eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Grado en Ingeniería Informática

Computación

Trabajo de Fin de Grado

Resolviendo el problema de asignación cuadrática mediante su descomposición en landscapes elementales

Autor

Xabier Benavides

2021

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Grado en Ingeniería Informática

Computación

Trabajo de Fin de Grado

Resolviendo el problema de asignación cuadrática mediante su descomposición en landscapes elementales

Autor

Xabier Benavides

Directores

Josu Ceberio, Leticia Hernando

Agradecimientos

Quiero dar las gracias a mis directores de proyecto Josu y Leticia por confiar en mí y ayudarme siempre que lo he necesitado. Espero seguir trabajando con vosotros en un futuro, ya que me habéis demostrado ser investigadores realmente capaces y, sobre todo, unas muy buenas personas. En estos meses he aprendido muchísimo, y esto se debe principalmente a vosotros.

Agradecer también el apoyo de mi familia, puesto que siempre ha estado dispuesta a echarme una mano. A mi pareja, por animarme en todo momento y alegrarme incluso en los peores días. Sin todos vosotros este proyecto no hubiese sido posible. Muchas gracias.

Índice general

Índice de figuras

Índice de tablas

1. CAPÍTULO

Introducción

El problema de asignación cuadrática (QAP) [1] es uno de los problemas de optimización combinatoria (COP) [2][3] más estudiados en la ciencia de la computación. Su planteamiento es simple: imaginemos que tenemos n fábricas que debemos asignar a n localizaciones distintas. Las fábricas tendrán un flujo de trabajo entre ellas (transporte de mercancías, personal...) que implica unos costes de planificación, logística y movilidad que hay que tener en cuenta. Debido a ello, nuestra meta será encontrar la asignación de fábricas a emplazamientos que minimice dichos costes. Todas las instancias de este problema tienen, por lo tanto, dos componentes principales: una matriz de distancias entre las localizaciones ($D_{n\times n}$) y una matriz de flujo de trabajo entre las fábricas ($H_{n\times n}$). La información de dichas matrices se tiene en cuenta en la función objetivo del problema:

$$f(x) = \sum_{i,j=1}^{n} d_{i,j} h_{x(i),x(j)} \tag{1.1}$$

donde $x(i)$ es la fábrica asignada a la localización i, $d_{i,j}$ es la distancia entre las localizaciones i y j, y $h_{p,q}$ es el flujo de trabajo entre las fábricas p y q. Dado que la función objetivo mide el coste de una asignación concreta, nuestro objetivo será encontrar la solución que minimice dicha función.

Aunque el QAP fue originalmente propuesto en el ámbito de la planificación logística de plantas de producción, su relevancia no se reduce tan solo a dicho ámbito. En [4], por ejemplo, se propone utilizar el QAP para configurar el diseño de un hospital de manera que las esperas y retrasos derivados de los desplazamientos entre consultas disminuyan.

En [5], por otro lado, se adopta un enfoque similar al QAP para encontrar la configuración de teclas más eficiente en el teclado de una máquina de escribir. Otros ejemplos de sus áreas de aplicación son el diseño del cableado de tableros eléctricos [6] y la planificación de líneas de producción paralelas [7]. Toda esta variedad de utilidades hace que el problema de asignación cuadrática sea uno de los problemas más importantes en lo que a optimización combinatoria se refiere, convirtiéndolo en un candidato ideal para este proyecto.

Al igual que muchos otros problemas similares, el QAP entra dentro de la clase *NP-hard* [8], por lo que no se conoce ningún algoritmo que sea capaz de resolverlo de forma exacta en un tiempo polinómico respecto a la dimensión del problema. Esto no es un inconveniente cuando trabajamos con instancias pequeñas ($n < 20$), pero a partir de cierto tamaño los algoritmos de búsqueda exhaustiva dejan de ser una opción viable. Por esta razón, la comunidad científica ha propuesto una gran variedad de métodos heurísticos y meta-heurísticos [9] que, aunque no aseguran obtener la solución óptima, permiten realizar aproximaciones razonablemente buenas en un tiempo asumible: búsquedas locales [10], algoritmos evolutivos [11]... Poco a poco, estos métodos han ido reemplazando a los algoritmos clásicos de la investigación operativa, convirtiendo a los algoritmos heurísticos en una alternativa simple y eficaz para la resolución de problemas de optimización combinatoria tales como el QAP.

No obstante, en ocasiones los métodos heurísticos convencionales no son suficientes para hallar resultados satisfactorios, por lo que es necesario considerar técnicas complementarias que permitan mejorar su rendimiento. Si bien existen muchas técnicas de este tipo, un ejemplo que nos interesa especialmente es la descomposición de la función objetivo del problema en varias sub-funciones independientes. Este enfoque fue propuesto en el ámbito de la multi-objetivización de problemas de un solo objetivo [12], y permite aprovechar las propiedades de las sub-funciones para aumentar la diversidad de la búsqueda y evitar caer en óptimos locales. Entre todas las técnicas de descomposición conocidas, en este proyecto nos centraremos en estudiar la descomposición en *landscapes* elementales (ELD) [13] aplicada al problema de asignación cuadrática [14].

Tomando todo esto en cuenta, el trabajo a realizar se divide en tres apartados diferenciados que se llevarán a cabo de forma secuencial:

1. Primero, se analizarán las propiedades de la descomposición en *landscapes* elementales del QAP en un *benchmark* de instancias. La meta de este primer apartado es entender mejor los distintos *landscapes* elementales y extraer patrones en función

de las características de las instancias que puedan ser de utilidad para su posterior optimización. Nos centraremos principalmente en dos aspectos:

- La complejidad de los *landscapes* medida en función del número de óptimos locales.
- La relación entre los *landscapes* de la descomposición y el *landscape* original del QAP.

Además del análisis experimental, también incluiremos desarrollos teóricos para demostrar formalmente algunas de las propiedades observadas.

2. En el segundo apartado, usaremos los conocimientos adquiridos para diseñar algoritmos heurísticos especializados que optimicen el QAP. Diseñaremos dos algoritmos distintos, uno basado en la búsqueda local y un algoritmo evolutivo.

3. Finalmente, compararemos los algoritmos diseñados y realizaremos un análisis estadístico para comprobar cual de los métodos presenta unos resultados más prometedores en un *benchmark* de instancias.

En base a los resultados del proyecto, se establecerán los pasos a seguir en un futuro para continuar investigando el método de descomposición en *landscapes* elementales como herramienta para resolver problemas de optimización combinatoria.

2. CAPÍTULO

Planificación y gestión del proyecto

En este capítulo se presenta la planificación y gestión prevista para el proyecto. Además, se proporcionan detalles sobre la consecución del mismo una vez ejecutado. Dado que este proyecto tiene un carácter marcadamente científico y de investigación, los procedimientos y objetivos planeados pueden sufrir modificaciones.

2.1. Objetivos del proyecto

El presente proyecto se fundamenta en la consecución de los siguientes objetivos y subobjetivos:

1. Analizar y estudiar la descomposición en *landscapes* elementales del QAP en un *benchmark* de instancias.
    a) Estimar la cantidad de óptimos locales en cada *landscape* de la descomposición.
    b) Cuantificar la relación entre los *landscapes* de la descomposición y el *landscape* original del QAP.
2. En base al conocimiento generado en las tareas realizadas para el primer objetivo, diseñar e implementar dos algoritmos heurísticos para la resolución del QAP.
    a) Diseñar e implementar un algoritmo basado en la búsqueda local.

   b) Diseñar e implementar un algoritmo evolutivo.

3. Medir y comparar el rendimiento de los algoritmos propuestos en un *benchmark* de instancias.

   a) Comprobar si los algoritmos propuestos mejoran a un algoritmo *baseline*.

   b) Realizar un análisis estadístico.

2.2. Estructura del proyecto

Para facilitar la gestión del proyecto, las tareas a realizar se han distribuido en varios paquetes de trabajo que se definen a continuación. La organización propuesta para el proyecto se puede observar de forma más visual en la estructura de descomposición de trabajo (Figura 2.1).

1. **Planificación y seguimiento:** Incluye la planificación del proyecto junto con el seguimiento y control que se realizará durante el desarrollo del mismo. Las reuniones semanales que se llevarán a cabo con los directores del proyecto son parte de este paquete de trabajo.

2. **Conocimientos previos:** Comprende el estudio inicial de los conceptos básicos necesarios para la correcta comprensión del proyecto: optimización combinatoria [2][3], problema de asignación cuadrática [1], *landscapes* elementales [15][16][17], descomposición en *landscapes* elementales [13][14] y estimación de óptimos locales [18].

3. **Estudio del ELD:** Este paquete de trabajo se divide en dos sub-apartados relacionados con el análisis de la descomposición en *landscapes* elementales del QAP:

   a) **Experimentación:** Diseño y realización de los experimentos necesarios para analizar los *landscapes* de la descomposición en un *benchmark* de instancias.

   b) **Análisis de resultados:** Estudio de los resultados obtenidos en la experimentación. Se compararán los resultados obtenidos en cada *landscape* elemental y se extraerán posibles patrones o características que resulten de utilidad.

4. **Desarrollo de algoritmos:** Incluye el diseño e implementación de dos algoritmos heurísticos especializados que resuelvan el QAP basándose en su descomposición en *landscapes* elementales. Este paquete de trabajo se divide en dos sub-apartados:

   a) **Búsqueda local:** Desarrollo de un algoritmo de búsqueda local.

   b) **Algoritmo evolutivo:** Desarrollo de un algoritmo evolutivo.

5. **Estudio comparativo:** Comprende la comparación de rendimiento entre los algoritmos desarrollados, así como las tareas a realizar para preparar y llevar a cabo la experimentación necesaria. Este paquete de trabajo se divide en dos sub-apartados:

   a) **Experimentación:** Diseño y realización de las pruebas en un *benchmark* de instancias.

   b) **Análisis de resultados:** Comparación y análisis estadístico de los resultados obtenidos en la experimentación.

6. **Memoria y presentación:** Abarca la redacción de la memoria del proyecto y la preparación de la presentación final del mismo.

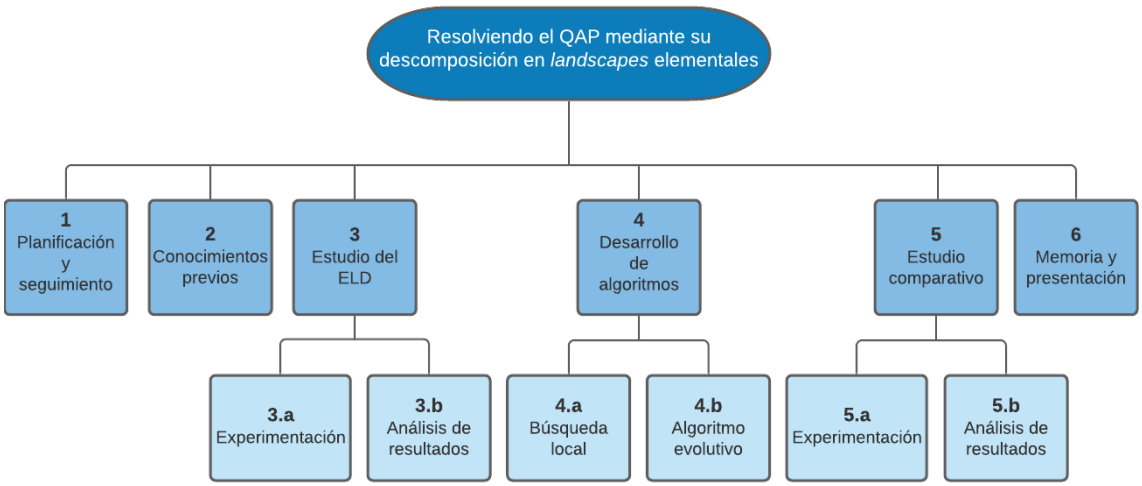


Figura 2.1: Estructura de descomposición de trabajo (WBS) del proyecto.

2.3. Planificación temporal

Una vez definidos los paquetes de trabajo que componen el proyecto, el siguiente paso es planificar cuánto tiempo vamos a dedicar a cada una de las tareas necesarias para

alcanzar los objetivos. Debido a que se trata de un proyecto basado enteramente en la investigación, se ha intentado realizar una planificación temporal flexible.

2.3.1. Dedicación estimada por paquetes de trabajo

A continuación se muestra la dedicación estimada para la realización del proyecto dividida por paquetes de trabajo. Es importante remarcar que estos tiempos son solo una estimación, lo que implica que en algunos casos podríamos exceder el tiempo previsto y en otros no llegar a cumplirlo. De todos modos, cualquier desviación excesiva debería ser inmediatamente corregida mediante los mecanismos de seguimiento y control.

1. **Planificación y seguimiento:** 20 horas (incluyendo las reuniones semanales).
2. **Conocimientos previos:** 50 horas.
3. **Estudio del ELD:** 40 horas (suma de los sub-apartados).
    a) **Experimentación:** 30 horas.
    b) **Análisis de resultados:** 10 horas.
4. **Desarrollo de algoritmos:** 100 horas (suma de los sub-apartados).
    a) **Búsqueda local:** 50 horas.
    b) **Algoritmo evolutivo:** 50 horas.
5. **Estudio comparativo:** 20 horas (suma de los sub-apartados).
    a) **Experimentación:** 10 horas.
    b) **Análisis de resultados:** 10 horas.
6. **Memoria y presentación:** 90 horas.

Además del tiempo previsto para cada apartado, se han reservado otras 30 horas que podrán ser repartidas entre aquellos paquetes de trabajo que lo necesiten. Contando con esto, se prevé una dedicación total de 350 horas. En la Figura 2.2 se muestra de forma más visual el porcentaje del proyecto estimado para cada uno de los paquetes de trabajo.

Figura 2.2: Distribución de la dedicación por paquetes de trabajo.

2.3.2. Calendario del proyecto

La duración del proyecto será de 7 meses, comenzando el 1 de julio de 2020 y finalizando el 31 de enero de 2021. En los dos primeros meses (julio, agosto) la dedicación al proyecto será a tiempo parcial, realizando la planificación y adquiriendo los conocimientos previos. A partir del tercer mes (septiembre) se comenzará a llevar a cabo el grueso del proyecto a tiempo completo. Finalmente, el último mes (enero) se reservará para finalizar la memoria y preparar la presentación final programada para la convocatoria ordinaria de febrero.

La organización temporal de las tareas a realizar se resume en el diagrama de Gantt que se muestra en la Figura 2.3. En dicho diagrama, la realización del proyecto se divide en periodos de aproximadamente dos semanas y se indican las tareas planificadas para cada uno de los periodos. Cada paquete de trabajo se presenta con un color diferente para facilitar la visualización.

Tal y como se ha mencionado anteriormente, se realizará una reunión semanal[1] con los directores del proyecto para afianzar el progreso realizado y planificar las tareas a completar en la siguientes semanas. Esta supervisión continuada permitirá ajustar el calendario descrito en el diagrama de Gantt de una forma rápida y eficaz siempre que sea necesario.

[1] A excepción del mes de agosto y las vacaciones de navidad.

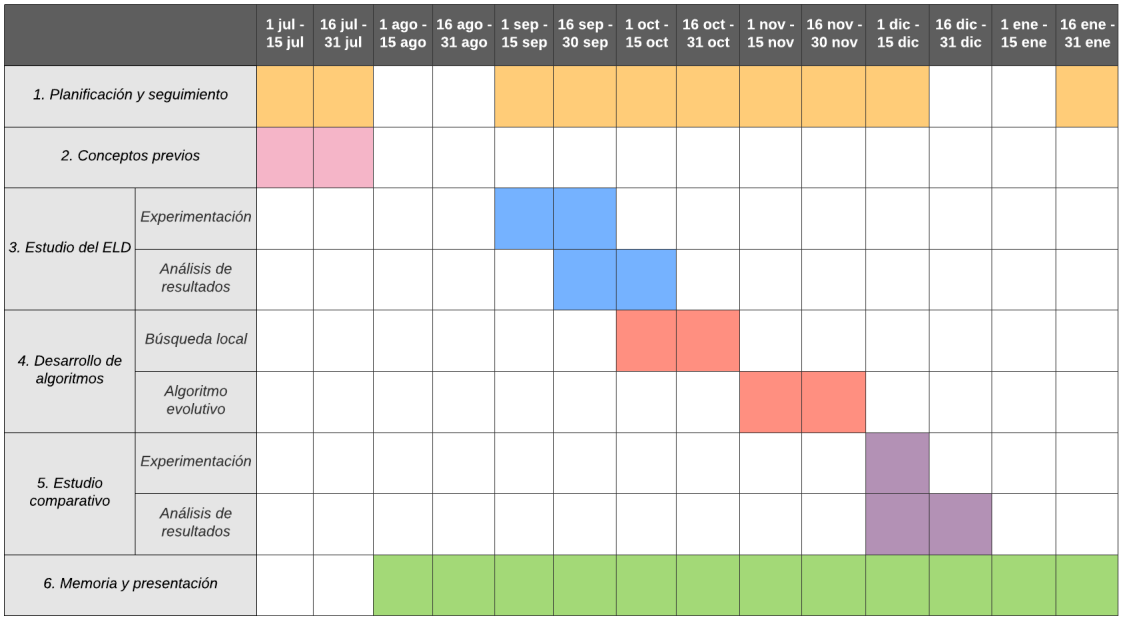


Figura 2.3: Diagrama de Gantt del proyecto.

2.4. Gestión de recursos

Para la consecución de los objetivos previstos en este proyecto, se utilizarán las herramientas tanto software como hardware enumeradas a continuación:

- **Software**
  - **Overleaf:** Editor de texto *online* basado en el sistema de composición de textos LaTeX. Se utilizará principalmente para la redacción de la memoria.
  - **Lucidchart:** Herramienta para la creación de diagramas en línea. Se usará para diseñar gráficos simples.
  - **Inkscape:** Editor de gráficos vectoriales que permite crear ilustraciones. Se utilizará para diseñar figuras complejas que requieran de un editor de imágenes.
  - **Atom:** Editor de código fuente con soporte para múltiples *plugins* y sistema de control de versiones Git integrado. Servirá para la redacción de código en lenguaje C.
  - **Jupyter Notebook:** Aplicación que proporciona un entorno de trabajo interactivo que facilita el desarrollo de código en varios lenguajes de progra-

mación. Además, también permite la inclusión de texto, figuras e incluso gráficos. Servirá para la redacción de código en lenguaje Python.

- **RStudio:** Entorno de desarrollo integrado para el lenguaje de programación R que incluye consola, editor de sintaxis y herramientas para la gestión del espacio de trabajo. Servirá para la redacción de código en lenguaje R.
- **Bitbucket:** Herramienta de gestión de código en línea que utiliza el sistema de control de versiones Git. Todo el código desarrollado para el proyecto se guardará en un repositorio de Bitbucket.
- **Dropbox:** Servicio de alojamiento de archivos multi-plataforma en la nube que permite almacenar, sincronizar y compartir archivos. Se crearán dos carpetas:
  - Una carpeta compartida entre alumno y profesores que servirá para intercambiar bibliografía o código cuando sea necesario.
  - Una carpeta personal que guardará una copia de seguridad de todo el trabajo realizado. Dicha carpeta se actualizará una vez a la semana.
- **LibreOffice Calc:** Aplicación de hojas de cálculo incluida en el paquete de software LibreOffice. Se utilizará para registrar los resultados de la experimentación de una forma más visual.

- **Hardware**

  - **Equipo propio:** Ordenador portátil que se utilizará para la realización del proyecto. Se trata de un Asus ROG Strix G15 G512LV-AL007 con 16GB de memoria RAM, 512GB de disco duro y un procesador Intel Core i7-10750H.
  - **Cluster de computadores:** Cluster perteneciente al grupo de investigación ISG que servirá para agilizar el proceso de experimentación. Dicho hardware cuenta con 800 unidades de cómputo (*cores*), 2.5TB de memoria RAM y 50TB de disco. Además, en la última actualización se han añadido varias unidades de procesamiento gráfico (GPU) apropiadas para realizar cómputos relacionados con *Machine Learning* y *Reinforcement Learning*.

Todos los recursos externos utilizados son gratuitos o de código libre, lo que implica que no será necesario ningún tipo de gasto relacionado con la compra de licencias software para la realización del proyecto.

2.5. Gestión de riesgos

El desarrollo de este proyecto conlleva una serie de riesgos que es importante tener en cuenta, ya que sus efectos pueden causar desde pequeños inconvenientes hasta situaciones en las que no sea posible continuar. Con el objetivo de evitar este tipo de circunstancias, se han identificado algunos de los riesgos más probables y se han preparado los correspondientes planes de contingencia:

- **Subestimar el tiempo de dedicación**
  - *Descripción:* Estimar menos horas de las realmente necesarias para la realización de alguno de los paquetes de trabajo.
  - *Impacto:* Proyecto inacabado o realizado de forma apresurada.
  - *Plan de contingencia:* Se han reservado 30 horas extra que podrán ser distribuidas entre todos los paquetes de trabajo. Si aún así algún apartado específico necesita más dedicación, se permitirá utilizar el tiempo sobrante de otros apartados siempre que no se superen las 350 horas totales.
- **Experimentación inicial no concluyente**
  - *Descripción:* Obtener resultados en la experimentación inicial que no muestren ventajas reales a la hora de utilizar la descomposición en *landscapes* elementales para resolver el QAP.
  - *Impacto:* Imposibilidad para diseñar métodos heurísticos que mejoren los resultados de los algoritmos del estado del arte.
  - *Plan de contingencia:* Se modificará el enfoque del proyecto para orientarlo hacia un marco más teórico. De esta forma, el diseño de los algoritmos solo servirá para comprender mejor la descomposición en *landscapes* elementales, dejando de lado las posibles aplicaciones prácticas de los métodos desarrollados.
- **Problemas en la experimentación debido a los recursos externos**
  - *Descripción:* Fallo en los recursos externos utilizados para agilizar los procesos de experimentación, ya sea por caídas del servidor o problemas de acceso desde el cliente.

  - *Impacto:* Retraso en la obtención de resultados en los procesos de experimentación.
  - *Plan de contingencia:* Durante el tiempo que el recurso externo se mantenga inaccesible se utilizará el equipo personal para avanzar con la experimentación. En caso de que la situación se alargue, se reorganizará el calendario en base a las nuevas circunstancias.

- **Pérdida del avance realizado**

  - *Descripción:* Perder los documentos o el código del proyecto de forma irreversible. Esto se puede deber, por ejemplo, a un fallo en el equipo personal utilizado para trabajar.
  - *Impacto:* Retroceso en el avance del proyecto que obliga a rehacer partes o incluso la totalidad del mismo.
  - *Plan de contingencia:* Se guardarán en todo momento dos copias del trabajo realizado, una en el equipo personal y otra en la nube. En caso de perder una de ellas, se procederá a realizar otra copia a la mayor brevedad posible.

2.6. Seguimiento y control

Tras definir y planificar todos los aspectos del proyecto a realizar, pasamos ahora a exponer las principales diferencias que han surgido durante el desarrollo del mismo. Esta sección ha sido redactada una vez finalizado el proyecto, por lo que el objetivo es comprobar el impacto de las modificaciones realizadas a la planificación inicial.

2.6.1. Objetivos del proyecto cumplidos

Todos los objetivos planteados inicialmente han sido abarcados en la realización del proyecto. Sin embargo, cabe destacar que ha habido varias modificaciones importantes:

- A la hora de realizar las experimentaciones no se ha podido considerar una muestra de instancias lo suficientemente grande, por lo que las conclusiones extraídas en algunos de los objetivos quedan sujetas a futuros análisis más completos.
- En lugar de desarrollar un nuevo algoritmo evolutivo hemos tomado un algoritmo ya existente y lo hemos modificado para adaptarlo al QAP.

2.6.2. Dedicación real por paquetes de trabajo

En la Tabla 2.1 se indican las horas de dedicación que se han necesitado para completar las tareas de cada paquete de trabajo y se comparan con la dedicación estimada al inicio del proyecto. Esta comparativa también se muestra de forma más visual en la Figura 2.4.

Tabla 2.1: Horas de dedicación estimadas y horas realmente dedicadas a cada paquete de trabajo. La dedicación total tomando en cuenta las 30 horas de libre asignación se indica entre paréntesis.

