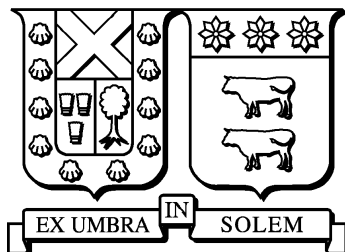


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE INFORMÁTICA

SANTIAGO – CHILE



“ANÁLISIS DE ESPACIOS DE PARÁMETROS DE
ALGORITMOS METAHEURÍSTICOS USANDO
REDES TRAYECTORIALES”

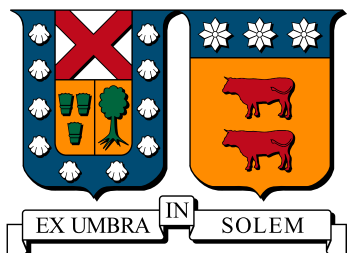
MARÍA INÉS RIVEROS C.

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INFORMÁTICO

PROFESOR GUÍA: NICOLÁS ROJAS M.

MARZO 2023

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO – CHILE



**“ANÁLISIS DE ESPACIOS DE PARÁMETROS DE
ALGORITMOS METAHEURÍSTICOS USANDO
REDES TRAYECTORIALES”**

MARÍA INÉS RIVEROS C.

**MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INFORMÁTICO**

PROFESOR GUÍA: NICOLÁS ROJAS M.

PROFESOR CORREFERENTE: ELIZABETH MONTERO U.

MARZO 2023

MATERIAL DE REFERENCIA, SU USO NO INVOLUCRA RESPONSABILIDAD DEL AUTOR O DE LA INSTITUCIÓN

Agradecimientos

Agradezco a todo aquel que ha sido un aporte a mi vida.

Resumen

Si bien el diseño de metaheurísticas ha demostrado ser una herramienta útil en la optimización, para alcanzar el mejor rendimiento de estos algoritmos es importante considerar los valores de sus parámetros. Por otro lado, las redes de búsqueda trayectorial son utilizadas para analizar, visualizar y contrastar el comportamiento de metaheurísticas. En este trabajo se propone realizar un análisis basado en dicha trayectoria para estudiar el comportamiento del algoritmo de sintonización Evoca considerando como algoritmo objetivo un algoritmo genético que resuelve el problema de los NK landscapes, variando el espacio de búsqueda de parámetros analizado. Para finalmente obtener las visualizaciones de las redes trayectoriales junto a las métricas asociadas a cada visualización, analizando los resultados obtenidos.

Palabras clave: Sintonización de parámetros, redes trayectoriales, espacio de búsqueda, Metaheurísticas

Índice de Contenidos

Índice de Contenidos	v
Lista de Tablas	viii
Lista de Figuras	ix
Introducción	1
1. Definición del Problema	4
1.1. Sintonización de parámetros	4
1.2. Redes de búsqueda trayectorial	7
1.2.1. Objetivos	8
2. Estado del Arte	10
2.1. Algoritmos de sintonización	11
2.1.1. ParamILS	12
2.1.2. Evoca	14
2.1.3. IRace	17
2.1.4. SMAC	19
2.2. Redes de Búsqueda Trayectorial	21
2.2.1. STNs fusionadas	23

2.2.2.	Aplicaciones de STN en la literatura	24
3.	Implementación	30
3.1.	Metodología de la implementación de la solución	31
3.1.1.	Evoca y STNs	31
3.1.2.	Obtención de redes trayectoriales	32
4.	Experimentos	35
4.1.	Algoritmo Objetivo	36
4.1.1.	NK landscapes	36
4.1.2.	Algoritmo Genético para NK landscapes	38
4.1.3.	Instancias de pruebas	39
4.1.4.	Parámetros	39
4.2.	Configuración de los experimentos	39
4.3.	Entorno de experimentación	40
5.	Resultados	41
5.1.	Métricas obtenidas para cada categoría	41
5.1.1.	Métricas obtenidas empleando precisión inicial	42
5.1.2.	Métricas obtenidas al disminuir la locación	44
5.2.	Métricas grupales	49
5.2.1.	Variación de la calidad de las configuraciones visitadas	49
5.2.2.	Nodos compartidos en cada ejecución	52
5.3.	Visualización de redes trayectoriales	54
	Conclusiones	72
5.3.1.	Trabajo Futuro	73

Índice de tablas

2.1. Resultados experimentales de configuraciones CPLEX[4]	14
4.1. Valores de N y K para los conjuntos de instancias de NK-landscapes	39
4.2. Parámetros del algoritmo genético para NK-landscapes	39
5.1. Descripción métricas individuales	42
5.2. Métricas de redes trayectoriales población 10- tamaño locación 0.01 x 0.01 x 1 . .	43
5.3. Métricas de redes trayectoriales población 20- tamaño locación 0.01 x 0.01 x 1 . .	44
5.4. Métricas de redes trayectoriales población 50- tamaño locación 0.01 x 0.01 x 1 . .	45
5.5. Métricas de redes trayectoriales población 10- tamaño locación 0.2 x 0.05 x 6 . . .	46
5.6. Métricas de redes trayectoriales población 20- tamaño locación 0.2 x 0.05 x 6 . . .	47
5.7. Métricas de redes trayectoriales población 50- tamaño locación 0.2 x 0.05 x 6 . . .	48
5.8. Nodos compartidos - tamaño locación 0.2 x 0.05 x 6	53

Índice de figuras

1.1. Proceso de sintonización	6
2.1. Esquema de algoritmo Irace. [8]	17
2.2. Ejemplo ilustrativo de una STN	23
2.3. Ejemplo ilustrativo de una STN fusionada [5].	24
2.4. Visualización de los modelos STN de genotipo, genotipo-fenotipo y fenotipo utilizando un diseño gráfico dirigido por fuerza [16]	26
2.5. STN aplicada a MOEA/D[17]	28
2.6. STN de auto-MOEA/D y la estrategia de no-restart en dos instancias de MOP . . .	28
3.1. Diagrama de propuesta de solución	31
5.1. Cantidad de arcos respecto a la cantidad de nodos	49
5.2. Cantidad de empeoramientos de una solución respecto a su calidad	50
5.3. Cantidad de mejoras de una solución respecto a su calidad	51
5.4. Cantidad de no variaciones en la solución respecto a su calidad	52
5.5. Categorías 2-20 fusionadas	52

5.7. Redes trayectoriales de categoría 2_20 - tamaño locación 0.01x0.01x1	55
5.8. Redes trayectoriales de categoría 2_20 - tamaño locación 0.2x0.05x6	56
5.9. Redes trayectoriales de categoría 2_38 - tamaño locación 0.01x0.01x1	56
5.10. Redes trayectoriales de categoría 2_38 - tamaño locación 0.2x0.05x6	57
5.11. Redes trayectoriales de categoría 2_52 - tamaño locación 0.01x0.01x1	57
5.12. Redes trayectoriales de categoría 2_52 - tamaño locación 0.2x0.05x6	58
5.13. Redes trayectoriales de categoría 3_20 - tamaño locación 0.01x0.01x1	58
5.14. Redes trayectoriales de categoría 3_20 - tamaño locación 0.2x0.05x6	59
5.15. Redes trayectoriales de categoría 3_34 - tamaño locación 0.01x0.01x1	59
5.16. Redes trayectoriales de categoría 3_34 - tamaño locación 0.2x0.05x6	60
5.17. Redes trayectoriales de categoría 3_48 - tamaño locación 0.01x0.01x1	61
5.18. Redes trayectoriales de categoría 3_48 - tamaño locación 0.2x0.05x6	61
5.19. Redes trayectoriales de categoría 4_20 - tamaño locación 0.01x0.01x1	62
5.20. Redes trayectoriales de categoría 4_20 - tamaño locación 0.2x0.05x6	62
5.21. Redes trayectoriales de categoría 4_30 - tamaño locación 0.01x0.01x1	63
5.22. Redes trayectoriales de categoría 4_30 - tamaño locación 0.2x0.05x6	63
5.23. Redes trayectoriales de categoría 4_40 - tamaño locación 0.01x0.01x1	64
5.24. Redes trayectoriales de categoría 4_40 - tamaño locación 0.2x0.05x6	64
5.25. Redes trayectoriales de categoría 5_20 - tamaño locación 0.01x0.01x1	65
5.26. Redes trayectoriales de categoría 5_20 - tamaño locación 0.2x0.05x6	65
5.27. Redes trayectoriales de categoría 5_28 - tamaño locación 0.01x0.01x1	66

5.28. Redes trayectoriales de categoría 5_28 - tamaño locación 0.2x0.05x6	66
5.29. Redes trayectoriales de categoría 5_38 - tamaño locación 0.01x0.01x1	67
5.30. Redes trayectoriales de categoría 5_38 - tamaño locación 0.2x0.05x6	67
5.31. Redes trayectoriales de categoría 6_20 - tamaño locación 0.01x0.01x1	68
5.32. Redes trayectoriales de categoría 6_20 - tamaño locación 0.2x0.05x6	68
5.33. Redes trayectoriales de categoría 6_26 - tamaño locación 0.01x0.01x1	69
5.34. Redes trayectoriales de categoría 6_26 - tamaño locación 0.2x0.05x6	69
5.35. Redes trayectoriales de categoría 6_32 - tamaño locación 0.01x0.01x1	70
5.36. Redes trayectoriales de categoría 6_32 - tamaño locación 0.2x0.05x6	70
5.37. Nodos compartidos por ejecución - tamaño locación 0.2 x 0.05 x 6	77

Introducción

Las metaheurísticas suelen ser una herramienta clave al tratar problemas de optimización combinatoria, pero para alcanzar el mejor rendimiento es importante considerar los valores de sus parámetros. Se busca obtener la mejor solución posible eficientemente, en espacial respecto al tiempo de ejecución, ya que debido a la alta dificultad de la tarea en cuestión tiende a ser una tarea que consume mucho tiempo. Al momento de plantear una metaheurística, el diseñador del algoritmo debe tener en cuenta 3 factores considerados de suma importancia, como:

1. Una **instancia** hace referencia a un problema específico que se le presenta a una metaheurística para que pueda encontrar una solución.
2. Las metaheurísticas a menudo utilizan números aleatorios como parte de su proceso de búsqueda de soluciones. Para ello se emplean las **semillas de números aleatorios** ya que permiten introducir aleatoriedad a las metaheurísticas pero a la vez tener la posibilidad de reproducir los resultados.
3. Los **parámetros** son valores que se utilizan para controlar el comportamiento de la metaheurística y por ende, su capacidad para encontrar soluciones óptimas.

Los parámetros son de suma importancia para el desempeño y comportamiento de una metaheurística, ya que al estar relacionados con el comportamiento que esta toma, inciden directamente en como se recorre el espacio de búsqueda, entendiendo como espacio de búsqueda al conjunto de todas las posibles soluciones a un problema. Existen varias categorías en las que se pueden basar los enfoques propuestos para sintonizar parámetros de algoritmos metaheurísticos:

1. **Enfoques basados en la experiencia o el conocimiento del experto**, estos enfoques se basan en la experiencia o el conocimiento del experto en el problema que se está abordando. El experto puede tener una intuición sobre qué parámetros son importantes y cómo ajustarlos para obtener buenos resultados.
2. **Enfoques basados en la búsqueda en el espacio de parámetros**, estos enfoques utilizan técnicas de búsqueda para explorar el espacio de parámetros y encontrar aquellos que dan lugar a resultados óptimos. Algunas técnicas comunes incluyen la búsqueda exhaustiva, la búsqueda aleatoria y la búsqueda basada en el gradiente.
3. **Enfoques basados en el aprendizaje automático**, estos enfoques utilizan algoritmos de aprendizaje automático para aprender de los resultados de los ajustes de parámetros anteriores y mejorar la selección de parámetros en el futuro.
4. **Enfoques basados en la optimización evolutiva**, estos enfoques utilizan algoritmos de optimización evolutiva, como el algoritmo genético, para encontrar el mejor conjunto de parámetros. Los individuos de la población representan diferentes conjuntos de parámetros y se seleccionan aquellos que dan lugar a resultados óptimos para formar la próxima generación.

Un área por explotar corresponde a analizar el espacio de búsqueda de un sintonizador de parámetros mediante el uso de redes trayectoriales. Cabe destacar que dentro del espacio de búsqueda se considera como solución a una configuración de parámetros que maximice el comportamiento de una metaheurística o algoritmo objetivo. Esta aproximación puede ser beneficiosa ya que:

1. Permite visualizar y comprender cómo se desplaza el sintonizador a través del espacio de parámetros y cómo evoluciona a lo largo del tiempo.
2. Ayuda a identificar patrones y tendencias en la trayectoria del sintonizador, lo que puede ser útil para entender cómo se relacionan los diferentes parámetros con el rendimiento del algoritmo.
3. Puede proporcionar una representación gráfica clara y concisa de la trayectoria del sintonizador, lo que puede ser útil para presentar y comunicar los resultados a terceros.

4. Puede permitir comparar de manera visual la trayectoria del sintonizador entre diferentes problemas o conjuntos de parámetros, lo que puede ser útil para determinar qué ajustes de parámetros son más efectivos en diferentes contextos.

Por lo que en esta memoria se propone una integración entre un algoritmo de sintonización de parámetros y redes trayectoriales, con la finalidad de analizar el comportamiento de un sintonizador y el efecto sobre el espacio de parámetros al modificar escenarios de sintonización.

Este documento está organizado de la siguiente manera. Inicialmente, en el Capítulo 1 se define el problema a abordar en la presente memoria y los objetivos generales y específicos. En el Capítulo 2 se contextualiza respecto a la sintonización de parámetros, describiendo algunos de los sintonizadores más conocidos y aplicaciones de estos. Además se realiza una definición de las redes de búsqueda trayectorial. Luego, en el Capítulo 3 se presenta la propuesta de solución, ahondando en la metodología aplicada. Seguido de una descripción de los experimentos realizados y la exposición de los resultados obtenidos en los Capítulos 4 y 5. Finalizando con las conclusiones del trabajo realizado.

Capítulo 1

Definición del Problema

Las metaheurísticas son estrategias inteligentes para diseñar o mejorar procedimientos heurísticos muy generales con un alto rendimiento. El término metaheurística apareció por primera vez en el artículo seminal sobre búsqueda tabú de Fred Glover en [6]. A partir de entonces, han surgido multitud de propuestas de pautas para diseñar buenos procedimientos para resolver ciertos problemas que, al ampliar su campo de aplicación, han adoptado la denominación de metaheurísticas.

1.1. Sintonización de parámetros

Considerando un problema de optimización, una metaheurística tiene por finalidad encontrar valores para las variables del problema que optimice la función objetivo y satisfaga todas las restricciones. Para completar esta ardua tarea se requiere de ciertos elementos, entre los cuales se encuentran los parámetros del algoritmo. Estos son valores que afectan directamente al desempeño que presenta el algoritmo, ya sea por la distribución y/o navegación del espacio de búsqueda, la naturaleza estocástica que presentan las metaheurísticas, la especificidad del problema que se abarca, la interrelación y/o variabilidad de los parámetros.

A partir de esto nace el problema de seteo de parámetros, que a su vez es un problema de optimización combinatoria en si mismo. Este problema tiene por finalidad la mejora del rendimiento de un algoritmo que resuelve un problema mediante la búsqueda de una combinación de valores

(configuración de parámetros) para los parámetros que optimice el comportamiento del algoritmo objetivo dada una métrica en específico. Este proceso es costoso en tiempo, debido a que requiere ejecutar el algoritmo objetivo constantemente para obtener información relevante del desempeño que muestra respecto a un conjunto de valores posibles de sus parámetros.

Para un algoritmo objetivo, el problema consiste en encontrar una configuración de parámetros que produzca el mejor rendimiento posible. El problema de seteo de parámetros para un algoritmo metaheurístico objetivo A puede definirse formalmente como:

Definición 1 *Dada una metaheurística A , el conjunto de parámetros de A , sean categóricos $P_c = \{c_1, \dots, c_k\}$ y/o numéricos $P_n = \{n_1, \dots, n_m\}$ y los dominios $D = \{d_1, \dots, d_{k+m}\}$ asociados a cada parámetro de A . Se tiene por meta encontrar la configuración de parámetros, es decir, valores para los parámetros P_c y P_n logrando que el algoritmo A tenga un desempeño adecuado **en un conjunto de instancias Π (dividido en instancias de training Π_{Tr} e instancias de testing Π_{Ts})***

Como se menciona en la Definición 1, existen parámetros de distintas clases, principalmente numéricos y categóricos. Cabe destacar que al conjunto de todas las configuraciones, se le denominará como espacio de configuraciones o parámetros. Los parámetros numéricos corresponden a valores reales o enteros como es el caso del tamaño de la población, las tasas de uso de operadores en algoritmos evolutivos. Además, esta clase de parámetros permiten definir medidas de distancia, es decir, se espera que los desempeños medidos del algoritmo muestren cierta similitud entre valores cercanos y diferencias entre valores lejanos. Mientras que en los parámetros categóricos no se puede cuantificar similitud entre los candidatos de manera natural ya que engloban los valores pertenecientes a un conjunto de elementos, como procedimientos o funciones que se pueden aplicar. Un ejemplo de parámetro categórico lo constituye el método de selección para usar en un algoritmo evolutivo. En este caso no es trivial definir similitud entre métodos como la ruleta, el k-torneo o la selección basada en ranking.

De manera simple, la sintonización de parámetros requiere de un escenario de configuraciones compuesto por un algoritmo, un conjunto de parámetros para el algoritmo y un conjunto de datos de entrada. El objetivo es encontrar los valores de los parámetros para los cuales el algoritmo obtenga el mejor rendimiento con los datos de entrada dados.

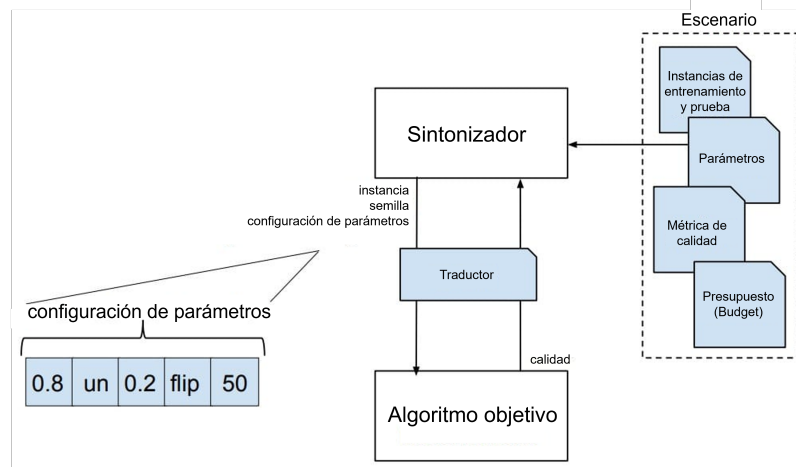


Figura 1.1: Proceso de sintonización

En la Figura 1.1 se muestra un diagrama básico del proceso de sintonización. Aquí se aprecia que cada proceso de sintonización requiere de un escenario con ciertos componentes para su ejecución. A continuación se describen dichos componentes:

- Instancias del problema, considerando la Definición 1 se tiene que el conjunto de instancias del problema se define como $\Pi = \Pi_{Tr} \cup \Pi_{Ts}$, donde Π_{Tr} y Π_{Ts} corresponden a todas las instancias pertenecientes a los conjuntos de entrenamiento y testeo respectivamente. Corresponden a los casos de prueba que se busca solucionar mediante el algoritmo objetivo, se utilizan en el proceso de sintonización con la finalidad de medir el comportamiento del algoritmo en las instancias de prueba al aplicar configuraciones de parámetros en específico.
- Métrica de calidad, para aplicar este criterio al momento de elegir un conjunto de valores o configuración de parámetros para ser asignada a los parámetros del algoritmo objetivo. La métrica más frecuente es la calidad de la solución encontrada en términos de la función objetivo que se está optimizando. Esta métrica de calidad es propia del problema particular que se resuelve, pero pueden considerarse otras como el número de evaluaciones de la función objetivo ó iteraciones al óptimo, entre otras.
- Se cuenta con un presupuesto (budget) para el proceso de sintonización. Dicho presupuesto puede estar relacionado con la cantidad de ejecuciones del algoritmo objetivo o el tiempo total del proceso. El presupuesto se fija tomando en consideración que la etapa de pruebas

para elegir una configuración de parámetros final no sea excesivamente costosa, considerando tiempo de ejecución.

- Los parámetros que se busca optimizar para mejorar el comportamiento del algoritmo objetivo son entregados al sintonizador. Para cada parámetro debe definirse al menos su respectivo dominio de valores para formar las configuraciones que se probarán durante el proceso de sintonización. En algunos casos es necesario definir un conjunto explícito de valores posibles a evaluar para cada parámetro.

Además, se cuenta con un **traductor** que sirve como puente o intermediario entre el sintonizador y la metaheurística objetivo (algoritmo objetivo) para evaluar las diversas configuraciones de parámetros. Dicho de otra forma, el traductor es quien comunica a la metaheurística cuáles son los valores que debe evaluar por parámetros (configuración de parámetros), junto con la instancia en que se deberá probar dicha configuración y la semilla aleatoria a utilizar.

Finalmente se tiene al sintonizador, que está a cargo de determinar las configuraciones de parámetros a evaluar, para determinar aquellas que presenten un mejor rendimiento para la metaheurística objetivo. Es importante tener en consideración que encontrar una adecuada configuración de parámetros es una ardua tarea. La tarea es especialmente compleja debido a interrelaciones entre los parámetros, variaciones que puedan presentar durante la ejecución, tiempo empleado para encontrar una combinación adecuada, entre otros. Debido a esto, el proceso resulta ser altamente costoso en computación por la extensa búsqueda [1]. Pese a esto, los algoritmos de sintonización de parámetros han demostrado mejoras significativas para el área en cuestión, debido a que su efecto se ve directamente reflejado en las soluciones que se encuentran para un problema.

1.2. Redes de búsqueda trayectorial

Una red de búsqueda trayectorial busca entender el proceso de búsqueda realizado por un algoritmo metaheurístico a través de la identificación y evaluación de zonas del espacio visitadas durante el proceso.

Una red de búsqueda trayectorial (Search Trajectory Network, STN) se puede definir como:

Definición 2 *Un grafo dirigido $STN = G(N, E)$, con un conjunto de nodos N , un conjunto de arcos E . Teniendo bajo consideración:*

- *Un nodo corresponde a una locación en la búsqueda trayectorial del proceso de búsqueda que se esta modelando. El conjunto de nodos es denotado como N .*
- *Un arco conecta dos locaciones consecutivas en la búsqueda trayectorial con la finalidad de armar un trazado del camino buscado por el algoritmo. El conjunto de arcos es denotado como E .*

Se ha demostrado que estos modelos de red son útiles para visualizar el comportamiento y analizar el balance entre exploración y explotación de los algoritmos estudiados [5].

Las redes trayectoriales han sido utilizadas para entender la dinámica de los algoritmos basados en población, donde se utilizan las conexiones para representar interacciones entre los individuos durante todas las generaciones con la finalidad de modelar la comunicación o influencia de individuos dentro de la población.

En la presente memoria se busca analizar el espacio de parámetros en base a la ruta que traza un sintonizador de parámetros mediante el uso de redes trayectoriales.

1.2.1. Objetivos

El objetivo principal de este trabajo de memoria es proponer, desarrollar y validar un acercamiento basado en redes trayectoriales para el análisis de espacios de búsqueda de parámetros usando algoritmos de sintonización .

Teniendo bajo consideración:

- Estudiar el estado del arte existente sobre sintonizadores de parámetros y redes trayectoriales de búsqueda.
- Realizar una integración entre algoritmos de sintonización de parámetros y redes trayectoriales.

- Analizar métricas sobre el espacio de parámetros descrito por los sintonizadores.
- Analizar el efecto sobre el espacio de parámetros, al modificar escenarios de sintonización.

Capítulo 2

Estado del Arte

Antes de entrar de lleno con la propuesta de este trabajo de memoria, es necesario realizar una revisión de la literatura relacionada. En este capítulo se presente una revisión de los métodos de sintonización de parámetros, los tipos de métodos de sintonización que existen y se describen algunos de los sintonizadores más conocidos. Posteriormente se presenta una definición de las redes de búsqueda trayectoriales que serán usadas para el análisis de espacios de los sintonizadores, indagando en sus características y avances en lo que va de investigación al respecto.

El problema de seteo de parámetros puede ser resuelto mediante dos maneras:

- El *control de parámetros* es una técnica que ajusta los valores de los parámetros del controlador en tiempo real mientras el sistema está en funcionamiento. Esta técnica se utiliza para adaptar los valores de los parámetros a las condiciones cambiantes del sistema y del entorno, de manera que el sistema pueda seguir funcionando de manera óptima. El *control de parámetros* es útil en situaciones en las que no se conocen exactamente los valores de los parámetros del sistema, ya que puede ajustar los valores de los parámetros sobre la marcha para adaptarse a las condiciones cambiantes.
- Los *sintonizadores de parámetros* son técnicas que calculan los valores óptimos para los parámetros del controlador antes de que el sistema esté en funcionamiento. Estas técnicas se utilizan para encontrar los valores óptimos para los parámetros del controlador, de manera que el sistema pueda funcionar de manera óptima cuando se activa. Los sintonizadores

de parámetros requieren un conocimiento previo del sistema y del modelo matemático del mismo, ya que utilizan técnicas matemáticas para determinar los valores óptimos para los parámetros del controlador. Los *sintonizadores de parámetros* son útiles en situaciones en las que los valores de los parámetros del sistema son estables y bien conocidos, ya que pueden proporcionar valores precisos para los parámetros del controlador.

Cabe destacar que dada la naturaleza de los parámetros a sintonizar y el enfoque dado a esta memoria, se decidió trabajar en esta ocasión con un sintonizador de parámetros.

