que se quiere aproximar una curva, el proceso es más complicado. Existen decenas de métricas desarrolladas, por lo cual en esta tesis se utilizaran cuatro de las más frecuentemente citadas en la bibliografía [Zitzler and Thiele, 1999, Zitzler et al., 2007, Jiang et al., 2014, Riquelme et al., 2015, Liefooghe and Derbe, 2016].

Una forma de evaluar la concordancia de nuestra aproximación con la Frontera de Pareto es medir la distancia media de las dos curvas (es decir, nuestra curva aproximada y la real). Suponiendo que se conoce la Frontera de Pareto asociada al problema (llamémosla *Real*)[13], se puede utilizar la métrica conocida como *Distancia Generacional* (GD). Este método promedia la distancia euclidiana entre las curvas, según la Ecuación 2.16

$$GD = \frac{\sqrt{\sum_{i=1}^{N} d_i^2}}{N} \tag{2.16}$$

Donde N es el número total de soluciones que componen la Frontera de Pareto calculada por el algoritmo y d_i es la distancia euclidiana mínima desde la solución i a la Frontera de Pareto Real. Mientras más pequeño es el valor de este indicador, más cercana está la curva medida a la Frontera de Pareto Real. Es una medida parcial de la convergencia a la Frontera de Pareto Real.

Por otro lado, con el indicador de *Uniformidad* (Δ, GS) se puede medir la cobertura o diversidad de la Frontera estimada respecto de la real. Se calcula mediante la Ecuación 2.17, donde e_i son los M puntos extremos de la Frontera de Pareto para cada objetivo[14], $d(e_i, S)$ es la mínima distancia euclidiana desde la curva S a medir (es decir, la curva generada por el algoritmo) al punto e_i, d_i es la distancia desde la solución i a la solución $i+1$ en la curva S, $\bar{d}$ es la media de N distancias d_i y $N = |S|$ es el número de puntos en la curva S.

$$\Delta = \frac{\sum_{i=1}^{M} d(e_i, S) + \sum_{i=1}^{N} |d_i - \bar{d}|}{\sum_{i=1}^{M} d(e_i, S) + \bar{d} \cdot N} \tag{2.17}$$

Mientras más pequeño el valor de este indicador, más uniformemente distribuida está la estimación de la Frontera de Pareto medida. Es decir, valores grandes de este indicador denotan que la aproximación hallada es parcial y tiende a agrupar muchas soluciones en pequeñas áreas.

[13] Obviamente, en una aplicación real de los algoritmos, dicha frontera no se conoce. Uno debe confiar en que el algoritmo resolvió problemas similares en el pasado, o bien, realizar pequeños experimentos para intentar extrapolar la bondad o no del mismo. Por ejemplo, diseñar una instancia mas chica, estimar la frontera real ejecutando 100 veces el algoritmo, consolidar todas las fronteras en una sola, y medir los resultados de una nueva ejecución contra esta frontera consolidada.

[14] Los puntos extremos de la Frontera de Pareto para cada objetivo son los puntos correspondientes a los valores máximos y mínimos de cada objetivo en la frontera.

Una medida más general es el *Hipervolumen Dominado* (DH) [Zitzler and Thiele, 1999]. Dicho indicador es una medida de calidad que evalúa al mismo tiempo la distancia de la Frontera de Pareto aproximada frente a la Frontera de Pareto de referencia[15], así como también la amplitud de la frontera estimada. Condensa en un mismo indicador las medidas de convergencia y diversidad. Ésta métrica computa el volumen del espacio de objetivos cuyos puntos son dominados por la curva evaluada. Mientras mayor es su valor, mejor es la aproximación. La cota máxima de este indicador es el hipervolumen dominado por la Frontera de Pareto real del problema [Zitzler et al., 2007]. Para calcularlo se define un punto W exterior a la curva (la peor solución encontrada, por ejemplo) y se construye un hipercubo de volumen v_i por cada punto i en la curva Q a medir. La unión de todos los hipercubos determina el hipervolumen total (Ecuación 2.18).

$$DH = volumen(\bigcup_{i=1}^{|Q|} v_i) \tag{2.18}$$

En este caso, mientras más parecido sea el hipervolumen de nuestra aproximación al de la Frontera de Pareto Real, mejor es el algoritmo.

Una medida similar al DH (por su generalidad) es el *Indicador Épsilon Aditivo* (AEI). Éste proporciona un factor que indica que tan distinta es una curva respecto a una curva patrón (o de referencia). En el contexto de Fronteras de Pareto, este factor representaría el factor de escala máxima (de entre los componentes de ambas curvas) por el cual se puede transformar la curva evaluada en la Frontera de Pareto. Mientras más pequeño es, más cerca está nuestra curva de la Frontera de Pareto. Su forma de cálculo se muestra en la Ecuación 2.19, donde R es la Frontera de Pareto, A es la curva medida y d el número de objetivos

$$AEI_{(+)} = \max_{r \in R} \min_{a \in A} \max_{i \in \{1..d\}} (a_i - r_i) \tag{2.19}$$

[15]La Frontera de Pareto de referencia, idealmente, debería ser la Frontera de Pareto Real. Pero este indicador se puede utilizar también para comparar dos Fronteras de Pareto aproximadas (para el mismo problema) y determinar cual de las dos es mejor aproximación (la mejor es la que arroje el valor mas grande para este indicador). Aquí, y en las referencias subsiguientes, el término *de referencia* atañe a la curva con la cual deseamos comparar nuestra aproximación.

2.6. Optimización multiobjetivo estocástica

Como se mencionó en la Sección 2.1, si $\vec{f}$, $\vec{g}$ y/o $\vec{r}$ son estocásticas, estamos ante un problema de optimización estocástico. Los problemas que se suelen encontrar en el mundo real son de este tipo, con múltiples objetivos y aleatoriedad en los objetivos y restricciones. Sin embargo, debido entre otras cosas al abaratamiento de los sistemas de cómputo, no fue hasta hace relativamente poco tiempo que la interacción entre estos dos paradigmas de optimización (estocástico y multiobjetivo) comenzó a ser tratado en profundidad [Adeyefa and Luhandjula, 2011, Abdelaziz, 2012, Gutjahr and Pichler, 2016]. De las múltiples alternativas en que pueda presentar y resolver esta aleatoriedad, en esta tesis se trata únicamente el caso en el cual solo $\vec{f}$ es aleatoria[16]. Por lo tanto, en adelante, cuando se hable de optimización multiobjetivo estocástica, se estará refiriendo a la optimización de problemas en los cuales al menos uno de sus objetivos no es determinístico. En la literatura se pueden encontrar tres familias de métodos para lidiar con estos tipos de problemas:

- El método estocástico
- El método multiobjetivo
- El método de dominancia estocástica (o probabilística)

El *método estocástico* consiste en reducir el problema a uno monoobjetivo estocástico (por ejemplo, mediante las técnicas de búsqueda de un objetivo vistas en la Sección 2.4.2). Y luego aplicar alguna de las técnicas de resolución que provee la optimización estocástica. Este método no será tratado aquí, porque no permite generar Fronteras de Pareto [Caballero et al., 2004].

En el *método multiobjetivo*, el tratamiento de problemas estocásticos consiste en transformarlos en versiones determinísticas [Caballero et al., 2004, Gutjahr, 2005, Tricoire et al., 2012, Hao et al., 2015, Gruler, 2018]. Para ello, en vez de intentar optimizar la función $\vec{f}$, se busca optimizar un indicador estocástico calculado en base a dicha función, usualmente la esperanza matemática [17] (Ecuación 2.4). Si se conoce a priori la distribución estadística de la población

[16] Es decir, a la búsqueda de Fronteras de Pareto en condiciones de aleatoriedad en la función objetivo

[17] La mediana, o el valor del percentil 95 %, son otras alternativas usuales.

asociada a $\vec{f}$, la esperanza podría calcularse analíticamente, pero en general la misma se calcula mediante algún tipo de simulación [Ruszczynski and Shapiro, 2009, Chen and Lee, 2011, Song, 2013, Diwekar and David, 2015]. Es decir, se definen un conjunto fijo de escenarios DS y, para cada solución $\vec{x}$ a evaluar, $\vec{f}(\vec{x})$ es calculada en cada uno de dichos escenarios. Luego, las replicaciones de $\vec{f}(\vec{x})$ son promediadas, obteniéndose así una estimación de $\vec{E}(\vec{f}(\vec{x}))$ (o del indicador estocástico elegido). Entonces, dado el conjunto de escenarios DS, la Ecuación 2.1, con $\vec{f}$ estocástica, se transformaría en la Ecuación 2.20 (o su equivalente, la Ecuación 2.4).

$$Min \ \frac{1}{|DS|} \cdot \sum_{j \in DS} \vec{f}(\vec{x})_j \tag{2.20}$$

Donde $\vec{f}(\vec{x})_j$ es la evaluación de la función $\vec{f}(\vec{x})$ en el escenario j.

Para el caso monoobjetivo, el principio de optimalidad escalar puede ser fácilmente redefinido usando el estadístico seleccionado. Si el estadístico elegido es la esperanza:

- Una solución $\vec{x}^*$ es un óptimo global débil para el problema monoobjetivo estocástico si y solo si no existe ningún $\vec{x} \in \vec{X} - \{\vec{x}^*\}$ tal que $E(f(\vec{x})) < E(f(\vec{x}^*))$.
- Una solución $\vec{x}^*$ es un óptimo global fuerte para el problema monoobjetivo estocástico si y solo si no existe ningún $\vec{x} \in \vec{X} - \{\vec{x}^*\}$ tal que $E(f(\vec{x})) \leq E(f(\vec{x}^*))$.

Para el caso multiobjetivo, bajo este método de resolución, el principio de optimalidad de Pareto se redefine como:

- Una solución $\vec{x}^*$ es un óptimo de Pareto débil para el problema multiobjetivo de m objetivos si y solo si no existe ningún $\vec{x} \in \vec{X} - \{\vec{x}^*\}$ tal que $E(f_i(\vec{x})) < E(f_i(\vec{x}^*)), \forall i \in [1..m]$.
- Una solución $\vec{x}^*$ es un óptimo de Pareto fuerte para el problema multiobjetivo de m objetivos si y solo si no existe ningún $\vec{x} \in \vec{X} - \{\vec{x}^*\}$ tal que $E(f_i(\vec{x})) \leq E(f_i(\vec{x}^*)), \forall i \in [1..m]$, con al menos una inequación cumpliéndose en forma estricta.

Partiendo de un problema de optimización sobre una función $\vec{f}$ estocástica, se llega a un problema de optimización de $\vec{E}(\vec{f})$, el cual es determinístico en el caso que se estime $\vec{E}$ sobre un

conjunto DS fijo. La Frontera de Pareto obtenida no se corresponde, a priori, con la de ningún escenario, sino con el desempeño medio de cada una de las soluciones. Este método funciona muy bien cuando la aleatoriedad a la que está sujeta la función objetivo es baja. Pero en el caso que los escenarios presenten alta variabilidad, no hay garantías que las soluciones encontradas estén cerca de la Frontera de Pareto asociada al escenario efectivamente presentado.

Una manera de evitar este último problema es incluyendo la aleatoriedad en el proceso de determinación de la dominancia. Esto es lo que hace el tercer grupo de métodos, la *dominancia estocástica*. En este sentido, [Gannouni et al., 2017] proponen una nueva definición de optimalidad de Pareto para el caso estocástico. Siendo $P(f_i(\vec{x}) \prec_n f_i(\vec{x}^*))$ la probabilidad que la solución $\vec{x}$ domine a la solución $\vec{x}^*$ en el conjunto DS, el criterio propuesto por dichos autores se puede expresar como:

- Una solución $\vec{x}^*$ es un óptimo de Pareto estocástico débil para el problema multiobjetivo de m objetivos si y solo si no existe ningún $\vec{x} \in \vec{X} - \{\vec{x}^*\}$ tal que $P(f_i(\vec{x}) \prec_n f_i(\vec{x}^*)) > P(f_i(\vec{x}^*) \prec_n f_i(\vec{x}))$.
- Una solución $\vec{x}^*$ es un óptimo de Pareto estocástico fuerte para el problema multiobjetivo de m objetivos si y solo si no existe ningún $\vec{x} \in \vec{X} - \{\vec{x}^*\}$ tal que$P(f_i(\vec{x}) \prec_n f_i(\vec{x}^*)) \geq P(f_i(\vec{x}^*) \prec_n f_i(\vec{x}))$, con al menos una inequación cumpliéndose en forma estricta.

Es decir, una solución domina a otra si, en el conjunto de escenarios DS, la primera es no-dominada respecto de la segunda una mayor cantidad de veces que el caso contrario. Este método no garantiza que las soluciones encontradas pertenezcan a la Frontera de Pareto de algún escenario puntual, pero si nos permite obtener soluciones que tienen menos chances de ser dominadas que todas las demás soluciones evaluadas.

Capítulo 3

Optimización vía Simulación

Se presenta en este capítulo la metodología conocida como *Optimización vía Simulación* (OvS), sus variantes y los tipos de problemas que se pueden abarcar con esta técnica. Al final del capítulo se introduce el concepto de metamodelo.

3.1. Optimización vía Simulación

Muchas veces, la formulación analítica de un problema de optimización es difícil de implementar en la práctica. No siempre es posible, o práctico, modelar matemáticamente la forma en que los valores de las variables de decisión afectan al comportamiento del sistema bajo estudio. Por dicho motivo, es conveniente en estos casos reemplazar la formulación analítica por una formulación algorítmica, en la cual los valores de la función objetivo y/o las restricciones se obtengan como resultado de ejecutar un algoritmo con la solución a evaluar como entrada. En el caso que dicho algoritmo sea una simulación, estamos ante lo que se conoce como OvS [Fu and Heally, 1997, Hong and Nelson, 2010, Sánchez et al., 2010, Nguyen et al., 2014, Fu, 2015, Xu et al., 2015, Bartz-Beielstein et al., 2018, Kang et al., 2018, Liu and Shi, 2019].

La OvS, también llamada *Simulated Optimization*, *Simulated-Driven Optimization* o *SimHeuristic* es un conjunto de metodologías de resolución de problemas de optimización en los cuales la evaluación de la función objetivo y de sus restricciones (o de parte de ellas) se lleva a cabo mediante la ejecución de una o mas simulaciones [Fu, 1994, Chen and Lee, 2011, Juan et al., 2015,

Amaran et al., 2016, Juan et al., 2018]. En la OvS, el problema representado por las Ecuaciones 2.1, 2.2 y 2.3 de la Sección 2.1 se reescriben como las Ecuaciones 3.1, 3.2 y 3.3 [Fu, 2015].

$$Min\ \vec{f}_{sim}(\vec{x}) \tag{3.1}$$

Sujeto a:

$$\vec{g}_{sim}(\vec{x}) \leq \vec{r} \tag{3.2}$$

$$\vec{x} \in \Theta \tag{3.3}$$

Donde $\vec{f}_{sim}(\vec{x})$ y $\vec{g}_{sim}(\vec{x})$ representan el resultado de evaluar la solución $\vec{x}$ mediante una o mas simulaciones.

En el enfoque de OvS, el algoritmo de optimización y el algoritmo de simulación actúan como dos cajas negras independientes (Figura 3.1). El algoritmo de optimización genera secuencias de soluciones a evaluar y recibe el resultado de dichas evaluaciones. El algoritmo de simulación recibe un conjunto de parámetros para configurar un modelo de simulación preexistente, ejecuta las simulaciones y devuelve las métricas asociadas a la performance del sistema simulado. Conectando los dos bloques, existe un proceso de *traducción* que convierte los valores de las variables de decisión en entradas para el simulador, y las salidas del simulador en valores de las funciones objetivo.

Al igual que para resolver un modelo analítico es necesario disponer del modelo en forma previa a la ejecución del algoritmo, aquí es necesario disponer en forma previa del modelo de simulación. Adicionalmente, han de definirse cuales de sus variables de entradas actuarán como *parámetros* (valores fijos para todas las simulaciones) y cuales como entradas propiamente dichas para el simulador (valores que se obtienen del optimizador). Asimismo, se necesitan codificar los procesos de traducción.

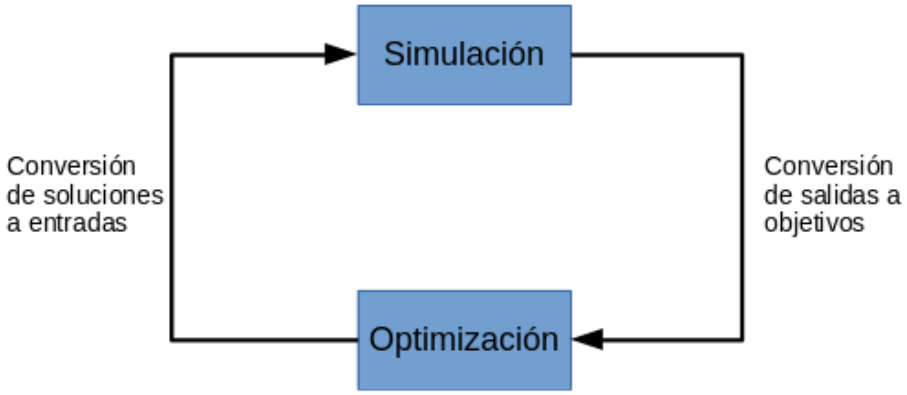


Figura 3.1: Esquema General de OvS. Los bloques de optimización y simulación funcionan como dos cajas negras.

La OvS ha encontrado un gran campo de aplicación en problemas estocásticos y/o difíciles de modelar analíticamente [Chen and Lee, 2011, Schmaranzer et al., 2019]. En los primeros, proporciona una forma práctica de tratar con la aleatoriedad y el ruido, los cuales muchas veces exigen realizar demasiadas simplificaciones en el modelado analítico. En los segundos, permite construir modelos lógicos mas cercanos al problema real a resolver, sin necesidad del uso de artefactos algebraicos. Estos dos tipos de dificultades marcaron la evolución del desarrollo de la OvS. Por un lado, los investigadores provenientes del mundo de la simulación se enfocaron en desarrollar métodos de búsqueda que aseguraran la convergencia estadística y soluciones robustas (haciendo foco en el uso de test de hipótesis, por ejemplo). Por el otro, los investigadores provenientes del mundo de la optimización se focalizaron en la adaptación de heurísticas y metaheurísticas a problemas complejos pero determinísticos, tratando de asegurar convergencia al óptimo global [Fu, 2002, Juan et al., 2015].

3.1.1. OvS Determinística

La OvS Determinística consiste en el uso de simulaciones para evaluar funciones, o bien determinísticas, o bien con un nivel de aleatoriedad despreciable. El problema a resolver se puede expresar completamente mediante las Ecuaciones 3.1, 3.2 y 3.3. En muchos casos, se pueden usar metaheurísticas convencionales para su resolución. El principal foco de interés es asegurar (o al menos, guiar el proceso hacia) la convergencia al óptimo global, evitando los óptimos

locales. Dado que la razón del uso de las simulaciones es tratar problema complejos, es común que estos últimos sean problemas de optimización multiobjetivo.

3.1.2. OvS Estocástica

En la OvS Estocástica, las funciones a evaluar son aleatorias, por lo tanto una única evaluación no es suficiente [Tian et al., 2018]. Por cada solución $\vec{x}$ a evaluar, se deben realizar múltiples simulaciones, obteniendo al final uno o mas indicadores estadísticos del rendimiento del sistema, usualmente, la esperanza. En este caso, $\vec{f}$ y $\vec{g}$ se calculan de acuerdo a las Ecuaciones 3.4 y 3.5:

$$\vec{f} \simeq \vec{E}(\vec{f}^*_{sim}(\vec{x})) \tag{3.4}$$

$$\vec{g} \simeq \vec{E}(\vec{g}^*_{sim}(\vec{x})) \leq \vec{r} \tag{3.5}$$

Donde E es la esperanza matemática y $\vec{f}^*_{sim}$ y $\vec{g}^*_{sim}$ son los resultados de evaluar (es decir, simular) las funciones f y g en un escenario individual perteneciente al diseño experimental DS [Kleijnen, 2015], donde cada escenario se codifica mediante un conjunto fijo de parámetros. Para obtener una buena estimación de la esperanza (o del indicador usado)[1], la cantidad de escenarios en el diseño experimental debe ser grande. Dado $N = |DS|$ la cantidad de escenarios en DS, la media aritmética muestral $\vec{J}_{DS}$ es un buen estimador de la esperanza E cuando $N \longrightarrow \infty$ (Ecuación 3.6, en la cual $\vec{h}^*$ puede ser indistintamente $\vec{f}^*$ o $\vec{g}^*$).

$$\vec{J}_{DS}(\vec{x}) = \frac{1}{N} \sum_{i=1}^{N} \vec{h}^*_{sim_i}(\vec{x}) \tag{3.6}$$

La esperanza suele ser utilizada en mediciones de rendimiento, es decir, en la evaluación de $\vec{f}$. Sin embargo, $\vec{g}$ admite otro criterio para ser evaluada, éste es, la esperanza de la proporción de cumplimiento de las restricciones. La Ecuación 3.7 muestra este enfoque. IF es una función

[1]Se hace incapié en que el indicador puede ser cualquier estadístico, pero en la mayoría de la bibliografía se selecciona la esperanza como indicador. En muchos casos de aplicación práctica, podría ser conveniente seleccionar un indicador de posición basado en cuartiles o percentiles.

vectorial que devuelve un vector booleano, en el cual cada componente vale 1 si la restricción asociada se satisface, y 0 en caso contrario.

$$\vec{J}_{DS}(\vec{x}) = \frac{1}{N}\sum_{i=1}^{N} IF(\vec{g}^{*}_{sim_i}(\vec{x}) \leq \vec{r}) \tag{3.7}$$

Cabe aclarar que, si bien no está representado en las ecuaciones anteriores, $\vec{r}$ puede ser tanto un vector aleatorio como determinístico.

A diferencia del caso determinístico, en la OvS Estocástica el foco está puesto en que el diseño experimental sea representativo, y que la convergencia al óptimo global sea estadísticamente significativa. Representatividad significa aquí que, si una solución A se elige por sobre otra solución B, A sea en promedio mejor que B en cualquier situación posible. Esta representatividad se suele asegurar mediante el uso de test de hipótesis.

3.1.3. Costo Computacional de la OvS

Pese a sus ventajas en el tratamiento de problemas estocásticos y/o complejos, la OvS tiene un punto negativo importante, su alto costo computacional[2] [Sacks et al., 1989, Kleijnen, 2009, Cristescu and Knowles, 2015, Díaz-Manríquez et al., 2016, Davins-Valldaura et al., 2017]. En general, ejecutar una simulación requiere de más recursos que evaluar una formulación analítica, lo cual sumado al hecho que se deben simular muchas soluciones y, en el caso estocástico, en múltiples escenarios, vuelve prohibitivo el uso de esta herramienta cuando se requiere una respuesta rápida.

El costo computacional de la OvS proviene de dos fuentes principales: la cantidad de simulaciones a realizar y el costo de cada simulación individual. Existen variaciones al proceso básico de OvS que ayudan a disminuir la cantidad de simulaciones necesarias, ya sea mediante alguna variedad de muestreo adaptativo (es decir, realizar menos simulaciones en las primeras iteraciones

[2]Cuando se habla de costo computacional de la OvS, no nos referimos a la clase de complejidad computacional de los problemas a resolver, sino al hecho que la evaluación de cada solución generada por el optimizador requiere de mas tiempo respecto a que si fuera una función analítica a calcular. Independientemente de si el problema es no-polinómico o no, de si la metaheurística utilizada requiere muchas o pocas evaluaciones, cada evaluación individual requiere, en general, de mayor esfuerzo que el cálculo del valor de una función analíticamente definida.

del algoritmo y más en las últimas, utilizando un modelo simplificado en vez de las simulaciones), o bien mediante el uso de otras herramientas que predigan el resultado de la simulación [Barton and State, 2010, Bittner and Hahn, 2013, Dellino and Meloni, 2015, Juan et al., 2015, Kleijnen, 2015, Amaran et al., 2016]. Existe también toda una rama de investigación dedicada a determinar como construir el diseño experimental (cantidad de escenarios y características de cada escenario) de manera tal de obtener el mejor resultado posible con el hardware disponible [Chen and Lee, 2011]. Sin embargo, no existe tanto desarrollo en cuanto a la reducción del costo en si de cada simulación individual [Banks, 1998, Fu, 2015].

3.2. Clasificación de problemas

En base al tipo de espacio de soluciones de un problema abordado mediante OvS, podemos clasificar a los mismos dentro de tres categorías [Fu, 1994]):

- Problemas de rankeo y selección (Ranking and Selection, RandB).
- Problemas de OvS Continua (Continue OvS, COvS).
- Problemas de OvS Discreta (Discrete Ovs, DOvS).

En el primer caso, el de los problemas de RandB, el espacio de soluciones es finito y su tamaño es *pequeño* [3]. Debido a ésto, es posible explorar todas las soluciones del espacio. Frente a un problema determinístico, el proceso completo de OvS consiste en evaluar todas las soluciones, ordenarlas según el valor de la función objetivo y seleccionar la mejor. En principio, bastaría con un algoritmo de *Fuerza Bruta* en el bloque optimizador. Pero, ante un problema estocástico, la situación es mas compleja. Teniendo fijo el número de soluciones a evaluar, es necesario determinar cuantas evaluaciones y en que escenarios son necesarias para asegurar con una probabilidad mayor a un α predefinido que la solución detectada por el algoritmo como óptima es realmente la óptima. Es decir, el foco está puesto en como asegurar, con el menor número de simulaciones posible, que la solución seleccionada es estadísticamente mejor que el resto.

[3] Pequeño depende de la potencia de cómputo disponible

Por otro lado, en los otros dos tipos de problemas, el espacio de soluciones es, o bien infinito (COvS y DOvS), o bien finito pero demasiado grande como para evaluar todas las soluciones potenciales (DOvS). En ambos casos, no se pueden evaluar todas las soluciones, por lo cual es necesario realizar un muestreo del espacio de soluciones. El foco de los algoritmos de resolución de este tipo de problemas está puesto en realizar un buen muestreo del espacio de soluciones, tratando que un entorno del óptimo global sea parte de dicho muestreo. Para el caso puntual que el espacio de soluciones sea finito, siendo $|X| < \infty$ el tamaño del espacio, n el total de las soluciones muestreadas y P la probabilidad que el óptimo global esté dentro de la muestra se cumple el límite enunciado en la Ecuación 3.8.

$$\lim_{n \longrightarrow |X|} P = 1 \tag{3.8}$$

Es decir, la probabilidad de muestrear el óptimo global converge a 1 a medida que el volumen de soluciones exploradas aumenta. Para el caso de los espacios de soluciones continuos, no hay garantías que la probabilidad P converja a 1 en el caso general [Fu, 2015].

3.3. Algoritmos

Como se mencionó previamente, la OvS evolucionó en dos grandes ramas: la adaptación de metaheurísticas y el desarrollo de métodos basados en muestreo estadístico.

Para el caso de OvS determinística, el empleo de metaheurísticas ha probado dar muy buenos resultados. De hecho, su aplicación no difiere del uso convencional de estos algoritmos [Kang et al., 2018], estando el foco puesto en la reducción de la cantidad de simulaciones necesarias [Carson and Maria, 1997, Sánchez et al., 2010], por ejemplo, usando modelos analíticos simplificados para delimitar el espacio de soluciones. Para el caso estocástico, existen tres variaciones comunes en el uso de metaheurísticas [Juan et al., 2015]:

- Ignorar la aleatoriedad y optimizar el problema suponiéndolo determinístico, para luego simular la solución seleccionada como óptima en DS a fin de evaluar su robustez.

- Tomar en cuenta la aleatoriedad, evaluando cada solución en todo DS para luego optimizar la esperanza E.
- Tomar en cuenta la aleatoriedad como en el caso anterior, pero incorporando el conocimiento de dicha aleatoriedad para la decisión de que fracción de DS simular y hacia donde explorar el espacio de soluciones.

En el caso de los algoritmos nacidos en el seno de la comunidad de simulación, éstos se basan en dos categorías principales [Fu, 2015, Kleijnen, 2015]:

- Modelado y optimización de Superficies de Respuesta (para COvS)
- Búsqueda Aleatoria Guiada (para COvS yDOvS)

Ambas metodologías se enfocan en la robustez estadística. Puntualmente, las diversas alternativas de *Búsqueda Aleatoria Guiada* son versiones del algoritmo de *Descenso por Gradiente* combinadas con el algoritmo de *Búsqueda Aleatoria* tradicional, en los cuales se da particular importancia a la cantidad de simulaciones necesarias para determinar la optimalidad local de las soluciones.

En esta tesis el foco está puesto en la utilización de metaheurísticas dentro del bloque de optimización.

3.4. Metamodelos y otras técnicas de predicción

A fin de lograr implementaciones prácticas de algoritmos de OvS, es común el uso de metamodelos y técnicas de predicción para reducir el costo computacional. En ambos casos nos referimos a modelos que, en base a un conjunto de entradas y salidas del simulador, son capaces de inferir, con un grado aceptable de precisión, que salidas tendrán otro conjunto diferente de entradas. Es decir, no se construye un modelo a priori con conocimiento del sistema a simular, sino que se estima un modelo de caja negra del tipo estímulo-respuesta en base a datos históricos [Sacks et al., 1989, Barton and State, 2010, Cristescu and Knowles, 2015, Kleijnen, 2015,

Kleijnen, 2019]. Pese a que existen algunas diferencias, en esta tesis se utilizan los términos *metamodelos* y *técnicas de predicción* a modo de sinónimos.

Existen varias herramientas que se utilizan para estimar las salidas del simulador, entre las cuales encontramos:

- Regresiones
- Máquinas de vectores soporte
- Redes neuronales artificiales
- Modelos estadísticos del tipo *Kriging*

De los nombrados anteriormente, las tres primeras corresponden a la categoría de algoritmos de *Machine Learning* (Aprendizaje Automático), mientras que la última es una herramienta adaptada de la geoestadística [Krige, 1951, Matheron, 1963, Sacks et al., 1989] perteneciente al grupo de algoritmos de interpolación.

Los metamodelos pueden ser utilizados en dos formas: previo al proceso de OvS o dentro del proceso de OvS. En el primer caso, antes de iniciar la optimización, se genera un conjunto de soluciones y se ejecutan las mismas en el simulador, a fin de obtener las respectivas salidas. Luego, con estas tuplas *entrada-salida*, se construye el metamodelo, el cual posteriormente será utilizado en el proceso de OvS en lugar del bloque de simulación (ya sea en todas o en algunas de las iteraciones). En el segundo caso, no se construye nada en forma previa. Se inicia el proceso de OvS ordinario y, a medida que se van generando tuplas *entrada-salida*, se construyen metamodelos parciales. Estos se usan en iteraciones posteriores en reemplazo del bloque de simulación, pero cuando la calidad del metamodelo decae demasiado, se vuelve a utilizar el bloque de simulación y a recolectar tuplas *entrada-salida* para mejorar la calidad del metamodelo actual.

Las herramientas del tipo Machine Learning se suelen utilizar previo al proceso de OvS, ya que suelen requerir muchas muestras para su entrenamiento. Al requerir muchas muestras, incluirlas dentro del proceso de OvS genera metamodelos inestables, muy sensibles al paso de las iteraciones. Los modelos tipo Kriging, sin embargo, funcionan bien con pocos da-

tos[4], por lo cual se suelen usar dentro del proceso de OvS, siendo recalculados muchas veces a lo largo de dicho proceso [Voutchkov and Keane, 2006, Li et al., 2008, Kleijnen, 2009, Ankenman et al., 2010, Bittner and Hahn, 2013]. Existen varios algoritmos de OvS que utilizan metamodelos tipo Kriging. Por ejemplo, en [Bittner and Hahn, 2013] se propone combinar el uso de metamodelos Kriging con una versión multiobjetivo del algoritmo de Optimización por Enjambre de Partículas. En [Todoroki and Sekishiro, 2008] se utilizan metamodelos para evaluar los objetivos en un Algoritmo Genético Multiobjetivo (Multi Objective Genetic Algorithm, MOGA). Y en [Li et al., 2008] se desarrolla el algoritmo K-MOGA, el cual usa una versión modificada del NSGA combinada con Kriging y utiliza el calculo de la varianza del metamodelo para determinar cuando es necesario re-calcular el mismo.

3.4.1. Metamodelos tipo Kriging

Los metamodelos del tipo Kriging fueron desarrollados por Krige y Matheron [Krige, 1951, Matheron, 1963] para modelar problemas del campo de la geoestadística y para el modelado de datos espaciales y fueron implementados como herramienta para modelar simulaciones por primera vez en [Sacks et al., 1989]. En este tipo de metamodelos, la función a modelar (idealmente, una simulación) es representada como un proceso estocástico gaussiano de la forma presentada en la Ecuación 3.9

$$Y(x) = \sum_{j=1}^{k} \beta_j \cdot f_j(x) + Z(x) \tag{3.9}$$

En la cual x es la entrada recibida por el simulador, Y la salida generada por el mismo, f es una función de ajuste y $Z(x)$ es un proceso gaussiano con media cero cuya covarianza para dos elementos diferentes (por ejemplo, x y w) viene definida por la Ecuación 3.10.

$$V(w, x) = \sigma^2 \cdot R(w, x) \tag{3.10}$$

Donde σ^2 es la varianza del proceso aleatorio Z y $R(w, x)$ la matriz de correlación de dicho

[4]Pocos datos es una descripción relativa respecto de la cantidad de datos necesitados por un algoritmo de Machine Learning.

proceso (cuadrada y de rango n_0). Hay muchas alternativas para elegir la matriz R, pero una selección habitual es la mostrada en el Ecuación 3.11 [Li et al., 2008].

$$R(w,x) = exp[-\sum_{n=1}^{N} \Theta|w_n - x_n|^2] \tag{3.11}$$

Con Θ un vector de parámetros Θ_n a calcular. La idea principal tras el modelado mediante la Ecuación 3.9 es que el proceso original se puede interpretar como un camino aleatorio alrededor de una regresión de sus entradas. El objetivo es entonces encontrar los β, f_j y $Z(x)$ mas apropiados a partir de un conjunto de datos de entrenamiento. A diferencia del método de mínimos cuadrados, no se busca minimizar el error de estimación en el conjunto de entrenamiento, sino hacer que este error sea cero para cualquier punto del conjunto de entrenamiento [Kleijnen, 2009, Kleijnen, 2015]. Asumiendo las funciones f como constantes [Sacks et al., 1989, Li et al., 2008], se puede estimar la respuesta y para cualquier entrada x^* mediante la Ecuación 3.12, donde $f = f_c$ es un vector con todos sus componentes iguales a 1 y r el vector de correlaciones del elemento x^* con cada elemento del conjunto de entrenamiento (Ecuación 3.13).

$$\hat{y} = \beta + r^t(x^*)R^{-1}(y - f_c\beta) \tag{3.12}$$

$$r = [R(x^*, x_1), ..., R(x^*, x_n)] \tag{3.13}$$

Dado que R solo depende del conjunto de observaciones x y del parámetro desconocido Θ, β puede ser expresado en base a ellos mediante la Ecuación 3.14, siendo el mejor estimador de Θ_n el expresado en la Ecuación 3.15.

$$\beta = (f_c^t R^{-1} f_c)^{-1} f_c^t R^{-1} y \tag{3.14}$$

$$\vartheta_n = max \frac{-[n_0 \ln(\sigma^2) + \ln(R)]}{2} \tag{3.15}$$

Estimar Θ en forma exacta (y R en consecuencia) suele ser complejo, por este motivo se utilizan metaheurísticas para realizar dicha estimación [Kleijnen, 2015].

La principal característica del metamodelo resultante es que el mismo es un estimador exacto para los valores de x usados en el entrenamiento, mientras que para valores internos al hipervolumen definido por las x más extremas, se comporta como un interpolador mediante *splines*. Los *splines* son curvas diferenciables definidas por polinomios que se utilizan para interpolar y suavizar curvas. En este caso, Kriging trata de suavizar las variaciones respecto a los puntos de entrenamiento manteniendo su capacidad de representar parte de la no linealidad de los datos. Esta interpolación funciona bastante bien a menos que, entre los puntos de entrenamiento, existan puntos de cambio de pendiente muy pronunciados. Por fuera del contorno usado para la estimación, la calidad de las predicciones por extrapolación desciende fuertemente.

Capítulo 4

Sistemas y Problemas de Tráfico

En este capítulo se describen los conceptos de sistemas de tráfico, simulaciones de tráfico y los problemas asociados abordados en esta tesis doctoral. Además, se presenta al simulador SUMO utilizado para obtener la evaluación de las soluciones proporcionadas por los algoritmos desarrollados en algunos de los problemas aquí planteados.

4.1. Sistemas de tráfico

Un *sistema de tráfico* es un conjunto de personas, vehículos, vías de tráfico, señalizaciones y políticas de circulación que interactúan entre sí en una determinada área geográfica y cuyo objetivo último es el permitir el movimiento ordenado de grupos de personas y bienes dentro de dicha área.

El movimiento de las personas y/o bienes dentro de un área urbana o hacia/desde éstas ocupa un porcentaje significativo de la actividad total realizada en dichas áreas geográficas. Si bien la necesidad de moverse de una localización a otra existió siempre, recién durante el Siglo XX, con la invención del automóvil, la mejora en las condiciones de vida, el desarrollo de grandes urbes y el acelerado crecimiento poblacional, los sistemas de tráfico se volvieron un componente esencial de la vida diaria, a la vez que generaron problemas desconocidos hasta entonces (congestionamientos, accidentes de tránsito, polución vehicular, etc) [Patriksson, 1994, Haberman, 1998, TRB, 2016].

Los sistemas de tráfico son sistemas complejos, que no se pueden entender analizando cada componente en forma individual. El comportamiento de los mismos, además de ser autorregulados, presenta características emergentes, como los embotellamientos espontáneos [Kerner, 2009].

El estudio de los sistemas de tráfico está orientado a tomar decisiones que permitan reducir los congestionamientos, el tiempo medio necesario para llegar de una localización a otra, la cantidad de accidentes, el volumen de polución y contaminación sonora, entre otras [Haberman, 1998]. Para ello, el ente administrador del sistema de tráfico puede operar en tres niveles diferentes: *el estratégico*, en el cual se opera a largo plazo, con predicciones de crecimiento poblacional y realizando intervenciones a nivel de infraestructura; *el táctico*, en el cual, con una infraestructura fija, se opera sobre variables como la sincronización de semáforos, políticas de estacionamiento, etc [Kerner, 2009]; y *el operativo*, en el cual se atienden contingencias puntuales en un horizonte de tiempo muy corto, mediante acciones como el re-direccionamiento del tráfico, por ejemplo [Treiber and Kesting, 2013].

