eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Master's Thesis

Computational Engineering and Intelligent Systems

Understanding the elementary landscape decomposition of the QAP and its effects on local search based algorithms

Xabier Benavides

Advisors
Josu Ceberio
Leticia Hernando

July 12th, 2022

Abstract

Elementary landscapes are a special type of combinatorial landscapes that have some properties that make them particularly suitable for working with meta-heuristic algorithms, specially local search based methods. In most of the cases a landscape is not elementary, however, if the neighborhood function meets some conditions, it can be decomposed into a set of elementary landscapes through a process that is called Elementary Landscape Decomposition (ELD). This is the case of the Quadratic Assignment Problem (QAP) under the swap neighborhood, which can be additively decomposed into three elementary landscapes.

This work has two main goals. First, we use the ELD approach to analyze the behaviour of two local search based meta-heuristics that optimize the QAP. In this sense, we intend to check whether incorporating the ELD during the optimization process improves the performance of meta-heuristic algorithms. Second, we propose an additional decomposition of the elementary landscapes that form the QAP. This decomposition defines a framework that helps us to characterize what is measured by each elementary landscape, hence giving us a deeper insight into the ELD of the problem.

Contents

List of Figures

List of Tables

List of Algorithms

CHAPTER 1

Introduction

Combinatorial Optimization Problems (COPs) [1, 2, 3] are generally complex problems. Their goal is to find the solutions that optimize one or more objective functions in a finite or numerable infinite discrete search space. Many real world situations that involve discrete variables can be modelled as COPs, to name a few, route planning [4, 5], community detection [6] or the analysis of DNA fragments [7, 8] are some illustrative examples. Therefore, it is easy to see that designing algorithms that efficiently solve this type of problems is a key objective in many areas.

In the case of NP-hard COPs [9], there is no algorithm capable of solving all instances of such problems in a polynomial time in the dimension of the instance (unless $P = NP$). Some of the most relevant COPs are classified as NP-Hard, such as the Traveling Salesman Problem (TSP) [10], the Linear Ordering Problem (LOP) [11] or the Graph Matching Problem (GMP) [12]. When the instance is small, this type of problems can be optimally solved by some exact algorithms. In this sense, Branch and Bound [13] and Branch and Cut [14] methods have been extensively used for solving small sized instances. Nevertheless, these strategies are not feasible for larger instances, since their computational cost grows exponentially with the problem size. Thus, in such cases, the meta-heuristic approaches emerge as an interesting alternative.

Meta-heuristic methods [15, 16] are optimization algorithms that do not guarantee to find the global optimum of the given problem. Instead, they provide good solutions in a reasonable amount of time, which is usually enough in many real world situations. Over the last decades, a huge number of different meta-heuristic have been proposed to solve all kind of combinatorial optimization problems, ranging from local search based algorithms that rely on neighborhood structures [17, 18, 19, 20] to population based algorithms inspired by natural processes [21, 22, 23, 24].

In recent years, one of the relevant problems in the combinatorial optimization field is the Quadratic Assignment Problem (QAP) [25, 26]. This problem has its origin in logistic planning, but it has applications in many other fields, such as layout design [27], keyboard configuration [28], backboard wiring [29] or parallel production scheduling [30]. Furthermore, some of the most relevant COPs are just particular cases of the QAP (e.g., the TSP [31]), which makes this problem one of the most studied COPs in the literature, and will be

the problem of interest in this work.

As the QAP is a NP-hard problem [32], a number of works in the field have been devoted to developing efficient meta-heuristic algorithms [33]. In particular, Memetic Algorithms (MA) [34, 35] have arisen as competitive methods for solving the QAP. Memetic algorithms are just population based algorithms that internally use some kind of local search procedure to improve the solutions during the search process. Some examples of state-of-the-art memetic algorithms for the QAP are, for instance, the Breakout Memetic Algorithm (BMA) [36] or the Improved Hybrid Genetic Algorithm (IHGA) [37]. Both of them are based on modified versions of the Tabu Search (TS) [19, 38], which has been proven to be one of the most efficient local search based methods for addressing the QAP [39].

In order to efficiently solve the QAP, a number of papers have shown [40, 41, 42, 43] that having prior knowledge about the problem is beneficial to propose efficient algorithm designs. As its NP-hard nature makes it difficult to directly study the characteristics of the QAP, decomposing the problem into a set of sub-problems that ease the analysis may be an interesting idea. This way, the decomposed version of the problem can be used to design new efficient meta-heuristic algorithms. Among all the decomposition strategies that are available for the QAP, the Elementary Landscape Decomposition (ELD) proposed by Chicano et al. [44, 31] has provided some interesting insights about the problem structure [45]. Briefly, the ELD is an additive decomposition method that divides a COP into a set of *landscapes* with some properties that are particularly interesting for working with local search based methods. This decomposition has been successfully used to create a new multi-objectivization framework for the QAP [46], obtaining a set of solutions that was much more diverse than the ones obtained by other single objective algorithms. Moreover, Rockmore et al. [47] discovered that there is a link between the ELD and the Fourier Decomposition of the problem, which renewed the interest of studying the ELD approach.

One of the drawbacks of the ELD is that it is difficult to understand what each landscape in the decomposition measures. This interpretability issue makes difficult to continue studying the ELD and its applications. In this work, we pretend to address this problem by trying to better understand the ELD approach and its effect on the behaviour of local search algorithms. With this aim, we focus on two main objectives:

- First, we compare the behaviour of two different local search meta-heuristic, a simple Tabu Search and a ELD based method proposed in [45]. Our idea is to analyze which are the advantages and disadvantages of considering the landscapes of the decomposition during the optimization. This way, we pretend to prove that the ELD approach can be useful to guide local search processes.

- Second, we propose a decomposition of the components of the ELD that allows us to analyze what does each landscape exactly measure. This new decomposition is studied from both theoretical and experimental point of views to better characterize the components of the ELD.

This document is organized as follows. Chapter 2 explains the basic terms and concepts related to the QAP and ELD that are necessary to fully understand this work. Chapter 3 shows the comparison between the local search based algorithms, and discusses the effects of the ELD in the optimization process. The proposed decomposition of the ELD

components is described and analyzed in Chapter 4. Finally, the conclusions and future research lines are presented in Chapter 5.

CHAPTER 2

Basic concepts

In this chapter, we present some of the key concepts that will be discussed throughout the rest of the document. Additional references are also included for the interested reader.

2.1 Combinatorial Optimization

Combinatorial Optimization (CO) is a branch of discrete mathematics in which the objective is to find the best possible solution over a set of finite or countably infinite discrete solutions. A Combinatorial Optimization Problem (COP) is defined by two main concepts: the search space (Ω) and the objective function (f).

- **Search space (Ω):** A finite or countably infinite set that contains all the feasible solutions.
- **Objective function (f):** A function $f : \Omega \to \mathbb{R}$ that assigns a fitness value to each solution in Ω. The objective function is used to compare the quality of the solutions. When a COP has more than one objective function, we say that the problem is multi-objective.

Considering a single objective COP, the goal is to find the global optima of the problem, that is, the solutions $x^* \in \Omega$ such that $f(x^*) = \min_{x \in \Omega} f(x)$ (assuming minimization).

2.1.1 Quadratic Assignment Problem

The Quadratic Assignment Problem (QAP) is a combinatorial optimization problem that was presented by Koopmans and Beckmann [25] as a mathematical model for the location of indivisible economic activities. Given a set of n facilities and n possible locations, the goal of the QAP is to find the facility-location assignment that minimizes the costs derived from the communications between facilities. In order to do that, we need the following information:

- The distances between locations, stored in a distance matrix $D_{n \times n} = [d_{ij}]$ where d_{ij} is the distance between the location i and the location j.

- The work flows between facilities, stored in a flow matrix $H_{n\times n} = [h_{pq}]$ where h_{pq} denotes the work flow between the facilities p and q.

Intuitively, any pair of facilities with a high communication work flow should be located near each other in order to reduce costs. By contrast, the facilities that have a low communication work flow should be located far apart from each other. The objective function that measures the quality of a solution for the QAP is formalized as follows:

$$f(\sigma) = \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}h_{\sigma(i)\sigma(j)} \tag{2.1}$$

where σ is a permutation of size n that represents the facility-location assignment and $\sigma(i)$ is the facility assigned to the location i. Thus, the search space of the problem is the set of all the permutations of size n, denoted as S_n. The goal in the QAP is to find the assignment $\sigma^* \in S_n$ that minimizes the objective function f.

2.2 Elementary Landscapes

Given a COP defined by the tuple (Ω, f), a neighborhood operator is defined as a function $N : \Omega \mapsto \mathcal{P}(\Omega)$, where $\mathcal{P}(\Omega)$ is the power set of Ω. In other words, the neighborhood operator is a function that assigns to each solution $x \in \Omega$ a set of solutions $N(x) \subset \Omega$ that is known as the neighborhood of x, creating a neighborhood structure (graph) that interconnects all the solutions in the search space. From now on, we will only consider symmetric ($y \in N(x) \Leftrightarrow x \in N(y)$) and regular ($|N(x)| = d$ for all $x \in \Omega$) neighborhood functions.

Then, a landscape of a combinatorial optimization problem [48, 49] can be represented as a triplet (Ω,f,N), where Ω is the search space of the problem, f is the objective function and N is a neighborhood operator. Many important concepts in combinatorial optimization are defined by the landscape, namely:

- **Local optima:** We define a local minimum as any solution $x_{min} \in \Omega$ such that $f(x_{min}) \leq f(y)$ for every $y \in N(x_{min})$. A local maximum is defined analogously.
- **Plateaus:** We define a plateau as a set of solutions $P \subset \Omega$ such that for all $x, y \in P$ satisfies $f(x) = f(y)$ and there is a path $(x = a_1, a_2, ..., a_k = y)$ such that $a_i \in P$ and $a_{i+1} \in N(a_i)$.

Thus, we can observe that studying the underlying landscape structure is crucial when analyzing COPs. Among all the possible landscapes, it has been shown in the literature that those that satisfy the Grover's wave equation [50], known as elementary landscapes, have some interesting properties that make them promising candidates for being solved using local search based algorithms [51, 49]. The Grover's wave equation is expressed as

$$\underset{y\in N(x)}{avg}\{f(y)\} = f(x) + \frac{k}{|N(x)|}\left(\bar{f} - f(x)\right) \tag{2.2}$$

where k is a characteristic constant and $\bar{f}$ represents the average objective function value of the entire search space of solutions. Equation 2.2 can only be satisfied when the objective function f is an elementary function, that is, when f is an eigenfunction of the Laplacian matrix of the graph structure induced by the neighborhood operator N [50, 49].

One of the advantages of elementary landscapes is that the Grover's wave equation allows computing the average objective function value of the neighborhood of any solution $x \in \Omega$ based on the objective value of that particular solution x. Moreover, the Grover's wave equation can also be used to compute the average objective function value of a partial neighborhood. For example, suppose that we have already explored a subset $M \subset N(x)$ of the neighborhood of a certain solution x. Then, the average objective function value of the remaining neighborhood $N(x) - M$ can be computed as

$$\underset{y \in (N(x)-M)}{avg}\{f(y)\} = \frac{1}{|N(x) - M|}\left(|N(x)|\Big(f(x) + \frac{k}{|N(x)|}\big(\bar{f} - f(x)\big)\Big) - \sum_{z \in M} f(z)\right) \tag{2.3}$$

From a practical point of view, this equation can be really useful when exploring the search space of an elementary landscape using its neighborhood structure. For example, let us consider two different solutions $x, y \in \Omega$. After exploring some of their neighboring solutions $M_x \subset N(x)$ and $M_y \subset N(y)$, we want to decide which of the remaining neighborhoods should be explored next. Considering that z_x and z_y are random solutions that belong to $N(x) - M_x$ and $N(y) - M_y$ respectively, we can use the previous equation to compute the expected values of both random solutions. Thus, if the expected value of z_x is better than the expected value of z_y, then it is preferable to continue exploring the $N(x)$ neighborhood, and vice versa. As the solution evaluation may be pretty costly, this approach may be useful to create efficient local search based algorithms.

In addition to the potential efficiency improvements, elementary landscapes are also interesting because of their common properties. In particular, this type of landscapes always satisfy the following [52]:

1. $f(x) < \bar{f} \Longrightarrow f(x) < \underset{y \in N(x)}{avg}\{f(y)\} < \bar{f}$
2. $f(x) = \bar{f} \Longrightarrow f(x) = \underset{y \in N(x)}{avg}\{f(y)\} = f(y) = \bar{f}$
3. $f(x) > \bar{f} \Longrightarrow f(x) > \underset{y \in N(x)}{avg}\{f(y)\} > \bar{f}$

These properties imply that in an elementary landscape the objective value of any local minimum is always equal to or lower than the average objective function value $\bar{f}$ (first condition), while the opposite happens in the case of the local maxima (third condition). Additionally, these constraints ensure that, if there exists any solution $x \in \Omega$ such that $f(x) = f(y)$ for every $y \in N(x)$, the entire landscape is flat, that is, all the solutions in the search space have the same objective function value (second condition).

Furthermore, the properties above also prove that certain types of plateaus cannot exist in elementary landscapes. Assuming that P_{elem} is a plateau of an elementary landscape,

due to Equation 2.2, $\underset{z \in N(x)}{avg}\{f(z)\} = \underset{w \in N(y)}{avg}\{f(w)\}$ for every pair of solutions $x, y \in P_{elem}$. Therefore, if there is a solution $x \in P_{elem}$ that only has both equal and worse neighbors, there cannot be any solution $y \in P_{elem}$ that only has both equal and better neighbors, and vice versa.