Paquete de trabajo		Estimado (horas)	Real (horas)	Diferencia
1. Planificación y seguimiento		20	26	+6
2. Conocimientos previos		50	42	-8
3. Estudio del ELD	Experimentación	30	50	+20
	Análisis de resultados	10	15	+5
4. Desarrollo de algoritmos	Búsqueda local	50	51	+1
	Algoritmo evolutivo	50	72	+22
5. Estudio comparativo	Experimentación	10	11	+1
	Análisis de resultados	10	8	-2
6. Memoria y presentación		90	97	+7
TOTAL		**320 (350)**	**372**	**+52 (+22)**

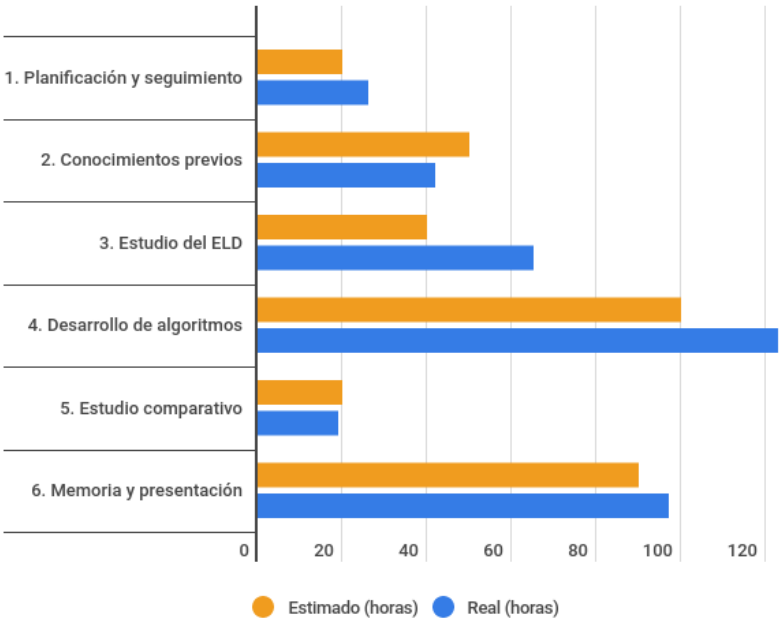


Figura 2.4: Comparativa entre la dedicación estimada (naranja) y la dedicación real (azul).

Como se puede observar, finalmente hemos necesitado 372 horas para completar el proyecto, lo que se traduce en 22 horas más de las previstas aún contando con las 30 horas de libre asignación. Esto se debe principalmente al estudio del ELD y al desarrollo de

los algoritmos, ya que entre ambos paquetes de trabajo hemos necesitado casi 50 horas más de lo previsto. Dado que estos dos apartados son el núcleo del proyecto, no nos ha importado aumentar su tiempo de dedicación aún a riesgo de no cumplir la planificación inicial. En cuanto al resto de paquetes de trabajo, no ha habido desviaciones demasiado significativas.

3. CAPÍTULO

Conceptos y conocimientos previos

Antes de comenzar con el proyecto en sí, es importante entender y comprender varios conceptos que serán de vital importancia durante el desarrollo del mismo. A continuación se proporciona una explicación resumida de cada uno de ellos.

3.1. Optimización combinatoria

La optimización combinatoria [2][3] es una rama de la optimización que agrupa la resolución de problemas cuyo objetivo es encontrar la mejor configuración posible para un conjunto de variables discretas en base a uno o más objetivos. Los problemas de optimización combinatoria (COP) se definen teniendo en cuenta dos conceptos: el espacio de búsqueda (Ω) y la función objetivo (f).

- **Espacio de búsqueda (Ω):** El espacio de búsqueda (también conocido como espacio de soluciones) es el conjunto de todas las soluciones posibles que satisfacen las restricciones de un problema. En los problemas de optimización combinatoria este conjunto debe ser discreto, pudiendo ser finito o infinito numerable.

- **Función objetivo (f):** La función objetivo es una función $f : \Omega \to \mathbb{R}$ que cuantifica la calidad de las soluciones en el espacio de búsqueda. El valor de la función objetivo para una solución concreta se denomina *fitness* de la solución, y se trata del valor que se busca minimizar o maximizar según el propósito del problema.

Dadas dos soluciones $x_1, x_2 \in \Omega$, diremos que x_1 es mejor que x_2 en base a f si se cumple lo siguiente:

- *Minimización:* $f(x_1) < f(x_2)$
- *Maximización:* $f(x_1) > f(x_2)$

Los problemas de optimización combinatoria pueden constar de una o varias funciones objetivo a optimizar. En este trabajo, nos centraremos en los problemas que se definen mediante una única función objetivo (también conocidos como problemas mono-objetivo).

En resumen, a la hora de resolver un problema de optimización combinatoria la meta es encontrar los óptimos globales del problema, es decir, aquellas soluciones en Ω que optimicen la función objetivo f. Dado un problema de minimización[1], se considerará óptimo global a toda solución $x^* \in \Omega$ que cumpla $f(x^*) = \min_{x \in \Omega} f(x)$.

Si bien algunos problemas de optimización combinatoria pueden ser resueltos en un tiempo polinómico respecto a la dimensión del problema, muchos de ellos pertenecen a la clase de complejidad *NP-hard* [8], lo que implica que no existen algoritmos exactos que puedan resolver todas sus instancias en un tiempo polinómico (suponiendo que $P \neq NP$). Algunos ejemplos clásicos de problemas de este tipo son el problema de ordenamiento lineal (LOP) [19], el problema de la mochila [20] o el problema de asignación cuadrática (QAP) [1], entre otros. Es precisamente este último el que vamos a estudiar en profundidad en este proyecto.

3.1.1. Problema de asignación cuadrática

El problema de asignación cuadrática (QAP) es un problema de optimización combinatoria que fue introducido por Koopmans y Beckmann en 1955 [1] como un modelo matemático que describe la manera de asignar un conjunto de actividades económicas indivisibles a un conjunto de localizaciones disponibles minimizando los costes derivados. El planteamiento del QAP nació de la necesidad de modelar de forma más precisa algunas situaciones de la vida real que no podían ser modeladas mediante el problema de asignación lineal (LAP) [21]. Debido a las similitudes entre ambos problemas, el QAP se puede considerar una extensión del LAP.

[1] Cualquier problema de maximización puede transformarse fácilmente en un problema de minimización multiplicando el valor de la función objetivo por -1.

De forma resumida, el LAP consiste en asignar n tareas a n recursos disponibles, teniendo en cuenta que una tarea solo puede ser asignada a un único recurso y viceversa. Cada posible asignación de una tarea p a un recurso i tiene un coste asociado $a_{p,i}$ que se guarda en la matriz $A_{n\times n}$. El objetivo es obtener la configuración que minimice el sumatorio del coste de todas las asignaciones realizadas, lo cual se calcula mediante la función:

$$f(x) = \sum_{i=1}^{n} a_{x(i),i} \tag{3.1}$$

donde $x(i)$ es la tarea asignada al recurso i. Una de las ventajas de este planteamiento es que puede ser resuelto en un tiempo polinómico respecto a la dimensión del problema. Sin embargo, el LAP asume que las asignaciones realizadas son independientes unas de otras, lo que en muchos casos se trata de una idealización que no se corresponde con la realidad. En dichas situaciones, es necesario utilizar el QAP para modelar el problema.

Tal y como se ha mencionado anteriormente, el QAP fue propuesto en el contexto de la planificación logística de plantas de producción. Este problema toma como punto de partida un conjunto de n plantas de producción indivisibles y un conjunto de n localizaciones físicas a las que deben ser asignadas. Cada planta p tiene un coste de construcción distinto para cada posible localización i denominado $b_{p,i}$, el cual se especifica en una matriz $B_{n\times n}$. Si bien hasta este punto el planteamiento es idéntico al mostrado en el LAP, todo cambia cuando tenemos en cuenta que las plantas de producción deben comunicarse entre ellas, ya sea para compartir recursos, servicios, personal... Por lo que además de tener en cuenta los costes de construcción, también debemos considerar los gastos que se producen por la interacción entre las plantas. Por ello, es necesario definir la cantidad de flujo de trabajo $h_{p,q}$ que hay entre cada planta p y cada planta q en una matriz $H_{n\times n}$. Cuanto más flujo de trabajo entre dos plantas, más cerca deberán ser construidas para minimizar los costes que conlleva dicha comunicación, por lo que otro factor a tener en cuenta es cuán lejos están unas localizaciones de otras. Esto nos lleva a considerar una última matriz $D_{n\times n}$ que indica la distancia $d_{i,j}$ que hay entre cada localización i y cada localización j. El gasto derivado de la interacción entre dos plantas se calcula, por lo tanto, como la multiplicación entre el flujo de trabajo entre ellas y la distancia que las separa. En base a esto, el coste total asociado a cada posible configuración se mide mediante la función objetivo:

$$f(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} d_{i,j}h_{x(i),x(j)} + \sum_{i=1}^{n} b_{x(i),i} \tag{3.2}$$

donde $x(i)$ es la planta de producción asignada a la localización i. Como se puede observar, la función objetivo se puede dividir en dos partes: la parte asociada al coste derivado de la comunicación entre las plantas y la parte asociada al coste de construcción de las mismas. La segunda parte es completamente equivalente a un LAP, y por lo tanto, no será objeto de estudio en este proyecto. Por este motivo, a partir de ahora dicha parte será omitida, y consideraremos la función objetivo simplificada:

$$f(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{i,j} h_{x(i),x(j)} \tag{3.3}$$

donde solo se tienen en cuenta los costes asociados al flujo de trabajo entre plantas. El objetivo principal de esta versión simplificada del QAP es asignar las plantas de producción a las localizaciones disponibles de forma que el coste que se deriva de la comunicación entre las mismas sea minimizado. De forma similar a lo visto en el LAP, este planteamiento presenta varias restricciones que condicionan las posibles soluciones al problema:

- Cada planta de producción debe ser asignada a una única localización. Además, debido a su naturaleza indivisible, no es posible asignar fracciones de plantas.
- A cada localización se le debe asignar una única planta de producción.

Debido a estas restricciones, el espacio de búsqueda de un QAP con n plantas y n localizaciones se describe como el conjunto de todas las permutaciones x de tamaño n, donde el valor de $x(i)$ se corresponde con la planta asignada a la localización i. En la Figura 3.1 se ilustra un ejemplo de un QAP con $n = 4$ y una posible solución para el mismo.

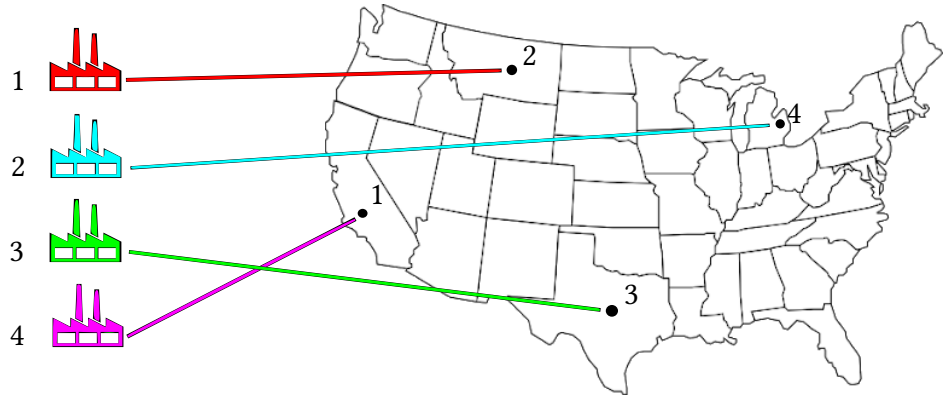


Figura 3.1: Ejemplo de una posible solución para un QAP con $n = 4$. La asignación de plantas de producción a localizaciones se codifica mediante la permutación $x = (4132)$.

A pesar de que hasta ahora solo hemos hablado de la planificación logística de plantas de producción, esta no es ni mucho menos la única área en la que podemos encontrar el QAP. De hecho, algunos de los problemas de optimización combinatoria más conocidos pueden ser expresados como casos particulares de este problema, lo que le hace adquirir una importancia especial. Uno de los ejemplos más claros es el problema del vendedor viajero (TSP) [22], el cuál ha sido ampliamente estudiado desde el siglo XIX. En este problema, se dispone de n ciudades que se han de visitar pasando una vez por cada una de ellas antes de volver a la ciudad de inicio. El objetivo es, por lo tanto, minimizar la distancia recorrida a la hora de realizar dicho tour entre ciudades. Para ello, se considera una matriz $W_{n\times n}$ que guarda la distancia $w_{p,q}$ que hay entre cada ciudad p y cada ciudad q, la cual se utiliza en la función objetivo que se busca minimizar:

$$f(x) = \sum_{i=1}^{n-1} w_{x(i),x(i+1)} + w_{x(n),x(1)} \tag{3.4}$$

donde $x(i)$ es la ciudad que se visita en la posición i del recorrido. Aunque en un principio el TSP y el QAP parezcan problemas completamente distintos, en realidad el primero es un caso particular del segundo si tenemos en cuenta lo siguiente:

- La matriz de flujo $H_{n\times n}$ del QAP se sustituye por la matriz de distancia $W_{n\times n}$.
- La matriz de distancia $D_{n\times n}$ del QAP se sustituye por la matriz de adyacencia $R_{n\times n}$ de un ciclo Hamiltoniano dirigido cuyos valores se definen a continuación:

$$r_{ij} = \begin{cases} 1 & \text{si } j \equiv (i \; mod \; n) + 1 \\ 0 & \text{en otro caso} \end{cases} \tag{3.5}$$

  Como se puede observar, la matriz de adyacencia está compuesta íntegramente de ceros excepto en las posiciones que representan la arista que va desde un nodo (i) al siguiente ($i+1$) [2]. Esto nos indica que la función objetivo solo tiene en cuenta la distancia recorrida desde cada ciudad a la inmediatamente posterior en el tour.

En base a esto, la función objetivo del TSP se puede reescribir de la siguiente forma:

$$f(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} r_{i,j} w_{x(i),x(j)} \tag{3.6}$$

[2] Excepto cuando $i = n$. En ese caso, se considera la arista que va desde el nodo n al nodo 1.

lo cual es un caso particular del QAP. De todos modos, este tan solo es un ejemplo, ya que hay muchos problemas de optimización combinatoria que no son más que reinterpretaciones del problema de asignación cuadrática. Esto nos permite extrapolar las conclusiones a las que lleguemos en este proyecto a una gran variedad de ámbitos, tales como el diseño de la distribución de instalaciones [4], la configuración de teclados [5], el cableado de tableros eléctricos [6] o la planificación de líneas de producción paralelas [7].

3.2. Landscapes

El concepto de *landscape* [23][24] en el ámbito de la optimización combinatoria se define como una tripleta (Ω, N, f) donde Ω es el espacio de búsqueda, f es la función objetivo que asigna un *fitness* a cada solución y N es la función de vecindario que conecta las soluciones del espacio de búsqueda en una estructura de vecindad. De forma más concreta, se denomina función de vecindario a la función $N : \Omega \mapsto \mathcal{P}(\Omega)$ que mapea los elementos en Ω a su conjunto potencia, asignando a cada solución $x \in \Omega$ un conjunto de soluciones $N(x) \in \mathcal{P}(\Omega)$ que se conoce como vecindario. Dicho conjunto de soluciones varía según la función de vecindario que se aplique, por lo que existen múltiples vecindarios posibles para un mismo espacio de búsqueda[3]. En el caso del QAP, una de las funciones de vecindario más utilizadas se basa en el operador de vecindario *swap*, el cual consiste en intercambiar dos elementos $x(i)$ y $x(j)$ $(i \neq j)$ en la solución original para crear una nueva solución vecina $y \in \Omega$. Es decir, si $x = \big(x(1)...x(i)...x(j)...x(n)\big)$, entonces $y = \big(x(1)...x(j)...x(i)...x(n)\big)$. A modo de ejemplo, dado el operador *swap* y la permutación (123), el vecindario resultante estaría compuesto por las soluciones (213), (321) y (132). El tamaño del vecindario derivado de este operador es, por consiguiente, $|N(x)| = \frac{n(n-1)}{2}$.

A partir de la definición de *landscape* se derivan los conceptos de mínimo y máximo local de un problema de optimización combinatoria:

- *Mínimo local:* Cualquier solución $x \in \Omega$ tal que $f(x) \leq f(y)$ para todo $y \in N(x)$.
- *Máximo local:* Cualquier solución $x \in \Omega$ tal que $f(x) \geq f(y)$ para todo $y \in N(x)$.

Como se puede observar, los óptimos locales en el espacio de búsqueda dependen del vecindario escogido, por lo que un óptimo local en un vecindario puede no ser óptimo

[3]Para simplificar, en este proyecto solo tendremos en cuenta vecindarios que sean simétricos $\big(y \in N(x) \leftrightarrow x \in N(y)\big)$ y regulares $\big(|N(x)|$ constante para todo $x \in \Omega\big)$.

local en otro vecindario. Este no es el caso de los óptimos globales, ya que por definición son óptimos locales en todos los vecindarios.

3.2.1. Landscapes elementales

Entre todos los posibles *landscapes*, existen algunos con ciertas propiedades que resultan de interés a la hora de abordar problemas de optimización. Este es el caso de los *landscapes* elementales [15][16], los cuales fueron descubiertos por L.K. Grover [17] al percatarse de que los procesos de búsqueda local en algunos problemas de optimización combinatoria podían ser modelados mediante una fórmula discreta similar a la ecuación de onda usada en la física. Esta fórmula, conocida como la ecuación de onda de Grover, permite calcular el *fitness* medio del vecindario de cualquier solución $x \in \Omega$ a partir del valor de $f(x)$:

$$\underset{y \in N(x)}{avg}\{f(y)\} = f(x) + \frac{k}{|N(x)|}\left(\bar{f} - f(x)\right) \tag{3.7}$$

donde $\bar{f}$ es el valor medio de la función objetivo en todo el espacio de búsqueda y k es una constante característica de valor fijo para todo el *landscape*. Todas aquellas tripletas (Ω, N, f) para las que se cumple la ecuación de onda de Grover se consideran, por lo tanto, *landscapes* elementales. De forma más concreta, esto implica que un *landscape* es elemental si y solo si su función objetivo f es una función elemental con respecto a la estructura de vecindad creada por la función de vecindario N [13][16][17].

Tal y como se demuestra en [25], una de las peculiaridades de los *landscapes* elementales es que cumplen con las siguientes expresiones:

- $f(x) < \bar{f} \Longrightarrow f(x) < \underset{y \in N(x)}{avg}\{f(y)\} < \bar{f}$
- $f(x) = \bar{f} \Longrightarrow f(x) = \underset{y \in N(x)}{avg}\{f(y)\} = \bar{f}$
- $f(x) > \bar{f} \Longrightarrow f(x) > \underset{y \in N(x)}{avg}\{f(y)\} > \bar{f}$

Estas características derivadas de la ecuación de onda confirman las conclusiones expuestas en [17] en las que se afirma que el *fitness* de los mínimos locales $x^* \in \Omega$ en un *landscape* elemental es siempre inferior a la media del *fitness* en todo el espacio de

búsqueda ($f(x^*) < \bar{f}$). De forma análoga, también demuestran que todos los máximos locales $y^* \in \Omega$ tienen un *fitness* superior a dicha media ($f(y^*) > \bar{f}$). Esto implica que los *landscapes* elementales describen una clase de problemas que tienen una estructura relativamente conocida.

Una de las principales ventajas que ofrecen los *landscapes* elementales es que dada cualquier solución $x \in \Omega$ es posible calcular el *fitness* medio de su vecindario $N(x)$ mediante la ecuación de onda de Grover. Sin embargo, esta no es la única utilidad de dicha ecuación, ya que tras explorar una parte del vecindario $M \subset N(x)$ también nos permite calcular el *fitness* medio de las soluciones restantes $N(x) - M$. En concreto, el valor esperado para una solución $y \in (N(x) - M)$ elegida de forma aleatoria se calcula de la siguiente forma [15]:

$$\underset{y \in (N(x)-M)}{avg}\{f(y)\} = \frac{1}{|N(x)-M|}\left(|N(x)|\left(f(x) + \frac{k}{|N(x)|}\left(\bar{f} - f(x)\right)\right) - \sum_{z \in M} f(z)\right) \tag{3.8}$$

Para entender las ventajas de este planteamiento, supongamos que disponemos de dos soluciones $x_1, x_2 \in \Omega$. Por otro lado, supongamos que hemos explorado parte de los vecindarios de dichas soluciones $M_1 \subset N(x_1)$ y $M_2 \subset N(x_2)$. Si y_1 e y_2 son soluciones aleatorias pertenecientes a $N(x_1) - M_1$ y $N(x_2) - M_2$ respectivamente, podemos calcular el valor esperado para ambas soluciones. Si el valor esperado para y_1 es mejor que el valor esperado para y_2, resulta más prometedor seguir explorando las soluciones restantes del vecindario de x_1, independientemente de si el *fitness* medio calculado para todo $N(x_1)$ es peor que el *fitness* medio calculado para todo $N(x_2)$ (Figura 3.2).

En resumen, los *landscapes* elementales proporcionan ventajas que los vuelven especialmente adecuados para la optimización combinatoria. Algunos ejemplos conocidos de *landscapes* elementales en problemas combinatorios son:

1. El TSP simétrico y antisimétrico con un operador de vecindario basado en cambiar dos aristas no contiguas del recorrido por otras dos aristas distintas que formen un tour (2-opt) [15].

2. El problema de coloración de grafos con un operador de vecindario que se basa en cambiar el color de un único vértice [17].

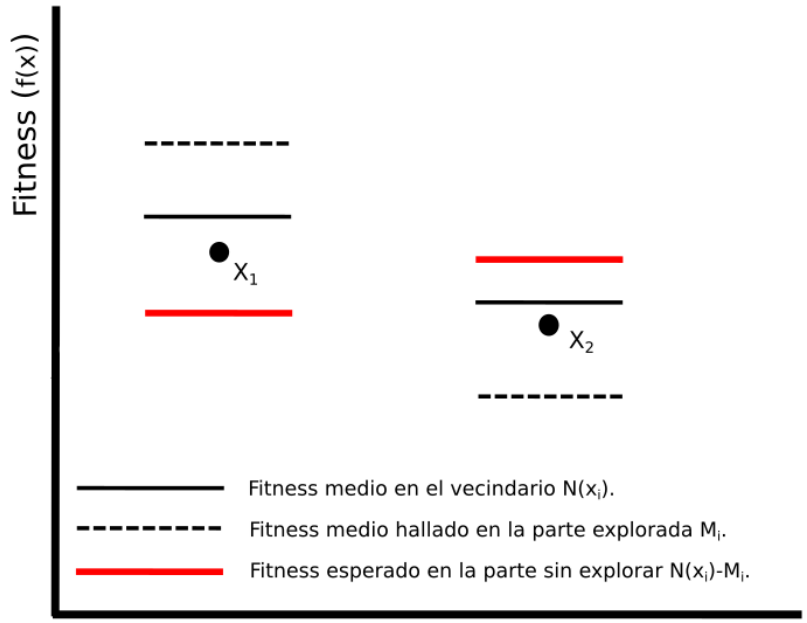


Figura 3.2: Asumiendo minimización, el *fitness* medio de $N(x_1)$ es peor que el *fitness* medio de $N(x_2)$. Sin embargo, el *fitness* esperado para una solución aleatoria en $N(x_1) - M_1$ es mejor que el *fitness* esperado para una solución aleatoria en $N(x_2) - M_2$, por lo que es más prometedor seguir explorando el vecindario de x_1.

3.2.2. Descomposición en landscapes elementales del QAP

Aunque no todos los *landscapes* son elementales, cualquier *landscape* cuyo vecindario sea simétrico ($y \in N(x) \leftrightarrow x \in N(y)$) se puede caracterizar como la combinación lineal de un conjunto de *landscapes* elementales independientes. El proceso mediante el que se obtienen dichos *landscapes* a partir del problema original se conoce como descomposición en *landscapes* elementales (ELD) [13].

Realizar el ELD de un *landscape* (Ω, N, f) se basa en encontrar un conjunto de funciones elementales $\{f_1, ..., f_m\}$ que formen m *landscapes* elementales junto con el espacio de búsqueda Ω y la función de vecindario N. Dicho conjunto de funciones se extrae de la descomposición de la función objetivo original f, por lo que se debe cumplir que $f = f_1 + ... + f_m$. Consideramos que una función f_i forma un *landscape* elemental si cumple la función de onda de Grover, es decir, si existen dos constantes a y b tal que para todas las soluciones $x \in \Omega$ se cumple:

$$\underset{y \in N(x)}{avg}\{f_i(y)\} = a \cdot f_i(x) + b \tag{3.9}$$

Dado que $\underset{y\in N(x)}{avg}\{f_i(y)\} = \frac{\sum_{y\in N(x)} f_i(y)}{|N(x)|}$, podemos multiplicar ambas partes de la ecuación por $|N(x)|$ para reducir la expresión:

$$\sum_{y\in N(x)} f_i(y) = c \cdot f_i(x) + d \tag{3.10}$$

donde $c = a|N(x)|$ y $d = b|N(x)|$. Si esta ecuación se cumple para ciertos valores de las constantes, se demuestra que f_i es una función elemental con respecto a la estructura de vecindad creada por la función de vecindario N, y por consiguiente, la tripleta (Ω, N, f_i) resulta ser un *landscape* elemental.

