UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE

"ESTUDIO DE *FITNESS LANDSCAPES* EN PROBLEMAS DE SINTONIZACIÓN DE PARÁMETROS MEDIANTE REDES DE ÓPTIMOS LOCALES"

GERMÁN MARCELO TREIMUN COSTA

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Dra. Elizabeth Montero Ureta
Profesor Correferente: Dra. María Cristina Riff

Noviembre - 2018

DEDICATORIA

Dedico este trabajo a mi madre, por el amor incondicional me ha brindado y por apoyarme siempre en la búsqueda de lo que me haga feliz. A mi padre, por todo el trabajo y sacrificio que ha hecho por sus hijos y familia, por todos los valores que me inculcó y me hicieron la persona que soy hoy en día. Finalmente, a mis hermanos Camila y Joe, por todas las alegrías que traen a mi vida.

AGRADECIMIENTOS

Quiero agradecer a la profesora Elizabeth Montero, quien me brindo este tema, quien con su gran conocimiento, apoyo, carisma y buena onda me guío correctamente, es una gran docente y sin ella este trabajo no existiría. Al profesor Nicolás Rojas por todo el apoyo y ayuda que nos brindo durante el trabajo y su increíble buena onda. A la profesora Gabriela Ochoa, por toda la ayuda brindada y dejarnos ser parte de su trabajo con redes de óptimos locales. Finalmente a la profesora María Cristinca Riff, por que sin ella y su calidad docente, nunca hubiera nacido mi interés en esta área, en un momento en que aún dudaba sobre los caminos a seguir en mi vida.

Además, este trabajo no hubiera sido posible sin todas esas personas que estuvieron presentes mi vida universitaria, por ello quiero mencionar a cada uno de ellos, por que significan mucho para mi.

A mis amigos de universidad quienes hicieron de esta etapa, inolvidable. Mis hermanos: Axel Simonsen, Jorge Basualdo, Fabian Viani, Cristian Valles, Martin Villanueva, Maximiliano Osorio, Germán ortiz, Tomás Gonzalez y Juanjo Ibarra. El team CC: Pablo Ibarra, Ignacio Araya y Ariel Sanhueza. Mis mechones favoritos: Sebastian Jorquera y Cesar Rojas. Mis amigos de la pensión: Diego Gajardo, Fernando Fuenzalida, Esteban y Roberto Opazo. Mis amigos de la FESW: Gabriela Gonzalez y Luis patito Ramirez. Y muchos amigos más que no tiene una clasificación pero no por ello menos importancia en mi vida: Marcelo Ávalos, Oscar Duarte, Renata Mella, Gonzalo Moya, Joey Koro, Rodrigo Cisternas, Nicolás Silva, Camilo Valenzuela, Carlos Bugueño y Cristopher Barraza.

Agradezco inmensamente a Bárbara Navarrete Figueroa, quien me ha entregado 5 años de su vida llenos de amor, de una compañía incondicional y un apoyo tan inmenso, que sin ella probablemente no estaría escribiendo estos agradecimientos ahora mismo. Bárbara te amo inmensamente.

Agradezco a mi hermano Diego Reyes Delgado, por ser un increíble amigo y el inmenso apañe que tuvimos este año, que los PES nunca mueran hermano.

Agradezco a Roberto Fuentes Zenteno, por ser el mejor amigo que pude tener en mi etapa universitaria y espero nunca cambies esa forma única y tan especial que tienes de ser.

Quiero mencionar que este trabajo de memoria no podria haber sido terminado sin la motivante ayuda de las listas de reproducción : “Playlist de los hombre musculos” de Martín Villanueva, “Cumbias Supremas” de Axel Simonsen y Todos los openings de Naruto.

Finalmente, Hala Madrid y nada más!

RESUMEN

Resumen— El diseño de meta-heurísticas ha demostrado ser una herramienta muy útil en la optimización, pero para alcanzar el mejor rendimiento pero para alcanzar el mejor rendimiento es importante considerar/tener en cuenta también los valores de los parámetros. La sintonización de parámetros resulta ser un proceso que consume bastante tiempo, realizar aportes que ayuden a mejorar este campo no es irrelevante. Un área sin estudiar, es cómo los algoritmos de sintonización trabajan sobre el espacio de búsqueda. En este trabajo se presenta un enfoque basado en redes de óptimos locales (LON), para estudiar los *fitness landscapes* de dos algoritmos de sintonización elegidos. Luego de obtener las redes para cada algoritmos mediantes métricas y visualización se realiza una comparación de los resultados obtenidos sobre los *fitness landscapes* de cada uno.

Palabras Clave— Sintonización de parámetros, redes de óptimos locales, espacio de búsqueda, ParamILS, Evoca.

ABSTRACT

Abstract— The design of meta-heuristic has demonstrated to be a very useful tool in optimization, but to reach the best performance it's important to consider and have in mind the values of the parameters. The tuning parameter turns out to be a process that it consumes a lot of time; to realize contributions that help to improve this field is not irrelevant. An area without studying is how the algorithms of tuning work on the space of search. In this work, an approach based on local optima networks (LONs) is presented, to study the fitness landscapes of two algorithms of tuning chosen. After obtaining the networks for every algorithm by metrics and visualization, a comparison is made between the results obtained on the fitness landscapes of each one.

Keywords— Fitness landscapes, Parameter tuning, local optima network, ParamILS, Evoca.

GLOSARIO

Basin: Conjunto de nodos en un *fitness landscape* que se agrupan en base al movimiento en el vecindario y en calidad.

Budget: Cantidad de ejecuciones máximas entregadas a un algoritmo de sintonización.

EVOCA: *Evolutionary Calibrator*, algoritmo de sintonización evolutivo.

Fitness Landscape: Espacio de búsqueda asociado a la calidad de cada solución en el espacio.

Fitness: Calidad de una solución o configuración de parámetros

First improvement: primera mejora, criterio para aceptar movimiento en vecindario de soluciones.

Friedman two-way analysis of variance by ranks: Prueba estadística para encontrar diferencias en resultados a raiz de multiples intentos.

Hill Climbing: Técnica búsqueda local.

Latin Hipercube Sampling: Método estadístico para generar una muestra casi aleatoria de valores de parámetros de una distribución multidimensional.

LON: *Local Optima Network*, red de óptimos locales que representa de forma comprimida un *fitness landscape*

ParamILS: *Parameter Iterated Local Search Method*, algoritmo de sintonización basado en búsqueda local iterativa.

Wheel-Crossover: Ruleta de cruzamiento.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El diseño de meta-heurísticas es una herramienta bastante poderosa a la hora de resolver problemas de optimización altamente complejos, pero a la vez diseñar una meta-heurística resulta ser un proceso de alta dificultad que consume mucho tiempo. Como lo ideal es obtener la mejor solución, el enfoque propuesto por el diseñador de la meta-heurística debe tomar en cuenta tres factores que son de suma importancia para obtener el mejor rendimiento: (1) Las instancias del problema, (2) las semillas del problema y finalmente (3) Los valores de los parámetros asociados al problema o el algoritmo que lo resuelve. (Ver figura 1)

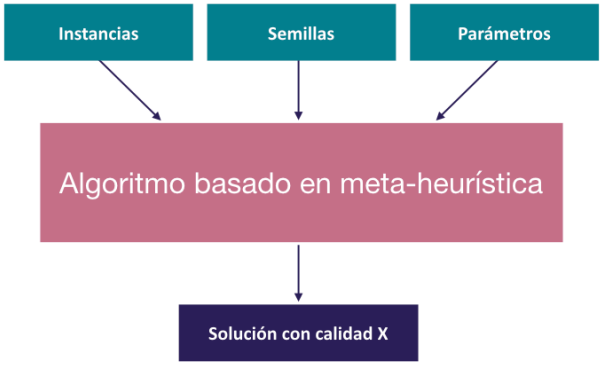


Figura 1: Diagrama básico rendimiento de meta-heurística
Fuente: Elaboración propia

De los tres factores mencionados el ultimo es de suma relevancia, esto porque obtener los mejores valores de los parámetros es un problema de optimización combinatoria en si. Conocido como el problema de sintonización de parámetros, lo que se busca resolver es encontrar los mejores valores posibles de los parámetros para obtener el mejor rendimiento de un algoritmo objetivo. Obtener una buena configuración de parámetros puede resultar un proceso costoso computacionalmente en terminos de tiempo, dado que se necesita realizar una gran cantidad de ejecuciones del algoritmo objetivo. A la fecha existen muchos enfoques propuestos para sintonizar parámetros, estos pueden ser clasificados en 4 categorías:

- **Muestreo**: Reducir el esfuerzo descartando configuraciones de mala calidad respecto a un diseño factorial.
- **Basado en modelos**: Construir modelos de calidad del espacio de búsqueda en los valores de parámetros.
- **Basado en selección**: Identificar el mejor conjunto de configuraciones desde un conjunto dado realizando un número reducidos de pruebas.

- **Basado en Búsqueda**: Realizar el proceso de sintonización como un proceso de búsqueda en un problema de optimización.

existen de muchos tipos y uno de estos tipos es la sintonización mediante otro algoritmo, el cual por ejemplo, puede estar basado en una búsqueda local.

A la fecha, en el campo de la sintonización de parámetros existe un área que casi no ha sido investigada, y esta corresponde a como trabajan los algoritmos de sintonización sobre el espacio de búsqueda, específicamente, sobre los *fitness landscapes*, los cuales corresponden a una representación del espacio de búsqueda asociado a la calidad de cada solución del espacio. Una analogía bastante útil para entender los *fitness landscapes* es pensar en ellos como si fueran montañas o cuencas (ambos ejemplos son útiles) donde el punto más alto o más bajo respectivamente corresponde al óptimo global.

En esta memoria se presenta un nuevo aporte al campo de la investigación asociado a la sintonización de parámetros. El enfoque propuesto consiste en realizar un estudio de los *fitness landscapes* que generan los algoritmos sintonizadores, para realizar el estudio se propone integrar la estrategia de visualización de redes de óptimos locales a los algoritmos sintonizadores de parámetros ParamILS y Evoca.

Este documento está organizado de la siguiente manera. Inicialmente, en la Sección 1 se define el problema a abordar en la presente memoria y los objetivos tanto generales como específicos. Luego, en la Sección 2 se realiza una revisión de la literatura asociada a la sintonización de parámetros, sus algoritmos y los algoritmos elegidos a trabajar en esta memoria. Además se definen las redes de óptimos locales utilizadas para realizar el estudio de los *fitness landscapes*. En la Sección 3 se presenta la propuesta de solución, sus fundamentos y la metodología aplicada para resolver el problema. En la Sección 3.3 se describen los experimentos realizados. Posteriormente, en la Sección 4.3 se muestran los resultados obtenidos de las pruebas realizadas, y se realiza un análisis de lo obtenido. Finalmente, en la Sección 5.3 se presentan las conclusiones de trabajo realizado.

CAPÍTULO 1

DEFINICIÓN DEL PROBLEMA

La sintonización de parámetros, en algunos casos conocida también como meta-optimización, corresponde a la búsqueda de los mejores valores de parámetros para un algoritmo que resuelve un problema de optimización combinatoria. Data desde los años 70 cuando Mercer y Sampson [13] intentaron encontrar la sintonización de parámetros óptima para un algoritmo genético. Los algoritmos de sintonización de parámetros han demostrado ser capaces de realizar significativas mejoras, pues los parámetros estos afectan a la solución, ya que, cada problema y las instancias de estos tienen parámetros propios, ya sean numéricos, categóricos o condicionales. Los parámetros pueden estar interrelacionados o incluso pueden variar durante la ejecución, por lo cual encontrar los mejores valores resulta ser un proceso complejo computacionalmente y que toma bastante tiempo.

Dada la importancia que tiene realizar una buena sintonización parámetros, es fundamental continuar realizando investigaciones y/o aportes que ayuden a comprender de mejor manera el proceso de sintonización, para así, realizar mejoras a los algoritmos ya propuestos o presentar nuevos enfoques. Existen variadas investigaciones y enfoques propuestos por distintos autores, los cuales son analizados con más detalle en las sección 2.2. Los métodos a abordar en esta memoria corresponden a ParamILS y Evoca.

El método ParamILS [10] funciona como una búsqueda local iterativa (ILS) que se mueve en un espacio de configuraciones discreto. Comienza con una configuración de parámetros por defecto, y mediante un proceso de *first improvement* mejora iterativamente la configuración obtenida. ParamILS además puede realizar una cantidad de perturbaciones y cuenta con una probabilidad de reiniciar la búsqueda. Evoca [20] por otro lado, busca de manera simple ayudar a los diseñadores de meta-heurísticas poco experimentados en sintonización de parámetros a obtener soluciones de buena calidad, sin necesidad de que los diseñadores deban adentrarse en el campo de la sintonización. Evoca es un algoritmo evolutivo para el cual se realiza un cálculo de su población en base a los parámetros a sintonizar y el dominio de estos.

El desafío a abordar en la presente memoria, con motivo de realizar aportes en el campo de la sintonización de parámetros, es realizar una mayor comprensión de los *fitness landscapes* sobre los que trabajan los algoritmos sintonizadores, visualizar cómo los algoritmos se mueven sobre el espacio de búsqueda, y responder preguntas como: ¿Qué tan difícil es para el algoritmo encontrar óptimos locales y/o globales?, ¿Se encuentran los óptimos agrupados?, ¿Existe correlación entre calidad y un óptimo local más recurrente? entre varias otras interrogantes o inclusive encontrar respuestas a preguntas que aún no han sido planteadas, puesto que el estudio de los *fitness landscapes* en la sintonización de parámetros, cuenta con una casi nula investigación. El enfoque más

cercano es el presentado recientemente por Pushak y Hoos en [19], donde los investigadores logran concluir que los espacios de búsqueda no son tan complejos como se cree.

Una gran herramienta para analizar y comprender los *fitness landscapes* es realizar una interpretación de ellos como grafos o redes de óptimos locales (LON). Este enfoque presentado por Ochoa et al. [17] el año 2008 se encuentra inspirado en el estudio de *energy landscapes* en el campo de la física química. Una red de óptimos locales representa los *fitness landscapes* mediante grafos en donde los nodos corresponden a los óptimos locales y los arcos se modelan acorde a la asociación que se quiere representar entre éstos. El primer enfoque de LONs propuesto fue representar los óptimos locales encontrados para *NK landscapes* mediante una búsqueda local iterativa (ILS), asociados entre si según la probabilidad de encontrarse en el vecindario de un óptimo local. A la fecha se han propuesto nuevos enfoques de LONs donde se varía la asociación entre los nodos ya sea mediante perturbaciones para escapar de óptimos locales o inclusive el cruzamiento entre individuos de una población.

Estudiar y entender como funcionan los algoritmos sintonizadores sobre los *fitness landscapes*, observar sus estructuras y encontrar respuesta a muchas interrogantes sobre estos mediante el uso de LONs resulta inmensamente motivante, puesto que es un área desconocida y los resultados que se puedan encontrar pueden hacer la diferencia a futuro en el campo de la optimización combinatoria.

1.1. Objetivos

El objetivo general de este trabajo es proponer, desarrollar y validar un acercamiento entre redes de óptimos locales y la sintonización de parámetros para el estudio de los *fitness landscapes*. Identificar características de los *fitness landscapes*, las cuales puedan influenciar en la búsqueda del algoritmo sintonizador. Utilizar el conocimiento obtenido para predecir el rendimiento del algoritmo o para mejorar el diseño de este mismo.

A continuación se presentan los objetivos específicos del presente trabajo:

- Realizar una integración entre algoritmos de sintonización de parámetros y redes de óptimos locales.
- Obtener información (métricas) relevantes sobre el proceso de sintonización de cada algoritmo.
- Observar el comportamiento de cada algoritmo de sintonización al variar la dificultad del problema que resuelve el algoritmo objetivo.
- Obtener mediante la información y visualización de las redes de óptimos locales el *fitness landscape* de cada algoritmo de sintonización.
- Realizar una comparación de los *fitness landscapes* obtenidos por ParamILS y Evoca bajo distintas condiciones.

CAPÍTULO 2

MARCO CONCEPTUAL

Antes de poder entrar de lleno en la propuesta de solución de esta memoria es necesario realizar una revisión a la literatura. Se debe contextualizar sobre la sintonización de parámetros, los tipos que existen y describir algunos de los sintonizadores más conocidos. A su vez es necesario realizar una definición formal de los algoritmos de sintonización elegidos a utilizar en el presente trabajo (Evoca y ParamILS) para entender en que consisten y cómo funcionan. Posteriormente se realiza una definición de las redes de óptimos locales (LON) que serán usadas para el estudio de los *fitness landscapes* y se indaga en sus características y avances. Para finalizar esta sección se presenta un estudio del trabajo relacionado por Pushak y Hoss [19], ya que es la única investigación encontrada en el area del trabajo presentado.

2.1. Sintonización de parámetros

De manera simple la sintonización de parámetros puede ser declarada informalmente de la siguiente manera: dado un escenario de configuraciones compuesto por un algoritmo, un conjunto de parámetros para el algoritmo y un conjunto de datos de entrada, se busca encontrar los mejores valores de los parámetros para los cuales el algoritmo obtenga el mejor rendimiento con los datos de entrada dados. En la figura 2 se muestra un diagrama básico del proceso de sintonización.

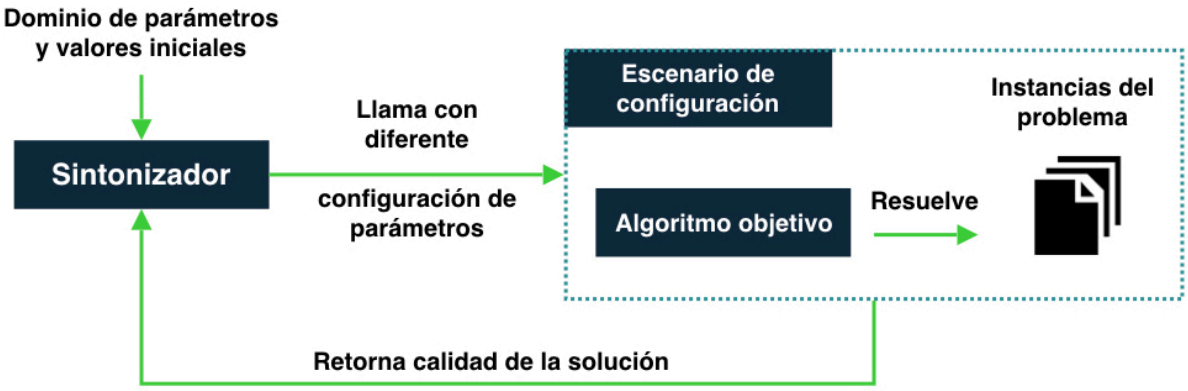


Figura 2: Diagrama básico del proceso de sintonización
Fuente: [9]

Existen distintos tipos de sintonización de parámetros [14] ; existe la sintonización hecha a mano, en donde el diseñador observa el rendimiento del algoritmo y va modificando los valores de los parámetros para obtener un resultado más óptimo. La sintonización por analogía, por otro lado, consiste en la idea de seguir pautas creadas por autores reconocidos. Otro tipo de sintonización es la basada en el diseño experimental, en la cual como su nombre lo indica, utiliza el diseño experimental para determinar aquellos valores que presentan mejor desempeño en base a algún criterio estadístico. Existen también los métodos de sintonización basados en búsqueda, como lo son ParamILS Y Evoca. Finalmente se encuentran los métodos híbridos, los cuales combinan métodos anteriormente nombrados.

2.2. Algoritmos de sintonización

A continuación, se describe algunos de los principales métodos en la literatura, junto con los enfoques seleccionados por ellos, y los resultados que obtuvieron.

2.2.1. F-Race, Revac y SPO

En el año 2014, Montero et al. [14] presentaron un documento como guía para principiantes en los métodos de sintonización de parámetros, haciendo mención en los tipos de sintonización y además realizando un buen resumen de los métodos F-Race [3], Revac [15] y SPO [2], por lo cual en este documento se utilizará el trabajo hecho en [14] para describir y comparar los métodos recientemente mencionados:

- **F-Race:** Este algoritmo, presentado por Birattari et al.[3] en el año 2002, consiste en un proceso iterativo, que en cada paso evalúa un conjunto de configuraciones candidatas, en una nueva instancia del problema o una nueva semilla y luego se elimina del conjunto la configuración que muestre el peor rendimiento estadístico.

  F-Race utiliza *Friedman two-way analysis of variance by ranks* () para comparar el rendimiento entre el conjunto de configuraciones de parámetros candidatas. Este método define tres parámetros: la cantidad de ejecuciones sin eliminación r, el nivel de confianza de las pruebas de hipótesis α, y el número máximo de ejecuciones del algoritmo sintonizado *budget*.