All these limitations on the characteristics of the local optima and plateaus are common to every landscape that follows the Grover's wave equation. Therefore, we can observe that elementary landscapes always have a well-known structure, which is particularly useful for dealing with COPs. However, many of the most relevant COPs cannot be expressed as a single elementary landscape. Nevertheless, given a symmetric neighborhood operator, any landscape that is not elementary can be expressed as a linear combination of a set of elementary landscapes. This decomposition process is known as the Elementary Landscape Decomposition (ELD) [44].

2.2.1 Elementary Landscape Decomposition of the QAP

The ELD for the Quadratic Assignment Problem that was proposed in [31] is based on the *swap* neighborhood operator, which has been widely used in the literature for solving the QAP [53]. Given a permutation $\sigma = \{\sigma(1), ..., \sigma(i), ..., \sigma(j), ..., \sigma(n)\}$, this neighborhood operator consists in exchanging two items $\sigma(i)$ and $\sigma(j)$ in order to obtain a new neighbor solution $\sigma' = \{\sigma(1), ..., \sigma(j), ..., \sigma(i), ..., \sigma(n)\}$. Hence, the original landscape used in the decomposition is (S_n, f, N), where S_n is the set of all permutations of size n, f is the objective function of the QAP, and N is the *swap* neighborhood operator. In what follows, we denote this landscape as L.

The ELD of L consists of finding a set of m elementary functions $\{f^1, f^2, ..., f^m\}$ that form m elementary landscapes along with the original search space and neighborhood operator. These elementary functions must satisfy that $f(\sigma) = f^1(\sigma) + f^2(\sigma) + ... + f^m(\sigma)$ for every $\sigma \in S_n$, so it is easy to see that the goal of the ELD is to decompose the original objective function f (Equation 2.1) into a set of sub-functions. In order to do that, we first rewrite f as follows:

$$f(\sigma) = \sum_{i,j=1}^{n} \sum_{p,q=1}^{n} d_{i,j} h_{p,q} \delta^p_{\sigma(i)} \delta^q_{\sigma(j)} \tag{2.4}$$

where δ^b_a represents the Kronocker delta function that returns 1 if $a = b$, and 0 otherwise. The function in Equation 2.4 can be easily separated into two different parts: the instance related part that depends on the distance and flow matrices ($\psi_{i,j,p,q} = d_{i,j} h_{p,q}$) and the problem related part that depends on σ ($\varphi_{(i,j)(p,q)}(\sigma) = \delta^p_{\sigma(i)} \delta^q_{\sigma(j)}$). Hence, we have that:

$$f(\sigma) = \sum_{i,j=1}^{n} \sum_{p,q=1}^{n} \psi_{i,j,p,q} \varphi_{(i,j)(p,q)}(\sigma) \tag{2.5}$$

It is important to remark that the value of $\psi_{i,j,p,q}$ does not vary depending on the input solution, so it is easy to see that f is just a linear combination of $\varphi_{(i,j)(p,q)}(\sigma)$. Thus, as any linear combination of elementary functions (with the same characteristic constant) is also an elementary function, we can focus on decomposing $\varphi_{(i,j)(p,q)}(\sigma)$, that is, the

problem related part. Taking this into account, f can be decomposed into three independent elementary functions [31]:

$$f^1(\sigma) = \sum_{\substack{i,j,p,q=1 \\ i\neq j \\ p\neq q}}^{n} \psi_{i,j,p,q} \frac{\phi^1_{(i,j)(p,q)}(\sigma)}{2n} \tag{2.6}$$

$$f^2(\sigma) = \sum_{\substack{i,j,p,q=1 \\ i\neq j \\ p\neq q}}^{n} \psi_{i,j,p,q} \frac{\phi^2_{(i,j)(p,q)}(\sigma)}{2(n-2)} \tag{2.7}$$

$$f^3(\sigma) = \sum_{i,p=1}^{n} \psi_{i,i,p,p}\varphi_{(i,i)(p,p)}(\sigma) + \sum_{\substack{i,j,p,q=1 \\ i\neq j \\ p\neq q}}^{n} \psi_{i,j,p,q} \frac{\phi^3_{(i,j)(p,q)}(\sigma)}{n(n-2)} \tag{2.8}$$

where $\phi^1_{(i,j)(p,q)}$, $\phi^2_{(i,j)(p,q)}$ and $\phi^3_{(i,j)(p,q)}$ are defined as:

$$\phi^m_{(i,j)(p,q)}(\sigma) = \begin{cases} \alpha & \text{if } \sigma(i) = p \wedge \sigma(j) = q \\ \beta & \text{if } \sigma(i) = q \wedge \sigma(j) = p \\ \gamma & \text{if } \sigma(i) = p \oplus \sigma(j) = q \\ \epsilon & \text{if } \sigma(i) = q \oplus \sigma(j) = p \\ \zeta & \text{if } \sigma(i) \neq p, q \wedge \sigma(j) \neq p, q \end{cases} \tag{2.9}$$

where $1 \leq i, j, p, q \leq n$ and $\alpha, \beta, \gamma, \epsilon, \zeta \in \mathbb{R}$. The operator $\oplus$ stands for the exclusive OR operator. The parameter values for each of the functions $m = 1, 2, 3$ are shown in Table 2.1.

Table 2.1: Specific values of the $\alpha, \beta, \gamma, \epsilon, \zeta$ parameters for each of the elementary landscapes.

	α	β	γ	ϵ	ζ
ϕ^1	n-3	1-n	-2	0	-1
ϕ^2	n-3	n-3	0	0	1
ϕ^3	2n-3	1	n-2	0	-1

As stated before, the ELD is an additive decomposition, so $f(\sigma) = f^1(\sigma)+f^2(\sigma)+f^3(\sigma)$ for every $\sigma \in S_n$. Each of the elementary functions satisfies the Grover's wave equation (Equation 2.2) for the search space and the neighborhood operator of L. Thus, they form three independent elementary landscapes: $L^1 = (S_n, f^1, N)$, $L^2 = (S_n, f^2, N)$ and $L^3 = (S_n, f^3, N)$. These elementary landscapes are, precisely, the components of the decomposition of the QAP.

Although the ELD of the problem consists of three components, this does not mean that all the instances of the QAP are composed of three non-constant elementary landscapes. For example, [45] proved that the objective function of L_1 is constant when at least one of the matrices that form the QAP is symmetric with respect to the main diagonal. Something similar happens in the case of the TSP, which, as we have mentioned before, is a special

case of the QAP. In TSP-like instances, the objective function of L_3 becomes constant for all the solutions in the search space [31]. Therefore, for practical purposes, the ELD may be simplified into less than three elementary landscapes depending on the characteristics of the given instance.

2.3 Variable Function Search: A modified Tabu Search

Regarding the meta-heuristics that solve the QAP, one of the most relevant local search based algorithms is the Tabu Search (TS) [54, 39]. Similar to a classical hill climbing algorithm, the TS is a meta-heuristic method that works by moving between solutions in the search space using neighborhood operators. That is, starting from an initial solution $x_0 \in \Omega$, the TS algorithm explores the search space of a problem by visiting a sequence of solutions $(x_0, x_1, x_2, ..., x_k)$ such that $x_{i+1} \in N(x_i)$. During each step of the search process, the algorithm always moves to the best neighboring solution according to the objective function f, regardless of whether the new solution x_{i+1} is better than the current solution x_i or not. This means that the TS algorithm allows moving to non-improving solutions, which is useful to avoid getting stuck in low quality local optima.

One of the disadvantages of this approach is that allowing non-improving movements could lead to cycles in the search process. In order to avoid this problem, the TS uses a special data structure to avoid returning to already visited solutions: the tabu list. In its simplest version, the tabu list is just a short term memory that contains the last T neighborhood movements. Those movements are considered to be tabu, that is, they cannot be performed again until they leave the tabu list. Therefore, when the algorithm explores the neighborhood of a solution, it excludes the tabu movements from the neighbor selection process. Then, once the new solution is selected, the algorithm moves to that solution and the tabu list is updated.

Forbidding neighborhood movements during the search process can make us miss potentially good solutions. For this reason, TS algorithms usually include an aspiration criterion [55] that allows them to ignore the tabu status of a solution under special circumstances. For example, if a tabu neighborhood movement would lead to a solution that is better than the best solution found until the moment, the tabu status could be ignored to encourage the exploration of more promising regions of the search space. The pseudocode of the basic TS algorithm is shown in Algorithm 1.

In addition to the basic Tabu Search, in this work we also analyze the behaviour of a modified version of the algorithm that uses the ELD of the problem to guide the search process: the Variable Function Search (VFS) [45]. The main difference between the TS and the VFS is that the VFS only moves from a solution $x \in \Omega$ to a neighboring solution $y \in N(x)$ if $\exists\, f^i \in F$ such that $f^i(y) < f^i(x)$ (assuming minimization), where $F = \{f^1, f^2, ..., f^m\}$ is the set of all the elementary functions of the ELD. Apart from this additional constraint, the VFS is exactly the same as a classical TS, so it is not difficult to see that the VFS can be potentially incorporated in any tabu search based algorithm.

Considering a problem with a known ELD, the value of the original objective function f for a certain solution $x \in \Omega$ can be computed as the sum of the values of the elementary functions of the ELD for that same solution x, that is, $f(x) = f^1(x) + f^2(x) + ... + f^m(x)$ for every $x \in \Omega$. Therefore, the ELD can be viewed as a multi-objectivization procedure that

Algorithm 1 *Tabu Search.*

Input:

x - Initial solution ($x \in \Omega$)
f - General objective function
N - Neighborhood operator
Tabu - Tabu criterion
Stop - Stopping criterion

Output:

$\mathbf{x}^*$ - Final solution ($x^* \in \Omega$)

```
1:  procedure TS (x, f, N, Tabu, Stop)                ▷ Assuming minimization.
2:      x* ← x
3:      while ¬Stop do
4:          x_aux ← NULL
5:          for y ∈ N(x) do                           ▷ Neighborhood exploration.
6:              if ¬Tabu(y) ∨ f(y) < f(x*) then
7:                  if x_aux = NULL ∨ f(y) < f(x_aux) then
8:                      x_aux ← y
9:                  end if
10:             end if
11:         end for
12:         if x_aux ≠ NULL then                      ▷ Update the current solution x.
13:             x ← x_aux
14:             if f(x) < f(x*) then                  ▷ Update the best solution x*.
15:                 x* ← x
16:             end if
17:         end if
18:     end while
19:     return x*                                     ▷ Return the best solution found.
20: end procedure
```

transforms a mono-objective problem into a multi-objective problem with m sub-objectives $f^1, f^2, ..., f^m$ [46]. The VFS exploits this idea by ensuring that at each step of the search the algorithm always moves to a solution that improves at least one of the sub-objectives of the problem. This constraint is particularly relevant when the algorithm has to escape from local optima, that is, when all the solutions in the neighborhood are equal or worse than the current solution. In such cases, the classical tabu search just moves to the best neighboring solution according to f, which could worsen the value of all the elementary functions of the ELD. This greedy strategy is not always optimal [45], since moving to solutions that are worse than the current local optimum for all the sub-objectives of the problem (also called dominated solutions [56]) may hinder the optimization process. Thus, the VFS approach arises as a promising alternative that guarantees that at least one of the elementary functions is improved even when escaping a local optimum. The pseudocode of the VFS is shown in Algorithm 2.

Algorithm 2 *Variable Function Search.*

Input:

x - Initial solution ($x \in \Omega$)

f - General objective function

$\mathbf{F_{ELD}}$ - Elementary functions of the ELD $\{f^1, ..., f^m\}$

N - Neighborhood operator

Tabu - Tabu criterion

Stop - Stopping criterion

Output:

$\mathbf{x}^*$ - Final solution ($x^* \in \Omega$)

procedure VFS ($x, f, F_{ELD}, N, Tabu, Stop$) ▷ Assuming minimization.
  $x^* \leftarrow x$
  while $\neg Stop$ **do**
    $x_{aux} \leftarrow NULL$
    for $y \in N(x)$ **do** ▷ Neighborhood exploration.
      if $\underset{f^i \in F_{ELD}}{\exists} f^i(y) < f^i(x) \wedge \Big(\neg Tabu(y) \vee f(y) < f(x^*)\Big)$ **then**
        if $x_{aux} = NULL \vee f(y) < f(x_{aux})$ **then**
          $x_{aux} \leftarrow y$
        end if
      end if
    end for
    if $x_{aux} \neq NULL$ **then** ▷ Update the current solution x.
      $x \leftarrow x_{aux}$
      if $f(x) < f(x^*)$ **then** ▷ Update the best solution x^*.
        $x^* \leftarrow x$
      end if
    end if
  end while
  return x^* ▷ Return the best solution found.
end procedure

CHAPTER 3

Algorithm comparison

As stated before, one of the main goals of this work is to study if the elementary landscape decomposition of the QAP can be used as a tool for guiding a local search process to efficiently solve the problem. For this purpose, we first carry out a comparative analysis of the behaviour of two different local search based algorithms[1]: the Tabu Search and the Variable Function Search.

3.1 Benchmark of instances

The experimental framework used in this chapter consists of a set of instances extracted from two different sources:

- 177 instances from the QAPLIB library [57], ranging from size 5 to 256.
- 112 instances from the *Dre* and *Tai-e* benchmarks proposed in [58], ranging from size 15 to 175.

In total, 289 instances that have been extensively analyzed in previous works [45, 46, 36, 37], which makes it easier to compare our results which those of the state-of-art methods. According to the literature [59], these instances can be divided into 4 different groups:

1. **Unstructured:** Randomly generated unstructured instances. The distance and flow matrices of these problems are generated based on a uniform distribution. It includes the *Lipa*, *Rou* and *Tai-a* benchmarks from the QAPLIB library.
2. **Grid:** Randomly generated instances whose distance matrix is based on the Manhattan distance between cells in a grid. It includes the *Nug*, *Scr*, *Sko*, *Tho* and *Wil* benchmarks from the QAPLIB library. We have also considered that the *Dre* benchmark is part of this group.