Teniendo en cuenta lo anterior, en este apartado nos centraremos en analizar la descomposición en *landscapes* elementales del QAP con la función de vecindario *swap* que hemos explicado al inicio de la Sección 3.2 [13][14]. Para empezar, comenzamos reescribiendo la función objetivo del problema (Ecuación 3.3) de la siguiente forma:

$$f(x) = \sum_{i,j=1}^{n} \sum_{p,q=1}^{n} d_{i,j} h_{p,q} \delta^{p}_{x(i)} \delta^{q}_{x(j)} \tag{3.11}$$

donde δ^b_a es la función delta de Kronocker que devuelve 1 si $a = b$ y 0 en caso contrario. Esta nueva función se compone de dos partes diferenciadas:

- La parte relacionada con la propia instancia, compuesta por la información de las matrices de distancia y flujo: $d_{i,j} h_{p,q}$. Esta parte no varía en función de x.
- La parte relacionada con el propio problema a resolver, compuesta por el producto entre las funciones delta de Kronocker: $\delta^{p}_{x(i)} \delta^{q}_{x(j)}$.

Organizar la función objetivo de esta manera permite obtener una combinación lineal de funciones (la parte relacionada con el problema) donde los coeficientes son determinados por la parte relacionada con la instancia. Esto nos interesa especialmente ya que la combinación lineal de funciones elementales con una misma constante característica k es también una función elemental. Partiendo de este planteamiento, se formula una función objetivo más general que facilitará la descomposición en *landscapes* elementales:

$$f(x) = \sum_{i,j,p,q=1}^{n} \psi_{i,j,p,q} \varphi_{(i,j)(p,q)}(x) \tag{3.12}$$

donde $\psi_{i,j,p,q} = (d_{i,j}h_{p,q})$ y $\varphi_{(i,j)(p,q)}(x) = (\delta^p_{x(i)}\delta^q_{x(j)})$[4]. Tomando esto en cuenta, podemos centrarnos en descomponer la parte relacionada con el problema (φ), ya que cualquier resultado que obtengamos puede ser extendido a una combinación lineal de φ, y por lo tanto, a f. Considerando la función de vecindario basada en el operador *swap*, diferenciamos tres escenarios distintos que analizaremos por separado:

1. ***i = j ∧ p = q***: En este caso, $\varphi_{(i,j)(p,q)} = \varphi_{(i,i)(p,p)}$. Los dos posibles valores de $\varphi_{(i,i)(p,p)}$ son los siguientes:

   - $\boldsymbol{\varphi(x) = 1}$: Toda solución x que cumpla $x(i) = p$ tendrá $n-1$ vecinos con $\varphi(y) = 0$ y el resto de vecinos con $\varphi(y) = 1$. Esto se debe a que el operador de vecindario *swap* modifica la posición i en $n-1$ vecinos (uno por cada posición $j \neq i$), mientras que en el resto se sigue cumpliendo que $y(i) = p$. Por lo tanto:
   $$\sum_{y \in N(x)} \varphi(y) = |N(x)| - n + 1 \tag{3.13}$$

   - $\boldsymbol{\varphi(x) = 0}$: Toda solución x que cumpla $x(i) \neq p$ tendrá un solo vecino con $\varphi(y) = 1$ y el resto de vecinos con $\varphi(y) = 0$. Esto se debe a que el operador de vecindario *swap* intercambia la posición i por la posición j tal que $x(j) = p$ en un solo vecino, mientras que en el resto se sigue cumpliendo que $y(i) \neq p$. Por lo tanto:
   $$\sum_{y \in N(x)} \varphi(y) = 1 \tag{3.14}$$

   Teniendo en cuenta la Ecuación 3.10, obtenemos el siguiente sistema lineal de ecuaciones:
   $$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} |N(x)| - n + 1 \\ 1 \end{pmatrix} \tag{3.15}$$

   Este sistema se resuelve como $c = |N(x)| - n$ y $d = 1$, lo que nos lleva a que $a = 1 - \frac{n}{|N(x)|}$ y $b = \frac{1}{|N(x)|}$. Finalmente, sustituimos a y b en la Ecuación 3.9:

$$\underset{y \in N(x)}{avg}\{\varphi(y)\} = \left(1 - \frac{n}{|N(x)|}\right)\varphi(x) + \frac{1}{|N(x)|} = \varphi(x) + \frac{n}{|N(x)|}\left(\frac{1}{n} - \varphi(x)\right) \tag{3.16}$$

[4]Para facilitar la comprensión de las ecuaciones, a partir de este punto se omitirán los parámetros de las funciones siempre que no haya confusión.

Comparando las Ecuaciones 3.16 y 3.7 se deduce que la función $\varphi_{(i,i)(p,p)}$ puede ser caracterizada por la ecuación de onda de Grover con $k = n$ y $\bar{\varphi} = \frac{1}{n}$. Por lo tanto, se trata de una función elemental.

2. ***i* ≠ *j* ∧ *p* ≠ *q***: Para analizar este caso, debemos introducir una familia de funciones auxiliares que, al igual que φ, asignan valores en $\mathbb{R}$ a permutaciones x de tamaño n:

$$\phi^{\alpha,\beta,\gamma,\varepsilon,\zeta}_{(i,j)(p,q)}(x) = \begin{cases} \alpha & \text{si } x(i) = p \wedge x(j) = q \\ \beta & \text{si } x(i) = q \wedge x(j) = p \\ \gamma & \text{si } x(i) = p \oplus x(j) = q \\ \varepsilon & \text{si } x(i) = q \oplus x(j) = p \\ \zeta & \text{si } x(i) \neq p,q \wedge x(j) \neq p,q \end{cases} \tag{3.17}$$

donde $1 \leq i, j, p, q \leq n$ con $i \neq j$ y $p \neq q$. Los parámetros α, β, γ, ε y ζ son parámetros reales ($\mathbb{R}$) que se deben determinar de antemano, y el símbolo $\oplus$ representa el operador de disyunción exclusiva (*XOR*). Los cinco posibles valores para $\phi^{\alpha,\beta,\gamma,\varepsilon,\zeta}_{(i,j)(p,q)}$ son los siguientes[5]:

- $\boldsymbol{\phi(x) = \alpha}$: Toda solución x que cumpla $x(i) = p$ y $x(j) = q$ tendrá un vecino con $\phi(y) = \beta$, $2(n-2)$ vecinos con $\phi(y) = \gamma$ y el resto de vecinos con $\phi(y) = \alpha$. Por lo tanto:

$$\sum_{y \in N(x)} \phi(y) = \beta + 2(n-2)\gamma + (|N(x)| - 2n + 3)\alpha \tag{3.18}$$

- $\boldsymbol{\phi(x) = \beta}$: Toda solución x que cumpla $x(i) = q$ y $x(j) = p$ tendrá un vecino con $\phi(y) = \alpha$, $2(n-2)$ vecinos con $\phi(y) = \varepsilon$ y el resto de vecinos con $\phi(y) = \beta$. Por lo tanto:

$$\sum_{y \in N(x)} \phi(y) = \alpha + 2(n-2)\varepsilon + (|N(x)| - 2n + 3)\beta \tag{3.19}$$

[5]En aras de la brevedad, a partir de este punto se ha omitido la explicación sobre por qué cada caso tiene una cierta cantidad de vecinos con valores de ϕ concretos. El razonamiento a seguir es completamente equivalente al explicado en la casuística de $\varphi_{(i,i)(p,p)}$, solo que en vez de considerar una solo posición (i) y un solo valor (p) ahora se consideran dos posiciones (i, j) y dos valores (p, q). Para más información, se recomienda acudir a [13] y [14].

- $\boldsymbol{\phi(x) = \gamma}$: Toda solución x que cumpla $x(i) = p$ o $x(j) = q$ pero no ambas tendrá un vecino con $\phi(y) = \alpha$, dos vecinos con $\phi(y) = \varepsilon$, $n-3$ vecinos con $\phi(y) = \zeta$ y el resto de vecinos con $\phi(y) = \gamma$. Por lo tanto:

$$\sum_{y \in N(x)} \phi(y) = \alpha + 2\varepsilon + (n-3)\zeta + (|N(x)| - n)\gamma \tag{3.20}$$

- $\boldsymbol{\phi(x) = \varepsilon}$: Toda solución x que cumpla $x(i) = q$ o $x(j) = p$ pero no ambas tendrá un vecino con $\phi(y) = \beta$, dos vecinos con $\phi(y) = \gamma$, $n-3$ vecinos con $\phi(y) = \zeta$ y el resto de vecinos con $\phi(y) = \varepsilon$. Por lo tanto:

$$\sum_{y \in N(x)} \phi(y) = \beta + 2\gamma + (n-3)\zeta + (|N(x)| - n)\varepsilon \tag{3.21}$$

- $\boldsymbol{\phi(x) = \zeta}$: Toda solución x que cumpla $x(i) \neq p, q$ y $x(j) \neq p, q$ tendrá dos vecinos con $\phi(y) = \gamma$, dos vecinos con $\phi(y) = \varepsilon$ y el resto de vecinos con $\phi(y) = \zeta$. Por lo tanto:

$$\sum_{y \in N(x)} \phi(y) = 2\gamma + 2\varepsilon + (|N(x)| - 4)\zeta \tag{3.22}$$

Teniendo en cuenta la Ecuación 3.10, obtenemos el siguiente sistema lineal de ecuaciones:

$$\begin{pmatrix} \alpha & 1 \\ \beta & 1 \\ \gamma & 1 \\ \varepsilon & 1 \\ \zeta & 1 \end{pmatrix} \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} \beta + 2(n-2)\gamma + (|N(x)| - 2n + 3)\alpha \\ \alpha + 2(n-2)\varepsilon + (|N(x)| - 2n + 3)\beta \\ \alpha + 2\varepsilon + (n-3)\zeta + (|N(x)| - n)\gamma \\ \beta + 2\gamma + (n-3)\zeta + (|N(x)| - n)\varepsilon \\ 2\gamma + 2\varepsilon + (|N(x)| - 4)\zeta \end{pmatrix} \tag{3.23}$$

Este sistema puede ser resuelto para algunas combinaciones de valores de los parámetros $\alpha, \beta, \gamma, \varepsilon$ y ζ. En este caso, nos interesan las siguientes tres combinaciones:

a) $\alpha = n-3, \beta = 1-n, \gamma = -2, \varepsilon = 0, \zeta = -1$

b) $\alpha = n-3, \beta = n-3, \gamma = 0, \varepsilon = 0, \zeta = 1$

c) $\alpha = 2n-3, \beta = 1, \gamma = n-2, \varepsilon = 0, \zeta = -1$

Con la primera combinación (a), el sistema se resuelve[6] como $c = \frac{n(n-5)}{2}$ y $d = -2n$,

[6]Recordemos que el tamaño del vecindario derivado del operador *swap* es $|N(x)| = \frac{n(n-1)}{2}$.

lo que nos lleva a que $a = 1 - \frac{2n}{|N(x)|}$ y $b = \frac{-2n}{|N(x)|}$. Contando con esto, sustituimos a y b en la Ecuación 3.9. Para simplificar, hemos definido que $\phi^1_{(i,j)(p,q)} = \phi^{n-3,1-n,-2,0,-1}_{(i,j)(p,q)}$.

$$\underset{y\in N(x)}{avg}\{\phi^1(y)\} = \Big(1 - \frac{2n}{|N(x)|}\Big)\phi^1(x) - \frac{-2n}{|N(x)|} = \phi^1(x) + \frac{2n}{|N(x)|}\Big(-1 - \phi^1(x)\Big) \tag{3.24}$$

Comparando las Ecuaciones 3.24 y 3.7 se deduce que la función $\phi^1_{(i,j)(p,q)}$ puede ser caracterizada por la ecuación de onda de Grover con $k = 2n$ y $\bar{\phi}^1 = -1$. Por lo tanto, se trata de una función elemental.

Con la segunda combinación (b), el sistema se resuelve como $c = \frac{(n-1)(n-4)}{2}$ y $d = 2(n-3)$, lo que nos lleva a que $a = 1 - \frac{2(n-1)}{|N(x)|}$ y $b = \frac{2(n-3)}{|N(x)|}$. Contando con esto, sustituimos a y b en la Ecuación 3.9. Para simplificar, hemos definido que $\phi^2_{(i,j)(p,q)} = \phi^{n-3,n-3,0,0,1}_{(i,j)(p,q)}$.

$$\underset{y\in N(x)}{avg}\{\phi^2(y)\} = \Big(1 - \frac{2(n-1)}{|N(x)|}\Big)\phi^2(x) - \frac{2(n-3)}{|N(x)|} = \phi^2(x) + \frac{2(n-1)}{|N(x)|}\Big(\frac{n-3}{n-1} - \phi^2(x)\Big) \tag{3.25}$$

Comparando las Ecuaciones 3.25 y 3.7 se deduce que la función $\phi^2_{(i,j)(p,q)}$ puede ser caracterizada por la ecuación de onda de Grover con $k = 2(n-1)$ y $\bar{\phi}^2 = \frac{n-3}{n-1}$. Por lo tanto, se trata de una función elemental.

Por último, con la tercera combinación (c), el sistema se resuelve como $c = \frac{n(n-3)}{2}$ y $d = n$, lo que nos lleva a que $a = 1 - \frac{n}{|N(x)|}$ y $b = \frac{n}{|N(x)|}$. Contando con esto, sustituimos a y b en la Ecuación 3.9. Para simplificar, hemos definido que $\phi^3_{(i,j)(p,q)} = \phi^{2n-3,1,n-2,0,-1}_{(i,j)(p,q)}$.

$$\underset{y\in N(x)}{avg}\{\phi^3(y)\} = \Big(1 - \frac{n}{|N(x)|}\Big)\phi^3(x) + \frac{n}{|N(x)|} = \phi^3(x) + \frac{n}{|N(x)|}\Big(1 - \phi^3(x)\Big) \tag{3.26}$$

Comparando las Ecuaciones 3.26 y 3.7 se deduce que la función $\phi^3_{(i,j)(p,q)}$ puede ser caracterizada por la ecuación de onda de Grover con $k = n$ y $\bar{\phi}^3 = 1$. Por lo tanto, se trata de una función elemental.

La razón por la que estas tres funciones elementales (ϕ^1,ϕ^2, ϕ^3) resultan de especial interés es que la función $\varphi_{(i,j)(p,q)}$ con $i \neq j$ y $p \neq q$ puede ser expresada como una combinación lineal de las mismas:

$$\varphi_{(i,j)(p,q)}(x) = \frac{1}{2n}\phi^1_{(i,j)(p,q)}(x) + \frac{1}{2(n-2)}\phi^2_{(i,j)(p,q)}(x) + \frac{1}{n(n-2)}\phi^3_{(i,j)(p,q)}(x) \tag{3.27}$$

Esta igualdad se demuestra en la Tabla 3.1 en la que se compara el valor del sumatorio de los tres componentes de la combinación lineal con el valor de la función $\varphi_{(i,j)(p,q)}$ en los cinco posibles escenarios.

Tabla 3.1: Combinación lineal de las funciones elementales comparada con $\varphi_{(i,j)(p,q)}$.

Condiciones	$\frac{\phi^1(x)}{2n}$	$\frac{\phi^2(x)}{2(n-2)}$	$\frac{\phi^3(x)}{n(n-2)}$	$\boldsymbol{\Sigma}$	$\boldsymbol{\varphi(x)}$
$x(i) = p \wedge x(j) = q$	$\frac{n-3}{2n}$	$\frac{n-3}{2(n-2)}$	$\frac{2n-3}{n(n-2)}$	**1**	**1**
$x(i) = q \wedge x(j) = p$	$\frac{1-n}{2n}$	$\frac{n-3}{2(n-2)}$	$\frac{1}{n(n-2)}$	**0**	**0**
$x(i) = p \oplus x(j) = q$	$\frac{-2}{2n}$	0	$\frac{n-2}{n(n-2)}$	**0**	**0**
$x(i) = q \oplus x(j) = p$	0	0	0	**0**	**0**
$x(i) \neq p,q \wedge x(j) \neq p,q$	$\frac{-1}{2n}$	$\frac{1}{2(n-2)}$	$\frac{-1}{n(n-2)}$	**0**	**0**

Dado que en todos los casos ambos valores son equivalentes, queda demostrado que la función $\varphi_{(i,j)(p,q)}$ con $i \neq j$ y $p \neq q$ se puede descomponer en tres funciones elementales con distinta constante característica.

3. ***i = j ⊕ p = q***: En este caso, el valor de $\varphi_{(i,j)(p,q)}$ siempre será cero. Esto se debe a que en el QAP no puede haber dos fábricas en una misma localización (si $x(i) = p$, $x(i) \neq q$ para todo q distinto de p) ni una misma fábrica en dos localizaciones (si $x(i) = p$, $x(j) \neq p$ para todo j distinto de i). Por ello, podemos dejar de lado este caso a la hora de realizar la descomposición de φ.

En resumen, la función objetivo generalizada del QAP (Ecuación 3.12) puede ser expresada como una combinación lineal de varias funciones elementales:

$$f(x) = \sum_{i,p=1}^{n} \psi_{i,i,p,p}\varphi_{(i,i)(p,p)}(x) + \sum_{\substack{i,j,p,q=1 \\ i\neq j \\ p\neq q}}^{n} \psi_{i,j,p,q}\left(\frac{\phi^1_{(i,j)(p,q)}(x)}{2n} + \frac{\phi^2_{(i,j)(p,q)}(x)}{2(n-2)} + \frac{\phi^3_{(i,j)(p,q)}(x)}{n(n-2)}\right) \tag{3.28}$$

Si agrupamos dichas funciones según su constante característica k[7], nos quedamos con solo tres funciones elementales:

$$f_1(x) = \sum_{\substack{i,j,p,q=1 \\ i\neq j \\ p\neq q}}^{n} \psi_{i,j,p,q}\frac{\phi^1_{(i,j)(p,q)}(x)}{2n} \tag{3.29}$$

$$f_2(x) = \sum_{\substack{i,j,p,q=1 \\ i\neq j \\ p\neq q}}^{n} \psi_{i,j,p,q}\frac{\phi^2_{(i,j)(p,q)}(x)}{2(n-2)} \tag{3.30}$$

$$f_3(x) = \sum_{i,p=1}^{n} \psi_{i,i,p,p}\varphi_{(i,i)(p,p)}(x) + \sum_{\substack{i,j,p,q=1 \\ i\neq j \\ p\neq q}}^{n} \psi_{i,j,p,q}\frac{\phi^3_{(i,j)(p,q)}(x)}{n(n-2)} \tag{3.31}$$

donde f_1 tiene la constante $k_1 = 2n$, f_2 tiene la constante $k_2 = 2(n-1)$ y f_3 tiene la constante $k_3 = n$. Como hemos explicado anteriormente, $f = f_1 + f_2 + f_3$. Estas tres funciones elementales forman junto con el espacio de búsqueda del QAP y la función de vecindario *swap* los tres *landscapes* elementales que componen la descomposición del problema.

[7] Recordemos que la combinación lineal de funciones elementales con una misma constante característica es también una función elemental.

4. CAPÍTULO

Análisis de los *landscapes* de la descomposición

Uno de los objetivos que más interesan en la consecución de este trabajo es entender mejor las características de los *landscapes* elementales derivados de la descomposición del QAP (Sub-sección 3.2.2) de forma que podamos utilizar dicho conocimiento para proponer algoritmos eficientes de optimización. En ese sentido, las características que creemos que son más interesantes son:

1. El número de óptimos locales [1] que posee cada uno de los *landscapes* de la descomposición. Esta medida parece estar relacionada con la dificultad que entraña hallar los óptimos globales [18][26], por lo que puede ser considerada como un indicativo indirecto de la complejidad del problema al resolverlo mediante algoritmos de búsqueda local.

2. La relación entre los *landscapes* de la descomposición y el *landscape* original del QAP, centrándonos sobre todo en las regiones localmente óptimas del espacio de búsqueda, es decir, aquellas en las que se agrupan los óptimos locales. De esta forma, podemos estudiar qué *landscapes* de la descomposición tienen un mayor peso en la estructura del problema.

Con esto en mente, hemos realizado un análisis de los *landscapes* de la descomposición en un *benchmark* de instancias. Para facilitar la comprensión y legibilidad, a partir de este punto los *landscapes* se denominarán de la siguiente forma:

[1] Dado que el QAP es un problema de minimización, siempre que hablemos de óptimos locales nos referiremos a mínimos locales a menos que se indique lo contrario.

- **L_0:** *Landscape* original del QAP cuya función objetivo (f_0) está descrita en las Ecuaciones 3.3 y 3.12.
- **L_1:** Primer *landscape* elemental de la descomposición del QAP cuya función objetivo (f_1) está descrita en la Ecuación 3.29.
- **L_2:** Segundo *landscape* elemental de la descomposición del QAP cuya función objetivo (f_2) está descrita en la Ecuación 3.30.
- **L_3:** Tercer *landscape* elemental de la descomposición del QAP cuya función objetivo (f_3) está descrita en la Ecuación 3.31.

Cabe recordar que en los cuatro *landscapes* el espacio de búsqueda (Ω) se compone por el conjunto de todas las permutaciones de tamaño n siendo n la dimensión del problema, mientras que la función de vecindario (N) se basa en el operador *swap* (Sección 3.2).

4.1. Instancias

La experimentación se ha realizado en base a 9 instancias extraídas de la librería QAPLIB [27]. Dichas instancias tienen características muy variadas, ya que se ha intentado abarcar un espectro muy amplio que nos permita comprobar cómo cambian las propiedades de los *landscapes* en función de las características de la instancia a resolver. Debido a las restricciones temporales del proyecto, no se han considerado tamaños mayores que $n = 30$. Las instancias elegidas son las siguientes:

- **Bur26a, Bur26b, Bur26c [5]:** La matriz de distancia de los problemas se compone del tiempo de desplazamiento medio de un estenotipista entre las teclas de una máquina de escribir, mientras que la matriz de flujo representa las frecuencias de aparición de los distintos pares de letras calculadas a partir de 100,000 pares de ejemplo en varios idiomas. El objetivo es encontrar la configuración de letras en el teclado de la máquina de escribir que minimice el tiempo medio de escritura. El tamaño de las instancias es $n = 26$.
- **Tai15b, Tai20b [28][29]:** Las matrices de distancia y flujo de los problemas están generadas de forma aleatoria pero no uniforme para asemejarse a problemas de la vida real. El tamaño de las instancias es $n = 15$ para *Tai15b* y $n = 20$ para *Tai20b*.

- **Chr15a [30]:** La matriz de distancia del problema se corresponde con la matriz de adyacencia de un árbol ponderado, mientras que la matriz de flujo se corresponde con la matriz de adyacencia de un grafo completo. Este caso particular del QAP en el que una de las matrices tiene estructura de árbol se denomina QAP-árbol [31]. El tamaño de la instancia es $n = 15$.
- **Esc16a [32]:** Los datos del problema provienen del análisis de circuitos secuenciales auto-comprobables donde el objetivo es minimizar el *hardware* necesario para realizar las comprobaciones. El tamaño de la instancia es $n = 16$.
- **Had20 [33]:** La matriz de distancia del problema se compone de las distancias de Manhattan de un complejo celular conectado en el plano, mientras que la matriz de flujo está generada con valores muestreados uniformemente del intervalo $[1, 10]$. El tamaño de la instancia es $n = 20$.
- **Rou20 [34]:** Las matrices de distancia y flujo del problema están generadas con valores muestreados del intervalo $[1, 100]$. El tamaño de la instancia es $n = 20$.

Uno de los aspectos que nos interesa comprobar en esta experimentación es el impacto que tiene la simetría[2] de las matrices de distancia y flujo de las que se componen las instancias en las propiedades de los *landscapes* de la descomposición. Por este motivo, hemos decidido agrupar las 9 instancias en base a esta característica:

- **Matrices de distancia y flujo asimétricas:** Bur26a, Bur26b, Bur26c.
- **Matriz de distancia simétrica y matriz de flujo asimétrica:** Tai15b, Tai20b.
- **Matrices de distancia y flujo simétricas:** Chr15a, Esc16a, Had20, Rou20.

Como se puede observar, ninguna de las instancias escogidas tiene una matriz de distancia ($D_{n\times n}$) asimétrica y una matriz de flujo ($H_{n\times n}$) simétrica. Esto se debe a que, dadas dos matrices $A_{n\times n}$ y $B_{n\times n}$, resolver una instancia I_1 con $D = A$ y $H = B$ es completamente equivalente a resolver otra instancia I_2 con $D = B$ y $H = A$ si para I_1 consideramos la función objetivo $f(x) = \sum_{i=1}^{n}\sum_{j=1}^{n}\left(d_{x(i),x(j)}h_{i,j}\right)$ donde $x(i)$ es la localización a la que se asigna la fábrica i. Esto también se cumple para cada uno de los *landscapes* de la descomposición, por lo que las características que observemos en las instancias en las que D es simétrica y H asimétrica se pueden extrapolar a las instancias en las que D es asimétrica y H simétrica, permitiéndonos así ahorrarnos varias experimentaciones redundantes.