En la literatura existen distintos enfoques para abordar el problema de sintonización de parámetros [1], como la *sintonización hecha a mano*. En la sintonización hecha a mano el principal enfoque es ajustar los parámetros mediante ensayo-error por parte del diseñador del algoritmo hasta obtener un rendimiento que el mismo diseñador considere adecuado. También existe la *sintonización por analogía*. En el caso de la sintonización por analogía se considera seguir las recomendaciones de expertos en el área de manera de re-utilizar dichas recomendaciones en el proceso de seteo de parámetros del nuevo algoritmo. Por otro lado, el *ajuste basado en diseño experimental* hace uso de técnicas de diseño experimental para determinar los valores que presentan un mejor desempeño en base a algún criterio estadístico. Las técnicas de

Junto con la *sintonización basada en búsqueda*, que como dice su nombre, recorre un espacio de búsqueda asociando los valores que los parámetros que pueden tomar obteniendo una combinación adecuada de valores, intentando garantizar un buen desempeño. Algunos ejemplos de esta clase de sintonizadores son los algoritmos ParamILS [4] y Evoca [2]. Finalmente, existen los *métodos híbridos*, que combinan los distintos tipos de sintonización de parámetros mencionados anteriormente.

2.1. Algoritmos de sintonización

A continuación se describen algunos algoritmos de sintonización de parámetros de la literatura, junto con los enfoques y sus principales resultados obtenidos.

2.1.1. ParamILS

Parameter iterated local search method es un método propuesto por Hutter et al. en [4]. El método busca dentro del espacio de configuraciones de parámetros para mejorar el comportamiento de un algoritmo. El método corresponde a un algoritmo basado en búsqueda local, ya que realiza una búsqueda local iterativa, empleando primera mejora.

Inicialmente se tiene un conjunto de parámetros, instancias del problema y un algoritmo a optimizar. Dada una configuración de parámetros inicial, se busca dentro de la vecindad de la configuración actual cambiando un único valor de parámetros de la solución candidata. El método incorpora como criterio de aceptación un procedimiento denominado *better* el cual posee dos variantes: BasicILS y FocusedILS explicadas más adelante. Y cuenta con una técnica de búsqueda local complementaria que es first improvement, si la configuración de parámetros generada es mejor o igual que la solución actual, es reemplazada como nueva solución actual. Este proceso se repite hasta encontrar el conjunto de valores de parámetros localmente óptimo. El sintonizador también incluye un método para escapar de óptimos locales. Para esto cuenta con una cantidad fija de perturbaciones para escapar de los óptimos locales, junto con una probabilidad de ocurrencia que reinicializa el proceso de búsqueda.

Como fue mencionado anteriormente, ParamILS cuenta con un criterio de comparación de nuevas soluciones. Este criterio está dado por un procedimiento llamado *better*. El enfoque más simple del procedimiento se conoce como BasicILS. En este caso se compara un par de configuraciones de parámetros usando una cantidad fija n de ejecuciones del algoritmo objetivo para decidir cuál de las dos configuraciones es mejor y retornarla con la mejor calidad asociada a las soluciones del sintonizador. Cada configuración de parámetros se ejecuta una cantidad fija de veces considerando un conjunto dado de instancias/semillas del conjunto de entrenamiento.

FocusedILS 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 considera variable el número de ejecuciones para estimar el costo estadístico asociado a la solución del sintonizador de parámetros. La idea es iniciar con pocas instancias/semillas y luego añadir más a las configuraciones de parámetros de mejor calidad

para obtener una estimación más confiable de su calidad.

FocusedILS compara dos configuraciones de parámetros en base a un número de ejecuciones con las mismas instancias y semillas. Luego se hace uso de lo que los autores [4] denominan *dominancia*. En este caso se dice que una configuración de parámetros domina a otra cuando se han realizado al menos tantas ejecuciones en la configuración dominante como en la configuración dominada, y el rendimiento de la configuración dominante (aplicada al algoritmo objetivo) sea al menos tan bueno como el de la configuración dominada.

ParamILS también dispone de una técnica que determina adaptativamente el presupuesto límite para cada ejecución del algoritmo conocida como capping. La idea tras esto se debe a que diferentes configuraciones de parámetros tienen tiempos de ejecución distintos, por lo que se necesita variar el tiempo límite dependiendo de cada configuración de parámetros a evaluar y el rendimiento que presenta al ser utilizada por el algoritmo objetivo durante el proceso de sintonización. Además, a menudo, la búsqueda de una configuración de parámetros que optimice el rendimiento pasa mucho tiempo evaluando algunas configuraciones de parámetros. Limitar el tiempo de ejecución por configuración ayuda a reducir el tiempo de ejecución total del proceso. Se determina un límite de evaluaciones de una configuración de parámetros basado en N ejecuciones, dicho de otra forma, un límite para tiempo de ejecución. Este procedimiento secuencialmente calcula un límite inferior cada vez que se ejecuta para una configuración de parámetros, entonces cada vez que hay una ejecución se suma el tiempo de ejecución y se divide por N . Si el límite inferior supera el límite dado por parámetro entonces se descartan las ejecuciones restantes para la configuración actual.

En [4] se realiza una comparativa entre los algoritmos ParamILS, BasicILS y FocusedILS. Para esto se utilizan cinco escenarios de configuraciones de CPLEX, específicamente: CPLEX-REGION S200, CPLEX-CONIC.SCH, CPLEX-CLS, y CPLEX-MIK.

CPLEX es un algoritmo masivamente parametrizado que se utiliza para resolver problemas de programación entera mixta (MIP). Fuera de sus 159 parámetros especificables por el usuario, se identificaron 81 parámetros que afectan la trayectoria de búsqueda de CPLEX.

Para cada escenario de configuración se enumeró el rendimiento de la prueba (tiempo de ejecución promedio penalizada sobre instancias de prueba) del algoritmo predeterminado, la media y su respectiva desviación estándar del rendimiento de la prueba en diez ejecuciones de BasicILS y

FocusedILS, el rendimiento de prueba de la ejecución de BasicILS y FocusedILS y el rendimiento de prueba (tiempo de ejecución promedio penalizado sobre instancias de prueba) del algoritmo predeterminado.

Escenario	Prueba de rendimiento				
	Default	$\bar{x} + \sigma$ 10 ejecuciones		Ejecución con mejor desempeño	
		BasicILS	FocusedILS	BasicILS	FocusedILS
CPLEX-REGIONS200	72	45 $\pm$ 24	11.4 $\pm$ 0.9	15	10.5
CPLEX-CONIC.SCH	5.37	2.27 $\pm$ 0.11	2.4 $\pm$ 0.29	2.14	2.35
CPLEX-CLS	712	443 $\pm$ 294	327 $\pm$ 860	80	23.4
CPLEX-MIK	64.8	20 $\pm$ 27	301 $\pm$ 948	1.72	1.19
CPLEX-QP	969	755 $\pm$ 214	827 $\pm$ 306	528	525

Tabla 2.1: Resultados experimentales de configuraciones CPLEX[4]

En la Tabla 2.1 se presentan el tiempo de ejecución promedio penalizado en segundos de CPU para cada una de las pruebas previamente indicadas. Se hace notar que el rendimiento de prueba de la ejecución de BasicILS y FocusedILS es mejor, en términos de rendimiento de entrenamiento, que ParamILS por defecto. Para el caso de la media y su respectiva desviación estándar, BasicILS presenta un mejor rendimiento en tres de los cinco escenarios, y Focused en dos de los cinco escenarios. Mientras que en cuanto al mejor rendimiento de prueba del algoritmo FocusedILS presenta soluciones mejores que BasicILS en cuatro de los cinco escenarios.

2.1.2. Evoca

Evolutionary Calibrator es una herramienta presentada por Montero et al. en [1] que tiene por fin ayudar en la elección de parámetros adecuados a diseñadores de algoritmos metaheurísticos con poca experiencia a obtener una configuración de parámetros de calidad sin tener un conocimiento profundo en la materia.

Evoca tiene por representación una cadena de caracteres donde cada elemento corresponde a un parámetro del algoritmo a sintonizar. El valor de cada parámetro se toma dentro del dominio correspondiente. Obteniendo una cadena de largo $m + k$, donde m y k corresponden a la cantidad de parámetros numéricos y categóricos respectivamente.

El procedimiento de inicialización de la población utiliza un diseño de hipercubo latino (latin

hypercube design) [10], donde cada dimensión hace referencia al dominio de cada parámetro dividida en p niveles iguales, eligiendo una única muestra en cada nivel. Se incluyen así, valores relevantes de cada parámetro de forma independiente. Para aquellos parámetros categóricos, en la mayoría de los casos, es posible incluir todos los valores debido a que tienen un dominio discreto, mientras que en los numéricos el diseñador del experimento de sintonización debe definir la precisión de los valores a considerar.

Una vez se tiene la población inicial de configuraciones, el algoritmo utiliza los operadores de cruzamiento y mutación para generar la descendencia de la población, *Wheel-crossover* y un *procedimiento de hill-climbing first improvement* respectivamente. El operador de cruzamiento se aplica en cada iteración y está enfocado en que el individuo generado herede buenos valores de parámetros desde toda la población. Cada valor del nuevo individuo se elige aleatoriamente desde todos los individuos de la población. Por otro lado, el procedimiento de mutación toma una copia del hijo creado por cruzamiento y trata de mejorarlo, si lo logra, reemplaza al segundo peor individuo de la población. En cada iteración a lo más son reemplazados dos individuos, uno por el operador de cruzamiento y otro por el de mutación. Además, en el caso de parámetros numéricos, se muta un valor al azar, por lo que se pueden generar valores inicialmente no incluidos en el conjunto de valores posibles de la población inicial. Cabe destacar que Evoca evalúa la calidad de una configuración utilizando R semillas, donde se recomienda usar un valor mayor que uno para tener información más confiable sobre el comportamiento del algoritmo.

Este algoritmo fue probado sintonizando un algoritmo genético que resuelve el problema de los NK landscapes [9].

En este caso fue capaz de obtener una buena solución, es decir, un conjunto de valores de buena calidad, junto con información útil en el proceso de diseño. Alcanzando un rendimiento similar a ParamILS para algunas instancias en las pruebas, pero superior al rendimiento del sintonizador REVAC [13].

En [11] se utiliza el sintonizador Evoca para estudiar medidas de robustez para tratar con la configuración de los parámetros de los algoritmos estocásticos, centrándose en un algoritmo evolutivo multiobjetivo (MOEA) basado en descomposición. Teniendo por objetivo verificar qué medidas

son las más útiles en tres escenarios caracterizados según la geometría del frente de Pareto correspondiente (lineal, convexo, cóncavo). Además, se busca identificar las configuraciones más robustas al escalar/aumentar el número de funciones objetivo en un problema de optimización multi-objetivo. Se utilizaron tres instancias de prueba donde la cantidad de objetivos y el número de variables de decisión es 3, 5, 7 y 7, 9, 11 respectivamente. Para dichas pruebas se combinaron los enfoques descritos a continuación para obtener un análisis adecuado de los resultados:

- **Media:** es una de las medidas más utilizadas en la literatura. Con esta medida, se busca la configuración que funciona mejor en la media de los casos. Sin embargo, esta medida no es adecuada cuando los resultados contienen valores atípicos ya que un valor atípico tiene un gran impacto en el propio valor medio.
- **Mediana:** elimina los valores atípicos. Sin embargo, esto significaría que no estamos interesados en todos los casos sino, en cambio, en la mayoría de ellos. De este modo, usar esta medida significaría que estamos interesados en la configuración que funciona bien para la mayoría de los escenarios pero podría fallar completamente en los demás.
- **Peor caso:** en este caso, la aptitud estaría representada por el peor resultado en el escenario dado. Esto sería optimizar en el peor de los casos y así garantizar que la configuración funcionaría al menos con esa calidad.

Se enfatizó en el “peor de los casos” ya que se apuntó a una configuración de parámetros que funcione en todos los tipos de problemas. Por lo que se utilizó como enfoques: *media-media*, *mediana-mediana*, *mejor-peor*, *media-peor*, *mediana-peor* y *peor-peor* [11].

A partir de estos resultados, es posible observar la recomendación de Evoca en el mejor-peor enfoque, es capaz de superar el baseline del algoritmo en la mayoría de los casos cuando los problemas son unimodales. Algo similar ocurre con la mediana. Sin embargo, para los problemas multimodales modales se puede observar un deterioro de la calidad de los resultados. Esto sugiere que tales problemas deberían estar en un escenario diferente y tienen su propia configuración para encontrar buenas soluciones.

Además, se hace notar que las configuraciones encontradas funcionan bien al aumentar el número de objetivos. Además, se observa que las diferentes medidas de robustez superan el baseline del

algoritmo en la mayoría de los problemas (excepto para el problemas multimodales). Esto sugiere que las medidas alternativas que tratan la incertidumbre pueden tener un impacto positivo en la búsqueda para configuraciones para un conjunto de escenarios.

2.1.3. IRace

Irace es un método que incluye al algoritmo de carrera *F-race* junto con varias extensiones y mejoras en el artículo [8]. *F-race* se puede definir como un proceso iterativo que en cada paso evalúa un conjunto de elementos, que en este contexto son configuraciones de parámetros. Una carrera corresponde a la comparación, acorde al desempeño, entre las configuraciones. A medida que una solución presenta un peor desempeño (demostrable) en comparación al resto, es descartada. De esta forma, permite enfocarse en aquellas configuraciones prometedoras, y evaluarlas en más instancias para obtener una mejor estimación de su comportamiento. Este proceso continua hasta alcanzar un mínimo de configuraciones sobrevivientes, un máximo de instancias utilizadas o un presupuesto computacional predefinido.

Iterated racing es un método para configuración automática que consiste en tres pasos. Primero se generan nuevas configuraciones de acuerdo a una distribución particular, para luego seleccionar las configuraciones élite de los recién muestreados por medio de carreras. Con esto se actualiza la distribución para sesgar el muestreo hacia las mejores configuraciones. Este proceso se repite hasta que se cumpla un criterio de término.

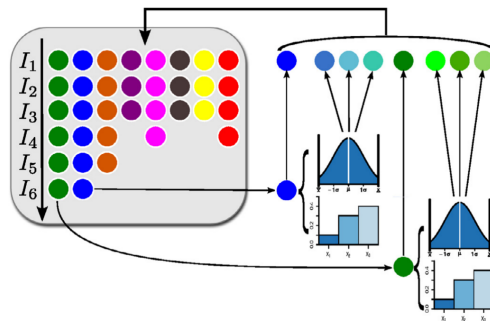


Figura 2.1: Esquema de algoritmo Irace. [8]

En la Figura 2.1 se aprecia el proceso simplificado de Irace, considerando seis instancias de pruebas ($I_1, \dots, I_6$). Cabe destacar que el conjunto inicial de configuraciones de Irace se genera

muestreando uniformemente el espacio de valores (representados en la imagen con círculos). Una vez se da inicio a la carrera, cada configuración se evalúa con la métrica de calidad. Las configuraciones son ejecutadas hasta que una cantidad fija de instancias han sido usadas. Luego, se pasa a una prueba estadística de los resultados obtenidos. Si hay evidencia suficiente para determinar que el desempeño de una configuración es deficiente en comparación al resto, será removida de la carrera (esto se visualiza al comparar la cantidad de elementos por iteración en la Figura 2.1, ya que se va reduciendo la cantidad de círculos o candidatos de la competencia/carrera), mientras que las configuraciones sobrevivientes avanzan a la siguiente iteración, procediendo con la actualización de las distribuciones de cada parámetro. Esto último se visualiza al llegar a la iteración 6 de la Figura 2.1, donde se generan nuevos candidatos a partir de las mejores soluciones para continuar con el proceso descrito.

Irace define la distancia entre dos configuraciones como la máxima diferencia entre cada ajuste de parámetros. Junto con incorporar *soft-restart* para evitar una convergencia prematura debido a los casos donde la distancia entre dos configuraciones candidatas es cero, llevando a un reinicio parcial de las distribuciones asociadas a las configuraciones elitistas que producen las soluciones candidatas con distancia cero.

Dentro de las limitaciones de Irace se tiene que el escenario para el que el algoritmo fue diseñado no tiene por objetivo la reducción de tiempo, a diferencia de otros sintonizadores, como es el caso de ParamILS o SMAC. Así Irace no alcanza el rendimiento que otros enfoques, ya que tiende a pasar demasiado tiempo evaluando configuraciones deficientes. Para esto, en [7] se propone una mejora a la eficiencia del algoritmo con el objetivo de la minimización del tiempo de ejecución mediante la integración de un mecanismo de término adaptativo (capping) inspirado en el utilizado por ParamILS.

Los escenarios utilizados para medir el rendimiento de $Irace_{cap}$ en [7] fueron tomados de estudios experimentales previos de otros sintonizadores de parámetros, como ParamILS y SMAC. Estos escenarios utilizan CPLEX - Regions 100, CPLEX - Regiones200, CPLEX - Corlat, Lingeling y Spear como algoritmos de destino, e involucran espacios de parámetros con 74, 137 y 26 parámetros, respectivamente.

Se compararon los resultados obtenidos por $Irace_{cap}$ con ParamILS y SMAC. Teniendo bajo

consideración que se decidió no incluir características de las instancias en el proceso de configuración y solo utilizar espacios de configuraciones de parámetros totalmente discretizadas; con la finalidad de aislar lo más posible el impacto del capping y para examinar si se volvería competitivo con otros configuradores que ya usaban esta técnica [7].

Teniendo en cuenta las características o espacios de parámetros no discretos introducirían factores adicionales que probablemente afecten el rendimiento más allá del impacto de la limitación.

Tras ejecutar cinco escenarios, se llega a que $Irace_{cap}$ obtiene la menor media en tres de los cinco escenarios, mientras que SMAC obtiene la media más baja en el escenario de Spear. Sobre uno de los tres escenarios de CPLEX, $Irace_{cap}$ obtiene el valor medio de rendimiento más bajo, aunque su rendimiento no es estadísticamente diferente al de ParamILS [7]. Demostrando así que $Irace_{cap}$ alcanza niveles de rendimiento competitivos con los de configuradores de última generación que han sido diseñados para funcionar bien en tiempo de ejecución en escenarios de minimización.

2.1.4. SMAC

El año 2011 Hutter et al. propusieron el algoritmo de sintonización de parámetros SMAC (Sequential Model-based Algorithm Configuration) en [3]. El algoritmo está inspirado en optimización basada en SMBO (Sequential Model-based Algorithm Configuration).

SMBO busca dar solución al problema de configuración de parámetros considerando al algoritmo objetivo como una caja negra. Esto produce ciertas limitaciones, entre las cuales se encuentra el uso de diseños experimentales de alto costo, confianza en modelos computacionalmente costosos y la suposición que todas las ejecuciones del algoritmo objetivo tienen los mismos costos de ejecución. Además, el uso de este tipo de métodos implica lidiar con limitaciones como únicamente admitir parámetros numéricos, solo optimizar el rendimiento del algoritmo objetivo para instancias individuales, entre otros.

SMAC busca eliminar algunas de las limitaciones de SMBO, al incluir un mecanismo de intensificación para múltiples configuraciones. En este caso se busca la reducción de la varianza al comparar el rendimiento entre dos configuraciones de parámetros usando instancias ejecutadas en común, específicamente un subconjunto seleccionado al azar de instancias y semillas. Además, se

incorpora un modelo para parámetros categóricos, el cual se basa en bosques aleatorios random forests [3], obteniendo predicciones precisas, y la cuantificación de la incertidumbre para una predicción. Además, SMAC es capaz de trabajar con espacios de búsqueda mixtos al usar un intercambio aleatorio de vecindario, incluyendo el conjunto de todas las configuraciones que difieren en el valor de exactamente un parámetro discreto, así como cuatro vecinos aleatorios para cada parámetro numérico.

Se emplea un método para predecir el tiempo de ejecución del algoritmo objetivo dada una combinación de valores de parámetros e instancias. Esto, dado que SMAC tiene como objetivo principal la reducción del tiempo de cálculo. De esta manera se controla dinámicamente el tiempo máximo asignado a cada ejecución descartando a priori configuraciones deficientes. Cabe destacar que se pueden emplear métricas distintas al tiempo de ejecución, ya que diferentes objetivos podrían funcionar de mejor manera para un algoritmo en específico.

Durante la inicialización de SMAC se realiza una única ejecución con la configuración de parámetros predeterminada del algoritmo objetivo (o una configuración aleatoria si ningún valor predeterminado está disponible) en una instancia seleccionada uniformemente al azar. Luego se intensifica tomando una lista de configuraciones prometedoras para compararlas a su vez con la configuración actual hasta que se alcance un presupuesto de tiempo dado. Dichas configuraciones son comparadas utilizando en todo momento instancias en que ambas fueron probadas. Es un procedimiento similar al empleado en FocusedILS, con la diferencia de que se basa en un subconjunto diferente seleccionado al azar de instancias y semillas, mientras que el procedimiento de ParamILS utiliza un orden fijo, el que puede volverse muy sensible.

El artículo [12] investiga el rendimiento de la evolución diferencial para ajustar hiperparámetros de algoritmos de aprendizaje supervisado para tareas de clasificación. Comparando el rendimiento de la evolución diferencial (DE) con la configuración de algoritmos basada en modelos secuenciales (SMAC). Concluyendo que DE supera a SMAC para la mayoría de conjuntos de datos cuando se ajusta un algoritmo de aprendizaje automático determinado. Por otro lado, en conjuntos de datos pequeños DE supera a SMAC en un 19 %. En un segundo experimento DE obtiene un 15 % más de victorias que SMAC. Mientras que en los presupuestos computacionales más ajustados SMAC presenta un mejor comportamiento que DE.

2.2. Redes de Búsqueda Trayectorial

Las redes de búsqueda trayectorial (Search Trajectory Network, STN) son utilizadas para analizar, visualizar y contrastar el comportamiento de los procesos de búsqueda de diferentes tipos de metaheurísticas [5]. Las redes se definen a partir de grafos compuesto por nodos y arcos que, en la mayoría de los casos pueden ser visualizados en dos o tres dimensiones.

La idea de las redes trayectoriales es recopilar información del proceso a medida que una metaheurística resuelve un problema de optimización. Para ello, se requiere de una solución representativa por cada iteración del algoritmo, generando con éstas, una trayectoria de búsqueda para la metaheurística en cuestión. Cada una de estas soluciones será entendida como un nodo del grafo que permitirá señalar la ubicación del proceso en diferentes pasos. Además se considera una partición predefinida del espacio de búsqueda que garantice que cada solución corresponda a un único elemento y una única ubicación. A continuación se recuerda la definición 3 del capítulo 1:

Definición 3 *Un grafo dirigido $STN = G(N, E)$, con un conjunto de nodos N , un conjunto de arcos E . Teniendo bajo consideración:*

- *Un nodo corresponde a una locación en la búsqueda trayectorial del proceso de búsqueda que se esta modelando. El conjunto de nodos es denotado como N .*
- *Un arco conecta dos locaciones consecutivas en la búsqueda trayectorial con la finalidad de armar un trazado del camino buscado por el algoritmo. El conjunto de arcos es denotado como E .*

Mediante el uso de estas redes se pueden calcular propiedades estructurales [14], entre aquellas que más destacan son:

1. El **Número de nodos** (n_{total}) esta directamente relacionado con la cantidad de espacio de búsqueda que fue explorado.
2. **Número de nodos con la mejor función de evaluación encontrada** (n_{best}) indica las diferentes locaciones mejor evaluadas del espacio de búsqueda.

3. **Número de nodos que finalizan una trayectoria** (nend) indica qué tan seguro es terminar en locaciones no óptimas.
4. Los **Nodos compartidos** (nshared) sirve como un indicador para cuantificar qué tanto tienden a atraer ciertas áreas del espacio a las trayectorias de soluciones.
5. El grado de un nodo es el número de aristas conectadas. Para el caso de las STN, se trabaja con grafos dirigidos, por lo que se distinguen grados entrantes y salientes. Al ponderar los “bordes”, es habitual utilizar el grado ponderado de un nodo, también llamado **fuerza** (strength) que está basado en el número de aristas conectadas al nodo, pero ponderado por el peso de cada borde.
6. Cuando se habla del concepto **mejor fuerza**, se hace referencia al peso de la solución con mejor calidad. En el caso que exista más de una configuración que alcanza la mejor calidad encontrada, se suman los pesos de todas las configuraciones. La idea tras esta métrica es obtener la cantidad de caminos que llegan a la mejor configuración.

La posibilidad de visualizar la red permite apreciar características estructurales que pueden ser difíciles de inferir solo a partir de las métricas de la red. Además, los atributos más importantes de los nodos y las aristas se asignan a las propiedades visuales (como el tamaño y el color) de las formas y líneas. Por ejemplo, el área de un círculo se puede hacer proporcional al grado del nodo para resaltar los nodos altamente conectados.

En la Figura 2.2 se presenta un ejemplo de red trayectorial, considera 10 ejecuciones independientes de Evoca. Cabe destacar que los cuadrados amarillos indican las ubicaciones de inicio de las 10 carreras. Los nodos rojos son las mejores soluciones encontradas. Los arcos de color gris, azul y verde corresponden a arcos de mejora, igualdad o empeoramiento de la calidad de las soluciones.