4.2. Modelado de sistemas de tráfico

Modelar un sistema de tráfico implica modelar, en forma parcialmente separada, cada uno de sus componentes, así como también sus interrelaciones. De acuerdo al objetivo que persiga el modelado podemos optar por la herramienta a utilizar, pero en forma general un sistema de tráfico se modela mediante la siguiente división: la *demanda de tráfico* (las necesidades de movimientos), la *oferta de tráfico* (la infraestructura y las políticas) y el *comportamiento del sistema* (el resultado de la interacción entre oferta y demanda). En esta tesis, el comportamiento del sistema se modeló mediante simulación, como se verá en la Sección 4.3.

4.2.1. Modelado de la infraestructura

La oferta de tráfico está compuesta por la red de tráfico en si misma (o sea, el conjunto interconectado de vías de circulación), y por el conjunto de políticas que rigen el comportamiento de los vehículos que circulan por dicha red. Estas políticas pueden tener una realidad física o

no, y se pueden clasificar en el espectro pasivas-activas: por ejemplo, el límite de velocidad es una política pasiva (una normativa cuyo cumplimiento depende de cada conductor), un cruce semaforizado es una política más activa que el límite (ya que si bien no bloquea el tráfico, indica cuando debería frenarse), y una cruce a nivel con barrera es más activa que el cruce semaforizado (implica un bloqueo físico). De los tres, los dos últimos poseen realidad física.

A efectos de modelado, la red de tráfico se representa mediante un grafo orientado $D = (N, E)$, en el cual cada arco orientado del conjunto E representa una vía de circulación y cada nodo de N un punto de cambio en la red (como por ejemplo, una intersección, una ampliación de carriles en una avenida, el inicio y fin de un paso subterráneo, etc). No hay límites a la cantidad de arcos que pueden conectar en forma directa dos nodos $a, b \in N$ en el sentido $a \rightarrow b$, ya que cada arco esta asociado a un carril de una arteria de uno o mas carriles. Por otro lado, los arcos del tipo $a \rightarrow a$ son inválidos (un nodo no se puede conectar con si mismo). En otras palabras, D es un multigrafo dirigido sin bucles. Tanto los arcos como los nodos tienen anotaciones asociadas que modelan, por ejemplo, la velocidad máxima de una vía, o la secuencia de activación de los semáforos. Cada anotación se representa mediante una tupla $T = (id,\ t,\ o,\ z_1,\ z_2,\ ..,\ z_n)$, donde id es una referencia al arco o nodo asociado, t el tipo de anotación, o el *offset* o desplazamiento desde el inicio del arco y las z_x las propiedades a codificar. Por ejemplo, un límite de velocidad de $60km/h$ se puede representar mediante la tupla ($arc01$, $limitevelocidad$, 0, 60), donde $id = arc01$ es la referencia a una vía de circulación, $t = limiteVelocidad$ explicita el tipo de anotación, $o = 0$ indica que el límite rige desde el inicio de la vía y $z_1 = 60$ es el valor del límite de velocidad. La representación mediante grafos se suele denominar en la bibliografía *Arc-Node View* y el proceso de agregar anotaciones *Linear Referencing System* (LRS) [Alvarado et al., 2010, Behrisch et al., 2011, Farahani et al., 2013, Garcia-Nieto et al., 2013, Elefteriadou, 2014]. Cada accidente particular de la red exige un conjunto de anotaciones diferentes, las cuales deben ser interpretadas a la hora de modelar el comportamiento del sistema.

4.2.2. Modelado de la demanda

La demanda de tráfico es el conjunto de todos los vehículos y peatones que circulan por el sistema, junto con las rutas seguidas por cada uno y sus tiempos de partida. Por razones tanto de

recolección de datos, como también debido al carácter no constante de la misma, la representación de la demanda se basa en modelado estadístico. Dentro del conjunto de modelos existentes, hay tres muy usados para modelar la demanda [Behrisch et al., 2011, Elefteriadou, 2014]:

- Matrices de Origen-Destino (*OD Matrix*)
- Conjuntos de flujos de tráfico (*Traffic Flows*)
- Conjuntos de viajes (*Traffic Trips*)

Las matrices de Origen-Destino son matrices que codifican la cantidad de vehículos promedio que se mueven desde un nodo a otro. No proporcionan información de tiempos ni de rutas asociadas al movimiento entre nodos. Suelen ser útiles para el modelado a largo plazo y/o analítico del sistema de tráfico.

Los conjunto de flujos de tráfico van un paso mas allá. Son grupos de tuplas $T = (id, volumen, tipo, tpoInicio, tpoFin, nodoInicio, nodoFin)$ en el que cada tupla representa a un conjunto de vehículos (de tamaño igual al *volumen* indicado) que viajan desde un nodo a otro, en una ventana de tiempo determinada. No contienen información relativas a rutas.

Los conjuntos de viajes extienden la definición de conjunto de flujos para indicar la ruta seguida por los vehículos del grupo. Se definen mediante una tupla $T = (id, volumen, tipo, tpoInicio, tpoFin, nodoInicio, nodoFin, ruta)$ donde ruta es una tupla de longitud variable $T_r = (nodo_1, nodo_2, ..., nodo_n)$ que indica la secuencia ordenada de nodos N por las cuales circula el vehículo para moverse desde *nodoInicio* a *nodoFin*.

Para modelar una demanda de tráfico existente, se deben utilizar técnicas de muestreo estadístico para relevar los datos. En forma muy simplificada, el relevamiento consiste en definir puntos de muestreo en la red y, durante un periodo determinado, medir las siguientes variables:

- Cantidad de vehículos circulando
- Velocidad media de cada vehículo
- Tipo de vehículo

En base a estos datos, se puede construir la matriz de Origen-Destino del conjunto de nodos relevados o realizar posteriores análisis para realiza una estimación de los flujos de tráfico.

4.2.3. Modelado del comportamiento del sistema

Usualmente, el modelado del comportamiento del sistema se suele realizar en forma analítica. Si bien en esta tesis no se hace uso de este tipo de modelado, sino que se recurre a simulación, se hará a continuación una breve reseña.

El modelado analítico consiste en definir un conjunto de ecuaciones que representen un aspecto del comportamiento total del sistema, generalmente en forma estática, tomando como base parámetros de la demanda o de la oferta. Por ejemplo, para calcular el flujo de saturación [1] de una intersección semaforizada, se puede utilizar la Ecuación 4.1 (extraída de [TRB, 2016]).

$$s = s_0 \cdot N \cdot f_a \cdot f_{PPp} \cdot f_e \cdot f_o \cdot f_u \cdot f_t \cdot f_{gd} \cdot f_{gi} \cdot f_{gdp} \cdot f_{gip} \tag{4.1}$$

Donde s es el flujo de saturación, s_0 el flujo de saturación base (una constante dependiente del volumen de población), N es el número de carriles, f_a es el factor de ajuste por ancho de carril, f_{PPp} el factor de ajuste por vehículos pesados y pendiente, f_e el factor de ajuste por carril de estacionamiento y cantidad de maniobras de estacionamiento, f_o es el factor de ajuste por parada de ómnibus urbanos, f_u es el factor de utilización medio de los carriles, f_t es el factor de ajuste por tipo de zona, f_{gd} y f_{gi} los factores de ajuste por giro a la derecha e izquierda, respectivamente,y f_{gdp} y f_{gip} los factores de ajuste por presencia de peatones en giro a la derecha e izquierda, respectivamente. Cada uno de los factores de ajuste se calcula mediante tablas y relevamientos en campo, muestreando ventanas de 15 minutos. En general el análisis se hace en el día del año que represente el percentil 80 del flujo de saturación, a modo de asegurar capacidad para casi todo el año pero sin sobredimensionar el sistema debido a situaciones atípicas.

Se puede ver que el modelado analítico posee varios puntos negativos. Primero, es estático, no contempla la dinámica temporal del tráfico. Segundo, es local, por ejemplo aquí se analiza

[1] Cantidad máxima de vehículos que pueden circular por unidad de tiempo

solo una intersección, sin vinculación con la intersección siguiente. Tercero, es determinístico. Cuarto, es muy específico, este modelo sirve para analizar un solo aspecto del sistema de tráfico, no pudiendo extrapolarse a otras situaciones (por ejemplo, intersecciones sin semaforizar).

Existen otras alternativas de modelado [Woensel and Vandaele, 2007] que solucionan uno o mas puntos, pero en general ninguna soluciona el problema de la localidad y/o el problema de la especificidad.

4.3. Simulaciones de tráfico

A diferencia de los modelos analíticos, las simulaciones de tráfico permiten representar el sistema de tráfico completo (en forma global y sin atender a un aspecto específico), tomando en cuenta el comportamiento dinámico del mismo y/o su aleatoriedad. Si bien es difícil su construcción, permite su reutilización para el tratamiento de múltiples problemas e, incluso, la evaluación de condiciones no existentes en el sistema con un alto grado de confianza. Existen 4 clases principales de modelos de simulación utilizados para estos sistemas: macroscópicos, mesoscópicos, microscópicos y submicroscópicos (Figura 4.1).

En una simulación macroscópica se modela el flujo vehicular en forma agregada, como si de un fluido o un gran conjunto de partículas se tratara. La dinámica vehicular se modela mediante ecuaciones diferenciales o ecuaciones en diferencias, determinísticas o estocásticas, las cuales representan el comportamiento de las variables de estado del sistema en función del tiempo. Estas simulaciones, si bien no aportan información acerca de la dinámica individual, y no modelan bien discontinuidades en el tráfico, son útiles para representar el flujo vehicular en grandes extensiones lineales como, por ejemplo carreteras.

Las simulaciones mesoscópicas trabajan en una escala menor, modelando el sistema de tráfico como un sistema se eventos discretos basado en teoría de colas, en el cual los vehículos son considerados como entidades transitorias que requieren servicio. Si bien representan cada vehículo

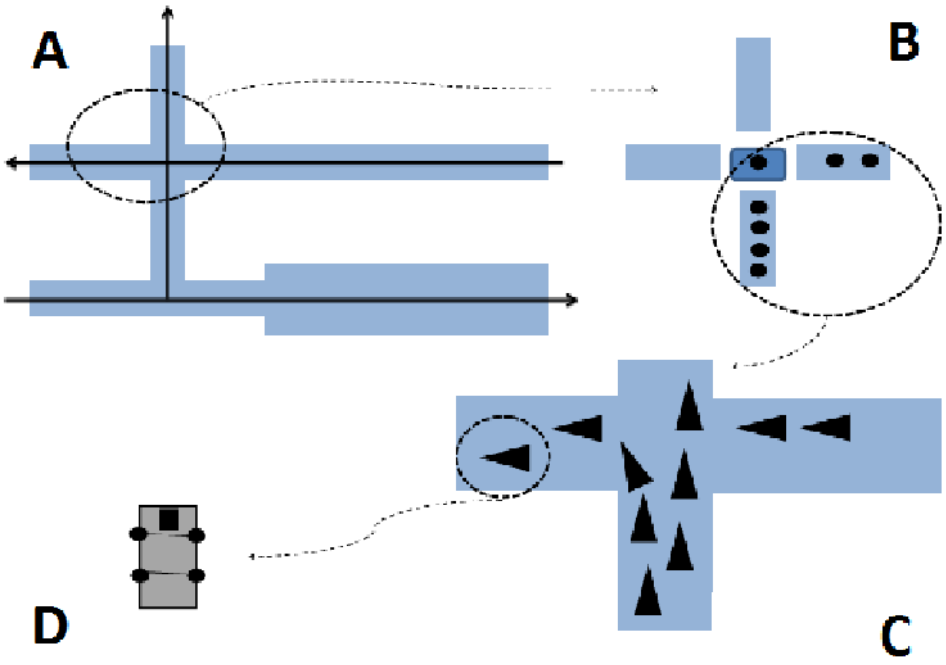


Figura 4.1: Tipos de simulaciones de tráfico: macroscópica (A), mesoscópica (B), microscópica (C) y sub-microscópica (D).

individualmente, la interacción entre los mismos se deriva de las políticas globales del sistema de tráfico. Estas simulaciones son útiles para modelar accidentes puntuales del sistema de tráfico como, por ejemplo, intersecciones semaforizadas y accesos a rutas. Al trabajar a escala de vehículos, permite modelar embotellamientos no espontáneos.

Las simulaciones microscópicas [Behrisch et al., 2011] modelan el sistema al nivel de cada vehículo y sus interacciones, sin definir pautas de comportamiento global. Cada vehículo es simulado individualmente y, puestos todos los vehículos en una red de tráfico, el comportamiento global emerge como consecuencia de la interacción de éstos entre si. Este tipo de simulación permite modelar en forma satisfactoria el tráfico en entornos urbanos, donde existen multitud de condiciones que anulan los supuestos de continuidad de flujo vehicular [2]. También permiten representar embotellamientos espontáneos [3]. La principal ventaja que ofrecen este tipo de simulaciones es que el modelado de todos sus elementos es comparativamente mas fácil que en el resto de las simulaciones, sin necesidad de hacer suposiciones sobre el comportamiento global del sistema. Como contrapartida, consumen mas tiempo de cómputo que los tipos de

[2]Los supuestos teóricos que permiten tratar el flujo de tráfico como un fluido continuo.

[3]Embotellamientos temporales debidos a diferencias de velocidad entre vehículos, causados porque la velocidad de circulación de cada vehículo individual no es constante, sino que presenta pequeñas fluctuaciones aleatorias que no pueden ser absorbidas por el sistema.

simulaciones previamente enunciados.

El último tipo de simulación es el tipo submicroscópico. Éstas modelan los vehículos al nivel de componentes. Es decir, se modelan en forma individual, para cada vehículo, el sistema de dirección, frenado, etc. Son útiles para evaluar condiciones de operación de vehículos individuales, pero no para conjuntos de vehículos.

En esta tesis nos enfocamos mayoritariamente en el uso de modelos de simulación microscópicos para el estudio del tráfico urbano.

4.3.1. Simulaciones de tráfico microscópicas

Una simulación de tráfico microscópica es un tipo de simulación de tiempo discreto [4] en la cual el comportamiento de cada vehículo es modelado y simulado individualmente. Cada vehículo en el sistema de tráfico está caracterizado por un código identificador (id), un conjunto de parámetros constantes que regulan la ruta a seguir, el momento en cual inicia su viaje, la velocidad máxima y el resto de los criterios de circulación del vehículo, y un conjunto de variables con información acerca de la velocidad y posición del vehículo (las cuales son actualizadas en cada intervalo t). El comportamiento de cada vehículo sigue un conjunto de leyes simples, representadas en una ecuación del tipo estímulo-respuesta, en el cual el principal factor que determina el comportamiento de cada vehículo es el vehículo situado inmediatamente enfrente. Este tipo de lógicas son conocidas como *Car-Following* [Li and Sun, 2012] [5]. Un ejemplo de lógica para un vehículo i es la siguiente:

- Si no hay ningún auto adelante (desde la posición del $vehiculo_i$ hasta una distancia crítica), acelerar hasta alcanzar la velocidad máxima.
- Si hay algún auto adelante y su velocidad es menor que la de $vehiculo_i$, desacelerar para igualar la velocidad de dicho vehiculo.

[4] El tiempo está modelado en forma de intervalos discretos de tamaño fijo, avanzando el reloj desde un intervalo a otro mediante saltos. Con un tamaño de intervalo lo suficientemente pequeño, se pueden modelar satisfactoriamente sistemas de tiempo continuo.

[5] No es el único tipo de lógica, pero suele ser la mas usada, ya sea en forma pura o mediante variaciones.

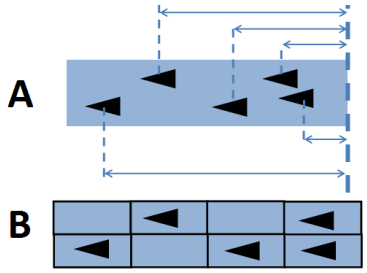


Figura 4.2: Modelado de espacio continuo (A) y de espacio discreto (B).

- Si hay algún auto adelante y su velocidad es mayor o igual que la de $vehiculo_i$, acelerar hasta igualar su velocidad o llegar a la velocidad máxima del $vehiculo_i$, lo que sea menor.

Además de las interacciones entre vehículos, también se agregan interacciones con el resto de los elementos del sistema, como semáforos por ejemplo.

A grandes rasgos, existen dos tipos de simulaciones microscópicas: de espacio discreto y de espacio continuo (ambas de tiempo discreto). Un esquema del concepto básico tras las mismas se muestra en la Figura 4.2. En las de espacio discreto, la red de tráfico es discretizada en un conjunto de celdas, cuyos estados determinan si la celda está ocupada por un vehículo o no [Nagel and Schreckenberg, 1992]. El funcionamiento es similar al de un autómata celular (siguiendo las reglas del tipo *Car-Following*). Un red de tráfico consistente en una única vía (la cual se representa mediante un grafo camino) se modela con un autómata celular unidimensional (Figura 4.3). Una red de tráfico mas compleja, requiere un autómata bidimensional (Figura 4.4). Esto también requiere ampliar la definición de estado para incluir áreas no transitables.

En un modelo de espacio continuo, por el contrario, el espacio no está discretizado. En vez de utilizar autómatas celulares, se simula cada vehículo en forma directa. A su vez, cada vehículo lleva asociado las coordenadas del mismo (el arco actual y la distancia al inicio de dicho arco) a fin de poder aplicar la estrategia *Car-Following*.

En ambos casos, basándonos en la información de cada vehículo, es posible obtener luego variables agregadas del sistema, como la velocidad promedio, el tiempo de espera máximo, etc [Krauss, 1998, Chowdhury et al., 2000].

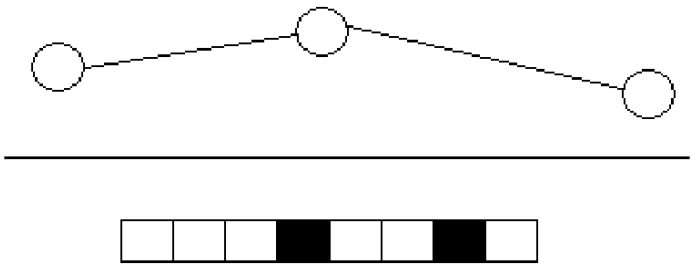


Figura 4.3: Modelado de espacio discreto de una red lineal. Las celdas negras corresponden a vehículos en la red.

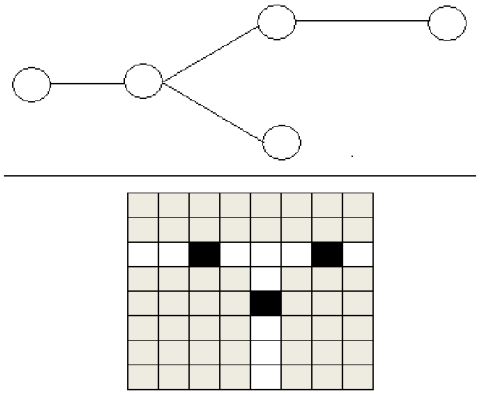


Figura 4.4: Modelado de espacio discreto de una red no-lineal. Las celdas negras corresponden a vehículos en la red.

4.3.2. SUMO: Simulation of Urban MObility

SUMO [Behrisch et al., 2011] en la herramienta elegida en esta tesis para simular la mayoría de los modelos de tráfico tratados [6]. Éste es un simulador microscópico de espacio continuo diseñado para la simulación de tráfico urbano. Permite modelar gran variedad de vehículos (incluyendo autobuses), peatones, semáforos (de programación fija e *inteligentes*) entre otras cosas. Mas aún, permite incluir *sensores* en la red de tráfico para poder medir cantidad de vehículos, velocidad media, contaminación (ambiental y sonora), etc. Es open-source, altamente portable y capaz de utilizar mapas completos de ciudades reales (obtenidos de *Google Maps*® u *Open Street Maps*).

[6] Para la resolución del SSBRP se ha utilizado un simulador mesoscópico construido *Ad-Hoc*.

SUMO provee una serie de aplicaciones adicionales para realizar tareas de pre y pos procesamiento de datos relativos a las simulaciones (como *DUAROUTER*, que construye conjuntos de viajes a partir de conjuntos de flujos o matrices Origen-Destino, o *NETGENERATE*, para crear redes de tráfico ficticias).

RSUMO: Interfaz en R para SUMO

Con el fin de facilitar la interacción de SUMO con los algoritmos propuestos en esta tesis, así como el análisis de los resultados de lo mismos, se desarrolló la biblioteca RSUMO para el lenguaje de programación *R* [Team, 2018]. RSUMO permite crear y modificar redes de tráfico, definir vehículos y rutas de circulación, ejecutar simulaciones y leer los resultados generados, entre otras tareas, usando la sintaxis estándar de R (puntualmente, está basada en el modelo de objetos *S4*). Este paquete se puede descargar desde `https://github.com/egbaquela/RSumo`.

RSUMO permite acceder, desde *R*, a muchas de las funciones de SUMO. El simulador debe estar previamente instalado en la computadora. Desde el paquete desarrollado se puede acceder a algunas de las aplicaciones provistas por SUMO, por ejemplo, para crear archivos de redes de tráfico o archivos de flujo vehicular. El simulador puede ser ejecutado desde *R* (de hecho, si hay múltiples versiones instaladas, se puede seleccionar cual utilizar) y muchos de los datos de salida pueden ser leidos también desde *R*, en formato *Data Frame*[7].

4.4. Problemas de optimización en sistemas de tráfico

Como se mencionó en la Sección 4.1, los problemas de tráfico se pueden agrupar en tres categorías: estratégicos, tácticos y operativos. Durante la investigación que concluyó con la elaboración de esta tesis, se utilizó el formalismo de OvS para tratar problemas correspondientes a los tres ámbitos. A nivel estratégico, se trató el problema del rediseño parcial de la red de tráfico para absorber la mayor demanda generada por el crecimiento poblacional. A niveles táctico y operativo, se trató el diseño de políticas de inversión de carriles de tráfico y resincronización

[7] A los efectos prácticos de analizar resultados, es mas útil disponer de una o mas tablas (como los *Data Frames* de *R*) en vez de un archivo *XML*.

de semáforos para mejorar la circulación vehicular, y a nivel puramente operativo, el ruteo de autobuses hacia/desde un único destino/origen.

4.4.1. Asignación de Orígenes y Destinos de Tráfico

El problema de Asignación de Orígenes y Destinos de Tráfico (ODTAP) es un problema de rediseño de la red de tráfico [Jia et al., 2019] cuyo objetivo es el de absorber el crecimiento proyectado de la demanda.

El crecimiento de una ciudad refleja una dinámica progresiva que pocas veces se planifica, dando lugar a una disminución de la performance general de los subsistemas que la integran [Xiao et al., 2006]. En el caso del subsistema de tráfico, un incremento poblacional implica más vehículos en el sistema, posibles congestionamientos, mayores tiempos medios de viaje, etc. [Li et al., 2010, He et al., 2011]. Si bien esto es difícil de controlar, es posible establecer políticas relativas a permisos de urbanización, creación de centros industriales y fomento de actividades en zonas estratégicas, que permitan direccionar la forma de crecimiento de la ciudad y su subsistema de tráfico asociado [Ferreira et al., 2010, Ferreira and Condessa, 2012, Haregeweyn et al., 2012].

Durante el desarrollo de esta tesis, se investigó el problema de decidir en dónde fomentar la instalación de complejos urbanos, industriales y/o comerciales de manera tal que el efecto sobre el sistema de tráfico sea el menor posible, considerando que la red de tráfico ya cuenta con una estructura y una dinámica de flujo que restringen las configuraciones posibles. A este problema de decisión planteado se lo llama *Repotenciación de la red de tráfico*, siendo el ODTAP el problema de optimización subyacente. El ODTAP es un problema de la familia de los problemas de diseño de redes de tráfico (NTDP). Este último es un problema *NP-Hard* en el cual se quiere construir una red de tráfico de manera tal de minimizar una función de desempeño del sistema, en general, el tiempo medio de viaje [Yang and Bell, 1998, Waller et al., 2006, Kepaptsoglou and Karlaftis, 2009, Farahani et al., 2013]. En este contexto, existen numerosos trabajos en la literatura relativos al NTDP, algunos de ellos relacionados con la problemática asociada al ODTAP y la planificación urbana [Council, 1995, Gilat, 2002,

Hanson and Giuliano, 2004, Ratner and Goetz, 2013]. Respecto de su resolución, las metaheurísticas son ampliamente usadas para tratar el NTDP, pero en general partiendo de un esquema no basado en el uso de simulaciones, sino en modelado analítico [Friesz et al., 1992, Fredik, 2010, Dinu and Bordea, 2011, Gallo et al., 2012]. Un caso de aplicación de OvS al NTDP lo encontramos en [Horvat and Tosic, 2012], en donde se utiliza un enfoque basado en algoritmos genéticos para la reestructuración de la red.

Definición del ODTAP

En el ODTAP se parte de la convención que el flujo de vehículos es generado y consumido solo en algunos nodos de la red. Esto significa que, en una ventana de tiempo dada, algunos de los nodos serán *orígenes* de tráfico, otros serán *destinos* y el mayor grupo se corresponderán a nodos de *transición* (es decir, nodos intermedios por los cuales los vehículos deben transitar para viajar desde su origen a su destino). No necesariamente se debe tomar en cuenta la totalidad de los nodos de la red, sino que muchos de los nodos y arcos se pueden agrupar para formar una descripción agregada de la red (por ejemplo, un barrio localizado en los suburbios puede ser modelado, a efectos prácticos, como un único nodo). A esta redefinición de la red la llamaremos la *estructura simplificada de la red* ($ESRT$).

Dada la estructura simplificada de la red actual ($ESRT_{actual}$) es posible ampliarla para incluir los nodos y arcos que no existen en la red actual, pero que pueden ser creados para absorber la variación de la demanda. A esta red la denominaremos estructura simplificada de la red potencial ($ESRT_{potencial}$). Sobre la $ESRT_{potencial}$ se puede definir la matriz Origen-Destino actual (OD_{actual}), en la cual los nodos que existen en la red potencial y no en la actual se representan mediante vectores columnas y filas nulos. El objetivo del ODTAP, entonces, es definir una nueva matriz Origen-Destino (OD_{opt}) y una nueva estructura de red ($ESRT_{opt}$) que minimice tanto los tiempos de viaje en la red como el costo de las reformas necesarias. Estamos entonces ante un problema de optimización combinatoria multiobjetivo. Cabe notar que $ESRT_{actual} \subseteq ESRT_{opt} \subseteq ESRT_{potencial}$.

Se propone una formulación de programación no-lineal entera para este problema. La Tabla 4.1 resume los parámetros del modelo y las variables de decisión.

Tabla 4.1: Parámetros y variables del modelo de programación no-lineal entera del ODTAP

Parámetro	Definición
N	Conjunto de nodos de la $ESRT_{potencial}$.
$N_{or} \subset N$	Conjunto de nodos de origen.
$N_{de} \subset N$	Conjunto de nodos de destino.
P	Conjunto de rutas dentro de la $ESRT_{potencial}$.
$i \in N_{or}$	Nodo de origen.
$j \in N_{de}$	Nodo de destino.
$p \in P$	Ruta.
$Pop_{i,j,p}$	Cantidad de vehículos que actualmente viajan desde i a j por p.
cap_i	Capacidad máxima del nodo i.
cap_j	Capacidad máxima del nodo j.
c_i	Costo de infraestructura unitaria en nodo i.
c_j	Costo de infraestructura unitaria en nodo j.
$incrementoDemanda$	Cantidad de vehículos adicionales a asignar.

Variable	Definición
$X_{i,j,p}$	Cantidad de nuevos vehículos asignados a viajes desde i a j por p.
W_i	Cantidad de nuevos vehículos asignados a viajes desde el origen i.
Y_j	Cantidad de nuevos vehículos asignados a viajes hacia el destino j.

Las tres variables de decisión son variables enteras no negativas. Es necesario definir una también una función T que dependa de los valores de todas las $X_{i,j,p}$ y los $Pop_{i,j,p}$, y que devuelva el tiempo medio de viaje en el sistema. T es una función no lineal, y su definición depende de las características de la $ESRT_{potencial}$ y de las políticas que regulan el comportamiento de los vehículos en el sistema [8] [TRB, 2016].

[8] T es, a priori, una función difícil de definir analíticamente en forma general, lo cual la hace adecuada para ser evaluada mediante simulación.

La formulación del problema es la siguiente [9]:

$$\text{mín } Z_1 = T \tag{4.2}$$

$$\text{mín } Z_2 = \sum_{i \in N_{or}} c_i \cdot W_i + \sum_{j \in N_{de}} c_j \cdot Y_j \tag{4.3}$$

Sujeto a:

$$W_i + \sum_{j \in N_{de}} \sum_{p \in P} Pop_{i,j,p} \leq cap_i, \forall i \in N_{or} \tag{4.4}$$

$$Y_j + \sum_{i \in N_{or}} \sum_{p \in P} Pop_{j,j,p} \leq cap_j, \forall j \in N_{de} \tag{4.5}$$

$$\sum_{i \in N_{or}} W_i = \sum_{j \in N_{de}} Y_j = incrementoDemanda \tag{4.6}$$

$$W_i = \sum_{j \in N_{de}} \sum_{p \in P} X_{ijp}, \forall i \in N_{or} \tag{4.7}$$

$$Y_j = \sum_{i \in N_{or}} \sum_{p \in P} X_{ijp}, \forall j \in N_{de} \tag{4.8}$$

La Ecuación 4.2 representa el tiempo de viaje total en el sistema, mientras que la Ecuación 4.3 representa el costo de transformar la $ESRT_{actual}$ en $ESRT_{opt}$. Las Ecuaciones 4.4 y 4.5 restringen la cantidad máxima de nueva demanda a asignar a cada nodo de origen y destino, respectivamente. La Ecuación 4.6 fuerza a que toda la demanda adicional sea asignada. Y las Ecuaciones 4.7 y 4.8 balancean las salidas de nodos orígenes y arribos a nodos destino con la cantidad de vehículos asignados a cada viaje.

[9] Este modelo podría ser ampliado incluyendo variables binarias en caso que se requiera incluir costos fijos, en caso que el uso de un nodo no existente en la red actual implique asignarle una capacidad mínima, o que dos nodos no puedan ser seleccionados para asignar demanda simultáneamente, por ejemplo.

4.4.2. Inversión de carriles de tráfico y resincronización de semáforos

La inversión temporal del sentido de circulación en carriles de calles o avenidas ha probado ser un método útil para resolver problemas de congestionamientos puntuales en sistemas de tráfico. Se ha aplicado con mucha efectividad a los congestionamientos causados por lo que se conoce como *Olas de Tráfico* [Zhao et al., 2014]. Estas son variaciones temporales en el flujo vehicular que se propagan a modo de ondas por parte del sistema de tráfico [Kamalanathsharma, 2011]. Suelen ser generadas por la dinámica del tráfico durante las horas pico o punta, durante algún evento masivo (como un recital) o bien por alguna emergencia (una evacuación, por ejemplo). Muchos estudios analizan su efectividad y aplicabilidad [Bede et al., 2010, Lambert and Wolshon, 2010, Cheng et al., 2011].

En términos de problemas de optimización, la inversión de carriles de tráfico es un problema multiobjetivo de optimización combinatoria en el cual se quieren seleccionar un conjunto de carriles a invertir de manera tal de cumplir con, al menos, los siguientes dos objetivos:

- Minimizar los tiempos medios de viaje en el sistema en la ventana de tiempo estudiada.
- Minimizar la cantidad de carriles invertidos.

El segundo objetivo es necesario por una cuestión práctica: es difícil coordinar los cambios temporales del sentido de circulación y existe el riesgo de causar accidentes de tráfico con ellos, debido al desconocimiento o desorientación de los conductores. Se puede observar que los dos objetivos están en conflicto: si bien la relación no es lineal, mientras menos carriles de tráfico se inviertan, menor será la reducción en los tiempos de viaje.

En la literatura referente al tema se suele tratar el problema mediante un modelo de programación matemática lineal o no lineal, en el cual se minimiza los tiempos medios de viaje en forma local, restringido al área en la cual se genera la ola de tráfico, sin tomar en cuenta los efectos globales sobre el sistema ni los efectos dinámicos del mismo (como los congestionamientos espontáneos debido a diferencias de velocidad relativas). La cantidad de cambios en el sentido de

circulación se suele acotar mediante una restricción, convirtiendo el problema de multiobjetivo a monoobjetivo [Bede and Péter, 2011, Hausknecht et al., 2011, Qing and Ziyou, 2014] [10]. Otros autores, como [Karoonsoontawong, 2010] usan un enfoque basado en OvS para el cómputo de los tiempos medios de viaje. Sin embargo, se restringe el problema al área afectada por la ola de tráfico sin contemplar un análisis global del impacto.

El aplicar una política de inversión de carriles para paliar los efectos de situaciones puntuales de congestionamiento podría llegar a ser poco eficiente si no se complementa con una resincronización temporal de los semáforos del sistema y no se consideran los efectos colaterales de dicha inversión (por ejemplo, costos asociados para asegurarse que todos los conductores conocen el nuevo patrón de circulación). La mayoría de los enfoques que buscan integrar inversión de carriles de tráfico y secuenciación de semáforos lo hacen en forma secuencial: primero optimizan la dirección de las vías de circulación y, sobre esa solución, buscan la mejor configuración de semáforos. Este enfoque tiende a generar soluciones subóptimas. En ese sentido, [Zhao et al., 2014] analiza la bibliografía existente y propone un modelo PLE multiobjetivo que integra la selección de carriles a invertir con la programación de los semáforos en el mismo paso, a fin de optimizar el sistema en una forma mas integral. En dicho modelo, la función objetivo consiste en la suma ponderada de sus objetivos y la formulación es totalmente determinística y estática, siendo los resultados evaluados a posteriori en un simulador de tráfico.

La investigación de este tema realizada en esta tesis consistió en el desarrollo de métodos multiobjetivos para calcular la Frontera de Pareto del sistema, tomando en cuenta la no-linealidad y aleatoriedad del sistema y el impacto global de los cambios realizados, no solo en el área cercana al epicentro de la ola de tráfico.

4.4.3. Ruteo de vehículos de transporte publico a destino único

Dentro de los problemas del tipo operativo relativos a sistema de tráfico se pueden incluir los problemas de ruteo de vehículos. Estos consisten en, partiendo de una demanda de transporte prefijada, definir las rutas del conjunto de vehículos disponibles para minimizar el costo total

[10] Esto es otra forma de modelado analítico que adolece de los mismos problemas que la forma vista en la Sección 4.2.3.

de los viajes. En esta tesis el foco ha estado puesto en problemas en el cual las personas a transportar deben alcanzar un único destino (el mismo para todas) por medio de autobuses, el cual se denomina *Problema de ruteo de autobuses escolares a una única escuela* (SSBRP).

El SSBRP [Newton and Thomas, 1969, Russell and Morrel, 1986, Bowerman et al., 1995] es una variedad del *Problema de ruteo de autobuses escolares* (SBRP), el cual a su vez es un subtipo del *Problema de ruteo de vehículos* (VRP). El VRP es un problema general en el cual, dada una flota de vehículos y un conjunto de clientes a visitar, se quieren asignar paradas a vehículos y definir las rutas de estos últimos a fin de minimizar el costo total de la operación. Es una generalización del conocido *Problema del viajante de comercio* (TSP) para el caso de disponer de mas de un vehículo [Laporte, 1988, Díaz-Parra et al., 2014]. El SBRP, por otro lado, consiste en asignar estudiantes a autobuses para transportarlos a sus respectivas escuelas. Las tres diferencias principales con el VRP son que los nodos de fin de cada ruta no pueden ser elegidos libremente (tienen que ser escuelas), que los nodos que representan escuelas pueden ser visitados por mas de un autobús y que, además, las paradas de los buses no están prefijadas, sino que deben ser seleccionadas a fin de reducir la distancia de los estudiantes desde sus hogares a dichas paradas [Park and Kim, 2010, Park and Kim, 2013].

Hay mucha investigación al respecto del SBRP [Corberán et al., 2002, Schittekat et al., 2006]. Modelos de programación lineal y no lineal, heurísticas y metaheurísticas son utilizados para resolver distintas variantes de este problema [Nayati, 2008, Prasetyo et al., 2011, de Lima, 2015, Calvete et al., 2016]. Si bien en base al caso particular analizado pueden tener objetivos diferentes, en general dos objetivos están siempre presentes:

- Minimizar el costo del transporte.
- Minimizar la distancia recorrida por los estudiantes desde sus hogares hasta las paradas.

Tenemos aquí dos objetivos en conflicto, ya que por ejemplo la solución de menor costo (no transportar a ningún estudiante) implica maximizar la distancia recorrida por los estudiantes (todos debería caminar desde sus hogares hasta su escuela). La manera de gestionar estos

objetivos en la mayor parte de la literatura consiste en ponderar los objetivos en una única función. Pero esto añade un grado de subjetividad muy alto, ya que es difícil encontrar una escala común para medir el costo económico del conjunto de rutas armadas y el esfuerzo de los estudiantes para llegar hasta las paradas.

Un caso particular que resulta interesante es aquel en el cual todos los estudiantes deban ser transportados a la misma escuela. Este problema es el SSBRP [Schittekat et al., 2013]. Si bien está planteado en términos de estudiantes y escuelas, puede ser aplicado, por ejemplo, al caso de transportar un gran volumen de trabajadores a un centro industrial (o a cualquier otra localización en la cual el empleador esté radicado).

Formulación matemática del SSBRP

El SSBRP puede ser modelado como un problema de programación lineal binaria multiobjetivo. El siguiente modelo está basado en el modelo propuesto por [Schittekat et al., 2013]:

$$\text{mín } Z_1 = \sum_{i \in V} \sum_{j \in V} c_{ij} \cdot \sum_{k=1}^{n} x_{ijk} \tag{4.9}$$

$$\text{mín } Z_2 = \sum_{l \in S} \sum_{i \in V} d_{il} \cdot s_{il} \tag{4.10}$$

Sujeto a:

$$\sum_{j \in V} x_{ijk} = \sum_{j \in V} x_{jik} = y_{ik}, \quad \forall i \in V, k = 1, ..., n \tag{4.11}$$

$$\sum_{i,j \in Q} x_{ijk} \leq |Q| - 1, \quad \forall Q \subseteq V \setminus \{v_0\}, \forall k \tag{4.12}$$

$$\sum_{k=1}^{n} y_{ik} \leq 1, \quad \forall i \in V \setminus \{0\} \tag{4.13}$$

$$\sum_{k=1}^{n} z_{ilk} \leq s_{il}, \quad \forall l \in S, \forall i \in V \tag{4.14}$$

$$\sum_{i \in V} \sum_{l \in S} z_{ilk} \leq C, \quad k = 1, ..., n \tag{4.15}$$

$$z_{ilk} \leq y_{ik}, \quad \forall i, l, k \tag{4.16}$$

$$\sum_{i \in V} \sum_{k=1}^{n} z_{ilk} = 1, \quad \forall l \in S \tag{4.17}$$

$$y_{ik} \in \{0, 1\}, \quad \forall i \in V, k = 1, ..., n \tag{4.18}$$

$$x_{ijk} \in \{0, 1\}, \quad \forall i, j \in V, i \neq j, k = 1, ..., n \tag{4.19}$$

$$z_{ilk} \in \{0, 1\}, \quad \forall i \in V, \forall l \in S, k = 1, ..., n \tag{4.20}$$

Donde x_{ijk} toma el valor 1 si el autobús k viaja desde i hacia j, y_{ik} vale 1 si el autobús k visita la parada i, y z_{ilk} es igual a 1 si el estudiante l se sube al autobús k en la parada i, s_{il} vale 1 si el estudiante l espera el autobús en la parada i, c_{ij} es la distancia que un autobús debe recorrer para ir desde la parada i a la j, y d_{il} es la distancia que el estudiante l debe caminar desde su hogar hasta la parada i.