[1]The code of the implemented algorithms is available in the following repository: https://github.com/XB-Repositories/TFM_Algorithms.

Table 3.1: Statistics of the TS and VFS algorithms. The table shows the median rank in each benchmark and the number of executions that obtained the first rank in each case (the total amount of executions is also shown in brackets). The different cell colors represent the four different types of benchmarks: *Unstructured* (Blue), *Grid* (Green), *Real-life* (Red) and *Real-life like* (Yellow).

		lipa	rou	tai-a	dre	nug	scr	sko	tho	wil	bur	chr	els	esc	had	kra	ste	tai-c	tai-b	tai-e
TS	Median	1.0	1.0	5.0	7.0	1.0	1.0	10.0	10.5	11.0	6.0	4.0	6.5	1.0	1.5	4.0	7.5	5.0	6.0	11.0
	Nº best	103 (180)	30 (40)	127 (400)	20 (120)	136 (190)	35 (40)	5 (130)	3 (30)	1 (20)	10 (80)	42 (140)	1 (10)	220 (250)	25 (50)	5 (30)	2 (30)	2 (20)	25 (130)	29 (1000)
VFS	Median	1.0	1.0	4.5	9.0	1.0	1.0	10.0	9.0	9.0	5.0	3.0	5.5	1.0	1.0	3.0	8.0	5.0	5.0	9.0
	Nº best	106 (180)	28 (40)	128 (400)	16 (120)	128 (190)	36 (40)	8 (130)	1 (30)	1 (20)	11 (80)	46 (140)	2 (10)	148 (250)	26 (50)	7 (30)	4 (30)	2 (20)	28 (130)	82 (1000)

3. **Real-life:** Real-life instances obtained from real-world applications of the QAP. It includes the *Bur*, *Chr*, *Els*, *Esc*, *Had*, *Kra*, *Ste* and *Tai-c* benchmarks from the QAPLIB library.

4. **Real-life like:** Randomly generated instances that are created in such a way that their structure is similar to that of real-life problems. It includes the *Tai-b* benchmark from the QAPLIB library. We have also considered that the *Tai-e* benchmark is part of this group.

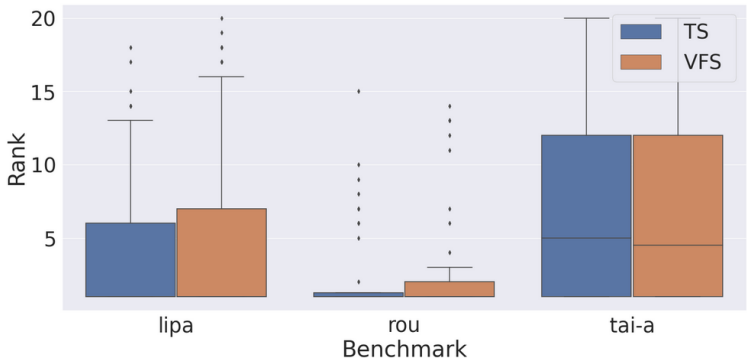
3.2 Experiments and results

In order to measure the performance of the VFS when compared to the basic TS strategy, we run both algorithms 10 times for each of the QAPLIB, *Dre* and *Tai-e* instances. The size of the tabu list in all cases is equal to the corresponding instance size (n). Regarding the stopping criterion, a maximum number of solution evaluations has been set: $1{,}000n^2$.

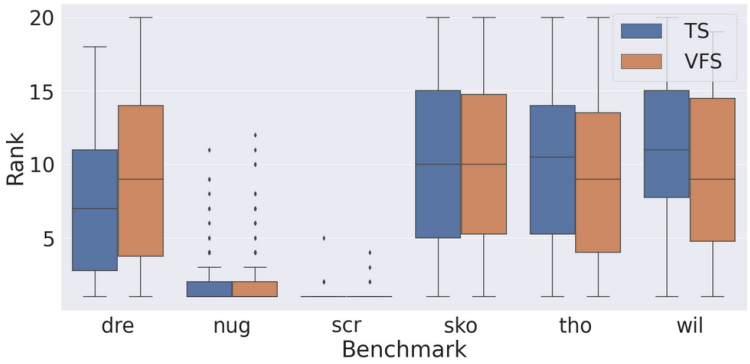
For each of the instances, the executions of both algorithms are ranked according to the resulting objective values, where rank 1 represents the best performing trial and rank 20 the worst. Our goal is to transform the obtained results to a common scale in order to better visualize the differences between the algorithms in different types of instances. The rankings obtained for each of the benchmarks are aggregated and summarized as box plots in Figure 3.1. The general statistics of the obtained results are also shown in Table 3.1.

As can be observed in both the box plots and the table, the performance of the TS and the VFS varies greatly depending on the benchmark of instances. Although both algorithms seem to have a similar performance in most cases, the VFS obtains slightly better median ranks on the *Bur*, *Chr*, *Els*, *Had*, *Kra*, *Tai-a*, *Tai-b*, *Tai-e*, *Tho* and *Wil* benchmarks. These differences are particularly remarkable in the case of the *Tai-e* benchmark, in which the number of first ranks obtained by the VFS is more than twice the number obtained by the TS.

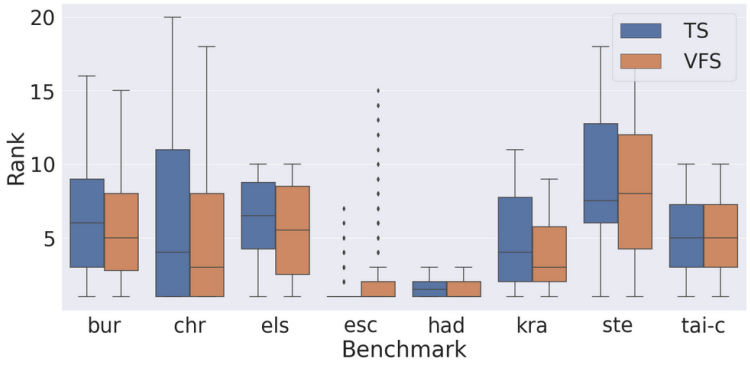
On the other hand, the TS obtains better median ranks on just the *Dre* and *Ste* benchmarks. If we look at the box plots, however, we can see that the TS appears to perform generally better on the *Esc*, *Lipa* and *Rou* benchmarks too. Thus, this suggests that, even though the TS is more promising in some particular cases, the VFS approach may be useful for solving many types of instances. This seems to be the case of *Real-life* and *Real-life like* types of instances, since the VFS achieves equal or better results than the TS on almost all the considered benchmarks.



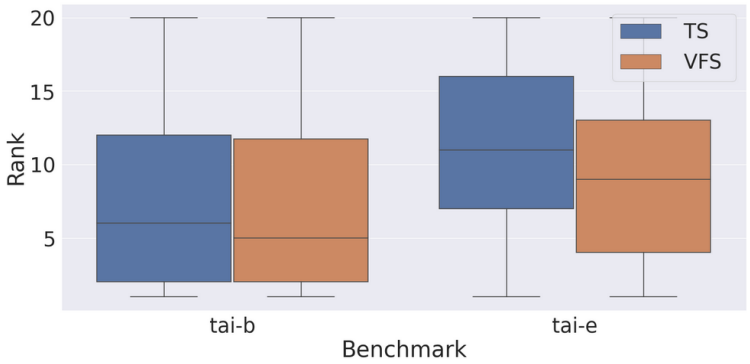
(a) *Unstructured* instances.



(b) *Grid* instances.



(c) *Real-life* instances.



(d) *Real-life like* instances.

Figure 3.1: Box plots of the execution rankings for each benchmark of instances.

Table 3.2: Expected probabilities for each of the possibilities of the Bayesian signed-rank tests. The option with the highest probability in each test is highlighted in bold. The different cell colors represent the four different types of benchmarks: *Unstructured* (Blue), *Grid* (Green), *Real-life* (Red) and *Real-life like* (Yellow).

	lipa	rou	tai-a	dre	nug	scr	sko	tho	wil	bur	chr	els	esc	had	kra	ste	tai-c	tai-b	tai-e
TS	0.24	**0.39**	**0.48**	**0.55**	0.34	0.23	**0.51**	**0.55**	0.34	0.07	0.32	0.00	**0.72**	0.00	0.33	0.44	0.00	0.17	0.06
Rope	**0.45**	0.33	0.05	0.06	**0.35**	**0.65**	0.03	0.03	0.02	0.44	0.10	**0.66**	0.28	**0.96**	0.02	0.03	**1.00**	**0.51**	0.18
VFS	0.32	0.28	0.47	0.39	0.31	0.11	0.46	0.42	**0.64**	**0.49**	**0.57**	0.34	0.00	0.04	**0.64**	**0.53**	0.00	0.32	**0.77**

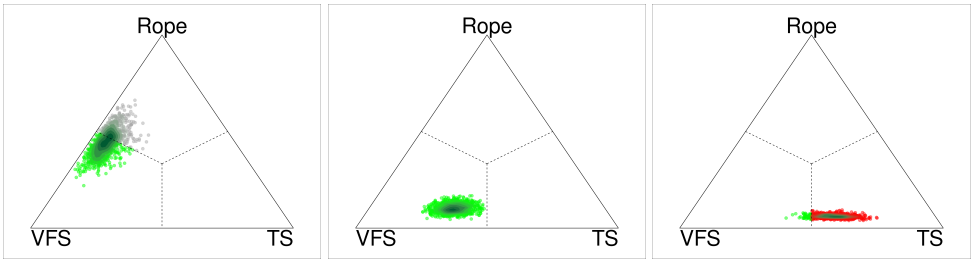
In order to check whether the differences found in the execution rankings are actually significant, we compute a statistical analysis for each benchmark of instances using the Bayesian signed-rank test [60, 61]. Given a set of performance measures obtained by two meta-heuristic algorithms, this method estimates the expected probability of each algorithm being the best for solving the considered test instances. The experimental data used to compute the statistical analyses consists of the relative errors with respect to the best known solutions obtained in the previous experimentation.

The Bayesian signed-rank test requires defining the region of practical equivalence (*rope*), which is the interval of performance difference under which both algorithms are considered to be equivalent (tie). Due to the differences in the scale of the relative errors, in this work the *rope* interval has been set independently for each benchmark of instances. In particular, the limits of the *rope* interval are calculated as $\pm 1\%$ of the average relative error obtained by both the TS and the VFS in the corresponding benchmark. Taking this into account, the results of the statistical analyses are shown as simplex plots in Figure 3.2.

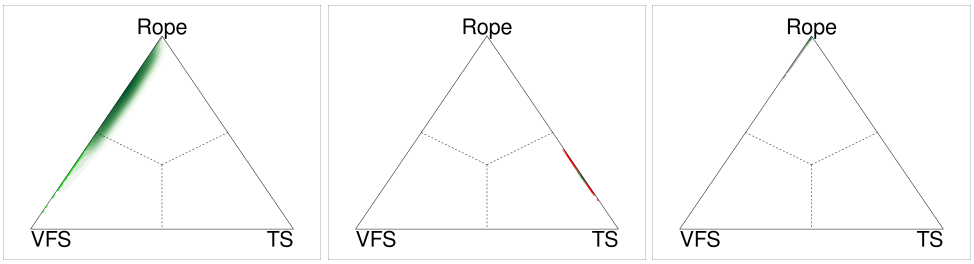
Briefly, each point in the simplex plots represents a sample of the posterior distribution of the probability of win-lose-tie. That is, the closer a point is to a vertex, the higher the probability of the corresponding option. If the points are closer to the TS vertex, for example, it means that the TS algorithm has a higher probability of being the best, and the same happens in the case of the VFS and *rope* vertices. Moreover, the dispersion of the point clouds gives us information about the uncertainty of the statistical analysis. If the points are close together, it means that the results of the analysis have a low uncertainty (e.g., the *lipa* and *nug* cases). In contrast, if the points are far apart, then the uncertainty of the analysis is higher (e.g., the *rou* and *scr* cases). Thus, the Bayesian signed-rank test allows us to differentiate between the uncertainty of the behaviour of the algorithms and the uncertainty of the statistical analysis (which is caused by the lack of data).

In order to summarize the results of the statistical analyses, Table 3.2 shows the expected probabilities of each option (TS-*rope*-VFS) for each benchmark of instances. The results obtained in the statistical analyses suggest that there is a difference between the behaviour of both algorithms. According to the computed expected probabilities, the TS has the highest chance of being the best candidate algorithm in the *dre* (0.55), *esc* (0.72), *rou* (0.39), *sko* (0.51), *tai-a* (0.48) and *tho* (0.55) benchmarks. However, it does not seem to perform much better than the VFS, since the expected probability of the TS option in those cases is always smaller than 0.6. The sole exception is the *esc* benchmark, which is a special case in which only the L^2 elementary landscape is non-constant [45].