Hay que tener bajo consideración que una ubicación es un subconjunto no vacío de soluciones resultante de una partición predefinida del espacio de búsqueda. En [5] se menciona una forma de dividir dicho espacio. Para esto, el espacio de búsqueda continuo se divide en hipercubos discretos de una longitud establecida, definiendo una ubicación como un hipercubo de soluciones. Para así conseguir que cada ubicación sea equivalente a estos hipercubos. Cabe destacar que para extraer

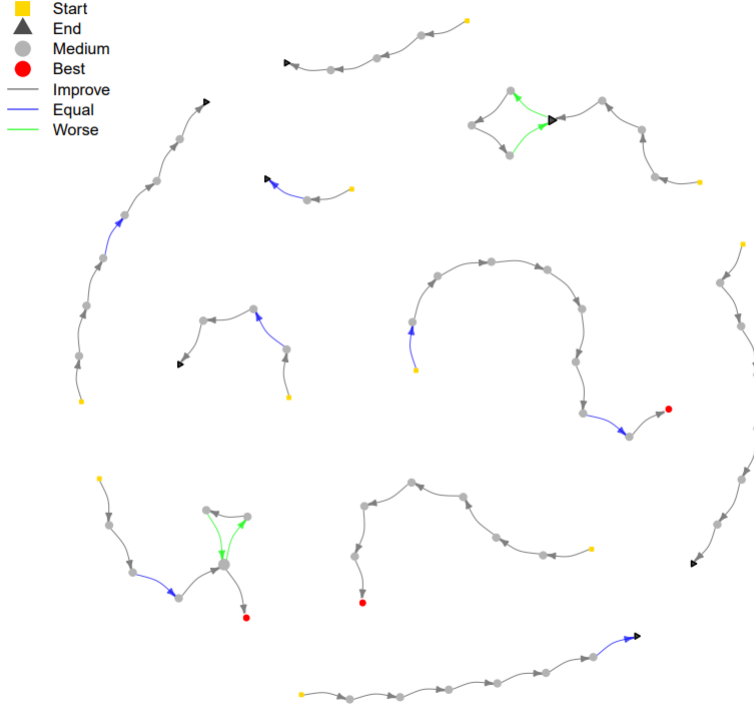


Figura 2.2: Ejemplo ilustrativo de una STN

información significativa del modelo STN, el tamaño de las particiones debe ser más grande para espacios de búsqueda más grandes.

2.2.1. STNs fusionadas

Las redes trayectoriales fusionadas o merged STNs corresponden, como su nombre lo indica a grafos de STN que contienen la unión de dos o más STNs. Valiéndose de la definición 3, formalmente se definen como:

Definición 4 Dado un par de grafos dirigidos $STN_A = G(N_A, E_A)$ y $STN_B = G(N_B, E_B)$ las STNs de los algoritmos A y B respectivamente (para una misma instancia), se construye STN_{merged} como la unión de los dos grafos. Dicho de otra forma, $STN_{merged} = G(N_A \cup N_B, E_A \cup E_B)$.

Cabe destacar que esta definición se puede extender a más grafos, siempre y cuando estos correspondan a una misma instancia para que el espacio de búsqueda sea efectivamente el mismo.

Además, los grafos combinados contienen los nodos y arcos que están en al menos uno de los grafos.

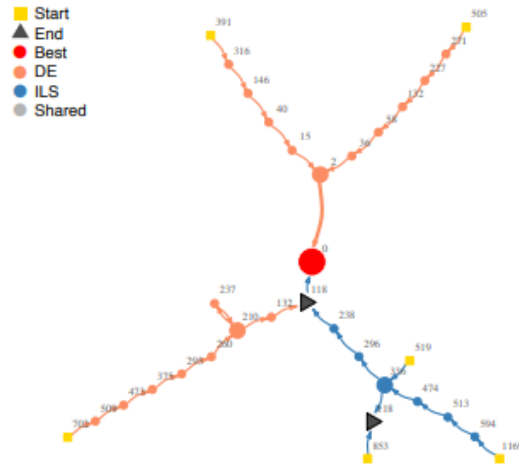


Figura 2.3: Ejemplo ilustrativo de una STN fusionada [5].

En la Figura 2.3 se presentan redes fusionadas, en específico la trayectoria de dos algoritmos (DE e ILS), considerando tres ejecuciones independientes cada uno. Cabe destacar que el tamaño de los nodos es proporcional a su grado entrante, las etiquetas de los nodos indican la diferencia en la evaluación del óptimo, los cuadrados amarillos indican las ubicaciones de inicio de las seis carreras. El nodo rojo corresponde a la solución con la mejor calidad encontrada. A dicha solución se llega mediante dos ejecuciones de DE y una de ILS en la red mostrada.

Cabe destacar que en el grafo de la figura 2.3 el tamaño de los nodos es proporcional a su calidad, las etiquetas de los nodos indican la diferencia en magnitud de la función de evaluación respecto de la solución óptima. Finalmente, el nodo rojo corresponde a la mejor solución encontrada, la cual es alcanzada por ILS y DE.

2.2.2. Aplicaciones de STN en la literatura

Durante noviembre del 2022 se presentó en [16] un estudio donde se visualizaban las trayectorias de búsqueda de un sistema de programación genética lineal basada en modelos de grafos. Esto, con la finalidad de analizar variaciones genotípicas como mutaciones en cambios fenotípicos

haciendo uso de redes trayectoriales. En este estudio se consideraron como nodos a genotipo-s/fenotipos y arcos a sus transiciones mutacionales. Además, se midieron cuantitativamente las características de los fenotipos, con la finalidad de conectar estas métricas con visualizaciones de trayectoria de búsqueda para así comprender mejor la progresión de la evolución y su comportamiento algorítmico, resultando que los más complejos fenotipos están sub-representados por menos genotipos y son más difíciles de descubrir por la evolución. Por otro lado, los fenotipos menos complejos son sobre-representados por genotipos, son más fáciles de encontrar y con frecuencia sirven como peldaños para la evolución.

En [16] se propusieron tres modelos STN:

1. **Genotipo STN**, donde las locaciones o nodos son genotipos únicos en el espacio de búsqueda, y los arcos representan las transiciones entre los genotipos.
2. **Genotipo-Fenotipo STN**, en esta instancia los nodos son fenotipos que agrupan componentes conectadas en el Genotipo STN que comparten el mismo fenotipo. Los arcos representan transiciones entre dichos nodos.
3. **Fenotipo STN**, donde los nodos son únicamente fenotipos en el espacio de búsqueda y los arcos representan transiciones consecutivas entre los fenotipos.

Para construir los modelos STN, se requiere de varias ejecuciones, donde las ubicaciones visitadas y sus transiciones son agregados en un solo modelo gráfico. Estos modelos retienen como nodos cada ubicación única y como arcos cada transición entre los lugares visitados, por lo que si algunas ubicaciones y transiciones puede aparecer varias veces durante el proceso de muestreo. Esto se registra midiendo la frecuencia de ocurrencia de cada nodo. Valores que son incluidos mediante etiquetas en las gráficas. Además su visualización tiene distintos tipos de nodos, los cuales se listan a continuación:

1. **Nodos neutrales**, aquellos cuyo nodo saliente adyacente tiene la misma aptitud.
2. **Nodos destino**, que tienen el fenotipo requerido.
3. **Nodos portales**, que enlazan con un nodo con aptitud mejorada. enlace al objetivo.

Cabe destacar que para el modelo Fenotipo STN se diferencian los nodos portales conectados directamente hacia un nodo destino. Además, se hace una distinción en la forma del nodo respecto a su posición en la trayectoria, como al inicio, final o al intermedio de la trayectoria. La Figura 2.4 es un claro ejemplo de lo descrito previamente.

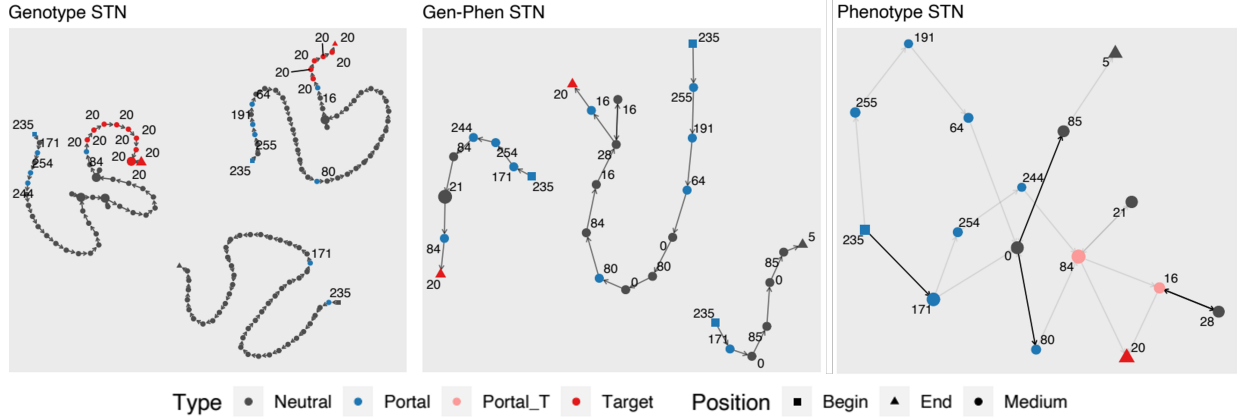


Figura 2.4: Visualización de los modelos STN de genotipo, genotipo-fenotipo y fenotipo utilizando un diseño gráfico dirigido por fuerza [16]

Una extensión de las STN fue presentada en [17] para modelar el comportamiento de los Algoritmos Evolutivos Multiobjetivo (MOEA), utilizando la idea de descomposición, donde un problema multiobjetivo se transforma en varios problemas de un solo objetivo.

En problemas de optimización Multiobjetivo (MOPs), una solución al problema es evaluada por múltiples funciones objetivo, que posiblemente están en conflicto. La característica clave de los MOP es que una sola solución rara vez es la solución óptima para todos los objetivos. Esto promueven la definición del concepto de “dominancia de Pareto”: Se dice que una solución x_j está dominada por x_i si x_i es mejor que x_j en un objetivo, y al menos equivalente en todos los demás. De lo contrario, x_j no está dominado por x_i . Cabe destacar que dos soluciones pueden ser mutuamente no dominadas (cada solución es mejor en al menos un objetivo diferente).

Esta extensión tiene por base hacer un seguimiento de un pequeño número de vectores de descomposición, relacionar una solución representativa con un vector y luego combinar las trayectorias vectoriales de cada vector en una sola STN multiobjetivo. Los pasos para asociar una solución a cada uno de los vectores son:

1. Inicialmente, se elige el número de vectores de descomposición n y se generan estos vectores

usando la técnica de descomposición descrita en [15].

2. En este punto se ordena un conjunto X de soluciones en función de su relación de dominancia. Donde se arma el subconjunto de soluciones X_{ND} (perteneciente a X), que contiene aquellas soluciones no dominadas en relación al resto (en base a Pareto). Estas soluciones son consideradas como el conjunto de soluciones de rango 0. Luego se repite este procedimiento en el conjunto $X - X_{ND}$, clasificando aquellas nuevas soluciones no dominadas como de rango 1 y así sucesivamente hasta clasificar todas las soluciones pertenecientes a X u obtener n clasificaciones.
3. En cada iteración, se selecciona una solución representativa para cada vector del conjunto de soluciones candidatas, haciendo uso de la función de agregación escalar, donde se mide la calidad de una solución en particular para un problema en específico con la finalidad de asociar cada solución a un subproblema en particular. En el caso de empate, se elige la solución más nueva, descartando las soluciones ya asociadas a ese vector.

Una vez clasificadas las soluciones, se asignan las ubicaciones de cada solución representativa. Obteniendo una STN fusionada que esta compuesta por n STNs de cada vector descompuesto, como es el caso de la Figura 2.5, donde se usan cuadrados amarillos para identificar el inicio de las trayectorias y triángulos negros para el final de las trayectorias. Los nodos rojos indican las mejores soluciones óptimas de Pareto y los nodos grises indican aquellas locaciones compartidas o visitadas por más de un vector. Los colores naranja, verde, morado, rosa y azul hacen referencia a las trayectorias de los vectores $V1$, $V2$, $V3$, $V4$ y $V5$ respectivamente

Otra aplicación de las redes trayectoriales es el análisis por componentes de algoritmos multiobjetivo diseñados automáticamente en problemas restringidos [18]. Considerando que el rendimiento de los algoritmos multiobjetivo varía según los problemas, dificulta el desarrollo de nuevos algoritmos o la aplicación de los existentes a nuevos problemas.

El artículo [18] busca analizar una instancia de Algoritmo Evolutivo Multiobjetivo basado en Descomposición (MOEA/D) de un punto de vista de los componentes en MOP con restricciones complicadas. El MOEA/D es un algoritmo que puede modificar su comportamiento considerando la limitaciones de los MOP que resuelve. Por lo que se concentran en analizar cómo los componentes MOEA/D afectan la dinámica del progreso de búsqueda, especialmente en MOP restringidos.

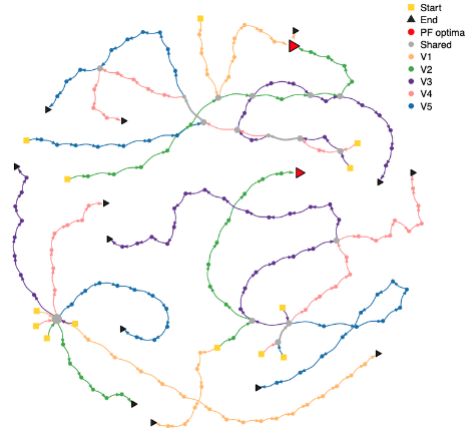


Figura 2.5: STN aplicada a MOEA/D[17]

En estudio se enfoca particularmente en cómo estas metaheurísticas exploran el espacio de decisión o búsqueda.

Haciendo uso de la Definición 4, y de la extensión mencionada recientemente [17] para combinar dos redes trayectoriales ó modelos STN de diferentes variantes de MOEA/D en una única STN fusionada y visualizar cómo distintas variantes exploran el espacio de búsqueda. A aquellas locaciones o nodos visitados por ambos algoritmos se les asignó un color diferente, como se puede apreciar en la Figura 2.6. Además, se tiene que las trayectorias de los STNs del auto-MOEA/D y cada variante se sobreponen, es decir, visitan regiones similares en el espacio. Esto indica que las regiones visitadas por ambos algoritmos tienden a ser más cercanas el uno del otro.

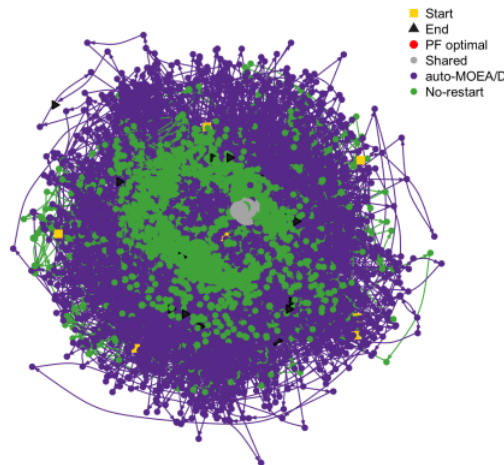


Figura 2.6: STN de auto-MOEA/D y la estrategia de no-restart en dos instancias de MOP

De este estudio se concluye que la mayoría de las métricas de STN pueden proporcionar información sobre la dinámica de búsqueda de los diferentes MOEA. Además, se encontró una alta correlación entre la métrica IGD y el número de nodos, aristas y nodos compartidos; y la varianza de la población. Lo que indica que las métricas del espacio de decisión pueden proporcionar información clara sobre la dinámica de diferentes MOEA en el espacio de decisión al mismo tiempo que se correlacionan con los incrementos en el rendimiento, reforzando la idea de que la caracterización de efectos de los componentes del algoritmo MOEA/D podrían ayudar en el desarrollo de MOEA aún más efectivos.

Capítulo 3

Implementación

Buscar mejoras dentro del campo de metaheurísticas es una tarea relevante debido a los altos costos asociados a mejorar su rendimiento considerando la dificultad de encontrar una configuración de parámetros que funcione adecuadamente para un contexto en particular. Una de las razones de estos costos se debe a la inmensa cantidad de ejecuciones requeridas para sintonizar un algoritmo. Considerando además, que se desea un desempeño sobre un conjunto de instancias (con características diferentes entre sí) y a la estocasticidad de los algoritmos.

Una herramienta útil para comprender y visualizar un espacio de parámetros son las STN, ya que permiten analizar el espacio de parámetros en base a la ruta que traza el sintonizador de parámetros Evolutionary Calibrator (EVOCA) mediante redes trayectoriales. Para esto, en este estudio, se sintoniza un Algoritmo Genético de la literatura propuesto para resolver problemas de NK-landscapes.

El desafío de esta memoria yace en realizar la integración entre un algoritmo de sintonización de parámetros y su visualización mediante el uso de redes trayectoriales. El proceso a seguir esta ejemplificado en la Figura 3.1.

Inicialmente se debe contar con un algoritmo a sintonizar (Algoritmo Objetivo) y la definición de un escenario de integración considerando: instancias, semillas y parámetros para resolver un problema en particular. Con la finalidad de maximizar su rendimiento se emplea un algoritmo sintonizador, el cual ejecutará muchas veces al algoritmo objetivo, variando los elementos que

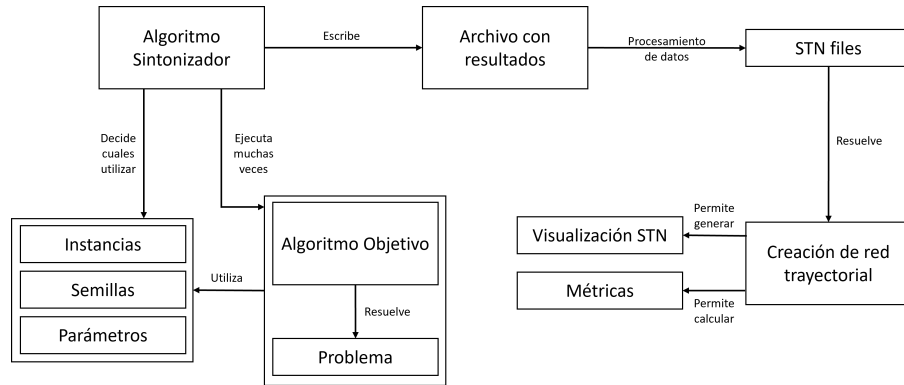


Figura 3.1: Diagrama de propuesta de solución

se requieren para el funcionamiento del mismo. A la vez, el algoritmo sintonizador recopila los resultados que obtiene de las ejecuciones del algoritmo objetivo y las decisiones que toma durante el proceso. Luego, estos resultados son llevados a un formato específico para la creación de las redes trayectoriales.

En el presente capítulo se presentan los fundamentos de la propuesta de solución y sus componentes, para luego explicar la metodología aplicada en la propuesta.

3.1. Metodología de la implementación de la solución

Para dar inicio a la implementación fue necesario estudiar el sintonizador a usar y la teoría de redes trayectoriales para luego comprender las implementaciones ya existentes. Esto involucra códigos de Evoca y scripts para obtener las STNs.

3.1.1. Evoca y STNs

La integración de Evoca con las STN requiere de un representante por cada iteración para conformar la trayectoria seguida por el algoritmo de forma gráfica. Aquí cada nodo es una configuración de parámetros y los arcos trazan el camino que realiza el sintonizador entre las diferentes configuraciones que visita. Al tratarse Evoca de un algoritmo poblacional, fue necesario instaurar

un criterio para determinar a dicho representante por cada iteración, considerando a la mejor configuración de cada iteración. Una vez se tiene al representante de la población de la iteración actual, se registra en un archivo el número correspondiente a la ejecución de la categoría, la calidad de la configuración de parámetros y la configuración en si.

3.1.2. Obtención de redes trayectoriales

Evoca es un algoritmo que trabaja con un conjunto de semillas aleatorias y un conjunto de instancias de donde se eligen las que serán utilizadas para la evaluación dentro del algoritmo. A esto se le suma la estocasticidad del algoritmo objetivo que para una misma instancia, el desempeño puede ser distinto para distintas semillas. Esto ocasiona que durante cada ejecución del algoritmo objetivo se emplee una semilla e instancia distinta, por lo que aunque la configuración de parámetros sea la misma, el resultado será diferente.

Para reparar los datos obtenidos al usar Evoca, asignando a cada configuración de parámetros un único valor de calidad asociado, se debe hacer un pre-procesamiento. Esto, debido a que las redes trayectoriales requieren que cada configuración de parámetros tenga asociado un único valor o aptitud. Así se considera utilizar un promedio ponderado entre todos los valores encontrados para una configuración. Luego, para construir las STNs y obtener las métricas correspondientes, se procedió a adecuar el archivo de entrada al formato esperado por los scripts encargados de generar las redes trayectoriales. El archivo 3.1 es un ejemplo de input para generar las redes trayectoriales.

1	Run	Fitness1	Solution1	Fitness2	Solution2
2	1	7.9	003002005	6.4	000002000
3	1	6.4	000002000	6.4	000002000
4	1	6.4	000002000	4.7	000002003
5	...	...	...	...	...

Archivo 3.1: Input para generar STNs

Aquí **Run** corresponde al número de la ejecución, **Fitness1** y **Solution1** son el valor o aptitud y la solución o configuración de la ubicación actual y **Fitness2** y **Solution2** de la ubicación nueva. Cabe destacar que cada paso de la trayectoria tiene una ubicación final e inicial, y el inicio del paso

siguiente es el mismo que el final del paso anterior.

Se deben usar múltiples ejecuciones/runs para construir un modelo STN. Es por eso que, cuando se encuentran configuraciones iguales en diferentes corridas, se debe decidir un único valor de calidad durante el pre-procesamiento.

Descripción de los scripts en R

Se tienen cuatro scripts para la generación de redes trayectoriales. Tanto para redes trayectoriales donde se visualiza más de un algoritmo o para redes trayectoriales donde se visualiza un único algoritmo. A continuación se listan y explican cada uno de los archivos:

1. Creación de la red, en este punto se crean los modelos STN empleando el input descrito recientemente en el archivo 3.1. En el caso de STNs combinadas, se puede visualizar hasta tres redes a la vez. Además, genera conjuntos de datos o modelos *.RData*, los cuales serán usados para graficar las redes trayectoriales en los pasos posteriores.
2. Gráfica de redes trayectoriales, hace uso de los modelos generados en el punto anterior para visualizar los modelos STN usando grafos dirigidos. Se emplea la Definición 3, donde un nodo corresponde a una locación en la búsqueda trayectorial del proceso de búsqueda que se está modelando, mientras que un arco es aquel que conecta dos locaciones consecutivas en la búsqueda trayectorial con la finalidad de armar un trazado del camino buscado por el algoritmo. Se entrega un único archivo *.pdf* que contiene el gráfico en cuestión.
3. Además, se cuenta con la opción de visualizar las redes trayectoriales con un diseño de árbol. Donde la principal diferencia es que dicha visualización es utilizada principalmente para modelar jerarquías o relaciones de orden entre elementos de datos, mientras que la gráfica de redes trayectoriales del punto anterior emplea grafos dirigidos, donde pueden utilizarse para representar cualquier tipo de relación entre elementos sin una estructura jerárquica específica. Cabe destacar que también hace uso de los modelos generados en el primer punto y entrega un único archivo *.pdf*.

4. Finalmente se calculan métricas asociadas a las STNs haciendo uso de los modelos generados en el primer punto. Se entrega un único archivo `.csv`. Dentro de las propiedades estructurales o métricas asociadas, se encuentra la cantidad total de nodos, el número de nodos con la mejor función de evaluación encontrada, nodos terminales, nodos compartidos, fuerza, mejor fuerza. Para mayor detalle revisar la Sección 2.2.

Para mayor detalle revisar el repositorio de GitHub (<https://github.com/gabro8a/STNs>)

En pocas palabras, en el presente capítulo se propuso un acercamiento para utilizar redes trayectoriales para analizar proceso de sintonizadores de parámetros. Se describe la metodología utilizada para la implementación de la solución. Se describió la identificación de aquellas configuraciones graficadas en las redes trayectoriales, también se detalló el método para obtener la calidad respectiva a cada solución encontrada por el sintonizador al momento de generar las STNs. Este paso es necesario debido al carácter estocástico en el espacio de búsqueda del sintonizador de parámetros.

En el siguiente capítulo se presentará en detalle los experimentos realizados con el algoritmo de sintonización, algoritmo objetivo, instancias y parámetros utilizados en el desarrollo de la memoria.

Capítulo 4

Experimentos

La optimización es un campo fundamental en la ciencia y la ingeniería, en el que se busca encontrar la mejor solución posible para un problema dado. Los algoritmos metaheurísticos son una clase de algoritmos de optimización que se basan en la exploración de un espacio de búsqueda para encontrar soluciones óptimas o subóptimas en un tiempo razonable.

Uno de los desafíos en la aplicación de algoritmos metaheurísticos es la selección adecuada de los parámetros del algoritmo. Los parámetros afectan significativamente la eficacia y eficiencia del algoritmo, y encontrar los valores óptimos de los parámetros puede ser un proceso complejo y costoso en términos de tiempo.

En este capítulo, se presenta una técnica innovadora para el análisis de espacios de parámetros de algoritmos metaheurísticos utilizando redes trayectoriales. Esta técnica es capaz de visualizar y analizar los comportamientos de los algoritmos en función de los valores de los parámetros, lo que puede ayudar a los investigadores y practicantes a comprender mejor la relación entre los parámetros del algoritmo y su desempeño.

Se presentan varios experimentos utilizando redes trayectoriales para analizar el espacio de parámetros de un algoritmo genético que resuelve NK landscapes de 15 categorías de instancias de problemas, con la finalidad de obtener las redes trayectoriales asociadas a la búsqueda realizada. En este capítulo se describe el algoritmo objetivo a sintonizar con sus categorías de instancias y sus parámetros.

4.1. Algoritmo Objetivo

Un sintonizador de parámetros es empleado para determinar la mejor combinación de parámetros a emplear en un algoritmo objetivo para resolver un problema. En este documento, para los experimentos se consideró como algoritmo objetivo un algoritmo genético que resuelve el problema de los NK landscapes. El algoritmo a utilizar es el propuesto por Pelikan en [9].

4.1.1. NK landscapes

El problema de los NK fitness landscapes consiste en un modelo matemático que se utiliza para representar la complejidad de un sistema en el que hay muchas variables interrelacionadas. Un “NK landscape” se utiliza para representar la complejidad de un sistema en el que hay muchas variables interrelacionadas. Se denomina así porque se suele usar para representar sistemas con N variables y K interacciones entre ellas.