La Ecuación 4.9 minimiza la distancia total recorrida por los autobuses. La Ecuación 4.10 minimiza la distancia total caminada por los estudiantes hasta las paradas. Las ecuaciones 4.11 y 4.12 se utilizan para eliminar subtours en las rutas de autobuses. La Ecuación 4.13 fuerza a visitar cada parada solo una vez, o no visitarla nunca. La Ecuación 4.14 maneja la asignación de estudiantes a paradas. La Ecuación 4.15 controla que no se exceda la capacidad de los autobuses. Las Ecuación 4.16 se asegura que el estudiante l no sea levantado en la parada i por el autobús k si dicho autobús no visita la parada i. La Ecuación 4.17 fuerza a que todos los estudiantes sean transportados a la escuela una sola vez. Por último, las Ecuaciones 4.18, 4.19 y 4.20 restringen a todas las variables a tomar dos posibles valores, 0 o 1.

4.4.4. SSBRP con localización de estudiantes aleatoria

Un caso particular del SSBRP es cuando la distribución espacial de estudiantes, así como su número, son estocásticos. Esta situación sucede, por ejemplo, en el caso que el destino único sea un lugar de trabajo (un centro industrial por ejemplo), exista una rotación significativa de empleados (como en la industria de la construcción) y no se disponga de la suficiente flexibilidad para cambiar las rutas de los vehículos de acuerdo a dicha rotación (por ejemplo, cuando las mismas están determinadas por contratos anuales o de mas largo plazo). Al igual que el SSBRP convencional, se debe asegurar que el 100 % de los estudiantes (o trabajadores) sean levantados por un ómnibus y llevados a destino. La formulación anterior se mantiene pero, en este caso, tanto d_{il} como $|S|$ son variables aleatorias. Y la asignación de estudiantes a paradas ya no es una variable de decisión. Además, dada la aleatoriedad en las posiciones iniciales de los pasajeros, se debe permitir que mas de un autobús pueda pasar por una misma parada.

Capítulo 5

Propuestas algorítmicas para resolver problemas de tráfico urbano

En este capítulo se exponen los algoritmos basados en OvS desarrollados para la solución de los problemas descriptos en la Sección 4.4. La motivación para la utilización de OvS se basa en que existe evidencia en la literatura que el uso de formalismos de optimización que incluyan simulación permite arribar a soluciones que, al implementarse, logran una mejor performance que las obtenidas con el modelado exclusivamente analítico [Fu, 2002, Li et al., 2008], siendo este tema poco explorado respecto de los sistemas de tráfico.

5.1. Implementación

Todos los algoritmos mostrados en esta Sección se programaron en *R* [Team, 2018], mientras que las simulaciones se ejecutaron o bien con SUMO [Behrisch et al., 2011] [1], o bien con un modelo de eventos discretos creado Ad-Hoc. Para la vinculación de R con SUMO se utilizó el paquete *RSUMO*[2], desarrollado durante esta tesis. Para las metaheurísticas, se usó el paquete *nsga2R* [Tsou, 2013] para ejecutar el algoritmo NSGA-II, junto a funciones del paquete *mco* [Mersmann, 2014].

[1] Ver Sección 4.3.2.
[2] Ver Sección 4.3.2

5.2. Evaluación del tiempo medio de viaje con SUMO

Tanto la resolución del ODTAP como la del problema de inversión de carriles de tráfico mediante OvS implican utilizar un simulador para determinar el tiempo medio de viaje asociado a cada solución. En ambos problemas, una solución determina una nueva red de tráfico y una nueva distribución de la demanda, por lo cual es necesario generar éstas antes de proceder a la simulación. Para evaluar este objetivo, se utilizó SUMO.

En su modo de uso mas simple, SUMO requiere como entradas un archivo de red, uno de demanda y uno de configuración. Con las definiciones dadas en ellos, ejecuta la simulación y genera uno o mas archivos de resultados. Tanto las entradas como las salidas están codificadas en lenguaje *XML*. El archivo de red contiene toda la estructura física y lógica de la red de tráfico: vías de circulación, velocidades máximas, semáforos, etc. El archivo de demanda contiene, para cada vehículo en el sistema, la secuencia de nodos a visitar en su viaje desde el punto de partida hasta el de destino, junto con toda la información relativa a las características y conducta del vehículo, y los tiempos de partida. El archivo de configuración define la cantidad de tiempo a simular y otros parámetros propios de la simulación en sí. Adicionalmente, SUMO permite el uso de archivos auxiliares como entradas adicionales, los cuales permiten definir características del sistema en forma mas detallada (por ejemplo, políticas complejas de sincronización de semáforos).

5.2.1. Construcción del archivo de red

A fin de generar una nueva red, SUMO proporciona un programa ejecutable denominado *NETGENERATE*. Este programa puede ser usado para generar archivos de red en forma sencilla, definiendo sus componentes por separado y dejando al software la responsabilidad de vincular dichos elementos correctamente. En uno de sus modos de operación, acepta como entradas archivos XML de nodos (intersecciones y cambios en las vías de circulación) y otro de arcos (vías de circulación) y los combina para construir el grafo de la red. El archivo de nodos consiste en una lista de nodos, en el cual cada elemento de dicha lista contiene información acerca del identificador único del nodo y sus coordenadas. El archivo de arcos contiene la lista de todas

```
<nodes>
  <node id="dl" x="0" y="0" />
  <node id="dr" x="1200" y="0" />
  <node id="tl" x="0" y="190.99" />
  <node id="tr" x="1200" y="190.99" />
</nodes>
```

Código 5.1: Ejemplo de contenido del archivo de nodos.

```
<edges>
  <edge id="1" from="dl" to="dr" />
  <edge id="2" from="dr" to="tr" />
  <edge id="3" from="tr" to="tl" />
  <edge id="4" from="tl" to="dl" />
</edges>
```

Código 5.2: Ejemplo de contenido del archivo de arcos.

las vías que conectan dos nodos cualesquiera. Cada elemento de la lista indica el identificador único de la vía, los identificadores de los dos nodos que conecta e información adicional acerca de las características de la vía, como velocidad máxima, densidad máxima, etc. Los Códigos 5.1 y 5.2 muestran ejemplos del contenido de un archivo de nodos y del archivo de arcos asociado, respectivamente. Como la red de tráfico es un grafo orientado, el archivo de arcos define también el sentido de la vía (la circulación es desde el nodo indicado en *from* hacia el indicado en *to*). Dado lo anterior, para el caso de vías de circulación en los dos sentidos, se necesitan de dos arcos con sentidos opuestos.

El resultado de invocar NETGENERATE se muestra en el Código 5.3. NETGENERATE define parámetros por defecto si no son indicados en los archivos de nodos o arcos (por ejemplo, la velocidad en la vía), además de crear toda la estructura de datos necesaria por el simulador (por ejemplo, vías ficticias de entrada y salida a la red, conexiones en las intersecciones para regular los giros, etc).

Modelado de intersecciones semaforizadas

Con el objeto de modelar intersecciones semaforizadas con semáforos de periodo fijo, SUMO permite definir un archivo de configuración de semáforos para regular sus estados de tiempos

```
<net>
  <edge id=":dl_0" function="internal">
    <lane id=":dl_0_0" index="0" speed="13.90" length="3.42"/>
  </edge>
    ...
  <edge id="1" from="dl" to="dr" priority="2">
    <lane id="1_0" index="0" speed="13.90" length="1209.46"/>
  </edge>
  ...
  <junction id="dl" type="priority" x="500.00" y="0.00"
    incLanes="source_0_4_0" intLanes=":dl_0_0 :dl_1_0">
    <request index="0" response="10" foes="10" cont="0"/>
    <request index="1" response="00" foes="01" cont="0"/>
  </junction>
  ...
  <connection from="1" to="2" fromLane="0" toLane="0" via=":dr_0_0"
    dir="s" state="M"/>
  <connection from="2" to="3" fromLane="0" toLane="0" via=":tr_0_0"
    dir="s" state="M"/>
  ...
</net>
```

Código 5.3: Ejemplo de contenido del archivo de red.

de cambio.

Dado un cruce de vías de tránsito, todos los semáforos del cruce mantienen su estado coordinado, es decir, sus luces se activan complementariamente (cuando uno esta en verde los demás están en rojo, por ejemplo). Por lo tanto, el estado de una intersección se puede definir mediante un vector en el cual cada componente es el estado de cada semáforo. En una intersección con 4 semáforos, por ejemplo, un conjunto de cuatro estados factibles podría ser: $\{rrrv, rrvr, rvrr, vrrr\}$. En este ejemplo, la primera componente indica que el semáforo 4 está en verde y el resto en rojo, la segunda indica que el semáforo 3 está en verde y así sucesivamente. Un conjunto de semáforos permanece en el mismo estado durante un tiempo predeterminado y luego cambia al siguiente estado en forma secuencial, volviendo a comenzar el ciclo una vez alcanzado el último estado. Es importante notar que los estados están asociados al conjuntos de los semáforos de una intersección, no a semáforos individuales.

NETGENERATE produce por defecto intersecciones semaforizadas con un tiempo de verde predefinido. Pero SUMO acepta como entrada adicional un archivo auxiliar con la configu-

```
<additional>
  <tlLogic id="0" programID="my_program" offset="0" type="static">
    <phase duration="31" state="GGggrrrrGGggrrrr"/>
    <phase duration="5"  state="yyggrrrryyggrrrr"/>
    <phase duration="6"  state="rrGGrrrrrrGGrrrr"/>
    <phase duration="5"  state="rryyrrrrrryyrrrr"/>
    <phase duration="31" state="rrrrGGggrrrrGGgg"/>
    <phase duration="5"  state="rrrryyggrrrryygg"/>
    <phase duration="6"  state="rrrrrrGGrrrrrrGG"/>
    <phase duration="5"  state="rrrrrryyrrrrrryy"/>
  </tlLogic>
</additional>
```

Código 5.4: Ejemplo de contenido del archivo auxiliar con definición de semáforos.

ración de los semáforos, en el cual para cada intersección pueden ser definidos sus estados y tiempos de verde (Código 5.4). En este caso, SUMO no utiliza la configuración definida por NETGENERATE, sino la configuración indicada en el archivo auxiliar.

5.2.2. Construcción del archivo de demanda

Respecto de la demanda, SUMO provee una aplicación denominada DUAROUTER, la cual tomando un archivo que codifique o una matriz Origen-Destino o un conjunto de flujos, y la red de tráfico asociada, permite generar el archivo de demanda de tráfico que SUMO necesita para ejecutar la simulación. Tanto para el ODTAP como para el problema de inversión de carriles se optó por generar conjuntos de flujos (Código 5.5) a partir de matrices Origen-Destino y pasar éstos a DUAROUTER para generar rutas para todos los vehículos (Código 5.6). Una definición de flujo está conformada por un identificador, los tiempos de inicio y fin del flujo, la cantidad y el tipo de vehículos asociados al flujo y los nodos de origen y fin. El archivo de rutas contiene la secuencia de nodos a visitar para cada vehículo.

5.3. Asignación de Orígenes y Destinos de Tráfico

Para resolver el ODTAP se desarrolló un algoritmo de OvS basado en el acoplamiento de una metaheurística multiobjetivo con un simulador microscópico de tráfico. El algoritmo se codificó

```
<flows>
  <flow id="0" from="edge0" to="edge1" begin="0" end="3600"
    number="100" vType="1" />
  <flow id="2" from="edge0" to="edge2" begin="0" end="3600"
    number="100" vType="1" />
</flows>
```

Código 5.5: Ejemplo de contenido del archivo de flujos.

```
<routes>
  <vType id="type1" accel="0.8" decel="4.5" sigma="0.5"
    length="5" maxSpeed="70" />

  <vehicle id="0" type="type1" depart="0">
    <route edges="0 a1 a2 a3 1" />
  </vehicle>
  <vehicle id="1" type="type1" depart="2">
    <route edges="0 a1 a2 a3 1" />
  </vehicle>
</routes>
```

Código 5.6: Ejemplo de contenido del archivo de rutas.

usando el SMPSO como base. El simulador de tráfico usado fue SUMO.

La meta de este esquema es resolver la versión multiobjetivo del ODTAP, en la cual se buscan minimizar los tiempos medios de viaje y los costos de construcción de la nueva infraestructura ante un incremento previsto de la demanda de tráfico. El segundo objetivo se calcula analíticamente en base a la selección de nodos a repotenciar y al costo de cada repotenciación. El primero, en cambio, necesita del uso de simulaciones. Para ello, se requiere que el simulador genere un listado de los viajes de cada vehículo en el sistema, y luego calcular el indicador de tiempo medio de viaje (Ecuación 5.1).

$$AD = \frac{\sum_{v=1}^{V} tt_v}{V} \tag{5.1}$$

Donde tt_v es el tiempo de viaje del vehículo v y V la cantidad de vehículos en el sistema.

5.3.1. Codificación de las soluciones y de otros parámetros

El formalismo elegido para codificar las soluciones del ODTAP es el de un vector de componentes enteras. Cada componente está asociada a un nodo origen o a un nodo destino de la $ESRT_{potencial}$, y el valor que tome indica la variación en la demanda asociada a ese nodo. Usando el mismo formalismo pueden codificarse las restricciones de cada nodo individual al incremento y decremento de la demanda (volviendo el chequeo de estas restricciones una comparación de componentes).

El costo de incremento de infraestructura por unidad de demanda presenta las características de un costo escalonado. Es decir, el costo de infraestructura unitario es diferente según el rango de variación de la demanda. Por lo tanto, se requiere un formalismo matricial para representarlo, en el cual las filas representan a los nodos y las columnas a los escalones de costos.

5.3.2. Algoritmo general

Algoritmo 4 OvS ODTAP

```
1: procedure OVS-ODTAP(params)
2:     criterioParada ← falso
3:     poblacion ← generarPoblacionInicial
4:     while criterioParada = falso do
5:         evaluaciones ← {}
6:         for all solucion ∈ poblacion do
7:             generarRed(solucion)
8:             calcularRutas(solucion)
9:             tiempoMedio ← simular()
10:            costo ← calcularCosto(solucion)
11:            evaluaciones ← evaluaciones ∪ {(solucion, tiempoMedio, costo)}
12:        end for
13:        if checkCriterioParada(evalauciones) = verdadero then
14:            criterioParada ← verdadero
15:        else
16:            poblacion ← crearNuevaProblacion(evaluaciones)
17:        end if
18:    end while
19:    return evaluaciones
20: end procedure
```

El procedimiento general de optimización propuesto se muestra en el Algoritmo 4. El algoritmo

toma como entradas la $ESRT_{actual}$, la $ESRT_{potencial}$, la matriz OD_{actual}, el incremento de demanda, los parámetros de operación de la metaheurística seleccionada y del simulador de tráfico (como el tiempo a simular). La línea 2 inicializa la bandera que detiene el bucle principal, y la línea 3 inicializa la población inicial de soluciones a evaluar. Las líneas $4-18$ se corresponden con el bucle principal del algoritmo. Mientras no se satisfaga el criterio de parada seleccionado, se evalúa cada solución de la población (líneas $5-12$) y, de acuerdo a la metaheurística seleccionada, se crea una nueva población. El proceso de evaluación de cada solución esta indicado en las líneas $7-10$. El costo de la solución se calcula analíticamente en la línea 10, mientras que el proceso para determinar el tiempo medio de viaje en las líneas $7-9$. La salida a entregar por el algoritmo es un conjunto de soluciones que aproximan la Frontera de Pareto del problema analizado.

El proceso de cálculo del tiempo medio de viaje (líneas $7-9$) es un proceso que requiere varias tareas. La primera consiste en generar la nueva red de tráfico codificada por el vector solución. Dado que el vector representa la variación en la cantidad de vehículos emitidos o recibidos en los nodos del tipo origen o destino, puede darse el caso que un nodo de la $ESRT_{potencial}$ con demanda asociada nula necesite ser activado (es decir, la componente asociada del vector solución sea no nula). En esta situación, resulta necesario reconstruir la red de tráfico a utilizar por el simulador. Luego, el cambio de demanda implica reconstruir la matriz OD y reestimar las rutas seguidas por todos los vehículos en el sistema (debido la variación en la densidad de vehículos en las vías existentes y la posible aparición de nuevas vías). Con estas dos tareas realizadas, recién se puede ejecutar la simulación. Un esquema simplificado de este proceso se puede observar en la Figura 5.1.

5.3.3. Implementación de la evaluación con SUMO

De acuerdo a lo indicado en la Sección 5.3.2, la evaluación del tiempo medio de viaje de una solución implica reconstruir la red y la demanda de tráfico.

Para la creación de la red, partiendo de los archivos XML de nodos y arcos asociados a la

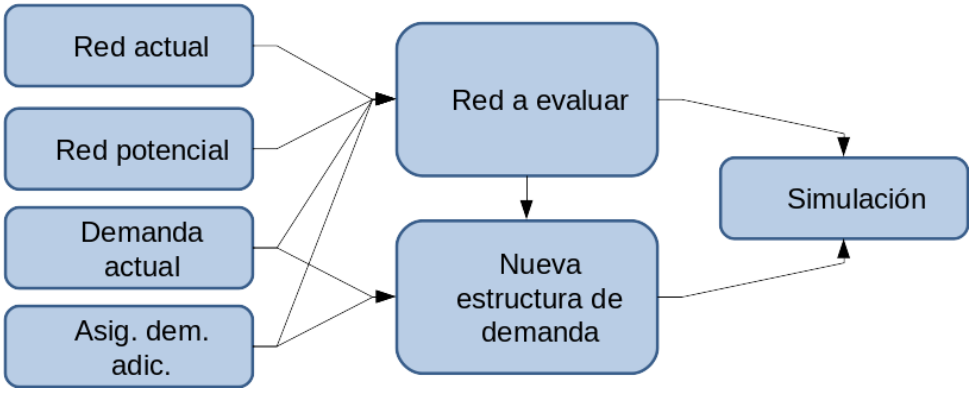


Figura 5.1: Obtención del tiempo medio de viaje en el ODTAP.

red de tráfico que representa a la $ESRT_{potencial}$ y los vectores que codifican la demanda en la red actual y la red a simular, se opera mediante sustracción para generar la red a simular. Es decir, se toma el vector de demandas de la $ESRT_{actual}$, se le suma el vector solución a evaluar, obteniendo un nuevo vector de demanda. Luego, se suprime de los archivos de nodos y arcos todas las referencias a los nodos con demanda igual a 0. Se llama a NETGENERATE y se obtiene la nueva red a simular.

Para la creación del archivo de demanda, el proceso seguido es aditivo. Tomando la matriz OD_{actual}, se suman las variaciones en forma proporcional a la distribución de destinos actual. En caso de nuevos destinos, el flujo se distribuye uniformemente. Por ejemplo, partiendo de la matriz OD de la Ecuación 5.2 (expresada en forma vectorial), el flujo entre dos nodos i y j se calcula como se indica en la Ecuación 5.3 sujeto a la restricciones de las Ecuaciones 5.4 y 5.5.

$$F = \{(o_1, d_1), \ldots, (o_1, d_n), \ldots, (o_n, d_1), \ldots, (o_n, d_n)\} \tag{5.2}$$

$$Vol(F_{ij}) = \frac{demanda(o_i) \cdot demanda(d_j)}{\sum_{i=1}^{n} o_i} \tag{5.3}$$

$$\sum_{i=1}^{n} o_i = \sum_{j=1}^{m} d_j \tag{5.4}$$

$$Vol(F_{ii}) = 0 \tag{5.5}$$

5.3.4. Implementación del proceso de optimización

A efectos de resolver el ODTAP, el Algoritmo 4 se implementó mediante un algoritmo SMPSO. Las líneas $5 - 12$ del algoritmo anterior codifican todo el proceso de evaluación de los dos objetivos para cada solución de la iteración en curso. La secuencia de tareas se codifican dentro del bloque $13 - 17$, de acuerdo a lo indicado en la Sección 2.4.2.

5.4. Inversión de carriles de tráfico - Sin sincronización de semáforos

Para resolver la versión multiobjetivo del problema de inversión de carriles de tráfico sin sincronización de semáforos se desarrolló un enfoque basado en OvS similar al utilizado para resolver el ODTAP (Sección 5.3): el acoplamiento de una metaheurística multiobjetivo con un simulador microscópico de tráfico. La metaheurística evaluada fue la SMPSO. El simulador de tráfico utilizado fue SUMO.

En ambos casos, las funciones objetivo a optimizar consisten en el tiempo medio de viaje en el sistema (Ecuación 5.6) y la cantidad de inversiones de carriles realizadas (Ecuación 5.7), ambos expresados en función de la configuración de carriles invertidos:

$$\text{máx } Z_1 = \frac{\sum_{i=1}^{m} Velocidad_i(P_{(x_1..x_n)})}{m} \tag{5.6}$$

$$\text{mín } Z_2 = \sum_{i=1}^{N} x_i \tag{5.7}$$

Donde x_i es una variable binaria que representa la decisión de invertir el carril i o no (1 =se invierte, 0 =no se invierte), Z_1 es la velocidad promedio en el sistema (siendo m la cantidad total de vehículos en el sistema y $Velocidad_i(P)$ la velocidad del vehículo i dado el patrón de circulación P creado por el conjunto de inversiones de carriles $\{x_1...x_n\}$) y Z_2 la cantidad de carriles invertidos, con un total de N carriles invertibles.

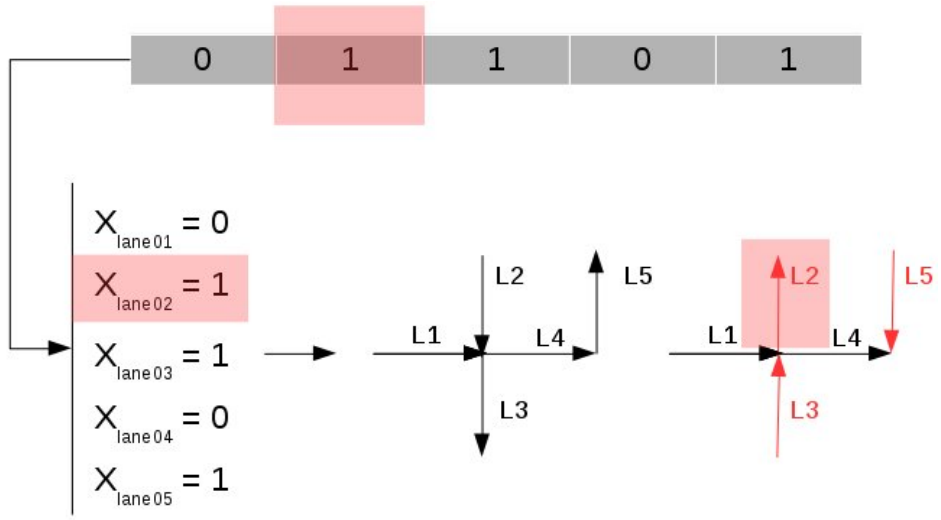


Figura 5.2: Mapeo entre el vector de soluciones, las variables de decisión y la inversión o no de carriles.

En esta sección se muestra primero la codificación, el algoritmo general y la implementación del caso sin sincronización de semáforos, y en la sección siguiente el mismo contenido para el caso con sincronización.

5.4.1. Codificación de las soluciones

Para codificar las soluciones, se utilizó un vector de tamaño n de componentes binarias, las cuales están asociadas en relación $1-1$ con las n variables x_i. Se utilizó el convenio del valor 1 para indicar una reversión del carril, y 0 para indicar un no-cambio de sentido. En la Figura 5.2 se observa un ejemplo de como el vector binario codifica las posibles soluciones. Dado la sección de red de tráfico original (abajo en el centro), se observa como la transformación a la nueva red (abajo a la derecha) se obtiene invirtiendo las vías asignadas al valor 1 en el vector solución.

5.4.2. Algoritmo general

El Algoritmo 5 muestra el procedimiento general de optimización. Éste funciona en forma similar al Algoritmo 4. Las diferencias se encuentran en las líneas 7, 8 y 10. Para cada solución

Algoritmo 5 OvS Inversión Carriles

```
1: procedure OVS-InversionCarriles(params)
2:     criterioParada ← falso
3:     poblacion ← generarPoblacionInicial
4:     while criterioParada = falso do
5:         evaluaciones ← {}
6:         for all solucion ∈ poblacion do
7:             generarRed(solucion)
8:             calcularRutas(solucion)
9:             tiempoMedio ← simular()
10:            inversiones ← calcularInveriones(solucion)
11:            evaluaciones ← evaluaciones ∪ {(solucion, tiempoMedio, inversiones)}
12:        end for
13:        if checkCriterioParada(evalauciones) = verdadero then
14:            criterioParada ← verdadero
15:        else
16:            poblacion ← crearNuevaProblacion(evaluaciones)
17:        end if
18:    end while
19:    return evaluaciones
20: end procedure
```

a evaluar es necesario reconstruir la red de tráfico, pero aquí el proceso de reconstrucción no genera ninguna modificación en los nodos, consiste solamente en la inversión del sentido de los arcos seleccionados. El cálculo de rutas también es más sencillo, ya que no hay que calcular el efecto de la demanda adicional, solo la distribución de las rutas dada la nueva estructura de la red. Y el costo de infraestructura es reemplazado en este algoritmo por la cantidad de carriles a invertir, el cual se obtiene sumando las componentes del vector solución. Un esquema del proceso de simulación de una solución concreta se puedo observar en la Figura 5.3.

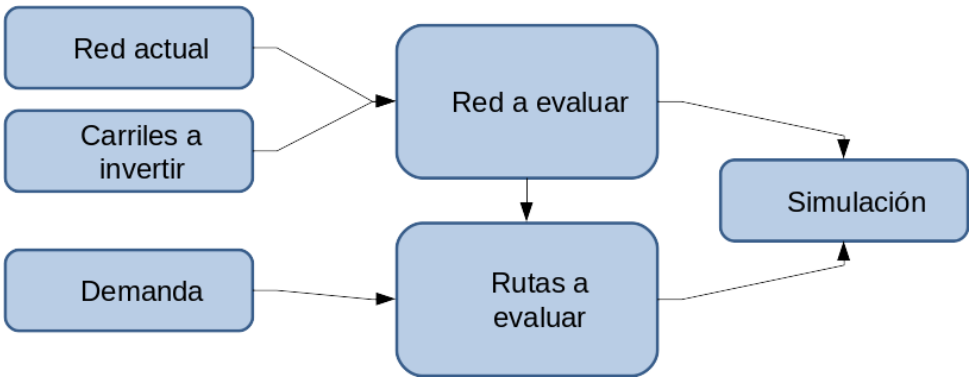


Figura 5.3: Obtención del tiempo medio de viaje en el problema de inversión de carriles.

5.4.3. Implementación de la evaluación con SUMO

De acuerdo a lo indicado en la Sección 5.4.2, la evaluación del tiempo medio de viaje de una solución implica reconstruir la red y la demanda de tráfico.

La tarea de reconstrucción de la red es mas simple que la reconstrucción vista en la Sección 5.3.3. La única operación a realizar para obtener una nueva distribución de los sentidos de circulación consiste en invertir, en el archivo XML de arcos, los parámetros *from* y *to* de las vías seleccionadas. La red de tráfico es un grafo orientado, por lo cual *from* y *to* definen no solo los nodos que conecta el arco, sinó también el sentido de circulación. Con el archivo de arcos modificado, se llama a NETGENERATE y se obtiene la nueva red.

Para el recálculo de la demanda, solo basta ejecutar DUAROUTER con el nuevo archivo de la red y el archivo de demanda original. DUAROUTER recalcula las rutas dada la nueva configuración de sentidos y genera el archivo de viajes.

5.4.4. Implementación del proceso de optimización

A efectos de resolver el problema de inversión de carriles, el Algoritmo 5 se implementó mediante un algoritmo SMPSO. La secuencia de tareas a realizar se codifican dentro del bloque $13 - 17$, de acuerdo a la estructura indicada en el Capítulo 2.

5.5. Inversión de carriles de tráfico con sincronización de semáforos

Para resolver este problema se adaptó el algoritmo formulado en la Sección 5.4 al tratamiento de la sincronización de semáforos. El enfoque de resolución seleccionado consiste en subordinar la sincronización de semáforos a la selección de carriles a invertir, convirtiendo a esta última en una decisión de carácter secundario. Este problema secundario se resolvió por medio de

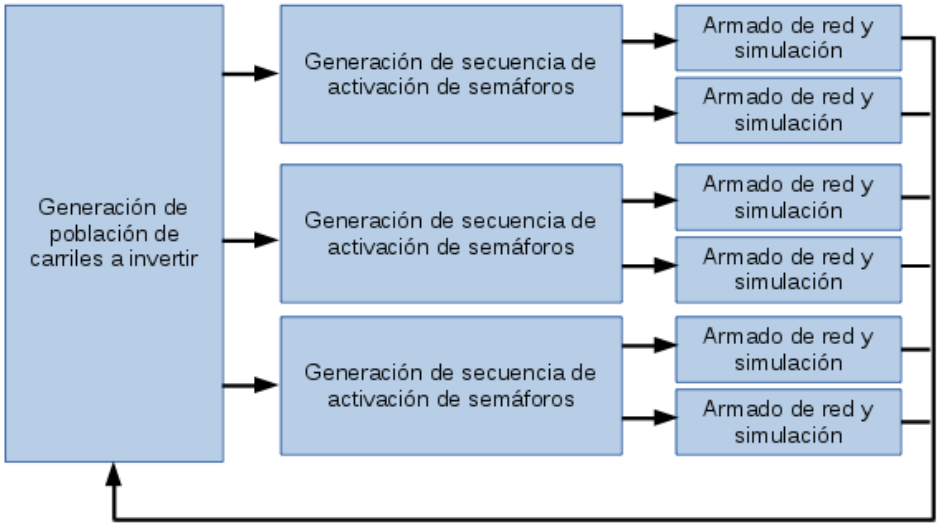


Figura 5.4: Esquema de resolución de la selección de carriles y la sincronización de semáforos.

un algoritmo de SA monoobjetivo, donde el objetivo a minimizar es el tiempo medio de viaje (Ecuación 5.6). En síntesis, se utilizó un esquema *Construir, Combinar, Resolver y Adaptar* (CMSA) [Blum and Raidl, 2016], en el cual la selección de carriles es un proceso de generación de subproblemas. El esquema general de resolución se puede observar en la Figura 5.4. Por cada alternativa de inversión de carriles a evaluar, se genera un nuevo problema de optimización. En este problema se seleccionan configuraciones de semáforos y, en base a éstas y al esquema de inversión, se genera la nueva red y se ejecuta la simulación, para evaluar el tiempo medio de viaje.

5.5.1. Codificación de las soluciones

Para codificar las soluciones, se utilizó una tupla de dos vectores, en el cual la primera componente de la tupla se corresponde con el vector de selección de carriles a invertir presentado en la Sección 5.4.1 y el segundo al vector de configuración de semáforos.

La codificación de este secuenciamiento para una intersección consiste entonces en la definición de un vector de componente enteras en la cual cada una representa la duración de un estado particular. Para codificar todos los semáforos de la red, se utiliza una lista de tantas componente

como intersecciones semaforizadas existan en la cual cada componente contiene el vector de enteros que representa la secuencia de activación de los semáforos de esa intersección particular.

5.5.2. Algoritmo general

El Algoritmo 6 (genérico para cualquier metaheurística seleccionada) es una variación del Algoritmo 5, en el cual la evaluación de los tiempos medios de viaje contempla el efecto de las intersecciones semaforizadas.

Algoritmo 6 OvS Inversión Carriles con Sincronización de Semáforos

```
1: procedure OVS-InversionCarrilesYSincSemaforos(params)
2:     criterioParadaInversion ← falso
3:     poblacionInversiones ← generarPoblacionInicialInversiones
4:     while criterioParadaInversion = falso do
5:         evaluaciones ← {}
6:         for all solucionInv ∈ poblacionInversiones do
7:             generarRed(solucionInv)
8:             criterioParadaSemaforos ← falso
9:             poblacionSemaforos ← generarPoblacionInicialSemaforos
10:            while criterioParadaSemaforos = falso do
11:                evaluacionesSem ← {}
12:                for all solucionSem ∈ poblacionSemaforos do
13:                    calcularRutas(solucionInv, solucionSem)
14:                    tiempoMedio ← simular()
15:                end for
16:                if checkCriterioParadaSem(evalaucionesSem) = verdadero then
17:                    criterioParada ← verdadero
18:                else
19:                    poblacion ← crearNuevaProblacion(evaluaciones)
20:                end if
21:            end while
22:            inv ← calcularInveriones(solucion)
23:            evaluaciones ← evaluaciones ∪ {(solucionInv, solucionSem, tiempoMedio, inv)}
24:        end for
25:        if checkCriterioParadaInversiones(evalauciones) = verdadero then
26:            criterioParadaInversiones ← verdadero
27:        else
28:            poblacion ← crearNuevaProblacionInversiones(evaluaciones)
29:        end if
30:    end while
31:    return evaluaciones
32: end procedure
```

La diferencia entre los Algoritmos 5 y 6 radica en el reemplazo de las líneas $8 - 9$ del primero

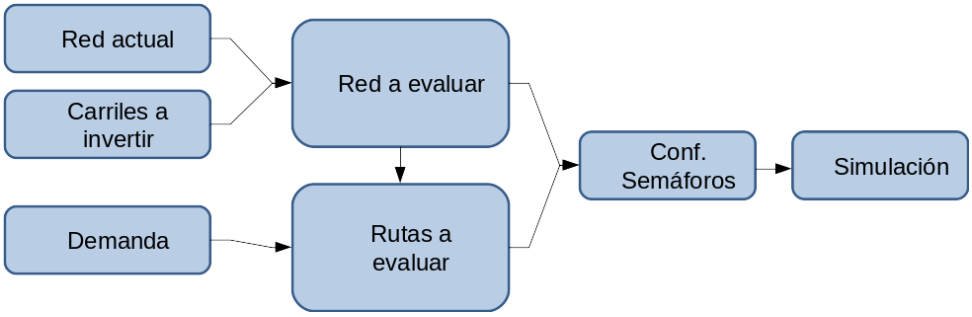


Figura 5.5: Obtención del tiempo medio de viaje en el problema de inversión de carriles con sincronización de semáforos.

por el bloque $8 - 21$ en el segundo. Para cada red de tráfico (línea 7) se ejecuta un algoritmo de optimización que intenta hallar la mejor configuración de semáforos para dicha red. Se crea una población de soluciones (configuraciones de semáforos) y, para cada solución, se generan las rutas de cada vehículo y se calcula el tiempo medio de viaje. Un esquema de este proceso se puede observar en la Figura 5.5.

5.5.3. Implementación de la evaluación con SUMO

La evaluación del tiempo de viaje sigue los mismos lineamientos de la Sección 5.4.3 con el agregado de información relativa a semáforos.

5.5.4. Implementación del proceso de optimización

Para resolver el problema de inversión de carriles con sincronización de semáforos, se utilizó la misma estructura indicada en la Sección 5.4.4 para la parte primaria del algoritmo (la selección de carriles). Para la parte secundaria (la sincronización de semáforos), se utilizó un algoritmo de Recocido Simulado convencional, en el cual el proceso de generación de vecinos consistió en un proceso de mutación aleatoria acotado al 10 % de las componentes del vector a mutar.

5.6. Ruteo de vehículos escolares

Para resolver la versión multiobjetivo del SSBRP[3][4] con demanda aleatoria se utilizó un simulador de eventos discretos diseñado Ad-Hoc junto con la metaheurística NSGA-II.

La meta del esquema utilizado fue la de definir las rutas para cada uno de los autobuses disponibles así como la selección de las paradas a visitar, de manera tal de minimizar los costos de operación (representados mediante la distancia total recorrida por los autobuses) y la distancia total recorrida por los estudiantes hasta las paradas.

A fin de resolverlo, se utilizó un procedimiento del tipo *Construct, Merge, Solve and Adapt (CMSA)* [Blum and Raidl, 2016] mediante el cual se asignó en una primera instancia paradas a vehículos, y en una segunda instancia se construyeron las rutas en forma individual para cada autobús y se evaluó la distancia recorrida por los estudiantes.

Se presenta en esta sección la metodología de resolución desarrollada para el caso determinístico y luego la misma es adaptada a la resolución del problema estocástico.

5.6.1. SSBRP con demanda determinística

Descomposición del SSBRP

Dada su naturaleza, el SSBRP puede ser fácilmente dividido en tres problemas relacionados: la selección de paradas, la asignación de estudiantes a paradas y el ruteo de los buses. A diferencia de las metodologías presentes en la literatura (LAR y ARL)[5][6], que en base a la asignación de estudiantes construyen las rutas de los autobuses, aquí la asignación de estudiantes se subordina a la asignación previa de paradas a autobuses y al armado de las rutas.

El problema de más alto nivel es la asignación de paradas a autobuses. Éste consiste en definir cual es el conjunto de paradas que debe recorrer cada autobús, sin determinar el orden del

[3] Al referenciar al SSBRP se utilizan los términos estudiantes, alumnos, pasajeros y trabajadores indistintamente, a modo de sinónimos para referirse a las personas que deben ser transportadas por los autobuses.

[4] Al referenciar al SSBRP se utilizan los términos autobús, bus y vehículos indistintamente, a modo de sinónimos para referirse a las unidades de transporte.

[5] LAR: Location-Allocation-Routing, conjunto de algoritmos que seleccionan paradas, asignan estudiantes y rutean vehículos

[6] ARL: Allocation-Routing-Location, conjunto de algoritmos que agrupan estudiantes por proximidad y asignan estos grupos a paradas, arman las rutas y realizan la asignación en detalle de estudiantes.

recorrido. En base a esta asignación, se pueden derivar dos grupos de problemas: el primero, un conjunto de problemas del tipo TSP en el cual se secuencian las paradas a recorrer por cada autobús de manera que el tiempo total de viaje sea mínimo; el segundo, un solo problema de asignación de estudiantes a autobuses, en el cual se trata de minimizar la distancia recorrida por los estudiantes a las paradas, restringido por la capacidad de cada bus.