[2] Una matriz $A_{n\times n}$ es simétrica si y solo si $a_{i,j} = a_{j,i}$ para todo $1 \leq i, j \leq n$.

4.2. Número de óptimos locales

En esta sección nos centraremos en analizar la estructura de los distintos *landscapes* de la descomposición desde el punto de vista del número de óptimos locales que posee cada uno de ellos. Con este objetivo, hemos realizado dos experimentos distintos pero estrechamente relacionados:

1. Estimar el número de óptimos locales (Sub-sección 4.2.1).
2. Agrupar los óptimos locales en mesetas (Sub-sección 4.2.2).

4.2.1. Experimento 1: Estimación del número de óptimos locales

Contabilizar de forma exacta el número total de óptimos locales en las instancias elegidas no es una opción temporalmente viable, por lo que se ha optado por utilizar un método de estimación que nos permita obtener valores aproximados. Por este motivo, hemos considerado la revisión de técnicas de estimación de óptimos locales propuesta en [18], y en base a ella se ha seleccionado el estimador *ChaoLee2* [35]. La razón por la que se ha optado por este método es que en [18] se recomienda su uso cuando el tamaño de muestra elegido es pequeño en comparación con el número total de óptimos locales, algo que es muy probable en nuestro caso debido a la alta cardinalidad del espacio de búsqueda del QAP. Aunque *ChaoLee2* fue propuesto inicialmente para estimar la cantidad de especies en una población, es posible adaptarlo a este problema de la siguiente forma:

1. Dada una muestra aleatoria de soluciones uniformemente distribuida $S = \{x_1, x_2, \ldots, x_M\} \subset \Omega$, se realiza una búsqueda local partiendo de cada solución $x_i \in S$.
2. A partir de las M búsquedas locales realizadas se obtienen r óptimos locales distintos $\{x_1^*, x_2^*, \ldots, x_r^*\}$ $(r \leq M)$.

De esta manera, podemos considerar que S es una muestra de individuos en la que se han hallado r especies distintas. Esto nos permite utilizar el método *ChaoLee2* para estimar la cantidad de especies en toda la población, o dicho de otra forma, el número de óptimos locales en todo el espacio de búsqueda. En este proyecto, la búsqueda local implementada para obtener los datos muestrales es el método *Iterative Best Improvement* (Algoritmo 1).

Algorithm 1 *Iterative Best Improvement.*

Input:

x - Solución inicial ($x \in \Omega$) **f** - Función objetivo **N** - Función de vecindario

Output:

$\mathbf{x}^*$ - Óptimo local ($x^* \in \Omega$)

```
1:  procedure MEJORA_ITERATIVA (x, f, N)
2:      x* ← x
3:      Mejora ← True
4:      while Mejora = True do
5:          Vecindario ← N(x*)
6:          Mejora ← False
7:          for y ∈ Vecindario do
8:              if f(y) < f(x*) then          ▷ Suponemos minimización.
9:                  x* ← y
10:                 Mejora ← True
11:             end if
12:         end for
13:     end while
14:     return x*          ▷ Óptimo local de f con la función de vecindario N.
15: end procedure
```

Al igual que otros métodos similares, *ChaoLee2* requiere la definición de lo que se conoce como óptimos fáciles de encontrar (especies abundantes) y óptimos difíciles de encontrar (especies raras). Un óptimo local se considera fácil de encontrar si aparece más de δ veces en la muestra, mientras que si aparece δ veces o menos se considera difícil de encontrar. El estimador se basa en el número de óptimos locales en cada grupo para realizar las estimaciones. Si bien δ puede tomar cualquier valor razonable, existen estudios [36] que recomiendan utilizar $\delta = 10$.

Uno de los inconvenientes de este enfoque es que *ChaoLee2* produce resultados imprecisos cuando la gran mayoría de los óptimos locales encontrados solo aparecen una vez en la muestra. Esto ocurre cuando la cantidad de búsquedas realizadas es demasiado pequeña en comparación con el número total de óptimos locales, por lo que si el *landscape* a analizar es muy rugoso, es decir, tiene muchos óptimos locales, es necesario adoptar enfoques alternativos. En estos casos, la estrategia utilizada consiste en tomar una muestra aleatoria de soluciones uniformemente distribuida $S = \{x_1, x_2, ..., x_M\} \subset \Omega$ y comprobar qué porcentaje de soluciones $x_i \in S$ son óptimos locales. Suponiendo que la proporción de óptimos locales en la muestra se corresponde con la proporción de óptimos locales en todo el espacio de búsqueda, este porcentaje nos servirá para sacar conclusiones. A este método de estimación alternativo lo denominamos *CountOptima*.

Experimentación y resultados

Las muestras utilizadas en las estimaciones constan de 200,000 búsquedas locales por instancia y *landscape*. El número de óptimos locales distintos hallados en cada caso se indica en la Tabla 4.1.

Tabla 4.1: Número de óptimos locales distintos obtenidos tras realizar 200,000 búsquedas locales en L_0, L_1, L_2 y L_3 para cada instancia comenzando desde soluciones aleatorias. Las instancias han sido agrupadas según la simetría de las matrices de distancia (D) y flujo (H).

	Bur26a	Bur26b	Bur26c	Tai15b	Tai20b	Chr15a	Esc16a	Had20	Rou20
L_0	169.606	196.590	149.738	2.046	23.251	37.405	199.580	34.267	187.232
L_1	190.491	199.979	195.252	200.000	200.000	200.000	200.000	200.000	200.000
L_2	83.134	141.689	55.346	5.125	17.942	41.893	199.580	95.927	193.590
L_3	13.823	199.057	40.837	2	1	4	200.000	768	1

D, H asimétricas. D simétrica, H asimétrica. D, H simétricas.

En base a estos resultados, hemos aplicado el método *ChaoLee2* con $\delta = 10$ tal y como se recomienda en [36] para estimar el número de óptimos locales en todo el espacio de búsqueda. En los casos en los que el número de óptimos distintos hallados en la muestra excede los 190,000 ($> 95\,\%$ del total de búsquedas), se ha realizado una estimación complementaria mediante el método *CountOptima* con 200,000 soluciones aleatorias. Los resultados obtenidos se indican en la Tabla 4.2.

Tabla 4.2: Estimación de óptimos locales a partir de los resultados muestrales obtenidos (Tabla 4.1). Los casos marcados en rojo implican que el número de óptimos locales distintos hallados en la muestra es superior a 190,000, por lo que en dichos casos se ha utilizado el método alternativo *CountOptima* con 200,000 soluciones aleatorias para estimar el porcentaje de óptimos locales en el espacio de búsqueda.

		Bur26a	Bur26b	Bur26c	Tai15b	Tai20b	Chr15a	Esc16a	Had20	Rou20
L_0	ChaoLee2	1.569.731	6.552.437	1.304.359	2.131	41.131	59.098	47.715.522	94.832	4.402.028
	CountOptima	-	0,00 %	-	-	-	-	0,0015 %	-	-
L_1	ChaoLee2	5.457.456	952.280.952	6.968.950	inf.	inf.	inf.	inf.	inf.	inf.
	CountOptima	0,00 %	0,00 %	0,00 %	100,0 %	100,0 %	100,0 %	100,0 %	100,0 %	100,0 %
L_2	ChaoLee2	279.052	1.147.330	145.401	5.787	28.443	72.729	47.715.522	445.087	6.151.671
	CountOptima	-	-	-	-	-	-	0,0015 %	-	0,00 %
L_3	ChaoLee2	13.823	21.300.868	41.400	2	1	4	inf.	768	1
	CountOptima	-	0,00 %	-	-	-	-	100,00 %	-	-

Los resultados de este experimento presentan escenarios muy distintos dependiendo de la simetría de las matrices que forman las instancias:

- En las instancias en las que alguna de las matrices es simétrica (Tai15b, Tai20b, Chr15a, Esc16a, Had20, Rou20), el porcentaje de óptimos locales en el espacio de

búsqueda del *landscape* L_1 estimado por el método *CountOptima* es del $100,00\,\%$ en todos los casos, es decir, ninguna de las 200,000 soluciones aleatorias que se han considerado en cada instancia tiene una solución mejor que ella en su vecindario. Por otro lado, en la mayoría de dichas instancias el *landscape* L_3 cuenta con muy pocos óptimos locales según el método *ChaoLee2*, llegando a tener una estimación de menos de 10 óptimos distintos en cuatro de las seis instancias analizadas.

- En las instancias asimétricas (Bur26a, Bur26b, Bur26c), la única característica común a las tres instancias analizadas es que, según el método *ChaoLee2*, el *landscape* L_1 tiene un número estimado de óptimos locales mucho mayor que L_2 y L_3.

Cabe destacar que hay varios casos en los que el método *ChaoLee2* estima que el número de óptimos locales en el espacio de búsqueda es infinito, lo cual es imposible. Esto ocurre cuando todos los óptimos locales encontrados en las búsquedas aparecen una única vez en la muestra, es decir, cuando se encuentran 200,000 óptimos distintos. En estos casos, el tamaño de la muestra es tan pequeño respecto al número total de óptimos locales que el estimador da error. De forma similar, el método *CountOptima* también produce resultados imposibles cuando la muestra es demasiado pequeña, indicando que el porcentaje de óptimos locales en el espacio de búsqueda es del $0,00\,\%$[3].

4.2.2. Experimento 2: Agrupación en mesetas

En el primer experimento los óptimos locales han sido considerados de forma independiente, por lo que hay aspectos que influyen en la estructura de los *landscapes* que no hemos tenido en cuenta. Uno de dichos aspectos son las mesetas compuestas por óptimos locales [37]. Una meseta es un conjunto de soluciones $P \subseteq \Omega$ en el que para todo $x, y \in P$ se cumple $f(x) = f(y)$ y existe un camino $(x = a_1, a_2, ..., a_k = y)$ tal que para todo i se cumple $a_i \in P$ y $a_{i+1} \in N(a_i)$. Una meseta compuesta por óptimos locales se comporta como una única región localmente óptima al aplicar algoritmos de búsqueda local, por lo que un *landscape* en el que la mayoría de óptimos se agrupan en unas pocas mesetas puede ser mucho menos rugoso de lo que indican las estimaciones. Por este motivo, para profundizar más en las características de los *landscapes*, hemos agrupado en mesetas los óptimos locales hallados en las búsquedas locales del primer experimento. Debido a las limitaciones temporales del proyecto, esta experimentación se ha llevado a cabo de la siguiente forma:

[3] Aunque se barajó la posibilidad de aumentar los tamaños de muestra para subsanar estos errores, las limitaciones temporales del proyecto lo impidieron.

- $\Theta \subset \Omega$ es el conjunto de óptimos locales que han sido hallados en la muestra de búsquedas locales.
- Dados dos óptimos locales $x^*, y^* \in \Theta$, consideramos que ambas soluciones pertenecen a una misma meseta si $f(x^*) = f(y^*)$ y existe un camino $(x^* = a_1^*, a_2^*, .., a_k^* = y^*)$ tal que para todo i se cumple $a_i^* \in \Theta$ y $a_{i+1}^* \in N(a_i^*)$.

Considerar solo los óptimos locales hallados en la muestra ($a_i^* \in \Theta$) nos permite acelerar los cálculos necesarios para esta experimentación, aunque los resultados no son tan precisos ya que podríamos estar considerando como dos mesetas distintas dos conjuntos de soluciones unidos por un camino de óptimos locales $(b_1^*, b_2^*, ..., b_k^*)$ en el que para todo i se cumple $b_i^* \notin \Theta$. Estos errores son más pronunciados cuantos más óptimos locales tenga el *landscape*, ya que el número de óptimos locales $b_i^* \notin \Theta$ aumenta de forma proporcional. De todos modos, esta experimentación nos permite obtener una cota superior aproximada del número de mesetas localmente óptimas en las muestras con un coste computacional asumible.

Experimentación y resultados

Los resultados muestrales utilizados en esta experimentación son los obtenidos en las búsquedas locales del primer experimento (Tabla 4.1). La cota superior aproximada del número de mesetas en las que se pueden agrupar los óptimos locales hallados en las búsquedas se indica en la Tabla 4.3.

Tabla 4.3: Cota superior del número de mesetas en las que se pueden agrupar los óptimos locales hallados en las muestras (Tabla 4.1). El número total de óptimos locales distintos se indica entre paréntesis. Se han omitido los casos en los que el porcentaje de óptimos locales estimado para todo el espacio de búsqueda es del 100,00 % (Tabla 4.2).

	Bur26a	Bur26b	Bur26c	Tai15b	Tai20b	Chr15a	Esc16a	Had20	Rou20
L_0	**127.779** (169.606)	**178.289** (196.590)	**104.218** (149.738)	**2.046** (2.046)	**23.251** (23.251)	**36.950** (37.405)	**191.970** (199.580)	**26.587** (34.267)	**187.229** (187.232)
L_1	**174.258** (190.491)	**199.579** (199.979)	**185.441** (195.252)	-	-	-	-	-	-
L_2	**38.554** (83.134)	**77.422** (141.689)	**22.416** (55.346)	**5.125** (5.125)	**17.942** (17.942)	**41.893** (41.893)	**191.970** (199.580)	**49.841** (95.927)	**193.590** (193.590)
L_3	**2** (13.823)	**169.851** (199.057)	**6** (40.837)	**1** (2)	**1** (1)	**1** (4)	-	**1** (768)	**1** (1)

Para visualizar mejor las diferencias, en la Tabla 4.4 se muestra el porcentaje de disminución del número de mesetas localmente óptimas con respecto al número total de

óptimos locales distintos hallados en las muestras. Dicho porcentaje se ha calculado de la siguiente forma:

$$P_{dism} = \left(1 - \frac{m-1}{o-1}\right) \cdot 100 \tag{4.1}$$

donde m es el número de mesetas y o el número total de óptimos locales.

Tabla 4.4: Porcentaje de disminución del número de mesetas localmente óptimas con respecto al número total de óptimos locales distintos hallados en las muestras (Tabla 4.1). Se han omitido los casos en los que solo se ha encontrado un óptimo local o el porcentaje de óptimos locales estimado para todo el espacio de búsqueda es del 100,00 % (Tabla 4.2).

	Bur26a	Bur26b	Bur26c	Tai15b	Tai20b	Chr15a	Esc16a	Had20	Rou20
L_0	24,66 %	9,31 %	30,40 %	0,00 %	0,00 %	1,22 %	3,81 %	22,41 %	0,00 %
L_1	8,52 %	0,20 %	5,02 %	-	-	-	-	-	-
L_2	53,62 %	45,36 %	59,50 %	0,00 %	0,00 %	0,00 %	3,81 %	48,04 %	0,00 %
L_3	99,99 %	14,67 %	99,99 %	100,00 %	-	100,00 %	-	100,00 %	-

Si bien agrupar en mesetas los óptimos locales hallados en las muestras conlleva variaciones en los resultados de L_1 y L_2 en algunas de las instancias (sobre todo en las asimétricas), en general el *landscape* que más se ve afectado es L_3. De hecho, en las instancias en las que alguna de las matrices es simétrica (Tai15b, Tai20b, Chr15a, Esc16a, Had20, Rou20), L_3 parece contar con una única meseta localmente óptima en todos los casos analizados. En las instancias asimétricas (Bur26a, Bur26b, Bur26c), por otro lado, el porcentaje de disminución en L_3 es del 99,99 % en dos de los tres casos, lo cuál resulta en menos de 10 mesetas localmente óptimas en las instancias Bur26a y Bur26c.

4.2.3. Conclusiones

- En las instancias en las que alguna de las matrices es simétrica (Tai15b, Tai20b, Chr15a, Esc16a, Had20, Rou20), tal y como observamos en el primer experimento, el porcentaje de óptimos locales estimado en el espacio de búsqueda del *landscape* L_1 es del 100,00 %. Esto puede ocurrir por dos motivos:

  1. Se trata de un *landscape* extremadamente rugoso, con muchas regiones localmente óptimas distintas.
  2. Se trata de un *landscape* cuya función objetivo tiene un valor constante en todo el espacio de búsqueda.

De acuerdo con lo señalado en [38], en este caso la opción correcta es la segunda, es decir, el valor de la función objetivo de L_1 es constante para todo $x \in \Omega$ cuando la matriz de distancia y/o la matriz de flujo son simétricas. Hasta donde llega nuestro conocimiento esta afirmación no ha sido demostrada formalmente en trabajos previos de la literatura, por lo que en el Anexo A de este proyecto hemos incluido una demostración matemática de esta característica de L_1.

- En las instancias en las que alguna de las matrices es simétrica (Tai15b, Tai20b, Chr15a, Esc16a, Had20, Rou20), el espacio de búsqueda del *landscape* L_3 parece tener muy pocas regiones óptimas. De hecho, en la mayoría de las instancias de este tipo solo se ha hallado una única meseta localmente óptima en L_3, lo que lo convierte en un *landscape* relativamente simple a la hora de resolverlo mediante algoritmos de búsqueda local. Como caso particular, en la instancia Esc16a nos encontramos con una situación similar a la del *landscape* L_1 ($100,00\,\%$ de óptimos locales), por lo que sospechamos que en este caso la función objetivo de L_3 también podría ser constante en todo el espacio de búsqueda.

- En base a los dos puntos anteriores, el *landscape* L_2 parece ser el más rugoso de los *landscapes* de la descomposición en las instancias en las que alguna de las matrices es simétrica (Tai15b, Tai20b, Chr15a, Esc16a, Had20, Rou20). De hecho, este *landscape* parece ser el que más se asemeja a L_0 en cuanto al número estimado de óptimos locales, por lo que sospechamos que ambos *landscapes* pueden estar especialmente relacionados en este tipo de instancias.

- En las instancias asimétricas (Bur26a, Bur26b, Bur26c), el *landscape* L_1 parece ser el más rugoso de los *landscapes* de la descomposición. Esta es la principal diferencia entre las instancias simétricas y asimétricas, ya que cabe recordar que en Tai15b, Tai20b, Chr15a, Esc16a, Had20 y Rou20 la función objetivo del *landscape* L_1 es constante.

- En las instancias asimétricas (Bur26a, Bur26b, Bur26c), el espacio de búsqueda del *landscape* L_3 también parece tener pocas regiones óptimas. De hecho, en dos de las instancias analizadas (Bur26a, Bur26c) el número de mesetas localmente óptimas halladas en el segundo experimento es inferior a 10. La única excepción es la instancia Bur26b, pero dado que se trata de una instancia con un alto número de óptimos locales en comparación con las demás, es posible que la cota superior calculada en la experimentación esté muy alejada del número real de mesetas.

4.3. Relación entre *landscapes*

En esta sección estudiaremos la relación entre el *landscape* original del QAP y los *landscapes* de la descomposición para comprobar cuál de los *landscapes* elementales tiene una mayor influencia en la estructura del problema a resolver. Con este objetivo, hemos realizado dos experimentos:

1. Cuantificar la correlación lineal entre los valores de las funciones objetivo (Subsección 4.3.1).
2. Estimar el porcentaje de óptimos locales compartidos entre *landscapes* (Sub-sección 4.3.2).

4.3.1. Experimento 3: Correlación lineal entre funciones objetivo

Lo primero que hemos comprobado a la hora de analizar la relación entre los *landscapes* es si el *fitness* en las funciones objetivo f_1, f_2 y f_3 guarda alguna dependencia lineal con el *fitness* en la función objetivo f_0. Para ello, se ha seleccionado una muestra aleatoria de soluciones uniformemente distribuida $S = \{x_1, x_2, ..., x_M\} \subset \Omega$ para cada una de las instancias y, en base a sus valores de *fitness*, se ha estimado el coeficiente de correlación de Pearson [39] entre las funciones objetivo de los distintos *landscapes*. El motivo por el que hemos elegido esta medida sobre otras como la covarianza es que el coeficiente de correlación de Pearson es independiente de la escala de las variables, lo cual nos interesa en este caso ya que las funciones objetivo tienen escalas muy diferentes.

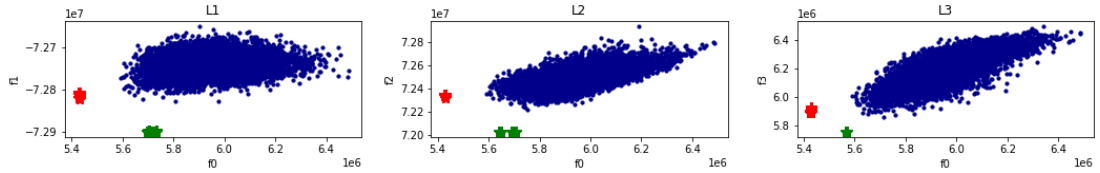
El coeficiente de correlación de Pearson (r) es una medida de dependencia lineal acotada en el intervalo $[-1, 1]$ que nos indica el nivel de correlación lineal entre dos variables cuantitativas X e Y. Cuanto más cerca de 1 esté el valor de r, mayor será la correlación lineal positiva entre ambas variables ($X \uparrow \Leftrightarrow Y \uparrow$), mientras que cuanto más cerca esté de -1, mayor será la correlación lineal negativa ($X \uparrow \Leftrightarrow Y \downarrow$). Si $r \approx 0$, entonces el valor de las dos variables no tendrá ninguna correlación lineal. Dada una muestra de tamaño M, el coeficiente de correlación de Pearson se puede estimar de la siguiente forma:

$$r = \frac{M(\sum_{i=1}^{M} x_i y_i) - (\sum_{i=1}^{M} x_i)(\sum_{i=1}^{M} y_i)}{\sqrt{M(\sum_{i=1}^{M} x_i^2) - (\sum_{i=1}^{M} x_i)^2}\sqrt{M(\sum_{i=1}^{M} y_i^2) - (\sum_{i=1}^{M} y_i)^2}} \tag{4.2}$$

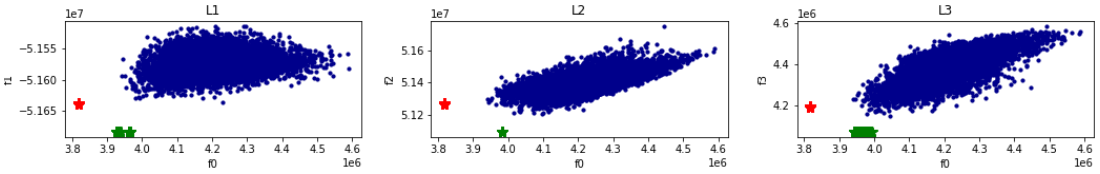
donde x_i e y_i son los valores de las variables X e Y en el elemento i-ésimo de la muestra.

Experimentación y resultados

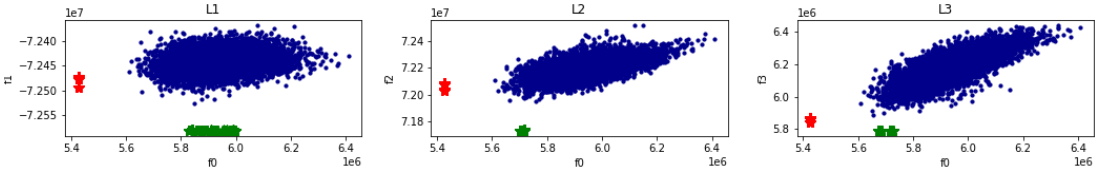
Las muestras seleccionadas para realizar esta experimentación constan de 10,000 soluciones aleatorias por instancia. En la Figura 4.1 se compara de forma visual el *fitness* de las soluciones en f_0 con el *fitness* de las soluciones en f_1, f_2 y f_3.



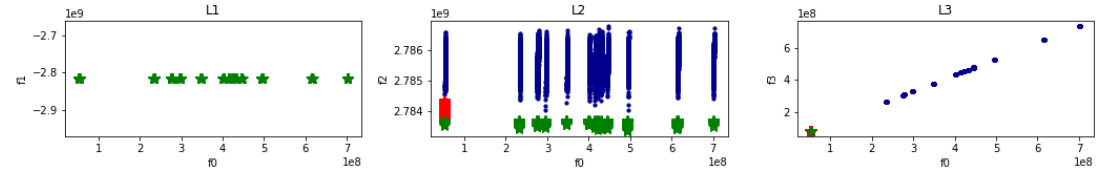
(a) Bur26a.



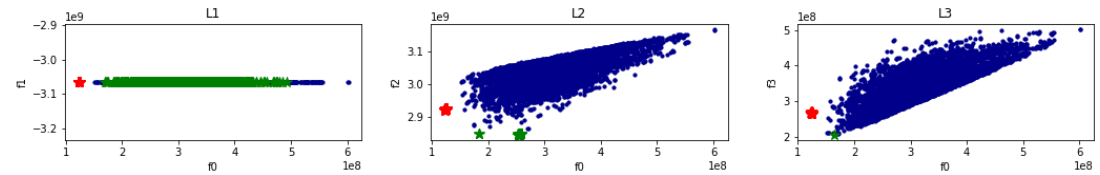
(b) Bur26b.