El término “landscape” se utiliza para hacer referencia a la forma en que las variables del sistema interactúan y cómo estas interacciones afectan al sistema en su conjunto. En un “NK landscape”, cada punto del espacio representa una configuración posible del sistema, y la altura de ese punto representa el valor de la función de fitness asociada a esa configuración. La función de fitness mide el rendimiento o la adaptación del sistema en esa configuración específica.

Un “NK landscape” está definido por los siguientes componentes:

1. Número total de genes (N): Este es el número total de genes en una solución candidata. Cada gen puede tener varios estados posibles, dependiendo del problema en cuestión.
2. Número de genes vecinos considerados (K): Este es el número de genes vecinos que se consideran en la función de aptitud. Por ejemplo, si $k = 2$, entonces cada gen tiene dos vecinos directos (el gen anterior y el siguiente) y su beneficio se calcula en función de esos dos vecinos.
3. Función de beneficio (b): Esta es la función que se utiliza para calcular el beneficio de cada gen en función de sus vecinos. El beneficio específico de cada gen depende del problema en

cuestión.

4. Función de aptitud (f): Esta es la función que se utiliza para calcular el fitness de una solución candidata. El fitness se calcula sumando los beneficios de cada gen.
5. Espacio de búsqueda: Este es el conjunto de soluciones candidatas válidas. Por ejemplo, en un problema de “NK landscape” binario, el espacio de búsqueda estaría compuesto por todas las posibles combinaciones de ceros y unos.

En resumen, un “NK landscape” es un problema de optimización en el cual el fitness de una solución está dado por una función que tiene en cuenta el beneficio de cada gen en relación con un número k de genes vecinos. La complejidad y la estructura del problema depende del número total de genes (N), el número de vecinos considerados (K), la función de beneficio (b), la función de aptitud (f) y el espacio de búsqueda.

Cabe destacar que el variar los valores para N o K puede aumentar o disminuir la complejidad del problema, ya sea por la cantidad de genes o interacción entre estos.

1. Un aumento en el valor de N (el número total de genes en una solución) aumenta la complejidad del problema, ya que hay más genes que considerar en la función de aptitud. Esto puede hacer que sea más difícil encontrar una solución óptima.
2. Un aumento en el valor de K (el número de genes vecinos considerados en la función de aptitud) también aumenta la complejidad del problema, ya que hay más interacciones entre los genes que considerar. Esto puede aumentar la cantidad de interacciones entre los genes, lo que puede hacer que sea más difícil encontrar una solución óptima.
3. Una disminución en el valor de n disminuye la complejidad del problema, ya que hay menos genes que considerar en la función de aptitud. Esto puede hacer que sea más fácil encontrar una solución óptima.
4. Una disminución en el valor de k disminuye la complejidad del problema, ya que hay menos interacciones entre los genes que considerar. Esto puede reducir la cantidad de interacciones entre los genes, lo que puede hacer que sea más fácil encontrar una solución óptima.

El modelo se puede expresar matemáticamente de la siguiente manera: Dado un conjunto de N variables $x_1, x_2, \dots, x_N$ y K interacciones entre ellas, se define un “NK landscape” como una función $f: R^N \rightarrow R$ que asigna a cada configuración del sistema un valor de fitness.

4.1.2. Algoritmo Genético para NK landscapes

El algoritmo utilizado para dar una solución al problema de los NK landscapes es propuesto en [9] por Pelikan et al. El algoritmo1 corresponde al pseudo-código del algoritmo genético, que comienza generando una población de tamaño aleatorio y utiliza Binary Tournament Selection para elegir a los padres, donde se seleccionan dos individuos al azar de la población. Luego, se comparan los valores de aptitud de estos dos individuos (soluciones). El individuo con la aptitud más alta es el ganador del torneo y es seleccionado para la reproducción y la mutación. Luego son transformados utilizando un operador de cruzamiento acorde a una probabilidad c_r . En específico se utiliza cruzamiento uniforme, donde se toman los bits de ambos padres y se seleccionan al azar para crear los dos descendientes. Seguido a esto, se aplica un operador de mutación (bitflip) considerando una probabilidad m_r . Bitflip es una técnica en la que se elige un bit al azar en un individuo y se invierte su valor.

Algoritmo 1 Genetic Algorithm for NK Landscapes

```

1:  $P \leftarrow \text{GenerateInitialPopulation}(p_s)$ ;
2: while Maximum_Iterations are not reach do
3:    $\text{EvaluatePopulation}(P)$ ;
4:    $P_{\text{parents}} \leftarrow \text{BinaryTournamentSelection}(P)$ ;
5:   while  $P_{\text{new}}$  is not complete do
6:     Select two parents  $p_1$  and  $p_2$  from  $P_{\text{parents}}$ ;
7:      $\text{uniform\_crossover}(p_1, p_2, c_r)$ ;
8:      $\text{mutation}(p_1, m_r)$ ;
9:      $\text{mutation}(p_2, m_r)$ ;
10:     $P_{\text{new}} \leftarrow P_{\text{new}} \cup p_1$ ;
11:     $P_{\text{new}} \leftarrow P_{\text{new}} \cup p_2$ ;
12:   end while
13:    $P \leftarrow P_{\text{new}}$ ;
14: end while

```

4.1.3. Instancias de pruebas

Las instancias a utilizar son las mismas propuestas en [9]. Estas corresponden a conjuntos de instancias donde se varían los valores de N y K como indica la Tabla 4.1. Se cuenta con tres valores distintos de K para cada N .

Como fue mencionado recientemente, a mayor valor para N o K , la dificultad del problema aumenta. Ya sea por la cantidad de variables o interacciones entre ellas respectivamente.

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

Tabla 4.1: Valores de N y K para los conjuntos de instancias de NK-landscapes

4.1.4. Parámetros

La Tabla 4.2 presenta una descripción de los parámetros del algoritmo objetivo considerados durante el proceso de sintonización. La tasa de cruzamiento y mutación corresponden a valores reales que van del 0.00 al 1.00 y 0.00 al 0.25 respectivamente. Mientras que el tamaño de la población corresponde a un valor entero que va desde 3 a 30.

Nombre	Descripción	Tipo	Dominio	Observación
c_r	Tasa de cruzamiento	numérico (real)	[0.00,1.00]	Uniform Crossover
m_r	Tasa de mutación	numérico (real)	[0.00,0.25]	Bit-flip
p_s	Tamaño población	numérico (entero)	[3,30]	-

Tabla 4.2: Parámetros del algoritmo genético para NK-landscapes

4.2. Configuración de los experimentos

La minimización de la cantidad de ejecuciones es importante en la optimización de algoritmos porque el proceso de ejecución de un algoritmo puede ser muy costoso computacionalmente y consumir muchos recursos, como tiempo, memoria y energía. Por lo tanto, si se busca una solución

óptima en un problema dado, se desea minimizar la cantidad de veces que el algoritmo se ejecuta para encontrar dicha solución.

Además, la minimización de la cantidad de ejecuciones puede ser especialmente importante en problemas de optimización complejos que requieren una gran cantidad de iteraciones para encontrar una solución satisfactoria. Si se utilizan métricas que requieren una gran cantidad de ejecuciones, el proceso de optimización podría ser demasiado costoso computacionalmente para ser considerado como práctico.

Se busca encontrar la mejor configuración de parámetros que minimiza la cantidad de evaluaciones que le toma al algoritmo genético encontrar el óptimo global. Con la finalidad de analizar el espacio de parámetros en base a la ruta que traza el sintonizador de parámetros Evoca. Teniendo bajo consideración que el algoritmo genético objetivo cuenta con una cantidad máxima de evaluaciones fijada en 100000, junto con una cantidad máxima de ejecuciones por parte del sintonizador, la cual también fue fijada en 100000.

4.3. Entorno de experimentación

El algoritmo Evoca para redes STN, fue modificado utilizando el lenguaje C++ con el compilador g++ 6.5.0

El algoritmo genético para NK landscapes programado en C, fue compilado con g++ versión 6.5.0.

Los scripts para realizar la integración entre los algoritmos sintonizadores y las redes trayectoriales fueron programados en lenguaje Python version 3.9.7.

Los scripts para generar las redes trayectoriales y su visualización fueron ejecutados en el lenguaje R versión 4.2.1.

Las pruebas para generar las redes de óptimos locales se realizaron en un computador con procesador Intel(R) Core(TM) i5-1035G1 CPU @1.00GHz, 12GB de RAM y sistema operativo Windows 10.

Capítulo 5

Resultados

En este capítulo se presentan los resultados obtenidos junto al análisis de estos tras realizar los experimentos mencionados en el capítulo anterior. Esto, con la finalidad de comprender el comportamiento que tiene Evoca al momento de recorrer el espacio de búsqueda en distintos escenarios. Por lo que se dividirá en tres secciones. El primero corresponde a las métricas extraídas al momento de generar las redes trayectoriales considerando como precisión para la locación la dada por los parámetros inicialmente. Para luego analizar el comportamiento de las redes al modificar la precisión de la locación. Se realiza un análisis para las distintas métricas obtenidas al obtener las STN, de manera individual y grupal, donde la agrupación puede ser por categoría con distinta locación o en base al valor de N. Posteriormente, se realiza un análisis visual de una de las categorías tratadas.

5.1. Métricas obtenidas para cada categoría

Las métricas presentadas a continuación se obtuvieron mediante archivos .CSV al momento de generar las redes trayectoriales. Estas fueron generadas individualmente para cada instancia.

5.1.1. Métricas obtenidas empleando precisión inicial

Tanto m_r como c_r tienen una precisión de 0.01 y p_s es entero. Por lo tanto, una locación tiene por dimensiones (c_r) 0.01 x (m_r) 0.01 x (p_s) 1. El individuo representativo por iteración es la mejor configuración de la población, por ende se tendría un individuo posible por locación. Cuando existe una repetición del individuo en distintas ejecuciones de los procesos de sintonización se considera como medida de calidad el promedio de dichas calidades.

Cabe destacar que se trabajó con tres tamaños de población inicial para el algoritmo de sintonización Evoca: 10, 20 y 50. En las tablas 5.2, 5.3 y 5.4 se presentan los resultados obtenidos para dichos experimentos. Donde cada columna hace referencia a un valor en específico, como se detallan a continuación.

Métrica	Descripción
<i>Nodes</i>	Cantidad de nodos presentes en la red trayectorial, de manera similar
<i>Edges</i>	Cantidad de arcos presentes en la red trayectorial.
<i>Nbest</i>	Cantidad de locaciones con la mejor aptitud encontrada.
<i>Nend</i>	Cantidad de nodos que finalizan una trayectoria.
<i>Strength</i>	Grado de la mejor configuración ponderado por la cantidad de ejecuciones
<i>Plength</i>	Largo promedio de la ruta más corta hacia la mejor configuración.
<i>Npaths</i>	Cantidad de rutas hacia la mejor configuración.
<i>Components</i>	Cantidad de componentes conexas de la red.

Tabla 5.1: Descripción métricas individuales

En la Tabla 5.2 se observa que son pocas las configuraciones representativas encontradas, específicamente una para todos los casos, exceptuando EVOCA 2-20 donde se tienen tres configuraciones con la mejor aptitud. Esta cantidad es baja considerando que se realizaron diez ejecuciones independientes del proceso de sintonización completo para cada categoría de problemas. Respecto a las componentes conexas y nodos finales de una trayectoria, se evidencia que prácticamente no hay conexión entre las ejecuciones independientes. Esto, pues ambos indicadores presentan valores entre 9 y 10 para todos los casos. Nuevamente esto se hace notar debido a los valores de strength y npaths, indicando que usualmente solo una componente del grafo llega a la mejor solución de la categoría.

En la Tabla 5.3 se presentan los resultados de las métricas de redes trayectoriales con población igual a 20. Nuevamente se observan pocas configuraciones representativas encontradas. Esto, en

category	nodes	edges	nbest	nend	components	strength	plength	npaths
EVOCA 2-20	71	63	3	7	10	0.3	7.0	3
EVOCA 2-38	82	76	1	9	9	0.1	9.0	1
EVOCA 2-52	107	100	1	9	10	0.1	12.0	1
EVOCA 3-20	76	66	1	9	10	0.1	3.0	1
EVOCA 3-34	70	61	1	9	9	0.1	7.5	2
EVOCA 3-48	75	67	1	9	9	0.1	7.0	1
EVOCA 4-20	71	67	1	9	10	0.1	3.0	1
EVOCA 4-30	67	62	1	9	9	0.1	5.0	1
EVOCA 4-40	89	84	1	9	9	0.1	9.5	2
EVOCA 5-20	86	77	1	9	10	0.1	7.0	1
EVOCA 5-28	84	75	1	9	10	0.1	8.0	1
EVOCA 5-38	76	66	1	9	10	0.1	3.0	1
EVOCA 6-20	87	80	1	9	10	0.1	7.0	1
EVOCA 6-26	73	64	1	9	9	0.1	6.0	2
EVOCA 6-32	87	80	1	9	10	0.1	5.0	1

Tabla 5.2: Métricas de redes trayectoriales población 10- tamaño locación 0.01 x 0.01 x 1

conjunto con un escenario de desconexión entre las ejecuciones independientes. Esto, por la cantidad de componentes identificadas en cada caso. Además, el valor para el strength es bajo, dejando en claro que solo una componente en cada caso llega a la mejor solución. Esto es coherente con los valores de npaths.

En la Tabla 5.4 se presentan los resultados de las métricas de redes trayectoriales con población igual a 50. Donde se observan tantas componentes conexas como ejecuciones independientes (entre 9 y 10 componentes en 10 ejecuciones para cada instancia). Esto se mantiene coherente con los valores de nend. Además, el valor para el strength nuevamente se mantiene bajo y tiene igual valor en todas las instancias, lo que indica que solo una componente/trayectoria llega a la mejor solución. Cabe destacar que la mejor solución encontrada es única para todas las instancias, solo encuentra una única configuración con la mejor aptitud (nbest). Respecto a los valores de npath, en la mayoría de los casos, los caminos que dan a la configuración con mejor aptitud son 1, lo que indica nuevamente que una única componente de la red llega a esta configuración.

A partir de los resultados presentados, se observa que en los tres escenarios presentados se refleja un comportamiento similar. En los tres casos se consiguen pocas soluciones con la mejor aptitud encontrada, una alta cantidad de componentes conexas (es decir baja conectividad entre diferentes

category	nodes	edges	nbest	nend	components	strength	plength	npaths
EVOCA 2-20	86	78	1	9	9	0.1	7.0	1
EVOCA 2-38	78	68	1	9	10	0.1	7.0	1
EVOCA 2-52	72	63	1	9	10	0.1	6.0	1
EVOCA 3-20	75	65	1	9	10	0.1	6.0	1
EVOCA 3-34	70	60	1	9	10	0.1	10.0	1
EVOCA 3-48	77	68	1	9	10	0.1	6.0	1
EVOCA 4-20	69	60	1	9	10	0.1	5.0	1
EVOCA 4-30	67	57	1	9	10	0.1	8.0	1
EVOCA 4-40	81	74	1	9	9	0.1	10.0	2
EVOCA 5-20	56	46	1	8	10	0.1	5.0	1
EVOCA 5-28	80	71	1	9	10	0.1	7.0	1
EVOCA 5-38	81	72	1	9	10	0.1	7.0	1
EVOCA 6-20	78	68	1	9	10	0.1	8.0	1
EVOCA 6-26	87	78	1	9	10	0.1	9.0	1
EVOCA 6-32	72	64	1	9	10	0.1	7.0	1

Tabla 5.3: Métricas de redes trayectoriales población 20- tamaño locación 0.01 x 0.01 x 1

procesos de sintonización), bajos valores para strength y npath (coherentes con la cantidad de componentes independientes).

Finalmente, sobre el largo promedio de los caminos a las mejores configuraciones, se observa que la cantidad se mueve entre 5 a 10 pasos en todos los escenarios estudiados. Esto puede representar que el algoritmo está realizando trayectorias cortas y directas hacia las mejores configuraciones. No se observa una tendencia marcada sobre la dificultad entre los casos de prueba y los resultados obtenidos.

5.1.2. Métricas obtenidas al disminuir la locación

Con el objetivo de evidenciar una mayor conectividad entre las configuraciones y analizar el comportamiento que tendría el sintonizador al emplear una precisión diferente, se modificó el tamaño de la locación.

El tamaño de una locación se define como $(c_r) 0.2 \times (m_r) 0.05 \times (p_s) 6$. En este caso, se considera como representante de cada locación el punto medio de la región, y su calidad asociada considera la mejor calidad encontrada dentro de la locación respectiva. Cabe destacar que nuevamente son

category	nodes	edges	nbest	nend	components	strength	plength	npaths
EVOCA 2-20	73	63	1	9	10	0.1	3.0	1
EVOCA 2-38	69	60	1	9	9	0.1	3.0	1
EVOCA 2-52	71	62	1	8	9	0.1	5.0	1
EVOCA 3-20	71	61	2	8	10	0.2	5.0	2
EVOCA 3-34	73	63	1	9	10	0.1	6.0	1
EVOCA 3-48	74	65	1	9	9	0.1	5.0	2
EVOCA 4-20	72	62	1	9	10	0.1	9.0	1
EVOCA 4-30	67	57	1	9	10	0.1	5.0	1
EVOCA 4-40	74	64	1	9	10	0.1	7.0	1
EVOCA 5-20	77	67	2	8	10	0.2	7.5	2
EVOCA 5-28	81	71	1	9	10	0.1	8.0	1
EVOCA 5-38	81	71	1	9	10	0.1	4.0	1
EVOCA 6-20	65	56	1	9	10	0.1	5.0	1
EVOCA 6-26	81	72	1	9	10	0.1	7.0	1
EVOCA 6-32	76	67	1	9	10	0.1	10.0	1

Tabla 5.4: Métricas de redes trayectoriales población 50- tamaño locación 0.01 x 0.01 x 1

utilizadas las métricas descritas en la Tabla 5.1 para analizar los procesos. Cabe destacar que se trabajó con los tres tamaños de población inicial para el algoritmo de sintonización Evoca: 10, 20 y 50.

En la Tabla 5.5 se presentan los resultados de las métricas de redes trayectoriales con población igual a 10. En este caso la relación lineal que existía entre nodos y arcos se pierde debido la distribución espacial de los nodos, las interacciones entre estos y a la aproximación de la locación en sí.

Por otro lado, la cantidad de componentes individuales se redujo a valores entre 1 y 2, lo que indica una alta conectividad de los grafos correspondientes. Este fenómeno es coherente con el aumento de caminos hacia la mejor solución encontrada (npaths). Respecto al largo de los caminos también hay una reducción ya que el promedio sin aplicar la aproximación de locación es de 6.6, mientras que ahora es 6.2. Esto comparando ambos escenarios con población inicial de 10 configuraciones debido al aumento de componentes conexas.

En el caso de nbest hay que destacar que se pierde una configuración con la mejor aptitud, esto ocurre debido a que dos de las tres soluciones se encontraban en una misma locación (actual), por lo que al realizar la aproximación son representadas por un mismo nodo.

category	nodes	edges	nbest	nend	components	strength	plength	npaths
EVOCA 2-20	52	59	2	7	2	0.2	9.1	16
EVOCA 2-38	57	73	1	7	1	0.1	6.0	9
EVOCA 2-52	62	93	1	9	1	0.1	7.4	9
EVOCA 3-20	50	63	1	7	1	0.1	8.5	9
EVOCA 3-34	47	56	1	8	1	0.1	6.0	2
EVOCA 3-48	35	60	1	5	1	0.1	4.6	8
EVOCA 4-20	54	66	1	8	1	0.1	7.5	8
EVOCA 4-30	44	60	2	8	1	0.3	5.0	18
EVOCA 4-40	43	77	1	7	1	0.1	5.0	9
EVOCA 5-20	54	73	1	8	2	0.1	7.0	9
EVOCA 5-28	47	69	1	7	1	0.1	6.6	9
EVOCA 5-38	53	61	1	7	2	0.1	2.0	1
EVOCA 6-20	62	75	1	8	1	0.1	8.4	10
EVOCA 6-26	55	64	1	8	2	0.1	5.0	4
EVOCA 6-32	54	73	1	8	1	0.1	5.3	8

Tabla 5.5: Métricas de redes trayectoriales población 10- tamaño locación 0.2 x 0.05 x 6

En la Tabla 5.6 se presentan los resultados de las métricas de redes trayectoriales con población igual a 20. En este caso la relación lineal que existía entre nodos y arcos nuevamente se pierde por la misma razón que anteriormente.

Además el aumento de caminos hacia la mejor solución encontrada hace coherente la disminución de componentes conexas. Respecto al largo de los caminos también hay una reducción ya que el promedio sin aplicar la aproximación de locación es de 7.2, mientras que ahora es 6.1 (Comparando ambos escenarios con población inicial 20). Cabe destacar que la variación del largo de los caminos no es constante respecto al cambio del tamaño de la población, ya que si bien están relacionados el largo de un camino con el tamaño de la población, también depende de la distribución de los nodos en el espacio de búsqueda y como estos se relacionan entre sí (arcos) para encontrar las mejores configuraciones.

En la Tabla 5.7 se muestran los resultados de las métricas tras realizar la aproximación de las locaciones en las redes trayectoriales para la población de tamaño igual a 50. Nuevamente se evidencia una alta conectividad del grafo debido a la reducción de componentes conexas o independientes. Además el aumento de caminos hacia la mejor solución encontrada se hace coherente con la disminución de componentes conexas. Respecto al largo de los caminos también hay una

category	nodes	edges	nbest	nend	components	strength	plength	npaths
EVOCA 2-20	61	76	2	8	1	0.2	4.5	4
EVOCA 2-38	45	65	1	8	1	0.1	7.7	9
EVOCA 2-52	43	58	1	7	1	0.1	7.3	10
EVOCA 3-20	51	63	1	9	1	0.1	10.0	8
EVOCA 3-34	51	60	1	9	1	0.1	8.4	7
EVOCA 3-48	45	64	1	8	1	0.2	4.5	10
EVOCA 4-20	46	56	1	8	1	0.1	6.7	7
EVOCA 4-30	45	53	1	8	1	0.1	6.0	7
EVOCA 4-40	38	69	1	4	1	0.1	4.1	8
EVOCA 5-20	38	45	1	5	1	0.1	4.7	7
EVOCA 5-28	50	68	1	8	1	0.2	4.4	9
EVOCA 5-38	47	60	1	8	1	0.1	5.2	10
EVOCA 6-20	48	64	1	8	1	0.1	10.0	9
EVOCA 6-26	49	73	1	7	1	0.2	6.0	7
EVOCA 6-32	44	60	1	5	1	0.2	3.1	8

Tabla 5.6: Métricas de redes trayectoriales población 20- tamaño locación 0.2 x 0.05 x 6

reducción ya que el promedio sin aplicar la aproximación de locación es de 5.9, mientras que ahora es 5.7 (Comparando ambos escenarios con población inicial 50).

En los tres escenarios presentados se refleja un comportamiento similar, una disminución de componentes conexas debido a que al aproximar las locaciones, se puede decir que en cada ejecución de las instancias se exploraron zonas cercanas, esto también se refleja en la disminución de la cantidad de nodos y arcos debido a la aproximación de locaciones, debido a que se redujeron la cantidad de nodos y arcos las componentes conexas disminuyeron, ya que al aproximar locaciones se dió el caso de unificar trayectorias que antes eran independientes entre si (debido a que contenian nodos que se encontraban en una misma zona). Por otro lado, también se evidenció un aumento en los valores referentes a npath, que es la cantidad de caminos hacia las mejores soluciones encontradas, esto era esperable debido al aumento de componentes conexas.

Por otro lado, Además, para las Tablas 5.2, 5.3 y 5.4 se evidencia una relacion linean en la Figura 5.1a respecto a la cantidad de nodos y arcos muestran una relación lineal. Mientras que en los casos aproximados esta relación se pierde debido a la aproximación de locación (ver Figura 5.1b). Esto ocurre ya que los nodos que fueron unificados producen bifurcaciones en los caminos de la red por lo que la cantidad reducida de nodos no es proporcional a la de arcos. Esto se visualiza

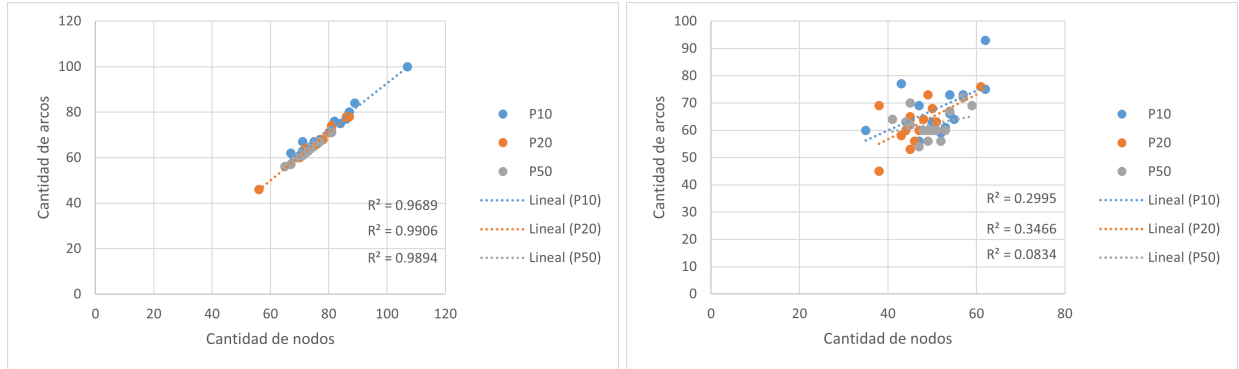
category	nodes	edges	nbest	nend	components	strength	plength	npaths
EVOCA 2-20	51	60	1	8	2	0.1	1.0	1
EVOCA 2-38	52	56	1	8	2	0.1	9.0	8
EVOCA 2-52	48	60	1	7	1	0.1	7.4	9
EVOCA 3-20	50	60	1	8	2	0.1	5.3	9
EVOCA 3-34	53	60	1	8	1	0.1	5.3	3
EVOCA 3-48	41	64	1	8	1	0.1	6.3	10
EVOCA 4-20	49	60	1	7	1	0.2	6.3	9
EVOCA 4-30	49	56	1	7	1	0.1	7.6	8
EVOCA 4-40	44	63	1	7	1	0.1	5.4	9
EVOCA 5-20	54	67	2	7	1	0.2	7.2	18
EVOCA 5-28	59	69	1	8	2	0.1	5.0	2
EVOCA 5-38	45	70	1	7	1	0.1	4.7	10
EVOCA 6-20	47	54	1	9	1	0.1	5.1	8
EVOCA 6-26	57	72	2	7	1	0.2	5.5	16
EVOCA 6-32	45	62	1	8	1	0.1	5.8	10

Tabla 5.7: Métricas de redes trayectoriales población 50- tamaño locación 0.2 x 0.05 x 6

en las Figura 5.1. Donde se muestra el eje X como cantidad de nodos, el eje Y como cantidad de arcos, cada punto corresponde al par (nodo, punto) para cada instancia.