Método de resolución

El algoritmo CMSA desarrollado está basado en el algoritmo NSGA-II. En forma resumida, consiste en un algoritmo NSGA-II convencional en el cual el espacio de soluciones a explorar es el asociado al problema de la asignación de buses a paradas, mientras que los valores de los dos objetivos se calculan mediante la construcción del conjunto de rutas óptimas para cada bus, por un lado, y la asignación óptima de estudiantes a buses y paradas, por otro.

Codificación de las soluciones

Para codificar las soluciones, la manera mas sencilla de representar la asignación de paradas a autobuses consiste en utilizar un vector de enteros normalizado, en el cual cada componente representa a una parada i y su valor el bus k asociado. Cada componente puede tomar cualquier valor en el intervalo $[0, k]$, siendo 0 un bus ficticio que representa que por esa parada no pasa ningún bus. Este método de codificación es un método rápido y sencillo que permite una lectura directa de las soluciones, además que asegura que todas las asignaciones son válidas. A priori, debido al hecho que en este problema todos los buses tienen la misma capacidad, esta codificación presenta el problema que la función de codificación no es inyectiva. Es decir, una misma asignación admite mas de una representación, por lo tanto el algoritmo de resolución se puede encontrar evaluando la misma solución una y otra vez. Por lo tanto, a fin de volver mas eficiente el proceso de cálculo del algoritmo, es necesario el uso de un diccionario que permita identificar soluciones equivalentes ya evaluadas y tomar de ellas los valores de sus objetivos.

Evaluación de las soluciones

Cada solución codificada según lo indicado en la Sección 5.6.1 determina qué bus es asignado a cada parada. Fijado esto, podemos construir los problemas derivados a fin de hallar los valores de los dos objetivos a minimizar.

Para cada bus se puede construir un problema de optimización del tipo TSP, en el cual se requiere buscar la secuencia óptima en la cual el autobús debe visitar las paradas asignadas. Esto se puede formalizar en el siguiente modelo de optimización:

$$\text{mín } Z_1 = \sum_{i \in V} \sum_{j \in V} c_{ij} \cdot x_{ij} \tag{5.8}$$

Sujeto a:

$$\sum_{j \in V} x_{ij} = \sum_{j \in V} x_{ji}, \quad \forall i \in V_k \tag{5.9}$$

$$\sum_{i,j \in Q} x_{ij} \leq |Q| - 1, \quad \forall Q \subseteq V_k \setminus \{v_0\} \tag{5.10}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j \in V, i \neq j \tag{5.11}$$

En el cual, para un determinado bus k, x_{ij} es una variable binaria que vale 1 si el bus viaja desde la parada i hasta la parada j. La Ecuación 5.8 minimiza el tiempo total de viaje del bus k, las Ecuaciones 5.9 y 5.10 impiden la creación de subtours y la Ecuación 5.11 fuerza a x_{ij} a ser una variable binaria. Si la cantidad de paradas a visitar por el bus k no es grande, el problema se puede resolver en forma exacta mediante programación lineal, en caso contrario se puede usar un algoritmo genético.

Respecto de los estudiantes, la asignación optima a buses y paradas se puede formalizar como

sigue:

$$\text{mín } Z_2 = \sum_{l \in S} \sum_{k=1}^{n} f_{lk} \cdot r_{lk}, \quad f_{lk} = \text{mín}\{d_{il}, \forall i \in V_k \subseteq V\} \tag{5.12}$$

Sujeto a:

$$\sum_{l \in S} r_{lk} \leq C_k, \quad k = 1, ..., n \tag{5.13}$$

$$\sum_{k=1}^{n} r_{lk} = 1, \quad \forall l \in S \tag{5.14}$$

$$\forall r_{lk} \in \{0; 1\}, \quad \forall l \in S, \quad k = 1, ..., n \tag{5.15}$$

Donde r_{lk} representa la asignación del estudiante l al bus k, f_{lk} es la distancia mas corta desde la ubicación del estudiante l al conjunto de paradas visitadas por el bus k y C_k es la capacidad del bus k. La Ecuación 5.12 minimiza la distancia total caminada hacia las paradas por los estudiantes. La Ecuación 5.13 fuerza a que el numero de estudiantes en el bus k no exceda la capacidad del mismo. La Ecuación 5.14 fuerza a que cada estudiante sea asignado a un único bus. Si la cantidad de variables r_{lk} no es grande, el problema se puede resolver en forma exacta mediante programación lineal, en caso contrario se puede usar un algoritmo genético.

Algoritmo general

En el Algoritmo 7 se muestra el procedimiento genérico de resolución propuesto. Dicho algoritmo toma como entradas la lista de autobuses, la lista de potenciales paradas, las coordenadas del destino, y la ubicación inicial de los estudiantes. La línea 2 inicializa la bandera que detiene el bucle principal, y la línea 3 inicializa la población inicial de soluciones a evaluar. Las líneas $4 - 22$ se corresponden con el bucle principal del algoritmo. Mientras no se satisfaga el criterio de parada seleccionado, se evalúa cada solución de la población (líneas $5 - 16$) y, de acuerdo a la metaheurística seleccionada, se crea una nueva población. El proceso de evaluación de

Algoritmo 7 OvS SSBRP determinístico

```
1: procedure OVS-SSBRP-DETERMINÍSTICO(params)
2:     criterioParada ← falso
3:     poblacion ← generarPoblacionInicial
4:     while criterioParada = falso do
5:         evaluaciones ← {}
6:         for all solucion ∈ poblacion do
7:             rutas ← []
8:             for all autobus ∈ conjuntoAutobuses do
9:                 paradasAVisitar ← obtenerParadas(solucion, autobus)
10:                ruta ← optimizarRecorrido(paradasAVisitar)
11:                rutas ← rutas ∪ ruta
12:            end for
13:            distTotalBuses ← calcularDistancia()
14:            distTotalEstudiantes ← asignarEstudiantes(estudiantes, rutas)
15:            evaluaciones ← evaluaciones ∪ {(solucion, distTotalBuses, distTotalEstud)}
16:        end for
17:        if checkCriterioParada(evalauciones) = verdadero then
18:            criterioParada ← verdadero
19:        else
20:            poblacion ← crearNuevaProblacion(evaluaciones)
21:        end if
22:    end while
23:    return evaluaciones
24: end procedure
```

cada solución está indicado en las líneas $7 - 14$. La distancia recorrida por los buses se calcula analíticamente en la línea 13, siendo las rutas construidas en las líneas $8 - 12$. La distancia recorrida por los estudiantes se calcula mediante la resolución del problema de PLE asociado, en la línea 14. La salida a entregar por el algoritmo es un conjunto de soluciones que aproximan la Frontera de Pareto del problema analizado.

A fin de reducir costos de computo, es menester incluir un caché que almacene las soluciones de los problemas de ruteo. Muchas de las soluciones generadas por el algoritmo NSGA-II exterior repetiran las asignaciones de paradas para varios buses. Y dado que el problema de ruteo es el subproblema mas oneroso respecto al tiempo de cómputo, el uso de un caché conseguiría ahorros importantes.

Implementación del algoritmo

El bloque exterior del algoritmo (la asignación de paradas a autobuses) se implementó mediante un algoritmo NSGA-II convencional. Para cada asignación de paradas a un autobús concreto, el TSP resultante se resolvió mediante un SA convencional, en el cual el proceso de generación de vecinos consistió en un proceso de permutación aleatoria de dos componentes del vector correspondiente a la solución actual. Y el subproblema de asignación de estudiantes se resolvió mediante PLE.

5.6.2. SSBRP con demanda estocástica

La resolución de esta versión del problema se basa en el algoritmo para el problema determinístico, reemplazando el bloque de asignación de estudiantes por un bloque de simulación que representa la subida de pasajeros a los autobuses.

Codificación de las soluciones y de otros parámetros

El formalismo elegido para codificar las soluciones del SSBRP es el de un vector de $m \cdot n$ componentes binarias. Cada componente representa la asignación o no del bus m a la parada n.

Algoritmo general

En el Algoritmo 8 se muestra el procedimiento genérico de resolución propuesto. Dicho algoritmo toma como entradas la lista de autobuses, la lista de potenciales paradas, las coordenadas del destino, y la distribución de probabilidad de la ubicación inicial de los estudiantes. La línea 2 inicializa la bandera que detiene el bucle principal, y la línea 3 inicializa la población inicial de soluciones a evaluar. Las líneas $4 - 22$ se corresponden con el bucle principal del algoritmo. Mientras no se satisfaga el criterio de parada seleccionado, se evalúa cada solución de la población (líneas $5 - 16$) y, de acuerdo a la metaheurística seleccionada, se crea una nueva población. El proceso de evaluación de cada solución esta indicado en las líneas $7 - 14$.

Algoritmo 8 OvS SSBRP estocástico

```
1: procedure OVS-SSBRP-Estocástico(params)
2:     criterioParada ← falso
3:     poblacion ← generarPoblacionInicial
4:     while criterioParada = falso do
5:         evaluaciones ← {}
6:         for all solucion ∈ poblacion do
7:             rutas ← []
8:             for all autobus ∈ conjuntoAutobuses do
9:                 paradasAVisitar ← obtenerParadas(solucion, autobus)
10:                ruta ← optimizarRecorrido(paradasAVisitar)
11:                rutas ← rutas ∪ ruta
12:            end for
13:            distTotalBuses ← calcularDistancia()
14:            distTotalEstudiantes ← simular()
15:            evaluaciones ← evaluaciones ∪ {(solucion, distTotalBuses, distTotalEstud)}
16:        end for
17:        if checkCriterioParada(evalauciones) = verdadero then
18:            criterioParada ← verdadero
19:        else
20:            poblacion ← crearNuevaProblacion(evaluaciones)
21:        end if
22:    end while
23:    return evaluaciones
24: end procedure
```

La distancia recorrida por los buses se calcula analíticamente en la línea 13, siendo las rutas construidas en las líneas $8-12$. La distancia recorrida por los estudiantes se calcula mediante simulación, en la línea 14, asumiendo que para cada escenario, cada estudiante se desplazará a la parada activa más cercana. Al final de dicha simulación, para todos los alumnos que no fueron trasladados a la escuela, se agrega una penalización. La salida a entregar por el algoritmo es un conjunto de soluciones que aproximan la Frontera de Pareto del problema analizado.

El proceso de armado de las rutas requiere resolver un TSP para cada autobús. En esta tesis se ha optado por resolverlo utilizando un procedimiento de SA. Para el cálculo de las distancias recorridas por los estudiantes, se simula la circulación de cada autobús, de manera de asegurarse que no se violan las restricciones de capacidad de ninguno. Para esta simulación, se utiliza un esquema de eventos discretos simples en el cual los eventos se corresponden con los arribos de autobuses a las paradas. Los autobuses actúan a modo de servidores de procesamiento mientras que los estudiantes como entidades transitorias a ser servidas.

Implementación del proceso de optimización

A efectos de resolver el SSBRP, el Algoritmo 4 se implementó mediante un algoritmo NSGA-II. Las líneas $5-21$ del algoritmo anterior codifican todo el proceso de evaluación de los dos objetivos para cada solución de la iteración en curso. La secuencia de tareas se codifican dentro del bloque $8-15$, de acuerdo a la estructura indicada en el Capítulo 2.

Para la construcción de rutas se utilizó un algoritmo de SA monoobjetivo convencional, en el cual el proceso de generación de vecinos consistió en un proceso de permutación aleatoria de dos componentes del vector correspondiente a la solución actual, con el objetivo de encontrar la ruta mas corta para el bus analizado, dadas las paradas asignadas.

Capítulo 6

Propuestas algorítmicas para problemas de optimización

En este capítulo se describen resultados teóricos que derivaron en la creación de algoritmos de optimización generales desarrollados también durante el transcurso de esta tesis doctoral. Se incluyen tres algoritmos distintos, implementados sobre la base del algoritmo NSGA-II: en el primero se desarrolla una versión novedosa del algoritmo NSGA-II que utiliza metamodelos con el fin de reducir el tiempo de cómputo en aplicaciones de OvS; el segundo es una versión en la cual, dado un esquema de OvS, se define en forma dinámica la longitud de las simulaciones a realizar, de manera tal de reducir el costo individual de cada evaluación (y por ende, el costo total); y en el tercero se desarrolla otra propuesta algorítmica que propone mejorar la calidad de las estimaciones de las Fronteras de Pareto en el caso de problemas estocásticos (no necesariamente basados en OvS). Tanto el primer como el segundo algoritmo intentan solucionar el principal problema de la OvS, el alto costo computacional [1], mientras que el tercero aporta una nueva forma de lidiar con la aleatoriedad en problemas multiobjetivo. Lo mas importante es que si bien las implementaciones se realizaron en base al algoritmo NSGA-II, los tres criterios desarrollados pueden ser exportados a cualquier algoritmo de OvS (los dos primeros) y a cualquier algoritmo de optimización multiobjetivo basado en Fronteras de Pareto (el tercero).

[1] Ver Sección 3.1.3

El capítulo comienza con un método genérico para evaluar cuándo un metamodelo genérico presenta un nivel de calidad adecuado para ser usado como reemplazo de las simulaciones.

6.1. Verificación de metamodelos mediante orden relativo

Los algoritmos de optimización, en general, y las metaheurísticas, en particular, basan la selección de soluciones en medidas de desempeño relativas. Es decir, partiendo de una solución x, el algoritmo se *moverá* hacia la solución x^* si $f(x^*) - f(x) < 0$. Si bien esto es muy simplista, el método de chequeo de diferencias, con variaciones dependiendo de cada algoritmo [2], es la forma en la que estos generan mejores soluciones en sus sucesivas iteraciones. Dada esta metodología de avance, en el cual se compara el resultado de la evaluación de las soluciones mediante relaciones de desigualdad, no resulta necesario[3] que, ante el uso de metamodelos, estos predigan el valor de los objetivos con exactitud, sino que permitan predecir si se cumple o no la desigualdad antes citada.

Se propone en esta sección un método genérico de verificación de metamodelos, para determinar cuando un metamodelo debe ser recalculado o no, basado en muestreo aleatorio, que apela al chequeo relativo de la bondad de una solución respecto de otra, sin importar los valores exactos predichos por el metamodelo.

6.1.1. Estructura del método

A fin de generalizar, supongamos un operador binario $<_{rel}$ y un par de soluciones (i, j), tal que $i <_{rel} j$ devuelva *Verdadero* si i es óptimo respecto de j, y *Falso* en caso contrario. La definición de optimalidad puede ser cualquiera: si es optimalidad escalar, $i <_{rel} j$ vale *Verdadero* si $i < j$, y si es optimalidad de Pareto, $i <_{rel} j$ vale *Verdadero* si i es no-dominada por j, por ejemplo.

[2] Introduciendo torneo en un GA, posibilidad de error en un SA, o evaluando las soluciones según su lugar en el ranking de Pareto, por ejemplo.

[3] Al menos no en todas las metaheurísticas.

Teniendo definido el operador $<_{rel}$, para predecir que tan bueno es el metamodelo, se define una función $I(i,j)$ la cual devuelve 1 si el resultado de $i <_{rel} j$ es el mismo utilizando el metamodelo que calculando el valor de los objetivos para i y j utilizando las funciones objetivo verdaderas (por ejemplo, en un entorno de OvS, simulando las soluciones), y 0 en caso contrario. Luego, en base a esta función, se puede definir una metodología de control basada en muestreo para determinar cuando es necesario recalcular el metamodelo o no. Dada una población de soluciones a evaluar de tamaño N, se toma una muestra de tamaño $k_{check} < N$ de la misma. Para estas k_{check} soluciones, se evalúan sus valores objetivos utilizando tanto las funciones objetivo verdaderas como los metamodelos. Conociendo cual es el resultado de las dos evaluaciones para cada solución, se arman pares de soluciones (i,j), se calcula el operador $<_{rel}$ asociado a las evaluaciones con las funciones reales, el operador $<_{rel}$ asociado a las evaluaciones con metamodelos, y por último se computa el resultado de $I(i,j)$ para cada par. Dado que $I(i,j)$ toma el mismo valor que $I(j,i)$, la cantidad de chequeos a realizar es de $N_{check} = \frac{k_{check}*(k_{check}-1)}{2}$. La decisión de utilizar el metamodelo para evaluar todas las soluciones o no se realiza en base a la proporción de resultados 1 de la función $I(i,j)$ respecto del total de parejas analizadas (Ecuación 6.1).

$$\frac{\sum_{i=1}^{k_{check}-1} \sum_{j=i+1}^{k_{check}} I(i,j)}{N_{check}} \tag{6.1}$$

Si la proporción es mayor que un valor crítico α se considera que los errores del metamodelo no son significativos y se continúa utilizando el mismo para evaluar al resto de las N soluciones. En cambio, si es menor que α, el metamodelo debe ser recalculado y las soluciones evaluadas mediante las funciones objetivo reales.

En esta tesis se utiliza la función $I(i,j)$, en el marco del algoritmo NSGA-II, para desarrollar el algoritmo K-NSGA-II. Pero este método puede ser fácilmente extrapolado a cualquier otro algoritmo evolutivo basado en OvS, sea multiobjetivo o no. También a algoritmos de búsqueda, en los cuales el control debe realizarse cada un cierto número prefijado de iteraciones. En todos los casos, agrega dos parámetros adicionales a la metaheurística en cuestión, α y k_{check} (y en el caso de un algoritmo de búsqueda, la frecuencia de actualización).

6.2. K-NSGA-II: Optimización multiobjetivo mediante metamodelos de Kriging

Tal como se describió en la Secciones 3.1.3 y 3.4, el principal problema práctico de la OvS es su alto costo computacional. Un enfoque popular en la literatura es el uso de metamodelos, de los cuales los metamodelos basados en Kriging son los más frecuentemente utilizados. En esta sección, se describe un algoritmo basado en el algoritmo NSGA-II que utiliza Kriging para estimar los resultados de las funciones objetivo. A diferencia de los algoritmos presentes en la literatura, el control de la calidad de las estimaciones se realiza mediante muestreo, disparando este muestreo la necesidad o no de recalcular en forma completa el metamodelo. Además, el algoritmo desarrollado utiliza dicho muestreo para mejorar en cada iteración el metamodelo en si mismo.

6.2.1. Estructura del K-NSGA-II

La estructura general del algoritmo K-NSGA-II se basa en el algoritmo NSGA-II con modificaciones a fin de adaptar su dinámica a la evaluación mediante metamodelos de las funciones objetivo. El esquema de optimización se divide así en tres fases:

- Una primera etapa (Etapa I) para generar el metamodelo inicial y realizar la primera iteración correspondiente al algoritmo genético.
- Una segunda etapa (Etapa II) consistente en hacer avanzar las iteraciones del algoritmo genético, evaluando las soluciones con el metamodelo y corrigiendo dicho metamodelo cuando sea necesario. El resultado de esta etapa es un recorte del espacio de soluciones, con una aproximación burda de la Frontera de Pareto.
- Una última etapa (Etapa III) de ajuste fino sobre pocas generaciones, en las cuales las soluciones no se evalúan mediante metamodelos, sino mediante simulaciones. Aquí se mejora la aproximación obtenida en la Etapa II.

Cada etapa es, básicamente, una variación del algoritmo NSGA-II, siendo entonces el algoritmo

K-NSGA-II un encadenamiento de estas tres variaciones: se ejecuta la Etapa I, tomando como base sus resultados se ejecuta la Etapa II y se finaliza ejecutando la Etapa III partiendo de la población final obtenida en la Etapa II.

La razón tras el uso de metamodelos para calcular el valor que toma cada uno de los objetivos en vez utilizar el simulador en sí mismo es que, aunque la construcción de dichos metamodelos es costosa en términos computacionales, la ejecución de una simulación tiene un costo similar o mayor, con lo cual al evitar ejecutar muchas simulaciones, el ahorro es significativo (en tiempo y memoria necesaria) [Kleijnen, 2015]. La situación óptima sería la de poder construir un buen metamodelo con pocos datos de entrenamiento (pocas simulaciones) y que los errores del mismo no fueran significativos, algo que no sucede en la práctica. En general, las estimaciones son muy buenas para puntos cercanos a los puntos de entrenamiento, pero el error aumenta a medida que nos alejamos de dichos puntos.

Como se verá en la Sección 6.2.3, K-NSGA-II hace uso de la tendencia a la convergencia de los algoritmos genéticos. En las primeras generaciones, un algoritmo genético es un algoritmo principalmente exploratorio, que trata con soluciones que, en promedio, tienen una distancia grande entre sí. En esta situación, los errores de predicción del metamodelo no son tan importantes, ya que pese a ello se puede estimar bien el valor del operador $\prec_n$. En las últimas generaciones, sin embargo, la distancia promedio entre las soluciones es mucho mas chica, cobrando más magnitud los errores en la predicción de $\prec_n$, por lo cual, se opta por realizar la evaluación en forma exacta, no aproximada (es decir, utilizando simulación). Un esquema general de la estructura de cada etapa se muestra en la Figura 6.1.

6.2.2. K-NSGA-II: Etapa I

La primera etapa consiste en la generación de la población inicial, la evaluación de las soluciones, el cálculo de los metamodelos (uno para cada objetivo) y la creación de la segunda generación de soluciones.

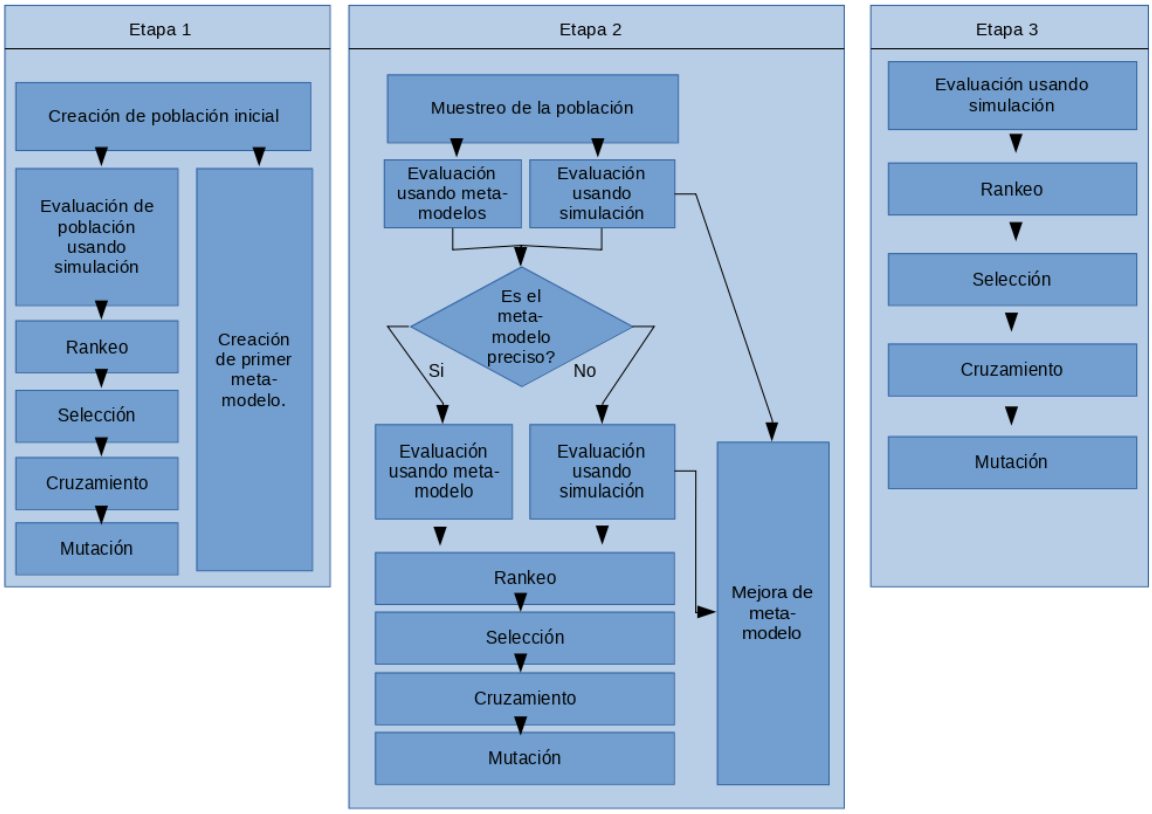


Figura 6.1: Esquema de cada una de las etapas del K-NSGA-II.

La generación de la población inicial se realiza en forma aleatoria. Siendo $p_s = |P|$ el tamaño de la población seleccionado, se crean $2p_s$ soluciones de manera estocástica y se evalúan todas mediante simulación. Con estos resultados, se estima el primer metamodelo (de aquí en adelante, hay que observar que cuando se dice metamodelo se está refiriendo a todo el conjunto de metamodelos, uno asociado a cada objetivo). Luego, del conjunto de tamaño $2p_s$ se eligen p soluciones aleatoriamente, siendo estas las que conforman la primera generación. Se aplican los procesos de selección, cruzamiento, mutación y elitismo obteniendo así la segunda generación.

En esta primera etapa, la selección mediante el operador $\prec_n$ se realiza en forma exacta, ya que se disponen de los resultados de las simulaciones de todas las soluciones a evaluar. Asimismo, se logra un metamodelo con una buena precisión de arranque, al considerar el doble de soluciones para su cálculo. En síntesis, la Etapa I consiste en un algoritmo NSGA-II convencional, sumado a un proceso de construcción del primer metamodelo.

6.2.3. K-NSGA-II: Etapa II

Es la etapa más compleja y que más ahorro computacional (a nivel de costos de simular) genera. Consiste en avanzar desde la generación 2 hasta el inicio de la Etapa III evaluando las soluciones mediante el metamodelo. Nuevamente, funciona como un NSGA-II convencional a excepción de la etapa de selección y del proceso de elitismo. Los dos cambios que se realizan consisten en que la evaluación se realiza mediante metamodelos, y que el elitismo se separa en dos, uno positivo y otro negativo (tendiente a la inclusión de malas soluciones).

Selección

La idea principal del K-NSGA-II consiste en evaluar las soluciones utilizando un metamodelo en vez de las funciones originales, pero asegurándose que la selección de individuos mediante el operador $\prec_n$ sea la misma que utilizando las funciones originales (el mismo criterio se aplica a las restricciones).

Como cualquier modelo de aproximación, Kriging tiene margen de error en su estimación. Incluso, como subproducto del cálculo del metamodelo, se obtiene un estimador de dicho error para cada solución a evaluar. La mayoría de los algoritmos de optimización basados en Kriging hace uso de este estimador para determinar cuando la predicción del metamodelo es aceptable o no. Pero a efectos de seleccionar individuos para crear la población de la próxima generación, la precisión en la estimación de los valores de las funciones objetivos no es tan importante. Lo importante es la estimación relativa de cual solución de cada par de soluciones es la no-dominada. Mientras que el metamodelo permita estimar el resultado del operador $\prec_n$ en forma correcta, el metamodelo es válido para seguir siendo utilizado (aún si el margen de error de cada predicción individual es alto), si la estimación de la aplicación de $\prec_n$ es incorrecta, se deben evaluar las soluciones mediante simulación y recalcular el metamodelo.

Para predecir que tan bueno es el metamodelo, utilizamos la función $I(i, j)$ definida en la Sección 6.1, junto con el procedimiento de control definido en la misma sección. Si la proporción de valores desfavorables en la función $I(i, j)$ es mayor que un valor crítico α se considera que los errores del metamodelo no son significativos y no se simulan el resto de las soluciones. En

cambio, si es menor que α, se simulan todas las soluciones de la población. En ambos casos, el metamodelo es recalculado: en el primer caso se agregan las soluciones usadas para la prueba, en el segundo caso todas las soluciones de la población. Por lo cual, independientemente del resultado de la generación actual, la siguiente utilizará un metamodelo mas preciso.

Elitismo aleatorio

El método aleatorio de evaluación de la calidad del metamodelo (Sección 6.2.3) tiene el problema de incluir un posible sesgo en el funcionamiento del proceso de clasificación y selección de soluciones en función de los individuos seleccionados en la muestra. Dado que el proceso es aleatorio, se corre el riesgo de seleccionar, para el proceso de chequeo mediante la función $I(i,j)$, solo individuos para los cuales el metamodelo funciona bien y no evaluar ninguno en los que el modelo funciona mal (o viceversa). Ello implica que el proceso de generación de la nueva población puede descartar soluciones buenas e incluir soluciones malas. Para paliar ese efecto, se modifica el proceso de elitismo para incluir una mayor diversidad, a costa de disminuir la velocidad de convergencia. Siendo n el tamaño de población, se define un valor de elitismo negativo $n_{ea} << n$. Partiendo de la población P_m, el proceso para generar P_{m+1} es el siguiente [4]:

1. Se crea la población P_{m+1}^{good} seleccionando los mejores individuos en un esquema de torneo.
2. Se crea la población Q_m a partir de P_{m+1}^{good}.
3. se construye $R_m = P_m \cup Q_m$
4. Se elijen los $n - n_{ea}$ mejores individuos de R_m para formar la población R_m^*.
5. Se crea la población P_{m+1}^{bad} seleccionando n_{ea} peores individuos de P_m mediante la aplicación del operador $i \prec_n j$ (en forma simple, sin torneo).
6. Se construye $P_{m+1} = R_m^* \cup P_{m+1}^{bad}$

[4]La simbología utilizada en esta sección es la misma que la utilizada en la Sección 2.4.2: P_m es la población actual, P_{m+1} es la población de la siguiente generación, R_m es el conjunto de individuos seleccionados para la próxima generación, y Q_m es el conjunto de descendientes generados a partir del conjunto P_m.

Esta variación del elitismo convencional tiene el efecto de reducir la probabilidad de convergencia prematura del algoritmo.

6.2.4. K-NSGA-II: Etapa III

Como se mencionó previamente, la Etapa II tiene como finalidad principal reducir el tamaño del espacio de soluciones a explorar. En muchos casos, la aproximación de la Frontera de Pareto al final de dicha etapa es buena, pero en otros casos solo copia la forma de la Frontera pero con soluciones dominadas, o bien ni siquiera alcanza a copiar la forma de la misma. Esto se debe a la dinámica de los algoritmos genéticos: la distancia media entre soluciones tiende a disminuir a medida que avanzan las generaciones, y para el metamodelo es mas difícil lograr una predicción correcta. Por lo tanto, en la Etapa III se toma como población inicial la población final de la Etapa II y se ejecuta un NSGA-II convencional de pocas generaciones (con selección sin metamodelos y elitismo tradicional). Esta etapa funciona como una etapa de ajuste fino o tunning, en la cual se explora una región muy acotada del espacio de soluciones a fin de mejorar la aproximación de la Frontera de Pareto devuelta por el algoritmo.

6.3. VSL-NSGA-II: Optimización multiobjetivo mediante simulaciones de longitud variable

En esta sección se describe un algoritmo basado en NSGA-II en el cual la longitud necesaria de las simulaciones en cada etapa se define en forma dinámica mediante muestreo. Tal como se describió en la Sección 3.1.3 y se mencionó en la introducción de este capítulo, el principal problema práctico de la OvS es su alto costo computacional. Existe mucha investigación al respecto de como armar el diseño experimental DS, como seleccionar cuantas simulaciones realizar en cada iteración, como distribuir los experimentos en el espacio probabilístico asociado, pero no así respecto a la longitud de las simulaciones en sí, eligiéndose en general métodos *estáticos* para definir su valor. El algoritmo VSL-NSGA-II funciona como un metamodelo que, utilizando simulaciones de longitud $l_{min} \leq l$ permite reemplazar simulaciones de longitud l.

6.3.1. Estructura del VSL-NSGA-II

La estructura general del VSL-NSGA-II es básicamente la misma que el NSGA-II canónico [5], modificándose solamente el procedimiento de evaluación de las soluciones. En un esquema de OvS convencional, se define de antemano la longitud l de cada simulación, y todas las simulaciones se ejecutan durante dicho periodo l. La propuesta del algoritmo desarrollado en esta sección consiste en, para cada iteración del algoritmo, solo simular en forma completa un pequeño número de soluciones, y en base a ellas estimar cual es la longitud de simulación mínima a partir de la cual las relaciones de optimalidad se mantienen.

Dado un conjunto de soluciones n a simular, se elije al azar un conjunto n^* (cuyo tamaño $|n^*| < |n|$ tiene que ser definido con anterioridad). Para este conjunto n^* se ejecutan las simulaciones necesarias usando la longitud l, llevándose un registro de los valores obtenidos para los objetivos en cada uno de los intervalos de longitud l^* en los que se puede dividir l (l^* también debe ser definido con anterioridad). Con este listado de los valores de los objetivos al final de la simulación y en cada uno de los subintervalos, se puede determinar a partir de qué subintervalo el orden relativo (respecto del criterio de optimalidad) entre soluciones se mantiene constante. Es decir, cuál es la mínima longitud de simulación l_{min} necesaria para arribar al mismo orden relativo que si se realizaran simulaciones completas. Luego, el resto de las soluciones se simulan un tiempo l_{min}.

A fin de detectar l_{min} con el menor margen de error, la búsqueda se debe realizar de adelante hacia atrás. Es decir, empezando con el orden relativo para la longitud l, se chequea si dicho orden se mantiene para la longitud $l - l^*$. Si no se mantiene, $l_{min} = l$. Si se mantiene, se chequea la longitud $l - 2 \cdot l^*$, donde si no se mantiene el orden relativo se hace $l_{min} = l - l^*$ y en caso contrario se repite el proceso hasta que el orden relativo no se mantenga.

El criterio de mantener el orden relativo es muy estricto, por lo cual podría reemplazarse por el criterio de mantener un nivel α definido con anterioridad (por ejemplo, 0,95).

Para el caso de problemas monoobjetivo, el criterio de optimalidad a utilizar es el de optimalidad escalar. Para el caso de problemas multiobjetivo en el cual dos o mas objetivos se puedan

[5] Ver Sección 2.4.2 para una descripción del mismo.

evaluar con el mismo modelo de simulación, se utiliza el criterio de optimalidad escalar, se calcula l_{min} para cada objetivo y se elije el l_{min} de mayor valor. Si todos los objetivos que requieren simulación para evaluarse se pueden evaluar con el mismo modelo de simulación, se puede utilizar el criterio de optimalidad de Pareto

6.4. Fronteras de Pareto estocásticas

Como se describió en el Capítulo 2, los enfoques de optimización estocástica multiobjetivo basados en la reducción del problema (a monoobjetivo estocástico o multiobjetivo determinística) no garantizan ni pertenencia ni cercanía de las soluciones encontradas a la Frontera Pareto de un escenario concreto. Solo aseguran un desempeño basado en la media [6]. Es decir, al momento de implementar una solución, el tomador de decisiones no puede estimar que tan buena es dicha solución. En ese sentido, el uso de un criterio de dominancia estocástica para estimar la Frontera de Pareto [Gannouni et al., 2017] permite obtener soluciones con un mejor grado de calidad ante escenarios con mucha aleatoriedad. En esta sección, se describe un enfoque alternativo para calcular la dominancia estocástica, no basado en probabilidades de dominancia, sino en considerar como una variable aleatoria el valor del ranking de Pareto asociado a cada solución. Dicho enfoque se implementa dentro del marco del algoritmo NSGA-II. La Sección 6.4.1 describe el método de generación de los escenarios a evaluar, la Sección 6.4.2 describe el nuevo criterio de optimalidad de Pareto para objetivos estocásticos desarrollado en esta tesis, la Sección 6.4.3 presenta un criterio para computar el ranking de Pareto que establece una distancia de indiferencia a la Frontera de Pareto, y la Sección 6.4.4 muestra su implementación en el marco del algoritmo NSGA-II. El algoritmo desarrollado se ha denominado *Stochastic Elitist Non-Dominated Sorted Genetic Algorithm* (S-NSGA-II).

6.4.1. Muestreo de las funciones objetivo

A fin de poder considerar la aleatoriedad de los objetivos en el algoritmo, un método posible es el uso de muestreo fijo. Este consiste en muestrear aleatoriamente, de una población Ω, un

[6] O del estadístico utilizado.

conjunto de N escenarios $w_i (i = 1, 2, ...N)$ convirtiendo así la población original en una muestra discreta, en la cual cada escenario es equiprobable [Gutjahr and Pichler, 2016]. Este conjunto de escenarios es referido también con el nombre de *Diseño Experimental* (DS).

Como prerequisito, entonces, para avanzar con el proceso de optimización, es necesario realizar el muestreo de los parámetros de $\vec{f}$ y construir el conjunto DS.

6.4.2. Criterio de optimalidad según ranking de Pareto

Dado un conjunto de soluciones factibles de un problema multiobjetivo, podemos establecer un ranking de Pareto entre ellas de la siguiente manera[7]:

1. A todas las soluciones no-dominadas del conjunto, le otorgamos el valor 1 del ranking.
2. Retiramos todas las soluciones no-dominadas del conjunto.
3. En este subconjunto del conjunto original, calculamos las soluciones no-dominadas y les otorgamos el valor 2 del ranking.
4. Retiramos todas las soluciones no-dominadas del subconjunto.
5. Repetimos el proceso hasta que todas las soluciones tengan un valor de ranking.

Dado un problema de optimización multiobjetivo estocástico, podemos repetir el proceso anterior para cada escenario, obteniendo para cada solución un conjunto de valores de ranking, uno por cada escenario. Debido a que los parámetros que definen a cada escenario son variables aleatorias, el valor en el ranking de Pareto de cada solución es también una variable aleatoria. En un problema determinístico, una solución es no-dominada respecto de otra si el valor en el ranking es menor (ver Capítulo 2). En un problema estocástico podemos plantear el postulado siguiente: *una solución es no-dominada respecto de otra si, aplicando el mismo estadístico al valor de ranking de cada una de las dos soluciones, el valor del estadístico de la primera es menor que el valor del estadístico de la segunda.*

[7] Ver Sección 2.4.2 para una explicación mas detallada.

En esta tesis, el estadístico elegido fue el *percentil* p, siendo p un parámetro a definir por el usuario.

6.4.3. Criterio de rankeo según distancia de indiferencia

Dada la naturaleza estocástica de los problemas bajo análisis, es poco probable que una solución pertenezca a la Frontera de Pareto de todos los escenarios evaluados. Es más, suponiendo tres escenarios, una solución $\vec{x}$ con valores de ranking de Pareto $(2, 2, 3)$ sería preferible a una solución $\vec{x}^*$ con valores de ranking $(1, 1, 7)$ [8]. La primera, si bien nunca está en la Frontera de Pareto, está muy cerca en todos los escenarios, mientras que la segunda es óptima en dos escenarios pero está muy lejos de la optimalidad en el tercero.