  F-Race comienza con un conjunto C de configuraciones de parámetros. El algoritmo requiere la definición de un conjunto discreto de valores para cada parámetro a sintonizar. Así C esta compuesto por el conjunto factorial de todos los valores de parámetros. Luego el algoritmo realiza r ejecuciones de cada parámetro en C para recolectar información suficiente antes de realizar una eliminación, la información se almacena en un arreglo de costos de cada parámetro.

Posteriormente, F-Race selecciona de manera aleatoria una instancia para continuar con las comparaciones de rendimiento entre las configuraciones de parámetros candidatas en C. F-Race evalúa el algoritmo sintonizado en la instancia y agrega esta información al arreglo de costos de cada parámetro. El algoritmo realiza la prueba de Friedman, y si se rechaza la hipótesis nula, se puede interpretar que al menos una configuración candidata tiende a mostrar un mejor rendimiento que al menos otra configuración. Se hacen comparaciones por pares entre las configuraciones de los parámetros en donde los que tienen un rendimiento estadísticamente inferior se eliminan del conjunto de configuraciones candidatas. F-Race termina cuando queda solo una configuración candidata o cuando se alcanza el máximo de ejecuciones (*budget*) predefinido.

- **Revac:** Este método es definido como un algoritmo de distribución [18]. Revac trabaja con una población, la cual puede ser entendida como una matriz con M filas, en la cual cada fila es una configuración de parámetros y contiene k elementos, correspondientes a los k parámetros a sintonizar. Para cada parámetro respectivo, Revac inicia la búsqueda con una distribución uniforme de los valores dentro de un rango dado. En cada paso, mediante el uso de operadores de transformación especialmente diseñados, se reduce el rango de valores mencionado para cada parámetro.

  Por otro lado, si un parámetro es más relevante que otro se establece en base a la entropía de los parámetros, de manera que a más alta entropía, menos relevancia tiene, y si tiene una entropía baja, el parámetro es más relevante.

  El algoritmo se inicia con una población aleatoria de M configuraciones de parámetros, se evalúa cada configuración de parámetros considerando solo una semilla aleatoriamente, en cada iteración solo se crea una nueva configuración de parámetros mediante un cruzamiento multi-parental de mutación. El cruzamiento multi-parental considera las mejores $N < M$ configuraciones de parámetros. El operador de mutación, calcula para cada parámetro un intervalo de mutación, luego se selecciona un valor aleatorio desde el intervalo de mutación y se asigna a la configuración hija.

- **SPO:** Presentado por T.Bartz-Bielstein et al. [2] en el año 2005, SPO consiste en un método que realiza iterativamente un análisis experimental de un conjunto de puntos de diseño (configuraciones de parámetros), luego estima el rendimiento del algoritmo objetivo mediante un modelo de proceso estocástico, y finalmente determina los nuevos puntos de diseño.

  El algoritmo parte con la construcción de un conjunto de puntos de diseño mediante un diseño de *Latin Hipercube Sampling*[1] (LHS) en el rango de valores definido para los parámetros. Los n puntos de diseño se eligen aleatoriamente de n intervalos, los cuales a su vez se obtienen dividiendo n veces el rango de cada parámetro.

[1] método estadístico para generar una muestra casi aleatoria de valores de parámetros de una distribución multidimensional

Los autores realizaron experimentos con los 3 métodos descritos junto a ParamISL (ver sección 2.4) y compararon los resultados, obteniendo ventajas y desventajas de cada método respecto a la calidad de los resultados, el esfuerzo que requieren los métodos, la información entregada, la usabilidad, qué tan amigables son con el usuario y el comportamiento bajo distintos escenarios.

Se utilizó un algoritmo genético estándar (SGA) para optimizar ocho funciones (Sphere, Rastrigin, Griewank, entre otras), con elitismo siempre presente, el criterio de *fitness* para SGA es la cantidad de evaluaciones para encontrar el óptimo, el cual está fijado para cada una de las funciones. Cada ejecución del algoritmo genético estándar realiza un máximo de 10^5 evaluaciones. Para los experimentos los autores de este documento plantearon 3 diferentes escenarios, cada uno con sus respectivas instancias de entrenamiento y de prueba. Finalmente en las conclusiones de [14], se realiza un resumen de los contenidos vistos en el documento, aquí los autores mencionan que dependiendo del enfoque se debe elegir el algoritmo adecuado.

Sobre la naturaleza de los parámetros, las conclusiones dicen que tanto F-Race, ParamILS y SPO pueden buscar parámetros numéricos y categóricos, mientras que Revac no realiza búsqueda de parámetros categóricos, por otro lado los requerimientos de entrada de Revac al igual que SPO solo necesitan la definición del rango de valores, mientras que en este aspecto, ParamILS necesita una definición discreta de los valores, al igual que F-Race, donde para este último resulta ser una tarea bastante ardua.

Revac, ParamILS y SPO, son métodos de naturaleza estocástica, por lo que al ejecutarlos varias veces, tienden a obtener resultados similares; diferente es el caso de F-Race, el cual depende del conjunto de semillas/instancias de *training*.

Otro punto a considerar es la interacción entre parámetros, aquí F-Race, ParamIlS y SPO trabajan con la noción de configuración, más que la noción de parámetros por si solos, es por ello que colocan más atención en la interacción de parámetros.

Sobre la salida esperada en el documento se definen 3 tipos de usuarios, para los cuales la salida esperada es distinta. El primer tipo de usuario es al cual le interesa comparar nuevas meta-heurísticas respecto a otras, aquí ParamILS y SPO pueden ser buenas opciones. El segundo tipo de usuario es el que quiere explorar y analizar las capacidades de las meta-heurísticas, el conjunto final de configuraciones de F-Race es una fuente importante de información, por lo cual lo hace una buena opción. Para el tercer tipo de usuario, que busca encontrar una buena configuración de parámetros para el problema, los autores recomiendan utilizar ParamILS, el cual realiza una búsqueda intensa y fuerte en el espacio de parámetros y encuentra de manera rápida buenas configuraciones.

Sobre la velocidad de los métodos se puede observar en la Tabla 1 los valores obtenidos en el documento para cada uno de los métodos en los 3 escenarios descritos.

	F-Race	Revac	ParamILS	SPO
Promedio	127.7	56.5	60.2	68.0

Tabla 1: Comparación tiempos de sintonización (F-race, Revac, ParamILS y SPO) [min]
Fuente: [14]

Finalmente como criterio de término, el único método que es capaz de terminar por sí solo, detectando si hay una configuración restante es F-Race, pero ParamILS y SPO pueden ser detenidos en cualquier momento entregando buenas soluciones de igual manera.

2.2.2. Irace

En febrero del año 2011, los autores Lopéz et al. publicaron el artículo *The irace Package: Iterated racing for Automatic Algorithm Configuration*[12] el cual habla de una extensión del método *F-race*. Su propósito principal es el de configurar automáticamente algoritmos de optimización para encontrar el conjunto más adecuado de parámetros para problemas de optimización. *Irace* implementa una 'carrera' de forma iterativa, la cual incluye el método *I/F-Race* como un caso especial, y consiste en 3 pasos:

- Tomar una muestra con nuevas configuraciones según una distribución particular.
- Seleccionar las mejores configuraciones de las muestras anteriormente tomadas por medio de carreras (*F-race*).
- Actualizar la distribución para sesgar el muestreo hacia las mejores configuraciones.

Estos pasos se repiten hasta que se cumple un criterio de término. En *irace*, cada parámetro tiene una distribución independiente, (distribución normal para parámetros numéricos o una distribución discreta para parámetros categóricos). La actualización de las distribuciones consiste en la modificación de estas (la media y la desviación estándar en el caso de la distribución normal, o los valores de probabilidad discretos de las distribuciones discretas).

Como *entrada* el algoritmo necesita un conjunto de instancias de muestreo, un espacio de parámetros y una función de costos. Además, se necesita un número de iteraciones (carreras) que ejecutará el método. Cada carrera comienza a partir de un conjunto de configuraciones candidatas. Luego, este número disminuye con el número de iteraciones, lo que significa que se realizarán más evaluaciones por configuración en iteraciones posteriores.

En la primera iteración, las primeras configuraciones candidatas se generan al muestrear el espacio de parámetros entregados como *entrada*. Cuando se inicia una carrera, cada configuración se evalúa en la primera instancia mediante la función de costos. Luego, se realiza una prueba estadística de los resultados. Si hay suficiente evidencia estadística para identificar que algunas configuraciones candidatas funcionan peor que otra, entonces se eliminan de la carrera, mientras que las demás (candidatos supervivientes) se ejecutan en la siguiente instancia. En cada iteración, se van modificando los valores de la media y la varianza, con el objetivo de que los valores muestreados se acerquen cada vez más al valor de la configuración principal, enfocando la búsqueda alrededor de los mejores parámetros encontrados a medida que aumenta el contador de iteraciones.

Para los experimentos de *irace* se usaron *soft-restart* para así evitar estancarse en óptimos locales. En la propuesta original de *I/F-Race*, la desviación estándar (en el caso de los parámetros numéricos) o la probabilidad discreta (en el caso de los categóricos) disminuye en cada iteración. Sin embargo, si el ajuste converge a unas pocas configuraciones muy similares en pocas iteraciones, se pierde la diversidad y las configuraciones candidatas recién generadas no serán muy diferentes de las ya probadas, sin explorar nuevas alternativas.

Para esto, el mecanismo *soft-restarts* verifica el estancamiento después de generar nuevos conjuntos de configuraciones candidatas. Consideramos que existe una convergencia prematura cuando la "distancia" entre dos configuraciones candidatas es cero.

Como experimento, se buscó optimizar los parámetros que recibía el algoritmo *Simulated Annealing*, mostrando claramente mejores resultados con los parámetros optimizados que con la configuración de parámetros por defecto.

2.2.3. SMAC

En el año 2011, Hutter et. presentaron en [8] un algoritmo de sintonización de parámetros inspirado en optimización basada en modelos secuenciales (SMBO), el cual resuelve el problema de la sintonización de parámetros como si el algoritmo objetivo fuera una caja negra. SMAC mejora el enfoque propuesto por SMBO ya que elimina algunas limitantes de este último, como lo es admitir solo parámetros numéricos o el hecho de realizar la optimización del algoritmo objetivo para instancias individuales.

El problema de sintonización de parámetros resuelto por SMAC puede definirse de la siguiente manera: Dado un algoritmo parametrizado A, un espacio de configuraciones de parámetros Θ, una distribución de instancias D y una métrica de rendimiento $m(\theta, \pi)$ que captura el rendimiento del algoritmo A con una configuración de parámetros $\theta \in \Theta$ en las instancias $\pi \in D$.

Sea $f(\theta) = E_{\pi \in D}[m(\theta, \pi)]$, esta función denota el rendimiento esperado de algoritmo A bajo la configuración de parámetros $\theta \in \Theta$. El problema consiste entonces en encontrar

la configuración de parámetros θ del algoritmo A que minimiza $f(\theta)$.

El algoritmo SMBO primero recopila todos los datos iniciales y luego itera sobre los siguientes pasos:

- Basado en los datos recopilados hasta el momento, construye un modelo que predice la distribución de probabilidad para que f evalué en puntos arbitrarios $\theta \in \Theta$;
- Utiliza el modelo creado para medir la conveniencia (*desirability*) $d(\theta)$ de aprender $f(\theta)$ para cada $\theta \in \Theta$ y para seleccionar $\theta \in \text{argmax}_{\theta \in \Theta}\ d(\theta)$;
- Evaluar $f(\theta)$, resultando en un nuevo punto de datos $\langle \theta, f(\theta) \rangle$

La función de conveniencia d sirve para controlar el equilibrio de exploración/explotación entre el aprendizaje de partes nuevas y desconocidas del espacio de búsqueda y la intensificación de la búsqueda local. El trabajo realizado por los autores se extiende a mejorar SMBO mediante mecanismos que permiten el manejo de:

- Espacios de parámetros discretos y condicionales.
- Ruido sustancial no Gaussiano (debido a la variación en la distribución del tiempo de ejecución del algoritmo sobre instancias problemáticas y múltiples ejecuciones independientes en la misma instancia).
- Evaluaciones de funciones parcialmente censuradas.
- Máximo de tiempo total disponible en lugar de número de evaluaciones de funciones.
- Optimización distribuida en *clusters*.

SMAC mostró resultados favorables en los experimentos realizados en [8] al compararse con algoritmos que resuelven el problema general de la sintonización de parámetros, como lo son ParamILS y GGA [1].

2.2.4. ParamILS y Evoca

Los algoritmos ParamILS y Evoca son explicados en detalle en las secciones 2.4 y 2.3 respectivamente, debido a que son los algoritmos a utilizar en este trabajo.

2.2.5. Enfoques multi-objetivos

Frentes de rendimiento

Otro método de sintonización es el propuesto por Johann Dréo et al. [4] el año 2009, quienes dicen que la sintonización de parámetros debe considerarse como un problema multi-objetivo a la hora de sintonizar meta-heurísticas. Los autores utilizan lo que ellos llaman "Frentes de rendimiento" que consisten en una colección de configuraciones de parámetros no dominadas, respecto a la velocidad (tiempo de ejecución) y la precisión óptima. En el documento se desarrolla una idea ya conocida, la cual dice que en los problemas de meta-heurísticas estocásticas, los objetivos de velocidad y encontrar el óptimo son contradictorios, por lo tanto para una instancia del problema y de meta-heurística dada, existe una colección de conjuntos de parámetros no dominados que optimizan por un lado la velocidad, y la precisión del óptimo por otro lado.

El método multi-objetivo de ajuste de parámetros propuesto por los autores que usa NSGA-II, permite agregar los otros parámetros en uno solo, el cual es la posición en el 'frente de parámetro', desde un comportamiento de producción, el cual es más veloz y menos óptimo, a un comportamiento de concepción que al contrario del anterior es lo más óptimo y menos veloz. Para los experimentos se utilizaron 2 problemas: *Meta-heuristics, continuous problem* y *Hybrid Evolutionary Algorithm, hard combinatorial problem*. Para el primero se utilizaron los parámetros de distintas meta-heurísticas, como la temperatura de un *Simulated Anealing* (SA), el tamaño de la población de un algoritmo evolutivo para problemas continuos, entre otros. Para el segundo se utilizaron parámetros relacionados a operadores de mutación del método *Divide and Evolve* [21], usado para resolver problemas de panificación temporal.

Para el primer experimento se obtuvo la identificación de correlación entre parámetros y los objetivos, además, los autores notaron que uno puede preferir un algoritmo diferente, dependiendo del uso específico, en el caso de ellos con su experimento, SA resultaba ser la mejor opción si se quería optar por un algoritmo veloz.

Para el segundo experimento, la relación entre parámetros y objetivos es débil. Este último experimento mostró también que el tiempo necesario para un conjunto de parámetros en un método de optimización compleja es bastante alto. Tras los resultados los autores concluyeron que su método es bastante adecuado para establecer parámetros de meta-heurísticas, y elegir una técnica a utilizar dependiendo del objetivo que se tenga, pero si se trabaja con muchos parámetros el costo computacional del 'frente de rendimiento' es bastante alto.

M-FETA

En el año 2010, S.K Smith et al.[22] propusieron un algoritmo de sintonización de parámetros multi-función, llamado M-FETA, el cual está basado en un algoritmo evolutivo multi-objetivo. Este algoritmo es capaz de aproximar el 'frente de pareto' de parámetros

de forma efectiva.

El interés de los autores son las configuraciones de parámetros 'robustas' que hacen que el AE (Algoritmo Evolutivo) que los utiliza sea más 'generalista' que 'especialista'. La calidad de estos depende del rendimiento de un AE en una colección de funciones $F = \{f_1, \ldots, f_M\}$. Al ser los AE estocásticos, se puede observar ruido en el rendimiento, y para mejorar la estimación de los parámetros, debería repetirse la medición con AE, pero esto es bastante costoso, por lo que utilizan el concepto de vecindario. Para un configuración $\bar{x}$ se tiene un vecindario $N_{\bar{x}}$ con los k individuos de menor distancia euclidiana a $\bar{x}$, esto es para validar la configuración $\bar{x}$ mediante su vecindario. partir de esto, los autores definen el concepto de dominancia entre dos vectores. La idea es que si se tienen dos vectores $\bar{x}$ e $\bar{y}$, se dice que el primero domina al otro si y solo si:

- $\exists f \in F$ tal que el rendimiento del AE en f basado en la data perteneciente a $N_{\bar{x}}$ es significativamente mejor que el rendimiento del AE en la data perteneciente a $N_{\bar{y}}$.
- $\forall e \in F(g \neq f)$ el rendimiento del AE en g para vectores en $N_{\bar{y}}$ no es significativamente mejor que el rendimiento para vectores en $N_{\bar{x}}$.

Basado en la dominancia, los autores pudieron hacer un *ranking* de los vectores de parámetros y realizar comparaciones entre estos.

Se tiene que una configuración $N_{\bar{x}}$ es mejor que una configuración $N_{\bar{y}}$, cuando el *ranking* de $N_{\bar{x}}$ es menor al de $N_{\bar{y}}$, o cuando ambos tienen el mismo *ranking*, pero $N_{\bar{x}}$ está más aislado que $N_{\bar{y}}$ (donde $\bar{x}$ es más aislado que $\bar{y}$, si está más lejos de sus vecinos que $\bar{y}$).

El sistema propuesto por los autores tiene dos propiedades importantes desde la perspectiva de muestrear nuevas configuraciones para ser probados. La primera propiedad, trata de un sesgo inherente para preferir configuraciones aislados como padres en M-FETA. Este sesgo viene del hecho de que los vecinos de una cierta configuracion están bastante lejos, lo cual resulta en una desviación estándar grande y esto a su vez hace que el *ranking* disminuya, en consecuencia, la probabilidad de ser elegido para reproducción aumenta. La segunda propiedad es que la configuración hijo probablemente esté cerca del padre, por lo cual disminuye la distancia en el vecindario y en consecuencia agudiza la estimación de su utilidad.

Posteriormente todas las configuraciones de parámetros que no son dominados, conforman el conjunto de parámetros de *pareto*, es decir, no hay otro vector que obtenga un rendimiento significativamente mejor en una de las funciones de prueba. Cada confuguración en el conjunto de *pareto* puede ser visto como 'robusto', pero todos ellos representan distintas compensaciones respecto al rendimiento. La opción a elegir en esta frontera está sujeta a las preferencias de cada usuario. Ejemplos de esto es cuando se quiere que todos los problemas tengan el mismo peso y se elige la configuración que

tiene mejor rendimiento, o cuando se elige como configuración el que tiene mejor rendimiento promedio en un problema, mientras mantiene un rendimiento mínimo dado en otro problema.

La configuración experimental consiste en 3 capas de arquitectura, en la capa de aplicación se utilizaron dos funciones de prueba de 10 dimensiones, Sphere y Rastrigin. Para Rastrigin se permitieron 8000 evaluaciones y para Sphere 4000. En la capa del algoritmo se utilizó un algoritmo genético simple con *N-point Crossover*, mutación de *bitflip*, K-torneo para la selección de los padres y selección de supervivencia selectiva. En la capa del diseño, M-FETA es usado para la sintonización de parámetros.

En los resultados el acercamiento multi-objetivo permitió identificar configuraciones específicas y varios pseudo generalistas o globales, en lugar de una sola configuración. Además este enfoque da la idea de las interacciones entre los rendimientos en las funciones de ciertas pruebas, y puede dar indicaciones sobre la solidez del algoritmo sintonizador, lo cual no podría haberse obtenido utilizando el enfoque de objetivo único.

2.3. Evoca

Evoca, quien recibe su nombre por *Evolutionary Calibrator* es una herramienta de sintonización de parámetros presentada el año 2013 por Riff y Montero [20]. La idea de este método es ayudar a los diseñadores de meta-heurísticas poco experimentados a obtener una sintonización de buena calidad, de forma simple, sin tener un conocimiento muy profundo en el tema.

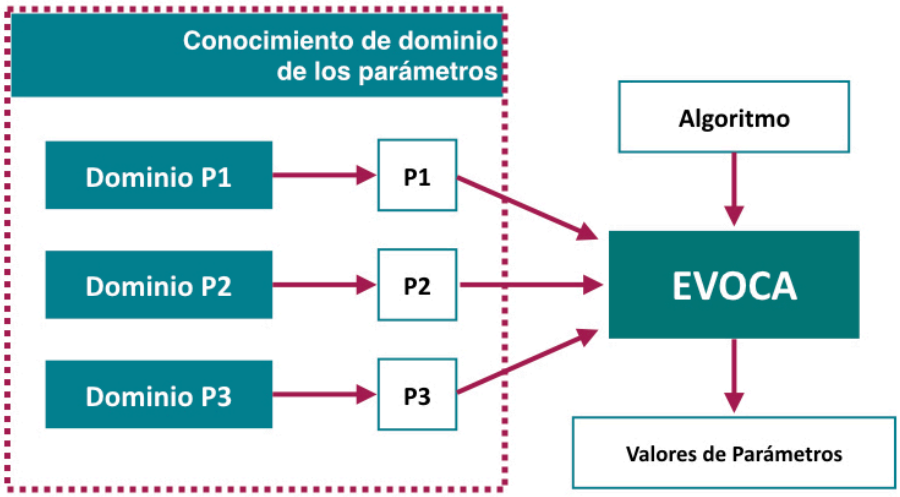


Figura 3: Diagrama de sintonización según Evoca
Fuente: [20]

Definición: Dada una meta-heurística A, sus parámetros numéricos $P_n = \{n_1, \ldots, n_m\}$

y categóricos $P_c = \{c_1, \ldots, c_k\}$ con sus dominios $D = \{d_1, \ldots, d_{m+k}\}$, el objetivo es encontrar los valores de P_n y P_c, tales que estos valores permitan al algoritmo obtener el óptimo. En la Figura 3 se puede observar como Evoca aborda la sintonización de parámetros.