Finalmente, el largo promedio de los caminos a las mejores configuraciones se reduce en aproximadamente un nodo, esto se debe a la aproximación de locación. Ya que al aproximar, existe la posibilidad de unificar nodos cercanos entre si, lo que crea dos posibilidades:

- El par de nodos ubicados en una misma zona se encuentran inicialmente unidos por un arco, esto hace desaparecer dicho arco (debido a que ambos nodos pasarían a estar representados por uno solo) y por ende se reduciría el largo del camino al último nodo.
- El par de nodos ubicados en una misma zona no se encuentran unidos inicialmente, lo que implica que hay dos rutas que interactúan con una misma zona del espacio, y al unificar a ambos nodos, dichas rutas tendrían un nodo en común, lo que también permite reducir el largo de un camino hacia las mejores configuraciones.



(a) Cantidad de arcos versus cantidad de nodos por instancia - locación 0.01 x 0.01 x 1 (b) Cantidad de arcos versus cantidad de nodos por instancia - locación 0.02 x 0.05 x 6

Figura 5.1: Cantidad de arcos respecto a la cantidad de nodos

5.2. Métricas grupales

A continuación se presentan métricas grupales, cada grupo esta conformado por individuos con el mismo valor de K respecto a las categorías de problemas NK-landscapes considerados. Inicialmente se presenta un análisis referente a la variación de calidad de las soluciones encontradas por el sintonizador (mejoramiento, empeoramiento y mantención de la calidad). Luego se presenta un análisis de la cantidad de nodos compartidos en cada ejecución (los nodos que comparten aquellas instancias con diferente tamaño de población inicial e igual K). Finalizando con la visualización de las redes trayectoriales, también agrupadas en base al mismo valor de K .

5.2.1. Variación de la calidad de las configuraciones visitadas

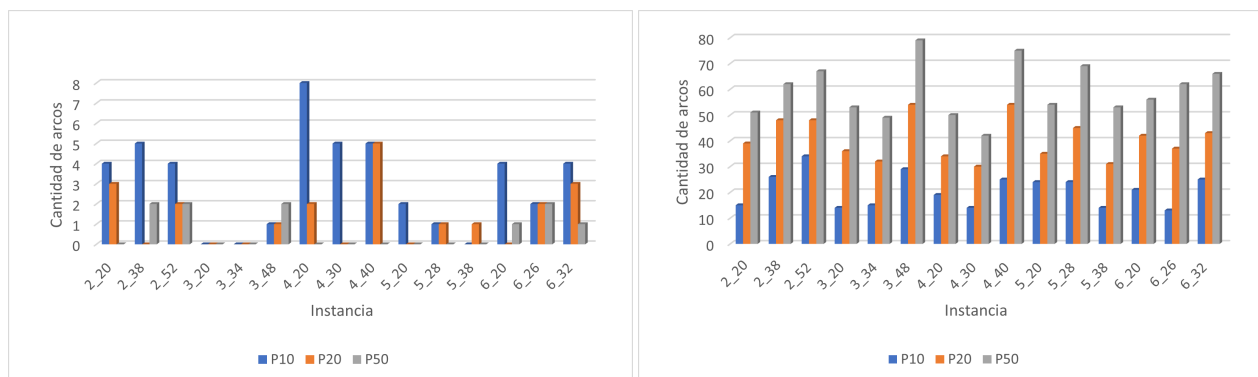
Con la finalidad de ver el comportamiento que tiene Evoca al momento de recorrer el espacio, se presenta un análisis del espacio de búsqueda del sintonizador de parámetros en base a la cantidad de arcos donde se empeora, mejora o mantiene la calidad de una solución o configuración de parámetros.

En las Figura 5.2 se tiene la cantidad de saltos entre configuraciones o arcos que llevaban a una solución de peor calidad. Donde en la Figura 5.2a se hace evidente que la ejecución con tamaño de población igual a 50 tiende a empeorar con menor frecuencia la calidad de las configuraciones

encontradas por el sintonizador que las poblaciones con tamaño inicial 10 y 20. Siendo esta última la que más tiende a empeorar las soluciones. Esto podría justificarse debido a que el tamaño de una población puede afectar a:

1. Variabilidad genética, una población más grande puede tener más variabilidad genética, lo que significa que puede contener una mayor variedad de soluciones potenciales. Esto puede aumentar las posibilidades de que el algoritmo evolutivo encuentre una buena solución.
2. Diversidad genética, una población más grande también puede tener más diversidad genética, lo que significa que puede contener una mayor variedad de genes diferentes. Esto puede ayudar a evitar que el algoritmo se atasque en un óptimo local y aumentar las posibilidades de que encuentre una buena solución.

Por otro lado, en la Figura 5.2b se tiene el caso contrario a la situación descrita anteriormente. Es decir, la ejecución con tamaño de población igual a 50 tiende a empeorar con mayor frecuencia la calidad de las configuraciones encontradas por el sintonizador que las poblaciones con tamaño inicial 10 y 20. Esto puede deberse a la aproximación en si, junto con la calidad inicial de aquellos nodos que fueron unificados y las relaciones que tenían con el resto de configuraciones dentro del espacio.



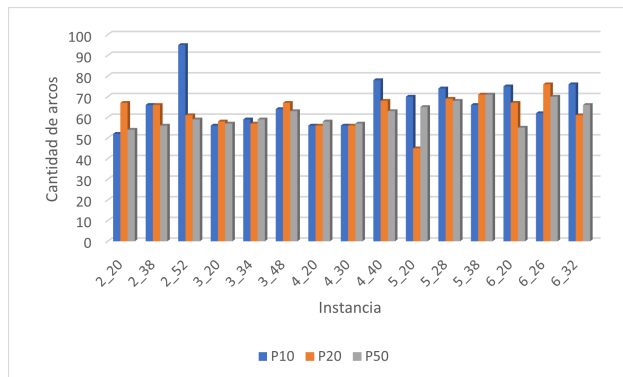
(a) Empeoramiento de calidad en la solución actual - tamaño locación 0.01 x 0.01 x 1 (b) Empeoramiento de calidad en la solución actual - tamaño locación 0.2 x 0.05 x 6

Figura 5.2: Cantidad de empeoramientos de una solución respecto a su calidad

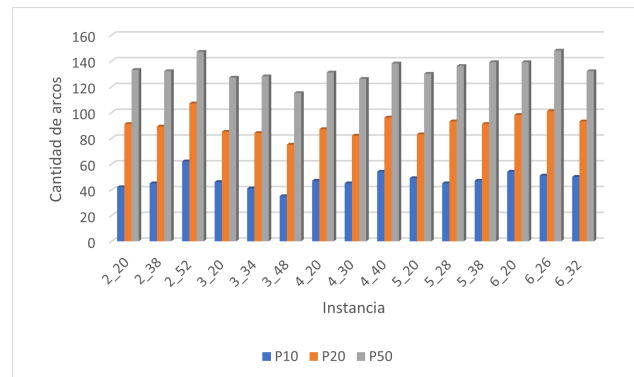
Los gráficos de las Figuras 5.3 representan la cantidad de veces en que la configuración encontrada es mejor que la actual en base a la calidad. Respecto a la Figura 5.3a se puede mencionar que

para todas las poblaciones se tiene una cifra estable. Mientras que en el caso de la Figura 5.3b se destacan dos cosas.

1. La tendencia a mejorar las soluciones de la población con tamaño 50 en comparación con las poblaciones 10 y 20, nuevamente se puede deber a los puntos mencionados recientemente, variabilidad y diversidad genética.
2. La reducción de casos de mejora para la población de tamaño 10 respecto a la Figura 5.3a, siendo de al menos 10 arcos aproximadamente. Esto nuevamente podría deberse a los nodos unificados al realizar la aproximación en el caso que un par de nodos coincida dentro de un mismo cuadrante.



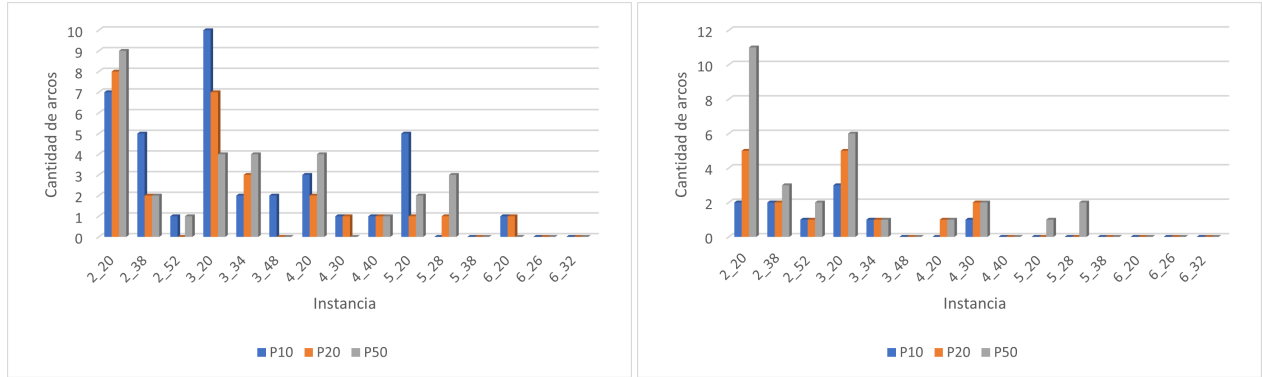
(a) Mejora de calidad en la solución actual - tamaño locación 0.01 x 0.01 x 1



(b) Mejora de calidad en la solución actual - tamaño locación 0.2 x 0.05 x 6

Figura 5.3: Cantidad de mejoras de una solución respecto a su calidad

Además, de la Figura 5.4 se tiene la cantidad de arcos de igualdad en las STN, o dicho de otra manera, los arcos donde no se varia la calidad de la solución. Esto se debe a que el encargado de la exploración en Evoca es la mutación. La mutación sólo cambia un valor a la vez por cromosoma. Por ende, los cambios son mínimos. Esto lleva a buscar continuamente soluciones similares y por ende, intensificar en una región reducida del espacio de búsqueda.

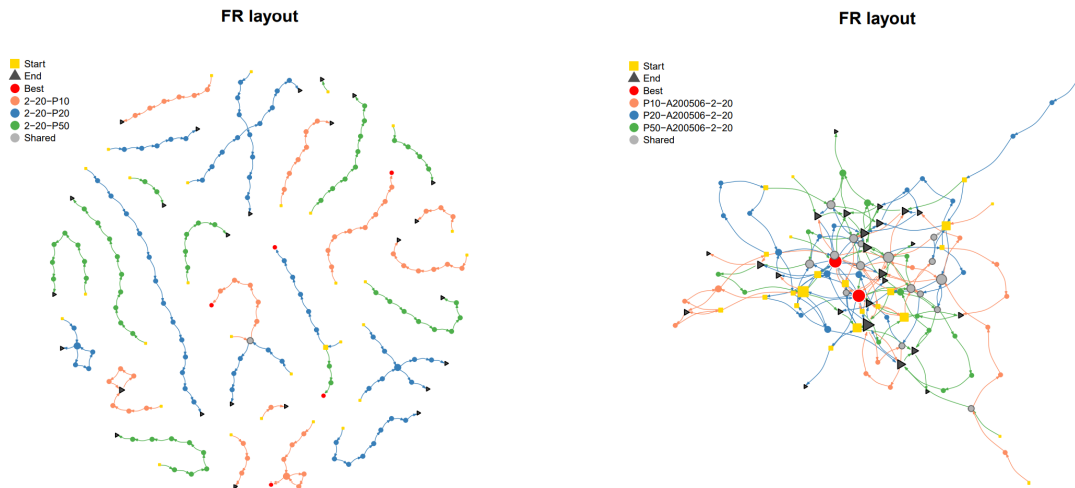


(a) Sin variación de calidad en la solución actual - tamaño locación 0.01 x 0.01 x 1 (b) Sin variación de calidad en la solución actual - tamaño locación 0.2 x 0.05 x 6

Figura 5.4: Cantidad de no variaciones en la solución respecto a su calidad

5.2.2. Nodos compartidos en cada ejecución

Según lo observado en las Tablas 5.2, 5.3 y 5.4 se evidencia el comportamiento de un algoritmo sintonizador con un fuerte énfasis en intensificar, es decir, concentrarse en una región específica del espacio de configuraciones con el fin de encontrar una solución óptima o muy cercana a ésta, lo que explicaría la baja cantidad de nodos compartidos empleado las locaciones iniciales. Esto visualiza en la Figura 5.5a. Mientras que al aproximar las locaciones se puede analizar de mejor manera zonas más visitadas y en qué momento de la ejecución de Evoca ocurre.



(a) 2-20 fusionada tamaño locación 0.01x0.01x1 (b) 2-20 fusionada tamaño locación 0.2x0.05x6

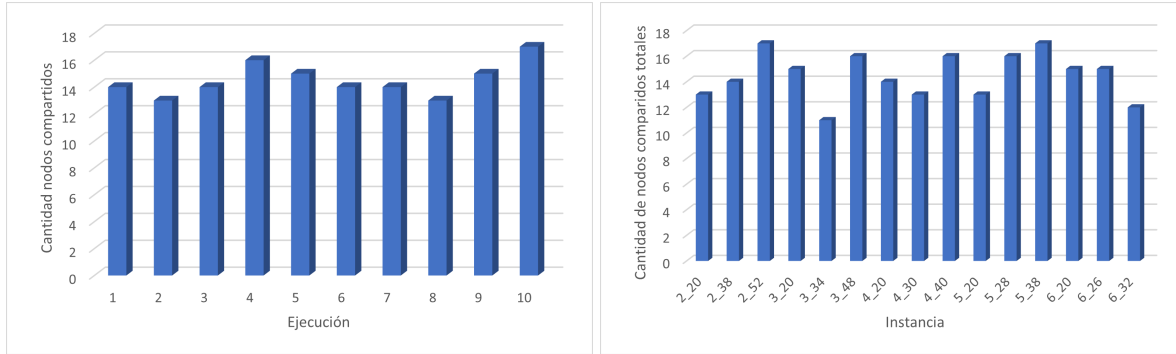
Figura 5.5: Categorías 2-20 fusionadas

En la Tabla 5.8 se tiene un conteo referente a la cantidad de nodos que comparte cada instancia en base a los diferentes tamaños de la población inicial (10, 20 y 50) por cada ejecución del algoritmo. Es decir, cada casilla contiene la cantidad de nodos que se comparten entre los tamaños de población 10, 20 y 50 en cada ejecución. En este caso el valor de 11 para la categoría 2-20 en la ejecución #1 significa que 11 nodos se repiten en la ejecución #1 entre los tamaños de población 10, 20 y 50. Si bien los valores no distan mucho entre sí, se puede apreciar un leve aumento en la cantidad de nodos compartidos en la mayoría de los casos a medida que el número de la ejecución es mayor. Por lo que se podría deducir una inclinación a también explorar el espacio de búsqueda, es decir, diversificar. Por otro lado, la última columna de la Tabla 5.8 corresponde al promedio de la cantidad de nodos que comparte cada instancia por ejecución para cada instancia, donde se aprecia una leve inclinación a aumentar la cantidad de nodos compartidos totales al aumentar el valor de N con K igual a 2 y 5. Mientras que con K igual a 6, se tiende a disminuir la cantidad de nodos compartidos. Cabe destacar que para los casos de K igual a 3 y 4, se tiene una cifra mayor con el valor más grande de N, pero no se presenta una tendencia clara como en los otros casos (Para mayor detalle revisar la Figura). Esto se relaciona con como afectan los valores de N y K al momento de diversificar o intensificar, y como varia dependiendo de los valores asociados a cada variable.

Instancia	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	$\bar{x}$
2-20	11	9	9	15	18	10	12	8	24	13	13
2-38	13	13	12	15	15	16	14	14	15	16	14
2-52	16	15	16	16	18	18	20	17	18	16	17
3-20	19	10	18	12	14	14	8	16	14	21	15
3-34	11	13	14	15	13	10	11	7	11	8	11
3-48	15	15	16	17	14	12	18	12	16	20	16
4-20	13	10	17	17	15	7	11	15	16	15	14
4-30	15	14	6	13	14	11	9	11	11	25	13
4-40	12	14	13	20	18	14	19	18	13	20	16
5-20	12	9	17	19	13	9	10	12	13	18	13
5-28	17	15	21	15	13	16	16	17	17	16	16
5-38	15	17	15	23	17	18	17	13	15	18	17
6-20	10	17	16	16	20	16	19	11	14	14	15
6-26	20	14	10	16	7	17	13	17	19	18	15
6-32	8	4	17	13	10	15	13	9	10	12	12
$\bar{x}$	14	13	14	16	15	14	14	13	15	17	

Tabla 5.8: Nodos compartidos - tamaño locación 0.2 x 0.05 x 6

En la Figura 5.6a se tiene el promedio de los datos representados en la Tabla 5.8. (para mayor detalle revisar las Figuras 5.37). Mientras que en la Figura 5.6b se tiene el promedio de los nodos compartidos para cada instancia de la Tabla 5.8.



(a) Promedio de nodos compartidos por ejecución (b) Promedio de nodos compartidos por instancia - tamaño locación 0.2 x 0.05 x 6

Además, al tratarse de ejecuciones independientes, es interesante identificar que se tiende a coincidir en configuraciones a medida que se explora más el espacio de búsqueda, esto podría ser un indicador de que esa zona presenta configuraciones de buena calidad.

Cabe destacar que se realizó este análisis únicamente con la locación aproximada debido a que la cantidad de nodos compartidos dentro de la locación original (0.01x0.01x1) era prácticamente nula, entonces se complejizaba analizar cercanía entre configuraciones y por ende zonas dentro del espacio.

5.3. Visualización de redes trayectoriales

En la Figura 5.7 se presentan las redes trayectoriales para K igual a 2 y N a 20 sin aproximar las locaciones. Las redes presentadas cuentan con 71, 86 y 73 nodos junto con 63, 78 y 63 arcos para las poblaciones de Evoca de 10, 20 y 50 configuraciones respectivamente. Se destaca que la población tamaño 10 encuentra tres soluciones óptimas mientras que en las poblaciones de tamaño 20 y 50 cada una encuentra una única configuración óptima. Por su parte, el camino más corto a una solución óptima lo tiene la red con mayor población inicial, es decir, 50 siendo igual a 3, mientras que para las otras poblaciones es igual a 7. Todas las STNs presentadas tienen una cantidad similar

de componentes conexas (9 para la población de tamaño 20 y 10 para las poblaciones de tamaño 10 y 50).

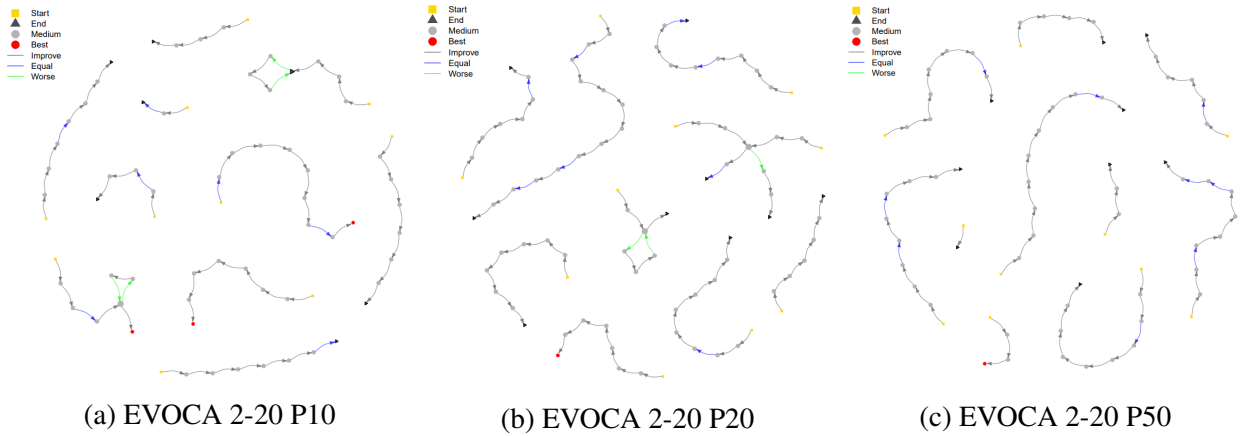


Figura 5.7: Redes trayectoriales de categoría 2_20 - tamaño locación 0.01x0.01x1

Las redes trayectoriales de la Figura 5.8 corresponden a la aproximación de locaciones del caso anterior (K igual a 2 y N a 20). En este caso la cantidad de nodos se redujo a 52, 61 y 51, mientras que la cantidad de arcos quedó en 59, 76 y 60 para las poblaciones 10, 20 y 50 respectivamente. La cantidad de rutas a las soluciones óptimas encontradas aumentó de tres a 16, una a cuatro para las poblaciones de tamaños 10 y 20. En las redes de la población con tamaño 50 se mantuvo en uno. Esto depende netamente de la distribución de las configuraciones o soluciones en el espacio de búsqueda, pero al tener mayor cantidad de soluciones encontradas en el experimento con población inicial igual a 10 es esperable que existan más caminos hacia la solución óptima (debido a que son más). Por otro lado, la cantidad de componentes conexas fue reducida de dos o una componente, lo que indica que las zonas visitadas por Evoca tienden a ser “cercanas”. Esto es coherente con el comportamiento esperado de los operador de cruzamiento y mutación que trabaja el sintonizador ya que hace que se enfoque en intensificar.

En la Figura 5.9 se presentan las redes correspondientes a K igual a 2 y N igual a 38, donde se tienen 82, 78 y 69 nodos, junto con 76, 68 y 60 arcos. En todas las redes se encuentra un único óptimo y se tiene una única ruta hacia este. Además la cantidad de componentes conexas nuevamente se encuentra entre 9 y 10, lo que indica que en cada ejecución el algoritmo se enfocó en intensificar en distintas zonas del espacio de búsqueda. Por otro lado, se destaca que el largo del camino hacia la solución óptima se reduce a medida que se aumenta la población inicial, pasa

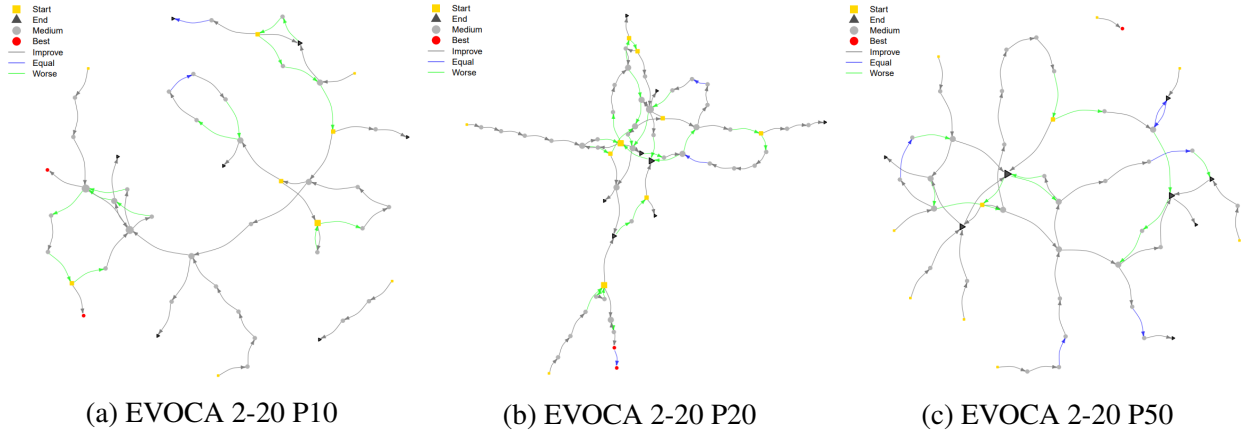


Figura 5.8: Redes trayectoriales de categoría 2_20 - tamaño locación 0.2x0.05x6

de nueve a siete para finalizar en tres.