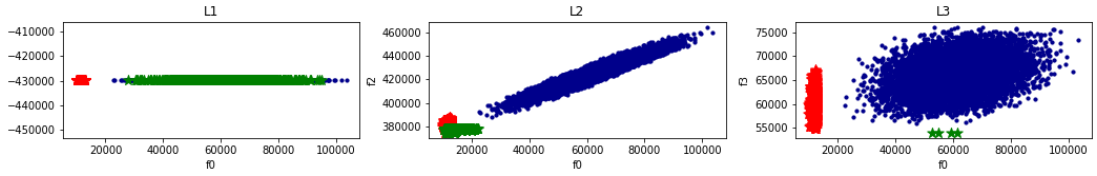
(c) Bur26c.



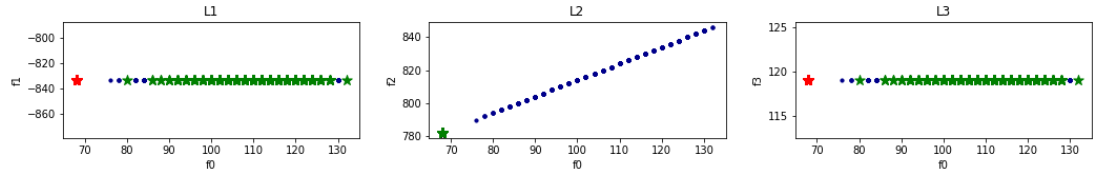
(d) Tai15b.



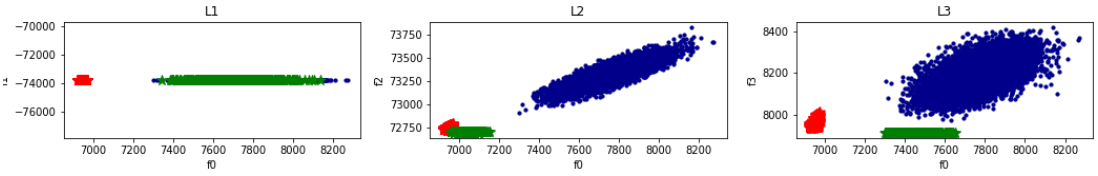
(e) Tai20b.



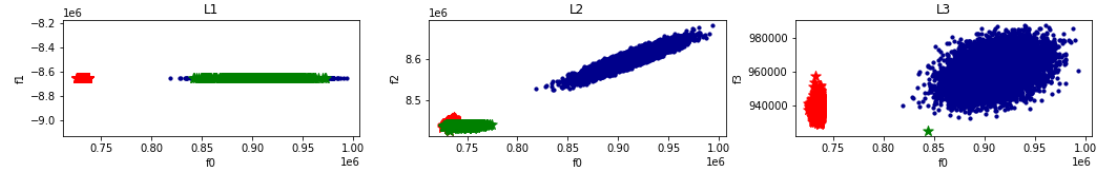
(f) Chr15a.



(g) Esc16a.



(h) Had20.



(i) Rou20.

Figura 4.1: Representación gráfica de los valores de *fitness* de 10,000 soluciones aleatorias para cada instancia (azul). El eje X de las gráficas marca el *fitness* de las soluciones en f_0, mientras que el eje Y marca el *fitness* de las soluciones en f_1, f_2 y f_3 respectivamente. Como punto de referencia, también se representan los 1,000 mejores óptimos locales de L_0 (rojo) y los 1,000 mejores óptimos locales del *landscape* de la descomposición correspondiente (verde) encontrados en las búsquedas locales del primer experimento (Tabla 4.1).

Como se puede apreciar, en algunos de los casos las soluciones aleatorias están muy agrupadas entorno a la diagonal de la gráfica. Esto se puede ver principalmente en la comparación entre las funciones objetivo f_0 y f_2 en las instancias en las que ambas matrices son simétricas (Chr15a, Esc16a, Had20, Rou20), por lo que ya a simple vista se intuye una fuerte correlación lineal positiva entre dichas funciones en este tipo de instancias. En el resto de casos, sin embargo, las soluciones aleatorias están en general

más distribuidas por toda la gráfica, por lo que no podemos sacar conclusiones sin antes cuantificar de forma más precisa las correlaciones. Una notable excepción es el caso de la función f_1 en las instancias en las que alguna de las matrices es simétrica (Tai15b, Tai20b, Chr15a, Esc16a, Had20, Rou20), ya que como hemos explicado anteriormente su valor es constante en todo el espacio de búsqueda.

En base a las muestras de soluciones aleatorias generadas, se ha estimado el coeficiente de correlación de Pearson entre el *fitness* en f_0 y el *fitness* en f_1, f_2 y f_3 para cada una de las instancias. Los resultados obtenidos se indican en la Tabla 4.5.

Tabla 4.5: Coeficiente de correlación de Pearson entre el *fitness* en f_0 y el *fitness* en f_1, f_2 y f_3 estimado a partir de las muestras aleatorias (Figura 4.1). La función objetivo que tiene la mayor correlación lineal positiva con f_0 en cada caso está marcada en verde.

	Bur26a	Bur26b	Bur26c	Tai15b	Tai20b	Chr15a	Esc16a	Had20	Rou20
f_1	0,182985	0,182161	0,178930	0,00	0,00	0,00	0,00	0,00	0,00
f_2	0,604764	0,641440	0,587050	-0,016312	0,650269	0,952239	1,00	0,838233	0,924578
f_3	0,774646	0,744030	0,785484	0,999997	0,758697	0,303525	0,00	0,554388	0,365383

Lo primero que podemos observar es que en prácticamente todos los casos encontramos una correlación lineal nula o positiva ($r \geq 0$) entre f_0 y las funciones objetivo de los *landscapes* de la descomposición, lo cuál tiene sentido ya que f_0 se calcula como la suma de f_1, f_2 y f_3. Teniendo esto en cuenta, los resultados de este experimento presentan escenarios muy distintos dependiendo de la simetría de las matrices que forman las instancias:

- Tal y como hemos apreciado previamente en las gráficas, en las instancias en las que ambas matrices son simétricas (Chr15a, Esc16a, Had20, Rou20) la función objetivo f_2 es la que más correlación lineal positiva presenta con f_0, con un coeficiente de correlación lineal mayor que 0,90 en tres de las cuatro instancias. Además, en estos casos las funciones f_1 y f_0 tienen un coeficiente de correlación lineal de 0 debido a que el valor de f_1 es constante.
- En las instancias en las que solo una de las matrices es simétrica (Tai15b, Tai20b), la función objetivo f_3 es la que más correlación lineal positiva presenta con f_0, aunque con una importante variabilidad entre ambas instancias. Además, en estos casos las funciones f_1 y f_0 tienen un coeficiente de correlación lineal de 0 debido a que el valor de f_1 es constante.
- En las instancias asimétricas (Bur26a, Bur26b, Bur26c), los resultados obtenidos son muy similares en las tres instancias analizadas, siendo en los tres casos la

función objetivo f_3 la que más correlación lineal positiva presenta con f_0 ($r \approx 0{,}75$), seguida por f_2 ($r \approx 0{,}60$) y f_1 ($r \approx 0{,}18$).

Cabe destacar que la información que nos proporciona el coeficiente de correlación de Pearson entre las funciones objetivo es prácticamente equivalente a los resultados que se obtienen al calcular los coeficientes W_i utilizados para el cálculo de la auto-correlación [40] que se describen en la Ecuación 22 de [14] (Anexo B). Esto nos sugiere que el enfoque utilizado en este experimento es de utilidad a la hora de cuantificar la relación entre los *landscapes* de la descomposición y el *landscape* original del QAP.

4.3.2. Experimento 4: Porcentaje de óptimos locales compartidos

Una vez analizada la relación entre los valores de *fitness* en las distintas funciones objetivo, queda comprobar si las similitudes que hemos encontrado se mantienen cuando consideramos la estructura de vecindario. Con este objetivo, hemos calculado el porcentaje de óptimos locales de L_0 hallados en las búsquedas locales del primer experimento que también son óptimos locales de L_1, L_2 o L_3. Esta métrica nos sirve para hacernos una idea sobre qué *landscapes* comparten más regiones localmente óptimas con L_0 en cada tipo de instancia, algo que puede ser de interés a la hora de diseñar algoritmos de búsqueda local que resuelvan el QAP haciendo uso de los *landscapes* de la descomposición.

Experimentación y resultados

Los resultados muestrales utilizados en esta experimentación son los obtenidos en las búsquedas locales del primer experimento (Tabla 4.1). El porcentaje de óptimos locales de L_0 hallados en las búsquedas que también son óptimos locales de L_1, L_2 o L_3 se indica en la Tabla 4.6.

Tabla 4.6: Porcentaje de óptimos locales de L_0 hallados en las muestras (Tabla 4.1) que también son óptimos locales de L_1, L_2 o L_3. Se han omitido los casos en los que el porcentaje de óptimos locales estimado para todo el espacio de búsqueda es del 100,00 % (Tabla 4.2).

	Bur26a	Bur26b	Bur26c	Tai15b	Tai20b	Chr15a	Esc16a	Had20	Rou20
L_1	0,00 %	0,00 %	0,00 %	-	-	-	-	-	-
L_2	0,00 %	0,00 %	0,00 %	0,68 %	0,19 %	9,57 %	100,00 %	0,89 %	39,19 %
L_3	0,00 %	0,00 %	0,00 %	0,00 %	0,00 %	0,00 %	-	0,00 %	0,00 %

El único *landscape* que parece compartir un número significativo de los óptimos locales de L_0 es el *landscape* L_2. Esto se puede observar principalmente en las instancias con ambas matrices simétricas (Chr15a, Esc16a, Had20, Rou20), y en menor medida en las instancias en las que solo una de las matrices es simétrica (Tai15b, Tai20b). En las instancias asimétricas (Bur26a, Bur26b, Bur26c), sin embargo, ninguno de los *landscapes* de la descomposición parece compartir óptimos locales con L_0, ya que el porcentaje calculado es del $0,00\,\%$ en todos los casos.

Toda esta información también se puede apreciar de forma visual en la Figura 4.1 del tercer experimento. Si observamos los puntos verdes y rojos que simbolizan los 1,000 mejores óptimos locales de los distintos *landscapes*, podemos comprobar que en los casos en los que el porcentaje de óptimos locales compartidos es mayor los colores se solapan en algunas regiones de la gráfica, lo cuál significa que los óptimos locales de ambos *landscapes* tienen valores de *fitness* similares. Lo contrario ocurre en los casos en los que los *landscapes* no comparten óptimos locales, ya que en general en estos casos los puntos verdes y rojos están mucho más separados entre sí.

4.3.3. Conclusiones

- En las instancias en las que ambas matrices son simétricas (Chr15a, Esc16a, Had20, Rou20), el *landscape* L_2 parece ser el *landscape* de la descomposición que más influencia la estructura de L_0. Esto se ha comprobado desde dos puntos de vista distintos:

  1. En el tercer experimento, f_2 es la función objetivo que tiene mayor correlación lineal positiva con f_0.
  2. En el cuarto experimento, L_2 es el único *landscape* que comparte un porcentaje significativo de los óptimos locales de L_0.

  Esto confirma las sospechas indicadas en las conclusiones de la Sección 4.2, en las que mencionábamos que el número estimado de óptimos locales de L_0 y L_2 parecía indicar que ambos *landscapes* guardaban algún tipo de relación en este tipo de instancias. Como caso particular, es interesante destacar que en la instancia Esc16a la estructura de los *landscapes* L_0 y L_2 es prácticamente equivalente ($r = 1$ y $100,00\,\%$ de óptimos compartidos).

- En las instancias en las que solo una de las matrices es simétrica (Tai15b, Tai20b), nos encontramos con resultados diferentes dependiendo de si tenemos en cuenta

la estructura de vecindario o no. Por un lado, f_3 parece ser la función objetivo que más correlación lineal positiva tiene con f_0, mientras que por otro, el *landscape* L_2 es el único que parece compartir óptimos locales con L_0. De todos modos, el porcentaje estimado de óptimos compartidos es bastante reducido en ambos casos ($< 1,00\,\%$), por lo que sospechamos que esta propiedad no es tan relevante y se debe a alguna característica propia de Tai15b y Tai20b. Por lo tanto, sería recomendable considerar más instancias de este tipo para tener una visión más global de sus propiedades.

- En las instancias asimétricas (Bur26a, Bur26b, Bur26c), ninguno de los *landscapes* de la descomposición parece influenciar de forma predominante la estructura de L_0. Si bien las funciones objetivo f_2 y f_3 parecen tener una correlación lineal positiva significativa con f_0 ($r > 0{,}50$), cuando consideramos los óptimos locales de cada *landscape* no encontramos ninguna relación ($0,00\,\%$ de óptimos compartidos). Esto nos sugiere que este tipo de instancias podrían tener una estructura más compleja que las instancias simétricas.

5. CAPÍTULO

Algoritmos de optimización

Una vez analizadas las características de los *landscapes* elementales que se obtienen de la descomposición del QAP, en este capítulo desarrollamos varios algoritmos heurísticos de optimización [9] que tratan de dar un paso adelante en términos de rendimiento haciendo uso de lo aprendido en capítulos anteriores[1]. En concreto, proponemos dos métodos distintos: un algoritmo basado en la búsqueda local y un algoritmo evolutivo. En las siguientes secciones explicaremos en detalle cada uno de los algoritmos propuestos y realizaremos una comparación de rendimiento entre ellos mediante un análisis estadístico. Para mantener la coherencia a lo largo del documento, seguiremos utilizando la nomenclatura descrita al principio del Capítulo 4 para referirnos a los *landscapes* y sus componentes.

5.1. Algoritmo de búsqueda local

Los algoritmos de búsqueda local [10] son métodos heurísticos de optimización que recorren el espacio de búsqueda del problema utilizando la estructura de vecindario para pasar de una solución a otra. Una búsqueda local, por lo tanto, se puede definir de forma resumida como un camino $(x_0, x_1, x_2, ..., x_k)$ tal que para todo i se cumple $x_i \in \Omega$ y $x_{i+1} \in N(x_i)$, donde N es una función de vecindario (Sección 3.2) y x_i es la solución actual en la i-ésima iteración del algoritmo.

[1] El código de los algoritmos implementados para este capítulo se encuentra disponible en el siguiente enlace: https://bitbucket.org/Xab_Ben/algoritmos/src/master/.

Además de la función de vecindario, otra de las claves de este tipo de métodos es el criterio mediante el que decidimos a qué solución vecina movernos en cada paso del algoritmo. Dado que estamos tratando de optimizar, esta decisión se suele basar en el *fitness* de las soluciones candidatas. Suponiendo minimización, un ejemplo es el método *Iterative Best Improvement* (Algoritmo 1) que utilizamos para la estimación de óptimos locales de la Sub-Sección 4.2.1, en el cuál solo nos movemos de una solución $x \in \Omega$ a otra solución $y \in N(x)$ si $f(y) < f(x)$, es decir, si hay una mejora en el *fitness*. Este método es una de las búsquedas locales más básicas, y sirve como base para muchos otros algoritmos más complejos como el que vamos a proponer en esta sección.

5.1.1. Descripción del algoritmo desarrollado

El mayor problema que presentan los métodos de búsqueda local es que pueden quedarse atrapados en óptimos locales de poca calidad, por lo que una gran parte del trabajo a realizar a la hora de desarrollar algoritmos de este tipo se resume en encontrar estrategias que nos permitan escapar de los óptimos locales de manera eficiente para alcanzar soluciones con un mejor *fitness*. En este proyecto, hemos diseñado una estrategia de escape que se basa en las funciones objetivo de los *landscapes* de la descomposición. En concreto, el algoritmo que proponemos para optimizar el QAP se fundamenta en los siguientes pasos:

1. Aplicamos una búsqueda local básica (*Iterative Best Improvement*) en base a la función objetivo original f_0 partiendo de una solución $x \in \Omega$. De esta forma, hallamos una solución $x^* \in \Omega$ que es un óptimo local del problema para la función de vecindario escogida N.

2. Para escapar del óptimo local, exploramos el vecindario $N(x^*)$ en base a las diferentes funciones objetivo de la descomposición (f_1, f_2, f_3). En este punto hay dos opciones:

   a) Encontramos una solución vecina $y \in N(x^*)$ que mejora a x^* en al menos una de las funciones objetivo de la descomposición. Volvemos al paso 1 con $x = y$.

   b) No encontramos ninguna solución vecina que mejore a x^* en al menos una de las funciones objetivo de la descomposición. Paramos la búsqueda y devolvemos la mejor solución encontrada hasta el momento según f_0.

El motivo por el que nos hemos decidido por esta estrategia es que f_0 es una combinación lineal de f_1, f_2 y f_3, lo que implica que optimizar en una de las funciones objetivo de la descomposición se traduce en optimizar en uno de los sub-objetivos que forman el problema. Esta propiedad nos permite ampliar la búsqueda local a regiones prometedoras del espacio de búsqueda a las que no se llegaría considerando solo f_0, favoreciendo además una mayor diversidad en las soluciones que se visitan.

Por lo tanto, el método que proponemos se inspira en el *Variable Neighborhood Search* (VNS) [41], con la diferencia de que en este caso en lugar del vecindario lo que varía es la función objetivo con la que medimos el *fitness* de las soluciones. En base a esto, el algoritmo desarrollado se ha denominado *Variable Function Search* (VFS). Los detalles de este método se describen en el Algoritmo 2.

Algorithm 2 *Variable Function Search.*

Input:

x - Solución inicial ($x \in \Omega$) $\quad$ $\mathbf{F_1}, \ldots, \mathbf{F_{k_{max}}}$ - Conjunto de k_{max} funciones objetivo ($k_{max} \geq 1$)

N - Función de vecindario $\quad$ **Tabu** - Criterio tabú

Output:

$\mathbf{x}^*$ - Solución final ($x^* \in \Omega$)

```
1:  procedure VFS (x, F, N, Tabu)
2:      x* ← x
3:      k ← 1                          ▷ F_1 es la función objetivo principal de la búsqueda.
4:      while k ≤ k_max do
5:          Vecindario ← N(x)
6:          Mejora ← False
7:          for y ∈ Vecindario do
8:              if F_k(y) < F_k(x) ∧ ¬Tabu(y) then      ▷ Suponemos minimización.
9:                  x ← y
10:                 Mejora ← True
11:             end if
12:         end for
13:         if Mejora = True then
14:             k ← 1
15:             if F_1(x) < F_1(x*) then      ▷ Actualizamos mejor solución en base a F_1.
16:                 x* ← x
17:             end if
18:         else
19:             k ← k + 1
20:         end if
21:     end while
22:     return x*                      ▷ Mejor solución hallada en base a F_1.
23: end procedure
```

Tal y como se puede observar, el algoritmo VFS consta de dos tipos de funciones objetivo: la función principal que se pretende optimizar y una o más funciones auxiliares que se usan para escapar de los óptimos locales. Como hemos explicado previamente, en nuestro caso la función objetivo principal es la del *landscape* original del QAP (f_0), mientras que las funciones objetivo de los *landscapes* de la descomposición (f_1, f_2, f_3) sirven como funciones auxiliares. En cuanto a la función de vecindario, hemos considerado la función *swap* (Sección 3.2) ya que es la que se utiliza en la descomposición del problema.

Por último, cabe destacar que el VFS requiere un criterio tabú similar a los que se usan en el *Tabu Search* [42] que limite a qué soluciones puede moverse el algoritmo en cada paso. Esto se debe a que, como la función de vecindario no varía durante la ejecución, al escapar de un óptimo local x^* el algoritmo siempre se quedará en una solución vecina $y \in N(x^*)$, siendo probable que en la siguiente iteración vuelva a moverse a x^*. En consecuencia, el objetivo del criterio tabú es evitar este tipo de bucles impidiendo en la medida de lo posible volver a soluciones ya visitadas. En este proyecto, esto ha sido implementado mediante una lista tabú que guarda los movimientos de vecindario que han sido realizados más recientemente, los cuáles no podrán volver a realizarse hasta que salgan de la lista.

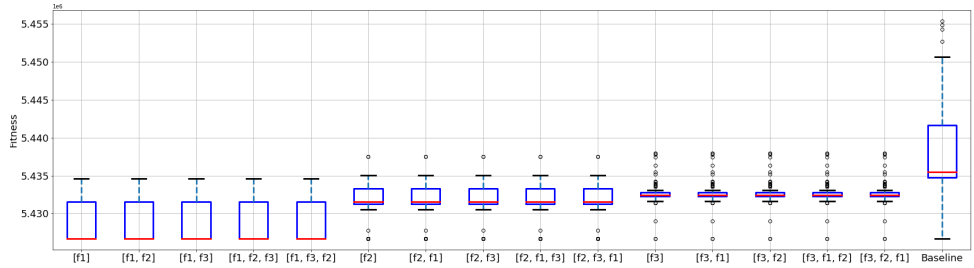
5.1.2. Experimentación y resultados

Con el objetivo de medir el rendimiento del algoritmo propuesto, hemos llevado a cabo una experimentación en las instancias descritas en la Sección 4.1. Dado que el rendimiento del VFS depende en gran medida de las funciones auxiliares que se utilizan para escapar de los óptimos locales, hemos probado todas las configuraciones que incluyan f_1, f_2 y/o f_3 para comprobar cuáles son más prometedoras. Además, hemos incluido un *Tabu Search* como algoritmo *baseline* cuyo criterio tabú es el mismo que el implementado en el VFS. Siendo n el tamaño de la instancia, el tamaño máximo de la lista tabú en ambos algoritmos se ha establecido a n.

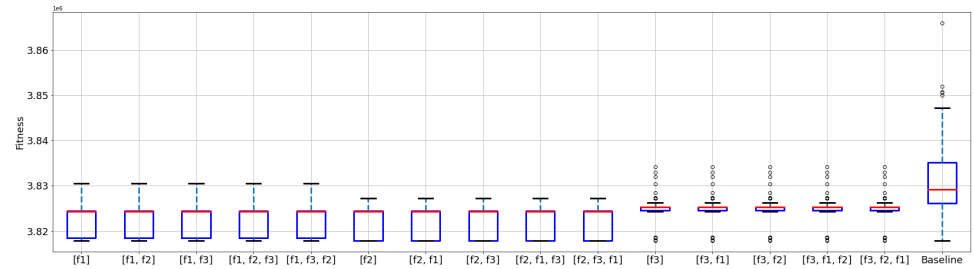
La experimentación consta de 100 repeticiones para cada escenario algoritmo-instancia con un máximo de $1000n^2$ evaluaciones por repetición. En la Tabla 5.1 se compara la mediana del *fitness* de las soluciones obtenidas en cada caso en forma de mapa de calor. De forma adicional, en la Figura 5.1 incluimos los *box-plots* calculados a partir de los resultados de las ejecuciones para tener una visión más detallada sobre el rendimiento de los algoritmos.

Tabla 5.1: Mediana del *fitness* en base a 100 repeticiones de los algoritmos. Los colores verdes indican los valores más bajos de cada instancia, mientras que los colores rojos indican los valores más altos. Se han omitido las configuraciones que incluyen f_1 en las instancias en las que alguna de las matrices es simétrica, ya que en dichos casos f_1 es constante. Además, también se indica el *fitness* de la mejor solución conocida para cada instancia [27] como punto de referencia.