Representación: La representación es un *string* en donde cada elemento corresponde a un parámetro, y los valores son tomados del dominio de parámetros. Por lo tanto el largo del *string* corresponde a la cantidad de parámetros.

Población inicial: Se obtiene considerando el número de parámetros a ser sintonizados y el tamaño de sus dominios iniciales; la idea es incluir todos los valores relevantes para cada parámetro. Para los parámetros categóricos y de dominio discretos lo anterior es posible, pero para valores numéricos se debe dividir el intervalo acorde a la precisión requerida por el diseñador.

Operadores: El algoritmo utiliza dos operadores. El primer operador es de cruzamiento para generar los hijos de toda la población, este operador es **Wheel-crossover**, el cual selecciona el gen de descendencia, para que luego el hijo generado reemplace al peor individuo de la población. El segundo operador es el de mutación, este operador consiste en un *Hill Climbing* con primera mejora, el cual toma una copia del hijo creado y trata de mejorarlo. Este hijo creado por la mutación reemplaza el segundo peor individuo de la población.

El Algoritmo 1, muestra el procedimiento de Evoca, donde R es la cantidad de semillas usadas para evaluar la configuración de parámetros.

Algoritmo 1: Evoca tuning method

Input : Target-algorithm, P_c, $precision_c$, P_n, D
Output : Parameter Values

```
GenerateInitialPopulation();
while Termination_Criterion is false do
    New_Population ← Current_population;
    Child ← Wheel-Crossover(New_Population);
    Evaluate Child using R random seeds;
    Replace the worst chromosome by Child in New_Population;
    Child_mutated ← Mutation(Child);
    Evaluate Child using R random seeds;
    if Child_mutated is better than Child then
        Replace the second worst chromosome by Child_mutated in New_Population
    end
end
```

Los autores, con Evoca, propusieron un algoritmo que es fácil de configurar para definir los datos de entrada. El algoritmo fue capaz de obtener valores de buena calidad y entregó información muy útil en el proceso de diseño. Evoca además es capaz de sin-

tonizar parámetros numéricos y categóricos. En el documento se muestra como Evoca obtiene un rendimiento muy similar a ParamILS para algunas instancias en las pruebas del algoritmo genético que resuelve el problema de los *NK landscape*, además de supera a Revac [20].

2.4. ParamILS Method

Parameter iterated local search method es un algoritmo para el seteo de parámetros presentado por primera vez por Hutter et Al. en [10] el año 2007. Este algoritmo basa su funcionamiento en una búsqueda local iterativa como su nombre lo indica y es capaz de funcionar con algoritmos deterministas o aleatorios.

2.4.1. Problema de configuración de parámetros

Antes de lograr entender cómo funciona ParamILS, es necesario definicir formalmente lo que los autores consideran en [9] y [10] como problema de seteo de parámetros e introducir la notación necesaria. Sea $\mathcal{A}$ un algoritmo objetivo, y $p_1, \ldots, p_k$, los parámetros de $\mathcal{A}$. El dominio de los posibles valores para cada parámetro p_i se denota como Θ_i. Además, se asume que los dominios de los parámetros son finitos. Los autores utilizaron $\Theta \subseteq \Theta_1 \times \ldots \times \Theta_k$ para denotar el espacio de todas las configuraciones de parámetros factibles, y $\mathcal{A}(\theta)$ denota una instanciación del algoritmo $\mathcal{A}$ con la configuración de parámetros $\theta \in \Theta$. Por otro lado $\mathcal{D}$ denota la probabilidad de distribución sobre el espacio Π de instancias del problema, y un elemento del espacio Π es π. Luego la definición formal del problema de seteo de parámetros se enuncia de la siguiente manera:

- $\mathcal{A}$ es un algoritmo parametrizado;
- Θ es el espacio de configuración del algoritmo $\mathcal{A}$;
- $\mathcal{D}$ es una distribución sobre las instancias del problema con dominio Π;
- k_{max} es el tiempo límite, después del cual cada ejecución de $\mathcal{A}$ es terminada si ésta continua ejecutándose.
- o es una función que mide el costo observado de ejecutar $\mathcal{A}(\theta)$ en una instancia $\pi \in \Pi$ con tiempo límite $k \in \mathbb{R}$;
- m es un parámetro estadístico de población (varianza, mediana, etc).

Además, los autores de los artículos denotaron la secuencia de ejecuciones por el sintonizador como $R = ((\theta_1, \pi_1, s_1, o_1), \ldots, (\theta_n, \pi_n, s_n, o_n))$, donde la i-ésima ejecución $R[i]$ esta dada por cinco valores:

- $\theta_i \in \Theta$ corresponde a la configuración de parámetros que se está evaluando;
- $\pi \in \Pi$ corresponde a la instancia con la que se está ejecutando el algoritmo;
- s_i denota el número de la semilla aleatoria utilizada en la ejecución;
- k_i es el tiempo límite de ejecución;
- o_i corresponde al costo observado de ejecución.

Para cada configuraciónθ, O_θ denota una distribución de costo inducida por una función o, aplicada a instancias π en D y a múltiples ejecuciones idependientes.Luego, el costo de una configuración θ candidata se define cómo:

$$c(\theta) := m(O_\theta)$$

Luego se tiene que el costo estimado de un costo $c(\theta)$ basado en una secuencia de ejecuciones R, como $\hat{c}(\theta, R) := \hat{m}(\{o_i|\theta_i = \theta\})$. Dónde $\hat{m}$ corresponde a la muestra estadística análoga al parámetro estadístico de población m.

En [9] también se realizan definiciones de los rendimientos de entrenamiento y prueba. Se tienen dos conjuntos de instancias. Primero, el de entrenamiento, mediante el cual se obtiene una configuración al haber realizado una secuencia de ejecuciones en un tiempo t y resolver el problema de seteo de parámetros. Por otro lado se deja un conjunto apartado de instancias y semillas aleatorias, ambos desconocidos para el sintonizador y se utilizan para la evaluación con la configuración titular encontrada.

2.4.2. Funcionamiento de ParamILS

Se observa que ParamILS, corresponde a un *iterated local search* que utiliza una configuración inicial aleatoria, un procedimiento (*better*) como criterio de aceptación (el cual posee dos variantes), su técnica de búsqueda local complementaria es *first improvement*, cuenta con una cantidad s de perturbaciones para escapar de los óptimos locales y mediante una probabilidad dada por el parámetro $p_{restart}$ se puede reiniciar el proceso de búsqueda a una configuración aleatoria. El Algoritmo 2 muestra la estructura principal de ParamILS.

2.4.3. El algoritmo BasicILS

Como ya se mencionó, ParamILS cuenta con un criterio de nuevas soluciones. Este criterio está dado por un procedimiento llamado *better*. El enfoque más simple del procedimiento *better* se conoce como *BasicILS*, específicamente, cuando en los artículos se

Algoritmo 2: ParamILS tuning method

Input : Initial configuration $\theta_0 \in \Theta$, algorithm paramters r, $p_{restart}$ and s
Output : Best parameter configuration θ found

$c_0 \leftarrow$ default paramter configuration $\theta \in \Theta$;
for $i \leftarrow 1$ **to** R **do**
  $\theta \leftarrow$ random $\theta \in \Theta$;
  if *Better*(θ, θ_0) **then**
    $\theta_0 \leftarrow \theta$
  end
end
$\theta_{ils} \leftarrow$ `IterativeFirstImprovement`(θ_0);
while *not* $Termination\_criterion$ **do**
  $\theta \leftarrow \theta_{ils}$;
  `/*-----------Perturbation-----------*/`
  for $i \leftarrow 1$ **to** s **do**
    $\theta \leftarrow$ random $\theta' \in Nbh(\theta)$;
  end
  `/*---------Basic local search---------*/`
  $\theta \leftarrow$ `IterativeFirstImprovement`(θ);
  `/*---------Aceptance Criterion---------*/`
  if *Better*(θ, θ_{ils}) **then**
    $\theta_{ils} \leftarrow \theta$;
  end
  if $p_{restart}$ **then**
    $\theta_{ils} \leftarrow$ random $\theta \in \Theta$;
  end
end
return Overall best θ found;

utiliza el término BasicILS(N), se hace referencia a la variación de ParamILS donde el procedimiento $better(\theta_1, \theta_2)$ retorna la configuración de parámetros con menor costo estimado $\hat{c}_N$ entre costos estadísticos $c(\theta_1)$ y $c(\theta_2)$. BasicILS(N) evalúa cada configuración de parámetros al ejecutar las mismas N instancias/semillas de entrenamiento.

2.4.4. El algoritmo FocusedILS

FocusedISL es una variante de ParamILS que selecciona el número de instancias de entrenamiento de manera adaptativa. FocusedILS maneja el problema de cómo elegir la cantidad de instancias de entrenamiento al variar el número de muestras de entrenamiento consideradas de una configuración de parámetros a otra. Se cuenta con número de ejecuciones disponibles para estimar el costo estadístico $c(\theta)$ para una configuración de parámetros θ, el cual se denomina $N(\theta)$. La idea es partir con pocas instancias/semillas y luego añadir más a las configuraciones de parámetros de mejor calidad.

FocusedISL compara θ y θ' en base a $N(\theta)$ ejecuciones con las mismas instancias y semillas.

Luego se hace uso de lo que los autores de [9] y [10] denominan como dominancia. Una configuración de parámetros θ domina a θ' cuando se han realizado al menos tantas ejecuciones en θ como en θ', y el rendimiento de $\mathcal{A}(\theta)$ en las $N(\theta')$ ejecuciones sea al menos tan bueno como $\mathcal{A}(\theta)$ en dichas ejecuciones.

Domination: θ_1 domina a θ_2 si y solo si $N(\theta_1) \geq N(\theta_2)$ y además $\hat{c}_{N(\theta_2)}(\theta_1) \leq \hat{c}_{N(\theta_2)}(\theta_2)$

2.4.5. Limitación adaptativa del tiempo de ejecución del algoritmo

ParamILS también cuenta con una técnica que determina de forma adaptativa el tiempo límite para cada ejecución del algoritmo, la idea principal tras esto es que algunas configuraciones de parámetros se ejecutan mucho más rápido que otras y por ende se necesita variar el tiempo limite dependiendo de la configuración y rendimiento. La técnica para determinar el tiempo de ejecución cuenta con una variante en caso que se quiera una actitud más agresiva con las configuraciones de peor rendimiento.

La técnica de limitación adaptativa de tiempo de ejecución realiza una implementación de un límite de evaluaciones de una configuración de parámetros basado en N ejecuciones, es decir, un límite para el tiempo de ejecución. Este procedimiento secuencialmente calcula un límite inferior cada vez que se ejecuta para una configuración de parámetros, entonces cada vez hay una ejecución se suma el tiempo de ejecución y se divide todo por N. Una vez que el límite inferior supera el límite dado por parámetro entonces se descartan las siguientes ejecuciones restantes para la configuración de parámetros dada.

A continuación se describirán las redes de óptimos locales para el estudio de *fitness landscapes*, sus características y algunas de sus implementaciones en la literatura.

2.5. Local Optima Network

Un *fitness landscape* de un problema combinatorial puede ser visto como un grafo en donde los vértices corresponden a una configuración. Si dos de estas configuraciones pueden ser conectadas mediante un operador de movimiento, se puede trazar un arco entre ellas. Esta lógica es la que siguió la Dra. Ochoa y compañía en [17] para realizar un estudio de *NK landscapes* mediante redes de óptimos locales.

Una red de óptimos locales o LON (*Local Optima Network*) consiste en una representación comprimida de *fitness landscapes*. En particular una LON consiste en un grafo dirigido $G = (V, A)$ en donde los nodos V contienen todos los óptimos locales del *fitness landscape*, mientras que los arcos A modelan la transición entre los óptimos locales. En

la figura 2.5 se puede apreciar un ejemplo simple de *fitness landscape* llevado a una LON.

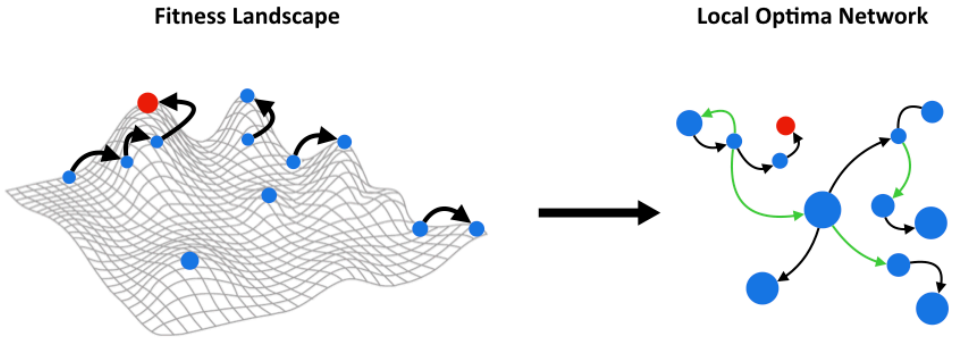


Figura 4: Diagrama de *fitness landscape* llevado a LON. Los nodos color azul corresponden a óptimos locales según el algoritmo de sintonización, mientras que el nodo rojo corresponde al óptimo global. En la red el tamaño de los nodos se asocia a cuantas veces se paso por ese nodo, los arcos color negros son arcos de mejora, mientras que los verdes corresponden a arcos de empeoramiento. Fuente: Elaboración propia.

Antes de continuar el estudio de LONs, es necesario entender el concepto de *basin* de atracción para un *fitness landscape*. Este término se refiere a áreas del espacio en las que se encuentran agrupados los óptimos locales (Ver figura 5). Esta idea proviene de lo que se conoce como *big-valley structure* en optimización combinatoria [25], es decir, que la distribución de los óptimos locales no es uniforme, si no que se encuentra agrupada en *basins* de atracción. En la literatura estos *basins* son conocidos también como *funnels*.

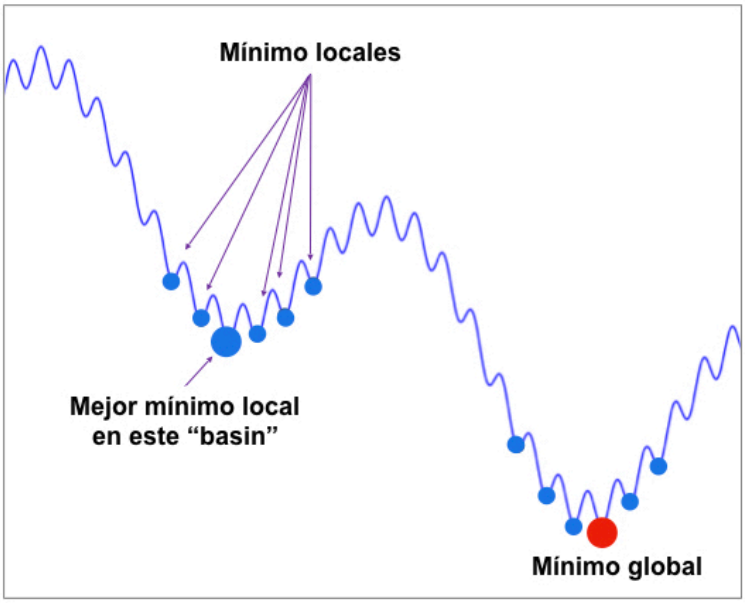


Figura 5: Ejemplo de *basins* de atracción o *funnel*
Fuente: G. Ochoa and N. Veerapen. SearchMaps: Visualising and Exploiting the Global Structure of Computational Search Spaces. GECCO 2018.

2.5.1. Características de las Redes

Dentro de la motivación para utilizar LONs en el estudio de *fitness landscapes* es que entre las características que se pueden encontrar en ellas son:

- **Distancia:** Mediante los enlaces que se crean entre los nodos se puede establecer una distancia entre ellos.
- **Topología (distribución de grado):** Información de qué tan conectados están los nodos. $p(k)$ es la probabilidad de que un nodo elegido aleatoriamente tenga k conexiones.
- **Cohesión:** Mediante estadísticas simples, como la densidad de los nodos y la distancia media entre el camino más corto, se puede establecer el coeficiente de agrupamiento (*clustering*).

2.5.2. Arcos con probabilidad de transición entre *basins* (A_{btp})

El primer enfoque para obtener el conjunto de arcos A en las LON es el basado en la probabilidad de transición entre *basins* [17]. Lo anterior es, dado un conjunto de arcos A_{btp}, existe un arco con un peso asociado entre un óptimo local lo_x y un óptimo local lo_y, si la búsqueda local se puede mover desde el *basin* de atracción $B(lo_x)$ hasta el *basin* $B(lo_y)$. Mientras mayor número de conexiones se encuentren entre dos *basins*, mayor es la probabilidad de transición entre dos óptimos locales, y por ende, el peso de los arcos es mayor. La probabilidad de realizar un movimiento entre dos soluciones s_x y s_y depende del número de vecinos "superiores" de s_x, y de cuántos de estos vecinos pertenezcan a $B(lo_y)$.

2.5.3. Arcos de Escape (A_{ee})

Un modelo alternativo para analizar *fitness landscapes* mediante LONs es el introducido por Vérel et al en [24]: *LONs with scapes edges*. El enfoque propuesto nuevamente utiliza un conjunto de arcos, llamado ahora A_{ee}. La diferencia está en que en este caso la idea es asignar el peso de los arcos en base a la probabilidad que tiene el algoritmo para escapar de los óptimos locales mediante una perturbación. Los arcos de escape están definidos mediante una función de distancia d en el *fitness landscape* (mínimo número de movimientos entre dos soluciones). Además se necesita definir un valor entero $D \geq 0$ el cual corresponde a la distancia máxima que puede aplicar la perturbación para llegar a la solución s, es decir:

$$d(lo_x, s) \leq D \wedge s \in B(lo_y)$$

2.5.4. LONs con *Partition Crossover* (PX)

En el año 2015, Ochoa et Al. [16] presentaron un análisis de redes de óptimos locales basados en *partition crossover for k-Bounded pseudo-Boolean functions* [23] utilizando NK_q *landscapes* como caso de estudio. El objetivo de este enfoque es utilizar *partition crossover* para definir la transición entre nodos de óptimos locales y así obtener una variante a las redes de óptimos locales, conocida como XLON (X en referencia al cruzamiento).

Es interesante estudiar esta variante de las redes de óptimos locales, ya que al momento de llevar a cabo la integración de Evoca con LONs, este enfoque puede ser el camino a seguir, puesto que Evoca es un algoritmo poblacional, el cual va realizando cruzamientos entre los individuos de la población.

El procedimiento a realizar para construir la red XLON, consiste en obtener dos padres x, y de la población, y realizar *partition crossover* entre ellos, obteniendo así un hijo w, al cual se le aplica un algoritmo de búsqueda local. En el caso de los autores, ellos aplicaron *Hill Climbing*, obteniendo así, del hijo w, un nuevo individuo llamado z y es este quien es agregado a la población, de forma tal que se crean los arcos entre los padres x, y a z. Algoritmo 3 describe el proceso recién descrito.

Algoritmo 3: XLON Construction

Input : The set of nodes V
Output : $XLON = G(V, A_{PX})$

```
for {x, y} ⊆ V do
    w ← PartitionCrossover(x, y);
    z ← HillClimbing(w);
    if x ≠ z y z ≠ y then
        A_PX ← A_PX ∪ {(x, z), (y, z)}
    end
end
```

2.5.5. LONs con *PageRank centrality*

En el estudio y análisis de las redes, el concepto de centralidad describe qué tan importante e influyente es un nodo. Una forma de medir esto, es mediante la centralidad de los vectores propios, donde *PageRank centrality* es una variante a esta última. Los acercamientos más recientes sobre redes de óptimos locales consisten en el uso de *PageRank centrality* para la predicción del rendimiento esperado de meta-heurísticas basadas en búsqueda local. Un ejemplo de uso de *PageRank centrality* es el usado por Google para asignar relevancia entre sitios web, el cual está basado en la idea de un usuario que navega la web dando clicks aleatoriamente.