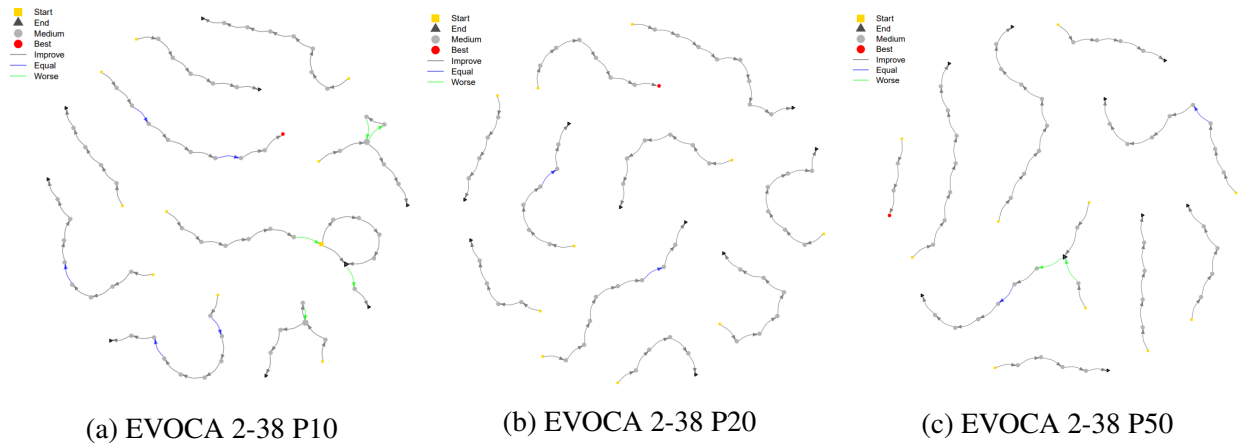


Figura 5.9: Redes trayectoriales de categoría 2_38 - tamaño locación 0.01x0.01x1

Las STNs presentadas en la Figura 5.10 corresponden a la aproximación de las locaciones de las redes de la Figura 5.9. La cantidad de nodos para los experimentos con población 10, 20 y 50 son 57, 45 y 52 respectivamente, mientras que en los arcos es 73, 65 y 56. Como se aprecia, la reducción de nodos fue mayor que la de arcos, esto se debe a la unificación de nodos que, a su vez, genera de bifurcaciones en las rutas. Esto, dado que si bien cada ejecución puede tener nodos en común, toman direcciones diferentes posterior a cada visita. Por otro lado, la cantidad de rutas hacia las soluciones óptimas encontradas aumenta prácticamente de igual manera para todas, inicialmente se tenía una única ruta para cada tamaño de población, ahora ese valor se encuentra entre ocho y nueve. Cabe mencionar que en la Figura 5.9c se destaca que a diferencia de los otros dos casos

se explora más el espacio de búsqueda, es decir, explora diferentes áreas en lugar de enfocarse en zonas específicas. Esto, debido a que la cantidad de bifurcaciones disminuye en comparación al resto.

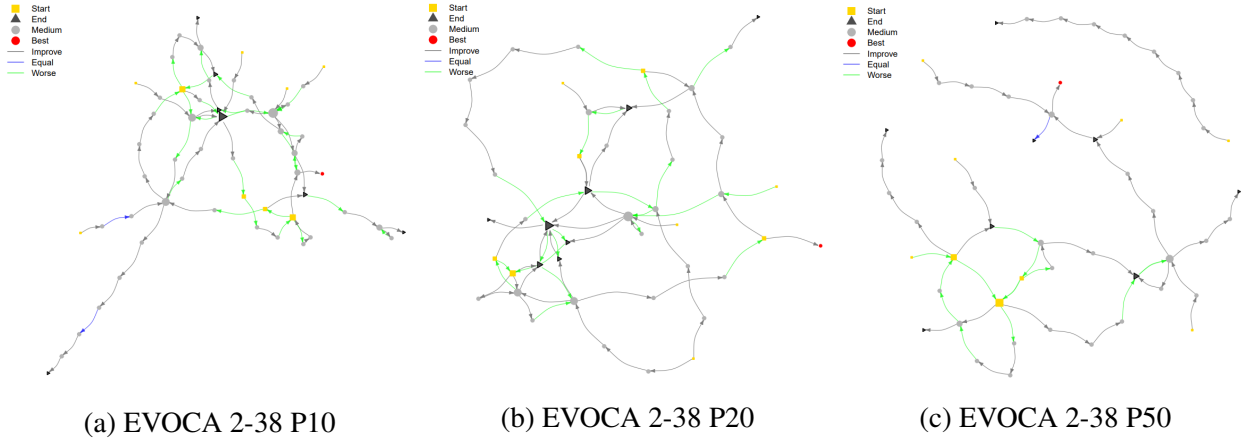


Figura 5.10: Redes trayectoriales de categoría 2_38 - tamaño locación 0.2x0.05x6

En las redes con K igual a 2 y N a 52 se tienen 107, 72 y 71 nodos; y 100, 63 y 62 arcos para los experimentos con población 10, 20 y 50 respectivamente. La cantidad de componentes se mantiene entre nueve y diez, las cuales se pueden apreciar en la Figura 5.11. Nuevamente se tiene una única solución óptima encontrada para todos los casos, junto con la tendencia a disminuir el largo del camino a dicha solución a medida que se aumenta la población inicial, donde es 12, 6 y 5 para las poblaciones de tamaños 10, 20 y 50, respectivamente.

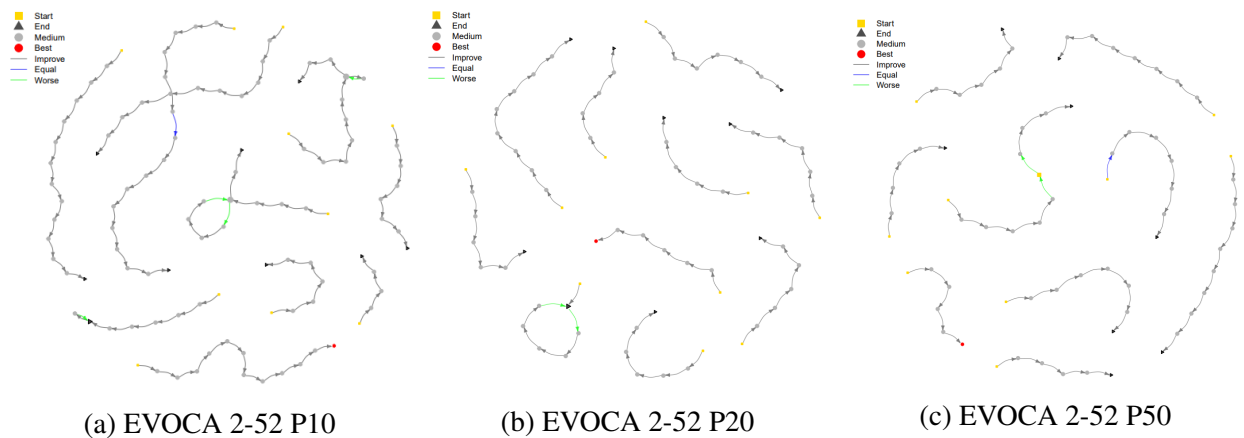


Figura 5.11: Redes trayectoriales de categoría 2_52 - tamaño locación 0.01x0.01x1

Para la versión aproximada de K igual a 2 y N a 52 nuevamente se tiene una mayor disminución

en la cantidad de nodos que de arcos, pero a medida que la población inicial aumenta se puede distinguir en la Figura 5.12 como se abarca mayor área del espacio de búsqueda. También la cantidad de componentes conexas aumenta de nueve a siete . Nueve para tamaño de población igual a 10 y siete para tamaño de población 20 y 50.

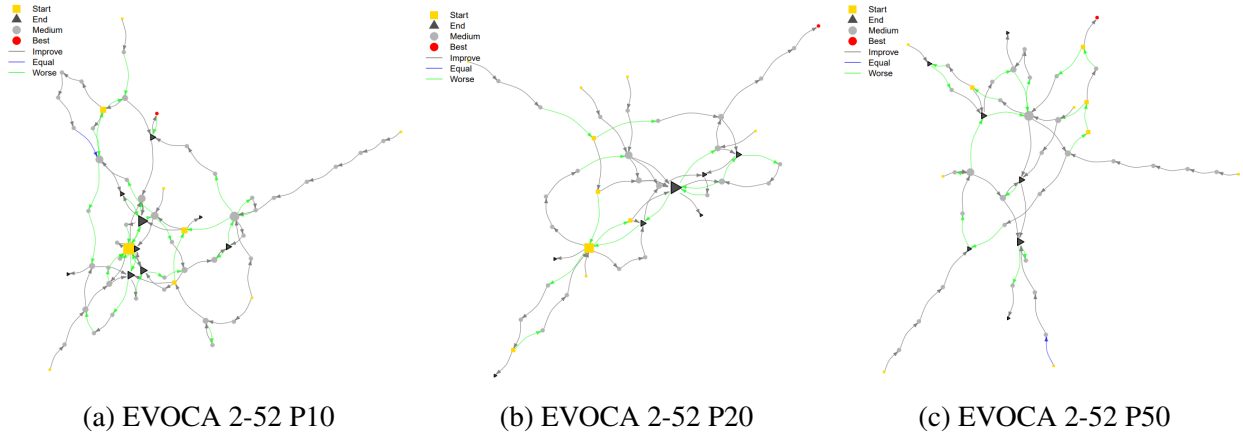


Figura 5.12: Redes trayectoriales de categoría 2_52 - tamaño locación 0.2x0.05x6

En la Figura 5.13 se tienen las STNs referentes a los valores 3 y 20 para K y N variando el p_s . Se destaca que en el caso de la Figura 5.13a se obtienen dos caminos para la solución óptima encontrada, mientras que en las redes para p_s igual a 20 y 50 se tiene únicamente un camino. El camino más corto hacia el óptimo es de largo seis, obtenido con p_s 50, le sigue p_s 10 con largo 7.5, y largo 10 para p_s 20.

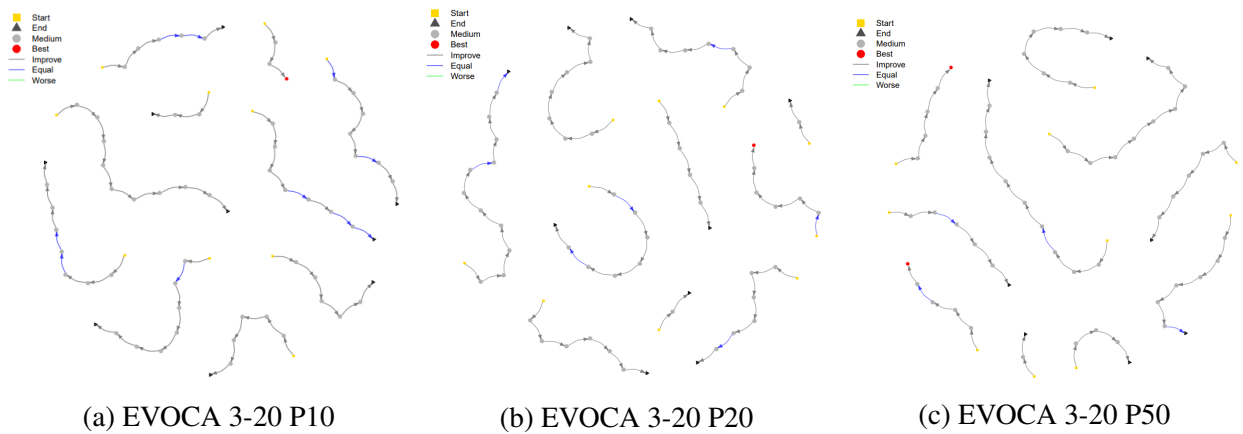


Figura 5.13: Redes trayectoriales de categoría 3_20 - tamaño locación 0.01x0.01x1

En la Figura 5.14 se tiene la aproximación de locación para las redes anteriormente comentadas

($N=3$, $K=30$). Lo que se destaca principalmente es el largo del camino más corto hacia el óptimo encontrado en cada red, donde continua manteniendo el primer puesto la población de tamaño 50 con 5.3, nuevamente se percibe que dicho experimento tiende a explorar más el espacio de búsqueda que aquellas con una menor población inicial.

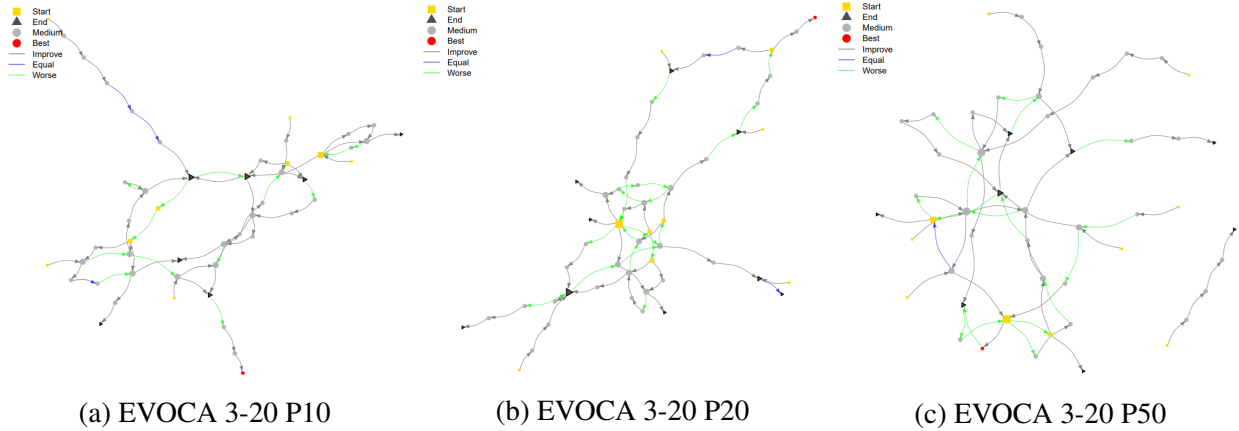


Figura 5.14: Redes trayectoriales de categoría 3_20 - tamaño locación 0.2x0.05x6

Para K igual a 3 y N a 34 se tienen las redes trayectoriales de la Figura 5.15, donde se tienen 70, 70 y 73 nodos y 61, 60 y 63 arcos para p_s 10, 20 y 50 respectivamente. Como se ha visto en todos los casos con la misma locación (0.01x0.01x1) la cantidad de componentes conexas se mantiene entre nueve y diez. El camino más corto hacia el óptimo encontrado lo sigue teniendo la población inicial de 50, pero en esta ocasión la población inicial de 10 cuenta con dos caminos hacia el óptimo encontrado, mientras que el resto tiene únicamente una ruta.

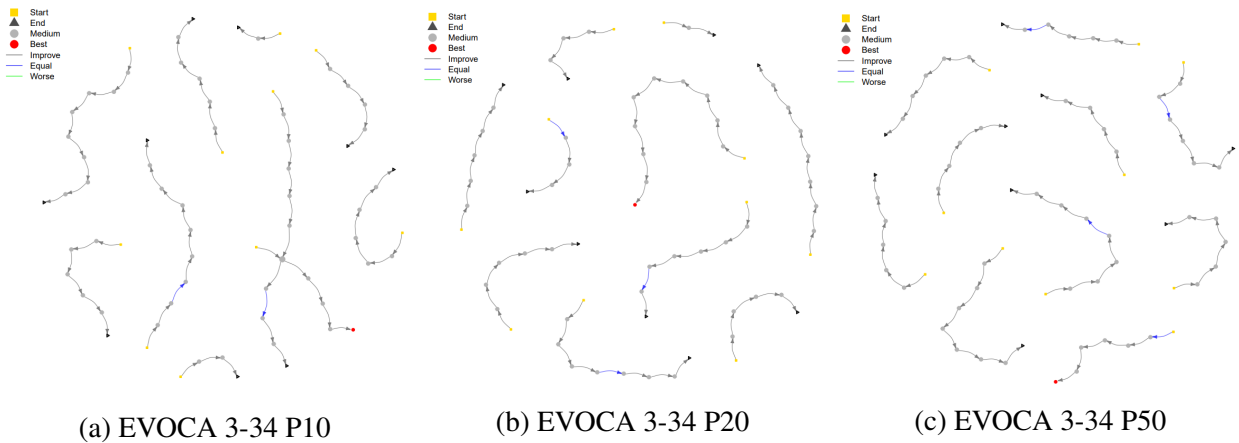


Figura 5.15: Redes trayectoriales de categoría 3_34 - tamaño locación 0.01x0.01x1

A partir de las métricas obtenidas tras obtener las redes trayectoriales con $K = 3$ y $N = 34$ que se muestran en la Figura 5.16 se mantiene con el camino más corto hacia el óptimo encontrado la población con tamaño 50, puntuando 5.3, seguida de p_s 10 con 6.0 y finalmente p_s 20 con 8.4 . Los valores obtenidos sin aproximar las locaciones son 6, 7.5 y 10, respectivamente. Cabe destacar que la cantidad de caminos que llevan al óptimo para el caso de la población de tamaño inicial 20 es 7, mientras que para p_s 10 y 50 es 2 y 3. También hay que tener en consideración la forma que tiene la red trayectorial en el caso de p_s igual a 20, la cual tiende a ser más lineal que el resto (no cuenta con un gran ciclo central).

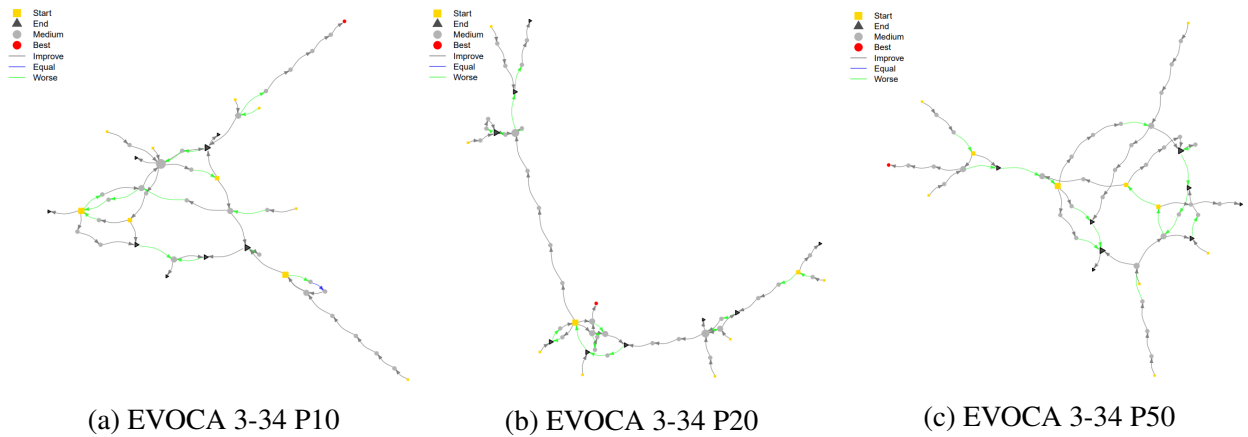


Figura 5.16: Redes trayectoriales de categoría 3_34 - tamaño locación 0.2x0.05x6

En la Figura 5.17 se presentan las redes trayectoriales para K igual a 3 y N a 48, presenta el mismo comportamiento que el resto, entre nueve y diez componentes conexas para todos los casos, una única solución óptima encontrada, con la diferencia que en el caso de la población inicial de tamaño 50 hay dos rutas hacia dicha solución, mientras que en las otras redes solo hay un camino.

La Figura 5.18 muestra las redes trayectoriales con locación aproximada para $K = 3$ y $N = 48$, donde se destaca una gran reducción de nodos (casi a la mitad) llegando a 35, 45 y 41 para p_s 10, 20 y 50 respectivamente. Mientras que la cantidad de arcos a lo más fue reducida en seis arcos. Por otro lado, el camino más corto hacia el óptimo encontrado es 4.5, seguido de 4.6 para las poblaciones 20 y 10, es decir, para este caso en cuestión, el *plength* de la población inicial 50 es mayor (6.3) que el resto. Finalmente, hay que mencionar que la cantidad de caminos que llegan al óptimo está entre 8 y 10.

Para los valores K y N de 4 y 20 respectivamente se presentan las redes trayectoriales en la

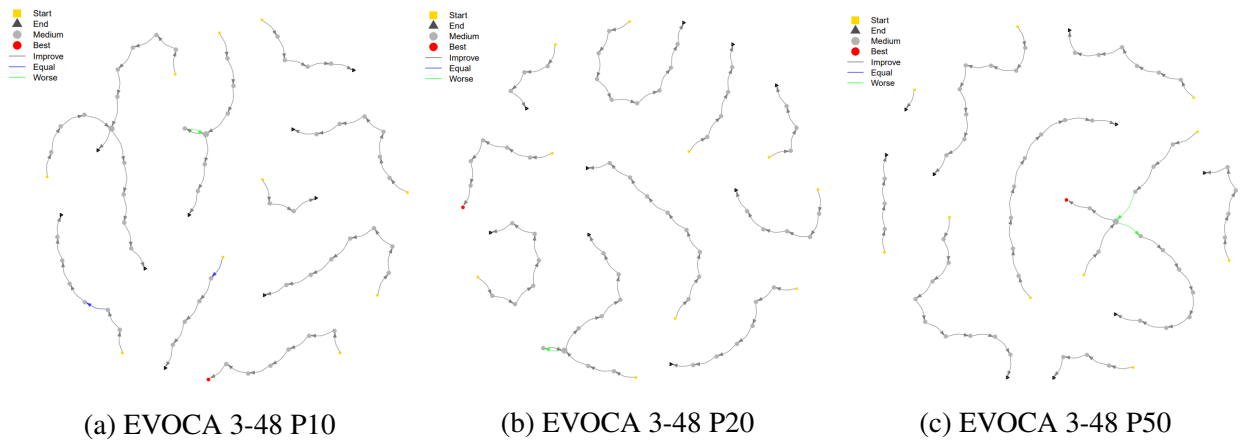


Figura 5.17: Redes trayectoriales de categoría 3_48 - tamaño locación 0.01x0.01x1

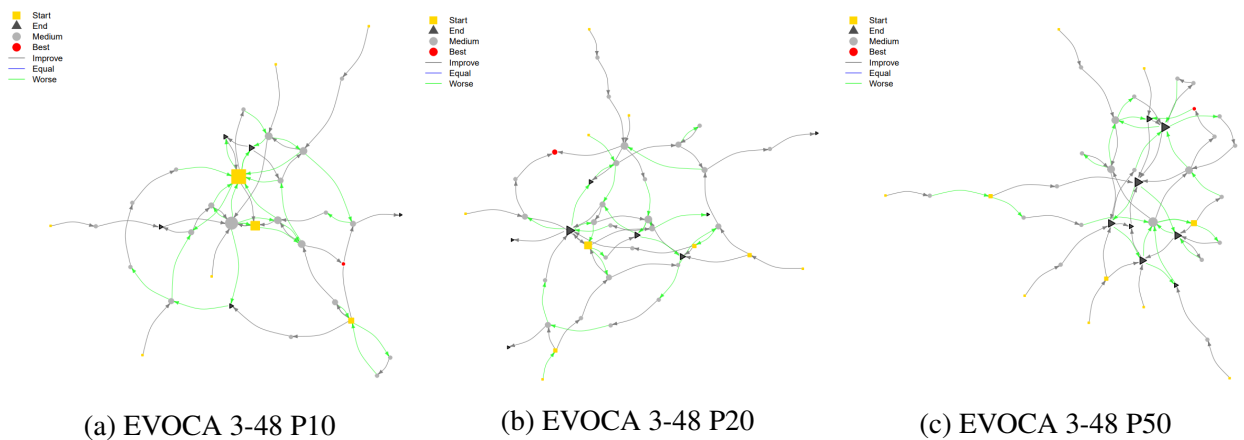


Figura 5.18: Redes trayectoriales de categoría 3_48 - tamaño locación 0.2x0.05x6

Figura 5.19. Lo que se destaca es que el *plength* o largo promedio hacia la mejor configuración encontrada es mayor para la población de tamaño 50 siendo de 9.0, seguida de la población de tamaño 20 con un largo de 5.0 y la población de tamaño 10 con largo 3.0. Hay que aclarar que nuevamente se encuentra una única solución por cada experimento y la cantidad de componentes conexas es 10 para todos los tamaños de población.

Al aproximar las locaciones de la Figura 5.19 se obtienen las redes de la Figura 5.20, donde se tiene el caso contrario referente al largo promedio para llegar a la mejor configuración, puntuando 7.5, 6.7 y 6.3 para los tamaños de población de 10, 20 y 50 respectivamente. Esto muestra que, a medida que se aumentaba la población, se tendió a investigar configuraciones pertenecientes a zonas cercanas. Así, al realizar la aproximación de locación, las soluciones cercanas pasaron a

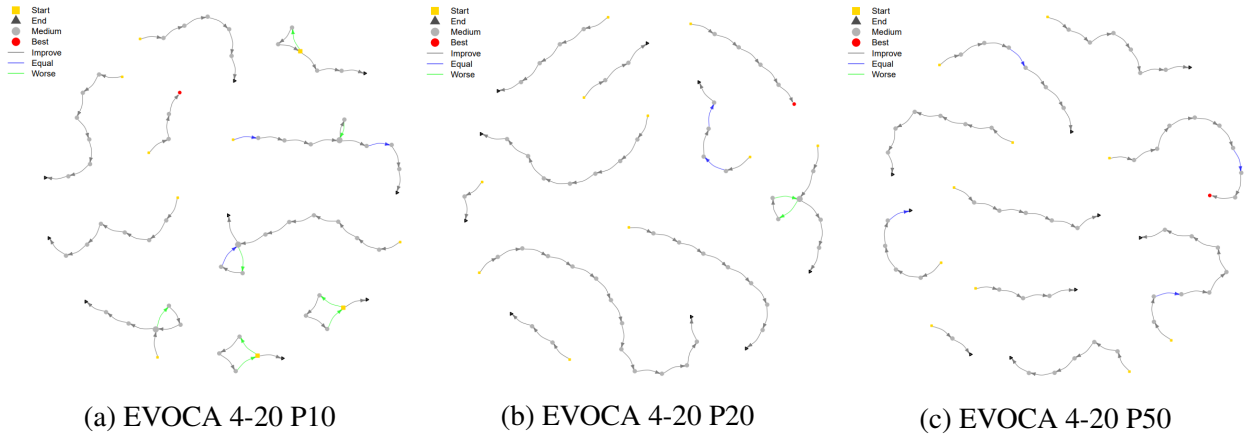


Figura 5.19: Redes trayectoriales de categoría 4_20 - tamaño locación 0.01x0.01x1

estar representadas por únicamente una locación y, por ende, se disminuyó el largo promedio para llegar al óptimo encontrado. Además, el tener entre 7 y 9 rutas hacia la mejor solución encontrada y aún así un valor bajo asociado al promedio nuevamente es un indicador de que el sintonizador trabajó con soluciones cercanas.