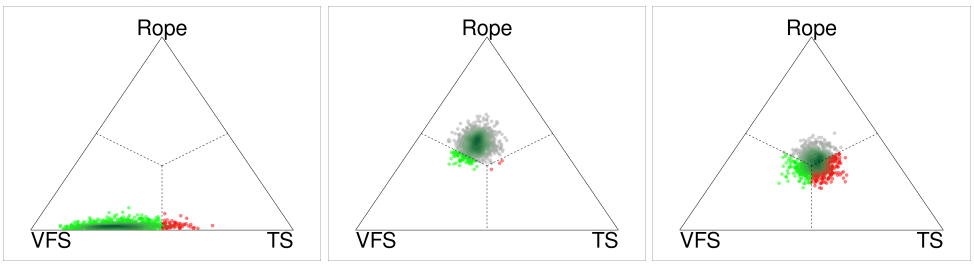
On the other hand, the performed statistical analyses show that the VFS is the most promising algorithm in the *bur* (0.49), *chr* (0.57), *kra* (0.64), *ste* (0.53), *tai-e* (0.77) and *wil* (0.64) benchmarks. In fact, the VFS option has a particularly high probability (> 0.6) in the



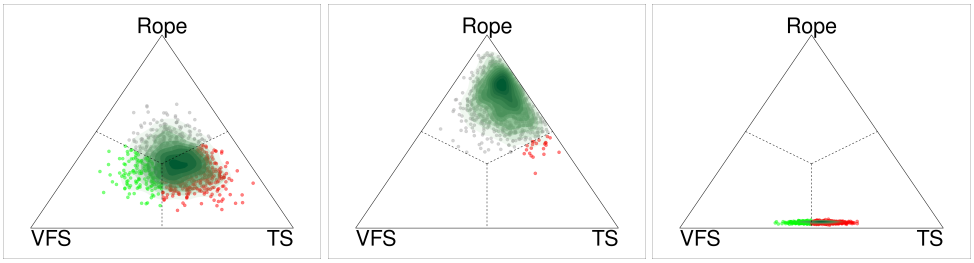
(a) *bur.* **(b)** *chr.* **(c)** *dre.*



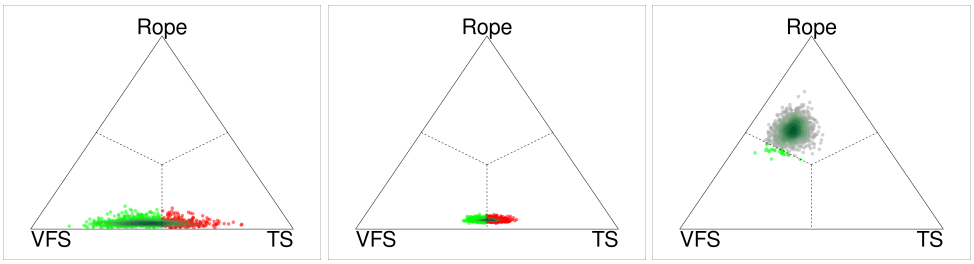
(d) *els.* **(e)** *esc.* **(f)** *had.*



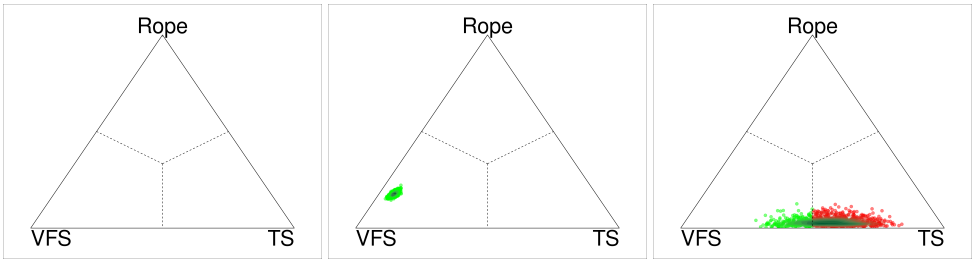
(g) *kra.* **(h)** *lipa.* **(i)** *nug.*



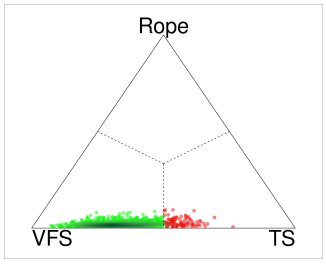
(j) *rou.* **(k)** *scr.* **(l)** *sko.*



(m) *ste.* **(n)** *tai-a.* **(o)** *tai-b.*



(p) *tai-c.* **(q)** *tai-e.* **(r)** *tho.*



(s) *wil.*

Figure 3.2: Results of the statistical analyses shown as simplex plots.

kra, *tai-e* and *wil* instance sets, which suggests that the VFS is much more effective than the TS in those cases.

Finally, there are some benchmarks in which the difference between the performance of the tested algorithms is not significant. Therefore, the *rope* option is the one that has the highest probability in those cases: *els* (0.66), *had* (0.96), *lipa* (0.45), *nug* (0.35), *scr* (0.65), *tai-b* (0.51) and *tai-c* (1.00).

If we analyze the differences between types of instances, the statistical analyses confirm that the VFS seems to be particularly effective for solving *Real-life* and *Real-life like* instances, since in such cases the VFS is equal to or better than the TS in 9 out of 10 benchmarks. The opposite happens in the case of *Unstructured* and *Grid* benchmarks, in which the TS seems to have a higher probability of being the best algorithm. In fact, in those cases the TS is equal to or better than the VFS in 8 out of 9 benchmarks.

3.3 Analysis and discussion

Once we have compared the performance of the TS and the VFS algorithms on a diverse set of instances, the next step is to try to understand the reasons for which the performance of the algorithms varies depending on the instance to be solved. For this purpose, in this section we analyze the characteristics of two benchmarks of instances in which the behaviour of the studied algorithms appears to be very different: the *Dre* and *Tai-e* benchmarks.

Both the *Dre* and the *Tai-e* benchmarks were proposed in [58]. The characteristics of these benchmarks are complementary to the ones from the QAPLIB library, and they are specifically designed to be difficult for local search algorithms, particularly the ones that are based on the swap neighborhood. A summary of the general characteristics of both benchmarks is shown below. For more information about the generation of the instances, we refer the interested reader to the original paper.

- **Dre:** Benchmark composed of symmetric instances[2] that are based on a rectangular grid of size $k \times l$. Each instance is created by generating a random permutation $\sigma*$ of size $n = k \times l$, assigning it to the grid, and then constructing the distance and flow matrices as follows:

  - Non-adjacent facilities are given a 0 work flow, while the adjacent facilities are given a random work flow between 1 and 10.
  - Adjacent locations (cells) are given a 1 distance, while the distance between non-adjacent locations is randomly generated between 2 and 10.

  Thus, it is ensured that $\sigma*$ is the global optimum of the problem. The instances generated using this technique have a high ruggedness, and the objective function increases/decreases pretty steeply when moving from one solution to another using swap neighborhood movements.

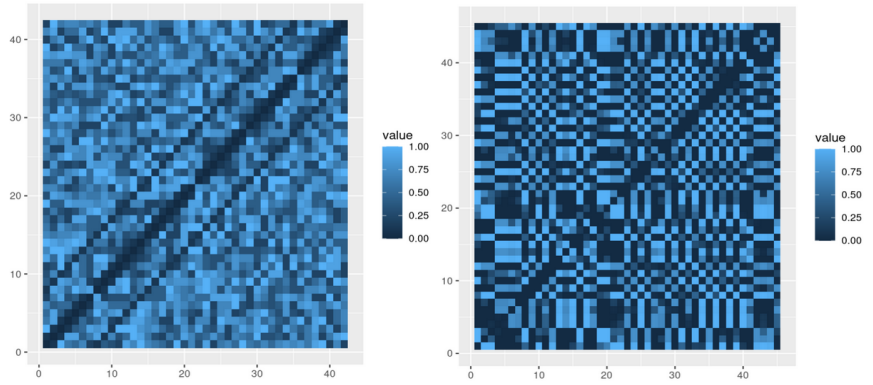
- **Tai-e:** Benchmark composed of symmetric instances in which the distances and flows are not uniformly generated. Instead, the two matrices that form the problem have a well defined block structure. Briefly, the distance and flow matrices of this type of instances are recursively generated as follows:

  1. First, t uniformly generated distance and flow matrices of size $s \times s$ are created.
  2. Then, the generated matrices are inserted into a $st \times st$ block diagonal matrix, where the elements outside the main diagonal are set to small non-negative values in the case of the flow matrix, and to relatively large positive values in the case of the distance matrix.
  3. The previous two steps are repeated u times, creating the final $stu \times stu$ instance.

  The instances generated using this technique can be used to represent the problem of assigning gates to airplanes in an airport [58, 62]. For example, if an airport is composed of u terminals with t different branches and s gates per branch, we know that the distances between gates of the same branch are relatively low, distances between gates located in different branches of the same terminal are more significant, and distances between gates belonging to different terminals are quite large. Therefore, in order to find a good solution for the problem, a meta-heuristic algorithm should match the low distance blocks from the distance matrix with the high work flow blocks from the flow matrix. As exchanging two blocks of size m requires m swap neighborhood movements, many local search iterations are needed for achieving significant improvements in the fitness of the solution [58]. Therefore, local search based algorithms may not be able to escape from low quality local optima, thus being inefficient for solving this type of instances.

In order to better visualize the different structures of the *Dre* and *Tai-e* benchmarks, Figure 3.3 shows the distance matrices of two representative instances (one from each benchmark) as heat maps. Notice the block structure of the *Tai-e* distance matrix in comparison to the grid based *Dre* instance.

[2] An instance is symmetric if both the distance and flow matrices are symmetric with respect to the main diagonal.



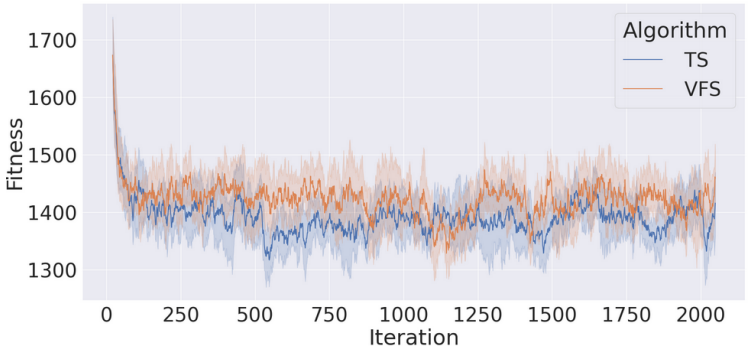
(a) *Dre42* instance. **(b)** *Tai45e01* instance.

Figure 3.3: Distance matrices of two instances from the *Dre* and *Tai-e* benchmarks, represented as normalized heat maps. The color of each cell represents the corresponding value in the distance matrix.

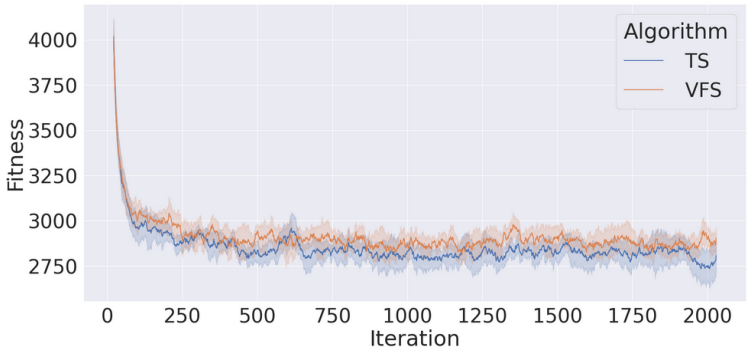
The experimentation carried out in Section 3.2 shows that the TS algorithm outperforms the VFS on the *Dre* benchmark, while the opposite happens in the case of the *Tai-e* instances. In order to analyze the reason, we first plot the objective values of the solutions that are explored during the executions of the previous experimentation. This includes both the general objective function f and the elementary functions of the ELD f^2 and f^3. Our goal is to use the ELD approach to compare the characteristics of the solutions that are visited during the TS and VFS so that we can better study the behaviour of each local search based procedure. As both the *Dre* and *Tai-e* instances are symmetric, the f^1 elementary function is always constant (Section 2.2.1), and thus, we can ignore it during the analysis. For the sake of brevity, we consider the following representative instances: *dre42*, *dre72*, *tai45e01* and *tai75e01* (Figures 3.4 and 3.5).

The results shown in Figure 3.4 confirm the previously extracted conclusions. In the case of the *Dre* instances, the line that represents the mean objective value of the solutions explored by the TS is nearly always below the line that corresponds to the VFS. As the QAP is a minimization problem, this indicates that the TS is exploring generally better solutions than the VFS during the entire execution of the algorithms. Conversely, in the case of the *Tai-e* instances, the mean objective value of the solutions explored by the VFS is slightly better than the mean of those explored by the TS. Moreover, the size of the standard deviation intervals in the *Tai-e* instances suggests that the objective value of the solutions explored by both algorithms varies greatly between runs.

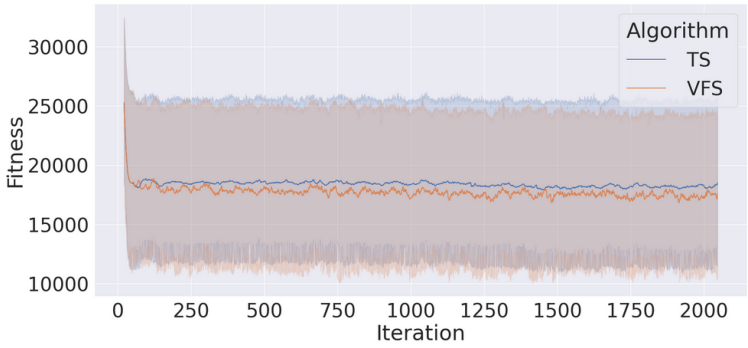
The differences in the behaviour of the algorithms are not limited to the evolution of the general objective function value. Instead, it seems like the optimization of each of the elementary functions also varies depending on the strategy that is used to solve the problem (Figure 3.5). In the case of the TS algorithm, it inherently focuses on optimizing f^2, while the VFS algorithm also tries to improve f^3. As mentioned in [45], the f^2 elementary function is generally the most relevant function in the decomposition, that is, it has the highest relative contribution to the variance of the objective function f. Therefore, given that the TS does not explicitly take into account each of the elementary landscapes, it seems that the algorithm implicitly optimizes the most relevant landscapes of the decomposition, ignoring those that produce small changes in the solution fitness. This is not the case of



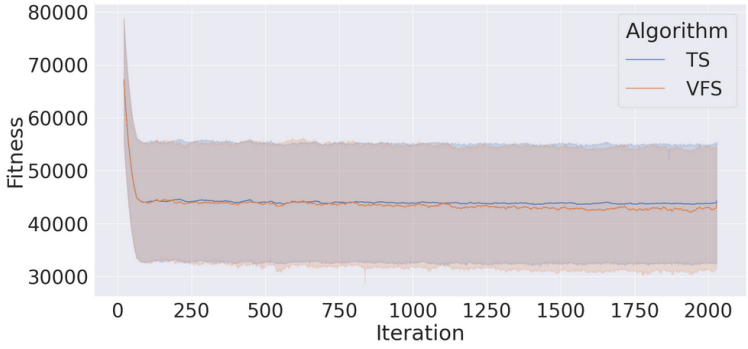
(a) Dre42.