Lo que los investigadores Herrmman y Rothlauf buscaron con sus enfoques, es que las redes de óptimos locales que utilizan *PageRank centrality* con arcos basados en probabilidad de transición entre *basins* [7] (Herrmann and Rothlauf, 2015) o arcos de escape [5] (Herrmann, 2016) logren predecir de mejor manera el rendimiento de la búsqueda local que otros criterios de predicción como *ruggedness* y *deceptiveness* para *NK landscapes*, con búsqueda local iterativa (ILS) y *Simulated Annealing*.

Para realizar el cálculo de *PageRank* de los nodos de una LON, lo primero que se necesita es obtener una matriz transición Π_{btp} o Π_{ee} de tamaño $|LO| \times |LO|$ (donde LO corresponde al conjunto de óptimos locales). Los elementos $\pi_{x,y}$ $(x, y \in LO)$ de la matriz Π son los arcos con peso $|a_{x,y}|$ de los arcos $a_{x,y} \in A_{btp}$ o $a_{x,y} \in A_{ee}$ respectivamente. Como la matriz π es estocástica, todas las filas y columnas deben ser normalizadas para que sumen 1. Luego, el vector de *PageRank* que contiene la centralidad de todos los nodos de la red de óptimos locales, es el vector P, el cual corresponde el vector propio de Π.

En sus artículos los autores confirman que lograron demostrar que *PageRank centrality* es un buen criterio de predicción de la dificultad de la búsqueda. Se obtuvo que *PageRank centrality* del óptimo global encontrado en una LON predice de forma precisa la tasa de éxito de encontrar el óptimo global. Por otro lado los promedios de los valores de aptitud de los nodos de óptimos locales con un peso asociado a su correspondiente *PageRank* demostraron realizar una buena predicción de la calidad de la solución. La investigación más reciente de *PageRank centrality* y LONs es el propuesto por Herrmann et Al. el año 2017 [6] en el cual se busca realizar una comparativa del trabajo previo: LONs con A_{btp} y LONs con A_{ee}. El resultado de la investigación arrojó que para ILS y *Simulated Annealing* la definición de los arcos de una LON debe ajustarse al mecanismo de diversificación de la meta-heurística. En particular, se encontró que LONs con A_{btp} capturan bien el mecanismo de diversificación de SA, mientras que por otro lado los LONs con A_{ee} capturan bien la dinámica de búsqueda de ILS.

2.6. Trabajo relacionado

Durante septiembre de año 2018 Pushak y Hoos presentaron en [19] un estudio sobre los *Landscapes* de los algoritmos de configuración de parámetros. En el documento los autores abarcan el mismo problema sobre el comportamiento de los algoritmos de sintonización sobre el espacio de búsqueda para encontrar respuestas a la dependencia del rendimiento del algoritmo en base a los valores de los parámetros.

La investigación llevada a cabo se basa en parámetros de tipo numérico que es uno de los casos más comunes en el proceso de sintonización de parámetros. La idea se basa en la hipótesis de que la mayoría de los parámetros tienen un efecto unimodal respecto al rendimiento del algoritmo objetivo incluso un efecto convexo. El objetivo de realizar el estudio de *Landscapes* en los problemas de configuraciones planteado por los autores

consiste en encontrar respuesta a las siguientes dos preguntas:

- ¿Cuando todos los demás parámetros son fijos, ¿son las respuestas de los parámetros numéricos individuales unimodales o convexos? si es así, ¿Con qué frecuencia?
- ¿Cuando todos los demás parámetros son fijos, ¿son las respuestas de los parámetros numéricos individuales unimodales o convexos **en instancias individuales**? si es así, ¿Con qué frecuencia?

Para encontrar respuesta a la interrogantes planteadas se propone estudiar el comportamiento de cada parámetro de forma independiente, fijando los demás, el rendimiento obtenido por el algoritmo objetivo al evaluar una configuración de parámetros donde solo se va variando un solo parámetro se conoce como: *response slice*. Para explicar la varianza entre las ejecuciones independientes del algoritmo objetivo y las instancias del problema, los autores utilizaron un procedimiento de *bootstrap* anidado, con este procedimiento se generan intervalos de confianza para el rendimiento observado del algoritmo objetivo. Mediante los intervalos de confianza se realizan procedimientos de prueba para determinar si existe evidencia suficiente para rechazar la hipótesis de conexidad y/o unimodalidad de un parámetro dado. Además los autores realizan una identificación de los parámetros más interesantes, en esta identificación se busca saber si el *response slice* es significativo o no, es decir, que el parámetro realmente produce un cambio en el rendimiento del algoritmo, la identificación se hace mediante los intervalos de confianza. Además se considera también la cantidad de óptimos locales y un análisis de la distancia entre distintas calidades. Los experimentos fueron realizados con 10 escenarios de configuración distintos, utilizando tres problemas de optimización bastante conocidos: SAT, MIP y TSP.

En sus resultados los autores encontraron un fuerte apoyo a su hipótesis de que las respuestas de los parámetros en conjuntos de instancias tienden a ser unimodales y convexas. También se encontró evidencia de que muchos parámetros tienen respuestas convexas y unimodales en casos problemáticos individuales. Además también lograron concluir que una pequeña porción de los parámetros tienen un comportamiento robusto para todos todas las instancias, pero al analizar por conjunto de instancias sus respuestas no son completamente robustas. Esto último resulta un caso muy interesante a continuar investigando.

2.7. Resumen del capítulo

En este capítulo se presentó una revisión de la literatura para los tópicos asociados al trabajo presentado en esta memoria. Primero que todo, se realizó un contextualización de la sintonización de parámetros para luego describir algunos de los principales métodos existentes. A grandes rasgos se puede observar que todos los algoritmos de sintonización

funcionan de forma similar tratando de mejorar en cada iteración, y las diferencias vienen dadas a la hora de elegir cómo se produce esa mejora. No se puede decir que un algoritmo es mejor que otro, ya que todos funcionan bien en distintos escenarios, dependiendo de instancias, problemas y semillas.

Respecto a los algoritmos de sintonización a utilizar en este trabajo, en la revisión de la literatura, ParamILS, el cual está basado en búsqueda local iterativa, mostró ser un algoritmo bastante robusto con distintas variantes, las cuales le permiten realizar una buena selección de instancias de entrenamiento y determinar de forma adaptativa el tiempo límite para cada ejecución del algoritmo objetivo. Es importante mencionar que ParamILS necesita una discretización de los parámetros a utilizar. Evoca por otro lado, es un algoritmo evolutivo que utiliza un operador de cruzamiento y un operador de mutación. A diferencia de ParamILS, Evoca no necesita una discretización del dominio de los parámetros y está pensado en diseñadores de meta-heurísticas poco experimentados en sintonización de parámetros.

Se realizó una definición de las redes de óptimos locales (LONs), las cuales permiten realizar una representación comprimida de un *fitness landscape* mediante un grafo $D = (A, V)$, donde V es el conjunto de nodos que contiene los óptimos locales del *fitness landscape* y A es el conjunto de arcos que modelan la transición entre los óptimos locales. Las redes de óptimos locales pertenecen al grupo de las redes complejas, lo que las hace una gran herramienta para obtener métricas y poder visualizar los *fitness landscape*. Además, se definieron distintas implementaciones de LONs encontradas en la literatura, como LONs que definen sus arcos entre óptimos locales mediante una probabilidad de transición entre *basins* de atracción o arcos de escape (perturbaciones).

Finalmente se realizó una revisión a la única investigación relacionada al presente trabajo. Los investigadores Pushak y Hoos en [19], presentaron un estudio donde se intenta abordar el problema sobre cómo se comportan los algoritmos de sintonización de parámetros, sobre el espacio de búsqueda. Si bien el problema a abordar es similar, el enfoque propuesto se aleja bastante de lo hecho en esta memoria, ya que los autores investigan el comportamiento de un parámetro al dejar los demás fijos en distintos escenarios de configuración.

En el siguiente capítulo, se realiza una definición de la propuesta de solución, sus fundamentos y la metodología aplicada para implementar la solución.

CAPÍTULO 3

PROPUESTA DE SOLUCIÓN

Como se mencionó en la sección 1, realizar mejoras a la sintonización de parámetros es una tarea relevante en el campo de la optimización combinatoria. Lo anterior, dado los altos costos asociados al rendimiento por la inmensa cantidad de ejecuciones que son necesarias para sintonizar, y la importancia de obtener el mejor resultado posible. **La propuesta de solución de esta memoria, es realizar aportes a la investigación mediante el estudio de los *fitness landscapes* sobre los cuales trabajan los algoritmos de sintonización**. Para poder estudiar este campo **se propone hacer uso de las redes de óptimos locales (LON)** definidas en las sección 2.5. **El desafío de esta memoria yace en realizar la integración entre algoritmos de sintonización de parámetros y LONs**. La figura 6 muestra el proceso de la propuesta recién descrita. En el presente capítulo se presentan los fundamentos de la propuesta de solución y sus componentes, para luego explicar la metodología aplicada a esta propuesta.

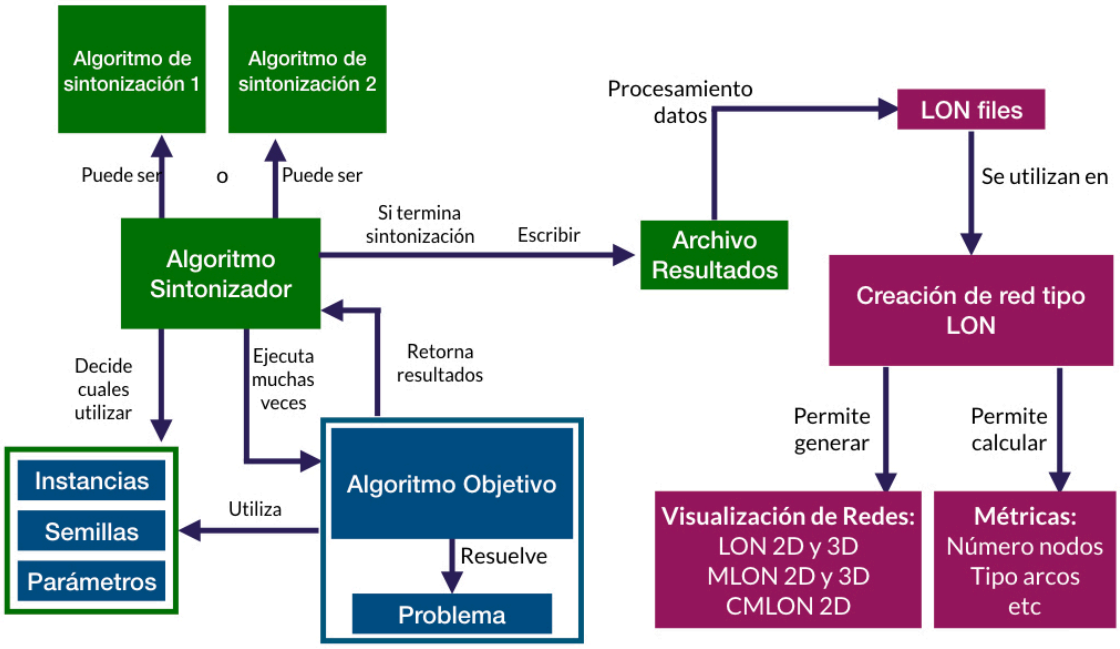


Figura 6: Diagrama de propuesta de solución
Fuente: Elaboración propia.

3.1. Fundamentos de la propuesta de solución

3.1.1. Estudio de *fitness landscapes* en algoritmos de sintonización

El estudio de *fitness ladscapes* en el campo de la optimización combinatoria no es algo nuevo. Estos han logrado ser una herramienta de gran ayuda para poder entender mejor cómo las técnicas de búsqueda local trabajan sobre el espacio de búsqueda, en consecuencia con los resultados obtenidos se pueden mejorar las implementaciones existentes o plantear nuevos enfoques adecuados al problema en estudio y su *fitness ladscapes*. Distinto es el caso de la sintonización de parámetros, en donde poca investigación existe a la fecha. El único enfoque presentado a la fecha por Pushak y Hoos [19] se basa en estudiar el espacio de búsqueda en el cual operan los sintonizadores de parámetros mediante la variación de un parámetro y dejando fijo los demás. Se debe notar que si bien se realiza un estudio del espacio de búsqueda en [19] dista bastante de la propuesta de esta memoria, ya que son enfoques muy distintos pero con un objetivo en común que es encontrar información respecto a los espacios de búsqueda de algoritmos de sintonización.

Por los motivos anteriormente mencionados, realizar estudio de *fitness landscapes* en sintonización de parámetros, puede ser una gran herramienta para lograr comprender cómo los algoritmos elegidos para esta memoria trabajan sobre el espacio de búsqueda, observar diferencias entre los algoritmos sintonizadores y sus similitudes. Toda la información que pueda ser obtenida del proceso de sintonización es útil ya que permite entender este mismo para ser mejorado en nuevas iteraciones.

Definición: un *fitness landscapes* corresponde a un trío (S, V, f), donde S es un conjunto de posibles soluciones, mejor conocido como el espacio de búsqueda. $V(s)$ contiene a los vecinos de una solución $s \in S$, y $f : S \to z$ es una función que puede ser interpretada como la encargada de dar una "altura" z a las posibles soluciones del espacio de búsqueda.

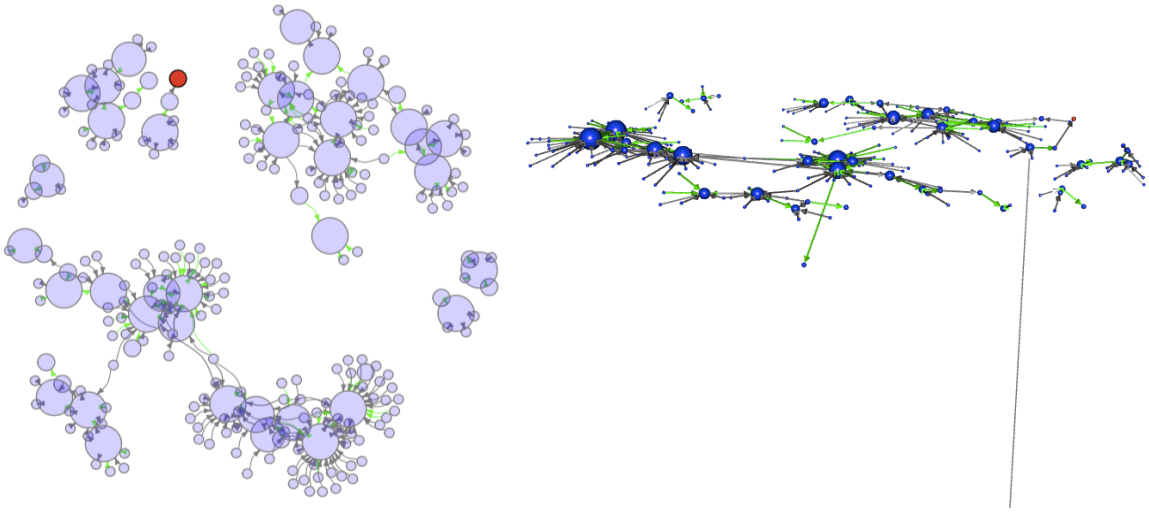
Para la sintonización de parámetros, el espacio de búsqueda corresponde a todas las combinaciones posibles de los parámetros, y el valor z depende del algoritmo sintonizador, ya que varios sintonizadores utilizan semillas e instancias distintas en cada ejecución, es decir, al ser la sintonización de carácter estocástico, se tienen muchos valores posibles de z para una misma configuración de parámetros,

por ende en distintas ejecuciones del mismo se obtienen distintos valores de z,.

3.1.2. Redes de óptimos locales (LONs) para el estudio de fitness landscapes

Cómo se estudio en la literatura, los *fitness landscapes* pueden ser representados mediante grafos (sección 2.5), tomando en cuenta solo los óptimos locales que encuentra un algoritmo o técnica de búsqueda local. Las redes de óptimos locales demostraron ser una buena herramienta para predecir el comportamiento de un algoritmo de búsqueda local para el problema de los *Nk landscapes*. Además de lo mencionado, las LONs cuentan con la capacidad de adaptarse a distintas relaciones entre los nodos de óptimos locales, en la literatura estudiada se presentaron asociaciones entre nodos mediante *basins* de atracción, operadores de cruzamiento o perturbaciones.

Una de las grandes innovaciones de las redes de óptimos locales para el estudio de *fitness landscapes* es utilizar herramientas de visualización, de esta manera mediante librerías gráficas como "igraph" para el lenguaje R podemos obtener ilustraciones de los óptimos locales encontrados por el algoritmo de búsqueda local, en el caso de este trabajo, corresponde a los óptimos locales del proceso de sintonización. En la figura 7 se aprecia un ejemplo de red de óptimos locales.



(a) LON 2D (b) LON 3D

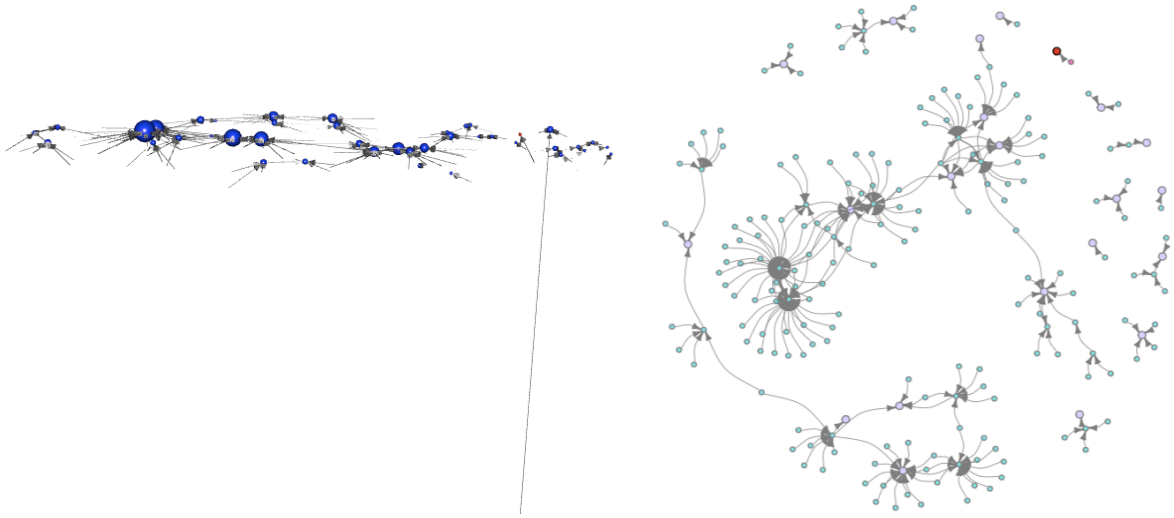
Figura 7: Ejemplo de visualización 2D y 3D de una LON

Las herramientas de visualización presentes en LONs ayudan a observar:

- Posición de los nodos en 2D y 3D (distancia)
- Grosor de los Arcos
- Tipo de arco (color, dirección)

- Tamaño de los nodos
- Tipo de nodo (color)
- Simetrías en la red
- Estructura de comunidades (Sub-grafos)

Además de las herramientas de visualización recién mencionadas, las redes de óptimos locales cuentan con 2 variantes, MLON (por *monotonic* LON) que consiste en un LON que solo tiene los arcos de mejora, y CMLON (por *compressed monotonic* LON) en donde los nodos cambian de color para identificar *basins* de atracción.



(a) MLON 3D (b) CMLON 2D

Figura 8: Ejemplo de visualización MLON y CMLON

Por otro lado, las redes de óptimos locales pertenecen a las denominadas *complex networks*[2]. En estas redes se puede encontrar una gran variedad de métricas para el análisis de lo que se desea estudiar, en el caso de LONs se pueden encontrar métricas como:

- Número de nodos
- Número de arcos (densidad)
- Número de óptimos globales
- Aptitud o calidad promedio de los óptimos locales
- Cantidad de componentes conexas

[2]En teoría de redes: red que posee cierta propiedades estadísticas y topológicas no triviales

- Distancia promedio de nodos al óptimo
- Centralidad (*PageRank*) del óptimo global
- Coeficiente de agrupamiento (*clustering*)

3.1.3. Algoritmos de sintonización escogidos

Los algoritmos de sintonización escogidos corresponden a ParamILS y Evoca. La selección de estos algoritmos se debe al tipo de búsqueda que aplican para encontrar el óptimo y las ventajas que ambos poseen sobre otros sintonizadores. Como se observó en la literatura, el algoritmo ParamILS consiste en una búsqueda local iterativa (ILS), al igual que la primera técnica de búsqueda local presentada para las redes de óptimos locales en [17], el cual puede ser aplicado tanto con arcos basados en probabilidad de transición entre *basins* de atracción o arcos de escape. Por todo lo anterior, ParamILS parece ser un buen candidato para realizar la primera integración entre LONs y sintonizadores. Otro enfoque es el que esta planteado por Evoca, el cual está basado en un algoritmo evolutivo, utilizando cruzamiento y mutación para obtener los nuevos individuos de la población. Esto último es similar al enfoque planteado en [16] donde se presenta un enfoque de LONs con *partition crossover*, en donde los arcos de la red se modelan mediante el cruzamiento entre los padres (conocida en la literatura como XLON).