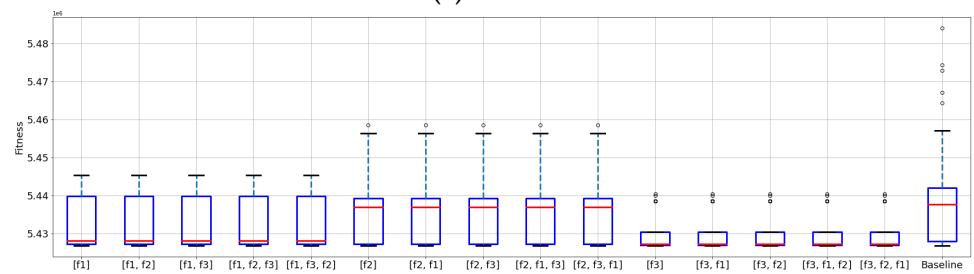
		Bur26a	Bur26b	Bur26c	Tai15b	Tai20b	Chr15a	Esc16a	Had20	Rou20
VFS	-	5442769	3830448.5	5444266	52029138.5	138362490	14389	70	6992	758659
	f1	5426670	3824376	5428202						
	f1, f2	5426670	3824376	5428202						
	f1, f2, f3	5426670	3824376	5428202						
	f1, f3	5426670	3824376	5428202						
	f1, f3, f2	5426670	3824376	5428202						
	f2	5431548	3824376	5436927	51981796	135766670	13348	70	6963	757029
	f2, f1	5431548	3824376	5436927						
	f2, f1, f3	5431548	3824376	5436927						
	f2, f3	5431548	3824376	5436927	51765268	123009513	9978	70	6948	728618
	f2, f3, f1	5431548	3824376	5436927						
	f3	5432449	3825291	5427227	51765268	123009513	9978	70	6948	728298
	f3, f1	5432449	3825291	5427227						
	f3, f1, f2	5432449	3825291	5427227						
	f3, f2	5432449	3825291	5427227	51765268	123009513	9978	70	6948	728298
	f3, f2, f1	5432449	3825291	5427227						
Baseline		5435461	3829239	5437623	51765268	135431383	9936	68	6948	726988
Mejor solución		5426670	3817852	5426795	51765268	122455319	9896	68	6922	725522



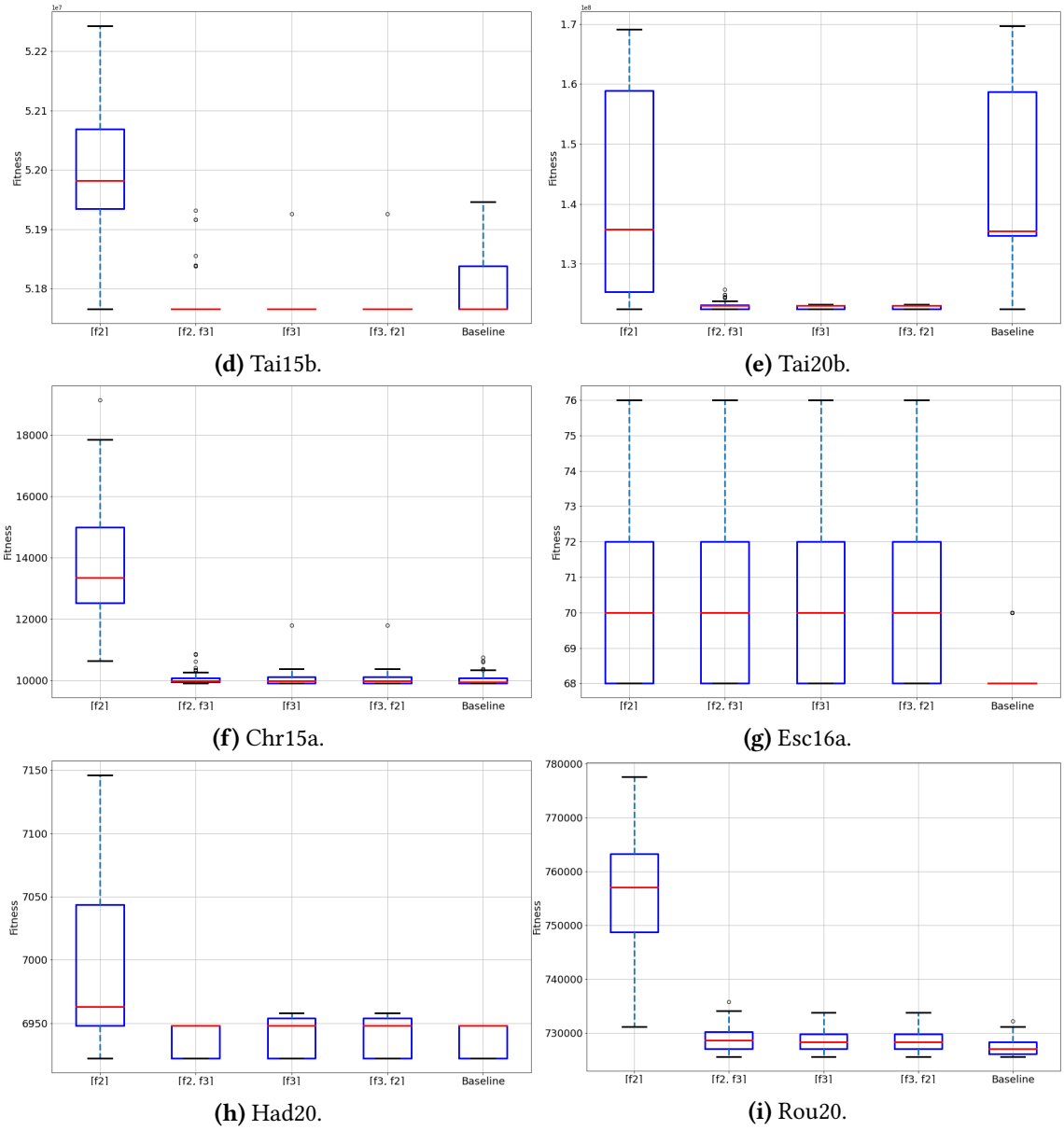
(a) Bur26a.



(b) Bur26b.



(c) Bur26c.



(d) Tai15b. **(e)** Tai20b.

(f) Chr15a. **(g)** Esc16a.

(h) Had20. **(i)** Rou20.

Figura 5.1: *Box-plots* generados a partir del *fitness* de las soluciones obtenidas en 100 repeticiones de los algoritmos. Se han omitido las configuraciones que incluyen la función f_1 en las instancias en las que alguna de las matrices es simétrica, ya que en dichos casos f_1 es constante.

Lo primero que podemos observar en los resultados es que las funciones auxiliares que se consideran en el VFS y el orden en el que estas se utilizan tiene un impacto reseñable en el rendimiento del algoritmo. En concreto, nos encontramos con varias situaciones distintas dependiendo de la simetría de las matrices que forman las instancias:

- En las instancias en las que alguna de las matrices es simétrica (Tai15b, Tai20b, Chr15a, Esc16a, Had20, Rou20), en general las configuraciones que mejor ren-

dimiento presentan son aquellas que incluyen la función f_3 entre sus funciones auxiliares. Además, tal y como se observa en los *box-plots*, estas configuraciones parecen presentar una menor dispersión en los resultados obtenidos.

- En las instancias asimétricas (Bur26a, Bur26b, Bur26c), no hemos encontrado un patrón claro sobre qué configuraciones son mejores, por lo que parece que en este tipo de instancias es necesario estudiar cada caso de forma independiente. De todas formas, parece que la función auxiliar que más determina el rendimiento del VFS es la primera de la configuración. En cuanto a la dispersión de los resultados obtenidos, las configuraciones del VFS que comienzan con la función f_3 son las que menos variabilidad presentan.

En cuanto a la comparación de rendimiento entre el VFS y el algoritmo *baseline*, el método propuesto parece ser mejor en los casos en los que alguna de las matrices es asimétrica (Bur26a, Bur26b, Bur26c, Tai15b, Tai20b). En las instancias en las que ambas matrices son simétricas (Chr15a, Esc16a, Had20, Rou20), sin embargo, los resultados del VFS son en general iguales o peores que los del *Tabu Search* independientemente de la configuración de funciones auxiliares que utilicemos.

5.1.3. Conclusiones

En base a los resultados obtenidos y considerando los conocimientos extraídos en el análisis del Capítulo 4, concluimos lo siguiente:

- Tomando como punto de referencia el algoritmo *baseline*, el VFS parece ser especialmente prometedor en aquellas instancias en las que alguna de las matrices es asimétrica (Bur26a, Bur26b, Bur26c, Tai15b, Tai20b).
- Las configuraciones de funciones auxiliares que mejores resultados presentan en el VFS parecen estar determinadas por las características de la instancia a resolver. En concreto, algunas de las características que más parecen influir son:
    - El porcentaje de los óptimos locales de L_0 que también son óptimos locales de los *landscapes* de la descomposición (Sub-sección 4.3.2). En general, cuanto menor sea dicho porcentaje para el *landscape* de la descomposición L_i, más eficaz resulta utilizar la función objetivo f_i para escapar de los óptimos locales, y viceversa. Esto se debe a que si nos encontramos con una solución

que es óptimo local tanto de L_0 como de L_i utilizar f_i como función auxiliar no nos servirá para escapar del óptimo local, por lo que cuanto menor sea la probabilidad de que esto ocurra mayor será la capacidad de escape del algoritmo. Como ejemplo, en el caso de las instancias en las que alguna de las matrices es simétrica (Tai15b, Tai20b, Chr15a, Esc16a, Had20, Rou20), el *landscape* que menor porcentaje de óptimos locales comparte con L_0 es el *landscape* L_3, lo que se corresponde con que las mejores configuraciones en este tipo de instancias sean las que incluyen f_3 como función auxiliar. En el caso de las instancias asimétricas (Bur26a, Bur26b, Bur26c), por otro lado, se estima que los tres *landscapes* de la descomposición tienen un $0,00\,\%$ de óptimos compartidos con L_0, lo que explicaría por qué no hay unanimidad en este tipo de instancias.

- El número de óptimos locales en el espacio de búsqueda (Sub-secciones 4.2.1 y 4.2.2). Como ya mencionamos al inicio del Capítulo 4, esta característica parece estar relacionada con la complejidad que entraña el proceso de búsqueda local, por lo que utilizar como función auxiliar la función objetivo de un *landscape* con pocos óptimos locales puede facilitar la convergencia del VFS. Sin embargo, esto también puede llevar a una menor diversidad en los resultados, ya que el proceso de búsqueda local se ve condicionado hacia las pocas soluciones localmente óptimas que se derivan de la función auxiliar. Sospechamos que es esto lo que ocurre en el caso de f_3, puesto que el *landscape* al que pertenece (L_3) es el que menos óptimos locales tiene en prácticamente todas las instancias analizadas.

5.2. Algoritmo evolutivo

Los algoritmos evolutivos [11] son métodos heurísticos de optimización que se inspiran en la teoría de la evolución de las especies propuesta por Charles Darwin. Una de las principales diferencias respecto a la búsqueda local es que los algoritmos evolutivos no mantienen una única solución actual, si no que trabajan sobre un conjunto de soluciones llamado población que es optimizado mediante un proceso iterativo que imita la evolución natural. Las variables que componen cada individuo en la población se conocen como genes. En la Figura 5.2 se muestra un ejemplo de estos conceptos para la codificación de soluciones que utilizamos en este proyecto.

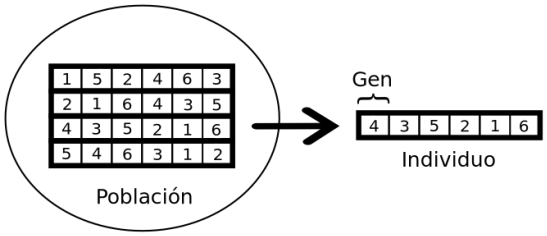


Figura 5.2: Ejemplo de gen, individuo y población con codificación basada en permutaciones en el contexto de un algoritmo evolutivo.

A la hora de optimizar la población, los algoritmos evolutivos hacen uso de un conjunto de operadores de búsqueda inspirados en procesos naturales. De forma general, los operadores más importantes son los siguientes:

- **Operadores de selección:** Seleccionan los individuos de la población que serán utilizados para crear nuevos individuos mediante criterios basados en mantener los genes más beneficiosos. Se inspiran en la selección natural.
- **Operadores de recombinación:** Combinan los genes de varios individuos seleccionados (denominados progenitores) para crear nuevos individuos. Se inspiran en el proceso de reproducción del mundo animal.
- **Operadores de mutación:** Modifican de forma aleatoria uno o más genes de los nuevos individuos para añadir diversidad al proceso de búsqueda. Se inspiran en la mutación genética.

Según las características de los operadores, los algoritmos evolutivos se pueden dividir en varias sub-categorías, entre las que se encuentran los algoritmos genéticos [43], la programación evolutiva [44], las estrategias de evolución [45][46] o los algoritmos de estimación de distribuciones [47].

5.2.1. Descripción del algoritmo desarrollado

A diferencia del algoritmo de búsqueda local, en este caso hemos tomado un algoritmo evolutivo ya existente y lo hemos adaptado para mejorar su rendimiento mediante el uso de la descomposición en *landscapes* elementales. En concreto, el algoritmo que hemos

elegido para este propósito es el *Multi-Objective Evolutionary Algorithm based on Decomposition* (MOEA/D) [48]. En esta sub-sección tan solo explicaremos los fundamentos básicos de dicho algoritmo, por lo que los detalles más específicos se pueden consultar en el artículo original.

El MOEA/D es un algoritmo para la resolución de problemas de optimización multi-objetivo (MOP), es decir, problemas en los que se dispone de un conjunto de funciones objetivo $\{f_1(x), f_2(x), ..., f_m(x)\}$ que se deben optimizar. Normalmente, como los sub-objetivos del problema se pueden contradecir, no suele existir una solución que sea óptima para todas las funciones de forma simultanea, por lo que la jerarquía entre las soluciones se suele medir en base al criterio de dominancia de Pareto. Suponiendo minimización, dadas dos soluciones $x_1, x_2 \in \Omega$, consideramos que x_1 domina a x_2 si y solo si $f_i(x_1) \leq f_i(x_2)$ para todo $1 \leq i \leq m$ y $f_j(x_1) < f_j(x_2)$ para al menos un índice $1 \leq j \leq m$. Toda solución $x^* \in \Omega$ que no sea dominada por ninguna otra solución en el espacio de búsqueda se considera óptimo de Pareto, y es por definición un óptimo global. El conjunto de todas las soluciones Pareto óptimas se denomina frente de Pareto (Figura 5.3).

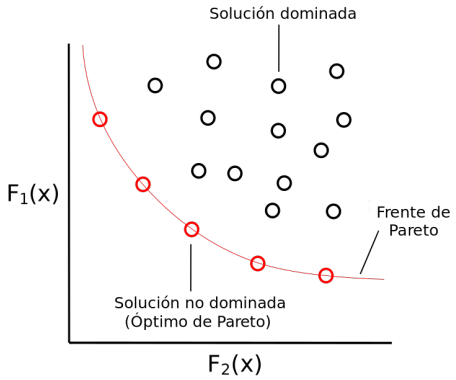


Figura 5.3: Ejemplo de frente de Pareto para un problema con dos funciones objetivo (F_1, F_2).

Uno de los problemas que presenta la dominancia de Pareto es que a priori no hay forma de ordenar la calidad de las soluciones que no se dominan entre sí. Esto también ocurre en el propio frente de Pareto, por lo que al resolver un MOP suele ser recomendable encontrar tantos óptimos de Pareto como sea posible para posteriormente elegir uno de ellos mediante criterios externos. En consecuencia, la meta de los algoritmos multi-objetivo no es encontrar un único óptimo global, sino una gran variedad de ellos.

Teniendo esto en cuenta, la principal particularidad que presenta el MOEA/D con respecto a otros algoritmos similares es que no considera el MOP como un único problema

con múltiples objetivos, si no que lo transforma en varios sub-problemas de un solo objetivo que se resuelven de forma simultanea. Si bien existen muchas maneras para generar estos sub-problemas a partir del MOP original, en este proyecto nos centraremos en el enfoque de Tchebycheff [49]. Según este enfoque, cada sub-problema se define como:

$$\underset{x\in\Omega}{\text{minimizar}} \quad g(x|\lambda, z^*) = \underset{1\leq i\leq m}{max}\{\lambda_i|f_i(x) - z_i^*|\} \tag{5.1}$$

donde $\lambda = (\lambda_1, \lambda_2, ..., \lambda_m)$ es un vector de pesos y $z^* = (z_1^*, z_2^*, ..., z_m^*)$ es un punto de referencia que se calcula[2] como $z_i^* = \text{mín}_{x\in\Omega} f_i(x)$. Para toda solución $x^* \in \Omega$ que sea parte del frente de Pareto existe un vector de pesos λ para el que el óptimo global del sub-problema es x^*. Además, el óptimo global de un sub-problema generado mediante esta técnica es siempre parte del frente de Pareto. Estas características nos permiten obtener diversos óptimos de Pareto mediante la resolución de sub-problemas mono-objetivo con distintos vectores de pesos.

El MOEA/D, por lo tanto, toma como punto de partida un conjunto de P vectores de pesos $\lambda^1, \lambda^2, ..., \lambda^P$ a partir de los cuales genera P sub-problemas mono-objetivo mediante la Ecuación 5.1. El algoritmo mantiene en cada iteración una población compuesta por P individuos que representan la mejor solución encontrada hasta el momento para cada uno de los sub-problemas. Esta población va mejorando iteración a iteración mediante operadores de recombinación y mutación como los utilizados en otros algoritmos evolutivos, con la diferencia de que la selección de los individuos para cada sub-problema p se realiza escogiendo las soluciones actuales de los T sub-problemas más cercanos a p con respecto a la distancia euclídea entre los vectores de pesos. Dicho conjunto de sub-problemas se conoce como vecindario de p. Dado que los sub-problemas con vectores de pesos similares también suelen tener óptimos globales similares, esta estrategia nos permite optimizar de forma más eficiente cada uno de los individuos de la población. Para más información sobre el funcionamiento del MOEA/D, se recomienda consultar el pseudocódigo de la Sección III.A de [48].

Como hemos explicado anteriormente, el MOEA/D solo sirve para problemas con múltiples objetivos, por lo que no se puede aplicar al QAP en su forma original. Sin embargo, si bien inicialmente el QAP consta de un solo objetivo, es posible convertirlo en un MOP en base a la descomposición en *landscapes* elementales del problema [38]. Teniendo en cuenta que $f_0 = f_1 + f_2 + f_3$, el QAP puede ser considerado como un MOP en el que las

[2]Calcular z^* de forma exacta es computacionalmente inasumible. Una alternativa es inicializar z^* de forma arbitraria e ir actualizando z_i^* con los mejores valores de *fitness* que se hallen durante la búsqueda.

funciones objetivo son los tres componentes de la suma: f_1, f_2 y f_3. De esta forma, es posible aplicar el MOEA/D. Este proceso en el que convertimos un problema mono-objetivo en multi-objetivo se denomina multi-objetivización [50].

Para resolver esta versión multi-objetivo del QAP, el MOEA/D ha sido implementado con los operadores de recombinación y mutación que se utilizan en [38]:

- **Recombinación:** Operador propuesto en la Figura 3 de [51].
- **Mutación:** Intercambio de dos genes (*swap*).

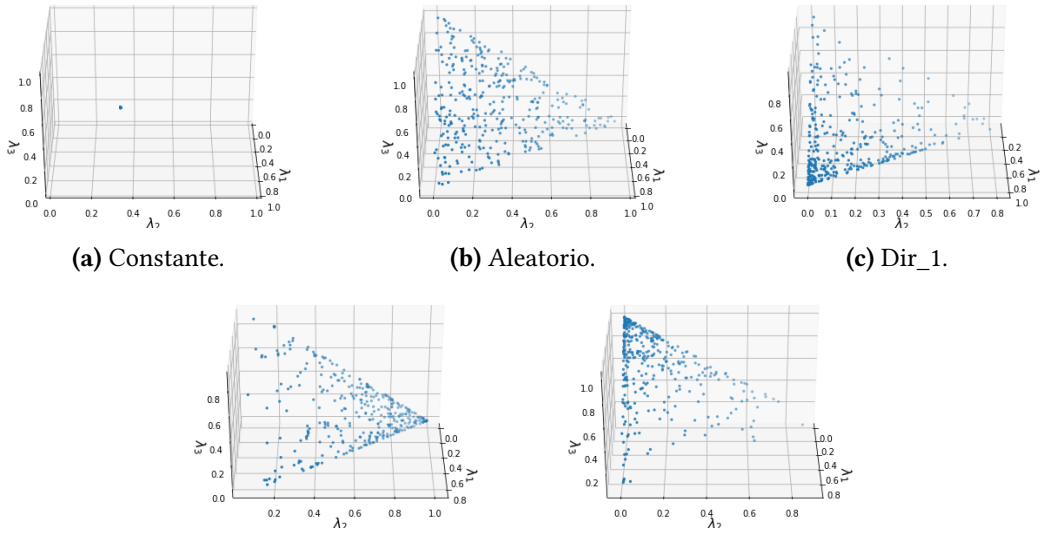
Puesto que en el QAP conocemos el *fitness* general de cada solución, el MOEA/D implementado devuelve la mejor solución encontrada durante la búsqueda en base a f_0.

5.2.2. Experimentación y resultados

Con el objetivo de medir el rendimiento del algoritmo propuesto, hemos llevado a cabo una experimentación en las instancias descritas en la Sección 4.1. Dado que dependiendo de las características de la instancia las funciones f_1, f_2 y f_3 tienen un nivel de correlación distinto con f_0 (Sub-sección 4.3.1), hemos decidido considerar varias estrategias de generación de vectores de pesos que den más importancia a una función objetivo o a otra. Para simplificar, hemos supuesto que la suma de los coeficientes de los vectores de pesos es siempre $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

- **Constante:** Los coeficientes de los vectores tienen el mismo valor ($\lambda_1 = \lambda_2 = \lambda_3$).
- **Aleatorio:** Los vectores se muestrean de una distribución de Dirichlet [52] con $\alpha_1 = 1$, $\alpha_2 = 1$, $\alpha_3 = 1$. Representa una distribución uniforme.
- **Dir_1:** Los vectores se muestrean de una distribución de Dirichlet con $\alpha_1 = 2$, $\alpha_2 = 0{,}5$, $\alpha_3 = 0{,}5$. El sub-objetivo f_1 tiene mayor importancia.
- **Dir_2:** Los vectores se muestrean de una distribución de Dirichlet con $\alpha_1 = 0{,}5$, $\alpha_2 = 2$, $\alpha_3 = 0{,}5$. El sub-objetivo f_2 tiene mayor importancia.
- **Dir_3:** Los vectores se muestrean de una distribución de Dirichlet con $\alpha_1 = 0{,}5$, $\alpha_2 = 0{,}5$, $\alpha_3 = 2$. El sub-objetivo f_3 tiene mayor importancia.

En la Figura 5.4 se representa gráficamente la forma en la que se distribuye una muestra de vectores de pesos generada mediante cada una de las cinco estrategias utilizadas. Las coordenadas de cada punto en las gráficas vienen determinadas por los coeficientes de un vector distinto. De esta forma, podemos observar que en las estrategias Dir_1, Dir_2 y Dir_3 la mayoría de puntos se agrupan en los vértices del triángulo, mientras que en la estrategia aleatoria los puntos se distribuyen de manera mucho más uniforme. Por último, en el caso de la estrategia constante todos los puntos se agrupan en el mismo sitio ya que todos los vectores tienen los mismos coeficientes.



(a) Constante. **(b)** Aleatorio. **(c)** Dir_1.

(d) Dir_2. **(e)** Dir_3.

Figura 5.4: Representación gráfica de muestras de vectores de pesos generadas mediante las cinco estrategias consideradas en esta experimentación. El eje Z de las gráficas se corresponde con el valor de λ_1, el eje X con el valor de λ_2 y el eje Y con el valor de λ_3.

De forma adicional, hemos incluido un algoritmo genético (GA) básico en la experimentación como algoritmo *baseline*. Los operadores de recombinación y mutación de dicho algoritmo son los mismos que hemos utilizado en el MOEA/D, mientras que el operador de selección es el torneo binario [53][38]. Siendo n el tamaño de la instancia, los parámetros que hemos considerado para los métodos evaluados son los siguientes[3]:

- **MOEA/D:**
  - Cantidad de sub-problemas (P): 351

[3]Los valores de los parámetros han sido escogidos en base a los utilizados en [48] y [38] con la intención de realizar una comparación lo más justa posible.

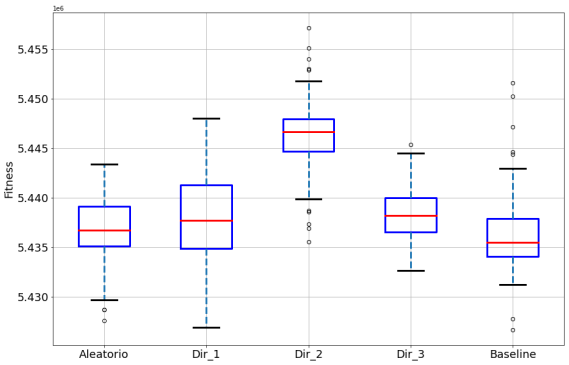
- Tamaño del vecindario de los sub-problemas (T): 10
- Tasa de mutación: 1,0

- **GA (*Baseline*):**
  - Tamaño de la población: 351
  - Tamaño de la selección: 351
  - Tamaño de la descendencia: 351
  - Tasa de mutación: 1,0

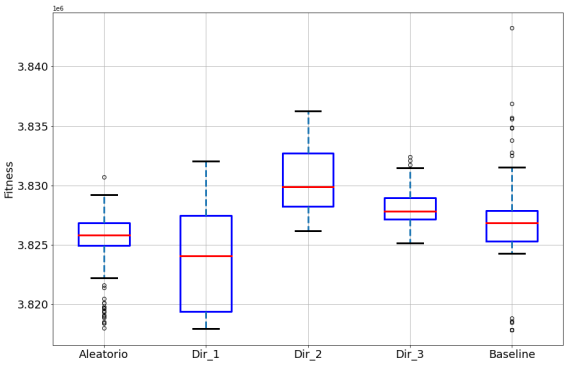
La experimentación consta de 100 repeticiones para cada escenario algoritmo-instancia con un máximo de $1000n^2$ evaluaciones por repetición. En la Tabla 5.2 se compara la mediana del *fitness* de las soluciones obtenidas en cada caso en forma de mapa de calor. De forma adicional, en la Figura 5.5 incluimos los *box-plots* calculados a partir de los resultados de las ejecuciones para tener una visión más detallada sobre el rendimiento de los algoritmos.