Una forma de favorecer la selección de soluciones en que tengan una alta probabilidad de estar cerca de la Frontera de Pareto, es establecer una distancia d de indiferencia. Esto es, si en un conjunto de soluciones evaluadas en un escenario individual, la solución $\vec{x}$ tiene una valor de ranking de Pareto menor o igual a d, se cambia el valor real por 1. Llamando $rank_i$ al valor en el ranking de Pareto en el escenario i, y $rank_i^*$ al valor luego de aplicar la distancia de indiferencia:

- $rank_i^*(\vec{x}) = 1$ si $rank_i(\vec{x}) \leq d$
- $rank_i^*(\vec{x}) = rank_i(\vec{x})$ si $rank_i(\vec{x}) > d$

Hay que notar que la distancia d tiene unidades de ranking de Pareto. Es decir, no se corresponde a las unidades de los objetivos, sino que es un parámetro de cercanía al óptimo de un conjunto finito de soluciones.

6.4.4. Stochastic Elitist Non-Dominated Sorted Genetic Algorithm

A fin de adaptar el algoritmo NSGA-II al ámbito estocástico, es necesario modificar la forma en la cual las soluciones son evaluadas y el criterio de selección de las mismas. El resto de las

[8] Obviamente, la preferencia termina siendo una cuestión subjetiva. No es una norma.

operaciones del algoritmo se mantienen sin cambios, tal cual lo mostrado en la Sección 2.4.2. El algoritmo añade un nuevo conjunto de parámetros a definir por el usuario:

- N: el tamaño del diseño experimental DS
- d: la distancia de indiferencia a la Frontera de Pareto.
- p: percentil a utilizar para comparar el valor del ranking de Pareto estocástico entre un conjunto de soluciones. Si $p = 100$, la comparación se realiza al nivel de valores máximos.

Evaluación y selección de soluciones para la próxima generación

En el proceso de evaluación de las soluciones, cada objetivo de cada solución es evaluado en cada uno de los escenarios definidos en DS. Por cada escenario $w_l \in DS$ tenemos entonces

El proceso de evaluación de las soluciones se gestiona a nivel de escenario. Por cada escenario $w_i \in DS$, se evalúan todas las soluciones y se calcula el valor de ranking de Pareto ($rank_i$) de cada una. Luego, se aplica a cada valor de ranking la distancia de indiferencia para obtener $rank_i^*$. Esta secuencia se repite para todos los escenarios de DS, generándose así, para cada solución, un conjunto de valores de $rank_i^*$. Finalmente, para cada solución es calculado el valor de ranking en el percentil p deseado, siendo este valor el valor el usado en la selección por torneo y la comparación mediante el Operador de Comparación de Agrupamiento (CCO).

Capítulo 7

Experimentos

En este capítulo se describen los experimentos realizados y los resultados a los que se arribó, respecto de los algoritmos y problemas presentados en los Capítulos 5 y 6. Se presentan primero los experimentos relativos a los algoritmos generales y, luego, los algoritmos aplicados a problemas de tráfico.

7.1. K-NSGA-II

En esta sección se muestran los experimentos realizados respecto del algoritmo K-NSGA-II, descripto en la Sección 6.2. El objetivo de las pruebas realizadas es determinar si el método logra reducciones significativas en la cantidad de evaluaciones necesarias (simulaciones en caso de OvS) sin comprometer la calidad de la estimación de la Frontera de Pareto. Para ello, los análisis se realizan sobre conjuntos de problemas definidos analíticamente y ampliamente usados en la bibliografía para comparar algoritmos multiobjetivo. A fin de evaluar el rendimiento del mismo, se lo compara con los algoritmos NSGA-II y K-MOGA. El primero, que no utiliza metamodelos, se emplea para definir el nivel de calidad a alcanzar. El segundo es otro algoritmo basado en NSGA que si utiliza metamodelos para reducir la cantidad de evaluaciones.

La Sección 7.1.1 detalla los problemas de prueba, la Sección 7.1.2 enuncia las métricas utilizadas, en la Sección 7.1.3 se muestran las configuraciones de todos los algoritmos y en la Sección 7.1.4 se muestran todas las comparaciones y el análisis de los resultados. Adicionalmente, la Sección

7.1.5 analiza el impacto del Elitismo Negativo. Con el fin de simplificar la escritura, se llamará K-NSGA-II a las versión del algoritmo propuesto que **no utiliza** la Etapa III, y K-NSGA-II-S3 a la versión completa del algoritmo.

7.1.1. Problemas evaluados

Con el fin de evaluar el funcionamiento del algoritmo propuesto, se utilizaron un conjunto de problemas de la literatura que son usualmente empleados para evaluar el rendimiento de algoritmos de optimización multiobjetivo [Deb et al., 2002]. Las Tablas 7.1 y 7.2 muestran, para cada problema, las funciones objetivo, las cotas de las variables de decisión y las restricciones.

Para generar las Fronteras de Pareto de referencia (benchmark) de cada uno de los problemas, se realizó el siguiente procedimiento :

1. Se muestrearon en forma uniforme 1000000 puntos del espacio de objetivos para cada problema. Para cada punto no factible, se remuetreo dicho punto.

2. Se ejecutaron 100 instancias del algoritmo NSGA-II para cada problema, seleccionando la población inicial de cada uno en forma aleatoria.

El procedimiento (1) permitió obtener una muestra muy grande del espacio de objetivos, mientras que el procedimiento (2) generó para cada problema un conjunto de 100 Fronteras de Pareto estimadas. Luego de ejecutar estos procedimientos, todos los puntos obtenidos fueron agrupados en un único conjunto y se calculó la Frontera de Pareto de este conjunto, usándose Frontera de referencia en todos los experimentos.

7.1.2. Métricas utilizadas

Con el fin de evaluar el rendimiento del K-NSGA-II, se utilizaron dos tipos de métricas: métricas de calidad y de costo. Para las primeras, se utilizaron las siguientes cuatro, previamente descriptas en la Sección 2.5

- Distancia Generacional (GD)

Tabla 7.1: Problemas para comparaciones en algoritmos K-NSGA-II, VSL-NSGA-II y S-NSGA-II - Parte 1

Problema	Objetivos	Cotas	Restricciones
Belegundu	$f_1(x,y) = -2 \cdot x + y$ $f_2(x,y) = 2 \cdot x + y$	$0 \leq x \leq 5$ $0 \leq y \leq 3$	$0 \geq -x + y - 1$ $0 \geq x + y - 7$
Binh1	$f_1(x,y) = x^2 + y^2$ $f_2(x,y) = (x-5)^2 + (y-5)^2$	$-5 \leq x, y \leq 10$	-
Binh2	$f_1(x,y) = 4 \cdot x^2 + 4 \cdot y^2$ $f_2(x,y) = (x-5)^2 + (y-5)^2$	$-5 \leq x, y \leq 10$	$0 \geq (x-5)^2 + y^2 - 25$ $0 \geq -(x-8)^2 - (y+3)^2 + 7{,}7$
Binh3	$f_1(x,y) = x - 10^6$ $f_2(x,y) = y - 2 \cdot 10^6$ $f_3(x,y) = x \cdot y - 2$	$10^{-6} \leq x, y \leq 10^6$	-
Deb3	$f_1(x,y) = 4 \cdot x$ $f_2(x,y) = g(y) \cdot h(f_1(x,y), g(y))$ *con :* $g(a) = (4 - 3 \cdot \exp(-((a-0{,}2)/0{,}02)^2))$ $h(b,c) = 1 - (\frac{b}{c})(0{,}25 + 3{,}75 \cdot (b-1))$	$0 \leq x, y \leq 1$	-
Fonseca1	$f_1(\vec{x}) = 1 - exp[-1 \cdot (x_1 - 1)^2 - 1 \cdot (x_2 + 1)^2]$ $f_2(\vec{x}) = 1 - exp[-1 \cdot (x_1 + 1)^2 - 1 \cdot (x_2 - 1)^2]$	$-1 \leq x_i \leq 1,$ $i = 1 \,..\, n$	-
Fonseca2	$f_1(\vec{x}) = 1 - exp[-\sum_{i=1}^{n}(x_i - \frac{1}{\sqrt{n}})^2]$ $f_2(\vec{x}) = 1 - exp[-\sum_{i=1}^{n}(x_i + \frac{1}{\sqrt{n}})^2]$	$-4 \leq x_i \leq 4,$ $i = 1 \,..\, 2$	-
Hanne1	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}) = x_2$	$-4 \leq x_i \leq 4,$ $i = 1 \,..\, 2$	$\sum_{i=1}^{i=2} x_i - 5 \leq 0$
Hanne2	$f_1(\vec{x}) = (x_1)^2$ $f_2(\vec{x}) = (x_2)^2$	$-4 \leq x_i \leq 4,$ $i = 1 \,..\, 2$	$\sum_{i=1}^{i=2} x_i - 5 \leq 0$
Hanne3	$f_1(\vec{x}) = \sqrt{x_1}$ $f_2(\vec{x}) = \sqrt{x_2}$	$-4 \leq x_i \leq 4,$ $i = 1 \,..\, 2$	$\sum_{i=1}^{i=2} x_i - 5 \leq 0$

Tabla 7.2: Problemas para comparaciones en algoritmos K-NSGA-II, VSL-NSGA-II y S-NSGA-II - Parte 2

Problema	Objetivos	Cotas	Restricciones
Hanne4	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}) = x_2$	$-4 \leq x_i \leq 4,$ $i = 1 \,..\, 2$	$x_2 - 5 + 0{,}5 \cdot x_1 \cdot \sin(4 * x_1) \leq 0$
Hanne5	$f_1(\vec{x}) = trunc(x_1) + 0{,}5 + c_2(\vec{x}) \cdot \sin(c_1(\vec{x}))$ $f_2(\vec{x}) = trunc(x_2) + 0{,}5 + c_2(\vec{x}) \cdot \sin(c_1(\vec{x}))$ $con:$ $c_1(\vec{x}) = 2 \cdot \pi \cdot (x_1 - trunc(x_1))$ $c_2(\vec{x}) = x_1 - trunc(x_1)$	$0 \leq x_i \leq 1,$ $i = 1 \,..\, 30$	$x_2 - 5 + 0{,}5 \cdot x_1 \cdot \sin(4 * x_1) \leq 0$
Jimenez	$f_1(\vec{x}) = 5 \cdot x_1 + 3 \cdot x_2$ $f_2(\vec{x}) = 2 \cdot x_1 + 8 \cdot x_2$	$10^{-6} \leq x, y \leq 10^6,$ $i = 1 \,..\, 2$	$x_1 + 4 \cdot x_2 - 100 \leq 0$ $3 \cdot x_1 + 2 \cdot x_2 - 150 \leq 0$ $200 - 5 \cdot x_1 - 3 \cdot x_2 \leq 0$ $75 - 2 \cdot x_1 - 8 \cdot x_2 \leq 0$
VNT	$f_1(\vec{x}) = \frac{(x_1)^2+(x_2)^2}{2} + \sin((x_1)^2 + (x_2)^2)$ $f_2(\vec{x}) = (3 \cdot x_1 - 2 \cdot x_2 + 4)^{\frac{1}{4}} + (x_1 - x_2 + 1)^{\frac{2}{27}} + 15$ $f_3(\vec{x}) = \frac{1}{(x_1)^2+(x_2)^2+1} - 1{,}1 \exp(-(x_1)^2 - (x_2)^2)$	$-3 \leq x_1 \leq 3$ $-2 \leq x_2 \leq 2$	-
ZDT1	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}) = g(\vec{x}) \cdot h(f_1(\vec{x}), g(\vec{x}))$ $con:$ $g(\vec{x}) = 1 + \frac{9}{29} \sum_{i=2}^{i=30} x_i$ $h(u, v) = 1 - \sqrt{\frac{u}{v}}$	$0 \leq x_i \leq 1,$ $i = 1 \,..\, 30$	-
ZDT2	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}) = g(\vec{x}) \cdot h(f_1(\vec{x}), g(\vec{x}))$ $con:$ $g(\vec{x}) = 1 + \frac{9}{29} \sum_{i=2}^{i=30} x_i$ $h(u, v) = 1 - (\frac{u}{v})^2$	$0 \leq x_i \leq 1,$ $i = 1 \,..\, 30$	-
ZDT3	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}) = g(\vec{x}) \cdot h(f_1(\vec{x}), g(\vec{x}))$ $con:$ $g(\vec{x}) = 1 + \frac{9}{29} \sum_{i=2}^{i=30} x_i$ $h(u, v) = 1 - \sqrt{\frac{u}{v}} - (\frac{u}{v}) \cdot \sin(10 \cdot \pi \cdot f_1(\vec{x}))$	$0 \leq x_i \leq 1,$ $i = 1 \,..\, 30$	-

- Uniformidad (GS)
- Hipervolumen Dominado (DH)
- Indicador Épsilon (AEI)

El DH y el AEI fueron seleccionados debido a que ambos condensan en un único valor la proximidad de las soluciones a la Frontera de Pareto real y la calidad de la distribución de las soluciones. La GD y la GS permiten evaluar en forma separada la cercanía a la Frontera de Pareto real y que tan agrupadas o no están las soluciones halladas [Zitzler and Thiele, 1999, Zitzler et al., 2007, Jiang et al., 2014, Riquelme et al., 2015, Liefooghe and Derbe, 2016].

Respecto de las métricas de costos, se eligió la cantidad de evaluaciones realizadas para cada objetivo sin el uso del metamodelo (en el caso de un esquema de OvS, esto es equivalente al número de simulaciones). Esta métrica permite evaluar el impacto del algoritmo en el costo computacional sin verse afectado por la duración media del proceso de evaluación de soluciones.

7.1.3. Configuraciones utilizadas

Para poder evaluar la calidad del K-NSGA-II y del K-NSGA-II-S3, se comparó su rendimiento con las metaheurísticas NSGA-II [Deb et al., 2002] y K-MOGA [Li et al., 2008]. Todos se ejecutaron utilizando un criterio de detención que consiste en finalizar la optimización cuando el DH de la generación actual es mayor o igual al 95 % de la Frontera de Pareto de referencia. Esto permite comparar todos los algoritmos en las mismas condiciones. En la Tabla 7.3, se muestra la configuración general de los algoritmos. En el caso de K-NSGA-II y K-NSGA-II-S3, $k_{check} = 10$ y $alpha = 0{,}90$. Finalmente, para K-NSGA-II-S3, se estimó la duración de la última etapa de manera tal que el 20 % de las ejecuciones estuvieran comprendidas dentro de dicha etapa. Debido al hecho de que todos los algoritmos son algoritmos pseudoaleatorios, cada ejecución se repitió 100 veces, con diferentes semillas aleatorias. Estos valores (los parámetros) surgieron de experimentos preliminares, donde se probaron distintas combinaciones de valores de estos parámetros. En la Sección 7.1.6 se analiza el impacto de dichos parámetros.

Tabla 7.3: Parámetros del algoritmo K-NSGA-II.

Parámetro	Valor
Tamaño de la población	50
Tamaño del torneo	2
Probabilidad de cruzamiento	0.8
Probabilidad de mutación	0.1
Tasa de elitismo	0.1

7.1.4. Comparación del rendimiento

A continuación, se muestran los resultados de la comparación entre los algoritmos. Las Tablas 7.4 y 7.5 muestran la cantidad de evaluaciones (simulaciones) necesarias para alcanzar el 95 % del Hipervolumen Dominado de la Frontera de Pareto de cada problema. La primer columna es el nombre del problema evaluado, la segunda la medida utilizada (promedio y percentiles 5 y 95) y el resto son los valores alcanzados por los algoritmos NSGA, K-MOGA, K-NSGA-II y K-NSGA-II-S3, respectivamente. En cada fila, el valor en **negrita** indica la menor cantidad de evaluaciones y el valor en *cursiva* la segunda menor cantidad de evaluaciones.

Con respecto al DH, se observa que el algoritmo K-NSGA-II necesita un número importante de evaluaciones para alcanzar el nivel de calidad objetivo. La reducción en la cantidad de evaluaciones es nula (en algunos casos, se necesitan mas evaluaciones). Pero, para el K-NSGA-II-S3, se observa que alcanza el nivel de calidad deseado usando solo una fracción de las evaluaciones realizadas por el NSGA-II convencional. Dado que el algoritmo K-NSGA-II y el K-NSGA-II-S3 difieren solamente en la ejecución o no de la Etapa III, la cual es básicamente un algoritmo NSGA-II convencional ejecutado durante pocas generaciones, se puede inferir que el efecto real de las Etapas I y II en el K-NSGA-II es el de reducir el espacio de soluciones a explorar a la cercanía de la Frontera de Pareto. Las Etapas I y II cumplen entonces un rol exploratorio, descartando vastas porciones del espacio de soluciones, apelando a que los errores en las predicciones del metamodelo no son significativos cuando la distancia entre soluciones es grande. Sin embargo, los errores del metamodelo tornan inviable su uso en las inmediaciones de la Frontera

Tabla 7.4: Cantidad de evaluaciones necesarias para alcanzar el 95 % del valor del Hipervolumen Dominado de la Frontera de Pareto de referencia - Parte 1

Problema	Medida	NSGA-II	K-MOGA	K-NSGA-II	K-NSGA-II-S3
Belegundu	Percentil 5	348	*204*	218	**192**
	Media	397	245	*226*	**207**
	Percentil 95	421	293	*287*	**248**
Binh1	Percentil 5	753	625	**536**	*573*
	Media	1520	*681*	799	**630**
	Percentil 95	2474	**763**	948	*775*
Binh2	Percentil 5	691	*554*	548	**502**
	Media	759	*603*	626	**588**
	Percentil 95	854	*691*	703	**624**
Binh3	Percentil 5	1417	*1309*	1539	**1106**
	Media	1752	*1486*	1805	**1308**
	Percentil 95	2003	*1960*	2158	**1897**
Deb3	Percentil 5	**368**	415	507	*395*
	Media	**401**	507	703	*413*
	Percentil 95	*487*	670	878	**481**
Fonseca1	Percentil 5	806	**437**	1047	*478*
	Media	1483	*574*	1378	**515**
	Percentil 95	2521	*807*	2042	**759**
Fonseca2	Percentil 5	742	*591*	1329	**382**
	Media	1461	*854*	1787	**462**
	Percentil 95	1763	*1203*	2177	**703**
Hanne1	Percentil 5	948	*608*	836	**535**
	Media	1104	*745*	973	**623**
	Percentil 95	1572	*871*	1168	**715**
Hanne2	Percentil 5	*857*	948	1045	**801**
	Media	1073	*902*	1231	**865**
	Percentil 95	1298	*1036*	1318	**960**

de Pareto, donde la distancia entre soluciones es pequeña, requiriéndose por ello la Etapa III.

Tabla 7.5: Cantidad de evaluaciones necesarias para alcanzar el 95 % del valor del Hipervolumen Dominado de la Frontera de Pareto de referencia - Parte 2

Problema	Medida	NSGA-II	K-MOGA	K-NSGA-II	K-NSGA-II-S3
Hanne3	Percentil 5	1063	*976*	1202	**905**
	Media	1136	*1085*	1382	**993**
	Percentil 95	1504	*1135*	1567	**1076**
Hanne4	Percentil 5	939	*746*	978	**675**
	Media	1076	*940*	1024	**710**
	Percentil 95	1194	*1029*	1130	**885**
Hanne5	Percentil 5	852	**817**	934	*827*
	Media	*968*	986	1056	**956**
	Percentil 95	1143	*1114*	1148	**1089**
Jimenez	Percentil 5	**564**	742	789	*675*
	Media	*767*	825	934	**734**
	Percentil 95	1273	*1087*	1240	**948**
VNT	Percentil 5	**309**	390	643	*376*
	Media	*543*	603	751	**538**
	Percentil 95	*875*	**859**	881	923
ZDT1	Percentil 5	1308	*1204*	1335	**1034**
	Media	1793	*1407*	1593	**1152**
	Percentil 95	2367	*1838*	2071	**1354**
ZDT2	Percentil 5	862	*738*	1154	**704**
	Media	1457	*963*	1265	**835**
	Percentil 95	2054	*1266*	1749	**1176**
ZDT3	Percentil 5	1375	*1070*	1290	**932**
	Media	1865	*1291*	1493	**1049**
	Percentil 95	2459	*1522*	1757	**1168**

K-NSGA-II-S3 funciona muy bien en problemas como *Belegundu* o *ZDT1*, ahorrando cerca del 50 % de las evaluaciones necesarias. Si bien en *Belegundu* el NSGA-II convencional logra un muy buen rendimiento, K-NSGA-II-S3 lo supera. *ZDT1* requiere muchas mas evaluaciones para ambos algoritmos, pero el rendimiento del K-NSGA-II-S3 sigue siendo superior.

Por otro parte, para algunos problemas como *Deb3*, *Hanne5*, *Jimenez*, y *VNT*, K-NSGA-II-S3 requiere de un número de evaluaciones igual o levemente superior al NSGA-II convencional. En estos problemas, la aproximación vía Kriging no resulta satisfactoria. Pero, como punto favorable, dada la forma de actualización del metamodelo en el algoritmo K-NSGA-II-S3, su peor rendimiento en los casos estudiados resulta similar al rendimiento normal del NSGA-II. En otros palabras, la cantidad de evaluaciones necesarias por el NSGA-II es la cota superior de la cantidad de evaluaciones necesarias por el K-NSGA-II-S3.

K-NSGA-II-S3 parece tener un nivel similar de variabilidad aleatoria que el NSGA-II convencional. Las amplitudes de los rangos de percentiles $5\,\% - 95\,\%$ aparentan tener similar tamaño en ambos algoritmos. Es interesante notar que, en los problemas donde K-NSGA-II-S3 tuvo un rendimiento mucho mejor al NSGA-II, los intervalos de los percentiles $5\,\% - 95\,\%$ no se solapan. Es decir, el peor rendimiento del K-NSGA-II-S3 en estos problemas resultó mejor que el mejor rendimiento del NSGA-II convencional.

Las Tablas 7.6 y 7.7 muestran la cantidad de evaluaciones necesarias para alcanzar el mismo nivel en el AEI que el NSGA-II, cuando este último alcanza el 95 % del Hipervolumen Dominado de la Frontera de Pareto. Los resultados informados en las tablas muestran que K-NSGA-II-S3 logra un rendimiento significativamente superior respecto del NSGA-II en 15 de 17 problemas, obteniendo el K-MOGA el segundo mejor resultado en 12 problemas. Más aún, en 2 de los problemas en los que no obtuvo el mejor rendimiento, obtuvo el segundo mejor rendimiento. Es importante resaltar que para los problemas *Fonseca1* y *Fonseca2* el ahorro en las evaluaciones es mayor al 50 % en comparación con el NSGA-II.

Los mismos datos para la GD son mostrados en las Tablas 7.8 y 7.9. Se observa similar rendimiento respecto de los otros dos indicadores ya evaluados. En 15 problemas K-NSGA-II-S3 obtiene el primer lugar respecto de cantidad de evaluaciones necesarias, seguido por el K-MOGA como el segundo mejor.

Respecto de la GS, se pueden hallar en las Tablas 7.10 y 7.11 los resultados obtenidos. Aquí se presentan algunas diferencias, ya que el K-NSGA-II-S3 es algoritmo que necesita el menor número de evaluaciones en 9 de 17 problemas, ocupando el segundo lugar en otros 7 problemas. K-NSGA-II-S3 y K-NSGA-II parecen tener una leve tendencia a generar soluciones mas

Tabla 7.6: Cantidad de evaluaciones necesarias para alcanzar el mismo valor que el NSGA-II para el Indicador Épsilon Aditivo - Parte 1

Problema	Medida	NSGA-II	K-MOGA	K-NSGA-II	K-NSGA-II-S3
Hanne1	Percentil 5	348	*204*	218	**192**
	Media	397	245	*226*	**207**
	Percentil 95	421	293	*287*	**248**
Binh1	Percentil 5	753	*580*	625	**546**
	Media	1520	*681*	814	**651**
	Percentil 95	2474	**763**	975	*808*
Binh2	Percentil 5	691	*548*	554	**502**
	Media	759	*603*	626	**588**
	Percentil 95	854	*691*	703	**624**
Binh3	Percentil 5	1417	*1309*	1554	**1136**
	Media	1752	*1486*	1862	**1378**
	Percentil 95	2003	*1960*	2259	**1937**
Deb3	Percentil 5	**368**	415	507	*395*
	Media	**401**	507	703	*413*
	Percentil 95	*487*	670	878	**481**
Fonseca1	Percentil 5	806	**437**	1047	*478*
	Media	1483	*574*	1378	**515**
	Percentil 95	2521	*807*	2042	**759**
Fonseca2	Percentil 5	742	*591*	1329	**382**
	Media	1461	*854*	1787	**462**
	Percentil 95	1763	*1203*	2177	**703**
Hanne1	Percentil 5	948	*608*	836	**535**
	Media	1104	**623**	745	*642*
	Percentil 95	1572	*871*	1168	**715**
Hanne2	Percentil 5	857	*848*	1045	**801**
	Media	1073	*902*	1231	**865**
	Percentil 95	1298	*1036*	1318	**960**

agrupadas que el resto de los algoritmos evaluados.

Tabla 7.7: Cantidad de evaluaciones necesarias para alcanzar el mismo valor que el NSGA-II para el Indicador Épsilon Aditivo - Parte 2

Problema	Medida	NSGA-II	K-MOGA	K-NSGA-II	K-NSGA-II-S3
Hanne3	Percentil 5	1063	*976*	1202	**905**
	Media	1136	*1085*	1382	**993**
	Percentil 95	1504	*1135*	1567	**1076**
Hanne4	Percentil 5	939	*746*	978	**675**
	Media	1076	*940*	1024	**710**
	Percentil 95	1194	*1029*	1130	**885**
Hanne5	Percentil 5	*852*	**817**	934	877
	Media	**968**	986	1056	**956**
	Percentil 95	1143	*1114*	1178	**1089**
Jimenez	Percentil 5	**564**	742	789	*675*
	Media	*767*	825	934	**734**
	Percentil 95	1273	*1087*	1240	**1048**
VNT	Percentil 5	**309**	390	643	*376*
	Media	*543*	603	751	**538**
	Percentil 95	*875*	**859**	1081	923
ZDT1	Percentil 5	1308	*1234*	1335	**1034**
	Media	1793	*1452*	1593	**1152**
	Percentil 95	2367	*1838*	2071	**1354**
ZDT2	Percentil 5	862	*738*	1154	**704**
	Media	1457	*963*	1265	**835**
	Percentil 95	2054	*1266*	1749	**1176**
ZDT3	Percentil 5	1375	*1070*	1290	**932**
	Media	1865	*1291*	1493	**1049**
	Percentil 95	2459	*1522*	1757	**1168**

Con el fin de tener un indicador general de que tan bueno es cada algoritmo, se calculó la distribución del promedio global de los valores medios obtenidos en cada indicador para cada problema. Para ello, para cada indicador, se remuestreó la, valga la redundancia, muestra de resultados usando *Bootstrap* [Davison and Hinkley, 1997]. Se realizaron 5000 muestreos con

Tabla 7.8: Cantidad de evaluaciones para alcanzar el mismo nivel de Distancia Generacional que el NSGA-II - Parte 1

Problema	Medida	NSGA-II	K-MOGA	K-NSGA-II	K-NSGA-II-S3
Belegundu	Percentil 5	348	*204*	218	**192**
	Media	397	267	*226*	**207**
	Percentil 95	421	306	*287*	**248**
Binh1	Percentil 5	753	625	**536**	*573*
	Media	1520	*681*	799	**630**
	Percentil 95	2474	**763**	948	*825*
Binh2	Percentil 5	691	554	*548*	**502**
	Media	759	*603*	626	**588**
	Percentil 95	854	*691*	703	**624**
Binh3	Percentil 5	1417	*1356*	1539	**1106**
	Media	1752	*1503*	1805	**1308**
	Percentil 95	2003	*1984*	2158	**1897**
Deb3	Percentil 5	**368**	415	507	*395*
	Media	**401**	507	703	*413*
	Percentil 95	*487*	670	878	**481**
Fonseca1	Percentil 5	806	**437**	1047	*482*
	Media	1483	*534*	1378	**515**
	Percentil 95	2521	**726**	2042	*778*
Fonseca1	Percentil 5	742	*591*	1329	**382**
	Media	1461	*854*	1787	**462**
	Percentil 95	1763	*1203*	2177	**703**
Hanne1	Percentil 5	948	*608*	836	**535**
	Media	1104	**623**	973	*635*
	Percentil 95	1572	*871*	1168	**715**
Hanne2	Percentil 5	857	*848*	1045	**801**
	Media	1073	*902*	1231	**865**
	Percentil 95	1298	*1036*	1318	**960**

repetición de los resultados del indicador en cuestión para el conjunto de problemas evaluados. El resultado es la distribución del promedio de cantidad media de evaluaciones necesarias para

Tabla 7.9: Cantidad de evaluaciones para alcanzar el mismo nivel de Distancia Generacional que el NSGA-II - Parte 2

Problema	Medida	NSGA-II	K-MOGA	K-NSGA-II	K-NSGA-II-S3
Hanne3	Percentil 5	1063	*976*	1202	**905**
	Media	1136	*1085*	1382	**993**
	Percentil 95	1504	*1135*	1567	**1076**
Hanne4	Percentil 5	939	*746*	978	**675**
	Media	1076	*940*	1024	**710**
	Percentil 95	1194	*1029*	1130	**885**
Hanne5	Percentil 5	852	**817**	934	*827*
	Media	*968*	984	1056	**956**
	Percentil 95	1143	*1114*	1148	**1089**
Jimenez	Percentil 5	742	**675**	789	*693*
	Media	*767*	825	934	**734**
	Percentil 95	1273	*1087*	1240	**1048**
VNT	Percentil 5	**309**	390	643	*376*
	Media	*543*	603	751	**538**
	Percentil 95	*875*	**859**	881	923
ZDT1	Percentil 5	1308	*1107*	1335	**1034**
	Media	1793	1452	*1357*	**1152**
	Percentil 95	2367	*1612*	2071	**1354**
ZDT2	Percentil 5	862	*746*	1154	**704**
	Media	1457	*855*	952	**835**
	Percentil 95	2054	*1208*	1749	**1176**
ZDT3	Percentil 5	1375	*1132*	1290	**932**
	Media	1865	*1376*	1493	**1049**
	Percentil 95	2459	*1645*	1757	**1168**

resolver los 17 problemas evaluados. O, en otra interpretación, el promedio de las evaluaciones necesarias para resolver un problema que ponderara las dificultades de estos 17 problemas. La distribución obtenida en una distribución sesgada al conjunto de problema testeados, que no sirve para realizar predicciones, pero que si permite comparar en forma gráfica el rendimiento

Tabla 7.10: Cantidad de evaluaciones para alcanzar el mismo nivel de Uniformidad que el NSGA-II - Parte 1

Problema	Medida	NSGA-II	K-MOGA	K-NSGA-II	K-NSGA-II-S3
Belegundu	Percentil 5	348	*204*	232	**197**
	Media	397	*222*	238	**212**
	Percentil 95	421	**258**	300	*263*
Binh1	Percentil 5	753	617	*586*	**543**
	Media	1520	**638**	809	*644*
	Percentil 95	2474	**782**	963	*789*
Binh2	Percentil 5	691	**513**	558	*517*
	Media	759	**595**	637	*606*
	Percentil 95	854	**629**	710	*633*
Binh3	Percentil 5	1417	*1144*	1579	**1130**
	Media	1752	**1336**	1831	*1347*
	Percentil 95	2003	**1920**	2185	*1932*
Deb3	Percentil 5	**368**	*400*	520	407
	Media	**401**	*424*	717	471
	Percentil 95	**487**	*496*	884	519
Fonseca1	Percentil 5	806	*497*	1061	**484**
	Media	1483	**529**	1387	*570*
	Percentil 95	2521	*812*	2052	**771**
Fonseca2	Percentil 5	742	**390**	1335	*403*
	Media	1461	**471**	1802	*489*
	Percentil 95	1763	*754*	2185	**708**
Hanne1	Percentil 5	948	650	842	547
	Media	1104	*732*	979	**638**
	Percentil 95	1572	*921*	1177	**725**
Hanne2	Percentil 5	857	*824*	1054	**813**
	Media	1073	*878*	1237	**820**
	Percentil 95	1298	*1071*	1324	**967**

de cada algoritmo para un conjunto fijo de problemas.

Tabla 7.11: Cantidad de evaluaciones para alcanzar el mismo nivel de Uniformidad que el NSGA-II - Parte 2

Problema	Medida	*NSGA-II*	*K-MOGA*	*K-NSGA-II*	K-NSGA-II-S3
Hanne3	Percentil 5	1063	*945*	1219	**909**
	Media	1136	*1061*	1410	**1030**
	Percentil 95	1504	*1183*	1604	**1097**
Hanne4	Percentil 5	939	*716*	1020	**693**
	Media	1076	*798*	1060	**741**
	Percentil 95	1194	*967*	1148	**904**
Hanne5	Percentil 5	*852*	860	963	**838**
	Media	*968*	990	1077	**927**
	Percentil 95	*1143*	1152	1184	**1131**
Jimenez	Percentil 5	**564**	*706*	812	742
	Media	*767*	794	956	**762**
	Percentil 95	1273	*1093*	1280	**1070**
VNT	Percentil 5	**309**	*395*	668	408
	Media	**543**	569	791	*567*
	Percentil 95	*875*	**789**	951	914
ZDT1	Percentil 5	1308	**1063**	1371	*1085*
	Media	1793	*1285*	1620	**1140**
	Percentil 95	2367	*1481*	2097	**1390**
ZDT2	Percentil 5	862	**726**	1187	*735*
	Media	1457	**857**	1293	*964*
	Percentil 95	2054	*1307*	1774	**1217**
ZDT3	Percentil 5	1375	*1060*	1311	**956**
	Media	1865	*1276*	1513	**1069**
	Percentil 95	2459	*1500*	1783	**1189**

En las Figuras 7.1a, 7.1b, 7.1c, y 7.1d los histogramas de color gris claro y la caja del gráfico de cuartiles nomenclada como *KN* representan al algoritmo K-NSGA-II-S3, mientras que el color gris y la nomenclatura *KM* representan al algoritmo K-MOGA. El color gris oscuro en los histograma denota superposición entre los dos algoritmos. El eje horizontal indica el cociente

entre el número medio de evaluaciones del algoritmo analizado y el del NSGA-II. El valor 1 en dicho eje se corresponde con el número medio de evaluaciones necesarias por el NSGA-II. Los histogramas muestran la frecuencia de cada valor y los gráficos de caja su distribución.

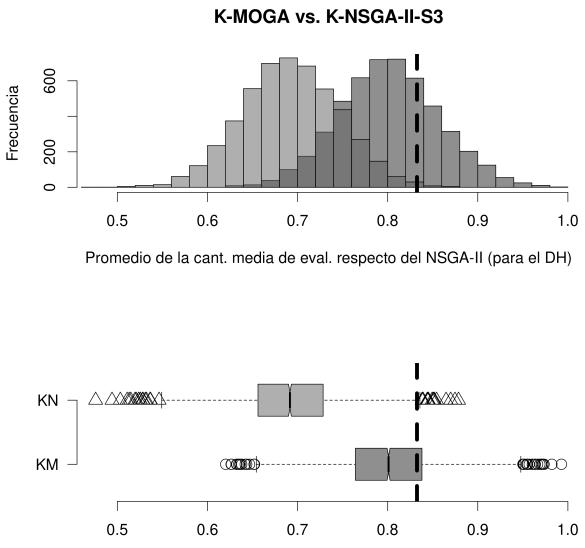
La Figura 7.1a muestra la comparativa entre el K-NSGA-II-S3 y el K-MOGA para el DH. El rendimiento medio de ambos algoritmos es mejor que la del NSGA-II, pero el K-NSGA-II-S3 obtiene mejores valores medios que el K-MOGA. Se puede observar también como, aproximadamente, el 75 % de la cantidad media de evaluaciones requeridas por el K-NSGA-II-S3 es mas chica que el 75 % de todos los remuestreos del NSGA-II (de hecho, solo para el caso de *Deb3*, K-NSGA-II-S3 tiene un peor rendimiento). Las Figuras 7.1b y 7.1c muestran que tanto el AEI como la GD tienen un comportamiento similar al DH. Pero, para el caso del GS, se observa en la Figura 7.1d, que el rendimiento del K-NSGA-II-S3 y el K-MOGA no presentan diferencias significativas, evidenciando la posible existencia de algún sesgo hacia el agrupamiento de soluciones.

En lineas generales, aunque la GS aparenta tener un comportamiento desacoplado del resto de los indicadores para el K-NSGA-II-S3, se observa que, para una meta fija en cada indicador, K-NSGA-II-S3 realiza una cantidad de evaluaciones menor que el NSGA-II. Más aún, de la información mostrada en las tablas se puede inferir que el algoritmo parece funcionar mejor (genera más ahorros porcentuales) cuando más evaluaciones necesita realizar el NSGA-II. Para chequear este hipótesis, se analizó la correlación entre las dos siguientes variables:

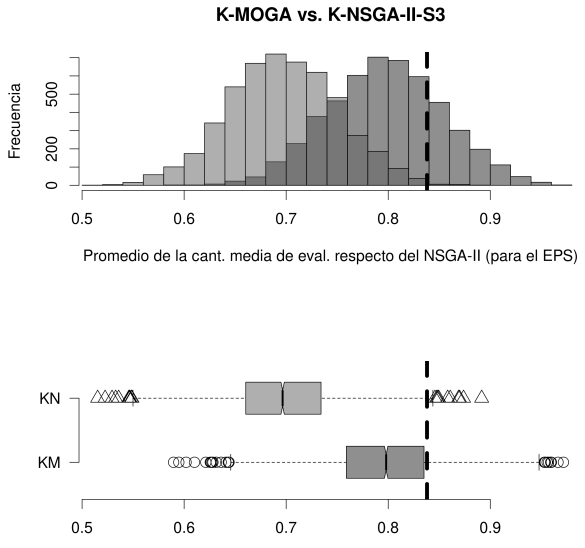
- $niter_{nsga-ii} = cantidad\ de\ evaluaciones\ realizadas\ por\ el\ NSGA\text{-}II$
- $niter_{k-nsga-ii-s3} = \dfrac{cantidad\ de\ evaluaciones\ realizadas\ por\ el\ K\text{-}NSGA\text{-}II\text{-}S3}{niter_{nsga-ii}}$

El cálculo de la correlación obtiene un valor de $-0{,}55$ para el caso del DH, $-0{,}52$ para el AEI, $-0{,}54$ para la GD y para la GS un valor de $-0{,}52$. Si bien no hay una fuerte correlación, se puede ver que hay una cierta influencia del esfuerzo necesario por el NSGA-II respecto de la magnitud de los ahorros conseguidos.