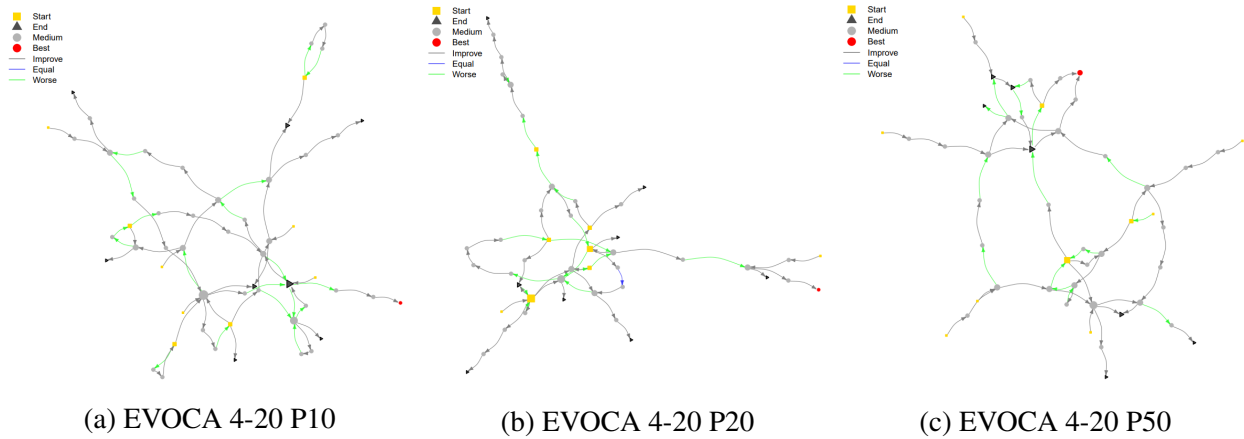


Figura 5.20: Redes trayectoriales de categoría 4_20 - tamaño locación 0.2x0.05x6

En los experimentos referentes a K igual a 4 y N 30, se obtuvieron las redes trayectoriales de la Figura 5.21 donde nuevamente se encuentra una única solución óptima por red, el largo del camino al óptimo es 5 para las poblaciones de tamaño 10 y 50, mientras que la población de tamaño 20 tiene un largo de 8. Esto depende directamente de la forma en que se recorra el espacio junto con el punto de inicio ya que en base a esto se arma la trayectoria en si.

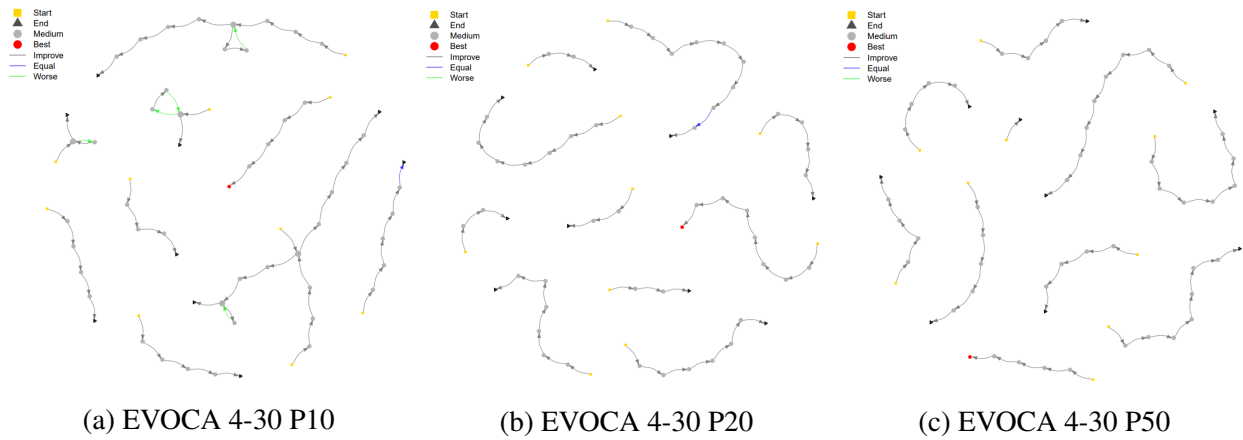


Figura 5.21: Redes trayectoriales de categoría 4_30 - tamaño locación 0.01x0.01x1

La visualización de las redes aproximadas para K igual a 4 y N a 30 se encuentra en la Figura 5.22. Lo que más se destaca es referente a la Figura 5.22a, donde la cantidad de caminos hacia el óptimo encontrado es 18 (al menos 10 veces más que para los demás casos), lo que indica que en este caso tendió a intensificar la búsqueda de soluciones en una zona específica del espacio de búsqueda ya que además, es el menor promedio de largo de los caminos o rutas hacia la mejor solución (5.0). Lo que indica que los puntos iniciales tienden a estar ubicados en zonas cercanas.

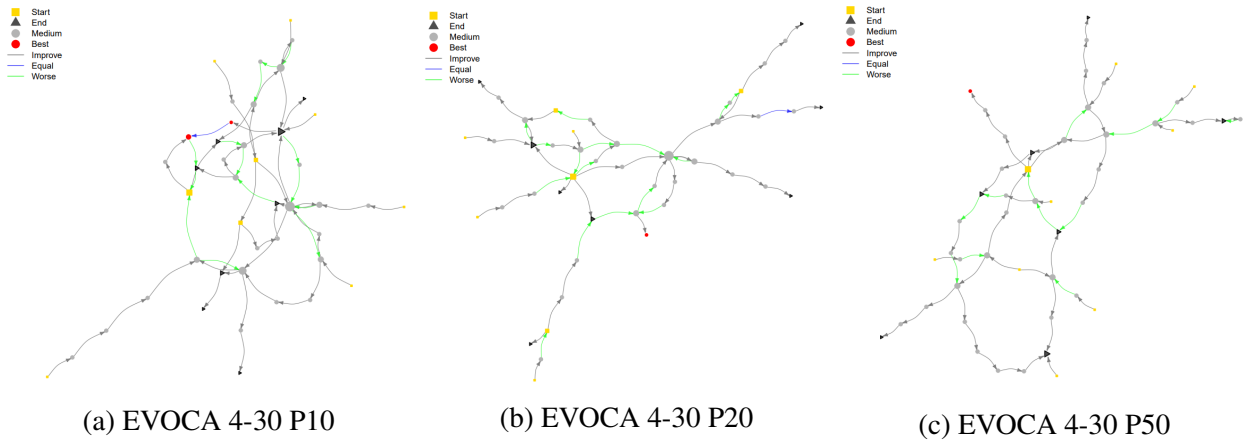


Figura 5.22: Redes trayectoriales de categoría 4_30 - tamaño locación 0.2x0.05x6

En el caso de las redes de la Figura 5.23 se tiene que la población de mayor tamaño tiene el camino más corto hacia el óptimo (7), pero se debe mencionar que nuevamente para todos los casos existe una única solución catalogada como óptima.

Al aproximar las locaciones de las redes con $N=4$ y $K=40$ se destaca que la red con p_s igual a

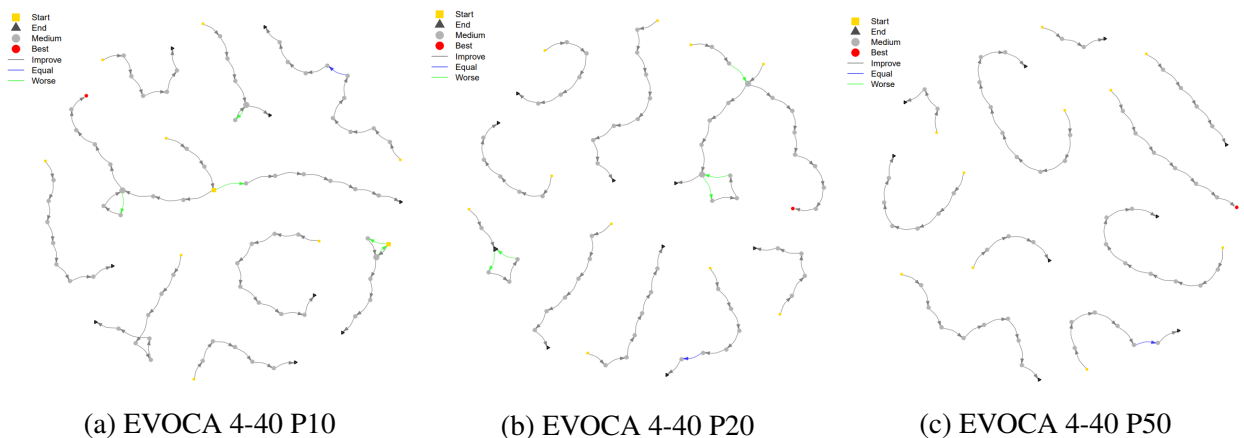


Figura 5.23: Redes trayectoriales de categoría 4_40 - tamaño locación 0.01x0.01x1

20 es quien tiene la ruta promedio más corta al óptimo encontrado (4.1) de un total de 8 caminos. Lo que nuevamente es un indicador de la tendencia de Evoca a intensificar en zonas del espacio de búsqueda.

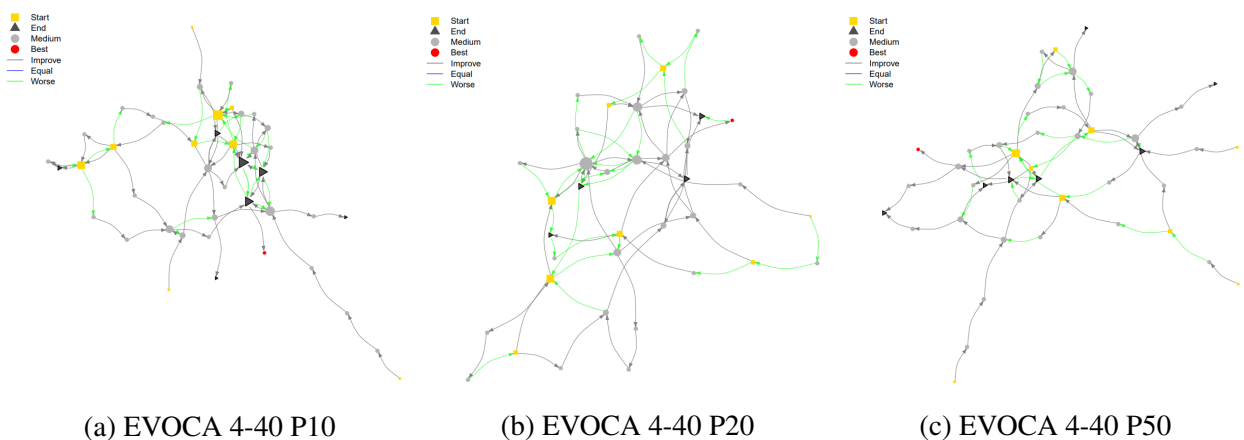


Figura 5.24: Redes trayectoriales de categoría 4_40 - tamaño locación 0.2x0.05x6

Para el caso de $N=5$ y $K=20$ se destaca que la red con población inicial de 20 tiene un total de 56 nodos y 46 arcos, mientras que para p_s 10 y 50 se tienen 86 nodos, 77 arcos y 77 nodos y 67 arcos, respectivamente. Lo que indica que se tiende a intensificar la búsqueda dentro del espacio de búsqueda. Por otro lado, la red con p_s logra encontrar dos soluciones óptimas.

Al analizar las redes de la Figura 5.26 donde se aproxima la locación, considerando para $K=5$ y $N=20$ se confirma que en el caso de p_s 20 se intensifica debido a que se tiene una única componente conexa con un largo promedio a la solución óptima de 4.7 (el menor valor para estos valores de N

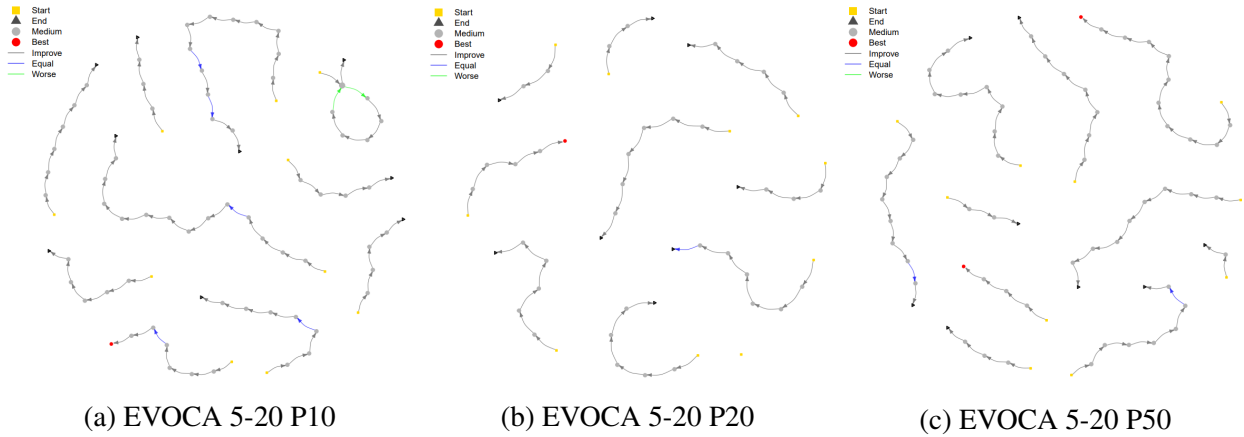


Figura 5.25: Redes trayectoriales de categoría 5_20 - tamaño locación 0.01x0.01x1

y K). Por otro lado, la red con p_s 50 cuenta con el valor más alto para llegar a un óptimo. Además, contar con dos soluciones con la mejor aptitud hace sentido a que tenga 18 caminos diferentes para encontrar un óptimo (dada la conectividad de la red, ya que solo hay una componente conexa).

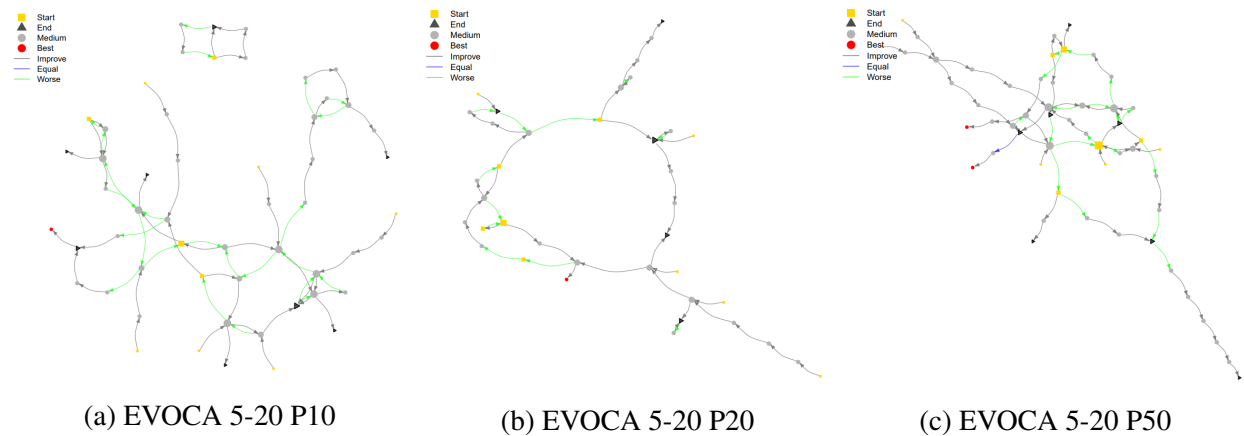


Figura 5.26: Redes trayectoriales de categoría 5_20 - tamaño locación 0.2x0.05x6

En el caso de $N=5$ y $K=28$ se tienen diez componentes conexas para los tres tamaños de población. Para mas detalle revisar Figura 5.27. Todas las soluciones tienden a comportarse de manera similar, entre 80 y 84 nodos, 71 y 75 arcos, una solución óptima encontrada por red donde el largo del camino se encuentra entre 7 y 8. Cabe destacar que la ruta encontrada hacia el óptimo es única.

Al aproximar las locaciones se destaca el comportamiento de la red con p_s igual a 50, ya que únicamente hay 2 rutas hacia el óptimo. Esto indica que, comparado con las redes de población 10 o 20, no tiene un comportamiento tan marcado respecto a la intensificación. Para mayor detalle

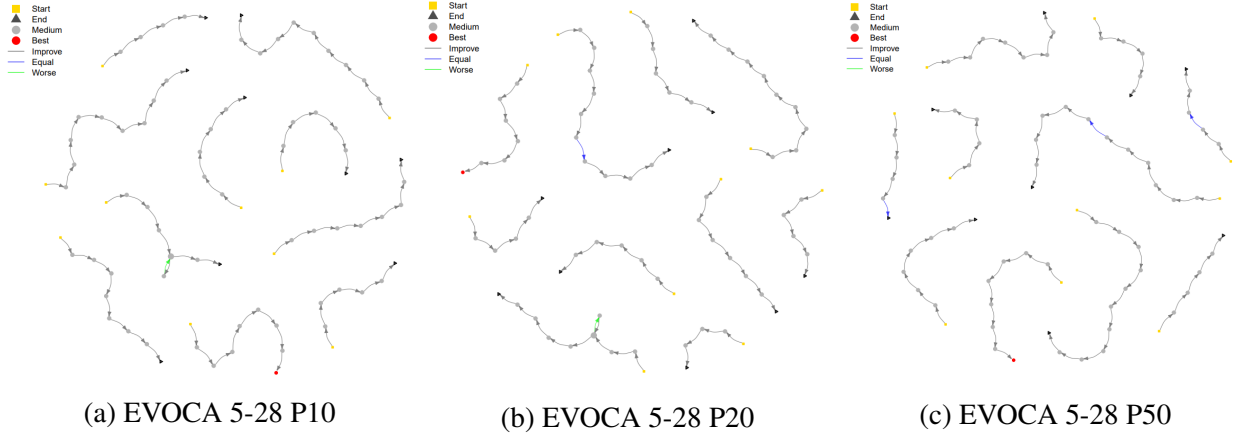


Figura 5.27: Redes trayectoriales de categoría 5_28 - tamaño locación 0.01x0.01x1

observar Figura 5.28.

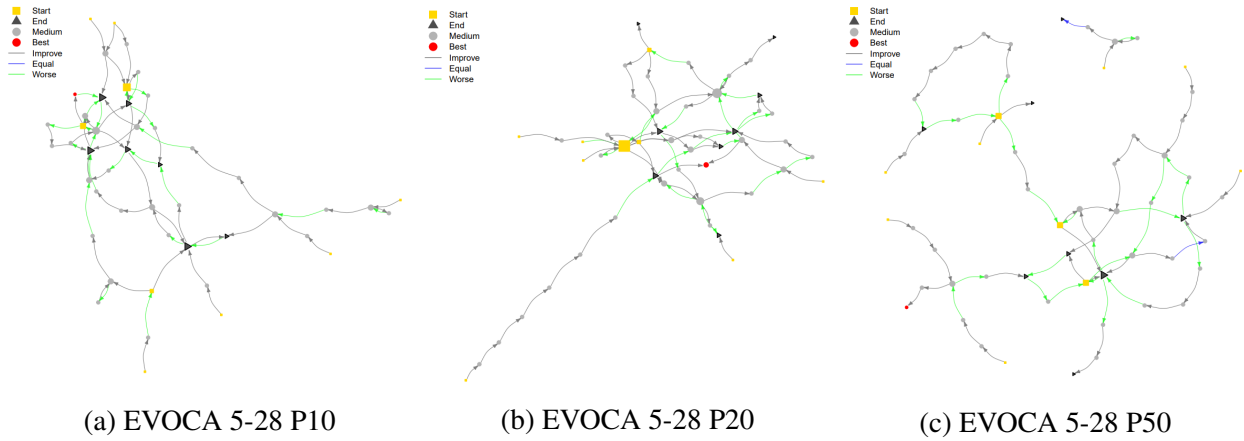


Figura 5.28: Redes trayectoriales de categoría 5_28 - tamaño locación 0.2x0.05x6

Respecto a la Figura 5.29, se mantiene el comportamiento general visto hasta el momento, diez componentes conexas, una única solución encontrada y un único camino hacia dicha configuración, el largo de la ruta varia entre 3 y 7.

Al aproximar las locaciones, considerando como K y N a 5 y 38 respectivamente, se obtienen las redes de la Figura 5.30. Se destaca que en la red con p_s igual a 10 se mantiene un único camino al óptimo, por lo que podría indicar que tiende a diversificar o considerar soluciones distantes en el espacio pero cuenta con únicamente dos componentes conexas, lo que indica que se centró en intensificar en 9 de las 10 ejecuciones independientes en zonas cercanas pero sin éxito, es decir,

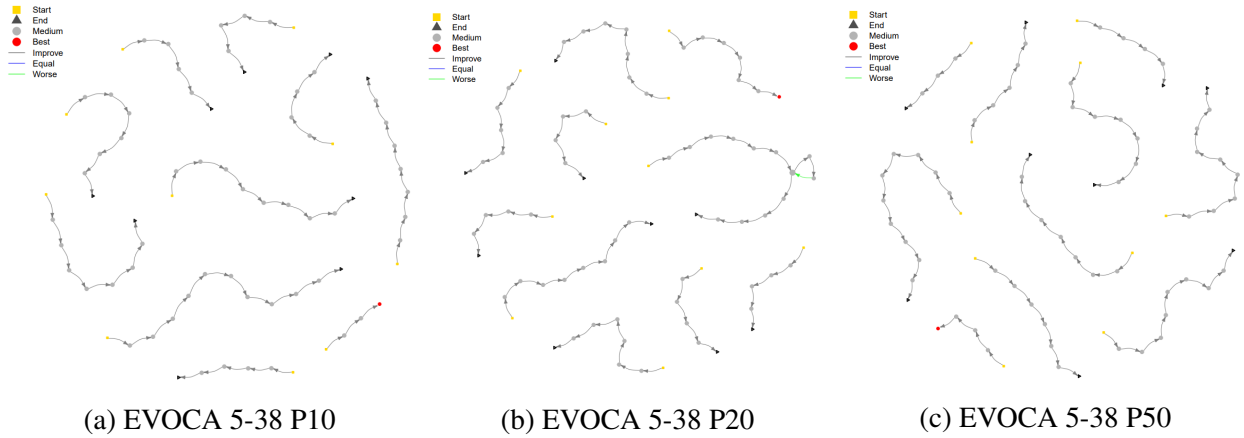


Figura 5.29: Redes trayectoriales de categoría 5_38 - tamaño locación 0.01x0.01x1

sin encontrar un valor óptimo en esa área.

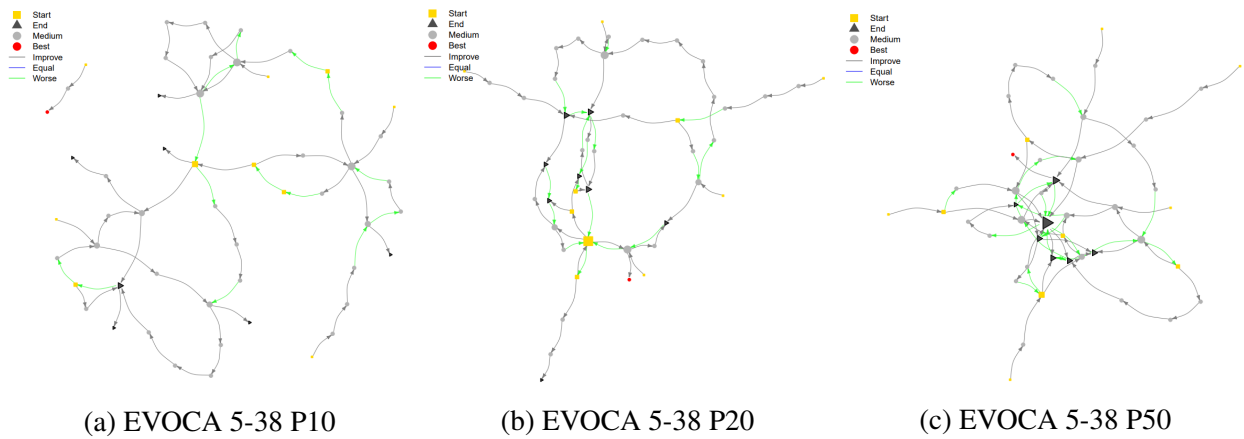


Figura 5.30: Redes trayectoriales de categoría 5_38 - tamaño locación 0.2x0.05x6

En la Figura 5.31 se presentan las redes trayectoriales con K igual a 6 y N a 20. Donde nuevamente se encuentra una única iteración en los tres experimentos (p_s igual a 10, 20 y 50). Todas las redes cuentan con 10 componentes conexas pero sigue habiendo una única ruta a la mejor solución encontrada donde el camino más corto se encuentra en la red con población inicial igual a 50 puntuando un camino promedio de 5.0.