Tabla 5.2: Mediana del *fitness* en base a 100 repeticiones de los algoritmos. Los colores verdes indican los valores de *fitness* más bajos de cada instancia, mientras que los colores rojos indican los valores más altos. Se ha omitido la estrategia Dir_1 en las instancias en las que alguna de las matrices es simétrica, ya que en dichos casos f_1 es constante. Además, también se indica el *fitness* de la mejor solución conocida para cada instancia [27] como punto de referencia.

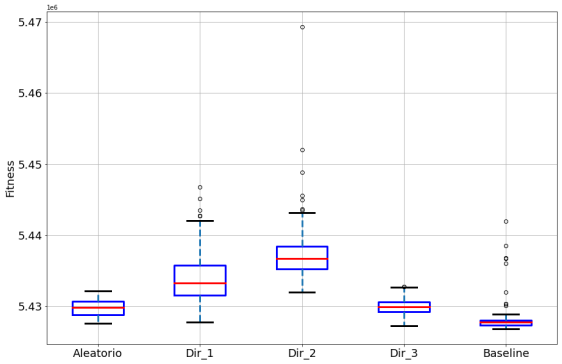
		Bur26a	Bur26b	Bur26c	Tai15b	Tai20b	Chr15a	Esc16a	Had20	Rou20
MOEA/D	Constante	5478769.5	3858009.5	5490631	52070629	147971660	15248	70	6987	761646
	Aleatorio	5436720.5	3825857.5	5429812.5	51765268	123143224	10092	68	6922	731668
	Dir_1	5437738.5	3824085.5	5433295						
	Dir_2	5446688.5	3829886	5436687	51838891	123159510	10644	68	6922	735491
	Dir_3	5438239	3827848.5	5429927	51875006	123326074	10106	68	6922	734337
Baseline		5435494	3826845	5427735	51765268	123009513	10455	68	6948	734865
Mejor solución		5426670	3817852	5426795	51765268	122455319	9896	68	6922	725522



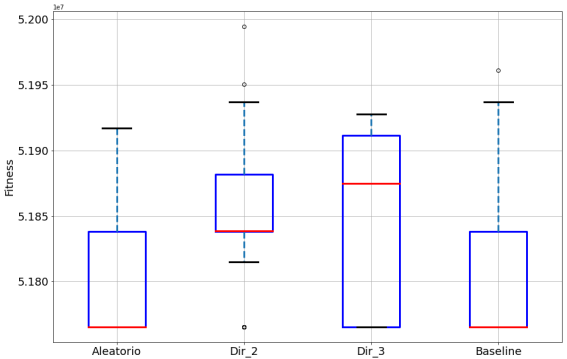
(a) Bur26a.



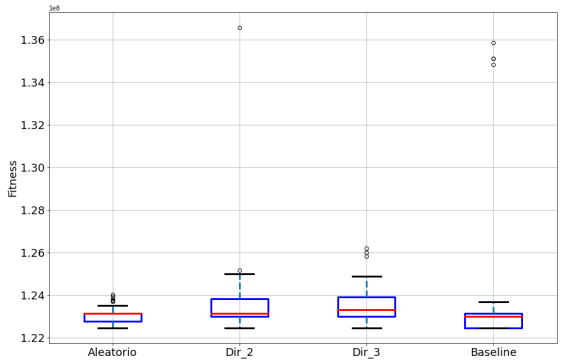
(b) Bur26b.



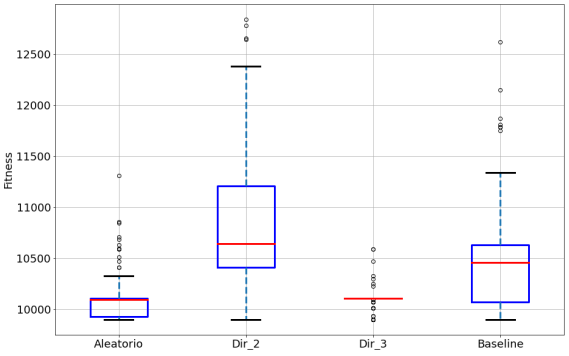
(c) Bur26c.



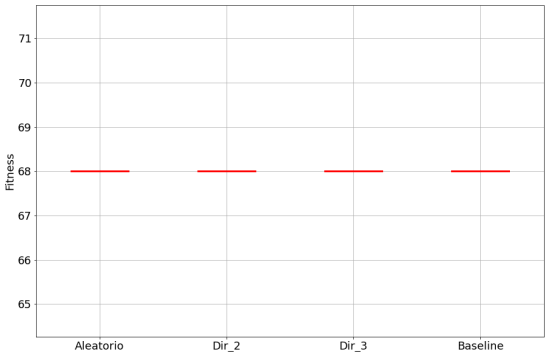
(d) Tai15b.



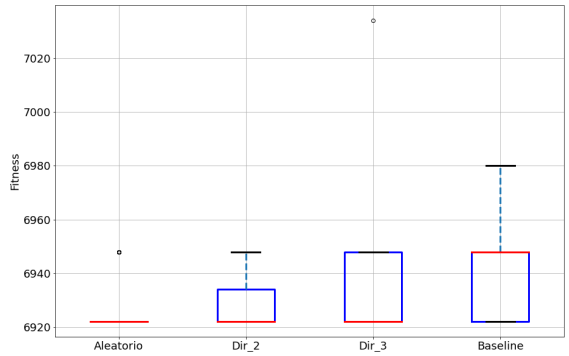
(e) Tai20b.



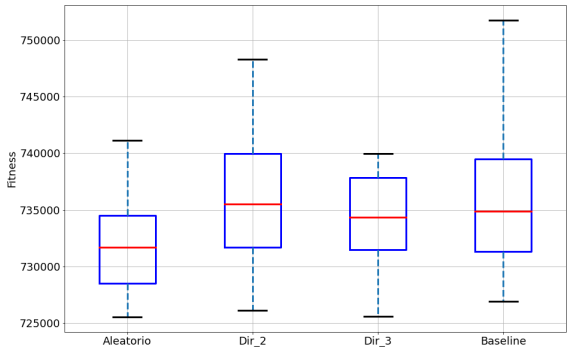
(f) Chr15a.



(g) Esc16a.



(h) Had20.



(i) Rou20.

Figura 5.5: *Box-plots* generados a partir del *fitness* de las soluciones obtenidas en 100 repeticiones de los algoritmos. Se ha omitido la estrategia Dir_1 en las instancias en las que alguna de las matrices es simétrica, ya que en dichos casos f_1 es constante.

Lo primero que podemos observar en los resultados es que las características de los vectores de pesos que se utilizan para generar los sub-problemas en el MOEA/D tienen un impacto reseñable en el rendimiento del algoritmo. Las estrategias de generación de vectores de pesos que se basan en dar más importancia a una función objetivo concreta no parecen ser muy efectivas, ya que en prácticamente todas las instancias la estrategia que ofrece mejores resultados es la uniformemente aleatoria.

En cuanto a la comparación de rendimiento entre el MOEA/D y el algoritmo *baseline*, el método multi-objetivo parece ser mejor en los casos en los que ambas matrices son simétricas (Chr15a, Esc16a, Had20, Rou20). En las instancias en las que alguna de las matrices es asimétrica (Bur26a, Bur26b, Bur26c, Tai15b, Tai20b), sin embargo, los resultados del MOEA/D son en general iguales o peores que los del GA independientemente de la estrategia de generación de vectores de pesos que utilicemos, siendo la única excepción la instancia Bur26b.

5.2.3. Conclusiones

En base a los resultados obtenidos y considerando los conocimientos extraídos en el análisis del Capítulo 4, concluimos los siguiente:

- Tomando como punto de referencia el algoritmo *baseline*, el MOEA/D parece ser especialmente prometedor en aquellas instancias en las que ambas matrices son simétricas (Chr15a, Esc16a, Had20, Rou20).

- No hemos encontrado una relación clara entre el rendimiento de las estrategias de generación de vectores de pesos y el nivel de correlación entre las funciones objetivo (Sub-sección 4.3.1). Sin embargo, es importante destacar que los parámetros de las distribuciones que hemos utilizado en Dir_1, Dir_2 y Dir_3 han sido elegidos de forma arbitraria, por lo que para poder sacar conclusiones más precisas sería recomendable ajustar dichos parámetros.

- A falta de un análisis más exhaustivo, parece que lo más prometedor es utilizar conjuntos de vectores de pesos uniformemente distribuidos para crear los sub-problemas, tal y como se recomienda en [48]. Esto puede deberse a que esta estrategia produce conjuntos de sub-problemas más diversos que las demás, lo que permite al algoritmo explorar una mayor parte del espacio de búsqueda.

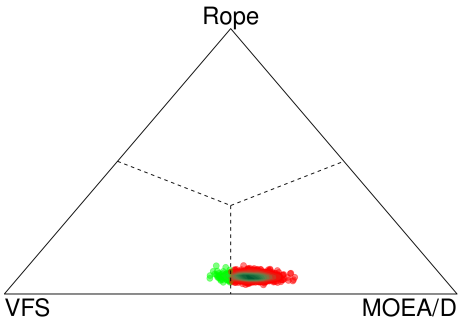
5.3. Análisis estadístico de los resultados experimentales

A la hora de comparar los algoritmos desarrollados hemos separado las instancias en tres grupos: por un lado las instancias en las que ambas matrices son simétricas (Chr15a, Esc16a, Had20, Rou20), por otro las instancias en las que solo una de las matrices es simétrica (Tai15b, Tai20b) y por último las instancias asimétricas (Bur26a, Bur26b, Bur26c). De esta forma, en vez de realizar un único análisis estadístico global hemos llevado a cabo tres análisis distintos, uno por cada grupo de instancias. Esto nos ha permitido comparar los resultados de los algoritmos en los tres escenarios de forma independiente, algo que nos interesa especialmente ya que el rendimiento de los algoritmos parece variar según la simetría de las matrices que forman las instancias. Teniendo esto en cuenta, el método que hemos utilizado para el análisis estadístico es el *Bayesian Signed-Rank Test* [54][55], el cuál es el equivalente Bayesiano del test de Wilcoxon [56]. La implementación utilizada está disponible en el paquete de R *scmamp* [57].

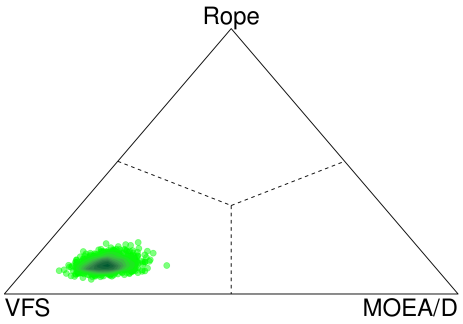
Como ya hemos mencionado anteriormente, los algoritmos desarrollados dependen en gran medida de dos factores: las funciones auxiliares en el caso del VFS y la estrategia de generación de vectores de pesos en el caso del MOEA/D. Dado que queremos llevar a cabo una comparación lo más justa posible, hemos considerado las estrategias más prometedoras para cada tipo de instancia según las experimentaciones de las Sub-secciones 5.1.2 y 5.2.2:

- **Ambas matrices simétricas:**
  - Funciones auxiliares (VFS): $[f_3]$.
  - Generación de vectores de pesos (MOEA/D): Aleatorio.
- **Una matriz simétrica:**
  - Funciones auxiliares (VFS): $[f_3]$.
  - Generación de vectores de pesos (MOEA/D): Aleatorio.
- **Ambas matrices asimétricas:**
  - Funciones auxiliares (VFS): $[f_1]$.
  - Generación de vectores de pesos (MOEA/D): Aleatorio.

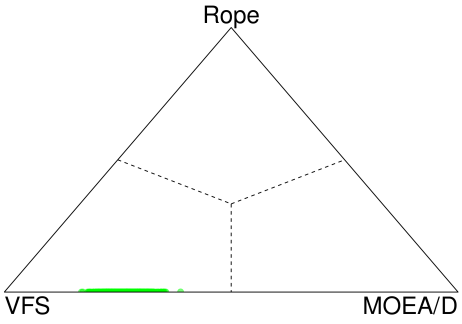
El análisis estadístico ha sido realizado sobre 100 repeticiones por instancia y método extraídas de las experimentaciones de las Sub-secciones 5.1.2 y 5.2.2. Como cada instancia dispone de escalas de *fitness* muy distintas, hemos utilizado los errores relativos respecto a la mejor solución conocida [27] para obtener resultados comparables. El método utilizado requiere la definición de lo que entendemos como *region of practical equivalence* (*rope*), es decir, el intervalo en el que los resultados de dos algoritmos se consideran prácticamente equivalentes. En nuestro caso, consideramos que el rendimiento de los algoritmos es equivalente cuando la diferencia en el error relativo con respecto a la mejor solución conocida es inferior a 10^{-6}. Tomando esto en cuenta, en la Figura 5.6 se muestra un resumen de los resultados del análisis estadístico para cada grupo de instancias.



(a) Ambas matrices simétricas.



(b) Una matriz simétrica.



(c) Ambas matrices asimétricas.

Figura 5.6: Resultados del *Bayesian Signed-Rank Test* mostrados en forma de diagramas ternarios.

De forma resumida, los puntos en las gráficas representan un muestreo de la distribución a posteriori de la probabilidad de ganar, perder o empatar. Es decir, cuanto más cerca estén los puntos del vértice MOEA/D, mayor probabilidad habrá de que el MOEA/D produzca mejores resultados, y vice-versa. Lo mismo ocurre con los vértices VFS y *rope*, siendo este último el que simboliza el empate. Por lo tanto, las regiones delimitadas por las líneas intermitentes representan las áreas de dominancia, esto es, las áreas donde la mayor probabilidad corresponde a su respectivo vértice. En base a esto, las gráficas muestran escenarios muy distintos en los tres grupos de instancias:

- En las instancias en las que ambas matrices son simétricas (Chr15a, Esc16a, Had20, Rou20), la mayoría de los puntos se encuentran en el área de dominancia del MOEA/D. Si calculamos la esperanza matemática de la probabilidad a posteriori de cada caso, obtenemos que hay un 0,511 de probabilidades de que el MOEA/D sea mejor, un 0,065 de que ambos algoritmos sean equivalentes y un 0,424 de que el VFS sea mejor. Por lo tanto, parece que el MOEA/D es algo más prometedor en este tipo de instancias, aunque el VFS también resulta competente.

- En las instancias en las que solo una de las matrices es simétrica (Tai15b, Tai20b), todos los puntos se encuentran en el área de dominancia del VFS. Si calculamos la esperanza matemática de la probabilidad a posteriori de cada caso, obtenemos que hay un 0,168 de probabilidades de que el MOEA/D sea mejor, un 0,113 de que ambos algoritmos sean equivalentes y un 0,719 de que el VFS sea mejor. Por lo tanto, parece que el VFS es mucho más prometedor en este tipo de instancias, teniendo una probabilidad de ser mejor casi cinco veces mayor que el MOEA/D.

- En las instancias asimétricas (Bur26a, Bur26b, Bur26c), todos los puntos se encuentran en el área de dominancia del VFS. Si calculamos la esperanza matemática de la probabilidad a posteriori de cada caso, obtenemos que hay un 0,262 de probabilidades de que el MOEA/D sea mejor, un 0,001 de que ambos algoritmos sean equivalentes y un 0,737 de que el VFS sea mejor. Por lo tanto, parece que el VFS es más prometedor en este tipo de instancias, teniendo una probabilidad de ser mejor casi tres veces mayor que el MOEA/D.

6. CAPÍTULO

Conclusiones y futuro trabajo

En este proyecto nos hemos centrado en analizar las características de la descomposición en *landscapes* elementales del QAP y sus posibles aplicaciones a la hora de desarrollar algoritmos heurísticos. Durante la consecución de estos objetivos hemos extraído varias conclusiones generales que creemos que es importante destacar:

1. La simetría de las matrices de distancia y flujo que forman las instancias del QAP tiene mucha influencia en los *landscapes* de la descomposición. Si bien no es el único factor a tener en cuenta, sí que es importante considerarlo a la hora de estudiar la descomposición en *landscapes* elementales del problema.

2. Se ha demostrado formalmente que la función objetivo del *landscape* L_1 es constante en las instancias en las que al menos una de las matrices (distancia o flujo) es simétrica. Por lo tanto, en dichos casos el problema se resume en optimizar la suma de dos sub-objetivos en lugar de tres.

3. La descomposición en *landscapes* elementales del QAP puede ser utilizada para desarrollar algoritmos heurísticos específicos para este problema. De hecho, hemos propuesto dos ejemplos muy diferentes (una búsqueda local y un algoritmo evolutivo) que han demostrado poder obtener mejores resultados que otros algoritmos más básicos en algunas instancias.

4. En relación con la primera conclusión, la simetría de las matrices que forman las instancias del QAP también parece afectar al rendimiento de los algoritmos, por lo

que puede ser buena idea tener en cuenta esta característica a la hora de decidirse por un método u otro.

Sin embargo, el trabajo realizado cuenta con ciertas limitaciones y posibles mejoras que sería interesante considerar en un futuro para validar y extrapolar todo lo observado en este proyecto. En concreto, los puntos que se deberían estudiar en más profundidad son los siguientes:

1. Ampliar la muestra de instancias considerada en las experimentaciones. Debido a las limitaciones temporales de este proyecto la muestra de instancias utilizada ha sido demasiado pequeña como para extraer conclusiones firmes, por lo que sería interesante comprobar qué ocurre si aumentamos el tamaño de la muestra. Con este objetivo, se proponen dos estrategias:

   a) Considerar instancias más grandes ($n > 30$). De este modo, podremos comprobar si las características y patrones que hemos observado en las instancias más pequeñas pueden extrapolarse a instancias más grandes y complejas.

   b) Generar instancias aleatorias. Esta estrategia evitará que las conclusiones que extraigamos se vean sesgadas o condicionadas por las características de los *benchmark* a los que pertenecen las instancias. Tal y como se ha observado en trabajos previos [58], la correcta generación de instancias aleatorias no es un proceso trivial, por lo que habrá que prestar especial atención al desarrollo de este aspecto.

2. Estudiar la complejidad de los *landscapes* de la descomposición de forma más detallada. Si bien en este proyecto solo hemos analizado el número de óptimos locales, existen muchas más medidas que tratan de cuantificar la complejidad de los problemas de optimización combinatoria desde diversos puntos de vista [59].

3. Optimizar los parámetros de los algoritmos desarrollados mediante técnicas de *Parameter Tuning* [60]. Esto incluye las funciones auxiliares en el caso del VFS y la estrategia de generación de vectores de pesos en el caso del MOEA/D, ya que son dos elementos decisivos para el rendimiento de los algoritmos.

4. En el caso del VFS, estudiar y aplicar las ventajas que nos ofrecen los *landscapes* elementales a la hora de realizar búsquedas locales. Por ejemplo, un aspecto interesante es que podemos conocer el *fitness* medio de las soluciones vecinas basándonos en el *fitness* de la solución actual, algo que podría ser aprovechado para mejorar el VFS y volverlo más eficiente.

5. Analizar los motivos por los que los algoritmos desarrollados funcionan mejor en un tipo de instancias o en otro. Aunque esto ha sido observado de forma experimental, no hemos sido capaces de descubrir a qué se debe. Por lo tanto, sería interesante estudiar este aspecto desde un punto de vista más teórico.

El trabajo realizado en este proyecto es, en definitiva, una introducción a las posibilidades que nos ofrece la descomposición en *landscapes* elementales del QAP. Por lo tanto, sería interesante continuar con esta línea de investigación en un futuro, ya que ha demostrado tener potencial tanto teórico como práctico.

A. ANEXO

Función objetivo de L_1 constante

En este anexo demostramos formalmente que la función objetivo del *landscape* L_1 es constante cuando al menos una de las dos matrices que forman la instancia es simétrica respecto a la diagonal principal. Comenzamos considerando la función objetivo de L_1:

$$f_1(x) = \sum_{\substack{i,j,p,q=1 \\ i \neq j \\ p \neq q}}^{n} d_{i,j} h_{p,q} \frac{\phi(x)^1_{(i,j)(p,q)}}{2n} \tag{A.1}$$

donde $d_{i,j}$ es la posición (i,j) de la matriz de distancia $D_{n\times n}$, $h_{p,q}$ es la posición (p,q) de la matriz de flujo $H_{n\times n}$ y la función $\phi(x)^1_{(i,j)(p,q)}$ tiene los siguientes posibles valores:

$$\phi(x)^1_{(i,j)(p,q)} = \begin{cases} n-3 & \text{si } x(i) = p \wedge x(j) = q \\ 1-n & \text{si } x(i) = q \wedge x(j) = p \\ -2 & \text{si } x(i) = p \oplus x(j) = q \\ 0 & \text{si } x(i) = q \oplus x(j) = p \\ -1 & \text{si } x(i) \neq p,q \wedge x(j) \neq p,q \end{cases} \tag{A.2}$$

con $i \neq j$, $p \neq q$ y $1 \leq i,j,p,q \leq n$. Teniendo esto en cuenta, reescribimos la Ecuación A.1 de la siguiente forma:

$$f_1(x) = \sum_{a=1}^{n-1} \sum_{b=a+1}^{n} \sum_{c=1}^{n-1} \sum_{d=c+1}^{n} g(x)_{(a,b),(c,d)} \tag{A.3}$$

donde $g(x)_{(a,b),(c,d)} = \left(d_{a,b}h_{c,d}\frac{\phi(x)^1_{(a,b)(c,d)}}{2n} + d_{b,a}h_{c,d}\frac{\phi(x)^1_{(b,a)(c,d)}}{2n} + d_{a,b}h_{d,c}\frac{\phi(x)^1_{(a,b)(d,c)}}{2n} + d_{b,a}h_{d,c}\frac{\phi(x)^1_{(b,a)(d,c)}}{2n}\right)$. Si podemos demostrar que $g(x)_{(a,b),(c,d)}$ siempre tiene el mismo valor independientemente de x, entonces queda demostrado que f_1 es una función constante. Para ello, analizamos primero los posibles casos de $g(x)_{(a,b),(c,d)}$:

1. **$x(a) = c \wedge x(b) = d$:** Si analizamos cada uno de los cuatro términos de $g(x)_{(a,b),(c,d)}$ por separado obtenemos que:

Término	**Configuración**	**Caso**	**Valor**
$d_{a,b}h_{c,d}\frac{\phi(x)^1_{(a,b)(c,d)}}{2n}$	$i=a,\ j=b,\ p=c,\ q=d$	$x(i)=p \wedge x(j)=q$	$d_{a,b}h_{c,d}\frac{n-3}{2n}$
$d_{b,a}h_{c,d}\frac{\phi(x)^1_{(b,a)(c,d)}}{2n}$	$i=b,\ j=a,\ p=c,\ q=d$	$x(i)=q \wedge x(j)=p$	$d_{b,a}h_{c,d}\frac{1-n}{2n}$
$d_{a,b}h_{d,c}\frac{\phi(x)^1_{(a,b)(d,c)}}{2n}$	$i=a,\ j=b,\ p=d,\ q=c$	$x(i)=q \wedge x(j)=p$	$d_{a,b}h_{d,c}\frac{1-n}{2n}$
$d_{b,a}h_{d,c}\frac{\phi(x)^1_{(b,a)(d,c)}}{2n}$	$i=b,\ j=a,\ p=d,\ q=c$	$x(i)=p \wedge x(j)=q$	$d_{b,a}h_{d,c}\frac{n-3}{2n}$

   donde la suma de los cuatro términos es:

$$g(x)_{(a,b),(c,d)} = d_{a,b}h_{c,d}\frac{n-3}{2n} + d_{b,a}h_{c,d}\frac{1-n}{2n} + d_{a,b}h_{d,c}\frac{1-n}{2n} + d_{b,a}h_{d,c}\frac{n-3}{2n} \quad \text{(A.4)}$$

2. **$x(a) = d \wedge x(b) = c$:** Si analizamos cada uno de los cuatro términos de $g(x)_{(a,b),(c,d)}$ por separado obtenemos que:

Término	**Configuración**	**Caso**	**Valor**
$d_{a,b}h_{c,d}\frac{\phi(x)^1_{(a,b)(c,d)}}{2n}$	$i=a,\ j=b,\ p=c,\ q=d$	$x(i)=q \wedge x(j)=p$	$d_{a,b}h_{c,d}\frac{1-n}{2n}$
$d_{b,a}h_{c,d}\frac{\phi(x)^1_{(b,a)(c,d)}}{2n}$	$i=b,\ j=a,\ p=c,\ q=d$	$x(i)=p \wedge x(j)=q$	$d_{b,a}h_{c,d}\frac{n-3}{2n}$
$d_{a,b}h_{d,c}\frac{\phi(x)^1_{(a,b)(d,c)}}{2n}$	$i=a,\ j=b,\ p=d,\ q=c$	$x(i)=p \wedge x(j)=q$	$d_{a,b}h_{d,c}\frac{n-3}{2n}$
$d_{b,a}h_{d,c}\frac{\phi(x)^1_{(b,a)(d,c)}}{2n}$	$i=b,\ j=a,\ p=d,\ q=c$	$x(i)=q \wedge x(j)=p$	$d_{b,a}h_{d,c}\frac{1-n}{2n}$

   donde la suma de los cuatro términos es:

$$g(x)_{(a,b),(c,d)} = d_{a,b}h_{c,d}\frac{1-n}{2n} + d_{b,a}h_{c,d}\frac{n-3}{2n} + d_{a,b}h_{d,c}\frac{n-3}{2n} + d_{b,a}h_{d,c}\frac{1-n}{2n} \quad \text{(A.5)}$$

3. **$x(a) = c \oplus x(b) = d$:** Si analizamos cada uno de los cuatro términos de $g(x)_{(a,b),(c,d)}$ por separado obtenemos que:

Término	Configuración	Caso	Valor
$d_{a,b}h_{c,d}\frac{\phi(x)^1_{(a,b)(c,d)}}{2n}$	$i=a,\ j=b,\ p=c,\ q=d$	$x(i)=p\oplus x(j)=q$	$d_{a,b}h_{c,d}\frac{-2}{2n}$
$d_{b,a}h_{c,d}\frac{\phi(x)^1_{(b,a)(c,d)}}{2n}$	$i=b,\ j=a,\ p=c,\ q=d$	$x(i)=q\oplus x(j)=p$	0
$d_{a,b}h_{d,c}\frac{\phi(x)^1_{(a,b)(d,c)}}{2n}$	$i=a,\ j=b,\ p=d,\ q=c$	$x(i)=q\oplus x(j)=p$	0
$d_{b,a}h_{d,c}\frac{\phi(x)^1_{(b,a)(d,c)}}{2n}$	$i=b,\ j=a,\ p=d,\ q=c$	$x(i)=p\oplus x(j)=q$	$d_{b,a}h_{d,c}\frac{-2}{2n}$

donde la suma de los cuatro términos es:

$$g(x)_{(a,b),(c,d)} = d_{a,b}h_{c,d}\frac{-2}{2n} + d_{b,a}h_{d,c}\frac{-2}{2n} \tag{A.6}$$

4. $\underline{\boldsymbol{x(a)=d\oplus x(b)=c}}$**:** Si analizamos cada uno de los cuatro términos de $g(x)_{(a,b),(c,d)}$ por separado obtenemos que:

Término	Configuración	Caso	Valor
$d_{a,b}h_{c,d}\frac{\phi(x)^1_{(a,b)(c,d)}}{2n}$	$i=a,\ j=b,\ p=c,\ q=d$	$x(i)=q\oplus x(j)=p$	0
$d_{b,a}h_{c,d}\frac{\phi(x)^1_{(b,a)(c,d)}}{2n}$	$i=b,\ j=a,\ p=c,\ q=d$	$x(i)=p\oplus x(j)=q$	$d_{b,a}h_{c,d}\frac{-2}{2n}$
$d_{a,b}h_{d,c}\frac{\phi(x)^1_{(a,b)(d,c)}}{2n}$	$i=a,\ j=b,\ p=d,\ q=c$	$x(i)=p\oplus x(j)=q$	$d_{a,b}h_{d,c}\frac{-2}{2n}$
$d_{b,a}h_{d,c}\frac{\phi(x)^1_{(b,a)(d,c)}}{2n}$	$i=b,\ j=a,\ p=d,\ q=c$	$x(i)=q\oplus x(j)=p$	0

donde la suma de los cuatro términos es:

$$g(x)_{(a,b),(c,d)} = d_{b,a}h_{c,d}\frac{-2}{2n} + d_{a,b}h_{d,c}\frac{-2}{2n} \tag{A.7}$$

5. $\underline{\boldsymbol{x(a)\neq c,d\wedge x(b)\neq c,d}}$**:** Si analizamos cada uno de los cuatro términos de $g(x)_{(a,b),(c,d)}$ por separado obtenemos que:

Término	Configuración	Caso	Valor
$d_{a,b}h_{c,d}\frac{\phi(x)^1_{(a,b)(c,d)}}{2n}$	$i=a,\ j=b,\ p=c,\ q=d$	$x(i)\neq p,q\wedge x(j)\neq p,q$	$d_{a,b}h_{c,d}\frac{-1}{2n}$
$d_{b,a}h_{c,d}\frac{\phi(x)^1_{(b,a)(c,d)}}{2n}$	$i=b,\ j=a,\ p=c,\ q=d$	$x(i)\neq p,q\wedge x(j)\neq p,q$	$d_{b,a}h_{c,d}\frac{-1}{2n}$
$d_{a,b}h_{d,c}\frac{\phi(x)^1_{(a,b)(d,c)}}{2n}$	$i=a,\ j=b,\ p=d,\ q=c$	$x(i)\neq p,q\wedge x(j)\neq p,q$	$d_{a,b}h_{d,c}\frac{-1}{2n}$
$d_{b,a}h_{d,c}\frac{\phi(x)^1_{(b,a)(d,c)}}{2n}$	$i=b,\ j=a,\ p=d,\ q=c$	$x(i)\neq p,q\wedge x(j)\neq p,q$	$d_{b,a}h_{d,c}\frac{-1}{2n}$

donde la suma de los cuatro términos es:

$$g(x)_{(a,b),(c,d)} = d_{a,b}h_{c,d}\frac{-1}{2n} + d_{b,a}h_{c,d}\frac{-1}{2n} + d_{a,b}h_{d,c}\frac{-1}{2n} + d_{b,a}h_{d,c}\frac{-1}{2n} \tag{A.8}$$

En base a esto, veamos lo que ocurre si la matriz de distancia D es simétrica (es decir, $d_{a,b} = d_{b,a}$). En este caso, los cinco posibles valores de $g(x)_{(a,b),(c,d)}$ son los siguientes:

- $\boldsymbol{x(a) = c \wedge x(b) = d}$:

$$\begin{aligned} g(x)_{(a,b),(c,d)} &= d_{a,b}h_{c,d}\frac{1-n}{2n} + d_{b,a}h_{c,d}\frac{n-3}{2n} + d_{a,b}h_{d,c}\frac{n-3}{2n} + d_{b,a}h_{d,c}\frac{1-n}{2n} \\ &= d_{a,b}\Big(h_{c,d}\big(\frac{1-n}{2n} + \frac{n-3}{2n}\big) + h_{d,c}\big(\frac{1-n}{2n} + \frac{n-3}{2n}\big)\Big) = \boldsymbol{d}_{a,b}\Big(\frac{-1}{n}\boldsymbol{h}_{c,d} + \frac{-1}{n}\boldsymbol{h}_{d,c}\Big) \end{aligned} \tag{A.9}$$

- $\boldsymbol{x(a) = d \wedge x(b) = c}$:

$$\begin{aligned} g(x)_{(a,b),(c,d)} &= d_{a,b}h_{c,d}\frac{n-3}{2n} + d_{b,a}h_{c,d}\frac{1-n}{2n} + d_{a,b}h_{d,c}\frac{1-n}{2n} + d_{b,a}h_{d,c}\frac{n-3}{2n} \\ &= d_{a,b}\Big(h_{c,d}\big(\frac{n-3}{2n} + \frac{1-n}{2n}\big) + h_{d,c}\big(\frac{n-3}{2n} + \frac{1-n}{2n}\big)\Big) = \boldsymbol{d}_{a,b}\Big(\frac{-1}{n}\boldsymbol{h}_{c,d} + \frac{-1}{n}\boldsymbol{h}_{d,c}\Big) \end{aligned} \tag{A.10}$$

- $\boldsymbol{x(a) = c \oplus x(b) = d}$:

$$\begin{aligned} g(x)_{(a,b),(c,d)} &= d_{a,b}h_{c,d}\frac{-2}{2n} + d_{b,a}h_{d,c}\frac{-2}{2n} \\ &= \boldsymbol{d}_{a,b}\Big(\frac{-1}{n}\boldsymbol{h}_{c,d} + \frac{-1}{n}\boldsymbol{h}_{d,c}\Big) \end{aligned} \tag{A.11}$$

- $\boldsymbol{x(a) = d \oplus x(b) = c}$:

$$\begin{aligned} g(x)_{(a,b),(c,d)} &= d_{b,a}h_{c,d}\frac{-2}{2n} + d_{a,b}h_{d,c}\frac{-2}{2n} \\ &= \boldsymbol{d}_{a,b}\Big(\frac{-1}{n}\boldsymbol{h}_{c,d} + \frac{-1}{n}\boldsymbol{h}_{d,c}\Big) \end{aligned} \tag{A.12}$$

- $\boldsymbol{x(a) \neq c,d \wedge x(b) \neq c,d}$:

$$\begin{aligned} g(x)_{(a,b),(c,d)} &= d_{a,b}h_{c,d}\frac{-1}{2n} + d_{b,a}h_{c,d}\frac{-1}{2n} + d_{a,b}h_{d,c}\frac{-1}{2n} + d_{b,a}h_{d,c}\frac{-1}{2n} \\ &= d_{a,b}\Big(h_{c,d}\big(\frac{-1}{2n} + \frac{-1}{2n}\big) + h_{d,c}\big(\frac{-1}{2n} + \frac{-1}{2n}\big)\Big) = \boldsymbol{d}_{a,b}\Big(\frac{-1}{n}\boldsymbol{h}_{c,d} + \frac{-1}{n}\boldsymbol{h}_{d,c}\Big) \end{aligned} \tag{A.13}$$

Por lo tanto, cuando la matriz de distancia D es simétrica, $g(x)_{(a,b),(c,d)} = d_{a,b}\Big(\frac{-1}{n}h_{c,d} + \frac{-1}{n}h_{d,c}\Big)$ para todo $1 \leq a,b,c,d \leq n$ con $a \neq b$ y $c \neq d$. Es decir, el valor de $g(x)_{(a,b),(c,d)}$ es independiente de x. En consecuencia, la función f_1 es constante en todo el espacio de búsqueda.

De forma similar, estudiemos ahora lo que ocurre cuando la matriz de flujo H es simétrica (es decir, $h_{c,d} = h_{d,c}$). En este caso, los cinco posibles valores de $g(x)_{(a,b),(c,d)}$ son los siguientes:

- $\boldsymbol{x(a) = c \wedge x(b) = d}$:

$$\begin{aligned} g(x)_{(a,b),(c,d)} &= d_{a,b}h_{c,d}\frac{1-n}{2n} + d_{b,a}h_{c,d}\frac{n-3}{2n} + d_{a,b}h_{d,c}\frac{n-3}{2n} + d_{b,a}h_{d,c}\frac{1-n}{2n} \\ &= h_{c,d}\Big(d_{a,b}\big(\frac{1-n}{2n} + \frac{n-3}{2n}\big) + d_{b,a}\big(\frac{1-n}{2n} + \frac{n-3}{2n}\big)\Big) = \boldsymbol{h}_{c,d}\Big(\frac{-1}{n}\boldsymbol{d}_{a,b} + \frac{-1}{n}\boldsymbol{d}_{b,a}\Big) \end{aligned} \tag{A.14}$$

- $\boldsymbol{x(a) = d \wedge x(b) = c}$:

$$\begin{aligned} g(x)_{(a,b),(c,d)} &= d_{a,b}h_{c,d}\frac{n-3}{2n} + d_{b,a}h_{c,d}\frac{1-n}{2n} + d_{a,b}h_{d,c}\frac{1-n}{2n} + d_{b,a}h_{d,c}\frac{n-3}{2n} \\ &= h_{c,d}\Big(d_{a,b}\big(\frac{n-3}{2n} + \frac{1-n}{2n}\big) + d_{b,a}\big(\frac{n-3}{2n} + \frac{1-n}{2n}\big)\Big) = \boldsymbol{h}_{c,d}\Big(\frac{-1}{n}\boldsymbol{d}_{a,b} + \frac{-1}{n}\boldsymbol{d}_{b,a}\Big) \end{aligned} \tag{A.15}$$

- $\boldsymbol{x(a) = c \oplus x(b) = d}$:

$$\begin{aligned} g(x)_{(a,b),(c,d)} &= d_{a,b}h_{c,d}\frac{-2}{2n} + d_{b,a}h_{d,c}\frac{-2}{2n} \\ &= \boldsymbol{h}_{c,d}\Big(\frac{-1}{n}\boldsymbol{d}_{a,b} + \frac{-1}{n}\boldsymbol{d}_{b,a}\Big) \end{aligned} \tag{A.16}$$

- $\boldsymbol{x(a) = d \oplus x(b) = c}$:

$$\begin{aligned} g(x)_{(a,b),(c,d)} &= d_{b,a}h_{c,d}\frac{-2}{2n} + d_{a,b}h_{d,c}\frac{-2}{2n} \\ &= \boldsymbol{h}_{c,d}\Big(\frac{-1}{n}\boldsymbol{d}_{a,b} + \frac{-1}{n}\boldsymbol{d}_{b,a}\Big) \end{aligned} \tag{A.17}$$

- $\underline{\boldsymbol{x(a) \neq c,d \wedge x(b) \neq c,d}}$**:**

$$\begin{aligned} g(x)_{(a,b),(c,d)} &= d_{a,b}h_{c,d}\frac{-1}{2n} + d_{b,a}h_{c,d}\frac{-1}{2n} + d_{a,b}h_{d,c}\frac{-1}{2n} + d_{b,a}h_{d,c}\frac{-1}{2n} \\ &= h_{c,d}\Big(d_{a,b}\big(\frac{-1}{2n} + \frac{-1}{2n}\big) + d_{b,a}\big(\frac{-1}{2n} + \frac{-1}{2n}\big)\Big) = \boldsymbol{h}_{c,d}\Big(\frac{-1}{n}\boldsymbol{d}_{a,b} + \frac{-1}{n}\boldsymbol{d}_{b,a}\Big) \end{aligned} \tag{A.18}$$

Por lo tanto, cuando la matriz de flujo H es simétrica, $g(x)_{(a,b),(c,d)} = h_{c,d}\Big(\frac{-1}{n}d_{a,b} + \frac{-1}{n}d_{b,a}\Big)$ para todo $1 \leq a,b,c,d \leq n$ con $a \neq b$ y $c \neq d$. Es decir, el valor de $g(x)_{(a,b),(c,d)}$ es independiente de x. En consecuencia, la función f_1 es constante en todo el espacio de búsqueda.

B. ANEXO

Coeficientes para el cálculo de la auto-correlación

Los coeficientes W_i utilizados para el cálculo de la auto-correlación que se describen en [14] se pueden considerar como una medida de la contribución de los *landscapes* de la descomposición al valor de la función objetivo de L_0, siendo W_1 la contribución de L_1, W_2 la contribución de L_2 y W_3 la contribución de L_3. Cuanto más alto sea el valor de W_i, mayor será la contribución del *landscape* L_i, y viceversa. Estos coeficientes se calculan mediante la siguiente expresión:

$$W_i = \frac{\overline{f_i^2} - \overline{f_i}^2}{\overline{f_0^2} - \overline{f_0}^2} \tag{B.1}$$

donde f_i se refiere a la función objetivo del *landscape* L_i. Como ejemplo, se han calculado los coeficientes W_1, W_2 y W_3 para las instancias descritas en la Sección 4.1. Dado que $\overline{f_i^2}$ no se puede calcular de forma exacta mediante una ecuación cerrada, los coeficientes han sido aproximados a partir de una muestra aleatoria de soluciones. Teniendo esto en cuenta, los resultados obtenidos se muestran en la Tabla B.1.

Tabla B.1: Coeficientes W_1, W_2 y W_3 utilizados para el cálculo de la auto-correlación que se describen en [14]. El coeficiente de mayor valor en cada caso está marcado en verde.

	Bur26a	Bur26b	Bur26c	Tai15b	Tai20b	Chr15a	Esc16a	Had20	Rou20
W_1	0,030883	0,033055	0,031696	0,00	0,00	0,00	0,00	0,00	0,00
W_2	0,363862	0,405233	0,342857	0,000006	0,426158	0,908042	1,00	0,699599	0,860496
W_3	0,605255	0,561712	0,625447	0,999994	0,573842	0,091958	0,00	0,300401	0,139504

Bibliografía

[1] Tjalling Koopmans and Martin Beckmann. Assignment problems and the location of economic activity. *Econometrica*, 25, 02 1955.

[2] Christos Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*, volume 32. 01 1982.

[3] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35:268–308, 01 2001.

[4] Alwalid Elshafei. Hospital layout as a quadratic assignment problem. *Journal of The Operational Research Society*, 28:167–179, 04 1977.

[5] Rainer Burkard and J. Offermann. Entwurf von schreibmaschinentastaturen mittels quadratischer zuordnungsprobleme. *Mathematical Methods of Operations Research*, 21, 08 1977.

[6] Nathan Brixius and Kurt Anstreicher. The steinberg wiring problem. 01 2002.

[7] Arthur Geoffrion and Glenn Graves. Scheduling parallel production lines with changeover costs: Practical application of a quadratic assignment/ lp approach. *Operations Research*, 24:595–610, 08 1976.

[8] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. 01 2009.

[9] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35:268–308, 01 2001.

[10] Thomas Stützle and Holger Hoos. Stochastic local search-foundations and applications. 12 2005.

[11] Thomas Bartz-Beielstein, Juergen Branke, Jorn Mehnen, and Olaf Mersmann. Evolutionary algorithms. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4, 05 2014.

[12] Joshua Knowles, Richard Watson, and David Corne. Reducing local optima in single-objective problems by multi-objectivization. 01 2001.

[13] Francisco Chicano, Darrell Whitley, and Enrique Alba. A methodology to find the elementary landscape decomposition of combinatorial optimization problems. *Evolutionary computation*, 19:597–637, 04 2011.

[14] Francisco Chicano, Gabriel Luque, and Enrique Alba. Elementary landscape decomposition of the quadratic assignment problem. pages 1425–1432, 01 2010.

[15] Darrell Whitley, Andrew Sutton, and Adele Howe. Understanding elementary landscapes. pages 585–592, 01 2008.

[16] Peter Stadler. Towards a theory of landscapes. *Complex Systems and Binary Networks, Lecture Notes in Physics*, 461:77–163, 01 1995.

[17] Lov Grover. Local search and the local structure of np-complete problems. *Operations Research Letters*, 12:235–243, 10 1992.

[18] Leticia Hernando, Alexander Mendiburu, and Jose Lozano. An evaluation of methods for estimating the number of local optima in combinatorial optimization problems. *Evolutionary computation*, 21, 12 2012.

[19] Josu Ceberio, Alexander Mendiburu, and Jose Lozano. The linear ordering problem revisited. *European Journal of Operational Research*, 241:686–696, 03 2015.

[20] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack Problems*. 01 2004.

[21] Rainer Burkard and Eranda Dragoti-Cela. *Linear Assignment Problems and Extensions*, pages 75–149. 01 1999.

[22] Karla Hoffman, Manfred Padberg, and Giovanni Rinaldi. *Traveling Salesman Problem*, pages 1573–1578. 01 2013.

[23] Sewall Wright. The roles of mutation, inbreeding, crossbreeding and selection in evolution. *Proceedings of the Sixth International Congress of Genetics*, 1:356–366, 01 1932.

[24] Christian Reidys and Peter Stadler. Combinatorial landscapes. *SIAM Review*, 44, 03 2001.

[25] Bruno Codenotti and Luciano Margara. *Local properties of some NP-complete problems*. International Computer Science Institute, 1992.

[26] Anton Eremeev and Colin Reeves. Non-parametric estimation of properties of combinatorial landscapes. volume 2279, pages 31–40, 04 2002.

[27] Rainer Burkard, Stefan Karisch, and Franz Rendl. Qaplib-a quadratic assignment problem library. *European Journal of Operational Research*, 55:115–119, 02 1991.

[28] Éric Taillard. Robust taboo search for the quadratic assignment problem. *Parallel Computing*, 17:443–455, 07 1991.

[29] Éric Taillard. Comparison of iterative searches for the quadratic assignment problem. *Location Science*, 3:87–105, 08 1995.

[30] Nicos Christofides and Enrique Benavent. An exact algorithm for the quadratic assignment problem on a tree. *Operations Research*, 37:760–768, 10 1989.

[31] Nicos Christofides and Michael Gerrard. Special cases of the quadratic assignment problem. Technical report, Carnegie-Mellon University, 1976.

[32] Bernhard Eschermann and Hans-Joachim Wunderlich. Optimized synthesis of self-testable finite state machines. pages 390–397, 07 1990.

[33] Scott Hadley, Franz Rendl, and Henry Wolkowicz. A new lower bound via projection for the quadratic assignment problem. *Mathematics of Operations Research*, 17:727–739, 08 1992.

[34] Catherine Roucairol. *Du sequentiel au parallele: la recherche arborescente et son application à la programmation quadratique en variables 0 et 1*. Institut national de recherche en informatique et en automatique, 1987.

[35] Anne Chao and shen-Ming Lee. Estimating the number of classes via sample coverage. *Journal of the American Statistical Association*, 87:210–217, 03 1992.

[36] Anne Chao and Mark Yang. Stopping rules and estimation for recapture debugging with unequal failure rates. *Biometrika*, 80:193–201, 03 1993.

[37] Leticia Hernando, Alexander Mendiburu, and Jose Lozano. Anatomy of the attraction basins: Breaking with the intuition. *Evolutionary Computation*, 27:1–32, 05 2018.

[38] Josu Ceberio, Borja Calvo, Alexander Mendiburu, and Jose Lozano. Multi-objectivising combinatorial optimisation problems by means of elementary landscape decompositions. *Evolutionary Computation*, 27:1–22, 02 2018.

[39] Philip Sedgwick. Pearson's correlation coefficient. *BMJ*, 345:e4483–e4483, 07 2012.

[40] Eric Angel and Vassilis Zissimopoulos. Autocorrelation coefficient for the graph bipartitioning problem. *Theoretical Computer Science*, 191:229–243, 01 1998.

[41] Nenad Mladenovic and Pierre Hansen. Variable neighborhood search. *Computers and Operations Research*, 24:1097–1100, 11 1997.

[42] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13:533–549, 01 1986.

[43] John Holland. Genetic algorithms and the optimal allocation of trials. *SIAM Journal of Computing*, 2:88–105, 06 1973.

[44] Lawrence Fogel, Alvin Owens, and Michael Walsh. *Artificial intelligence through a simulation of evolution.* 01 1965.

[45] Ingo Rechenberg. *Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution.* PhD thesis, 01 1971.

[46] Hans-Paul Schwefel. *Evolutionsstrategie und numerische Optimierung.* PhD thesis, 01 1975.

[47] Pedro Larranaga and Jose Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation.* 01 2002.

[48] Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *Evolutionary Computation, IEEE Transactions on*, 11:712–731, 01 2008.

[49] Kaisa Miettinen. *Nonlinear Multiobjective Optimization*, volume 38. 01 1998.

[50] Joshua Knowles, Richard Watson, and David Corne. Reducing local optima in single-objective problems by multi-objectivization. 01 2001.

[51] Meng-Hiot Lim, Yu Yuan, and Sigeru Omatu. Efficient genetic algorithms using simple genes exchange local search policy for the quadratic assignment problem. *Computational Optimization and Applications*, 15:249–268, 03 2000.

[52] Samuel Kotz, Narayanaswamy Balakrishnan, and Norman Johnson. *Continuous Multivariate Distributions: Models and Applications*, volume 1, pages 485–527. 01 2000.

[53] Khalid Jebari. Selection methods for genetic algorithms. *International Journal of Emerging Sciences*, 3:333–344, 12 2013.

[54] Alessio Benavoli, Giorgio Corani, Janez Demsar, and Marco Zaffalon. Time for a change: A tutorial for comparing multiple classifiers through bayesian analysis. *Journal of Machine Learning Research*, 18, 06 2016.

[55] Borja Calvo, Josu Ceberio, and Jose Lozano. Bayesian inference for algorithm ranking analysis. pages 324–325, 07 2018.

[56] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics*, 1, 11 1944.

[57] Borja Calvo and Guzmán Santafé. scmamp: Statistical comparison of multiple algorithms in multiple problems. *R Journal*, 8:1–8, 01 2016.

[58] Josu Ceberio, Alexander Mendiburu, and Jose Lozano. Are we generating instances uniformly at random? pages 1645–1651, 06 2017.

[59] Mohammad Najaran and Adam Prugel-Bennett. On the landscape of combinatorial optimization problems. *Evolutionary Computation, IEEE Transactions on*, 18:420–434, 06 2014.

[60] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems*, pages 2546–2554, 01 2011.