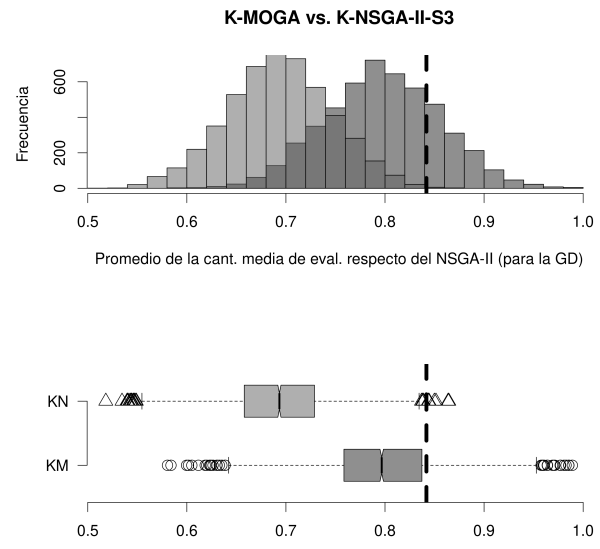
Por último, es interesante resaltar que K-NSGA-II-S3 logra reducciones muy significativas en la cantidad de evaluaciones necesarias para los problemas *ZDT1*, *ZDT2*, y *ZDT3*. A pesar que



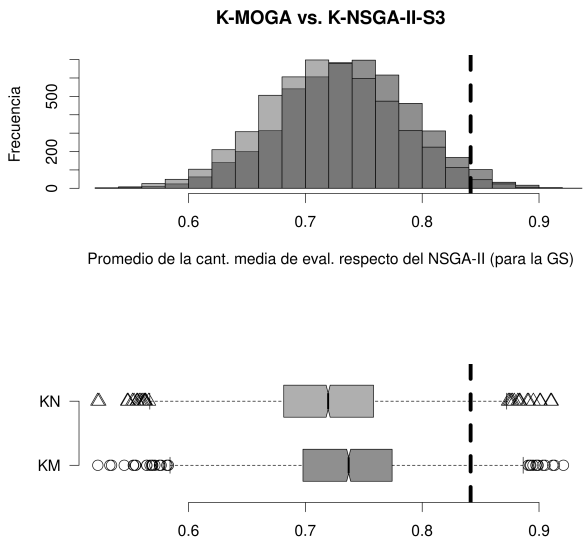
(a) Hipervolumen Dominado



(b) Indicador Épsilon Aditivo



(c) Distancia Generacional



(d) Uniformidad

Figura 7.1: Comparativas del promedio de la cantidad media de evaluaciones para obtener el mismo nivel de calidad que el NSGA-II en cada uno de las 4 métricas seleccionadas.

estos problemas tiene un dominio de 30 dimensiones, la aproximación de las funciones objetivo vía Kriging no parece tener mayores problemas.

7.1.5. Análisis del impacto del Elitismo Negativo

Para medir el efecto del uso del Elitismo Negativo dentro del esquema de optimización del K-NSGA-II, se ejecutó el algoritmo durante 50 generaciones para cada problema, con y sin el uso de dicho elitismo. La Tabla 7.12 muestra la media de la diferencia porcentual en el Hipervolumen Dominada para el caso de no usar el Elitismo Negativo respecto del caso de usarlo. Los resultados son dispares. En ciertos problemas, como en *Fonseca1* el uso del Elitismo Negativo es contraproducente. Sin embargo, en el problema *ZDT1*, su uso mejora en forma significativa los resultados. En ambos casos, el uso del Elitismo Negativo aumenta la cantidad de evaluaciones necesarias. Como corolario del análisis, si bien el resultado es dispar, los incrementos en la calidad de las estimaciones son de mayor magnitud que los decrementos, por lo cual, dado que en un problema real no se conocerá su impacto a priori, sería recomendable su uso.

7.1.6. Análisis de los nuevos parámetros

Para determinar como los nuevos parámetros introducidos por el K-NSGA-II impactan en el proceso de optimización, se realizaron repeticiones de los experimentos para cada problema variando el rango de valores de los nuevos parámetros. Los valores de los parámetros utilizados se muestran en la Tabla 7.13 y los resultados obtenidos en las Tablas 7.14 y 7.15.

El valor de k_{check} tiene a priori un efecto directo sobre el número de evaluaciones, ya que fija la cantidad de evaluaciones que son realizadas, como mínimo, en cada generación. Pero su impacto real algo mas complejo. Reducir demasiado el valor de k_{check}, si bien fija una cantidad mínima menor de evaluaciones, incrementa la cantidad de generaciones que el algoritmo debe de ser ejecutado, debido a que el metamodelo tiende a poseer una menor calidad (debido a la menor frecuencia de actualizaciones). Para un nivel de calidad fijo, el número de ejecuciones termina aumentando. Por otro lado si k_{check} es incrementado, el efecto inmediato es un aumento en el número de ejecuciones, sin una mejora proporcional en la calidad del algoritmo. Variar el valor de α tiene un efecto similar al caso de k_{check}. Valores mayores de α implican mas recálculos del metamodelo, y valores menores implican mas generaciones debido a la menor calidad del modelo. Configurar a priori los valores de k_{check} y α puede ser complicado, pero se puede utilizar

Tabla 7.12: Porcentaje medio de variación en el Hipervolumen Dominada cuando el Elitismo Negativo no es utilizado.

Problema	% Variación en Hipervolumen
Belegundu	2 %
Binh1	−4 %
Binh2	0 %
Binh3	2 %
Deb3	−3 %
Fonseca1	7 %
Fonseca2	12 %
Hanne1	1 %
Hanne2	3 %
Hanne3	−1 %
Hanne4	−2 %
Hanne5	1 %
Jimenez	3 %
VNT	−8 %
ZDT1	−23 %
ZDT2	−9 %
ZDT3	−15 %

Tabla 7.13: Parámetros

Parámetro	Valor
k_{check}	$[5; 10; 20]$
α	$[0{,}05; 0{,}10; 0{,}20]$
n_{ea}	$[0{,}05; 0{,}10; 0{,}20]$

una metodología como la *iRace* [Ibanez et al., 2016] o similar para ajustar estos parámetros.

Tabla 7.14: Número de evaluaciones necesarias para alcanzar el 95 % del Hipervolumen Dominado para diferentes valores de k_{check}.

Problema	$k_{check} = 5$	$k_{check} = 10$	$k_{check} = 15$
Belegundu	237	207	312
Binh1	727	651	729
Binh3	1428	1378	1484
Deb3	512	413	503
Fonseca1	606	515	616
Fonseca2	534	462	529
Hanne1	645	623	729
Hanne2	952	847	923
Hanne3	1118	1012	1093
Hanne4	742	724	776
Hanne5	1050	968	1183
Jimenez	795	744	898
VNT	704	562	722
ZDT1	1400	1302	1521
ZDT2	960	846	1025
ZDT3	1276	1121	1274

7.2. VSL-NSGA-II

En esta sección se muestran los experimentos realizados respecto del algoritmo VSL-NSGA-II, descripto en la Sección 6.3. El objetivo de estos experimentos es determinar si este algoritmo logra ahorros significativos en el tiempo total simulado para un nivel predeterminado de calidad.

7.2.1. Problemas evaluados

Con el fin de evaluar el funcionamiento del algoritmo propuesto, se utilizaron un conjunto de problemas de la literatura que son usualmente empleados para evaluar el rendimiento de algoritmos de optimización multiobjetivo [Deb et al., 2002]. Estos problemas son los mismos

Tabla 7.15: Número de evaluaciones necesarias para alcanzar el 95 % del Hipervolumen Dominado para diferentes valores de α.

Problema	$\alpha = 0{,}8$	$\alpha = 0{,}90$	$\alpha = 0{,}95$
Belegundu	263	207	356
Binh1	690	651	1034
Binh2	643	588	723
Binh3	1429	1378	1682
Deb3	491	413	492
Fonseca1	482	515	761
Fonseca2	430	462	608
Hanne1	640	623	735
Hanne2	859	847	1009
Hanne3	1062	1012	1176
Hanne4	703	724	861
Hanne5	1005	968	1247
Jimenez	841	744	1089
VNT	635	562	958
ZDT1	1379	1302	158
ZDT2	889	846	1249
ZDT3	1232	1121	1426

utilizados para evaluar al K-NSGA-II en la Sección 7.1.1 Las Tablas 7.1 y 7.2 muestran, para cada problema, las funciones objetivo, las cotas de las variables de decisión y las restricciones.

Debido a que el algoritmo analizado en esta sección funciona reduciendo la longitud de las simulaciones en un esquema de OvS, no se pueden usar las funciones anteriores en forma directa, sino que fue necesario realizar un artificio para *simular* el avance del reloj de simulación. El artificio usado fue el siguiente:

1. Se consideró una longitud de simulación máxima de 100, formada por 100 intervalos de tamaño 1.

2. Se asumió que en el tiempo 100, la función evaluada alcanza su valor.

3. Se calculó una tasa de crecimiento constante, igual al valor de la función dividido entre 100, la cual genera una tendencia creciente lineal en los valores de la función respecto del tiempo.

4. En cada intervalo, se generó un ruido aleatorio uniforme comprendido en el rango [−100 %, +100 %] respecto del valor anteriormente determinado para la tasa de crecimiento.

5. Finalmente, dado que por el ruido aleatorio la función no obtiene el valor objetivo al tiempo 100, se realiza un ajuste a posteriori para escalar los datos de manera tal que el ultimo de la serie coincida con el valor esperado.

El valor alcanzado por la función objetivo en el tiempo de simulación t (sin el ajuste a posteriori) se puede calcular mediante la Ecuación 7.1. Conociendo el valor esperado y el obtenido al tiempo $t = 100$, la fórmula final para el cálculo del valor de la función al tiempo t se muestra en la Ecuación 7.1. Esta ecuación, junto con el proceso de generación descripto anteriormente, permite representar valores de objetivos que se acumulen en el tiempo (como cantidad de vehículos que llegan a destino, producción al final del día, etc.).

$$f(\vec{x}, t) = \sum_{i=0}^{t} (tasaCrecimiento(\vec{x}) + ruido(i)) \tag{7.1}$$

$$f(\vec{x}, t)^* = \frac{f_{esperado}(\vec{x}, t = 100)}{f(\vec{x}, t = 100)} \cdot f(\vec{x}, t) \tag{7.2}$$

7.2.2. Métricas utilizadas

Con el fin de evaluar el rendimiento del VSL-NSGA-II, se utilizaron dos tipos de métricas: métricas de calidad y de costo. Para las primeras, se utilizó el DH, previamente descripto en la Sección 2.5. Para las métricas de costo, se eligió el tiempo total simulado. Esta métrica consiste en la suma del tiempo simulado para cada una de las evaluaciones realizadas por el algoritmo.

7.2.3. Configuraciones utilizadas

Para poder evaluar la calidad del VSL-NSGA-II, se comparó su rendimiento con la metaheurística NSGA-II. Ambos algoritmos se ejecutaron utilizando un criterio de detención que consiste en finalizar la optimización cuando el DH de la generación actual es mayor o igual al 95 % de la Frontera de Pareto de referencia. Esto permite comparar todos los algoritmos en las mismas condiciones. La configuración de los algoritmos fue la misma que la usada en la Sección 7.1.3 (ver Tabla 7.3). Para el VSL-NSGA-II, la cantidad de individuos muestreados es de 10 (el 20 %) de la población y se exigen 5 avances de reloj consecutivos con el mismo orden relativo entre soluciones, con un nivel α del 95 %. Además, el criterio de optimalidad usado fue la optimalidad de Pareto. Debido al hecho de que los dos algoritmos son algoritmos pseudoaleatorios, cada ejecución se repitió 100 veces, con diferentes semillas aleatorias. Los valores de los parámetros surgieron de experimentos preliminares, donde se probaron distintas combinaciones de valores de estos parámetros.

7.2.4. Comparación del rendimiento

A continuación, se muestran los resultados de la comparación entre el NSGA-II y el VSL-NSGA-II. Las Tablas 7.16 y 7.17 muestran el tiempo total simulado para cada problema tratado. La primer columna indica el problema, la segunda la medición realizada (percentil 5 %, media y percentil 95) y las dos siguientes el tiempo total simulado para el NSGA-II y el VSL-NSGA-II respectivamente.

Se observa que el algoritmo VSL-NSGA-II supera consistentemente al NSGA-II en el tiempo total simulado, logrando una reducción de costos de entre el 20 % y el 30 % para la mayoría de los problemas. La variabilidad en los resultados aparenta ser similar, debido al hecho que el ahorro en las tres medidas utilizadas es similar en casi todos los problemas estudiados. Este ahorro en tiempo total simulado debería conllevar un ahorro (no estudiado en esta tesis) de los requerimientos de memoria, ya que al realizar simulaciones mas cortas, el uso total de memoria debería reducirse.

Dado lo uniforme de los resultados, no se observa dependencia del esfuerzo requerido por el

Tabla 7.16: Tiempo total simulado para alcanzar el 95 % del valor del Hipervolumen Dominado de la Frontera de Pareto de referencia - Parte 1

Problema	Medida	NSGA-II	VSL-NSGA-II
Belegundu	Percentil 5	34800	27876
	Media	39700	31835
	Percentil 95	42100	37804
Binh1	Percentil 5	75300	56486
	Media	152000	106432
	Percentil 95	247400	185604
Binh2	Percentil 5	69100	44936
	Media	75900	57770
	Percentil 95	85400	64915
Binh3	Percentil 5	141700	99214
	Media	175200	119166
	Percentil 95	200300	146243
Deb3	Percentil 5	36800	25402
	Media	40100	29695
	Percentil 95	48700	35565
Fonseca1	Percentil 5	80600	64515
	Media	148300	99416
	Percentil 95	252100	181558
Fonseca2	Percentil 5	74200	51957
	Media	146100	105205
	Percentil 95	176300	130481

NSGA-II ni del tipo de problema analizado. El rendimiento del VSL-NSGA-II parece estar solamente relacionada a la evolución temporal de la función a optimizar.

Pese a lo promisorios de los resultados, cabe aclarar que se estudio el algoritmo con un subconjunto pequeño de los problemas tratables por OvS, puntualmente aquellos en los cuales los objetivos bajo interés presentan un comportamiento acumulativo de tendencia lineal. El VSL-NSGA-II debería presentar similares resultados para cualquier objetivo acumulativo con tendencia creciente, no solo lineal, pero no fue evaluado en esas condiciones. Puntualmente, un

Tabla 7.17: Tiempo total simulado para alcanzar el 95 % del valor del Hipervolumen Dominado de la Frontera de Pareto de referencia - Parte 2

Problema	Medida	NSGA-II	VSL-NSGA-II
Hanne1	Percentil 5	94800	67355
	Media	110400	72889
	Percentil 95	157200	119490
Hanne2	Percentil 5	85700	58330
	Media	107300	69786
	Percentil 95	129800	84385
Hanne3	Percentil 5	106300	77631
	Media	113600	81840
	Percentil 95	150400	106796
Hanne4	Percentil 5	93900	74200
	Media	107600	81215
	Percentil 95	119400	104372
Hanne5	Percentil 5	85200	63094
	Media	96800	73603
	Percentil 95	114300	88037
Jimenez	Percentil 5	56400	43477
	Media	76700	56001
	Percentil 95	127300	95521
VNT	Percentil 5	30900	20131
	Media	54300	35855
	Percentil 95	87500	65679
ZDT1	Percentil 5	130800	94216
	Media	179300	143456
	Percentil 95	236700	182286
ZDT2	Percentil 5	86200	63805
	Media	145700	107835
	Percentil 95	205400	135605
ZDT3	Percentil 5	137500	107674
	Media	186500	150560
	Percentil 95	245900	191850

tipo de problemas para los cuales se cumplen los supuestos bajo los cuales se evaluó el algoritmo son aquellos que se pueden modelar mediante *Dinámica de Sistemas* (SD) y en los que el objetivo de interés se corresponda con una variable de stock con variables de flujo asociadas lineales o constantes (determinísticas o estocásticas) [Forrester, 1961][1].

7.2.5. Impacto de la no-linealidad respecto del tiempo

Como se mencionó en la anterior sección, el algoritmo diseñado es muy sensible respecto de la linealidad o no respecto del tiempo de los objetivos evaluados. La Figuras 7.2 , 7.3 y 7.4 muestran la distribución del tiempo de simulación necesario para detectar el orden relativo en las soluciones en el problema *ZDT3*, cuando la variabilidad respecto de la tasa de crecimiento es $+/-50\,\%$, $+/-100\,\%$ y $+/-200\,\%$, respectivamente. Se puede observar claramente que, mientras mas no-lineal es la función respecto del tiempo, mas tiempo de simulación se debe ejecutar. Para construir estas gráficas se compararon 100 grupos de 100 soluciones generadas al azar, calculándose el tiempo de simulación que permite detectar el $95\,\%$ del orden relativo en cada uno de los grupos. Para el primer caso, la media del tiempo de simulación es 20, para el segundo 50 y para el tercero 73.

7.3. S-NSGA-II

En esta sección se muestran los experimentos realizados respecto del algoritmo S-NSGA-II, descripto en la Sección 6.4. El objetivo de estos experimentos es determinar si este algoritmo logra mejoras en la calidad de las soluciones obtenidas respecto a escenarios reales.

7.3.1. Problemas evaluados

Con el fin de evaluar el funcionamiento del algoritmo propuesto, se utilizaron un conjunto de problemas de la literatura que son usualmente empleados para evaluar el rendimiento de

[1]Sistemas de tráfico simplificados y a escala agregada pueden ser simulados mediante esta metodología, la cual califica como simulación macroscópica.

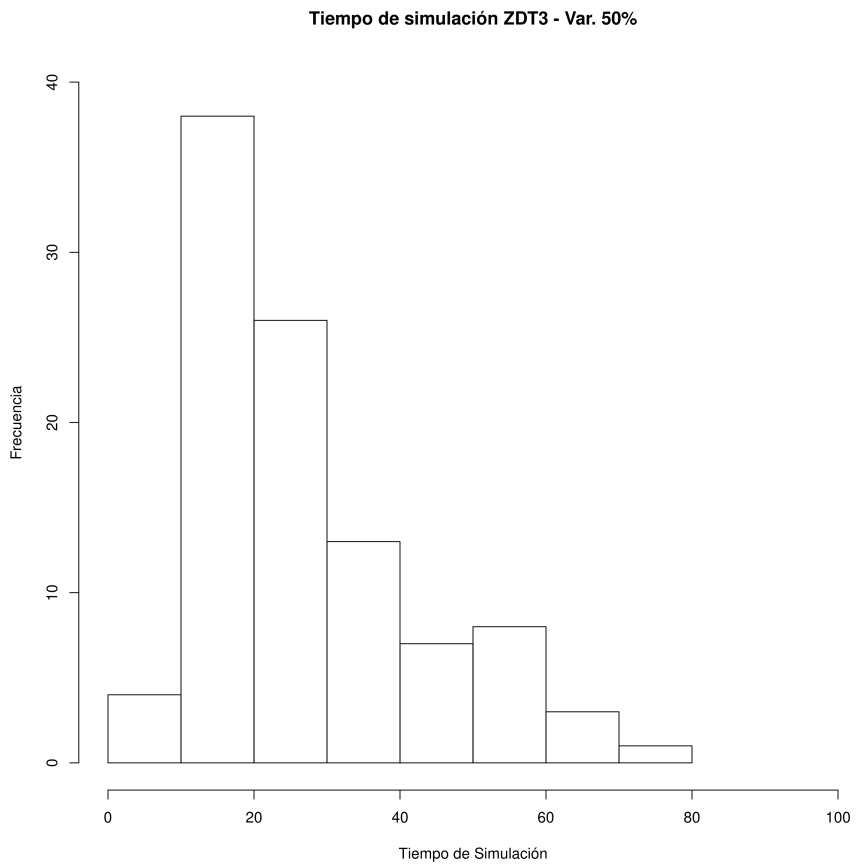


Figura 7.2: Tiempo de simulación medio necesario para el ZDT3 con $+/-50\,\%$ de variabilidad respecto a la tasa

algoritmos de optimización multiobjetivo [Deb et al., 2002], agregándoles ruido aleatorio. Se utilizó un subconjunto de los mismos problemas utilizados para evaluar al K-NSGA-II en la Sección 7.1.1. Puntualmente, se seleccionaron los problemas *Belegundu*, *ZDT1*, *ZDT2* y *ZDT3*. Las Tablas 7.1 y 7.2 muestran, para cada problema, las funciones objetivo, las cotas de las variables de decisión y las restricciones.

A fin de agregar ruido aleatorio, cada uno de los objetivos a evaluar de cada problema se reemplazaron por la Ecuación 7.3, en la cual, para el objetivo i, fa_i es la versión aleatoria de la función evaluada, f_i es la función determinística original y u es una variable aleatoria perteneciente a una distribución uniforme continua $U(a, b)$ acotada por los valores a y b. Los parámetros de la distribución uniforme se seleccionaron de manera tal de evaluar tres escenarios

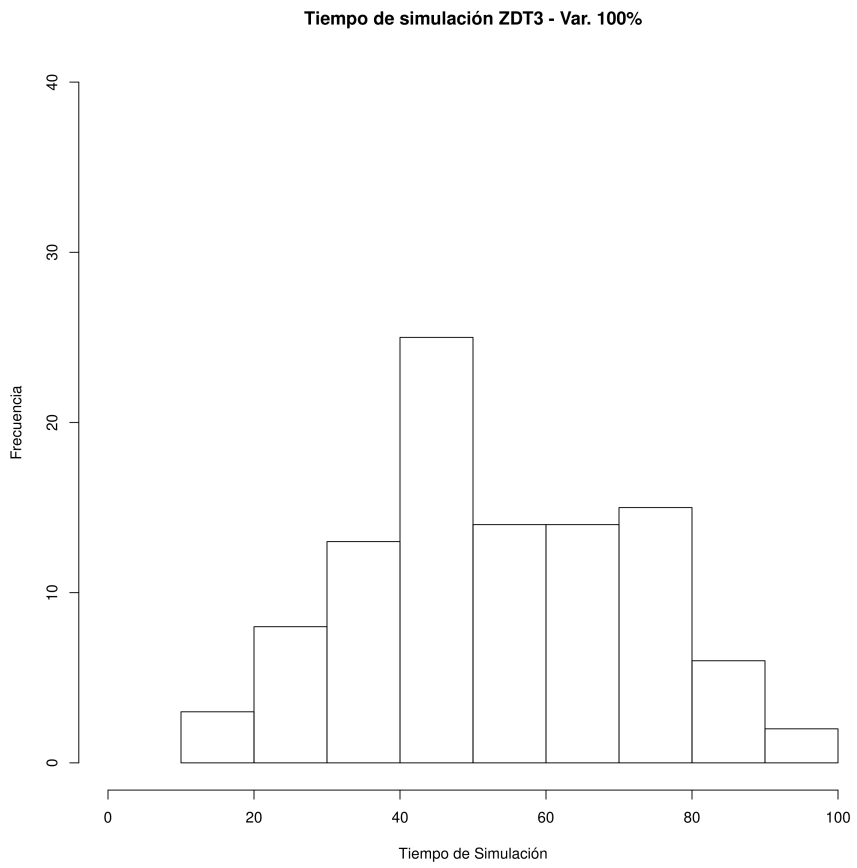


Figura 7.3: Tiempo de simulación medio necesario para el ZDT3 con $+/-100\,\%$ de variabilidad respecto a la tasa

de aleatoriedad: $(-0{,}5; 0{,}5)$, $(-1; 1)$ y $(-2; 2)$.

$$fa_i(\vec{x}) = f(\vec{x}) \cdot u \tag{7.3}$$

Para cada problema y escenario, u se muestreó 130 veces para cada solución de la frontera de referencia, utilizándose 30 muestras como escenarios de entrenamiento, y las 100 restantes fungiendo como escenarios de test.

A fin de validar los resultados, el S-NSGA-II se comparó con el NSGA-II (usando este último el criterio de Optimalidad de Pareto de la esperanza de cada objetivo[2]).

[2]Es decir, la Frontera de Pareto de los valores medios de los objetivos en los escenarios evaluados, ver Sección 2.6

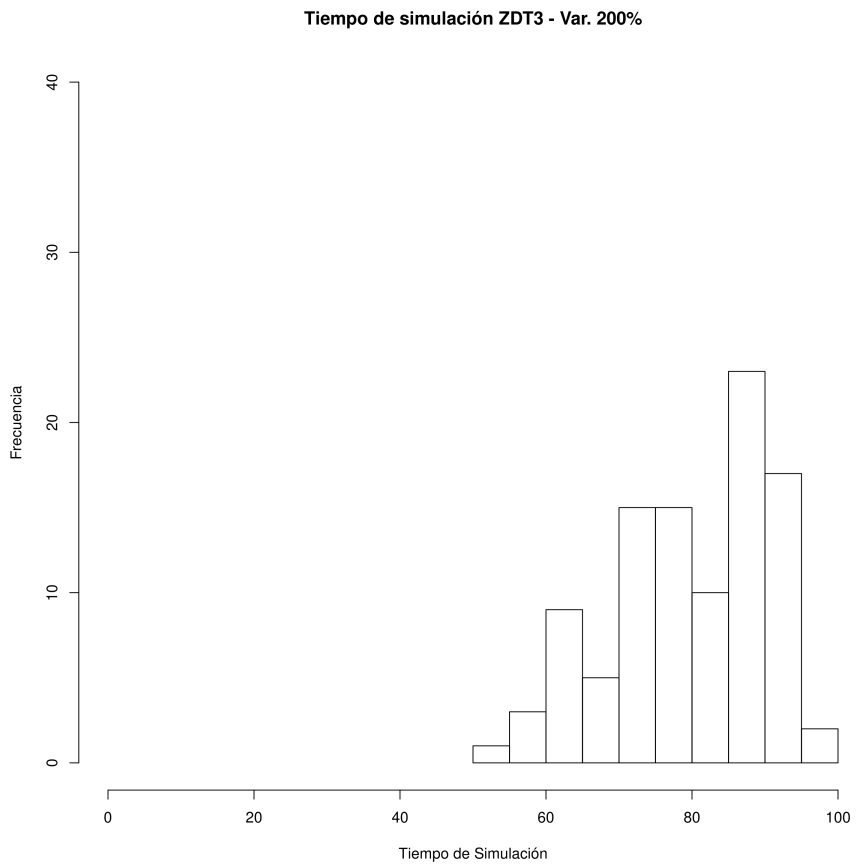


Figura 7.4: Tiempo de simulación medio necesario para el ZDT3 con $+/-200\,\%$ de variabilidad respecto a la tasa

7.3.2. Métricas utilizadas

Con el fin de evaluar el rendimiento del S-NSGA-II, se utilizó como métrica el porcentaje del hipervolumen dominado por el S-NSGA-II respecto del hipervolumen dominado por el NSGA-II, según lo indicado en la Ecuación 7.4.

$$\%DH = \frac{DH_{S-NSGA-II} - DH_{NSGA-II}}{DH_{NSGA-II}} \tag{7.4}$$

Donde DH_x es el hipervolumen dominado por el método x.

7.3.3. Configuraciones utilizadas

Para poder evaluar la calidad del S-NSGA-II, se comparó su rendimiento con la metaheurística NSGA-II. Ambos algoritmos se ejecutaron utilizando un criterio de detención fijo, es decir, una

Tabla 7.18: Parámetros

Parámetro	Valor
Cantidad de Generaciones	20
Tamaño de la población	50
Tamaño del torneo	2
Probabilidad de cruzamiento	0.8
Probabilidad de mutación	0.1
Tasa de elitismo	0.1

número fijo de generaciones. La configuración utilizada se muestra en la Tabla 7.18. Para el S-NSGA-II, además, se utilizaron dos distancias d a la Frontera de Pareto (0, y 4), con un valor de p igual a 75. Los valores de los parámetros surgieron de experimentos preliminares, donde se probaron distintas combinaciones de valores de estos parámetros.

7.3.4. Comparación de los resultados

A continuación, se analizan los resultados de la comparación entre el enfoque convencional vía NSGA-II y el S-NSGA-II. Utilizando los 30 escenarios de entrenamiento generados para cada combinación de problema y configuración de la función aleatoria uniforme, se ejecutó el NSGA-II (promediando las evaluaciones de cada objetivo) y al S-NSGA-II con distancias 1 y 4. Luego, se evaluaron las soluciones respecto de los 100 problemas de test, calculándose para cada uno de ellos el indicador seleccionado. Por último, se calcularon las medianas de los valores encontrados. Los resultados se pueden observar en las Tablas 7.19, 7.20 y 7.21

S-NSGA-II consigue alcanzar un mejor valor en el indicador DH que la versión del NSGA-II utilizando la metodología basada en medias para la obtención de la Frontera de Pareto estocástica. El parámetro d no parece tener una gran influencia en los resultados, ya que con $d = 1$ o $d = 4$ los resultados son similares. Comparando *Belegundu* con el resto de las funciones, se observa que las características de la función objetivo a optimizar afectan significativamente la calidad de los algoritmos utilizados. No se observa el nivel de mejoras en *Belegundu* que el observado en las

Tabla 7.19: Valor promedio de la mediana del indicador de comparación de Hipervolúmenes Dominados. Uniforme $(-0{,}5; 0{,}5)$

Problema	Mediana Promedio ($d = 1$)	Mediana Promedio ($d = 4$)
Belegundu	0,93	1,03
ZDT1	1,21	1,25
ZDT2	1,12	1,13
ZDT3	1,39	1,41

Tabla 7.20: Valor promedio de la mediana del indicador de comparación de Hipervolúmenes Dominados. Uniforme $(-1; 1)$

Problema	Mediana Promedio ($d = 1$)	Mediana Promedio ($d = 4$)
Belegundu	1,44	1,53
ZDT1	7,57	8,41
ZDT2	2,70	2,89
ZDT3	9,94	11,16

tres funciones de la familia *ZDT*. De hecho, en el escenario con baja aleatoriedad, S-NSGA-II tiene la misma (o levemente inferior) performance que el NSGA-II.

En todos los casos, se observa como S-NSGA-II funciona mejor a medida que mayor es la aleatoriedad, lo cual era el resultado esperado dado el funcionamiento del algoritmo. Las diferencias con el método basado en promediar objetivos se incrementan a medida que los parámetros de la distribución uniforme permiten mayor variabilidad en los resultados de las evaluaciones.

7.4. Problema de Asignación Origen-Destino del Tráfico (ODTAP)

En esta sección se muestran los experimentos realizados respecto del algoritmo de resolución del (ODTAP), descripto en la Sección 5.3. Para este problema, el objetivo fue evaluar el desempeño del algoritmo propuesto en distintas configuraciones de red. Debido a la falta de datos para

Tabla 7.21: Valor promedio de la mediana del indicador de comparación de Hipervolúmenes Dominados. Uniforme $(-2; 2)$

Problema	Mediana Promedio ($d = 1$)	Mediana Promedio ($d = 4$)
Belegundu	6,74	7,61
ZDT1	17,93	18,37
ZDT2	11,73	12,82
ZDT3	22,09	24,26

benchmarks en la bibliografía, se construyó un conjunto de escenarios, se los optimizó 50 veces a cada uno con el algoritmo desarrollado (utilizando un tamaño de población de 250 individuos), se consolidaron todas las soluciones halladas y se calculó la Frontera de Pareto sobre este conjunto. Esta última frontera se utilizó luego a modo de patrón para mensurar la bondad del algoritmo de optimización. La metaheurística evaluada fue la SMPSO. Como el algoritmo es pseudo-aleatorio, se ejecutó el mismo 10 veces por escenario.

La Sección 7.4.1 detalla los problemas de prueba, la Sección 7.4.2 enuncia las métricas utilizadas, en la Sección 7.4.3 se muestran las configuraciones de todos los algoritmos y en la Sección 7.4.4 se muestran todas las comparaciones y el análisis de los resultados.

7.4.1. Problemas evaluados

Con el fin de evaluar el esquema general y la metaheurística testeada, fue necesario generar un conjunto de escenarios sobre los cuales ejecutar el procedimiento de optimización. La construcción de escenarios para este problema tiene tres partes diferenciadas:

- Construcción de la red de tráfico.
- Construcción de la demanda actual de tráfico.
- Determinación del incremento en la demanda.

Las dos primeras partes se explican a continuación. La tercera consiste en la definición del valor porcentual del incremento.

Se utilizó como base tres redes de topología grilla, de tamaño 10×10, 25×50 y 75×50. Estas redes fueron modificadas aleatoriamente para obtener 3 nuevas redes por cada una, dando lugar a 9 redes de tráfico. Para cada una de estas 9 redes, se definieron 3 escenarios de demanda, quedando así con 27 combinaciones de red y demanda. Por último, para cada combinación, se definieron dos posibles incrementos, 5 % y 15 %, dando lugar a 54 escenarios.

Construcción de la red de tráfico

Para la construcción de las redes de tráfico, se siguió un procedimiento consistente en generar una red de topología grilla de nxm nodos de lados e, iterativamente, eliminar en forma aleatoria nodos (y los arcos asociados) o arcos. Para generar las redes, se utilizó la aplicación del paquete SUMO denominada NETGENERATE, a través RSUMO. Un esquema del procedimiento se puede observar en la Figura 7.5. La imagen A muestra la red tipo grilla cuadrada original. En la imagen B se ve el efecto de eliminar un arco de la grilla original. Por último, la imagen C muestra el efecto de eliminar un nodo y sus arcos asociados.

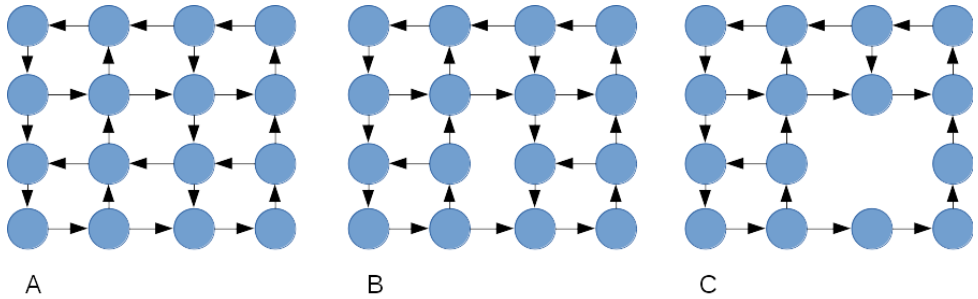


Figura 7.5: A: Red tipo grilla cuadrada inicial. B: eliminación de un arco en A. C: eliminación de un nodo en A.

El número total de cambios realizados a la red se fijó de acuerdo a la Ecuación 7.5, donde m y n son la cantidad de nodos de lado que definen el tamaño de la red tipo grilla. Para cada cambio a realizar, la probabilidad de eliminar nodos o arcos se fijó en 50 % para cada una de las dos alternativas, y la selección del nodo o arco a eliminar se realizó mediante una distribución

de probabilidad uniforme discreta.

$$numerodecambios = \sqrt{\text{mín}(m, n)} \tag{7.5}$$

Una vez definida la red, se seleccionaron aleatoriamente el 10 % de los nodos totales, asignándose como nodos de origen y destino. El resto de los nodos se definió como de transbordo.

Construcción de la demanda actual de tráfico

Creadas las redes de tráfico, es necesario definir la demanda de tráfico asociada a ellas. Para una red particular, la cantidad de vehículos circulando por hora se define seleccionando un valor aleatorio con probabilidad uniforme en el rango $[\text{máx}(m, n); 10 \cdot \text{máx}(m, n)]$. Definida la cantidad de vehículos totales, para cada uno es asignado un nodo origen y un nodo destino, seleccionado aleatoriamente de la lista de nodos.

7.4.2. Métricas utilizadas

Con el fin de evaluar el rendimiento de cada algoritmo, se utilizaron las siguientes métricas, previamente descriptas en la Sección 2.5:

- Distancia Generacional (GD)
- Uniformidad (GS)

El interés al desarrollar este algoritmo estuvo puesto en la capacidad del mismo para generar muchas alternativas diferentes con el mismo trade-off entre si (es decir, que todas fueran igual de óptimas).

7.4.3. Configuración del algoritmo

Los parámetros bajo los cuales se ejecutó el SMPSO son los mostrados en la Tabla 7.22. Por otra parte, el tiempo de simulación para evaluar a cada solución se fijó en 1,200 segundos. Estos

Tabla 7.22: Parámetros.

Parámetro	Valor
Cantidad de iteraciones	100
Tamaño de la población	100
Tamaño de la lista de líderes	10
Probabilidad de mutación	0.1

valores surgieron de experimentos preliminares, donde se probaron distintas combinaciones de valores de estos parámetros.

7.4.4. Resultados

En esta sección se presentan los resultados de los experimentos realizados.

Resultados globales para cada escenario

En la Tabla 7.23 se resumen los resultados de las 10 ejecuciones del algoritmo realizadas para cada uno de los 54 escenarios con un tamaño de población de 100. Los resultados se agrupan por tamaño de la red original que generó el escenario y por el tamaño del incremento en la población de vehículos. Estos dos criterios de agrupamiento se muestran en la primera y segunda columna. La tercer y cuarta columna muestran la media y el coeficiente de desvío para la GD y la quinta y sexta columna los mismos estadísticos pero para la GS.

Se observa como, a mayor tamaño de la red y mayor cantidad de vehículos, la calidad del algoritmo tiende a disminuir, si bien logra encontrar soluciones cercanas a la Frontera de Pareto. Se observa también que la GS aparenta ser mas estable que la GD, en el sentido que las medias y los coeficientes de desvío de la primera se mantienen en un rango mas acotado que para la segunda.

Es interesante notar que la variabilidad de los resultados es alta, con un coeficiente de desvío que ronda el 40 %. Esto nos permite inferir que o bien el algoritmo necesita de mas generaciones,

Tabla 7.23: Distancia generacional y uniformidad agrupados por tamaño de grilla y crecimiento poblacional

Escenario	Incr. Pob.	Media GD	Coef. Desv GD	Media GS	Coef. Desv. GS
10x10	1.05	0.37	0.42	0.33	0.43
10x10	1.15	0.44	0.41	0.37	0.56
25x50	1.05	0.48	0.42	0.31	0.38
25x50	1.15	0.48	0.47	0.35	0.42
75x75	1.05	0.49	0.51	0.38	0.52
75x75	1.15	0.58	0.49	0.41	0.48

para poder lograr mayor convergencia a la frontera óptima, o bien debe ser ejecutado mediante varias repeticiones y elegir la mejor frontera aproximada.