(b) Dre72.

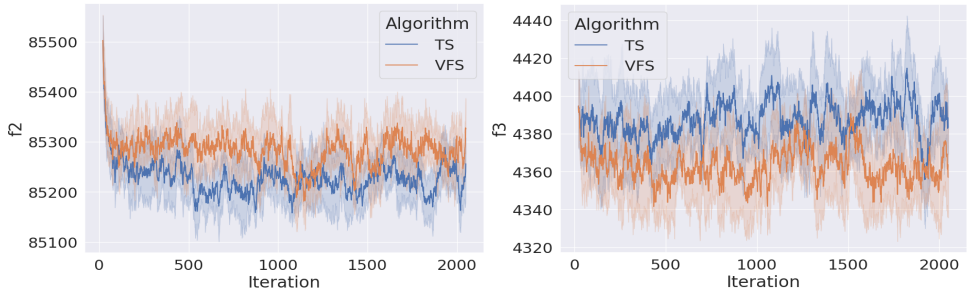


(c) Tai45e01.



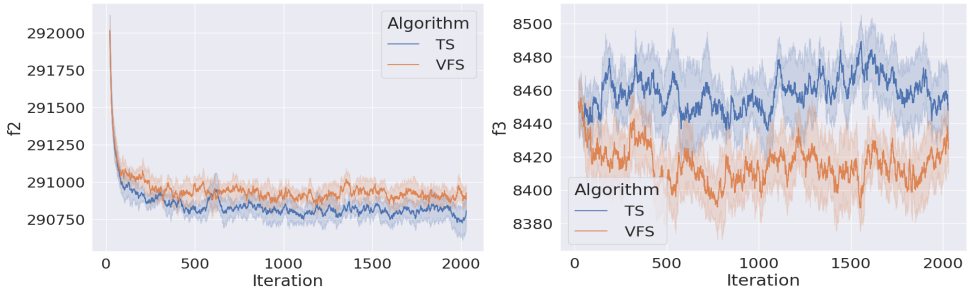
(d) Tai75e01.

Figure 3.4: Evolution of the f objective function during the 10 runs of the TS and VFS algorithms. The solid lines indicate the mean of f in each iteration, while the shaded areas represent the corresponding standard deviations.



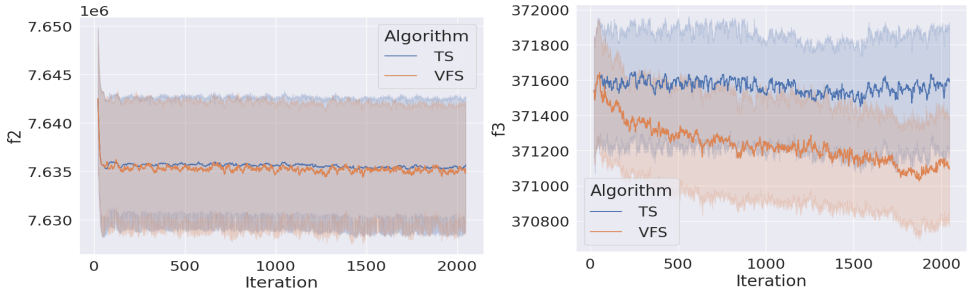
(a1) f^2. **(a2)** f^3.

(a) Dre42.



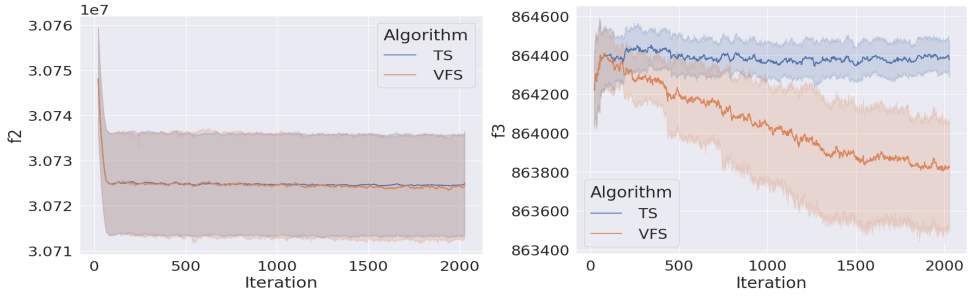
(b1) f^2. **(b2)** f^3.

(b) Dre72.



(c1) f^2. **(c2)** f^3.

(c) Tai45e01.



(d1) f^2. **(d2)** f^3.

(d) Tai75e01.

Figure 3.5: Evolution of the f^2 and f^3 elementary functions during the 10 runs of the TS and VFS algorithms. The solid lines indicate the mean of f^2 and f^3 in each iteration, while the shaded areas represent the corresponding standard deviations.

the VFS, since ensuring that every movement of the algorithm improves at least one of the elementary landscapes causes all elementary functions to be optimized during the search, regardless of their relevance.

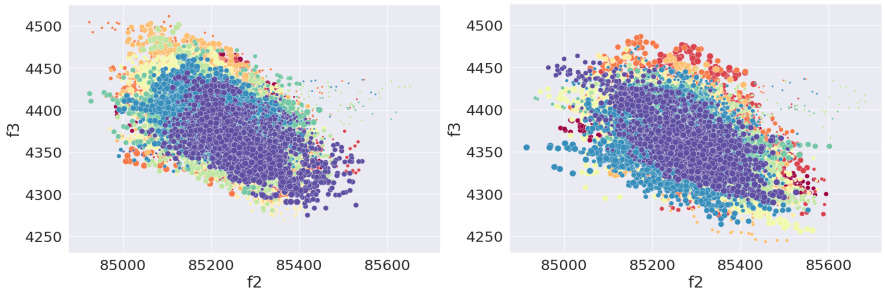
The consequences of these behaviours are different for each of the benchmarks. In the case of the *Dre* instances, optimizing the f^2 elementary function as much as possible (TS) produces better results than trying to obtain solutions with a better f^3 value (VFS). Just the opposite happens in the case of the *Tai-e* instances, in which the f^2 elementary function converges quickly, and therefore, improving f^3 is the only way to avoid stagnation in the search process. Thus, this confirms that some types of instances are more suitable for being solved using the VFS, and vice versa.

However, the previous analysis is still not enough to answer why the TS and VFS perform differently on the *Dre* and *Tai-e* benchmarks. In order to answer this question, we need to go one step further and compare the evolution of the f^2 and f^3 function values during the execution of the TS and VFS algorithms. This will help us to understand which is the relationship between the elementary functions, thus, providing a better insight into the decisions that are made at each step of the algorithms. With this goal, the f^2 and f^3 function values of all the solutions explored during the execution of the algorithms are shown as scatter plots in Figure 3.6.

As can be seen in the plots, the structure of the objective function space of both the *Dre* and the *Tai-e* instances is very different. In the case of the *Dre* instances, all the explored solutions are grouped into a unique contiguous region of the objective function space. Moreover, we can see that the solutions that have lower f^3 values (Y axis) generally have higher f^2 values (X axis), and vice versa. Therefore, it seems like there is a negative correlation between both elementary functions. As f^2 is the most relevant elementary function, this could explain why improving f^3 is not a good strategy in this case. This type of problems in which improving one of the objective functions worsens the others are especially suitable for population based multi-objective algorithms such as the ones in [46].

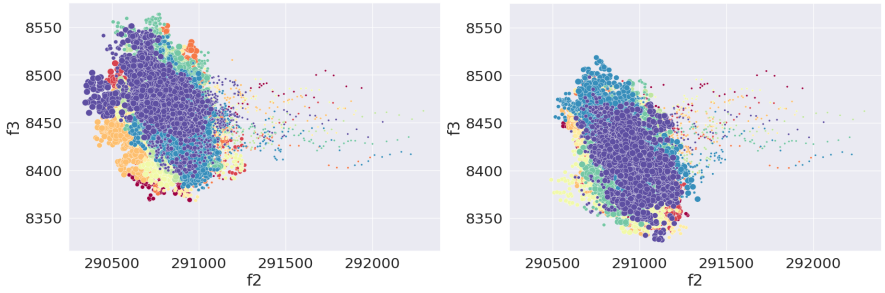
Regarding the *Tai-e* instances, the solutions explored during the executions of the algorithms are grouped into different areas of the objective function space, creating *clusters* of solutions with similar f^2 and f^3 values. This grouping is particularly evident if we focus on the f^2 elementary function, in which two different clusters can be easily distinguished in both the *tai45e01* and the *tai75e01* instances. In those cases, it appears that the starting point of the algorithms determines, to a great extent, the solutions that are visited during the search process, which could explain the objective value variation observed in Figure 3.4. In fact, none of the TS and VFS executions have been able to transfer from one group of solutions to the other. This suggests that there are some sub-optimal regions of the objective function space that are difficult to escape using local search processes, similar to the *funnels* or *sinks* that arise when studying Local Optima Networks (LON) [63, 64]. As explained at the beginning of this section, this may be due to the block structure of the *Tai-e* instances [58]. Finally, we can also observe that there is not a negative correlation between the f^2 and f^3 functions in this case, so improving f^3 will not necessarily worsen the f^2 value.

Based on the previous conclusions, we now can explain the behaviour of the algorithms when solving both the *Dre* and the *Tai-e* instances.



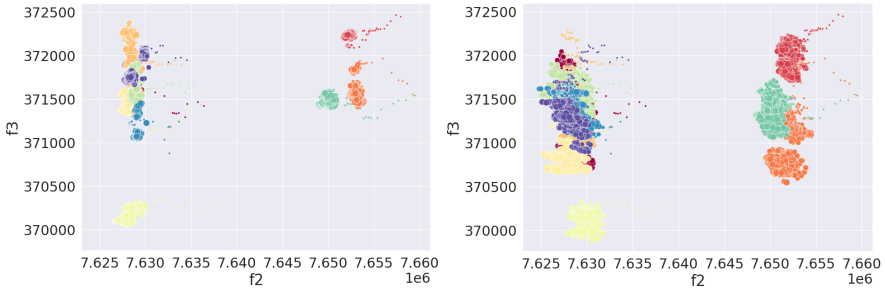
(a1) TS. **(a2)** VFS.

(a) Dre42.



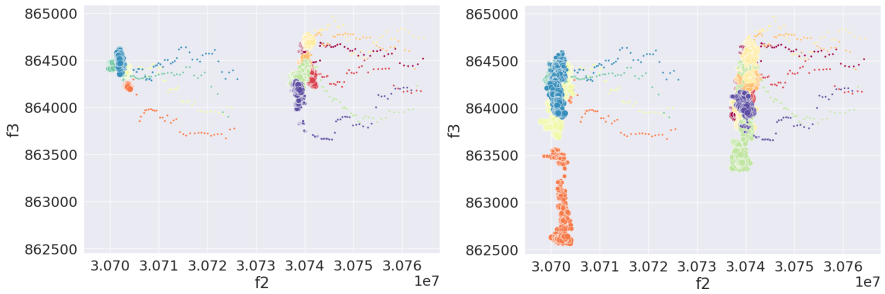
(b1) TS. **(b2)** VFS.

(b) Dre72.



(c1) TS. **(c2)** VFS.

(c) Tai45e01.



(d1) TS. **(d2)** VFS.

(d) Tai75e01.

Figure 3.6: Comparison between the f^2 and f^3 fitness values obtained during the 10 runs of the TS and VFS algorithms. Different colors represent different runs, and the size of the points is directly proportional to the corresponding iteration number.

- In the case of the *Dre* instances, the negative correlation between the elementary functions makes it more promising to just focus on improving the most relevant landscape, or at least doing it jointly. Therefore, the TS algorithm is more suitable for this type of instances.
- In the case of the *Tai-e* instances, the f^2 elementary function rapidly converges to local optima due to the block structure of the problem. Thus, focusing just on the most relevant landscape is not enough in this case. As there is not a negative correlation between the elementary functions, optimizing f^3 may also improve the general fitness of the problem, and therefore, the VFS algorithm produces generally better results than the TS.

It is important to remark that this analysis has only taken into account a small set of instances from two different benchmarks. Thus, a more complete experimentation should be carried out to confirm the conclusions of this section. However, our goal is not to explain the behaviour of the algorithms in all possible situations, but to demonstrate that the TS and VFS algorithms have systematically different performances depending on the characteristics of the instance to be solved. Therefore, analyzing the problem (and particularly, its ELD) may give us hints about which algorithm is more promising for solving each type of instance.