ParamILS, como se analizó en la sección 2.4, resulta ser un algoritmo bastante robusto, con componentes refinados para realizar una buena sintonización de parámetros. Este sintonizador en comparación con otros algoritmos obtiene algunos de los mejores resultados considerando tiempo y características. Evoca, por otro lado es un algoritmo relativamente nuevo y bastante simple, pero su fuerte está en los buenos resultados que se pueden obtener sin tener un profundo conocimiento en sintonización de parámetro. Tanto Evoca como ParamILS son algoritmos estocásticos los cuales obtienen resultados similares (no iguales) al ejecutarse varias veces, puesto que se ejecutan con distintas semillas aleatorias. Esta característica permite que puedan realizarse comparaciones justas entre ambos respecto a los datos de entrada.

3.2. Metodología de la implementación de la solución

El primer paso para la implementación de la solución, posterior a estudiar los métodos de sintonización a utilizar y la teoría de las redes de óptimos locales, fue entender las implementaciones ya existentes con las que se debía trabajar. Esto involucra los códigos de ParamILS, Evoca y los scripts para obtener las LONs.

3.2.1. ParamILS y LONs

La implementación con ParamILS es la que resulta más simple, puesto el primer enfoque de LONs es bastante similar, el criterio utilizado para identificar los óptimos locales es cuando el algoritmo de ParamILS realiza la fase de *basic local search*, que consiste en *iterative first improvement*, por lo tanto, el arco de transición se define entre las configuraciones de parámetros anterior y posterior a la fase *basic local search*. Algoritmo 4 muestra de forma simple y resumida la modificación a Algoritmo 2 presentado en la sección 2.4 mediante la función `writeLONfile`.

Algoritmo 4: ParamILS tuning method for LON

```
   /*---Start the search with the default parameter configuration        */
1  θ_ils ←IterativeFirstImprovement(θ_0);
2  while not Termination_criterion do
3  |   θ ← θ_ils;
   |   /*-----------Perturbation-----------                              */
   |   /*---------Basic local search---------                           */
4  |   θ ← IterativeFirstImprovement(θ);
5  |   writeLONfile(θ_ils, θ) ;
   |   /*---------Aceptance Criterion---------                          */
6  end
7  return Overall best θ found;
```

3.2.2. Evoca y LONs

Por otro lado, la implementación de LONs con Evoca resulta más complicada. Cabe mencionar que en esta implementación, cada individuo de la población corresponde a una configuración de parámetros con su respectiva aptitud. Primero que todo, es necesario modificar las clases relacionadas a las configuraciones de parámetros para posteriormente identificar un óptimo local ya conocido. El siguiente paso fue definir un criterio para reconocer un óptimo local, el cual añadir a la LON. Un óptimo local en Evoca se define como el nuevo individuo de la población que posea mejor aptitud, es decir, al actualizar la población de Evoca, que como se describió en la sección 2.3, primero genera un individuo nuevo en base a *Wheel-Crossover*. Posteriormente, al realizar la mutación del nuevo individuo, este no puede empeorar, ya que la mutación corresponde a un *Hill Climbing* con *first improvement*. Es por ello que si la solución mejora su aptitud, el nuevo óptimo local será el individuo mutado. Inversamente, si la solución se mantuvo igual, el nuevo óptimo local será el individuo generado a partir del cruzamiento. Una vez identificado el óptimo local se debe agregar al conjunto de óptimos locales conocidos y se deben crear las relaciones (arcos) entre los padres que también son óptimos locales. Se realiza una verificación para cada padre, y si el padre

corresponde a un óptimo local se llama a la función `writeLONfile`. Algoritmo 4 ilustra la modificación a Evoca presentada por algoritmo 1 en la sección 2.3.

Algoritmo 5: Evoca tuning method for LON

```
1  GenerateInitialPopulation();
2  while Termination_Criterion is false do
       /*-----------Create new population-----------                    */
       /*----------- Create Child by wheel-crossover-----------         */
       /*----------- Create Child_mutated from Child-----------         */
3      if Child_mutated is better than Child then
4          findLocalOptimaFathers(Child_mutated);
5      end
6      else
7          findLocalOptimaFathers(Child);
8      end
9  end
10 Procedure: findLocalOptimaFathers(Selected_Child);
11 for i ← 1 to number_fathers do
12     if Current_father is local optima then
13         writeLONfile(Current_father,Selected_Child)
14     end
15 end
```

3.2.3. Obtención de redes de óptimos locales

La construcción de una red de óptimos locales requiere de una gran cantidad de informaciones (`file.LON`); proveniente de una gran cantidad de ejecuciones de los algoritmos de sintonización para poder crear la red. El motivo de la gran cantidad de ejecuciones, se debe a que permite obtener mayor cantidad de nodos de óptimos locales por los que pasó el algoritmo de sintonización y permite observar cómo el algoritmo comienza a construir la solución. Además, como los algoritmos utilizados son estocásticos, utilizar una baja cantidad de ejecuciones del algoritmo no es representativo pues dependen de la semilla aleatoeia utilizada. Tanto ParamILS como Evoca son algoritmos que trabajan con un conjunto de semillas aleatorias y utilizan un conjunto de instancias de donde se eligen las que serán utilizadas por el algoritmo. Lo anterior ocasiona que para el algoritmo objetivo a sintonizar, durante cada ejecución, esta se realiza con una semilla distinta y una instancia distinta, es decir, aunque la configuración de parámetros sea la misma, si se encuentran en distintas ejecuciones el resultado será distinto.

Las redes de óptimos locales necesitan que una configuración tenga solo un valor posible, es por ello que se decidió utilizar un promedio ponderado de todos los valores encontrados para una posible configuración y reemplazarlo en todos los archivos de una ejecución del algoritmo. Posteriormente se eliminan los *outliers* del conjunto de datos

ya que pueden entorpecer la visualización de la data y no son datos representativos. Luego, los archivos `file.LON` deben ser parseados y llevados al formato `file.dat` para generar los grafos mediante el lenguaje R, generando así el conjunto de datos `RData` sobre los cuales se puede graficar las redes de óptimos locales y obtener las métricas deseadas para comparar resultados y concluir.

3.3. Conclusiones del Capítulo

En este capítulo se propuso un enfoque basado en redes de óptimos locales(LONs) para el estudio de *fitness landscapes*. En primer lugar, se presentaron los fundamentos de la propuesta de solución, en donde se justifico el estudio de *fitness landscapes*, el uso de redes de óptimos locales para el estudio de estos y por que hacerlo con los algoritmos de sintonización: ParamILS y Evoca. En la segunda parte del capítulo se describió la metodología utilizada para la implementación de la solución. Se describió como se aplico la identificación de óptimos locales en los algoritmos de sintonización. En ParamILS se realizo posterior a la búsqueda local iterativa con *first improvement* generando así las transiciones entre óptimos locales. Por otro lado, Evoca identifica al óptimo local luego de obtener el individuo de la mutación y se compara con el individuo de cruzamiento, el mejor de estos genera arcos de transición con los padres del cruzamiento que sean óptimos locales. Finalmente dado el carácter estocástico en los *fitness landscapes* de algoritmos sintonizadores es necesario reparar los datos, para eso se calculo un promedio ponderado de las distintas calidades encontradas para una misma configuración.

En el siguiente capítulo, se presenta en detalle los experimentos que se realizaron con los algoritmos de sintonización, el algoritmo objetivo, las instancias y los parámetros utilizados.

CAPÍTULO 4
EXPERIMENTOS

Los experimentos realizados en este trabajo consisten en ejecutar los algoritmos de ParamILS y Evoca, para sintonizar los parámetros de un algoritmo genético que resuelve *NK landscapes* de 15 distintas categorías de instancias de problemas, y así obtener redes de óptimos locales para cada una de las categorías, y para cada uno de los algoritmos de sintonización. En este capítulo se describe el algoritmo objetivo a sintonizar con sus categorías de instancias y sus parámetros.

4.1. Algoritmo Objetivo

El proceso de sintonización de parámetros está enfocado en determinar los mejores valores de los parámetros que serán utilizados por un algoritmo objetivo para resolver un problema. En esta memoria, para los experimentos se consideró un algoritmo genético que resuelve el problema de los *NK landscapes*.

4.1.1. NK landscapes

El problema de los *NK fitness landscapes* fue introducido por Kauffman [11] como un modelo sintonizable de *rugged fitness landscape*. Un *NK fitness landscape* consiste en una función definida sobre cadenas binarias de largo fijo y está caracterizado por dos parámetros: (1) N, para el número total de bits y (2) K, para el tamaño del vecindario. Para cada bit se especifican los K vecinos y se da una función de fitness que determina la aptitud tanto del bit como de sus vecinos. *NK landscapes* es un problema NP-completo cuando $K > 1$, aunque algunas variantes de *NK landscapes* se pueden resolver en un tiempo polinomial y existen algoritmos de aproximación para otros casos, *NK landscapes* sigue siento un reto para cualquier algoritmo de optimización.

Definición del problema: Un *NK fitness landscape* se define formalmente por los siguiente componentes:

- El número de bits N
- El número de vecinos de un bit, K
- Un conjunto K de vecinos $\prod(X_i)$, para el i-ésimo bit; $\forall i \in \{0, \ldots, N-1\}$
- Una sub-función f_i la cual define un valor real para cada combinación de valores de X_i, y $\prod(X_i)$ $\forall i \in \{0, \ldots, N-1\}$. Típicamente, cada subfunción se define como una tabla de búsqueda con valores de tamaño 2^{K+1}.

La función objetivo f_{NK} a maximizar corresponde a:

$$f_{NK}(X_0, X_1, \ldots, X_{N-1}) = \sum_{0}^{N-1} f_i(X_i, \prod(X_i))$$

4.1.2. Algoritmo Genético para *NK landscapes*

El algoritmo a utilizar para resolver el problema de los *NK landscapes* es el propuesto por Pelikan en [18]. El algoritmo comienza generando una población de tamaño ps de manera aleatoria y se utiliza `BinaryTournamentSelection` para seleccionar a los padres. Los padres son transformados utilizando un operador de cruzamiento acorde a una probabilidad cr y un operador de mutación (*bit flip*) considerando una probabilidad mr. Algoritmo 6 muestra el pseudo-código del algoritmo genético recién descrito.

Algoritmo 6: Genetic Algorithm for NK Landscapes

```
1  P ← GenerateInitialPopulation(p_s);
2  while Maximum_Iterations are not reach do
3      EvaluatePopulation(P);
4      P_parents ← BinaryTournamentSelection(P);
5      while P_new is not complete do
6          Select two parents p_1 and p_2 from P_parents;
7          uniform_crossover(p_1, p_2, c_r);
8          mutation(p_1, m_r);
9          mutation(p_2, m_r);
10         P_new ← P_new ∪ p_1;
11         P_new ← P_new ∪ p_2;
12     end
13     P ← P_new ;
14 end
```

4.1.3. Instancias

Las instancias a utilizar son las mismas instancias propuestas en [18]. Estas corresponden a 15 conjuntos de instancias en donde varían N y K, cada uno de estos conjuntos de instancias cuenta con 1000 instancias.

K	2			3			4			5			6		
N	20	38	52	20	34	48	20	30	40	20	28	38	20	26	32

Tabla 2: Valores de N y K para los conjuntos de instancias de *NK-landscapes*

4.1.4. Parámetros

La tabla 3 muestra los parámetros y valores considerados durante los procesos de sintonización. La columnas de valores posibles y valores iniciales muestran el conjunto considerado por ParamILS y su valor inicial, mientras que Evoca trabaja en todo el rango.

Nombre	Descripción	Tipo	Valores posibles	Valores iniciales
c_r	tasa de cruzamiento	numérico (real)	$\{0,0,1,\ldots,0,9,1,0\}$	0,5
m_r	tasa de mutación	numérico (real)	$\{0,00,0,01,\ldots,0,24,0,25\}$	0,12
p_s	tamaño de la población	numérico (entero)	$\{3,4,\ldots,29,30\}$	0,18

Tabla 3: Parámetros del algoritmo genético para *NK-landscapes*

4.2. Configuración de los experimentos

Con lo anteriormente definido, lo que se busca realizar con los sintonizadores es encontrar la mejor configuración de parámetros que minimiza la cantidad de evaluaciones (cantidad de veces que se evalúa la función objetivo) que le toma algoritmo genético descrito encontrar el óptimo global, para el problema de los *Nk landscapes* (el óptimo global es conocido para los conjuntos de instancias utilizados). El algoritmo genético cuenta con una cantidad máxima de evaluaciones el cual fue fijado en 100.000, por otro lado para los conjuntos de instancias definidos en la tabla 2, se realizan 100 repeticiones por conjunto de instancias, ya que al ser algoritmos estocásticos se necesita una cantidad de información representativa del proceso de sintonización por algoritmo. Finalmente se tiene que los algoritmos sintonizadores necesitan una cantidad máxima de ejecuciones del algoritmo a sintonizar, este valor fue fijado en 160160 para ambos.

4.3. Entorno de experimentación

El algoritmo ParamILS para redes LON, fue modificado utilizando el lenguaje ruby en su versión 2.5.3.

El algoritmo Evoca para redes LON, fue modificado utilizando el lenguaje C++ con el compilador g++ versión 4.2.1.

El algoritmo Genético para *NK landscapes* programado en C, fue compilado con g++ versión 4.2.1 .

Los scripts para realizar la integración entre los algoritmos sintonizadores y las redes de óptimos locales, fueron programados en el lenguage Python en su versión 3.6.5.

Los scripts para generar los grafos de LONs y su visualización, fueron programados en el lenguaje R en su versión 3.5.1.

Las pruebas realizadas con los sintonizadores y la generación de datos para crear las redes de óptimos locales, fueron realizadas en una máquina virtual de Google Cloud [3] con 8 procesadores virtuales, 30GB de memoria RAM y sistema operativo Debian 9.

Las pruebas para generar las redes de óptimos locales, se realizaron en un computador con procesador Intel(R) Core(TM) i5-5257U CPU @2.7GHz, 8GB de RAM y sistema operativo macOS Mojave.

[3] aqui info google cloud

CAPÍTULO 5
RESULTADOS

Posterior a realizar los experimentos, se logró observar que con todos los datos generados para crear las redes se tiene demasiada información para realizar la visualización de estas, pero a la vez, se tiene suficiente información para obtener métricas y conclusiones representativas. Por este motivo, es que este capítulo se divide en 2 secciones. El primero corresponde a las métricas que se pueden extraer de todos los datos obtenidos mediante la creación de un grafo de óptimos locales. Se realiza un análisis para las distintas métricas obtenidas mediante tablas y gráficos. Posteriormente, con la información obtenida en la primera sección de este capítulo, se seleccionan dos conjuntos de instancias en donde los dos algoritmos de sintonización se comporten de manera muy similar y muy diferente, para así poder comparar entre diferentes casos. Con los conjuntos de instancias seleccionados, se realiza la visualización de las redes de óptimos locales con menos cantidad de datos y se hace un análisis tanto visual de la red como de indicadores específicos a obtener de los casos elegidos.

5.1. Métricas obtenidas para todas las instancias y ejecuciones

La obtención de métricas se realizo mediante archivos CSV de gran tamaño lo cuales contienen toda la información respecto a una red de óptimos locales y los *basins* de este, añadir las tablas en esta sección carece de sentido, puesto no se puede realizar un análisis de ellas. De todas formas el material se encuentra en un repositorio de GitHub (https://github.com/znxelox/LON_for_Tuning)

5.1.1. Cantidad de arcos y nodos

El primer aspecto a notar es el que la figura 9, donde se muestra el hecho de que el algoritmo ParamILS encuentra muchos menos óptimos locales que el algoritmo Evoca, es decir, la cantidad de nodos de Evoca es mucho mayor, obteniendo alrededor de 7000 nodos, mientras que para ParamILS la cantidad de nodos que se encuentra fluctúa entre 2000 y 3000. La figura 9 muestra lo anteriormente señalado por conjunto de instancias.

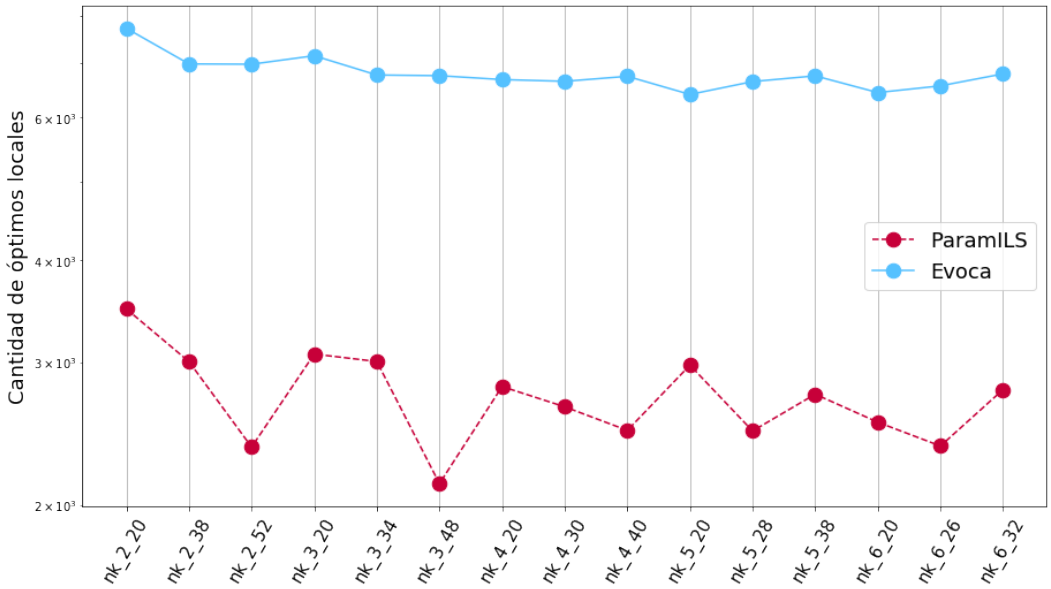


Figura 9: Cantidad de óptimos locales por sintonizador vs conjunto de instancias
Fuente: Elaboración propia.

La explicación lógica para la gran diferencia entre cantidad de óptimos locales encontrados, es la definición de óptimo local de cada algoritmo para identificar las transiciones entre nodos (ver sección 3.2). En el caso de ParamILS, el criterio para escribir una transición entre óptimos locales, se ve afectado por la gran cantidad de evaluaciones que necesitan las configuraciones de parámetros candidatas, ya que se busca encontrar soluciones de mejor calidad y esto puede ocurrir en pocas ocasiones, dado que la mayoría de las soluciones candidatas del vecindario de ParamILS tiene peor calidad. El único caso en que escapa de soluciones en que no mejora la calidad, es mediante perturbaciones. La conclusión a obtener de esto es que al algoritmo sintonizador le toma bastantes evaluaciones encontrar soluciones de mejor calidad, ya que al ser un algoritmo de búsqueda local iterativa con *first improvement* tiende a explorar poco el vecindario de soluciones. Por otro lado, Evoca identifica transiciones entre óptimos locales por cada padre del cruzamiento, si es que el padre corresponde a un óptimo local. Además, no necesita que la nueva solución tenga una mejor calidad, por lo cual escribe muchas más transiciones que ParamILS. Es decir, Evoca realiza una mayor exploración del espacio de búsqueda, permitiendo al algoritmo moverse entre configuraciones de peor calidad y luego mediante la mutación, que consiste en una búsqueda local (*Hill Climbing*), realizar una explotación del espacio de búsqueda.

Otro aspecto no menor a observar en la figura 9, es que al aumentar la dificultad del problema de los *NK landscapes*, es decir, aumentar el valor de K, Evoca encuentra una cantidad de nodos similar para los distintos conjuntos de instancias, pero con ParamILS el caso es distinto, en algunos conjuntos de instancias encuentra muchos menos nodos. Lo anterior puede sugerir que el rendimiento de ParamILS (cantidad de evaluaciones de

configuraciones de parámetros) se ve afectado ante escenarios de distinta dificultad, lo cual no necesariamente quiere decir que la calidad de sus configuraciones sean afectadas, si no más bien cuanto le cuesta encontrar una buena configuración.