Al aproximar la locación de las redes descritas recientemente se tienen 62, 48 y 47 nodos junto a 75, 64 y 54 arcos para un p_s igual a 10, 20 y 50 respectivamente. En cuanto al largo de los caminos para llegar a la mejor solución encontrada la población inicial de tamaño 50 se mantiene en primer lugar con 5.1, mientras que el *plength* para las redes restantes aumenta, para el p_s igual

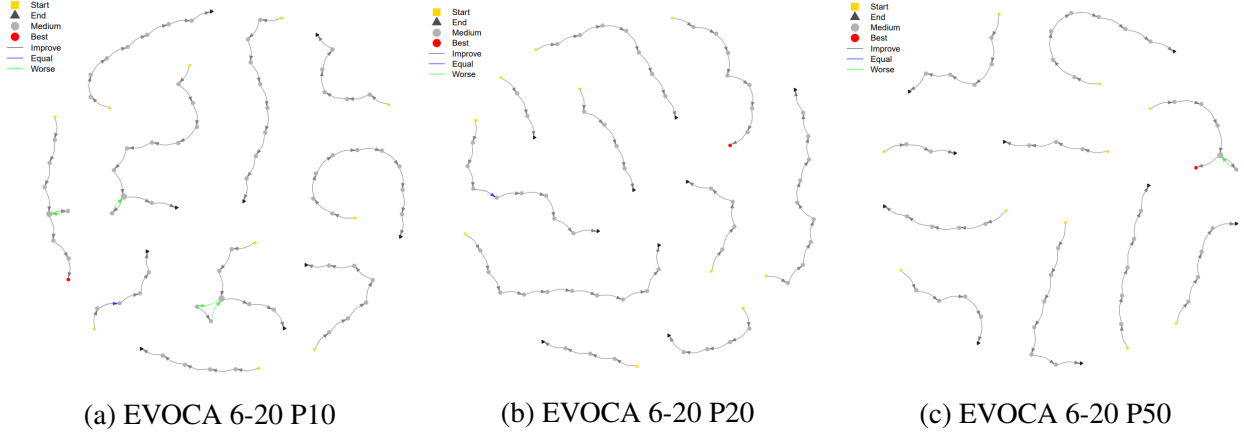


Figura 5.31: Redes trayectoriales de categoría 6_20 - tamaño locación 0.01x0.01x1

a 10 aumenta de 7.0 a 8.4 y para el p_s de 20 pasa de 8.0 a 10.0. En este caso aquellas redes con un valor mayor de *plength* tienden a tener sus nodos de inicio más alejados o repartidos por el espacio, mientras que para la población de tamaño inicial igual a 50 se da que la mayoría de sus puntos de partida se encuentran cerca de la solución óptima. También hay que considerar que, frente a valores considerados como fuera de lo común u “outliers”, el promedio tiende a no ser una buena métrica, lo que en parte ayuda a explicar el aumento del *plength*.

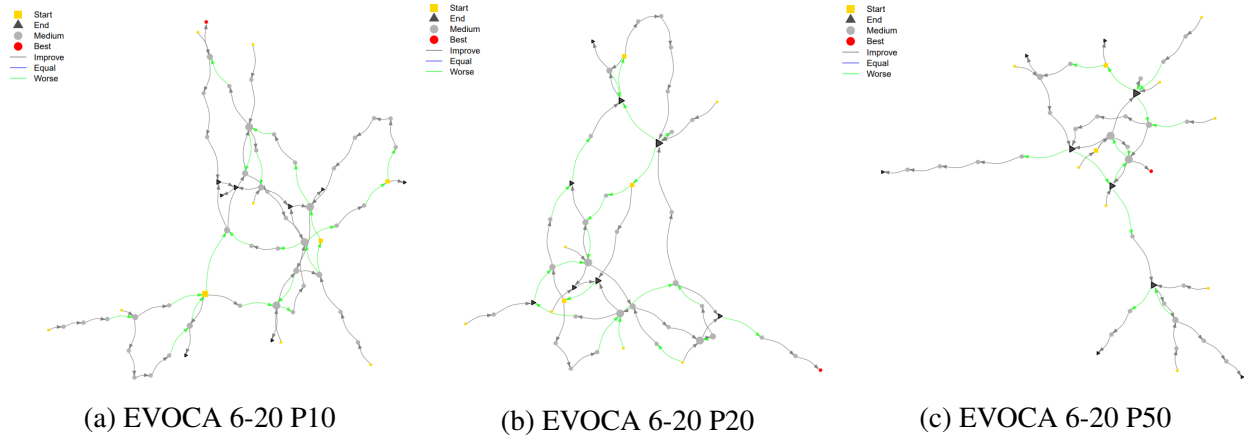


Figura 5.32: Redes trayectoriales de categoría 6_20 - tamaño locación 0.2x0.05x6

Las redes trayectoriales presentadas en la Figura 5.33 presentan el mismo comportamiento descrito, entre 9 y 10 componentes conexas una única mejor solución encontrada por cada red, con un único camino a dicha solución, exceptuando a la red con población inicial 10 (Figura+5.34a), debido a que existen 2 caminos para llegar al óptimo encontrado.

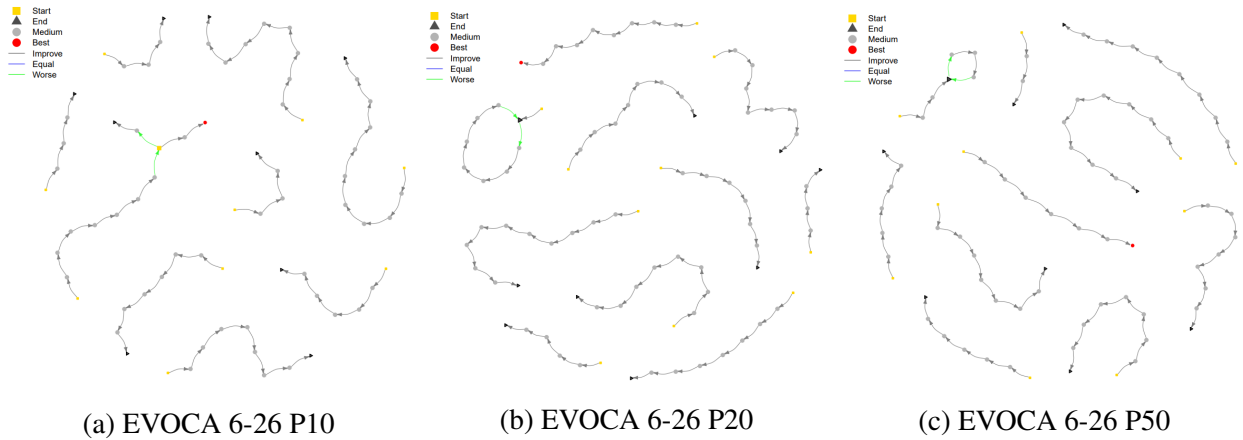


Figura 5.33: Redes trayectoriales de categoría 6_26 - tamaño locación 0.01x0.01x1

Al aproximar las locaciones para $N=6$ y $K=26$ se tienen las gráficas de la Figura 5.34 donde se destaca que en la red con p_s igual a 10 se tienen 2 componentes conexas, lo que indica que se exploraron más zonas “lejanas” del espacio de búsqueda que a lo que se acostumbra normalmente. Esto en base a lo visto en las demás redes. Por otro lado, en la red con mayor población (50) se tiene un total de 16 caminos a la zona en que se encuentra la solución óptima, lo que indica que se tendió a intensificar la búsqueda en zonas específicas del espacio de búsqueda.

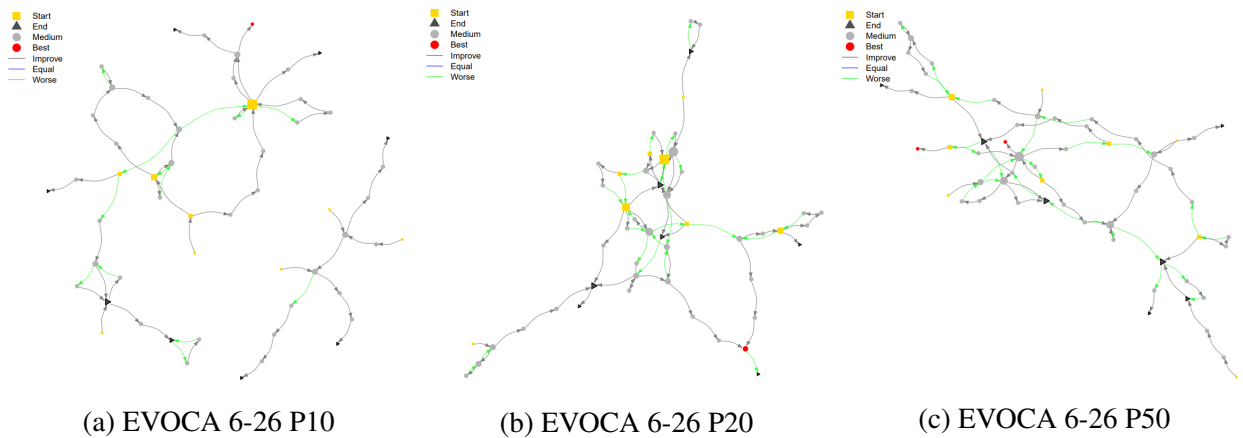


Figura 5.34: Redes trayectoriales de categoría 6_26 - tamaño locación 0.2x0.05x6

A continuación en la Figura 5.36 se presentan las redes trayectoriales correspondientes a las instancias con K igual a 6 y N igual a 32. Todas presentan 10 componentes conexas (cada una por cada ejecución independiente realizada). Aquella que encuentra el óptimo más rápido corresponde

a la con p_s igual a 10, puntuando un camino de largo 5.0, seguida de la red con tamaño de población 20 y 50 con 7.0 y 10.0 respectivamente. Todas con un único camino hacia la mejor solución encontrada.

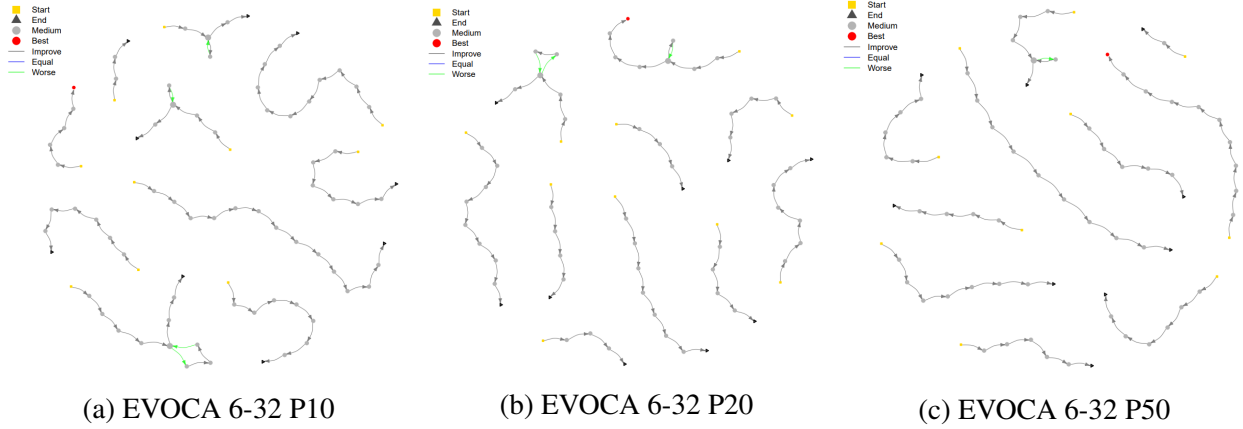


Figura 5.35: Redes trayectoriales de categoría 6_32 - tamaño locación 0.01x0.01x1

Tras aproximar la locación de las redes con $N=6$ y $K=32$, la cantidad de caminos aumentó a 8 para las poblaciones 10 y 20, mientras que para la población 50 aumentó a 10. El camino promedio para llegar a la mejor solución encontrada para p_s 20 y 50 se redujo de 7.0 a 3.1 y 10.0 a 5.8, lo que indica que en estos algoritmos se intensificó en zonas específicas del espacio de búsqueda al tratar con soluciones cercanas que terminaron siendo representadas por un único nodo o locación.

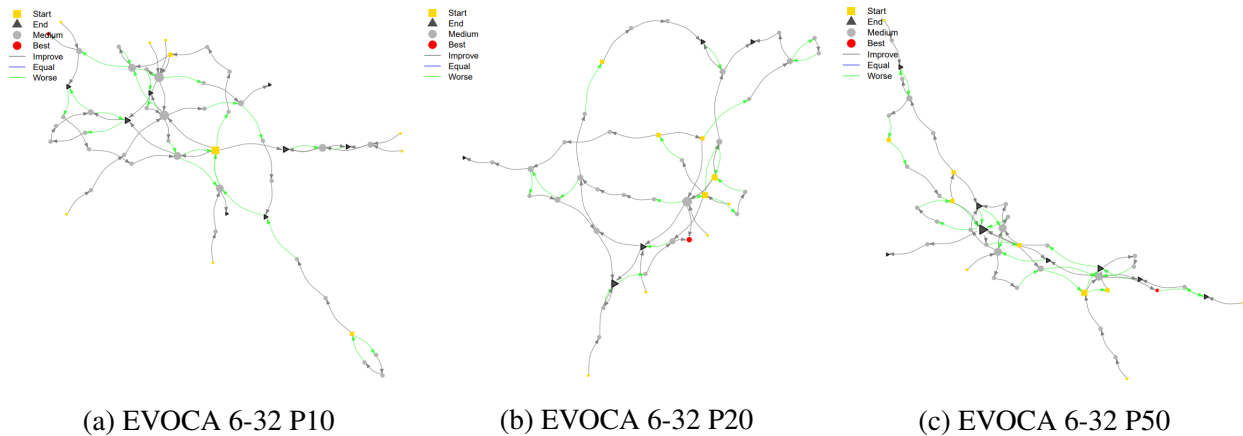


Figura 5.36: Redes trayectoriales de categoría 6_32 - tamaño locación 0.2x0.05x6

En resumen, en las redes no aproximadas que se muestran en las Figuras 5.7, 5.9, 5.11, 5.13, 5.15, 5.19, 5.21, 5.23, 5.25, 5.27, 5.29, 5.31, 5.33 y 5.35 se observa que todos los nodos de inicio

están separados, demostrando las características de las diferentes soluciones iniciales que genera el sintonizador. En cada uno de los casos se puede también reconocer la mejor configuración y cómo es alcanzada por, en su mayoría, una única trayectoria del proceso.

Al visualizar las redes aproximadas de las Figuras 5.8, 5.10, 5.12, 5.14, 5.16, 5.16, 5.20, 5.22, 5.24, 5.26, 5.28, 5.30, 5.32, 5.34 y 5.36, las redes se encuentran altamente más conectadas. En su mayoría se tiende a intensificar con mayor intensidad para la población de tamaño 50, esto puede deberse a que en Evoca la mutación es la encargada de la exploración, donde el cambio que se aplica a las configuraciones/soluciones es mínimo y el hecho de que la red traza la trayectoria utilizando únicamente al mejor de cada iteración hace que sea alta la posibilidad de que dichas soluciones sean similares entre sí, haciendo coherente la alta intensificación en la región.

Conclusiones

El problema que abarcan los sintonizadores de parámetros dentro del área de la optimización metaheurística es de suma importancia debido a que los algoritmos que resuelven estos problemas pueden contar con una gran cantidad de parámetros, interrelaciones entre estos, etc, en consecuencia, encontrar aquellas configuraciones o valores para parámetros que obtengan un buen desempeño del algoritmo objetivo es un proceso complejo.

En esta memoria se propuso un enfoque basado en la integración de redes trayectoriales al algoritmo sintonizador de parámetros Evoca con la finalidad de analizar el espacio de parámetros en base a la ruta trazada. Al tratarse de un sintonizador basado en un algoritmo genético, se eligió un representante por cada iteración de Evoca. Luego se tuvo que hacer una reparación de los datos obtenidos tras la ejecución del sintonizador debido a que la naturaleza estocástica provocaba tener distintas calidades para una misma configuración, es por ello que se acordó trabajar con el promedio para todos los valores de una configuración de parámetros al momento de armar el input esperado por los scripts que armaban las STNs.

En específico, se buscó comprender el desplazamiento que realiza el sintonizador a través del espacio de parámetros o espacio de búsqueda y cómo evoluciona a lo largo del tiempo. Respecto a este punto, en el capítulo anterior se dió a conocer que el algoritmo tiende a explorar el espacio mediante trayectorias lineales e independientes (considerando las diferentes ejecuciones por categoría). Por otro lado, en la mayoría de las categorías se notó la tendencia a aumentar la cantidad de nodos (configuraciones de parámetros) en común a medida que las ejecuciones aumentaban.

Los experimentos realizados por el sintonizador Evoca permitieron realizar comparaciones bajo distintas dificultades. Donde se apreció que Evoca presenta un fuerte énfasis tanto en la explotación

como exploración del espacio de búsqueda. En el caso de explotar se evidencia el comportamiento de un algoritmo con un fuerte énfasis en intensificar. Mientras que al aproximar las locaciones se puede analizar de mejor manera las “zonas” más visitadas y en que momento en que ocurre (ejecución). Detectándose una inclinación a explorar el espacio de búsqueda, es decir, diversificar. Con todo lo mencionado anteriormente, queda claro que el uso de herramientas visuales como redes trayectoriales ayuda a entender y analizar el proceso de sintonización de parámetros.

5.3.1. Trabajo Futuro

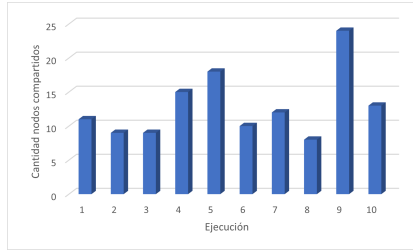
Respecto al proceso de sintonización, se propone aumentar la cantidad de parámetros por escenario y tratar con parámetros de distinta naturaleza. Respecto al sintonizador, realizar modificaciones en los movimientos que toma para determinar las configuraciones visitadas dentro del espacio de búsqueda con la finalidad de reproducir los experimentos para ver como afectan los movimientos al comportamiento del algoritmo.

Bibliografía

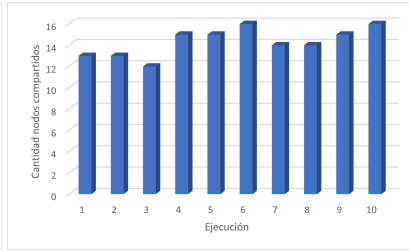
- [1] Bertrand Neveu Elizabeth Montero, María Cristina Riff. A beginner's guide to tuning methods. Applied Soft Computing, 17:39–51, 2014.
- [2] María Cristina Riff Elizabeth Montero. A new algorithm for reducing metaheuristic design effort. In IEEE Congress on Evolutionary Computation, 2013.
- [3] Holger H. Hoos Frank Hutter and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. Applied Soft Computing, 2011.
- [4] Kevin Leyton-Brown Frank Hutter, Holger H. Hoos. Paramils: An automatic algorithm configuration framework. Journal of Artificial Intelligence Research, 36:267–306, 2009.
- [5] Christian Blum Gabriela Ochoa, Katherine M. Malan. Search trajectory networks: A tool for analysing and visualising the behaviour of metaheuristics. Applied Soft Computing, 2011.
- [6] F. Glover. Future paths for integer programming and links to artificial intelligence. pages 5. 533–549., 1986.
- [7] Holger H. Hoos Thomas Stützle Leslie Pérez Cáceres, Manuel López-Ibáñez. An experimental study of adaptive capping in irace. 10556:235–250, 2017.
- [8] Leslie Pérez Cáceres Mauro Birattari Thomas Stützle Manuel López-Ibáñez a, Jérémie Dubois-Lacoste. The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives, 3:43–58, 2016.
- [9] David E. Goldberg Martin Pelikan and Fernando G. Lobo. A survey of optimization by building and using probabilistic models. 2002.
- [10] M. D. McKay, R. J. Beckman, and W. J. Conover. Comparison the three methods for selecting values of input variable in the analysis of output from a computer code. Technometrics; (United States), 21:2, 5 1979.
- [11] Carlos Ignacio Hernandez Castellanos Carlos A. Coello Coello Miriam Pescador-Rojas, Denis Pallez. Studying the effect of robustness measures in offline parameter tuning for estimating the performance of moea/d. pages 204–211, 2018.

- [12] Julia Gastinger Tobias Jacobs Sebastien Nicolas Anett Schülke Mischa Schmidt, Shahd Sa-farani. On the performance of differential evolution for hyperparameter tuning. pages 1–8, 2019.
- [13] Volker Nannen and Agoston E Eiben. Efficient relevance estimation and value calibration of evolutionary algorithm parameters. 2007.
- [14] M. E. J Newman. The structure and function of complex networks. 45(2):167–256, 2003.
- [15] Bo Guo Rui Wang, Tao Zhang. An enhanced moea/d using uniform directions and a pre-organization procedure. 2013 IEEE Congress on Evolutionary Computation, CEC 2013, pages 2390–2397, 06 2013.
- [16] Wolfgang Banzhaf Ting Hu, Gabriela Ochoa. Phenotype search trajectory networks. 11 2022.
- [17] Claus Aranha Yuri Lavinas and Gabriela Ochoa. Search trajectories networks of multiobjective evolutionary algorithms. 2022.
- [18] Gabriela Ochoa Claus Arantha Yuri Lavinas, Marcelo Ladeira. Component-wise analysis of automatically designed multiobjective algorithms on constrained problems. 2022.

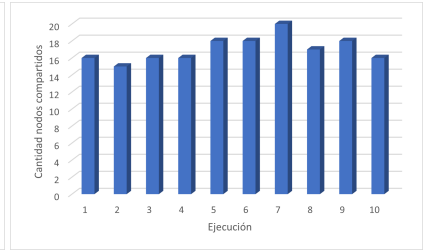
Anexo



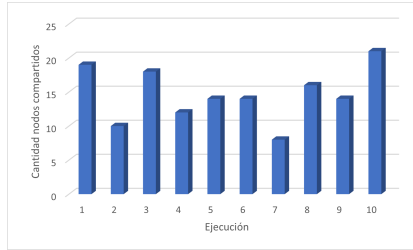
(a) EVOCA 2-20



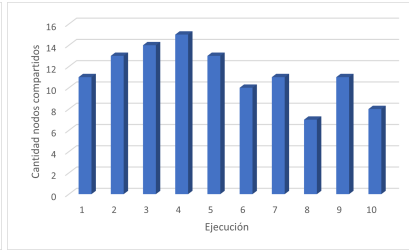
(b) EVOCA 2-38



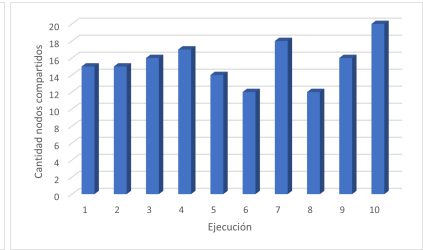
(c) EVOCA 2-52



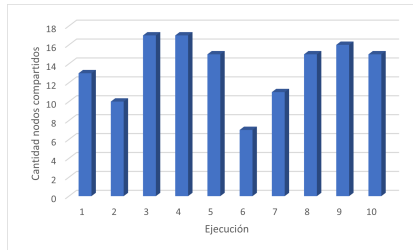
(d) EVOCA 3-20



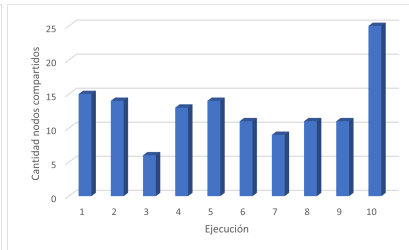
(e) EVOCA 3-34



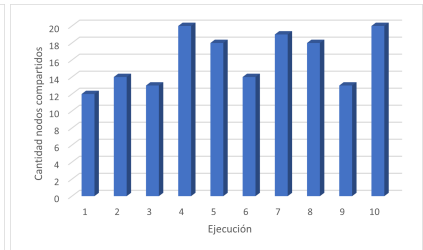
(f) EVOCA 3-48



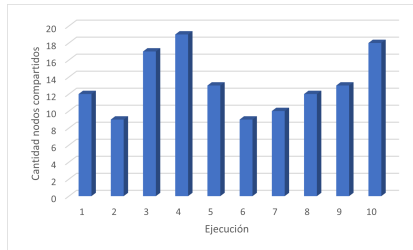
(g) EVOCA 4-20



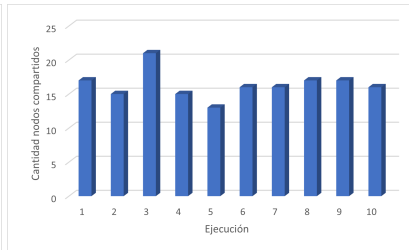
(h) EVOCA 4-30



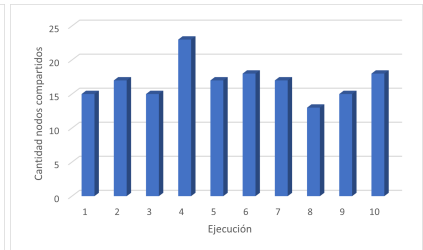
(i) EVOCA 4-40



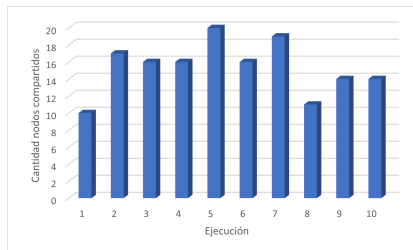
(j) EVOCA 5-20



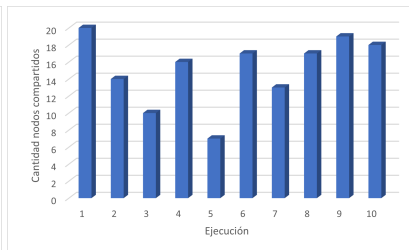
(k) EVOCA 5-28



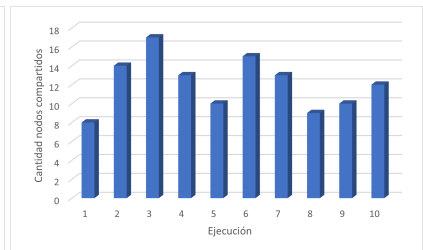
(l) EVOCA 5-38



(m) EVOCA 6-20



(n) EVOCA 6-26



(ñ) EVOCA 6-32

Figura 5.37: Nodos compartidos por ejecución - tamaño locación 0.2 x 0.05 x 6