Distancia Generacional

En la Tabla 7.24 se observan los valores de las métricas analizadas para uno de los escenarios de tamaño 10×10 con incremento de la población de vehículos de 1,15. La media de este conjunto es de 0,40 y el coeficiente de desvío 0,39. La mayoría de las ejecuciones se mantuvieron en el mismo rango de valores para la GD, salvo dos observaciones atípicas, en una de las cuales se presume el efecto de la presencia de óptimos locales (el ítem 4). El valor tan bajo observado en el ítem 3 se corresponde con una observación en la cual se encontró con un conjunto de soluciones que mantenían un bajo tiempo medio de viaje con una configuración de bajo costo de infraestructura. Para este escenario puntual se volvió a correr el algoritmo con una población de 250 individuos. Los resultados se muestran en la Tabla 7.25. Se puede observar una mejora en los indicadores (con un incremento del valor de la GD para el ítem 3 debido al no quedar atrapada en el grupo de soluciones anterior). La media de este nuevo conjunto de pruebas es 0,24 y el coeficiente de desvío 0,47

Tabla 7.24: Distancia generacional y uniformidad en grilla $10x10$ con un factor de incremento de población de 1,15

Ejecución	Dist. Generacional	Uniformidad
1	0.3533009	0.2291444
2	0.4108350	0.4742433
3	0.0837602	0.4537517
4	0.7143926	0.3012058
5	0.3307537	0.1526166
6	0.4584290	0.1768969
7	0.3969564	0.2108491
8	0.4987557	0.3343977
9	0.4229219	0.2606873
10	0.3682840	0.1414944

Tabla 7.25: Distancia generacional y uniformidad en grilla $10x10$ e incremento de población de 1,15 (tamaño de población = 250)

Ejecución	Dist. Generacional	Uniformidad
1	0.2106560	0.1374553
2	0.3108199	0.3622892
3	0.1789973	0.2721308
4	0.2022944	0.2191117
5	0.2454660	0.1283510
6	0.3290329	0.3134578
7	0.4945728	0.2041503
8	0.0927502	0.3940818
9	0.2933584	0.1594452
10	0.0771648	0.3538501

Uniformidad

La Tabla 7.24 presenta los valores de GS para el escenario presentado en la Sección 7.4.4. Se observa que el comportamiento aquí es diferente, con una gran cantidad de soluciones con buenos valores de GS, y algunas soluciones con GS alta. La media del conjunto es 0,27 y el coeficiente de desvío 0,40. Ejecutado nuevamente el algoritmo con una población de tamaño 250 no se observaron mejoras significativas en este indicador (la nueva media es 0,25 y el coeficiente de desvío 0,36)

Evolución de las soluciones generadas por el algoritmos

En las figuras 7.6 y 7.7 se pueden ver las soluciones de las iteraciones 25 y 100 para una ejecución del algoritmo, comparadas con la Frontera de Pareto de referencia, para el escenario de la Sección 7.4.4. El ajuste en la iteración 100 es muy bueno, con casi todas las soluciones devueltas por el algoritmo pertenecientes a la frontera. Se observa además que, si bien no hay una gran diversidad de soluciones, se barre la frontera casi en su totalidad. En la iteración 25 podemos ver que si bien la cercanía de las soluciones a la frontera es muy baja, el algoritmo ya encontró varios individuos que pertenecen al conjunto óptimo.

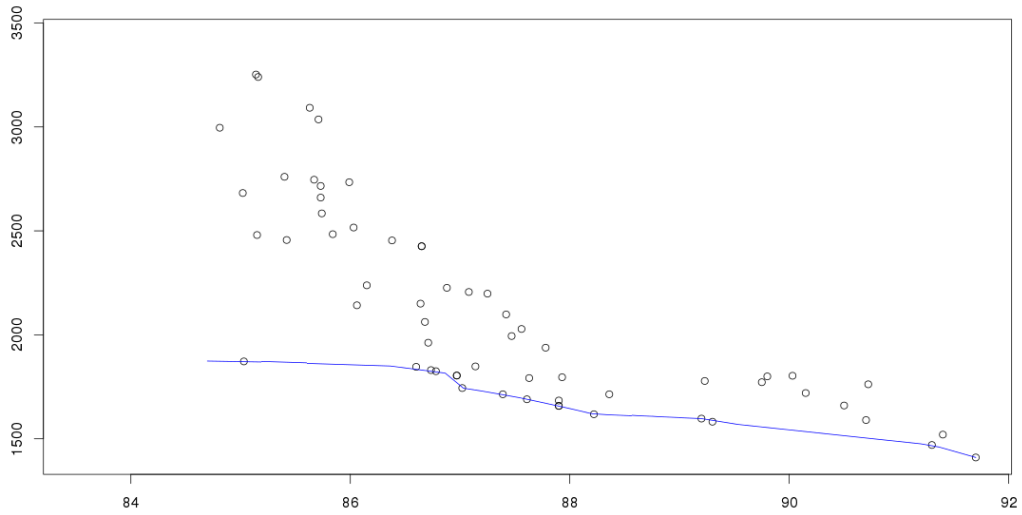


Figura 7.6: Iteracion 25. La linea poligonal es la Frontera de Pareto de referencia, los puntos son los individuos en la última generación.

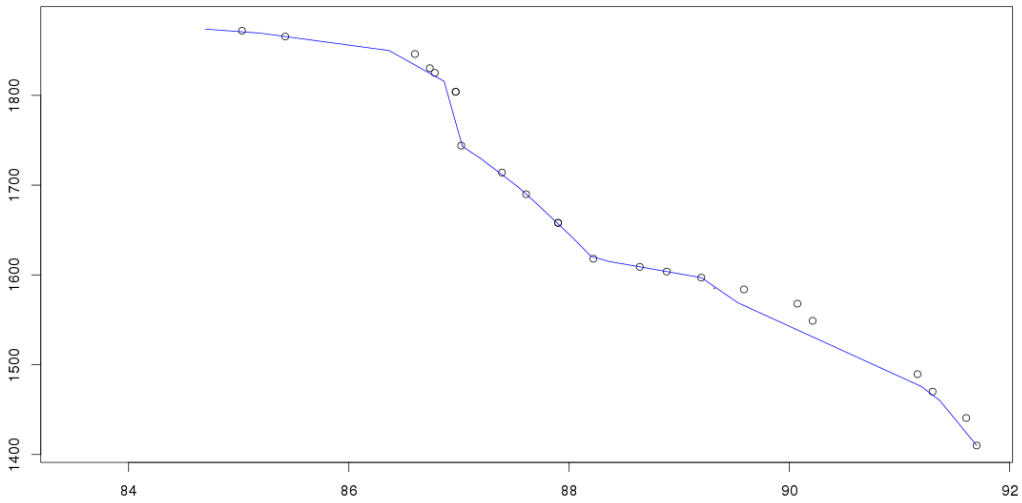


Figura 7.7: Iteracion 100. La linea poligonal es la Frontera de Pareto de referencia, los puntos son los individuos en la última generación.

En base a los resultados de esta sección y de las secciones anteriores, se puede inferir que el algoritmo desarrollado necesita de un mayor número de iteraciones o de una población más grande para poder lograr una mayor cercanía a la Frontera de Pareto de referencia. Si bien los resultados con una población de 100 individuos no son malos, el incremento a 250 mejora sustancialmente los valores obtenidos para la GD en el escenario analizado. La distribución de las soluciones respecto de la Frontera de Pareto de referencia no varía demasiado con mas individuos, el algoritmo (posiblemente por el mecanismo del cálculo de la CD), logra una buena distribución de las soluciones con un número bajo de individuos.

Costo computacional

Por último, se evaluó el costo computacional del algoritmo propuesto en términos de tiempo total simulado. Se comparó el uso de la versión basada en SMPSO con dos nuevas versiones basadas en K-NSGA-II y VSL-NSGA-II. Para estos dos últimos, se ejecutaron tantas generaciones como las necesarias para arribar a los mismos valores de GD mostrados en la Tabla 7.23. La configuración utilizada es la misma que la mostrada en la Tabla 7.3 excepto que el tamaño de población de modificó a 100. Los parámetros propios de estos dos últimos algoritmos son los mismos que los mostrados en las Secciones 7.1.3 y 7.2.3. Los resultados se muestran en la

Tabla 7.26: Tiempo simulado para resolver el ODTAP, implementado con SMPSO, K-NSGA-II y VSL-NSGA-II (medido en múltiplos de 1000)

Escenario	Incr. Pob.	SMPSO	K-NSGA-II	VSL-NSGA-II
10x10	1,05	7945	5875	6880
10x10	1,15	8164	5934	6951
25x50	1,05	9610	6348	7082
25x50	1,15	9841	6591	7334
75x75	1,05	11274	7756	8276
75x75	1,15	11754	7941	8631

Tabla 7.26. La primera columna es el tamaño del escenario, la segunda el incremento de la demanda, y desde la tercera a la quinta el tiempo total simulado del SMPSO, el K-NSGA-II y el VSL-NSGA-II. Tanto K-NSGA-II como VSL-NSGA-II logran ahorros significativos, pero K-NSGA-II obtiene consistentemente los mejores resultados. Se puede ver también que el ahorro parece comportarse en forma inversa a la calidad: en las instancias chicas, donde la calidad del algoritmo es mejor, cuesta mas reducir el costo computacional que en las instancias en las cuales se está mas lejos de la frontera.

7.5. Problema de Inversión de Carriles con Sincronización de Semáforos

En esta sección se muestran los experimentos realizados para validar el algoritmo de inversión de carriles considerando el efecto de la resincronización de los semáforos. Con el fin de evaluar nuestra metodología, se realizaron dos tipos de experimentos. En primera instancia, se compararon los resultados del algoritmo contra los resultados obtenidos empleando la metodología propuesta por [Zhao et al., 2014]. Luego, se recreo la Frontera de Pareto de un nuevo modelo (mediante instancias sucesivas de nuestro algoritmo) y se obtuvieron indicadores para la GD y la GS de las instancias individuales respecto de la misma. El procedimiento de OvS fue implementado en el software R [Team, 2018] utilizando la biblioteca RSUMO.

Tabla 7.27: Parámetros bloque selección de carriles.

Parámetro	Valor
Cantidad de iteraciones	100
Tamaño de la población	100
Tamaño de la lista de líderes	10
Probabilidad de mutación	0.1

Tabla 7.28: Parámetros bloque configuración semáforos.

Parámetro	Valor
Cantidad de Iteraciones	100
Temperatura Inicial	1000
Tamaño de vecindario	8
Método de generación de vecinos	Mutación de componentes

7.5.1. Configuración del algoritmo

La Tabla 7.27 muestran los parámetros bajo los cuales se ejecutó el bloque de selección de carriles a invertir del algoritmo propuesto en todos los casos analizados, mientras que la Tabla 7.28 muestra los parámetros del bloque que calcula la sincronización de semáforos. Estos valores surgieron de experimentos preliminares, donde se probaron distintas combinaciones de valores de estos parámetros.

7.5.2. Comparación de OvS respecto de un modelo analítico

A fin de validar la calidad de las soluciones, se resolvieron los problemas propuestos por [Zhao et al., 2014]. Los mismos consisten en el análisis de una avenida con intersecciones semaforizadas, en donde el objetivo es sincronizar los semáforos e invertir carriles a fin de disminuir el nivel de congestión. El esquema del escenario propuesto por dicho autor se observa en la Figura 7.8.

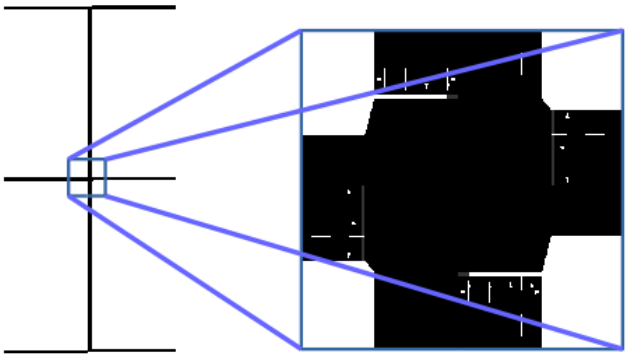


Figura 7.8: Izq: Red de tráfico propuesta por [Zhao et al., 2014] para evaluar el algoritmo. Der: Zoom sobre intersección central.

De los seis carriles de la avenida, los dos centrales son los únicos que se pueden invertir. La duración de cada fase de semáforos está comprendida entre los 60 y 120 segundos. El flujo de vehículos máximo en cada carril es de 1900 vehículos/hora y se desea que el flujo efectivo se ubique como máximo al 0,85 de dicho valor. Siguiendo la metodología de [Zhao et al., 2014], se probaron 6 escenarios sobre este modelo, cada uno con diferentes patrones de circulación. El algoritmo basado en OvS se ejecutó 30 veces por escenario.

En la tabla 7.29 se pueden observar los resultados. Para cada uno de los 6 escenarios se observa el número de carriles invertidos que se utilizó en la comparación y el nivel de saturación (cantidad de vehículos circulando respecto de la cantidad esperada) de cada intersección. Mientras mas cercana a 1 está la saturación, mejor es el desempeño del algoritmo respecto a este objetivo. Cada columna corresponde a un escenario distinto. El primer bloque de filas contiene las soluciones halladas por el modelo analítico. Se muestra primero la cantidad de carriles invertidos, y luego la saturación (cantidad de vehículos circulando respecto de la cantidad esperada) en cada intersección. El segundo bloque de filas muestra la misma información para el esquema de OvS (se muestran los valores medios) para aquellas soluciones de la Frontera de Pareto estimada con la misma cantidad de carriles invertidos. Hay que notar que este modelo es muy simple, con poco impacto neto de los efectos dinámicos del flujo de tráfico, la no localidad y la aleatoriedad en la velocidad de los vehículos, cosas si consideradas en el esquema de OvS. Sin embargo, se observa que la utilización del enfoque basado en OvS alcanza resultados en promedio similares

Tabla 7.29: Comparativa de OvS vs. Modelo Analítico.

	Esce. 01	Esce. 2	Esce. 03	Esce. 04	Esce. 05	Esce. 06
Carriles Invertidos	0	0	0	2	2	2
Analítico - Saturación Intersección 1	0.683	0.553	0.523	0.602	0.596	0.802
Analítico - Saturación Intersección 2	0.784	0.791	0.827	0.774	0.753	0.709
Analítico - Saturación Intersección 3	0.523	0.647	0.691	0.689	0.654	0.741
Carriles Invertidos	0	0	0	2	2	2
OvS - Saturación Intersección 1	0.714	0.569	0.520	0.631	0.621	0.839
OvS - Saturación Intersección 2	0.818	0.832	0.833	0.806	0.781	0.756
OvS - Saturación Intersección 3	0.575	0.656	0.689	0.707	0.693	0.789

pero superando en algunos casos la versión analítica por una diferencia de 0,05 (debido a que el modelo analítico evalúa la saturación más simplificadamente que la simulación).

De aquí se pueden extrapolar un resultado importante: primero, el modelo analítico funciona muy bien en un escenario muy acotado como el estudiado, y, segundo, la OvS logra replicar sus resultados. Por lo tanto, para escenarios mas complejos, en los que la dificultad de realizar el modelo analítico es mayor, se puede suponer que la OvS lograría resultados similares (con una mayor facilidad de modelado).

7.5.3. Análisis de la calidad de aproximación de la Frontera de Pareto

La segunda tanda de experimentos consistió en evaluar la calidad de la Frontera de Pareto hallada en una ejecución del algoritmo contra una estimación mas completa de la Frontera de Pareto. Debido a que el problema estudiado en la Sección 7.5.2 presenta una Frontera de Pareto muy pequeña (se pueden invertir 0, 1 o 2 carriles solamente), se testeó el algoritmo con una red de tráfico del tipo grilla cuadrada de 20 calles de lado. El origen de la ola de tráfico se situó en uno de los vértices de la grilla, fluyendo la misma hacia los lados opuestos del trazado. Para la demanda de tráfico, se crearon 10 escenarios distintos de flujo de vehículos sobre esta misma red.

A fin de evaluar la bondad del algoritmo, se calcularon los índices de GD y GS de la Frontera generada. Se ejecutaron 50 instancias del algoritmo, aproximándose la Frontera de Pareto real

Tabla 7.30: Distancia generacional y uniformidad

Escenario	Núm. de Vehículos Totales	Dist. Generacional	Uniformidad
1	1.500	0.28 [0.22-0.30]	0.32 [0.27-0.35]
2	1.500	0.23 [0.19-0.25]	0.35 [0.31-0.39]
3	1.500	0.27 [0.24-0.30]	0.38 [0.33-0.41]
4	1.500	0.45 [0.32-0.48]	0.46 [0.41-0.59]
5	2.500	0.16 [0.12-0.18]	0.23 [0.20-0.28]
6	2.500	0.32 [0.27-0.33]	0.26 [0.197-0.33]
7	2.500	0.21 [0.18-0.26]	0.24 [0.20-0.28]
8	4.000	0.10 [0.07-0.12]	0.18 [0.14-0.21]
9	4.000	0.25 [0.23-0.28]	0.19 [0.17-0.24]
10	4.000	0.31 [0.29-0.34]	0.21 [0.17-0.23]

del sistema mediante la unión de las 50 Fronteras de Pareto generadas y su posterior recálculo según el criterio de dominancia. Esto nos permite obtener un conjunto más reducido de puntos que no son dominados por ninguna de las 50 Fronteras de Pareto generadas. Luego, se calcularon los valores de GD y GS de las 50 fronteras iniciales contra este último conjunto de puntos. En la Tabla 7.30 se pueden observar los resultados (se muestran los valores medios y los percentiles 5 y 95). La primera columna indica el escenario, la segunda la cantidad de vehículos en dicho escenario, la tercera columna la media de la GD junto al valor mínimo y máximo hallado y la cuarta los mismos datos para la GS.

Se puede observar que los valores de los indicadores no están influenciados fuertemente por el volumen en si de los vehículos circulando en el sistema. Las medias de los dos indicadores obtienen buenos resultados, pero en la mayoría de las 50 iteraciones realizadas por escenario, la Frontera de Pareto de referencia se logra estimar en forma parcial (en caso contrario, tanto la GD como la GS serían mucho mas cercanas a cero, dado la forma en que fueron construidos los test).

En base a los resultados de estos test y los presentados en la Sección 7.5.2, se puede inferir que, si bien el algoritmo presenta un buen rendimiento, necesita de un número mayor de solu-

ciones evaluadas (esto es, mas generaciones y un tamaño de población mayor) para aproximar correctamente la Frontera de Pareto.

7.6. Problema de Ruteo de Autobuses

En esta sección se muestran los experimentos realizados respecto del algoritmo de resolución del SSBRP, descripto en la Sección 5.6. Para este problema, el objetivo fue evaluar el desempeño del algoritmo propuesto bajo distintas alternativas de paradas disponibles y distribución de pasajeros. Debido a la falta de datos para benchmarks en la bibliografía, se utilizó el conjunto de problemas desarrollados en [Schittekat et al., 2013]. En el mencionado trabajo, los autores desarrollan 120 instancias de prueba para una versión monoobjetivo determinística del problema, en la cual se intenta minimizar la distancia total recorrida por los autobuses acotando la distancia máxima a caminar por los pasajeros. Modificando esta cota y resolviendo el problema, es posible obtener puntos de la Frontera de Pareto para la versión multiobjetivo determinística del problema. El primer conjunto de test consistió en resolver con el algoritmo propuesto en su versión determinística parte de los problemas antes citados, comparando el resultado con la frontera de referencia obtenida variando la cota de distancia caminada. El segundo grupo de test consistió en convertir estos problemas en problemas multiobjetivo estocásticos, calcular una Frontera de Pareto de referencia mediante la repetición de ejecuciones y analizar la aproximación de estas ejecuciones a dicha frontera.

7.6.1. SSBRP determinístico

En esta sección se detallan los experimentos realizados con la versión determinística del algoritmo, a fin de validar los resultados y poder tener un grado de confianza mayor al extrapolar su utilización al caso estocástico.

Tabla 7.31: Instancias seleccionadas para evaluar el SSBRP

Instancia	Cant. Paradas	Cant. Pasajeros	Cap. Autobuses	Max. Dist. a pie
inst1-1s5-25-c25-w5	5	25	25	5000
inst10-1s5-50-c50-w5	5	50	50	5000
inst20-1s5-100-c50-w10	5	100	50	10000
inst30-5s10-50-c50-w20	10	50	50	20000
inst50-1s20-100-c50-w5	20	100	50	5000
inst59-9s20-200-c25-w10	20	200	25	10000
inst70-8s20-400-c50-w20	20	400	50	20000
inst90-2s40-800-c50-w5	40	800	50	5000
inst97-8s80-400-c25-w5	80	400	25	5000
inst112-5s80-800-c50-w40	80	800	50	40000

Problemas evaluados

Para evaluar la versión determinística del algoritmo, se seleccionaron 10 instancias del conjunto de problemas desarrollados en [Schittekat et al., 2013]. La Tabla 7.31 muestra las características de los problemas seleccionados. La primer columna indica el nombre de la instancia de prueba (el problema a tratar), la segunda la cantidad de paradas disponibles, la tercera la cantidad de pasajeros que hay que transportar, la cuarta la capacidad máxima de los autobuses y la quinta la máxima distancia a pie.

Para cada problema, se calculó la Frontera de Pareto de referencia de su problema multiobjetivo determinístico asociado haciendo variar la *Máxima distancia a pie* entre 0 y el doble del valor indicado en la Tabla 7.31, en pasos de tamaño igual al 5 % del valor de cota propuesto por [Schittekat et al., 2013] (es decir, para la instancia *inst1-1s5-25-c25-w5* se hizo variar esta cota entre 0 y 10000 en intervalos de tamaño 250)[3]. Respecto de la cantidad de buses, este conjunto de problemas asume que no hay restricciones en la cantidad de vehículos, por lo tanto, a efectos de poder resolver el problema mediante la codificación definida en la Sección 5.6.2, se limita la cantidad de vehículos al doble del cociente de la cantidad de pasajeros a transportar respecto de la capacidad de los autobuses.

[3]En el conjunto de problema desarrollado por [Schittekat et al., 2013], un valor de cota igual a 0 no arroja soluciones factibles, ya que ningún pasajero tiene una localización inicial en las mismas coordenadas que alguna parada.

Métricas utilizadas

Con el fin de evaluar el rendimiento del algoritmo, se utilizaron solo métricas de calidad: GD y GS. Las mismas fueron previamente descriptas en la Sección 2.5.

Configuraciones utilizadas

El algoritmo se ejecutó utilizando la configuración indicada en la Tabla 7.32 para el bloque exterior de asignación de paradas a autobuses, mientras que la configuración indicada en la Tabla 7.33 fue utilizada en el bloque de construcción de rutas. La asignación de pasajeros a los autobuses no se simuló en este caso, sino que se resolvió vía PLE, utilizando el solver *Coin-OR CBC*. Los valores de los parámetros surgieron de experimentos preliminares, donde se probaron distintas combinaciones de valores de estos parámetros.

Tabla 7.32: Parámetros bloque exterior SSBRP

Parámetro	Valor
Cantidad de Generaciones	100
Tamaño de la población	50
Tamaño del torneo	2
Probabilidad de cruzamiento	0.8
Probabilidad de mutación	0.1
Tasa de elitismo	0.1

Tabla 7.34: Resultados SSBRP determinístico.

Instancia.	Dist. Generacional	Uniformidad
inst1-1s5-25-c25-w5	0.004	0.176
inst10-1s5-50-c50-w5	0.006	0.181
inst20-1s5-100-c50-w10	0.007	0.201
inst30-5s10-50-c50-w20	0.009	0.231
inst50-1s20-100-c50-w5	0.009	0.259
inst59-9s20-200-c25-w10	0.012	0.331
inst70-8s20-400-c50-w20	0.013	0.394
inst90-2s40-800-c50-w5	0.016	0.458
inst97-8s80-400-c25-w5	0.017	0.546
inst112-5s80-800-c50-w40	0.020	0.616

Tabla 7.33: Parámetros bloque interior SSBRP

Parámetro	Valor
Cantidad de Iteraciones	100
Temperatura Inicial	1000
Tamaño de vecindario	4
Método de generación de vecinos	Permutación de dos componentes

Cada experimento se repitió 30 veces, computándose las medias de los indicadores elegidos.

Resultados

Los valores medios de GD y GS respecto de la frontera de referencia se muestran en la Tabla 7.34. Los resultados muestran que las Fronteras de Pareto estimadas por el algoritmo se encuentran muy cercanas a las Fronteras de Pareto de referencia. Sin embargo, existe una evidente tendencia hacia el agrupamiento de soluciones a medida que el tamaño del problema aumenta. La GD tiene

también esta tendencia, pero los valores obtenidos siguen siendo muy buenos, a diferencia de la GS, que exhibe en las instancias mas grandes un estado de agrupamiento bastante elevado. Esto es, en parte, debido a la utilización de la misma cantidad de generaciones y el mismo tamaño de población en todos los problemas, y por otro lado, debido al hecho que las Fronteras de Pareto de referencia fueron construidas para tener un cantidad máxima de puntos igual a 40[4]. Amén a esto, la aproximación lograda por el algoritmo es muy buena, pudiendo calificarse este enfoque alternativo de resolución como válido[5].

A fin de tener una visión un poco mas clara de las soluciones halladas, la Figura 7.9 muestra una de las fronteras obtenidas para la instancia *inst1-1s5-25-c25-w5*. Además, la Figura 7.10 muestra como la aproximación fue evolucionando a través de las generaciones 5, 10 y 20 para dicha instancia. En esta última imagen, se muestra la población completa a cada generación, y se puede observar como se generan soluciones no factibles en las primeras iteraciones. En ambas imágenes, el eje vertical es la distancia total caminada por los pasajeros, y el eje horizontal la distancia total recorrida por los buses. Por último, la Figura 7.11 muestra las rutas y la distancia caminada a pie para una de las soluciones de la frontera aproximada.

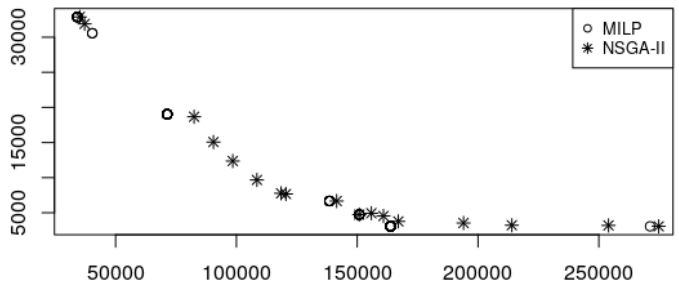


Figura 7.9: Frontera de Pareto de referencia (puntos) y aproximada (asteriscos) para la instancia inst1-1s5-25-c25-w5.

[4]Si la frontera de referencia tiene pocos puntos, es posible que muchas soluciones no dominadas no se encuentren en ella. Si entre dos soluciones que efectivamente se encuentran en la frontera de referencia existen varias soluciones que fueron incluidas, la GS puede detectar agrupamiento, cuando en realidad el algoritmo de resolución está trabajando bien.

[5]Asignar paradas a autobuses, rutear autobuses y luego asignar pasajeros, a diferencia de los otros enfoques existentes en la literatura.

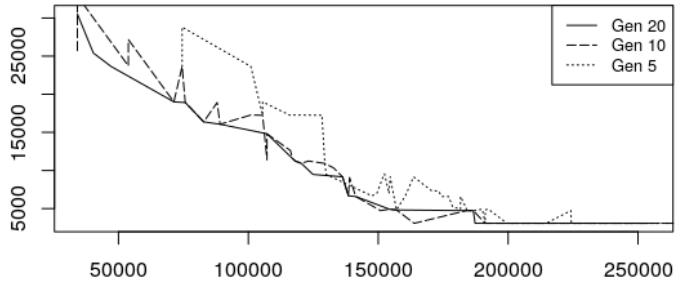


Figura 7.10: Fronteras obtenidas para la instancia inst1-1s5-25-c25-w5 en las generaciones 5, 10 y 20.

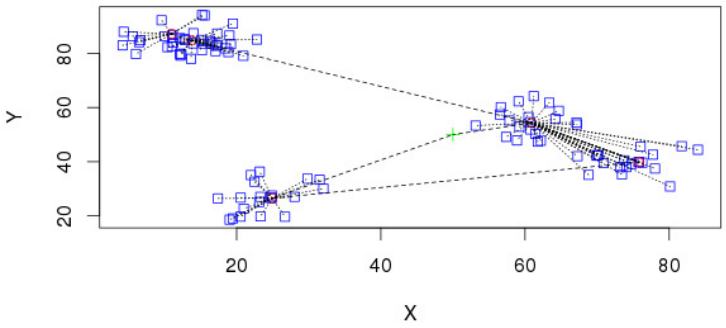


Figura 7.11: Rutas de autobuses y caminatas de pasajeros de una de las soluciones halladas para la instancia inst1-1s5-25-c25-w5

7.6.2. SSBRP estocástico

En esta sección se detallan los experimentos realizados con la versión estocástica del algoritmo.

Problemas evaluados

Para evaluar la versión estocástica del algoritmo, se seleccionaron las mismas 10 instancias usadas en la Sección 7.6.1 (ver Tabla 7.31). A fin de aleatorizar la demanda, se optó por

aleatorizar la distancia entre las localizaciones de los pasajeros y las paradas. Para ello, se calculó la distancia media existente entre pasajeros y paradas (llamada dmt) y para cada pasajero l la distancia media a cada uno de los demás (dm_l). Luego, la distancia da_{ls} aleatorizada entre el pasajero l y la parada s se calculó de acuerdo a la Ecuación 7.6, donde d_{ls} es la distancia indicada en la instancia, y $u(l)$ es una variable aleatoria continua distribuida en forma uniforme en el intervalo $(0, \text{máx}(dmt, dm_l))$[6].

$$da_{ls} = d_{ls} + u(l) \tag{7.6}$$

Para cada problema se construyeron 100 escenarios (80 de entrenamiento y 20 de test), y para cada escenario fue calculada la Frontera de Pareto de acuerdo al método descripto en la Sección 7.6.1, asumiendo ahora que más de un autobús puede levantar pasajeros en una misma parada. Adicionalmente, se construyó, por el método anterior, la Frontera de Pareto considerando la media del objetivo asociado a la distancia. Por último, a fin de poder realizar las comparaciones necesarias, se mantuvo un registro de todas las soluciones factibles generadas en los procedimientos anteriores y se construyó un ranking de Pareto general.

Métricas utilizadas

Con el fin de evaluar el rendimiento del algoritmo, se utilizó una métrica de calidad: el promedio del valor medio en el ranking de Pareto obtenido por las soluciones de las fronteras aproximadas en los escenarios de test. Usando los conjuntos de puntos generados para calcular las Fronteras de Pareto de los escenarios de entrenamiento y test, se puede estimar el valor de ranking de Pareto de cada punto de la frontera generada por el algoritmo analizado. El resultado es un indicador empírico que mide que tan cerca o no de la no-dominancia se encuentra la frontera hallada por el algoritmo[7].

[6] Se usó una distribución uniforme por cada uno de los estudiantes.

[7] A efectos de comparar dos o mas alternativas de algoritmos, usando los datos previamente recolectados, se presenta como una buena opción.

Tabla 7.35: Resultados SSBRP Estocástico utilizando NSGA-II.

Instancia.	Promedio Ranking Medio
inst1-1s5-25-c25-w5	4.3
inst10-1s5-50-c50-w5	4.6
inst20-1s5-100-c50-w10	5.1
inst30-5s10-50-c50-w20	5.2
inst50-1s20-100-c50-w5	5.5
inst59-9s20-200-c25-w10	5.7
inst70-8s20-400-c50-w20	5.8
inst90-2s40-800-c50-w5	6.0
inst97-8s80-400-c25-w5	6.3
inst112-5s80-800-c50-w40	6.4

Configuraciones utilizadas

El algoritmo se ejecutó utilizando la configuración indicada en la Tabla 7.32 para el bloque exterior de asignación de paradas a autobuses, mientras que la configuración indicada en la Tabla 7.33 fue utilizada en el bloque de construcción de rutas. La asignación de pasajeros a los autobuses fue simulada bajo el supuesto que cada pasajero se dirigirá a la parada habilitada mas cercana. Estos valores (los parámetros) surgieron de experimentos preliminares, donde se probaron distintas combinaciones de valores de estos parámetros.

Para cada problema, el algoritmo se ejecutó en los escenarios de entrenamiento , computándose el indicador elegido en el conjunto de escenarios de test.

Resultados

Los resultados de los experimentos realizados se muestran en la Tabla 7.35. Se observa que la aproximación convencional a la optimización estocástica, por medias, no resultó muy buena para este problema. El promedio de las medias de las distancias a las fronteras de cada escenario de test es muy alto, significando que al momento de implementar las soluciones en la práctica, estas estarán lejos del trade-off óptimo. También se observa el mismo efecto de la versión

Tabla 7.36: Resultados SSBRP Estocástico utilizando S-NSGA-II.

Instancia.	Promedio Ranking Medio
inst1-1s5-25-c25-w5	2.7
inst10-1s5-50-c50-w5	2.9
inst20-1s5-100-c50-w10	3.2
inst30-5s10-50-c50-w20	3.3
inst50-1s20-100-c50-w5	3.6
inst59-9s20-200-c25-w10	3.7
inst70-8s20-400-c50-w20	3.7
inst90-2s40-800-c50-w5	4.1
inst97-8s80-400-c25-w5	4.2
inst112-5s80-800-c50-w40	4.5

determinística respecto a la baja de calidad a medida que el tamaño de los problemas aumenta, efecto parcialmente debido a la utilización del mismo conjunto de parámetros para todos los problemas.

A fin de tratar de mejorar los resultados, se realizaron nuevamente los experimentos usando, en el algoritmo desarrollado, la metaheurística S-NSGA-II en vez de la NSGA-II. El parámetro de distancia máxima del S-NSGA-II se fijó en 2 y el valor de p en 75. Los resultados obtenidos se muestran en la Tabla 7.36.

En este caso, se detecta una mejora sustancial en el promedio de la media de los valores de ranking, de casi dos puntos de ranking en los problemas mas grandes, lo cual permite tener mayor confianza en que, cuando se implemente alguna de las soluciones generadas por el algoritmo, el resultado real estará cercano al óptimo.

Capítulo 8

Conclusiones

8.1. Conclusiones del trabajo desarrollado

En esta tesis doctoral se han presentado algoritmos basados en *Optimización vía Simulación* para el tratamiento de problemas asociados a los sistemas de tráfico urbano. En particular se abordan tres problemas de optimización estocásticos multiobjetivo: el rediseño de la red de tráfico para absorber el incremento poblacional proyectado, la inversión de carriles de tráfico como opción para el tratamiento de las olas de tráfico y la resolución del problema de transporte de personas sobre una red de tráfico. Además, se presentaron tres algoritmos generales, aplicables a cualquier tipo de problema, no solamente de tráfico: un algoritmo que utiliza metamodelos para reducir el costo computacional de la OvS, un segundo algoritmo que intenta lograr el mismo objetivo usando tiempos de simulación variables y un tercer algoritmo que permite obtener Fronteras de Pareto mas robustas que en el método tradicional [1].

Durante el transcurso del desarrollo de esta tesis se planteó un problema nuevo en sistemas de tráfico al cual se denominó *Problema de Asignación Origen-Destino del Tráfico* (ODTAP). El mismo se modeló en forma multiobjetivo, utilizando simulación para evaluar uno de los dos objetivos propuestos y la metaheurística SMPSO para rastrillar el espacio de soluciones, presentando un enfoque novedoso para tratar problemas de planificación urbana, el cual logra

[1] *Mas robustas* no hace referencia aquí a las metodologías de *Optimización Robusta* sino al hecho que las soluciones siguen estando cercanas la Frontera de Pareto al momento de su implementación.

una muy buena aproximación de la Frontera de Pareto asociada a este problema.

Otro de los algoritmos diseñados consistió en un algoritmo para resolver el problema de inversión en carriles de tráfico, problema multiobjetivo tratado usualmente en forma analítica. Se utilizó una simulación de tráfico para la evaluación de la circulación en el sistema y el algoritmo SMPSO para guiar la búsqueda a la Frontera de Pareto del problema. Comparado su rendimiento con uno de los enfoques analíticos presentes en la literatura, el algoritmo aquí presentado muestra un muy buen desempeño, igualando o superando a dicho enfoque analítico. Y analizando la proximidad a la Frontera de Pareto en los problemas de test creados, se encuentran muy buenos resultados.

El último problema resuelto fue una versión multiobjetivo y estocástica del *Problema de ruteo de vehículos escolares* (SBRP) que denominamos Problema de Ruteo de Autobuses Escolares a una Única Escuela (SSBRP, Single School Bus Routing Problem). Si bien este problema tiene larga data dentro del campo de la Investigación Operativa, se presenta aquí en forma novedosa el hecho de considerar aleatoria la distribución espacial de los estudiantes (o pasajeros) en un esquema de optimización multiobjetivo. Se resolvió primero una versión determinística, la cual logra muy buenos resultados respecto de los problemas de testeo usados, y luego la versión estocástica del mismo (basada en la versión determinística), la cual genera buenas aproximaciones de la Frontera de Pareto.

Debido a que los algoritmos de optimización basados en OvS adolecen de un alto costo computacional, se propusieron aquí dos alternativas de reducción de dicho costo. Una de ellas basada en el uso de metamodelos de Kriging (K-NSGA-II), mientras que la otra basada en la reducción del tiempo de simulación (VSL-NSGA-II). Ambas presentan como aspecto novedoso la validación de los metamodelos o de la reducción de tiempo de simulación mediante el uso de muestreos respecto a simulaciones reales o completas, según el caso. K-NSGA-II logra reducir hasta en un 50 % la cantidad de evaluaciones necesarias para estimar la Frontera de Pareto en algunos de los problemas de testeo. VSL-NSGA-II logra reducciones algo menores, cerca del 25 % del costo de simulación, pero en forma mas consistente a lo largo de todo el conjunto de problemas de prueba. Sin embargo, es muy dependiente de la forma en que el objetivo a evaluar evolucione en el tiempo. Ambos algoritmos, además, fueron testeados en el marco del ODTAP,

logrando significativas reducciones en el tiempo total simulado.

Por último, se propuso un algoritmo de optimización multiobjetivo estocástico que hace uso de un criterio de selección de soluciones orientados al buen desempeño durante la implementación de las soluciones en la realidad (S-NSGA-II). Contrastado contra la metodología convencional para el tratamiento de problemas multiobjetivo estocásticos, se observa un rendimiento superior en el algoritmo aquí desarrollado. Dicho algoritmo fue también evaluado en el marco del SSBRP con demanda aleatoria, mejorando los resultados previamente logrados por la versión convencional del algoritmo creado para tal fin, obteniendo en promedio Fronteras de Pareto aproximadas dos unidades de ranking de Pareto mas cercanas a las fronteras reales de los escenarios de test.

8.2. Aplicaciones directas del trabajo desarrollado

Los algoritmos desarrollados en el Capítulo 5 tienen aplicación directa en problemas de optimización de tráfico urbano. Puntualmente, el algoritmo de resolución del ODTAP podría ser generalizado para abarcar mas aspectos relativos a la planificación urbana. El algoritmo de resolución del SSBRP tiene aplicaciones principalmente a nivel comercial, ya que la situación planteada es muy común en empresas que proporcionan servicio de traslado a sus empleados. Los algoritmos desarrollados en el Capítulo 6 tienen aplicaciones no solo en problemas relacionados con tráfico, sino en problemas de planificación de manufactura, confección de cronogramas, anidado de piezas, etc. La OvS es muy utilizada en diseño mecánico, donde en general las simulaciones consumen más tiempo todavía que las simulaciones de tráfico, por lo cual las metodologías de ahorro de costo propuestas deberían generar mayores beneficios.