En las figuras 10 y 11 se muestran la cantidad de arcos entre los óptimos locales encontrados por cada algoritmo algoritmo sintonizador en cada instancia. Las redes de óptimos locales cuentan con tres tipos de arcos: (1) Arcos de mejora, que corresponden al caso en el cual desde un óptimo local, el movimiento del algoritmo me lleva a otro óptimo local de mejor calidad (*fitness*), (2) Arcos de igualdad, estos arcos corresponden al caso en cual de un óptimo local el algoritmo sintonizador llega a otro de la misma calidad, y finalmente (3) Arcos de empeoramiento, que lógicamente llevan a un óptimo local de menor calidad. Estos arcos se muestran en cada gráfico de las figuras 10 y 11 como los arcos de mejora, empeoramiento e igualdad.

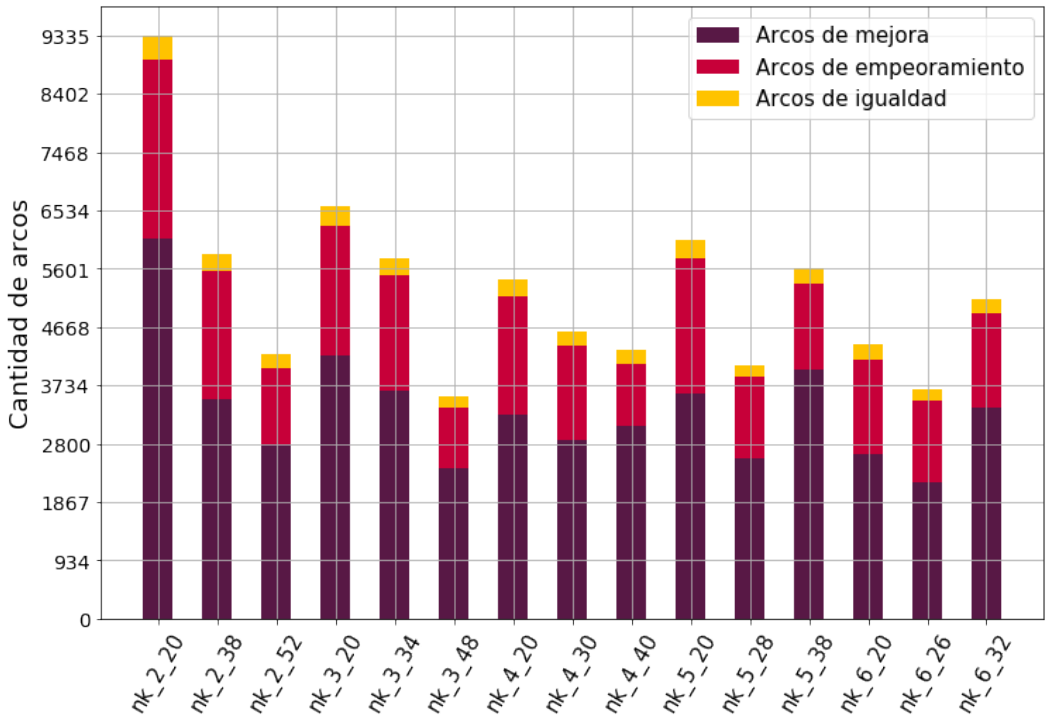


Figura 10: ParamILS: cantidad de arcos por tipo vs conjunto de instancias
Fuente: Elaboración propia.

Al observar las figura se nota claramente que al igual que con la cantidad de nodos, la cantidad de arcos es tremendamente superior en Evoca, tanto así que por ejemplo en el caso de la instancia `nk_2_20` ($K = 2$ y $N = 20$) la cantidad de arcos totales de ParamILS corresponde a un mínimo porcentaje de los arcos totales de Evoca. Pero más relevante es notar que en ParamILS lo que predomina son los arcos de mejora, y en Evoca predominan los arcos de empeoramiento, esta información respalda lo concluido sobre la cantidad de óptimos locales, ParamILS encuentra una cantidad mucho menor

de óptimos locales, pero se mueve la mayoría de las veces a un óptimo de mejor calidad, mientras que con Evoca el caso es totalmente contrario, mayor cantidad de nodos de peor calidad. Sobre los arcos de igualdad, para ambos algoritmo parece mantenerse el hecho de que no suelen ir comúnmente de un óptimo local a otro con igual calidad.

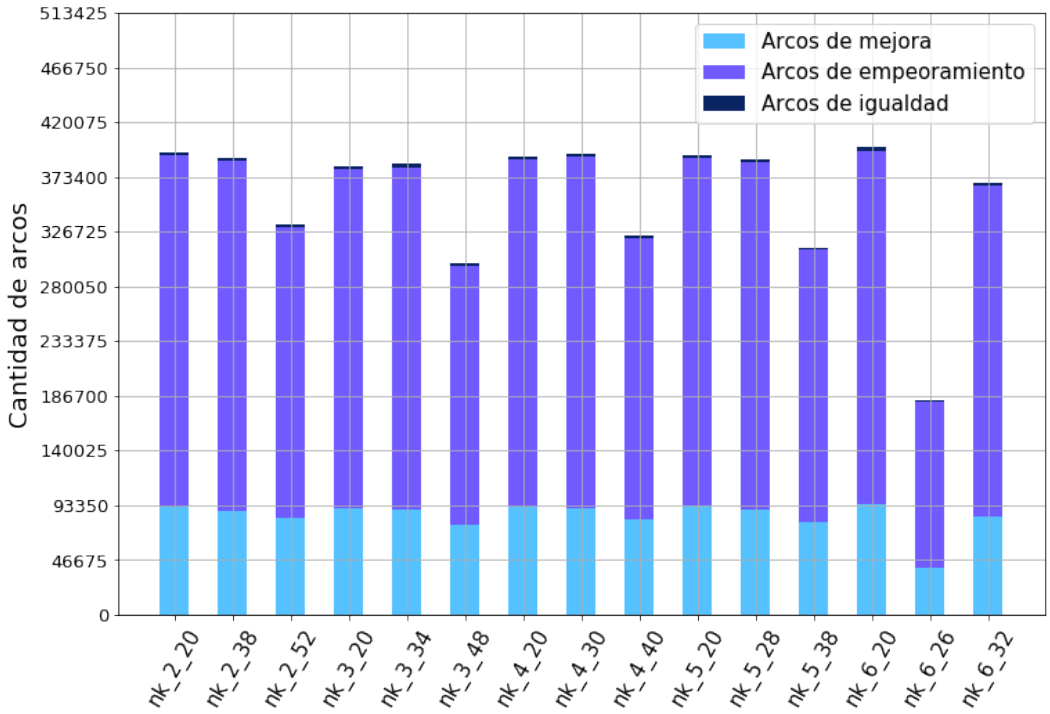


Figura 11: Evoca: cantidad de arcos por tipo vs conjunto de instancias
Fuente: Elaboración propia.

5.1.2. Calidad de las mejores soluciones

Al realizar el análisis de la cantidad de nodos, se observó el hecho de que ParamILS ve afectado su rendimiento en cuanto a cantidad de evaluaciones que le toma cambiar de solución a una mejor, mientras que Evoca se mantiene constante. En el caso de la calidad (*fitness*) de la mejor solución encontrada ocurre exactamente lo contrario. En la figura 12 se muestra el *fitness* del óptimo local vs los conjuntos de instancias por algoritmo sintonizador y se observa claramente como ParamILS aun cuando se aumente la dificultad del problema de los *NK landscapes* la calidad de solución encontrada se mantiene casi constante. Con Evoca ocurre todo lo contrario, es decir al variar la dificultad la calidad de la mejor solución encontrada, esta comienza a variar significativamente. Este aspecto es sumamente relevante, puesto que la sintonización de parámetros busca encontrar el mejor valor de los parámetros posibles el fin de obtener el mejor valor de lo que se busca optimizar (en este caso la cantidad de evaluaciones del algoritmo genético). El gráfico de la figura 12 además sugiere que el rendimiento del algoritmo

genético respecto al número de evaluaciones se ve afectado por las configuraciones de parámetros en las que se mueve Evoca. Evoca al aceptar nuevas configuraciones de menor calidad como óptimos locales quita la posibilidad de explotar mayormente el espacio de búsqueda y en consecuencia que el resultado de la calidad de la mejor solución sea irregular al cambiar la dificultad. Esto último es una gran debilidad de Evoca, ya que un algoritmo de sintonización de parámetros debe tratar de entregar resultados similares para el algoritmo objetivo en general, aunque las instancias aumenten su dificultad.

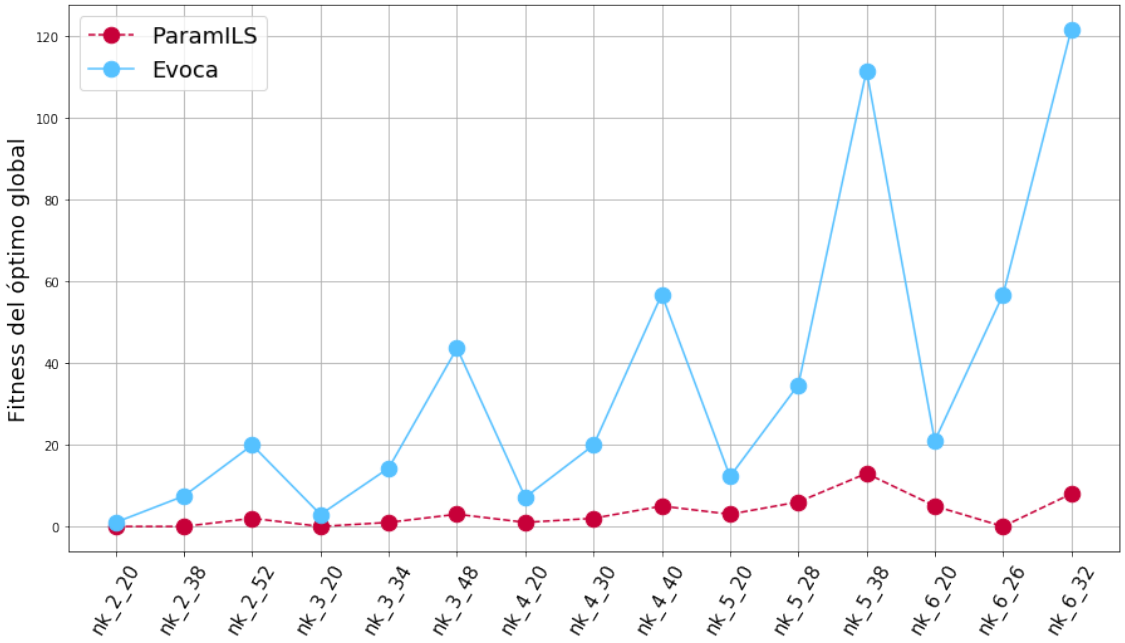


Figura 12: Fitness del óptimo global vs conjunto de instancia
Fuente: Elaboración propia.

5.1.3. *Basins* de atracción

Mediante las métricas de coeficiente de clustering o grado del grafo es posible obtener las *basins* de atracción para cada algoritmo sintonizador. Básicamente el procedimiento consiste en buscar los nodos más conectados (encontrar alguna estructura de comunidad) y que estén conectados con un óptimo local, luego se obtienen todos los nodos que se conectan mediante arcos de mejoramiento hasta llegar al óptimo local. El concepto de *basins* de atracción definido en la sección 2.5 resulta muy interesante para comprender los *fitness landscapes* que generan los algoritmos de sintonización.

El primer análisis sobre *basins* es la cantidad de estos que encuentra cada algoritmo de sintonización, la figura 13 muestra la cantidad de basins de atracción vs los distintos conjuntos de instancias. Se aprecia claramente que Evoca encuentra una mayor cantidad de *basins* lo cual puede deberse a que realiza una mayor exploración del espacio de

búsqueda. Fuera del caso particular de la instancia `nk_2_20` ambos algoritmos parecen mantenerse casi constantes respecto a la cantidad de de *basins* encontrados para los distintos conjuntos de instancias.

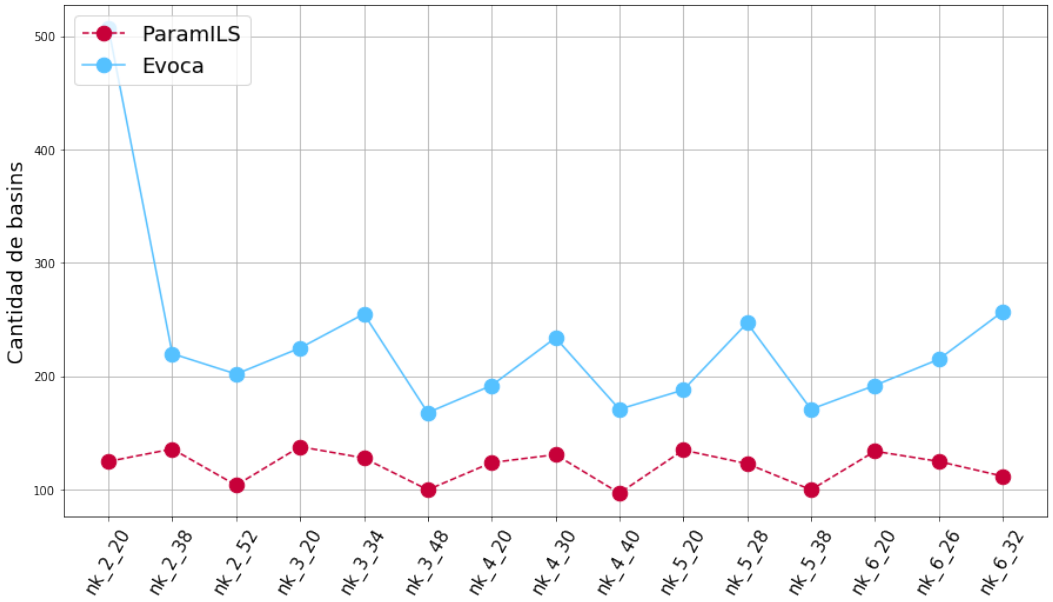


Figura 13: Cantidad de *basins* por sintonizador vs conjunto de instancias
Fuente: Elaboración propia.

Mediante los boxplots en la figura 14 se puede hacer un análisis un poco más profundo sobre la calidad del mejor óptimo local de cada *basin*, es interesante notar que si bien el valor del óptimo global en ParamILS es casi siempre el mismo, para sus *basins* no ocurre lo mismo. De lo anterior se puede inferir que ParamILS se mueve entre "cuencas" o "montañas" con una calidad bastante diferente a la del óptimo global, siendo en varios casos éste un outlier del conjunto de óptimos locales que están en la "cima de la montaña". Evoca en contra-parte, mantiene un comportamiento bastante similar al que muestran sus óptimos globales, por lo cual si lo llevamos a que los *basins* son montañas, todas las cimas están muy cercanas al óptimo global. Con lo recien descrito se puede tener ya una idea de cómo son los *fitness landscapes* de los algoritmos sintonizadores, si lo llevamos a la analogía de montañas o paisajes, ParamILS podría verse como montañas sin mucha altitud pero una de sus montañas es un pico alto que sobresale a las demás, mientras que desde el lado de Evoca su paisaje se asemeja más a una cordillera donde todas las montañas tienen casi la misma altitud y una es levemente más alta que las demás.

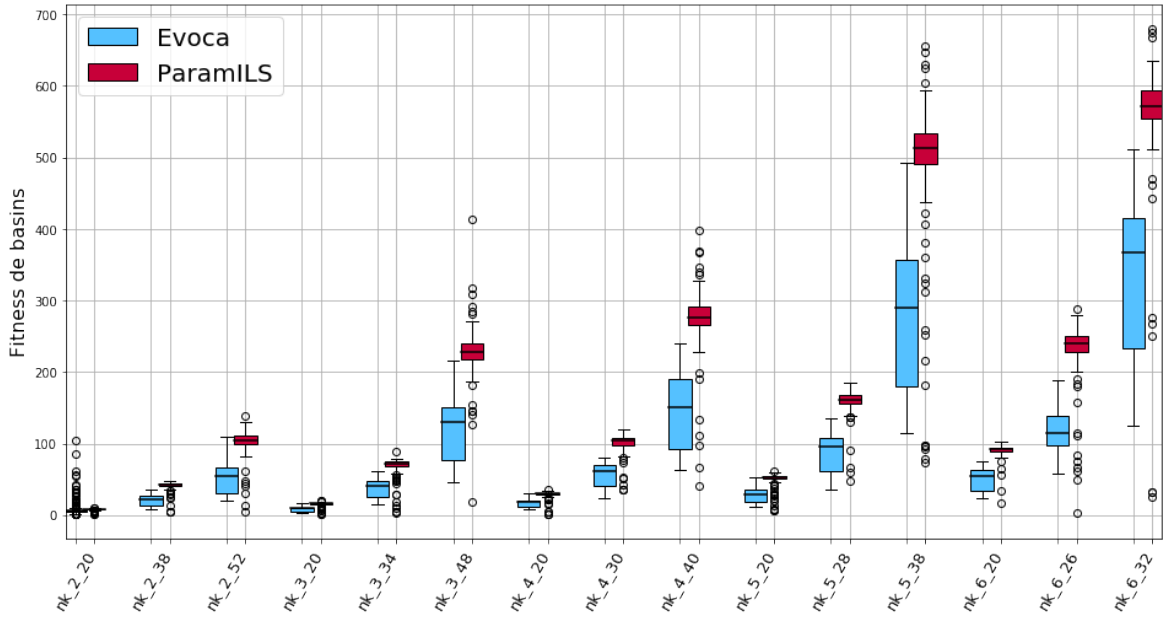


Figura 14: Boxplots fitness de *basins* por sintonizador vs conjunto de instancias
Fuente: Elaboración propia.

Un aspecto que también se debe analizar es el tamaño de los *basins* de atracción, el cual se define mediante la cantidad de óptimos locales que lo componen. La figura 15 muestra el tamaño de todos los *basins* encontrados por el algoritmo sintonizador en escala logarítmica para todos los conjuntos de instancias. Claramente Evoca supera con creces el tamaño de los *basins* encontrados por ParamILS y la explicación es bastante sencilla, Evoca explora más el espacio de búsqueda, encontrando soluciones de más baja calidad y por ello mismo sus *basins* cuentan con una mayor cantidad de nodos, los cuales en la mayoría de los casos logran conectar con soluciones de mejor calidad, generando así bastantes arcos de transición y por ende óptimos locales más conectados.

Es interesante observar también lo que sucede solo al mirar el *basin* del óptimo global. En la figura 16, se muestran los tamaños de los basins pertenecientes al óptimo global de cada conjunto de instancia en escala logaritmica. Acá se observa que en el caso de ParamILS el tamaño del *basin* del óptimo global, es inferior respecto al tamaño encontrado promedio de todos los *basins*. Por otro lado con Evoca sucede exactamente lo contrario, *basins* para el óptimo global de mayor tamaño al promedio de los demás. Esta información respalda lo mencionado anteriormente de que, mediante la analogía de montañas o paisajes, ParamILS corresponde a un conjunto de montañas con un gran pico sobresaliente a los demás (bastante delgado) mientras que Evoca se encuentra entre varias montañas del casi la misma altura, pero el óptimo global pertenece a la más alta y más robusta.

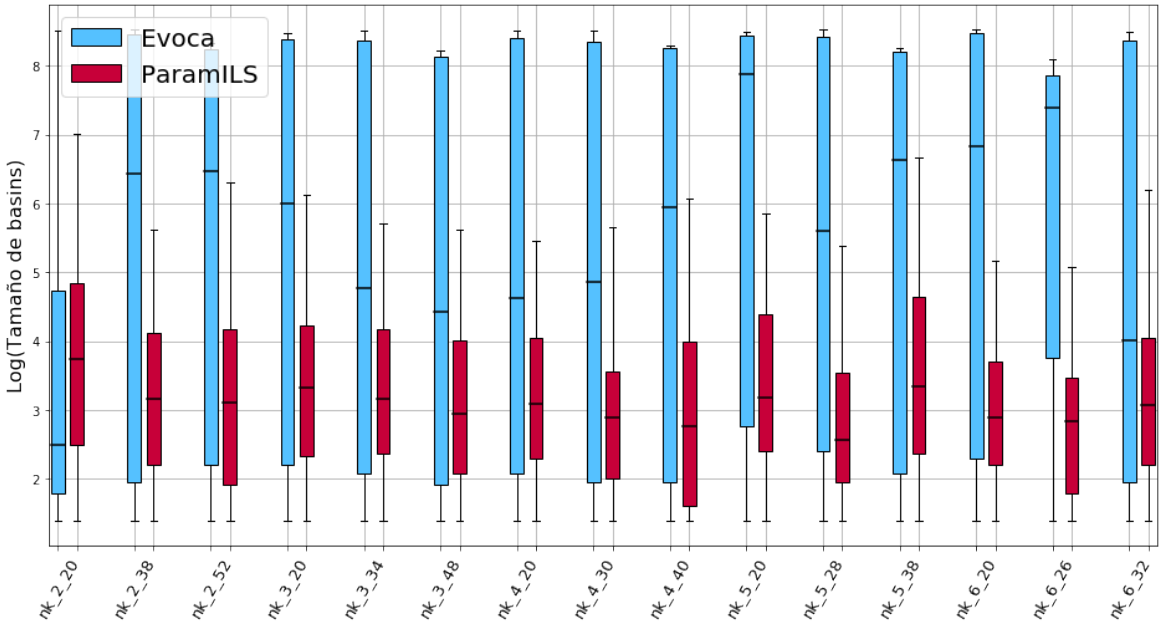


Figura 15: Boxplots tamaño de *basins* por sintonizador vs conjunto de instancias
Fuente: Elaboración propia.