CHAPTER 4

Analysis of the Elementary Landscape Decomposition

Until this point, we have analyzed the behaviour of two different local search based algorithms taking into account the elementary landscapes of the decomposition of the QAP. However, the main drawback of working with the ELD is that we do not really know the aspects of the solutions that are being evaluated by each of the elementary landscapes and their effects on the optimization process. Thus, it is sometimes difficult to decide which landscapes should be optimized at each step of a local search based algorithm (as the VFS). In order to address this problem, in this chapter we propose a decomposition of the components of the ELD that tries to facilitate the analysis of the elementary landscapes.

Before explaining the proposed decomposition, it is important to remark that this approach focuses on decomposing the objective functions of the elementary landscapes, that is, the elementary functions. Thus, for the sake of clarity, in this chapter we mainly talk about the f^1, f^2 and f^3 functions, and not about the landscapes as a whole (which include also the search space and the neighborhood function). However, the reader should keep in mind that the landscape concept is always implicitly present.

4.1 Decomposition of the elementary components

First, let us rewrite the elementary functions of the decomposition (Equations 2.6, 2.7, 2.8) as follows.

$$f^1(\sigma) = \sum_{a=1}^{n-1} \sum_{b=a+1}^{n} \sum_{c=1}^{n-1} \sum_{d=c+1}^{n} \frac{g^1_{(a,b),(c,d)}(\sigma)}{2n} \tag{4.1}$$

$$f^2(\sigma) = \sum_{a=1}^{n-1} \sum_{b=a+1}^{n} \sum_{c=1}^{n-1} \sum_{d=c+1}^{n} \frac{g^2_{(a,b),(c,d)}(\sigma)}{2(n-2)} \tag{4.2}$$

$$f^3(\sigma) = \sum_{a=1}^{n-1} \sum_{b=a+1}^{n} \sum_{c=1}^{n-1} \sum_{d=c+1}^{n} \frac{g^3_{(a,b),(c,d)}(\sigma)}{n(n-2)} \tag{4.3}$$

where f^m is the elementary function that corresponds to the L^m elementary landscape, and $g^m_{(a,b),(c,d)}(\sigma) = \psi_{a,b,c,d}\phi^m_{(a,b)(c,d)}(\sigma) + \psi_{b,a,c,d}\phi^m_{(b,a)(c,d)}(\sigma) + \psi_{a,b,d,c}\phi^m_{(a,b)(d,c)}(\sigma) + \psi_{b,a,d,c}\phi^m_{(b,a)(d,c)}(\sigma)$. As can be seen, the $\sum_{i,p=1}^{n} \psi_{i,i,p,p}\varphi_{(i,i)(p,p)}(\sigma)$ term has been removed from the f^3 function since its value is 0 when all the elements in the main diagonals of the distance and flow matrices are zeros, which happens in virtually all the available benchmark instances. If we focus on symmetric instances, $\psi_{a,b,c,d} = \psi_{b,a,c,d} = \psi_{a,b,d,c} = \psi_{b,a,d,c}$, so we can simplify the previous auxiliary function as $g^m_{(a,b),(c,d)}(\sigma) = \psi_{a,b,c,d}\big(\phi^m_{(a,b)(c,d)}(\sigma) + \phi^m_{(b,a)(c,d)}(\sigma) + \phi^m_{(a,b)(d,c)}(\sigma) + \phi^m_{(b,a)(d,c)}(\sigma)\big)$. Based on Equation 2.9, $g^m_{(a,b),(c,d)}$ has three different possible outcomes when the instance is symmetric:

- If $\sigma(a) = c \wedge \sigma(b) = d$ or $\sigma(a) = d \wedge \sigma(b) = c$, then $g^m_{(a,b),(c,d)} = (2\alpha^m + 2\beta^m)\psi_{a,b,c,d}$.
- If $\sigma(a) = c \oplus \sigma(b) = d$ or $\sigma(a) = d \oplus \sigma(b) = c$, then $g^m_{(a,b),(c,d)} = (2\gamma^m + 2\epsilon^m)\psi_{a,b,c,d}$.
- If $\sigma(a) \neq c, d \wedge \sigma(b) \neq c, d$, then $g^m_{(a,b),(c,d)} = 4\zeta^m\psi_{a,b,c,d}$.

where the parameters $\alpha^m, \beta^m, \gamma^m, \epsilon^m, \zeta^m$ depend on the value of m, that is, the elementary function that we are referring to. If we consider the three cases separately, we can decompose $g^m_{(a,b),(c,d)}$ as follows:

$$\chi^m_{(a,b)(c,d)}(\sigma) = \begin{cases} (2\alpha^m + 2\beta^m)\psi_{a,b,c,d} & \text{if } \sigma(a) = c \wedge \sigma(b) = d \text{ or} \\ & \sigma(a) = d \wedge \sigma(b) = c \\ 0 & \text{Otherwise} \end{cases} \tag{4.4}$$

$$\omega^m_{(a,b)(c,d)}(\sigma) = \begin{cases} (2\gamma^m + 2\epsilon^m)\psi_{a,b,c,d} & \text{if } \sigma(a) = c \oplus \sigma(b) = d \text{ or} \\ & \sigma(a) = d \oplus \sigma(b) = c \\ 0 & \text{Otherwise} \end{cases} \tag{4.5}$$

$$\tau^m_{(a,b)(c,d)}(\sigma) = \begin{cases} 4\zeta^m\psi_{a,b,c,d} & \text{if } \sigma(a) \neq c, d \wedge \sigma(b) \neq c, d \\ 0 & \text{Otherwise} \end{cases} \tag{4.6}$$

Finally, these auxiliary functions χ^m, ω^m and τ^m can be used to decompose each elementary function f^m into three sub-functions:

$$f^m_1(\sigma) = \sum_{a=1}^{n-1}\sum_{b=a+1}^{n}\sum_{c=1}^{n-1}\sum_{d=c+1}^{n} \frac{\chi^m_{(a,b),(c,d)}(\sigma)}{k^m} \tag{4.7}$$

$$f^m_2(\sigma) = \sum_{a=1}^{n-1}\sum_{b=a+1}^{n}\sum_{c=1}^{n-1}\sum_{d=c+1}^{n} \frac{\omega^m_{(a,b),(c,d)}(\sigma)}{k^m} \tag{4.8}$$

$$f^m_3(\sigma) = \sum_{a=1}^{n-1}\sum_{b=a+1}^{n}\sum_{c=1}^{n-1}\sum_{d=c+1}^{n} \frac{\tau^m_{(a,b),(c,d)}(\sigma)}{k^m} \tag{4.9}$$

where $k^1 = 2n$, $k^2 = 2(n-2)$, $k^3 = n(n-2)$ and $f^m(\sigma) = f^m_1(\sigma) + f^m_2(\sigma) + f^m_3(\sigma)$ for every $\sigma \in S_n$. Note that this decomposition is only valid for symmetric instances in which all the elements in the main diagonals of the distance and flow matrices are zeros.

In this work, we only focus on the instances that meet the previous conditions, which are the vast majority of the available benchmark instances described in Section 3.1. However, it is important to remark that similar decompositions can be proposed for other types of instances.

The main advantage of the proposed decomposition is that each of the previous sub-functions (f_1^m, f_2^m, f_3^m) evaluates different aspects of a given solution. For each combination of values a, b, c and d such that $1 \leq a < b \leq n$ and $1 \leq c < d \leq n$, we have that:

- The f_1^m sub-function only returns the output of $g^m_{(a,b),(c,d)}$ when $\sigma(a) = c \wedge \sigma(b) = d$ or $\sigma(a) = d \wedge \sigma(b) = c$. That is, this function only evaluates the combinations of locations-facilities in which both current facilities (c and d) are in the current locations (a and b) in the given solution σ.
- The f_2^m sub-function only returns the output of $g^m_{(a,b),(c,d)}$ when $\sigma(a) = c \oplus \sigma(b) = d$ or $\sigma(a) = d \oplus \sigma(b) = c$. That is, this function only evaluates the combinations of locations-facilities in which just one of the current facilities (c or d) is in one of the current locations (a or b) in the given solution σ.
- The f_3^m sub-function only returns the output of $g^m_{(a,b),(c,d)}$ when $\sigma(a) \neq c, d \wedge \sigma(b) \neq c, d$. That is, this function only evaluates the combinations of locations-facilities in which neither of the current facilities (c and d) is in the current locations (a and b) in the given solution σ.

Thus, it is easy to see that the elementary functions do not only consider information about the current facility-location assignment (f_1^m), but also information about the configurations that could be reached by modifying the facilities assigned to each pair of locations (f_2^m, f_3^m). Since the cases β, γ, ϵ and ζ of Equation 2.9 cancel each other out when the elementary landscapes are combined [31], this additional information is not present in the original landscape of the QAP, and only arises when the ELD is computed.

4.2 Theoretical study

Once we have defined the decomposition of the elementary functions, we now can use this new framework to analyze the aspects of the solutions that are being measured by each elementary landscape. First, let us replace the parameters α^m, β^m, γ^m, ϵ^m and ζ^m in the auxiliary functions $\chi^m_{(a,b)(c,d)}$, $\omega^m_{(a,b)(c,d)}$ and $\tau^m_{(a,b)(c,d)}$ with their actual values according to Table 2.1.

- In the case of the f^1 elementary function, $\alpha^1 = n-3$, $\beta^1 = 1-n$, $\gamma^1 = -2$, $\epsilon^1 = 0$ and $\zeta^1 = -1$. Therefore:

$$\chi^1_{(a,b)(c,d)}(\sigma) = \begin{cases} -4\psi_{a,b,c,d} & \text{if } \sigma(a) = c \wedge \sigma(b) = d \text{ or} \\ & \sigma(a) = d \wedge \sigma(b) = c \\ 0 & \text{Otherwise} \end{cases} \tag{4.10}$$

$$\omega^1_{(a,b)(c,d)}(\sigma) = \begin{cases} -4\psi_{a,b,c,d} & \text{if } \sigma(a) = c \oplus \sigma(b) = d \text{ or} \\ & \sigma(a) = d \oplus \sigma(b) = c \\ 0 & \text{Otherwise} \end{cases} \tag{4.11}$$

$$\tau^1_{(a,b)(c,d)}(\sigma) = \begin{cases} -4\psi_{a,b,c,d} & \text{if } \sigma(a) \neq c,d \wedge \sigma(b) \neq c,d \\ 0 & \text{Otherwise} \end{cases} \tag{4.12}$$

- In the case of the f^2 elementary function, $\alpha^2 = n-3$, $\beta^2 = n-3$, $\gamma^2 = 0$, $\epsilon^2 = 0$ and $\zeta^2 = 1$. Therefore:

$$\chi^2_{(a,b)(c,d)}(\sigma) = \begin{cases} (4n-12)\psi_{a,b,c,d} & \text{if } \sigma(a) = c \wedge \sigma(b) = d \text{ or} \\ & \sigma(a) = d \wedge \sigma(b) = c \\ 0 & \text{Otherwise} \end{cases} \tag{4.13}$$

$$\omega^2_{(a,b)(c,d)}(\sigma) = 0 \tag{4.14}$$

$$\tau^2_{(a,b)(c,d)}(\sigma) = \begin{cases} 4\psi_{a,b,c,d} & \text{if } \sigma(a) \neq c,d \wedge \sigma(b) \neq c,d \\ 0 & \text{Otherwise} \end{cases} \tag{4.15}$$

- In the case of the f^3 elementary function, $\alpha^3 = 2n-3$, $\beta^3 = 1$, $\gamma^3 = n-2$, $\epsilon^3 = 0$ and $\zeta^3 = -1$. Therefore:

$$\chi^3_{(a,b)(c,d)}(\sigma) = \begin{cases} (4n-4)\psi_{a,b,c,d} & \text{if } \sigma(a) = c \wedge \sigma(b) = d \text{ or} \\ & \sigma(a) = d \wedge \sigma(b) = c \\ 0 & \text{Otherwise} \end{cases} \tag{4.16}$$

$$\omega^3_{(a,b)(c,d)}(\sigma) = \begin{cases} (2n-4)\psi_{a,b,c,d} & \text{if } \sigma(a) = c \oplus \sigma(b) = d \text{ or} \\ & \sigma(a) = d \oplus \sigma(b) = c \\ 0 & \text{Otherwise} \end{cases} \tag{4.17}$$

$$\tau^3_{(a,b)(c,d)}(\sigma) = \begin{cases} -4\psi_{a,b,c,d} & \text{if } \sigma(a) \neq c,d \wedge \sigma(b) \neq c,d \\ 0 & \text{Otherwise} \end{cases} \tag{4.18}$$

As can be observed, f^1, f^2 and f^3 are just different linear combinations of all the possible $\psi_{a,b,c,d}$ values such that $1 \leq a < b \leq n$ and $1 \leq c < d \leq n$. The only differences between the elementary functions are the coefficients that are used in the linear combination and the value of the k^m constant. Therefore, analyzing the values of those coefficients can help us understand which of the sub-functions have a negative or positive contribution to the objective value of each elementary function. Considering that the values in the distance and flow matrices of the input instances are non-negative (which, again, happens in virtually all the benchmark instances), we know that $\psi_{a,b,c,d} \geq 0$ for all $1 \leq a < b \leq n$ and $1 \leq c < d \leq n$. With this information, we can determine that:

- **$\mathbf{f^1}$:** According to Equations 4.10, 4.11 and 4.12, all the auxiliary functions return the same value when their particular conditions are met. As stated in [45], this means that the value of f^1 is independent of σ, and thus, the f^1 elementary function is constant (due to the symmetry of the distance and flow matrices). Based on the output values of the auxiliary functions and the k^1 constant, the f^1_1, f^1_2 and f^1_3 sub-functions always have a negative contribution to the f^1 elementary function value. Thus, the value of f^1 is always equal to or less than 0.