8.3. Trabajos futuros

Como trabajos futuros quedan pendiente la continuación del estudio de las propiedades de metamodelos en el proceso de reducción de costo computacional, así como la combinación con técnicas de reducción de tiempo de simulación, lo cual podría incrementar aún más la

magnitud del ahorro de costo computacional. Puntualmente hablando del VSL-NSGA-II es necesario realizar un estudio de las características dinámicas de las simulaciones que permiten el uso de este algoritmo.

También es necesario realizar una evaluación de las propiedades teóricas del ODTAP, así como la detección de casos particulares, que puedan dar lugar a procedimientos simplificados de resolución.

Respecto del S-NSGA-II, se plantea como trabajo futuro el análisis de las propiedades de las Fronteras de Pareto estocásticas y la variable de puntuación en el ranking de Pareto.

Publicaciones

Aquí se resume la producción científica del autor de esta tesis en colaboración con sus directores. Estas publicaciones han sido fruto de las investigaciones originales desarrolladas a lo largo de esta tesis doctoral. Las publicaciones avalan el interés, la validez, y las aportaciones de este trabajo en la literatura, ya que estos trabajos han aparecido publicados en foros de prestigio y, por tanto, han sido sometidos a procesos de revisión por investigadores especializados.

Publicaciones en Revistas indexadas en Scopus

- Baquela, E. G. and Olivera, A. C. (2019). A novel hybrid multi-objective metamodel-based evolutionary optimization algorithm. *Operations Research Perspectives*, 6:Article 100098

Publicaciones como Capítulo de Libro en Editoriales Internacionales

- Baquela, E. G. and Olivera, A. C. (2018). A multi-objective optimization via simulation framework for restructuring traffic networks subject to increases in population. In L. Amodeo, E-G. Talbi, F. Y., editor, *Recent Developments in Metaheuristics*, volume 62 of *Operations Research/Computer Science Interfaces Series*. Springer
- Baquela, E. G. and Olivera, A. C. (2016a). *Handbook of Research on Modern Optimization Algorithms and Applications in Engineering and Economics*, chapter Optimization of traffic network design using nature-inspired algorithm: An Optimization via Simulation Approach, pages 266–287. 9781466696440. IGI Global
- Baquela, E. G. and Olivera, A. C. (2015). *Engineering Optimization 2014*, volume IV of *978-1-138-02725-1*, chapter Robust Assignment of Fleet Size and Travel Routes for Transportation to a Single-Destiny Using Optimization via Simulation, pages 557–559. Taylor & Francis Group

Publicaciones en Conferencias Internacionales Indexadas en Qualis

- Baquela, E. G. and Olivera, A. C. (2016b). Un enfoque multiobjetivo multinivel para resolver el problema de inversion de carriles de tráfico y sincronización de seáforos. In *Proceedings of the XVIII Latin-Iberoamerican Conference on Operations Research, CLAIO 2016*, ISBN:978-956-9892-00-4, Santiago de Chile, Chile. Department of Industrial and System Engineering of the School of Engineering, Pontificia Universidad Católica de Chile

- Baquela, E. G. and Olivera, A. C. (2014b). An ovs-multiobjective algorithm approach for lane reversal problem. In *VIII ALIO/EURO Workshop on Applied Combinatorial Optimization*, Montevideo, Uruguay. Facultad de Ingeniería, Universidad de la República.8-10/12

- Baquela, E. G. and Olivera, A. C. (2013a). An optimization via simulation approach for the travel salesman problem with stochastic journey times. In *XXVI EURO - INFORMS 26th European Conference on Operational Research*, Rome, Italy. Sapienza University of Rome. 01-04/07

Publicaciones en Conferencias Internacionales con referato

- Baquela, E. G. and Olivera, A. C. (2014a). Combining genetic algorithms with traffic simulations for restructuring traffic networks subject to increases in population. In *Proceedings of 5th International Conference on Metaheuristics and Nature Inspired Computing (META2014).*

- Baquela, E. G. and Olivera, A. C. (2014c). Robust assignment of fleet size and travel routes for transportation to a single-destiny using optimization via simulation. In H.C., R., H., H., C.M., M. S., J.M., G., A.L., A., J.O., F., F., M., and J.F.A., M., editors, *4th International Conference on Engineering Optimization Book of Abstracts*, 978-989-96276-6-6, page 175. IDMEC Instituto de Ingeniería Mecánica - Instituto Superior Técnico, Universidad de Lisboa. 8-11/09

Publicaciones en Conferencias Nacionales

- Baquela, E. G. and Olivera, A. C. (2013b). Planeación del crecimiento urbano mediante OvS considerando el impacto del tráfico. In Edutecne, editor, *Memorias del COINI 2013 UTN FRSR*, San Rafael. 06-08/11

Bibliografía

[Abdelaziz, 2012] Abdelaziz, F. B. (2012). Solution approaches for the multiobjective stochastic programming. *European Journal of Operational Research*, 216(1):1–16.

[Adeyefa and Luhandjula, 2011] Adeyefa, A. S. and Luhandjula, M. K. (2011). Multiobjective stochastic linear programming: An overview. *American Journal of Operations Research*, 1:203–213.

[Alvarado et al., 2010] Alvarado, J. R. G., Pereira, J. L. F., and Rossetti, R. J. F. (2010). A scalable data support model for traffic simulation in gis. In *13th AGILE International Conference on Geographic Information Science*.

[Amaran et al., 2016] Amaran, S., Sahinidis, N. V., Sharda, B., and Bury., S. J. (2016). Simulation optimization: a review of algorithms and applications. *Annals*, (240):351–380.

[Ankenman et al., 2010] Ankenman, B., Nelson, B., and Staum, J. (2010). Stochastic kriging for simulation metamodeling. *Operation Reseach*, 58(2):371–382.

[Antunes et al., 2016] Antunes, C. H., Alves, M. J., and Clímaco, J. (2016). *Multiobjective Linear and Integer Programming*. EURO Advanced Tutorials on Operational Research. Springer International Publishing Switzerland.

[Banks, 1998] Banks, J., editor (1998). *Handbook of Simulation - Principles, Methodology, Advances, Applications, and Practice*. Wiley Interscience.

[Baquela and Olivera, 2013a] Baquela, E. G. and Olivera, A. C. (2013a). An optimization via simulation approach for the travel salesman problem with stochastic journey times. In

XXVI EURO - INFORMS 26th European Conference on Operational Research, Rome, Italy. Sapienza University of Rome. 01-04/07.

[Baquela and Olivera, 2013b] Baquela, E. G. and Olivera, A. C. (2013b). Planeación del crecimiento urbano mediante OvS considerando el impacto del tráfico. In Edutecne, editor, *Memorias del COINI 2013 UTN FRSR*, San Rafael. 06-08/11.

[Baquela and Olivera, 2014a] Baquela, E. G. and Olivera, A. C. (2014a). Combining genetic algorithms with traffic simulations for restructuring traffic networks subject to increases in population. In *Proceedings of 5th International Conference on Metaheuristics and Nature Inspired Computing (META2014).*

[Baquela and Olivera, 2014b] Baquela, E. G. and Olivera, A. C. (2014b). An ovs-multiobjective algorithm approach for lane reversal problem. In *VIII ALIO/EURO Workshop on Applied Combinatorial Optimization*, Montevideo, Uruguay. Facultad de Ingeniería, Universidad de la República.8-10/12.

[Baquela and Olivera, 2014c] Baquela, E. G. and Olivera, A. C. (2014c). Robust assignment of fleet size and travel routes for transportation to a single-destiny using optimization via simulation. In H.C., R., H., H., C.M., M. S., J.M., G., A.L., A., J.O., F., F., M., and J.F.A., M., editors, *4th International Conference on Engineering Optimization Book of Abstracts*, 978-989-96276-6-6, page 175. IDMEC Instituto de Ingeniería Mecánica - Instituto Superior Técnico, Universidad de Lisboa. 8-11/09.

[Baquela and Olivera, 2015] Baquela, E. G. and Olivera, A. C. (2015). *Engineering Optimization 2014*, volume IV of *978-1-138-02725-1*, chapter Robust Assignment of Fleet Size and Travel Routes for Transportation to a Single-Destiny Using Optimization via Simulation, pages 557–559. Taylor & Francis Group.

[Baquela and Olivera, 2016a] Baquela, E. G. and Olivera, A. C. (2016a). *Handbook of Research on Modern Optimization Algorithms and Applications in Engineering and Economics*, chapter Optimization of traffic network design using nature-inspired algorithm: An Optimization via Simulation Approach, pages 266–287. 9781466696440. IGI Global.

[Baquela and Olivera, 2016b] Baquela, E. G. and Olivera, A. C. (2016b). Un enfoque multiobjetivo multinivel para resolver el problema de inversion de carriles de tráfico y sincronización de seáforos. In *Proceedings of the XVIII Latin-Iberoamerican Conference on Operations Research, CLAIO 2016*, ISBN:978-956-9892-00-4, Santiago de Chile, Chile. Department of Industrial and System Engineering of the School of Engineering, Pontificia Universidad Católica de Chile.

[Baquela and Olivera, 2018] Baquela, E. G. and Olivera, A. C. (2018). A multi-objective optimization via simulation framework for restructuring traffic networks subject to increases in population. In L. Amodeo, E-G. Talbi, F. Y., editor, *Recent Developments in Metaheuristics*, volume 62 of *Operations Research/Computer Science Interfaces Series*. Springer.

[Baquela and Olivera, 2019] Baquela, E. G. and Olivera, A. C. (2019). A novel hybrid multiobjective metamodel-based evolutionary optimization algorithm. *Operations Research Perspectives*, 6:Article 100098.

[Barceló, 2010] Barceló, J. (2010). *Fundamentals of Traffic Simulation*. Springer.

[Barton and State, 2010] Barton, R. and State, P. (2010). Metamodel-based optimization. In *NSF Workshop on Simulation Optimization*.

[Bartz-Beielstein et al., 2018] Bartz-Beielstein, T., Zaefferer, M., and Zaefferer, M. (2018). Optimization via multimodel simulation. *Structural and Multidisciplinary Optimization*, 58(3):919–933.

[Bede and Péter, 2011] Bede, Z. and Péter, T. (2011). The mathematical modeling of reversible lane system. *Transportation Engineering*, 39(1):7–10.

[Bede et al., 2010] Bede, Z., Szabó, G., and Peter, T. (2010). Optimization of road traffic with the applied of reversible directions lanes. *Transportation Engineering*, 38(1):3–8.

[Behrisch et al., 2011] Behrisch, M., Bieker, L., Erdmann, J., and Krajzewicz, D. (2011). Sumo - simulation of urban mobility: An overview. In *SIMUL 2011, The Third International Conference on Advances in System Simulation*, pages 63–68, Barcelona, Spain.

[Bittner and Hahn, 2013] Bittner, F. and Hahn, I. (2013). Kriging-assisted multi-objective particle swarm optimization of permanent magnet synchronous machine for hybrid and electric cars. In *Electric Machines & Drives Conference (IEMDC), 2013 IEEE International*.

[Blum and Raidl, 2016] Blum, C. and Raidl, G. R. . (2016). *Hybrid Metaheuristics: Powerful Tools for Optimization*. Artificial Intelligence: Foundations, Theory, and Algorithms. Springer International Publishing Switzerland.

[Bonnans, 2019] Bonnans, J. F. (2019). *Convex and Stochastic Optimization*. Universitext. Springer Nature Switzerland AG.

[Bowerman et al., 1995] Bowerman, R., Hall, B., and Calamai, P. (1995). A multi-objective optimization approach to urban school bus routing: Formulation and solution method. *Transportation Research Part A: Policy and Practice*, 29(2):107–123.

[Brownlee, 2012] Brownlee, J. (2012). *Clever Algorithms: Nature-Inspired Programming Recipes*. -.

[Caballero et al., 2004] Caballero, R., Cerdá, E., Muñoz, M., and Re, L. (2004). Stochastic approach versus multiobjective approach for obtaining efficient solutions in stochastic multiobjective programming problems. *European Journal of Operational Research*, 158(3):633–648.

[Calvete et al., 2016] Calvete, H., Galé, C., Iranzo, J. A., and Toth, P. (2016). A biobjective school bus routing problem. In *28th European Conference on Operational Research*, pages 1–8.

[Cao and He, 2019] Cao, S. and He, S. (2019). *Nonlinear Combinatorial Optimization*, chapter Discrete Convex Optimization and Applications in Supply Chain Management, pages 81–122. Springer Optimization and Its Applications. Springer Nature Switzerland AG.

[Caramia and Dell'Olmo, 2008] Caramia, M. and Dell'Olmo, P. (2008). *Multi-objective Management in Freight Logistics Increasing Capacity, Service Level and Safety with Optimization Algorithms*, chapter Multi-objective Optimization, pages 11–36. Springer London, London.

[Carrasqueira et al., 2015] Carrasqueira, P., Alves, M. J., and Antunes, C. H. (2015). A bi-level multiobjective pso algorithm. In *Evolutionary Multi-Criterion Optimization - 8th International Conference*.

[Carson and Maria, 1997] Carson, Y. and Maria, A. (1997). Simulation optimization: Methods and applications. In *Proceedings of the 1997 Winter Simulation Conference*.

[Chen and Lee, 2011] Chen, C.-H. and Lee, L. H. (2011). *Stochastic Simulation Optimization - An Optimal Computing Budget Allocation*, volume 1 of *Series on System Engineering and Operations Research*. World Scientific Publishing Co. Pte. Ltd.

[Cheng et al., 2011] Cheng, W., Li, X., and Liu, S. (2011). Research on the traffic control and organization coordinate applications of reversible lanes. In *3rd International Conference on Awareness Science and Technology (iCAST)*, pages 72–75.

[Chinneck, 2015] Chinneck, J. W. (2015). *Practical Optimization: A Gentle Introduction*. Systems and Computer Engineering, Carleton University, Ottawa, Canada.

[Chowdhury et al., 2000] Chowdhury, D., Santen, L., and Schadschneider, A. (2000). Statistical physics of vehicular traffic and some related systems. In *Physics Report 329*, pages 199–329.

[Corberán et al., 2002] Corberán, A., Fernández, E., Laguna, M., and Martí, R. (2002). Heuristic solutions to the problem of routing school buses with multiple objectives. *Journal of the Operational Research Society*, 53:427–435.

[Council, 1995] Council, N. R. (1995). *Expanding Metropolitan Highways:Implications for Air Quality and Energy Use Special Report 245*. The National Academies Press.

[Cristescu and Knowles, 2015] Cristescu, C. and Knowles, J. (2015). Surrogate-based multiobjective optimization: Parego update and test. In *Workshop on Computational Intelligence (UKCI)*.

[Dantzig, 1963] Dantzig, G. B. (1963). Linear programming and extensions. Technical report, United States Air Force Project RAND.

[Davins-Valldaura et al., 2017] Davins-Valldaura, J., Moussaoui, S., Pita-Gil, G., and Plestan, F. (2017). Parego extensions for multi-objective optimization of expensive evaluation functions. *Journal of Global Optimization*, 67(1):79–96.

[Davison and Hinkley, 1997] Davison, A. and Hinkley, D. (1997). *Bootstrap Methods and Their Application*. Cambridge University Press.

[de Lima, 2015] de Lima, F. M. S. (2015). *A mixed load rural school bus routing problem with heterogeneous fleet: a study for the brazilian problem*. PhD thesis, Universidade Federal de Minas Gerais, Minas Gerais, Brazil.

[Deb, 2001] Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley. ISBN: 978-0-471-87339-6.

[Deb et al., 2002] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197.

[Dellino and Meloni, 2015] Dellino, G. and Meloni, C., editors (2015). *Uncertainty Management in Simulation-Optimization of Complex Systems Algorithms and Applications*. Operations Research - Computer Science Interfaces Series. Springer.

[Dennis and Schnabel, 1996] Dennis, J. E. and Schnabel, R. B. (1996). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Society for Industrial Mathematics.

[Denysiuk and Gaspar-Cunha, 2017] Denysiuk, R. and Gaspar-Cunha, A. (2017). *Evolutionary Multi-Criterion Optimization - 9th International Conference, EMO 2017, Münster, Germany, March 19–22, 2017 Proceedings*, volume 10173 of *Lecture Notes in Computer Science*, chapter Weighted Stress Function Method for Multiobjective Evolutionary Algorithm Based on Decomposition, pages 176–190. Springer.

[Díaz-Manríquez et al., 2016] Díaz-Manríquez, A., Toscano, G., Barron-Zambrano, J. H., and Tello-Leal, E. (2016). A review of surrogate assisted multiobjective evolutionary algorithms. *Computational Intelligence and Neuroscience*.

[Díaz-Parra et al., 2014] Díaz-Parra, O., Ruiz-Vanoye, J. A., Loranca, B. B., Fuentes-Penna, A., and Barrera-Cámara, R. A. (2014). A survey of transportation problems,. *Journal of Applied Mathematics*, 2014:17.

[Dinu and Bordea, 2011] Dinu and Bordea (2011). A new genetic approach for transport network design and optimization. *Bulletin of the Polish Academy of Sciences - Technical Sciences*, 59(3):263 – 272.

[Diwekar and David, 2015] Diwekar, U. and David, A. (2015). *BONUS Algorithm for Large Scale Stochastic Nonlinear Programming Problems*. SpringerBriefs in Optimization. Springer New York Heidelberg Dordrecht London.

[Durillo et al., 2009] Durillo, J. J., Garcia-Nieto, J., Nebro, A. J., Coello, C. A. C., Luna, F., and Alba, E. (2009). Multi-objective particle swarm optimizers: An experimental comparison. In *EMO 2009*.

[Elarbi et al., 2017] Elarbi, M., Bechikh, S., Said, L. B., and Datta, R. (2017). *Recent Advances in Evolutionary Multi-objective Optimization*, chapter Multi-objective Optimization: Classical and Evolutionary Approaches, pages 1–30. Springer.

[Elefteriadou, 2014] Elefteriadou, L. (2014). *An Introduction to Traffic Flow Theory*, volume 84 of *Springer Optimization and Its Applications*. Springer New York Heidelberg Dordrecht London.

[Farahani et al., 2013] Farahani, R. Z., Miandoabchi, E., Szeto, W., and Rashidi, H. . (2013). A review of urban transportation network design problems. *European Journal of Operational Research*, 229:281 – 302.

[Ferreira and Condessa, 2012] Ferreira, J. A. and Condessa, B. (2012). Defining expansion areas in small urban settlements an application to the municipality of tomar (portugal). *Landscape and Urban Planning*, 107:281–302.

[Ferreira et al., 2010] Ferreira, J. A., Condessa, B., e Almeida, J. C., and Pinto, P. (2010). Urban settlements delimitation in low-density areas an application to the municipality of tomar (portugal). *Landscape and Urban Planning*, 97(3):156 – 167.

[Forrester, 1961] Forrester, J. W. (1961). *Industrial dynamics*. M.I.T. Press.

[Fredik, 2010] Fredik, H. (2010). Towards the solution of large-scale and stochastic traffic network design problems. Master's thesis, Uppsala Universitet.

[Friesz et al., 1992] Friesz, T. L., Cho, H.-J., Mehta, N. J., Tobin, R. L., and Anandalingam, G. (1992). A simulated annealing approach to the network design problem with variational inequality constraints. *Transportation Science*, 26(1):18–26.

[Fu and Heally, 1997] Fu, M. and Heally, K. (1997). Techniques for optimization via simulation: an experimental study on an (s, s) inventory system. *IIE Transactions*, 29(3).

[Fu, 1994] Fu, M. C. (1994). Optimization via simulation: A review. *Annals of Operations Research*, 53:199–247.

[Fu, 2002] Fu, M. C. (2002). Optimization for simulation: Theory vs. practice. *INFORMS Journal on Computing*, 14(3):192–215.

[Fu, 2015] Fu, M. C., editor (2015). *Handbook of Simulation Optimization*, volume 216 of *International Series in Operations Research & Management Science*. Springer Science+Business Media New York.

[Gallo et al., 2012] Gallo, M., DAcierno, L., and Montella, B. (2012). A meta-heuristic algorithm for solving the road network design problem in regional contexts. *Procedia - Social and Behavioral Sciences*, 54(0):84 – 95. Proceedings of EWGT2012 - 15th Meeting of the EURO Working Group on Transportation, September 2012, Paris.

[Gannouni et al., 2017] Gannouni, A., Ellaia, R., and Talbi, E.-G. (2017). Solving stochastic multiobjective vehicle routing problem using probabilistic metaheuristic. *MATEC Web Conf.*, 105:00001.

[Garcia-Nieto et al., 2013] Garcia-Nieto, J., Olivera, A., and Alba, E. (2013). Optimal cycle program of traffic lights with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 17:823 – 839.

[García-Gonzalo and Fernández-Martínez, 2012] García-Gonzalo, E. and Fernández-Martínez, J. L. (2012). A brief historical review of particle swarm optimization (pso). *Journal of Bioinformatics and Intelligent Control*, 1:3–16.

[Gazis, 2002] Gazis, D. C. (2002). *Traffic Theory*. Kluwer Academic Publishers.

[Gendreau and Potvin, 2010] Gendreau, M. and Potvin, J.-Y., editors (2010). *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*. Springer, 2 edition.

[Gilat, 2002] Gilat, M. (2002). *Coordinated Transportation and Land Use Planning in the Developing World: The Case of Mexico City*. Massachusetts Institute of Technology, Department of Urban Studies and Planning.

[Gruler, 2018] Gruler, A. C. (2018). *Simheuristics to support efficient and sustainable freight transportation in smart city logistics*. PhD thesis, Department of Computer Science, Multimedia and Telecommunications - Open University of Catalonia.

[Gutjahr, 2005] Gutjahr, W. J. (2005). Two metaheuristics for multiobjective stochastic combinatorial optimization. In Lupanov, O. B., Kasim-Zade, O. M., Chaskin, A. V., and Steinhöfel, K., editors, *Stochastic Algorithms: Foundations and Applications*, pages 116–125, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Gutjahr and Pichler, 2016] Gutjahr, W. J. and Pichler, A. (2016). Stochastic multi-objective optimization: a survey on non-scalarizing methods. *Annals of Operations Research*, 236(2):475–499.

[Haberman, 1998] Haberman, R. (1998). *Mathematical Models. Mechanical Vibrations, Population Dynamics, and Traffic Flow: An Introduction to Applied Mathematics*. Number 21 in Classics in Applied Mathematics. Society for Industrial and Applied Mathematics. This SIAM edition is an unabridged republication of the work first published by Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1977.

[Hanson and Giuliano, 2004] Hanson, S. and Giuliano, G. (2004). *The Geography of Urban Transportation*. New York: Guilford Press.

[Hao et al., 2015] Hao, X., Gen, M., Lin, L., and Süer, G. A. (2015). Effective multiobjective eda for bi-criteria stochastic job-shop scheduling problem. *Journal of Intelligent Manufacturing*, 28(3).

[Haregeweyn et al., 2012] Haregeweyn, N., Fikadu, G., Tsunekawa, A., Tsubo, M., and Meshesha, D. T. (2012). The dynamics of urban expansion and its impacts on land use land cover change and small-scale farmers living near the urban fringe: A case study of bahir dar, ethiopia. *Landscape and Urban Planning*, 106(2):149 – 157.

[Hausknecht et al., 2011] Hausknecht, M., Au, T., Stone, P., Fajardo, D., and Waller, T. (2011). "dynamic lane reversal in traffic management. In *Proceedings of IEEE Intelligent Transportation Systems Conference (ITSC)*.

[He et al., 2011] He, C., Tian, J., Shi, P., and Hu, D. (2011). Simulation of the spatial stress due to urban expansion on the wetlands in beijing, china using a gis-based assessment model. *Landscape and Urban Planning*, 101(3):269 – 277.

[Henderson et al., 2003] Henderson, D., Jacobson, S., and Johnson, A. (2003). *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, chapter The Theory and Practice of Simulated Annealing, pages 287–319. Springer, Boston, MA.

[Hong and Nelson, 2010] Hong, L. and Nelson, B. (2010). A brief introduction to optimization via simulation. In *Proceedings of the 2009 Winter Simulation Conference (WSC)*.

[Horvat and Tosic, 2012] Horvat, A. and Tosic, A. (2012). Optimization of traffic networks by using genetic algorithms. *Elektrotehniski Vestnik*, 79:197 – 200.

[Ibanez et al., 2016] Ibanez, M. L., Dubois-Lacoste, J., Caceres, L. P., Stutzle, T., and Birattari, M. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*.

[Jia et al., 2019] Jia, G.-L., Ma, R.-G., and Hu, Z.-H. (2019). Review of urban transportation network design problems based on citespace. *Mathematical Problems in Engineering*.

[Jiang et al., 2014] Jiang, S., Ong, Y., Zhang, J., and Feng, L. (2014). Consistencies and contradictions of performance metrics in multiobjective optimization. *IEEE Transactions on Cybernetics*, 44(12):2391–2404.

[Juan et al., 2015] Juan, A. A., Faulin, J., Grasman, S. E., Rabe, M., and Figueira, G. (2015). A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives*, 2:62–72.

[Juan et al., 2018] Juan, A. A., Kelton, W. D., Currie, C. S. M., and Faul, J. (2018). Simheuristics applications: Dealing with uncertainty in logistics, transportation, and other supply chain areas. In *Proceedings of the 2018 Winter Simulation Conference*.

[Kaliszewski et al., 2016] Kaliszewski, I., Miroforidis, J., and Podkopaev, D. (2016). *Multiple Criteria Decision Making by Multiobjective Optimization - A Toolbox*, volume 242 of *International Series in Operations Research & Management Science*. Springer.

[Kamalanathsharma, 2011] Kamalanathsharma, R. K. (2011). Reversible lanes: State of implementation on a global level. In *Annual ATES Student Paper Competition*.

[Kang et al., 2018] Kang, B. G., Choi, S. H., Kwon, S. J., Lee, J. H., and Kim, T. G. (2018). Simulation-based optimization on the system-of-systems model via model transformation and genetic algorithm: A case study of network-centric warfare. *Complexity*.

[Karoonsoontawong, 2010] Karoonsoontawong, A. (2010). Genetic algorithms for dynamic contraflow problem. *Journal of Society for Transportation and Traffic Studies*, 1(2):1–17.

[Kepaptsoglou and Karlaftis, 2009] Kepaptsoglou, K. and Karlaftis, M. (2009). Transit route network design problem. *Journal of Transportation Engineering*, 35(8):491–505.

[Kerner, 2009] Kerner, B. S. (2009). *Introduction to Modern Traffic Flow Theory and Control - The Long Road to Three-Phase Traffic Theory*. Springer-Verlag Berlin Heidelberg.

[Kim and Weck, 2006] Kim, L. and Weck, O. D. (2006). Adaptive weighted sum method for multi-objective optimization: a new method for pareto front generation. *Structural and Multidisciplinary Optimization*, 2(31):105–116.

[Kleijnen, 2009] Kleijnen, J. P. (2009). Kriging metamodeling in simulation: A review. *European Journal of Operational Research*, 192(3):707 – 716.

[Kleijnen, 2015] Kleijnen, J. P. (2015). *Design and Analysis of Simulation Experiments*, volume 230 of *International Series in Operations Research & Management Science*. Springer International Publishing Switzerland, second edition edition.

[Kleijnen, 2019] Kleijnen, J. P. C. (2019). *High-Performance Simulation-Based Optimization*, volume 833 of *Studies in Computational Intelligence*, chapter Simulation Optimization Through Regression or Kriging Metamodels, pages 115–136. Springer Nature Switzerland AG.

[Krauss, 1998] Krauss, S. (1998). *Microscopic Modeling of Traffic Flow: Investigation of Collision Free Vehicle Dynamics*. Hauptabteilung Mobilität und Systemtechnik des DLR Köln.

[Krige, 1951] Krige, D. (1951). A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Chemical, Metallurgical and Mining Society of South Africa*, 52(6):119–139.

[Kumar et al., 2010] Kumar, M., Husain, M., Husain, M., Upreti, N., and Gupta, D. (2010). Genetic algorithm: Review and application. *Journal of Information & Knowledge Management*.

[Lambert and Wolshon, 2010] Lambert, L. and Wolshon, B. (2010). Characterization and comparison of traffic flows on reversible roadroad. *journal of Advanced Transportation*, 44(2):113–122.

[Laporte, 1988] Laporte, G. (1988). *Vehicle Routing: Methods and Studies*, chapter Location-routing problems, pages 163–198.

[Li et al., 2008] Li, M., Li, G., and Azarm, S. (2008). A kriging metamodel assisted multi-objective genetic algorithm for design optimization. *Journal of Mechanical Design*.

[Li and Sun, 2012] Li, Y. and Sun, D. (2012). Microscopic car-following model for the traffic flow: the state of the art. *Journal of Control Theory and Applications*, 10(2):133–143.

[Li et al., 2010] Li, Y., Zhu, X., Sun, X., and Wang, F. (2010). Landscape effects of environmental impact on bay-area wetlands under rapid urban expansion and development policy: A case study of lianyungang, china. *Landscape and Urban Planning*, 94(3,4):218 – 227.

[Liefooghe and Derbe, 2016] Liefooghe, A. and Derbe, B. (2016). A correlation analysis of set quality indicator values in multiobjective optimization. In *Genetic and Evolutionary Computation Conference (GECCO 2016)*.

[Liu and Shi, 2019] Liu, L. and Shi, L. (2019). Simulation optimization on complex job shop scheduling with non-identical job sizes. *Asia-Pacific Journal of Operational Research*, 36(5).

[Luenberger and Ye, 2008] Luenberger, D. G. and Ye, Y. (2008). *Linear and nonlinear programming*. Springer Science+Business Media, LLC.

[Luke, 2013] Luke, S. (2013). Essentials of metaheuristics.

[Matheron, 1963] Matheron, G. (1963). Principles of geostatistics. *Economic Geology*, 58(8).

[Mersmann, 2014] Mersmann, O. (2014). *mco: Multiple Criteria Optimization Algorithms and Related Functions*. R package version 1.0-15.1.

[Nagel and Schreckenberg, 1992] Nagel, K. and Schreckenberg, M. (1992). A cellular automaton model for freeway traffic. *Journal de Physique I*, 2:2221–2229.

[Nayati, 2008] Nayati, M. A. K. (2008). *School Bus Routing and Scheduling using GIS*. VDM Verlag Dr. Müller.

[Newton and Thomas, 1969] Newton, R. M. and Thomas, W. H. (1969). Design of school bus routes by computer. *Socio Economic Planning Science*, 1(3):75–85.

[Nguyen et al., 2014] Nguyen, A. T., Reiter, S., and Rigo, P. (2014). A review on simulation-based optimization methods applied to building performance analysis. *Applied Energy*, 113:1043–1058.

[Nocedal and Wright, 2006] Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer Science+Business Media, LLC, second edition edition.

[Park and Kim, 2010] Park, J. and Kim, B.-I. (2010). The school bus routing problem: A review. *European Journal of Operational Research*, 202(2):311 – 319.

[Park and Kim, 2013] Park, J. and Kim, B.-I. (2013). Model and algorithm for solving school bus problem. *Journal of Emerging Trends in Computing and Information Sciences*, 4(8):311–319. ISSN 2079-8407.

[Patriksson, 1994] Patriksson, M. (1994). *The Traffic Assignment Problem - Models and Methods*. Topics in Transportations Series.

[Prasetyo et al., 2011] Prasetyo, D. H., Muhamad, J., and Fauzi, R. (2011). Modelling school bus in favor of needy student: The conceptual framework. *International Journal of Social Science and Humanity*, 1(1):–.

[Qing and Ziyou, 2014] Qing, L. and Ziyou, G. (2014). Managing rush hour congestion with lane reversal and tradable credits. *Mathematical Problems in Engineering*, 2014.

[Rabadi, 2016] Rabadi, G., editor (2016). *Heuristics, Metaheuristics and Approximate Methods in Planning and Scheduling*, volume 236 of *International Series in Operations Research and Management Science*. Springer.

[Ratner and Goetz, 2013] Ratner, K. A. and Goetz, A. R. (2013). The reshaping of land use and urban form in denver through transit-oriented development. *Cities*, 30(0):31 – 46.

[Riquelme et al., 2015] Riquelme, N., Lucken, C. V., and Baran, B. (2015). Performance metrics in multi-objective optimization. In *2015 XLI Latin American Computing Conference (CLEI)*.

[Russell and Morrel, 1986] Russell, R. A. and Morrel, R. B. (1986). Routing special-education school buses. *Interfaces*, 16(5):56–64.

[Ruszczynski and Shapiro, 2009] Ruszczynski, A. and Shapiro, A. (2009). *Lectures on stochastic programming: modeling and theory*, chapter Stochastic Programming Models, pages 1–26. Society for Industrial and Applied Mathematics and the Mathematical Programming Society.

[Ryu et al., 2009] Ryu, J., Kim, S., and Wan, H. (2009). Pareto front approximation with adaptive weighted sum method in multiobjective simulation optimization. In *Proceedings of the 2009 Winter Simulation Conference (WSC)*, pages 623–633.

[Sacks et al., 1989] Sacks, J., Welch, W., Mitchell, T., and Winn, H. (1989). Design and analysis of computer experiments. *Statistical Science*, 4(4).

[Schittekat et al., 2013] Schittekat, P., Kinable, J., Sörensen, K., Sevaux, M., Spieksma, F., and Springael, J. (2013). A metaheuristic for the school bus routing problem with bus stop selection. *European Journal of Operational Research*, 229(2):518–528.

[Schittekat et al., 2006] Schittekat, P., Sevaux, M., and Sorensen, K. (2006). A mathematical formulation for a school bus routing problem. In *Service Systems and Service Management, 2006 International Conference on*, volume 2, pages 1552–1557.

[Schmaranzer et al., 2019] Schmaranzer, D., R. Braune, R., and Doerner, K. F. (2019). Multi-objective simulation optimization for complex urban mass rapid transit systems. *Annals of Operations Research*.

[Schrijver, 1986] Schrijver, A. (1986). *Theory of Linear and Integer Programming*. John Wiley & Sons Ltd.

[Shrestha and Mahmood, 2016] Shrestha, A. and Mahmood, A. (2016). Improving genetic algorithm with fine-tuned crossover and scaled architecture. *Journal of Mathematics*, 2016.

[Shukla et al., 2015] Shukla, A., Pandey, H. M., and Mehrotra, D. (2015). Comparative review of selection techniques in genetic algorithm. In *2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)*, pages 515–519.

[Song, 2013] Song, D.-P. (2013). *Optimal Control and Optimization of Stochastic Supply Chain Systems*. Springer London Heidelberg New York Dordrecht.

[Sánchez et al., 2010] Sánchez, D., Amodeo, L., and Prins, C. (2010). *Artificial Intelligence Techniques for Networked Manufacturing Enterprises Management*, chapter Meta-heuristic

Approaches for Multi-objective Simulation-based Optimization in Supply Chain Inventory Management, pages 249–269. Springer Series in Advanced Manufacturing. Springer London.

[Team, 2018] Team, R. C. (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

[Tian et al., 2018] Tian, Y., Ye, B., Estupiñá, M. S., and Wan, L. (2018). Stochastic simulation optimization for route selection strategy based on flight delay cost. *Asia-Pacific Journal of Operational Research*, 35(6).

[Todoroki and Sekishiro, 2008] Todoroki, A. and Sekishiro, M. (2008). Modified efficient global optimization for a hat-stiffened composite panel with buckling constraint. *AIAA Journal*, 46(9):2257–2264.

[TRB, 2016] TRB (2016). *Highway Capacity Manual, Sixth Edition: A Guide for Multimodal Mobility Analysis*. Transportation Research Board - National Academies of Science - USA.

[Treiber and Kesting, 2013] Treiber, M. and Kesting, A. (2013). *Traffic Flow Dynamics - Data, Models and Simulation*. Springer-Verlag. Translated by Martin Treiber and Christian Thiemann.

[Tricoire et al., 2012] Tricoire, F., Graf, A., and Gutjahr, W. J. (2012). The bi-objective stochastic covering tour problem. *Computers & Operations Research*, (39):1582–1592.

[Tsou, 2013] Tsou, C.-S. (2013). *nsga2R: Elitist Non-dominated Sorting Genetic Algorithm based on R*. R package version 1.0.

[Umbarkar and Sheth, 2015] Umbarkar, A. and Sheth, P. (2015). Crossover operators in genetic algorithms: A review. *ICTACT Journal on Soft Computing*, 6(1).

[Vanderbei, 2008] Vanderbei, R. J. (2008). *Linear Programming - Foundations and Extensions*. International Series In Operations Research and Management Sciences. Springer, 3 edition.

[Voutchkov and Keane, 2006] Voutchkov, I. and Keane, A. (2006). Multiobjective optimization using surrogates. In *Proceedings of the International Conference on Adaptive Computing in Design and Manufacture*.

[Waller et al., 2006] Waller, S. T., Mouskos, K. C., Kamaryiannis, D., and Ziliaskopoulos, A. K. (2006). A linear model for the continuous network design problem. *Computer-Aided Civil and Infrastructure Engineering*, 21:334–345.

[Woensel and Vandaele, 2007] Woensel, T. and Vandaele, N. (2007). Modeling traffic flows with queueing models: A review. *Asia-Pacific Journal of Operational Research*, 24(4):435–461.

[Xiao et al., 2006] Xiao, J., Shen, Y., Ge, J., Tateishi, R., Tang, C., Liang, Y., and Huang, Z. (2006). Evaluating urban expansion and land use change in shijiazhuang, china, by using gis and remote sensing. *Landscape and Urban Planning*, 75(1,2):69 – 80.

[Xu et al., 2015] Xu, J., Huang, E., Chen, C.-H., and Lee, L. H. (2015). Simulation optimization: A review and explorationin the new era of cloud computing and big data. *Asia-Pacific Journal of Operational Research*.

[Yang and Bell, 1998] Yang, H. and Bell, M. G. H. (1998). Models and algorithms for road network design: a review and some new developments. *Transport Reviews*, 18(3):257–278.

[Zhao et al., 2014] Zhao, J., Ma, W., Liu, Y., and Yang, X. (2014). Integrated design and operation of urban arterials with reversible lanes. *Transpormetrica B: Transport Dynamics*, 0:1–21.

[Zitzler et al., 2007] Zitzler, E., Brockhoff, D., and Thiele, L. (2007). The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration. In Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., and Murata, T., editors, *Evolutionary Multi-Criterion Optimization*, pages 862–876, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Zitzler and Thiele, 1999] Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271.