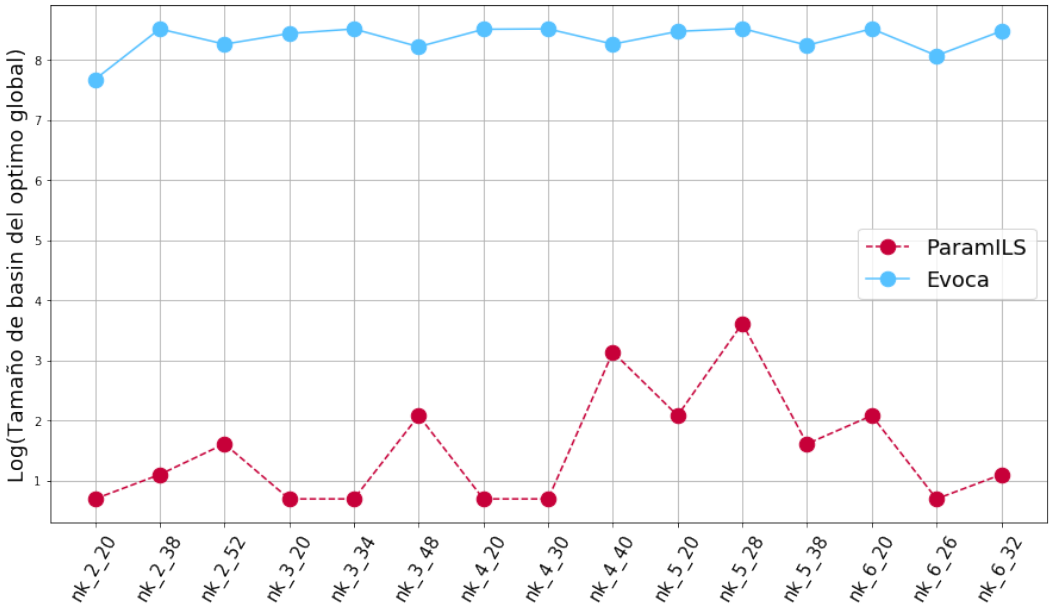


Figura 16: Tamaño de *basin* del óptimo global por sintonizador vs conjunto de instancias
Fuente: Elaboración propia.

5.2. Visualización de redes de óptimos locales

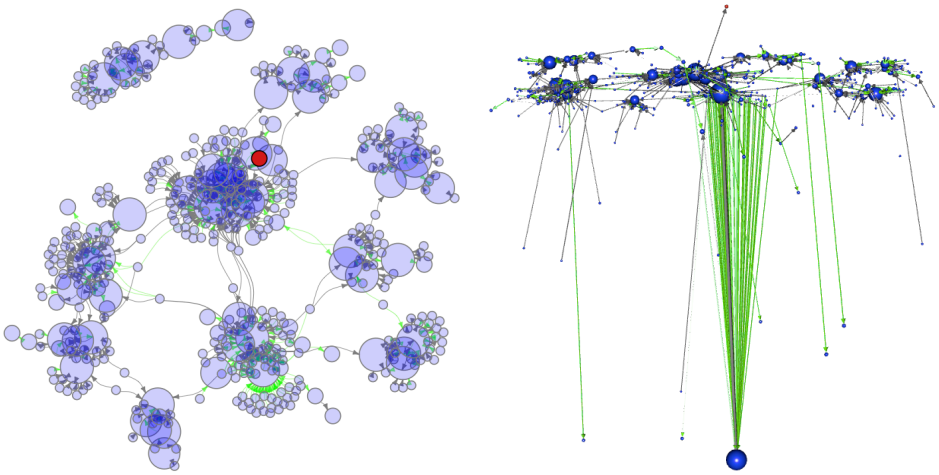
Luego del análisis de las métricas obtenidas con todo el conjunto de datos se debe seleccionar los conjuntos de instancias a analizar visualmente. El primer conjunto elegido es para los valores de $K = 2$ y $N = 20$, puesto que tras el análisis gráfico se observó que este conjunto de instancia se comporta de forma similar para ambos sintonizadores. El caso contrario ocurre con los valores de $K = 6$ y $N = 32$ en donde el comportamiento de los algoritmos es bastante distinto. Obviamente este último es el segundo conjunto elegido. Las redes de óptimos locales utilizadas para las visualizaciones fueron construidas con 10 ejecuciones de cada algoritmo.

Para el análisis visual de las redes de óptimos locales es necesario identificar sus componentes para realizar una buena interpretación de estas. Primero que todo los nodos que tienen un color azul corresponden a los óptimos locales, el tamaño de estos nodos depende de cúantas veces el algoritmo sintonizador pasó por ese óptimo local, es decir, qué tan recurrente es esa solución. Los nodos rojos, por otro lado, corresponden a cuando se alcanzo el óptimo global. Por otro lado los arcos de la red tienen distintos colores, para los arcos de mejoramiento se utiliza un color gris oscuro, mientras que para los arcos de empeoramiento se utiliza un color verde, los arcos de igualdad al ser tan pocos no aparecen en las visualizaciones. El grosor de los arcos está dado por la cantidad de veces que se pasó por ese arco. Para las visualizaciones en tres dimensiones la altura se interpreta como la calidad de la solución. Es así como los nodos de color rojo siempre se encontrarán en la parte superior, los nodos grises deben tener aunque sea una mínima pendiente positiva y por el contrario los nodos verdes una mínima pendiente negativa.

Las variantes de las redes de óptimos locales, MLON (*Monotonic*) y CMLON (*Compresed Monotonic*) tienen también diferencias en la visualización respecto a LONs. En el caso de MLONs, en estas redes ya no existen los arcos de empeoramiento, y solo posee arcos gris oscuro de mejoramiento. Para las redes del tipo CMLON por otro lado se encuentran nuevos colores en los nodos, donde los nodos color turquesa son los nodos mediante se llegan a los óptimos locales de mejor calidad, los cuales tienen un color celeste. Para el nodo del óptimo global particularmente se tiene que para todos los nodos que llevan al óptimo global se marcan con un color rosado lo que permite identificar el *basin* del óptimo global.

5.2.1. ParamILS: instancia $k = 2$ y $n = 20$

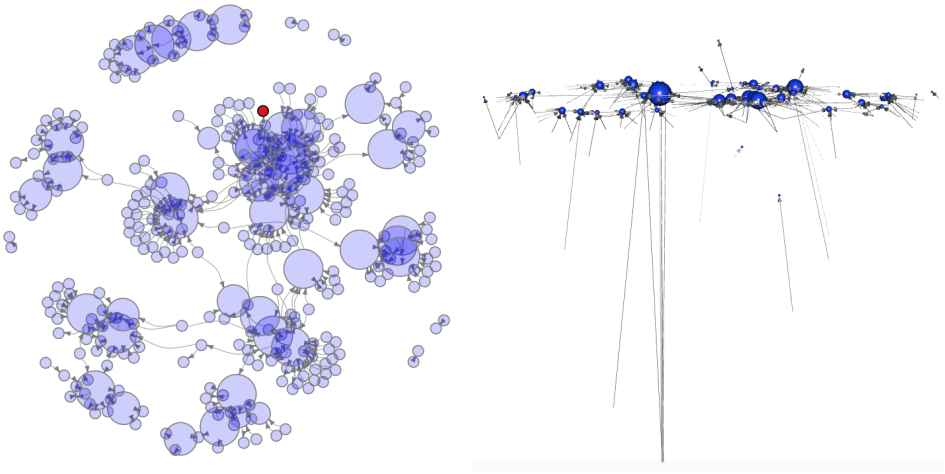
Al observar la visualización en dos dimensiones de la figura 17 lo primero que resalta es que los nodos parecen estar agrupados o tener una estructura de comunidades. Pueden darse 2 explicaciones ante esta situación, la primera puede ser que se traten de distintas ejecuciones de algoritmo sintonizador, al ser estas ejecuciones estocásticas puede que se trabaje en áreas del espacio de búsqueda distintas y no lograron conectarse mediante una transición de óptimos locales. La segunda opción es similar pero puede ser que al ser ParamILS un algoritmo de búsqueda local iterativa que acepta solo nuevas configuraciones de mejor o igual calidad se aleja de la sección del espacio de búsqueda mediante una perturbación.



(a) Visualización LON 2D (b) Visualización LON 3D

Figura 17: Visualización LON para ParamILS con instancia $k = 2$ y $n = 20$

Otro detalle a notar en la figura 17, es que en la visualización 3D se ve claramente cómo la mayoría de los óptimos locales se encuentran casi a la misma altura, es decir tienen una calidad bastante similar, y el óptimo global se encuentra apartado a una mayor altura. Esto se puede apreciar mucho mejor en la visualización 3D de la figura 18, al desaparecer los nodos de empeoramiento se nota claramente que ParamILS suele moverse ente soluciones de una calidad bastante similar, apoyando lo encontrado al realizar el análisis de las métricas para todos los datos.



(a) Visualización MLON 2D (b) Visualización MLON 3D

Figura 18: Visualización MLON para ParamILS con instancia $k = 2$ y $n = 20$

Finalmente la figura 19 permite realizar un análisis de los *basins* de atracción encontrados por el sintonizador. El aspecto más relevante a destacar en esta visualización es el bajo tamaño del *basin* de óptimo global, pudiendose apreciar solo 2 nodos color rosado, nuevamente hace sentido a las métricas obtenidas en la sección anterior donde se observo que ParamILS posee *basins* de tamaño pequeño.

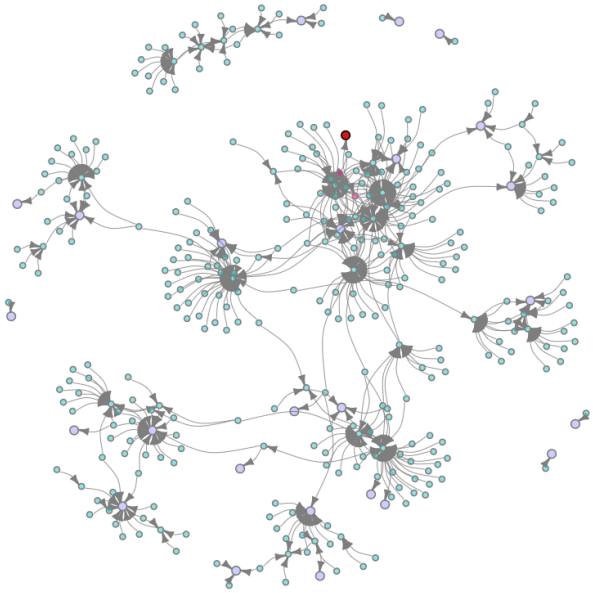
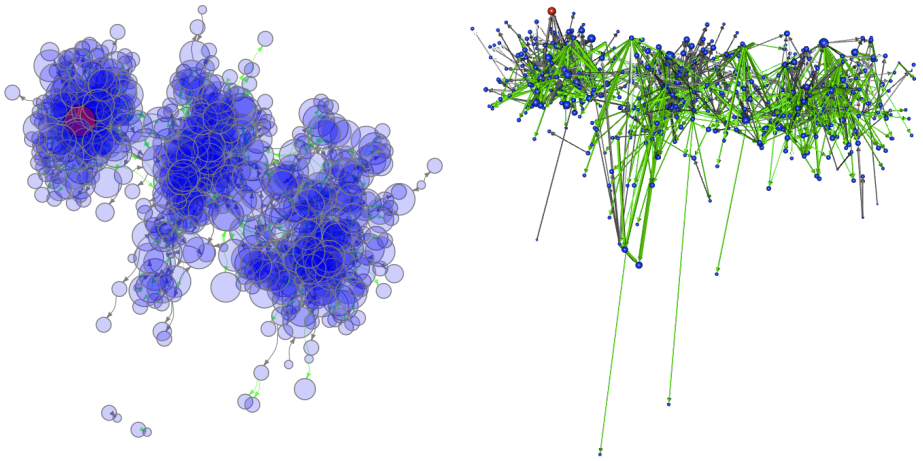


Figura 19: Visualización CMLON para ParamILS con instancia $k = 2$ y $n = 20$
Fuente: Elaboración propia.

5.2.2. Evoca: instancia $k = 2$ y $n = 20$

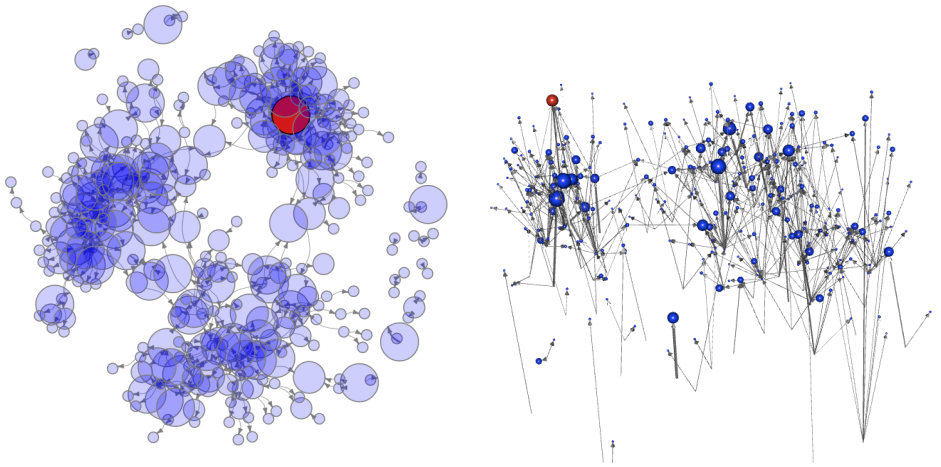
Para la misma instancia anterior pero ahora con Evoca como algoritmo sintonizador las visualizaciones resultan muy distintas (ver figura 20). Evoca presenta de forma significativa una mayor cantidad de nodos, los cuales no se encuentran agrupados de la misma forma que en el caso de ParamILS. El hecho de que Evoca presente esta estructura puede deberse a que, al realizar una mayor exploración del espacio de búsqueda, generan una mayor conexión entre distintas zonas del espacio. La visualización 3D además muestra claramente los arcos de empeoramientos como dominantes, no solo son una mayor cantidad sino que el grosor de los nodos de color verde es mayor a los de color gris, todo esto es esperado dado el análisis de las métricas obtenidas. Otro detalle a observar es el tamaño del nodo correspondiente al óptimo global, el cual es mucho mayor que en el caso de ParamILS, esto quiere decir que se llega al óptimo global más veces, pero no necesariamente que sea de la misma calidad o mejor que el de ParamILS.



(a) Visualización LON 2D (b) Visualización LON 3D

Figura 20: Visualización LON para Evoca con instancia $k = 2$ y $n = 20$

Para las redes de tipo MLON ilustradas en la figura 21 se aprecia de forma clara que Evoca contruye los *basins* en forma de montañas casi de altura similar al óptimo global, muy distinto a ParamILS donde el óptimo global destacaba solitario sobre una especie de *plateau*. Respecto al análisis del *basin* del óptimo global, la figura 22 muestra claramente como Evoca posee una cantidad mayor de nodos que pertenecen al *basin*, respecto a ParamILS, y además muestra lo mucho más conectada que se encuentra la red.



(a) Visualización MLON 2D (b) Visualización MLON 3D

Figura 21: Visualización MLON para Evoca con instancia $k = 2$ y $n = 20$

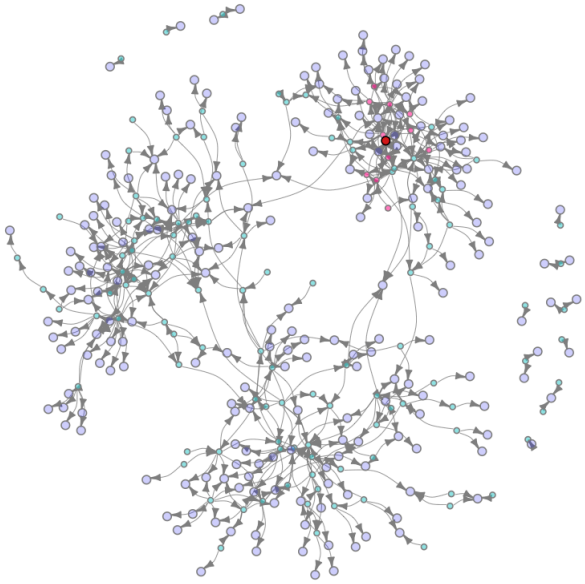


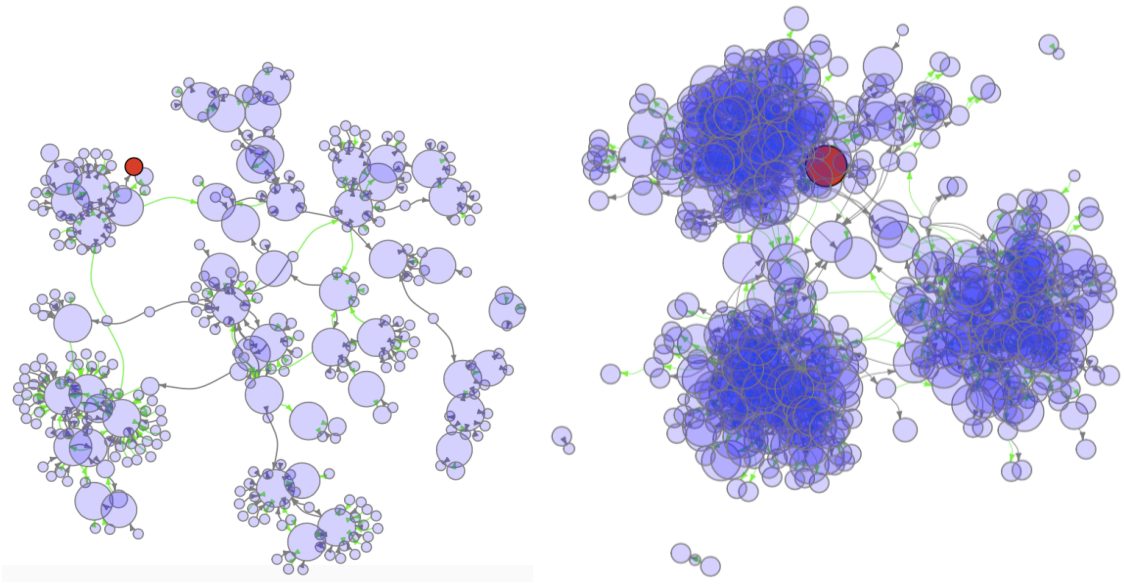
Figura 22: Visualización CMLON para Evoca con instancia $k = 2$ y $n = 20$
Fuente: Elaboración propia.

Luego de analizar los conjuntos de instancias, que mostraron comportamientos similares para los dos algoritmos de sintonización, mediante la visualización de las redes de óptimos locales creadas con los datos generados por ambos algoritmos sintonizadores de parámetros, se pudo corroborar la información obtenida al analizar las métricas obtenidas. Evoca encuentra una cantidad mayor de nodos y arcos que ParamILS, los arcos de ParamILS en su mayoría son arcos de mejora. Evoca presenta una mayor cantidad de *basins* al observar las visualizaciones de MLON respecto a paramILS. El *basin* del

óptimo local de Evoca es de mayor tamaño al de ParamILS. Finalmente, la conclusión que tal vez es más relevante es la corroboración del tipo de *fitness landscape* que construye cada uno de los algoritmos, efectivamente, ParamILS mediante la visualización confirmó que sus *basins* no tienen una gran diferencia de calidad entre ellos (esto incluye el mejor óptimo del *basin*) y que el óptimo global se encuentra a una calidad significativamente distinta al resto de los óptimos locales. Evoca, al contrario mostró que la mayoría de sus *basins* se acercan en calidad a su óptimo global.

5.2.3. Comparación instancia $k = 6$ y $n = 32$

Al realizar el análisis de las visualizaciones de redes tipo LON, MLON y CMLON para el nuevo conjunto de instancias se encuentran bastantes similitudes con el conjunto de instancias anterior. En efecto parece ser que las diferencias entre los dos algoritmos sintonizadores se acentúan de mayor forma comparado con el conjunto de instancia anterior.