- **f^2:** The f^2 elementary function is composed of just two non-zero sub-functions, since $\omega^2_{(a,b)(c,d)}$ (Equation 4.14) is always 0 regardless of the input solution, and thus, f^2_2 also equals 0. Regarding the f^2_1 and f^2_3 sub-functions, we can observe that:
  - Based on the output values of $\chi^2_{(a,b)(c,d)}$ (Equation 4.13) and the k^2 constant, the value of the f^2_1 function is always equal to or greater than 0 if $n \geq 3$. In fact, the exact value of f^2_1 only depends on the sum of all the $\psi_{a,b,c,d}$ such that $\sigma(a) = c \wedge \sigma(b) = d$ or $\sigma(a) = d \wedge \sigma(b) = c$. If that sum increases, then the value of f^2_1 also increases.
  - Based on the output values of $\tau^2_{(a,b)(c,d)}$ (Equation 4.15) and the k^2 constant, the value of the f^2_3 function is always equal to or greater than 0 if $n \geq 3$. In fact, the exact value of f^2_3 only depends on the sum of all the $\psi_{a,b,c,d}$ such that $\sigma(a) \neq c, d \wedge \sigma(b) \neq c, d$. If that sum increases, then the value of f^2_3 also increases.
- **f^3:** With respect to the f^3 elementary function, we can observe that:
  - Based on the output values of $\chi^3_{(a,b)(c,d)}$ (Equation 4.16) and the k^3 constant, the value of the f^3_1 function is always equal to or greater than 0 if $n \geq 3$. In fact, the exact value of f^3_1 only depends on the sum of all the $\psi_{a,b,c,d}$ such that $\sigma(a) = c \wedge \sigma(b) = d$ or $\sigma(a) = d \wedge \sigma(b) = c$. If that sum increases, then the value of f^3_1 also increases.
  - Based on the output values of $\omega^3_{(a,b)(c,d)}$ (Equation 4.17) and the k^3 constant, the value of the f^3_2 function is always equal to or greater than 0 if $n \geq 3$. In fact, the exact value of f^3_2 only depends on the sum of all the $\psi_{a,b,c,d}$ such that $\sigma(a) = c \oplus \sigma(b) = d$ or $\sigma(a) = d \oplus \sigma(b) = c$. If that sum increases, then the value of f^3_2 also increases.
  - Based on the output values of $\tau^3_{(a,b)(c,d)}$ (Equation 4.18) and the k^3 constant, the value of the f^3_3 function is always equal to or **less** than 0 if $n \geq 3$. In fact, the exact value of f^3_3 only depends on the sum of all the $\psi_{a,b,c,d}$ such that $\sigma(a) \neq c, d \wedge \sigma(b) \neq c, d$. If that sum **increases**, then the value of f^3_3 **decreases**.

In addition to the sign of the sub-functions, studying the coefficients in χ^m, ω^m and τ^m can give us information about which sub-functions have a higher relative contribution to the general objective value of each elementary function. However, we have to take into account that, for any solution $\sigma \in S_n$, the amount of combinations of a, b, c and d such that $1 \leq a < b \leq n$ and $1 \leq c < d \leq n$ that meet the first condition of each auxiliary function is different. From the total number of possible combinations $\frac{(n^2-n)^2}{4}$, only $\frac{n^2-n}{2}$ combinations satisfy that $\sigma(a) = c \wedge \sigma(b) = d$ or $\sigma(a) = d \wedge \sigma(b) = c$ (α~β case). On the other hand, exactly $(n^2 - n)(n - 2)$ combinations satisfy that $\sigma(a) = c \oplus \sigma(b) = d$ or $\sigma(a) = d \oplus \sigma(b) = c$ (γ~ϵ case). Finally, the remaining $\frac{(n^2-n)((n-2)^2-(n-2))}{4}$ combinations fulfill that $\sigma(a) \neq c, d \wedge \sigma(b) \neq c, d$ (ζ case). If we look at Figure 4.1, we can observe that the amount of combinations that meet the first condition of τ^m increases very fast with the instance size ($O(n^4)$), while the number of combinations corresponding to χ^m and ω^m grows slower ($O(n^2)$ and $O(n^3)$ respectively).

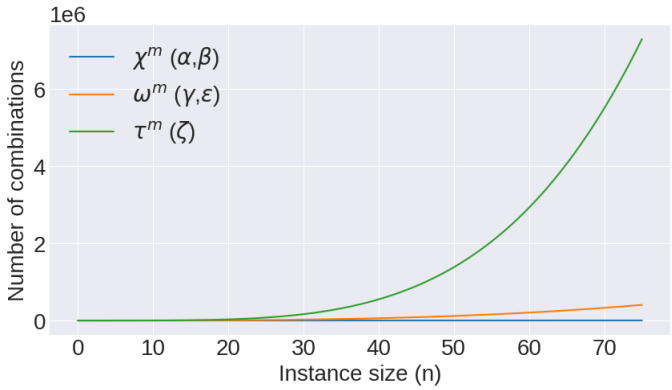


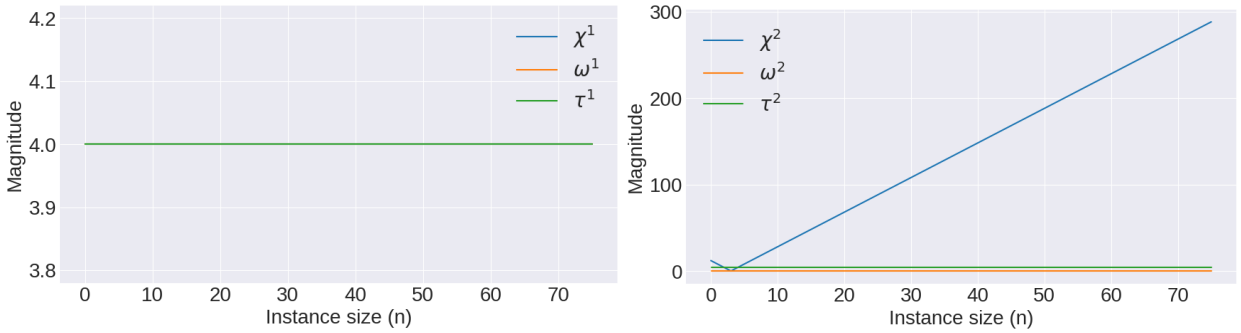
Figure 4.1: Comparison between the number of combinations of a, b, c and d that satisfy the first condition of each auxiliary function (χ^m, ω^m, τ^m) as a function of the instance size n.

Thus, if we focus on the summations of Equations 4.7, 4.8 and 4.9, the f_3^m sub-function is the one with the highest amount of "significant" terms (that is, the ones that satisfy the first condition of the corresponding auxiliary function), followed by f_2^m and f_1^m (for large enough values of n). However, this is not enough to analyze which sub-functions have a higher objective value contribution in each elementary function, since there is another factor that has to be taken into account: the magnitude of the coefficients in the linear combination (Figure 4.2). If we look at the output values of the χ^m, ω^m and τ^m auxiliary functions, we have that:

- **$\mathbf{f^1}$:** According to Equations 4.10, 4.11 and 4.12, all the auxiliary functions return $-4\psi_{a,b,c,d}$ when their conditions are met. Therefore, every term in the summations of f_1^1, f_2^1 and f_3^1 has the same relative contribution to the fitness of the elementary function f^1.
- **$\mathbf{f^2}$:** According to Equations 4.13 and 4.15, the magnitude of the multiplication factor that is applied to the corresponding $\psi_{a,b,c,d}$ value is higher in the case of χ^2 when $n > 4$. Therefore, the individual terms in the summation of f_1^2 have a higher relative contribution to the fitness of the elementary function f^2 than those in the summation of f_3^2.
- **$\mathbf{f^3}$:** According to Equations 4.16, 4.17 and 4.18, the χ^3 auxiliary function is the one that has the multiplication factor with the highest magnitude when $n > 2$. Conversely, τ^3 has the lowest magnitude factor when $n > 4$. Therefore, the individual terms in the summation of f_1^3 have the highest relative contribution to the f^3 function value, followed by f_2^3 and f_3^3.

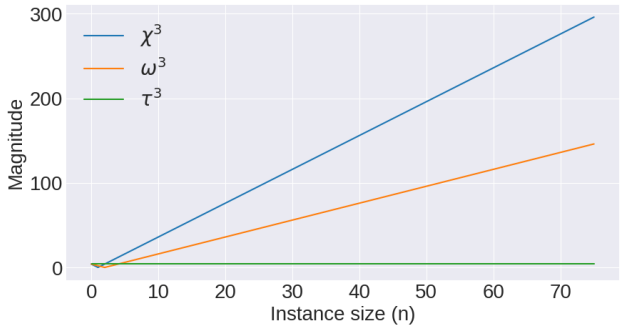
Thus, although the summations in f_2^m and f_3^m have generally more significant terms than f_1^m, the weight of each term in the first sub-function is generally higher.

Based on both the number of significant terms and their corresponding coefficients, we now can compute the exact average values of the sub-functions of the proposed decomposition. As all the sub-functions are just linear combinations of $\psi_{a,b,c,d}$, the expected values can be computed considering that $E[cX] = cE[X]$, where X is a random variable (in this case, $\psi_{a,b,c,d}$) and c is a constant (in this case, the number of significant terms multiplied by



(a) f^1.

(b) f^2.



(c) f^3.

Figure 4.2: Evolution of the magnitude of the coefficients that are used in the auxiliary functions (χ^m, ω^m, τ^m) as a function of the instance size.

their coefficient) [65]. The calculated values are shown in Table 4.1. As can be observed in the table, the f_3^m sub-function is the one with the largest absolute expected value in the f^1 and f^2 elementary functions ($O(n^3)$), and thus, the one with the highest objective value contribution in those cases. In the case of f^3, however, both the f_2^3 and f_3^3 sub-functions have a similar expected contribution ($O(n^2)$).

Until now, we have analyzed some relevant characteristics of the elementary landscapes using the decomposition framework that we have proposed. However, we have not yet studied the effects of these features on local search optimization processes. That is, we still have to analyze how the elementary functions (and thus, the corresponding sub-functions) vary when moving from one solution to another in the search space.

As we have mentioned before, the value of each elementary function depends on the combinations of a, b, c and d that meet the conditions of the f_1^m, f_2^m and f_3^m sub-functions. When moving between solutions using local search processes, the combinations that correspond to each of the sub-functions change, modifying the elementary function values [31]. Considering the swap neighborhood, moving from any solution $\sigma \in S_n$ to another neighboring solution $\sigma' \in N(\sigma)$ produces the following changes (Figure 4.3):

- $2(n-2)$ combinations switch from the α~β case (f_1^m) to the γ~ϵ case (f_2^m), and vice versa.
- $2(n-2)(n-3)$ combinations switch from the γ~ϵ case (f_2^m) to the ζ case (f_3^m), and vice versa.

Table 4.1: Expected values of the sub-functions over the entire search space as a function of the instance size (n). $\bar{\psi}$ represents the average value of $\psi_{a,b,c,d}$ for every $1 \leq a, b, c, d \leq n$ such that $a \neq b$ and $c \neq d$.

		Expected value
f^1	f_1^1	$\frac{-4}{2n}\frac{n^2-n}{2}\bar{\psi} = -(\mathbf{n-1})\bar{\boldsymbol{\psi}}$
	f_2^1	$\frac{-4}{2n}(n^2-n)(n-2)\bar{\psi} = -\mathbf{2}(\mathbf{n-1})(\mathbf{n-2})\bar{\boldsymbol{\psi}}$
	f_3^1	$\frac{-4}{2n}\frac{(n^2-n)((n-2)^2-(n-2))}{4}\bar{\psi} = -\frac{(\mathbf{n-1})((\mathbf{n-2})^\mathbf{2}-(\mathbf{n-2}))}{\mathbf{2}}\bar{\boldsymbol{\psi}}$
f^2	f_1^2	$\frac{4n-12}{2(n-2)}\frac{n^2-n}{2}\bar{\psi} = \frac{(\mathbf{n-3})(\mathbf{n^2-n})}{\mathbf{n-2}}\bar{\boldsymbol{\psi}}$
	f_2^2	$\frac{0}{2(n-2)}(n^2-n)(n-2)\bar{\psi} = \mathbf{0}$
	f_3^2	$\frac{4}{2(n-2)}\frac{(n^2-n)((n-2)^2-(n-2))}{4}\bar{\psi} = \frac{(\mathbf{n^2-n})(\mathbf{n-3})}{\mathbf{2}}\bar{\boldsymbol{\psi}}$
f^3	f_1^3	$\frac{4n-4}{n(n-2)}\frac{n^2-n}{2}\bar{\psi} = \frac{\mathbf{2}(\mathbf{n-1})^\mathbf{2}}{\mathbf{n-2}}\boldsymbol{\psi}$
	f_2^3	$\frac{2n-4}{n(n-2)}(n^2-n)(n-2)\bar{\psi} = \mathbf{2}(\mathbf{n-1})(\mathbf{n-2})\bar{\boldsymbol{\psi}}$
	f_3^3	$\frac{-4}{n(n-2)}\frac{(n^2-n)((n-2)^2-(n-2))}{4}\bar{\psi} = -(\mathbf{n-1})(\mathbf{n-3})\bar{\boldsymbol{\psi}}$

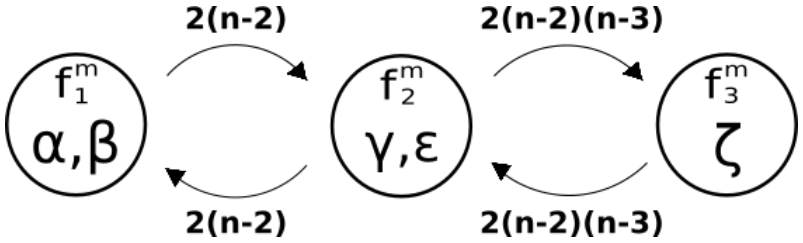


Figure 4.3: Transition graph that represents the amount of combinations that switch from one case (sub-function) to another after performing a swap movement.