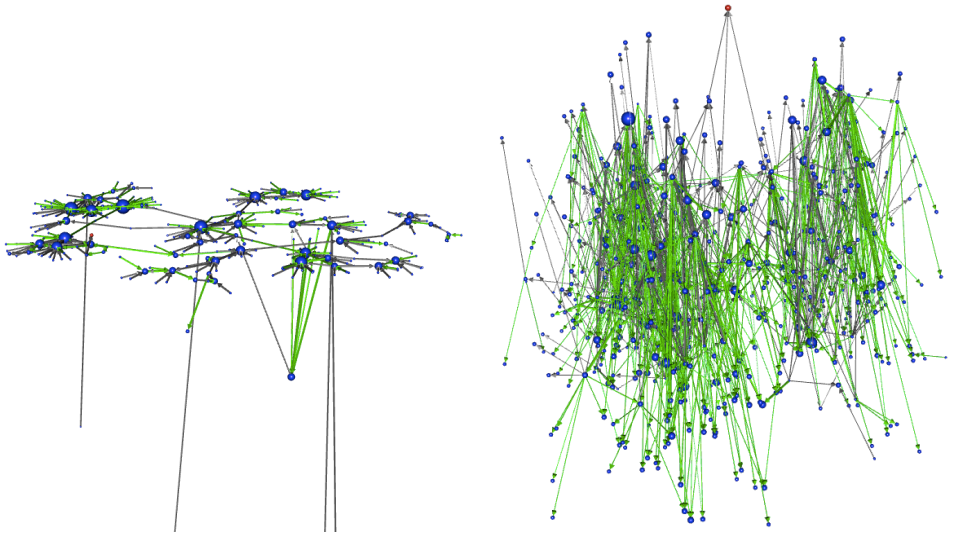


(a) ParamILS LON 2D (b) Evoca LON 2D

Figura 23: Comparación LON 2D entre sintonizadores con instancia $k = 6$ y $n = 32$

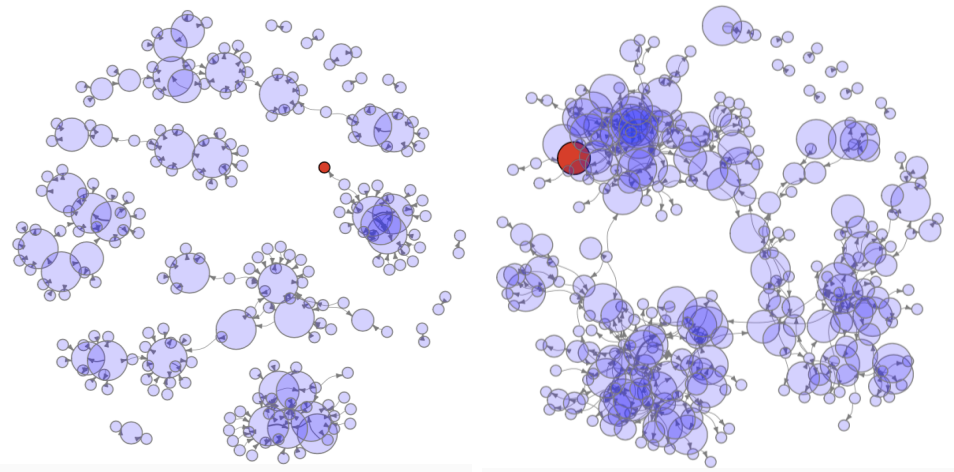
La figura 23 muestra como nuevamente Evoca cuenta con una mayor cantidad de nodos que ParamILS, y el óptimo global parece tener mayor recurrencia respecto al óptimo global de ParamILS. Evoca presenta una clara *clusterización* de sus nodos, ya que se aprecian fácilmente tres grandes conjuntos de nodos altamente conectados, contrario al caso de ParamILs donde a pesar de si observar estructuras de comunidades esto ocurre por la poca conexión entre ellas. Las visualizaciones en 3D (ver figura 24) de ambos sintonizadores muestran la clara diferencia del *fitness landscape* que construyen ambos

algoritmos. Nuevamente se corrobora la idea de que ParamILS construye planicies estilo *plateau* con óptimos locales de alta calidad, mientras que Evoca construye muchos *basins* los cuales parten desde una muy baja calidad hasta calidad casi iguales al óptimo global. Lo anterior puede ser apreciado de mejor forma en la figura 26 al observar solo los arcos de mejoramiento.



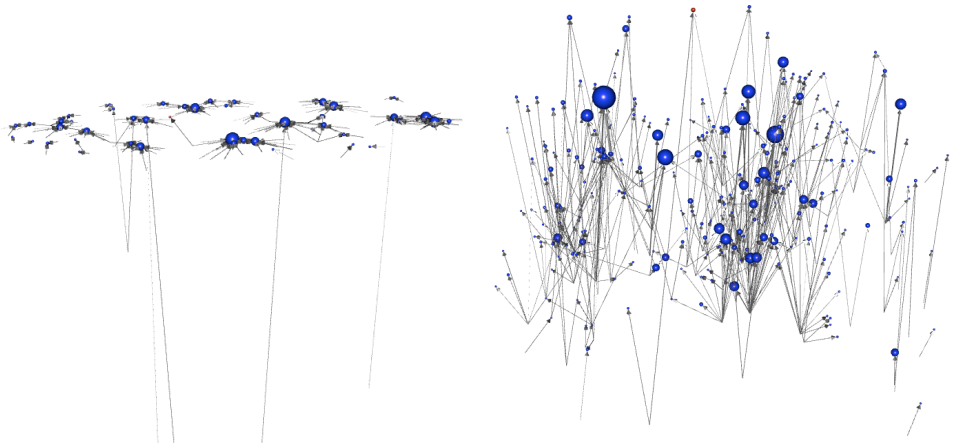
(a) ParamILS LON 3D (b) Evoca LON 3D

Figura 24: Comparación LON 3D entre sintonizadores con instancia $k = 6$ y $n = 32$



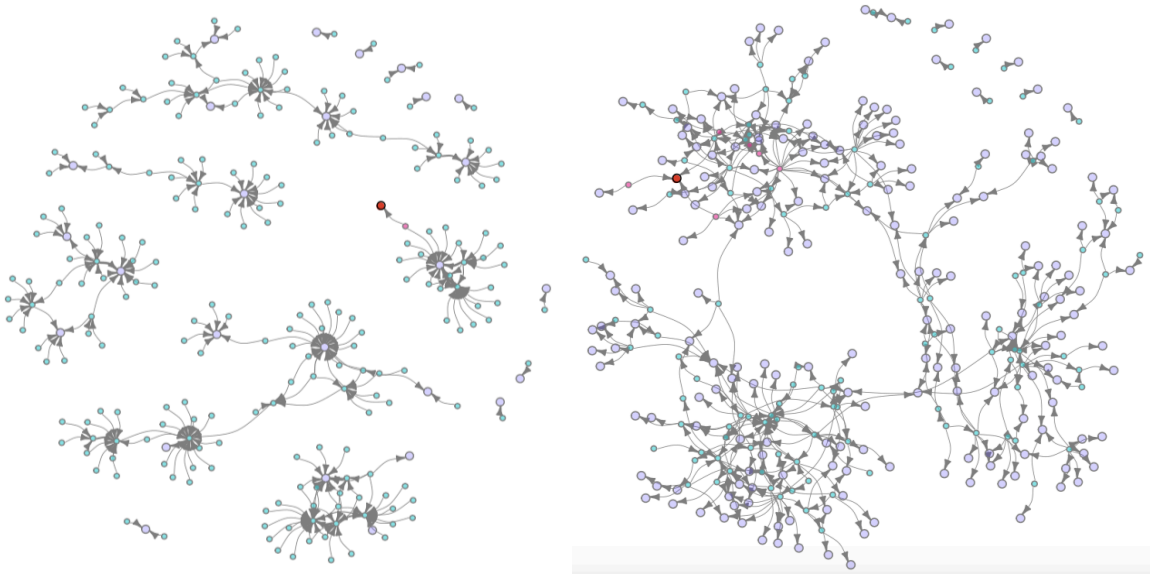
(a) ParamILS MLON 2D (b) Evoca MLON 2D

Figura 25: Comparación MLON 2D entre sintonizadores con instancia $k = 6$ y $n = 32$



(a) ParamILS MLON 3D (b) Evoca MLON 3D

Figura 26: Comparación MLON 3D entre sintonizadores con instancia $k = 6$ y $n = 32$



(a) ParamILS CMLON 2D (b) Evoca CMLON 2D

Figura 27: Comparación CMLON 3D entre sintonizadores con instancia $k = 6$ y $n = 32$

Respecto a las redes del tipo CMLON la figura 27 deja claro el tipo de agrupación que ocurre para los dos sintonizadores, mientras que ParamILS agrupa sus nodos en base a la poca conexión existente entre ellos, Evoca lo hace en base al exceso de conexiones que hay. Por otro lado nuevamente al obsercar los nodos de color rosado se ve que el tamaño del *basin* del óptimo global es muy pequeño para ParamILS, no así para Evoca.

5.3. Conclusiones del Capítulo

En este capítulo se mostraron los resultados obtenidos al realizar la integración entre sintonización de parámetros y redes de óptimos locales para el estudio de los *fitness landscapes*, que generan los algoritmos de sintonización. Mediante las métricas, a partir de las redes de óptimos locales, se logró realizar un análisis bastante profundo sobre los *fitness landscapes* de cada algoritmo. El gran descubrimiento realizado corresponde a la estructura del *fitness landscapes*, dado el tipo de búsqueda que realizan los algoritmos. Mientras que ParamILS, explota el espacio de búsqueda, escribe pocas transiciones entre óptimos locales, y esto se traduce en una baja cantidad de nodos en la red y que la mayoría de los arcos sean de mejora. Además, se descubrió que los *basins* de atracción de ParamILS tienden a estar compuestos de una baja cantidad de óptimos locales, los cuales poseen una gran calidad pero significativamente menor a la del óptimo global. En cambio Evoca, realiza una exploración del espacio de búsqueda más que explotación, por lo cual escribe muchas veces transiciones entre óptimos locales. Lo anterior se traduce en una gran cantidad de nodos y una gran cantidad de arcos, donde los arcos predominantes son los de empeoramiento. Respecto a los *basins* de atracción, Evoca cuenta con una gran cantidad, donde cada uno de ellos tiende a tener una gran cantidad de óptimos locales y los óptimos de mejor calidad de la mayoría de los *basins* se asemejan en calidad al óptimo global.

Finalmente, si se utiliza la analogía de montañas o paisajes a los *fitness landscapes*, una buena representación de cada uno de los algoritmos seria la que se muestra en la figura 28.

(a) ParamILS: Gran montaña con planicies y un pico

(b) Evoca: Múltiples montañas de altura cercana

Figura 28: Tipo de paisaje o “montaña” por sintonizador

CAPÍTULO 6
CONCLUSIONES

El problema de la configuración de parámetros es un asunto de suma importancia en el área de la optimización meta-heurística. Los algoritmos que resuelven estos problemas pueden contar con una gran cantidad de parámetros, en consecuencia , encontrar los mejores valores de los parámetros resulta un proceso bastante complejo. En esta memoria se propuso un enfoque basado en el estudio de los *fitness landscapes* que generan los algoritmos de sintonización para así comprender mejor el proceso, tanto de cada algoritmo sintonizador, como el proceso en general.

El estudio de los *fitness landscapes* asociados a cada algoritmo, se realizó mediante la integración redes de óptimos locales a los algoritmos de sintonización, permitiendo así, trabajar sobre el espacio de óptimos locales que encuentra cada algoritmo sintonizador como si se tratase de un grafo. Lo anterior, permite el uso de herramientas disponibles en redes complejas, como la obtención de métricas de cada red y la visualización de la red, en consecuencia se obtiene una visualización del *fitness landscape* del algoritmo sintonizador, algo no hecho hasta ahora.

El primer algoritmo, ParamILS, fue integrado a LONs mediante una identificación de óptimo local cada vez que algoritmo realizo un movimiento en el espacio de búsqueda de una configuración de parámetros a otra. Evoca, el segundo algoritmo escogido se integro con LONs mediante una definición de óptimo local basada en cruzamiento de configuraciones de parámetros consideradas padres, una configuración hijo y una configuración de mutación. Posteriormente se debió hacer una reparación de los datos obtenidos, ya que al ser la sintonización de parámetros estocástica, se tenían distintas calidades para una misma configuración, es por ello que se calculo el promedio ponderado para todos los valores de una configuración de parámetros y fue se reemplazo en los archivos para generar las redes de óptimos locales.

Los experimentos realizados por los sintonizadores con el algoritmo genético que resuelve el problema de los *NK landscapes*, permitieron realizar comparaciones bajo distintas dificultades y entre los algoritmos de sintonización. A través de los resultados obtenidos se pudo observar que ParamILS es un algoritmo que explota el espacio de búsqueda, lo que en consecuencia se traduce en un *fitness landscape* con óptimos locales de alta calidad, pero en baja cantidad. Además, los *basins* encontrados en ParamILS indican ser pequeños y alejados de la calidad del óptimo global. Si pensamos en el *fitness landscape* de paramILS como una montaña, sería una montaña bastante alta, con una especie de planicie en su cima y un pico que destaca en esta planicie, en donde se encuentra el óptimo global. En contraparte, Evoca mostró comportamientos de un algoritmo con fuerte énfasis en la exploración, a pesar de tener un operador de mutación enfocado en explotar. Evoca encuentra una gran cantidad de óptimos locales, debido a su explo-

ración del espacio de búsqueda, y crea por eso mismo una gran cantidad de arcos de transición entre estos óptimos, siendo la mayoría arcos de empeoramiento. Los *basins* de Evoca, mostraron tener un gran tamaño, y llegar en su óptimo localde mejor calidad a valores cercanos a la calidad del óptimo global. Aplicando la misma analogía de montaña mencionada con ParamILS, el *fitness landscape* de Evoca puede ser representado por un conjunto de montañas para los cuales todas las cimas están a una altura similar.

Además Evoca demostró ser un algoritmo bastante informativo, el cual en este sentido favorece el diseño de meta-heurísticas. Por otro lado ParamILS es más dependiente de los valores de los parámetros iniciales, depende donde parta que tan buena sea la ejecución ya que explota el espacio de búsqueda.

Como se observó, los dos algoritmos estudiados presentaron *fitness landscapes* bastante distintos. Lo anterior, nos lleva a concluir que el proceso de sintonización depende mucho del tipo de movimiento que se realice en el espacio de búsqueda. Una probabilidad más alta de realizar una perturbación en paramILS, o un operador de mutación más agresivo (mayor explotación) en Evoca podrían hacer que ambos algoritmos tengan un comportamiento similar. La información obtenida mediante los resultados permite tener un gran conocimiento de como funciona cada algoritmo sintonizador, sus debilidades y fortalezas. Este punto es muy importante, pues al identificar debilidades y fortalezas se puede mejorar el proceso en bases a esta información. Con todo lo mencionado anteriormente, queda claro que sin lugar a dudas, el uso de herramientas como las redes de óptimos locales es una gran ayuda para entender este campo sin investigación y continuar mejorando el proceso de sintonización de parámetros.

6.1. Trabajo Futuro

Respecto al proceso de sintonización se propone utilizar escenarios con más parámetros y realizar un estudio que varíe la cantidad de valores de cada parámetro. Se propone realizar mejoras en las definiciones de óptimo local para cada algoritmo sintonizador. Para las transiciones entre óptimos locales se podría utilizar arcos con probabilidad de transición entre *basins*, o arcos de escape de óptimos locales (perturbación) los cuales entregan mayor información de características especificas del proceso de sintonización. Utilizar el *PageRank* en las redes de óptimos locales según lo estudiado en la literatura parece ser una gran opción para predecir el rendimiento de los algoritmo de optimización, integrarlo al proceso de sintonización podría entregar información nueva sobre la importancia de los nodos más dominantes. Finalmente, respecto a los algoritmos de sintonización escogidos en este trabajo, realizar modificaciones a ParamILS y Evoca en los movimientos que toman en el espacio de búsqueda y volver a ejecutar los experimentos en las mismas condiciones para corroborar si se pudo mejorar el proceso de sintonización de cada algoritmo bajo las conclusiones obtenidas.

ANEXOS

REFERENCIAS BIBLIOGRÁFICAS

[1] Carlos Ansótegui, Meinolf Sellmann, and Kevin Tierney. A gender-based genetic algorithm for the automatic configuration of algorithms. In Ian P. Gent, editor, *Principles and Practice of Constraint Programming - CP 2009*, pages 142–157, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[2] Thomas Bartz-Beielstein, Christian Lasarczyk, and Mike Preuss. Sequential parameter optimization. In *Congress on Evolutionary Computation*, 2005.

[3] Mauro Birattari and Thomas Stützle and Luís Paquete and Klaus Varrentrapp. A racing algorithm for configuring metaheuristics. In *GECCO*, 2002.

[4] Johann Dréo. Using performance fronts for parameter setting of stochastic metaheuristics. In *GECCO*, 2009.

[5] Sebastian Herrmann. *Determining the Difficulty of Landscapes by PageRank Centrality in Local Optima Networks*, pages 74–87. 2016. Exported from https://app.dimensions.ai on 2018/10/23.

[6] Sebastian Herrmann, Gabriela Ochoa, and Franz Rothlauf. Pagerank centrality for performance prediction: the impact of the local optima network model. *Journal of Heuristics*, 24(3):243–264, Jun 2018.

[7] Sebastian Herrmann and Franz Rothlauf. Predicting heuristic search performance with pagerank centrality in local optima networks. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, pages 401–408, New York, NY, USA, 2015. ACM.

[8] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proc. of LION-5*, page 507–523, 2011.

[9] Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown, and Thomas Stützle. Paramils: An automatic algorithm configuration framework. *J. Artif. Int. Res.*, 36(1):267–306, September 2009.

[10] Frank Hutter, Holger H. Hoos, and Thomas Stützle. Automatic algorithm configuration based on local search. In *AAAI*, 2007.

[11] S. A. KAUFFMAN. Adaptation on rugged fitness landscapes. *Lectures in the Science of Complexity*, 1:527–618, 1989.

[12] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, and Thomas Stützle and Mauro Birattari. The irace package: Iterated racing for automatic algorithm configuration. 2011.

[13] R.E. Mercer and J.R. Sampson. Adaptive search using a reproductive metaâ plan. *Kybernetes*, 7(3):215–228, 1978.

[14] Elizabeth Montero, María Cristina Riff, and Bertrand Neveu. A beginner's guide to tuning methods. *Appl. Soft Comput.*, 17:39–51, 2014.

[15] Volker Nannen and A. E. Eiben. Relevance estimation and value calibration of evolutionary algorithm parameters. In *IJCAI*, 2007.

[16] Gabriela Ochoa, Francisco Chicano, Renato Tinós, and Darrell Whitley. Tunnelling crossover networks. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, pages 449–456, New York, NY, USA, 2015. ACM.

[17] Gabriela Ochoa, Marco Tomassini, Sébastien Verel, and Christian Darabos. A study of nk landscapes' basins and local optima networks. In *Genetic and Evolutionary Computation Conference - GECCO 2008*, pages 555–562. ACM, 2008.

[18] Martin Pelikan, David E. Goldberg, and Fernando G. Lobo. A survey of optimization by building and using probabilistic models. *Comp. Opt. and Appl.*, 21:5–20, 2002.

[19] Yasha Pushak and Holger Hoos. Algorithm configuration landscapes: - more benign than expected? In Anne Auger, Carlos M. Fonseca, Nuno Lourenço, Penousal Machado, Luís Paquete, and Darrell Whitley, editors, *Parallel Problem Solving from Nature - PPSN XV - 15th International Conference, Coimbra, Portugal, September 8-12, 2018, Proceedings, Part II*, volume 11102 of *Lecture Notes in Computer Science*, pages 271–283. Springer, 2018.

[20] María Cristina Riff and Elizabeth Montero. A new algorithm for reducing metaheuristic design effort. In *IEEE Congress on Evolutionary Computation*, 2013.

[21] Marc Schoenauer, Pierre Savéant, and Vincent Vidal. Divide-and-evolve: a new memetic scheme for domain-independent temporal planning. In *EvoCOP*, 2006.

[22] Selmar K. Smit, A. E. Eiben, and Zoltán Szlávik. An moea-based method to tune ea parameters on multiple objective functions. In *IJCCI*, 2010.

[23] Renato Tintos, Darrell Whitley, and Francisco Chicano. Partition crossover for pseudo-boolean optimization. In *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*, FOGA '15, pages 137–149, New York, NY, USA, 2015. ACM.

[24] Sébastien Verel, Fabio Daolio, Gabriela Ochoa, and Marco Tomassini. Local Optima Networks with Escape Edges. In *International Conference on Artificial Evolution (EA-2011)*, pages 10 – 23, Angers, France, October 2011.

[25] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world'networks. *Nature*, 393:440 EP –, 06 1998.