As can be observed, the f_2^m and f_3^m sub-functions are the ones that lead to the largest amount of variations ($O(n^2)$). However, we have to take into account that, as we have explained before, the magnitude of the significant terms in each of the sub-functions is very different, and hence, the magnitude of the individual variations that happen after a swap movement is different too. For example, in the case of the f^1 elementary function (Equations 4.10, 4.11, 4.12), all the sub-functions have the same exact coefficient, so it is easy to see that the f_2^1 and f_3^1 sub-functions will generally suffer the largest change in the objective value. On the other hand, if we focus on f^2 (Equations 4.13, 4.14, 4.15), the f_1^2 sub-function has a $O(n)$ coefficient, while f_3^2 has a $O(1)$ coefficient and f_2^2 is always 0. Thus, in this case, it is more difficult to discern which of the sub-functions (f_1^2 or f_3^2) undergoes a greater objective value variation when moving between neighbor solutions. Finally, regarding the f^3 elementary function (Equations 4.16, 4.17, 4.18), both the f_1^3 and f_2^3 sub-functions have a $O(n)$ coefficient, which suggests that f_2^3 is the one that suffers the largest fitness variations.

The sub-functions that undergo a greater objective value variation can give us hints about what aspects are being considered during the optimization of each elementary function. For example, if we optimize f^3, the sub-function that has the greatest impact on

the neighborhood movement selection (the one with the greatest variation) is f_2^3, that is, we are mainly considering the combinations of locations-facilities in which just one of the current facilities (c or d) is in one of the current locations (a or b) in the given solution. So, as can be seen, the proposed decomposition provides a better insight into the real meaning of each of the elementary components and their effect on the optimization process.

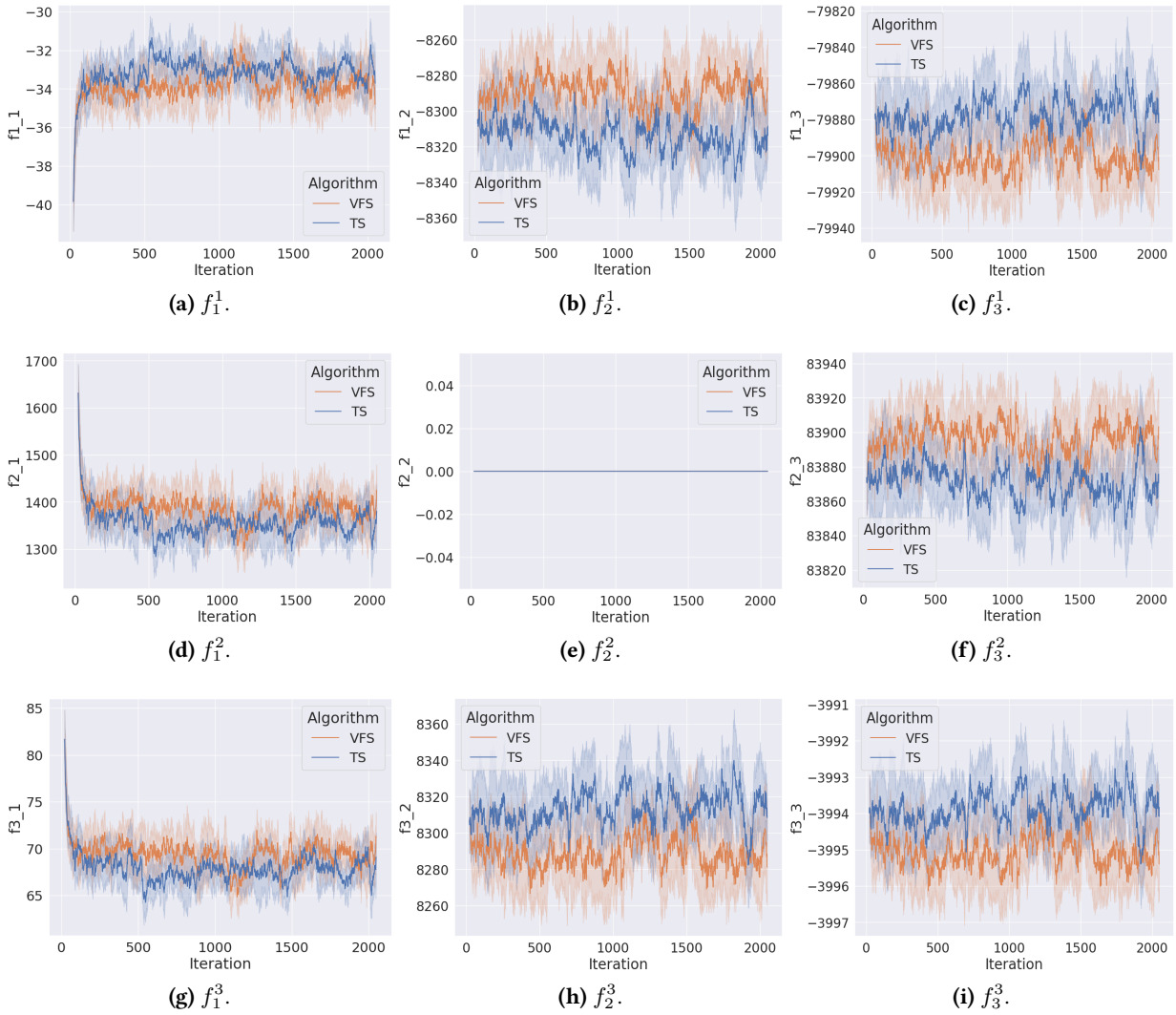
4.3 Experimental study

In order to experimentally verify the characteristics of the decomposition discussed in the previous section, we now consider two representative benchmark instances that were already used in Section 3.3: *dre42* and *tai45e01*. As a reminder, these two instances are part of the *Dre* and *Tai-e* benchmarks, which contain difficult instances with very different problem structures [58].

First, let us analyze the effects of the elementary function optimization in the decomposed sub-functions. With this aim, we decompose the fitness evolution of the elementary functions during the TS and VFS executions of Section 3.3 (Figures 4.4 and 4.5). As can be seen, in this case f^1 is also included since its individual sub-functions are not constant.

The first interesting thing that can be observed in Figures 4.4 and 4.5 is that some sub-functions are not being minimized during the optimization process. That is, even when the objective value of an elementary function decreases, this does not mean that all the sub-functions of the decomposition are reduced. As we have seen before, the sign and optimization direction of the sub-functions depends on the sign of the coefficients, so one same sub-function could be minimized in one of the elementary functions and maximized in another. For example, the f_1^1 sub-function is maximized during the optimization of both the *dre42* and *tai45e01* instances, while f_1^2 and f_1^3 are minimized during that same process. If we take into account the information given by the f_1^m sub-function and the sign of its coefficients in each elementary function, we can observe that this phenomenon means that the local search algorithms are minimizing the sum of all the $\psi_{a,b,c,d}$ such that $\sigma(a) = c \wedge \sigma(b) = d$ or $\sigma(a) = d \wedge \sigma(b) = c$. That is, they are minimizing the sum of the cases in which both current facilities (c and d) are in the current locations (a and b) in the given solution σ. This is precisely what the original objective function of the QAP (f) measures [31], so it makes sense because both the TS and VFS are based on minimizing f. This analysis can be repeated for the f_2^m and f_3^m sub-functions in order to explore which combinations of locations and facilities are being minimized or maximized.

In addition to the sign and optimization direction, another interesting characteristic that has been mentioned during the theoretical analysis is the variation of the sub-function values when moving between neighbor solutions in the search space. This issue has been studied by computing the objective value variations between a set of random (swap) neighbor solutions in the search space of the *dre42* and *tai45e01* instances. In particular, we have considered 10,000 random pairs of solutions $\sigma_1, \sigma_2 \in S_n$ such that $\sigma_2 \in N(\sigma_1)$ per instance. The objective value variations $f_w^m(\sigma_1) - f_w^m(\sigma_2)$ have been computed for each of the sub-functions in the decomposition ($1 \leq m, w \leq 3$). Then, a comparison between the fitness variations of the sub-functions that belong to each elementary function has been carried out. The obtained results are shown in Figures 4.6 and 4.7. The mean magnitude differences between the variations of the sub-functions are also shown in Figure 4.8.



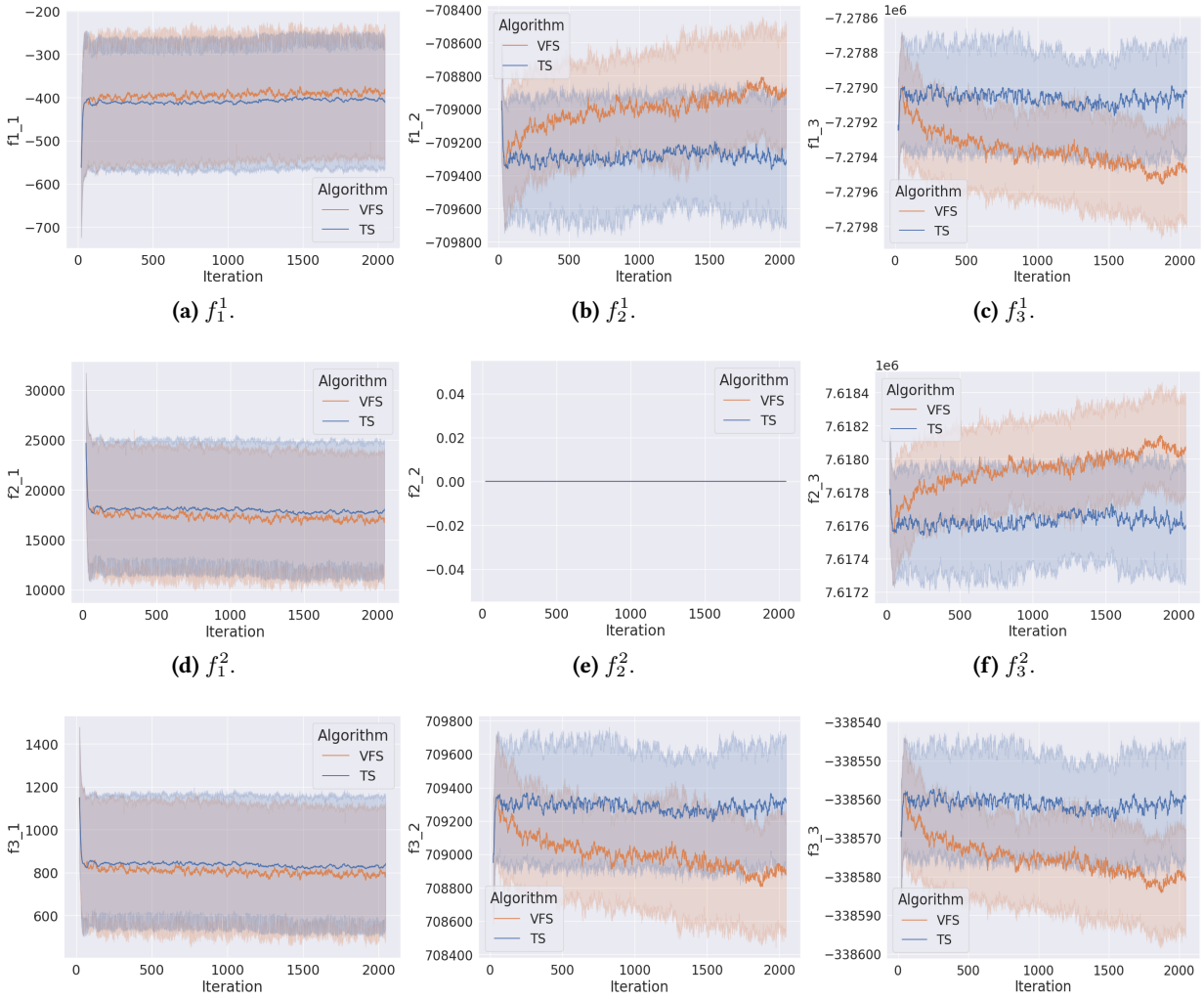
(a) f_1^1. **(b)** f_2^1. **(c)** f_3^1.

(d) f_1^2. **(e)** f_2^2. **(f)** f_3^2.

(g) f_1^3. **(h)** f_2^3. **(i)** f_3^3.

Figure 4.4: Evolution of the sub-functions of the proposed decomposition during the 10 runs of the TS and VFS algorithms for the *dre42* instance. The solid lines indicate the mean of the sub-functions in each iteration, while the shaded areas represent the corresponding standard deviations.

First, the point clouds in Figures 4.6 and 4.7 show that the magnitude of the variations in the sub-functions that belong to each elementary function are pretty different, which is also confirmed by the mean magnitude differences computed in Figure 4.8. In the case of f^1, for example, both the f_2^1 and f_3^1 sub-functions exhibit a wider range of variation than f_1^1 in both instances. This means that the f_2^1 and f_3^1 sub-functions have a larger impact on the neighborhood movement decision process of local search algorithms. Regarding the rest of the elementary functions, f_1^2 seems to be the most influential sub-function (the one with the largest variations) in the case of f^2, while the f_2^3 sub-function seems to dominate the decision process when we consider f^3. Additionally, if we examine all the elementary functions at the same time, the magnitude of the fitness variations in f_1^2 seems to be higher than the magnitude of the variations in any other sub-function of the decomposition, and not only those in f^2. Therefore, this particular sub-function seems to be the most influential one when optimizing the general objective function f using local



(a) f_1^1. **(b)** f_2^1. **(c)** f_3^1.

(d) f_1^2. **(e)** f_2^2. **(f)** f_3^2.

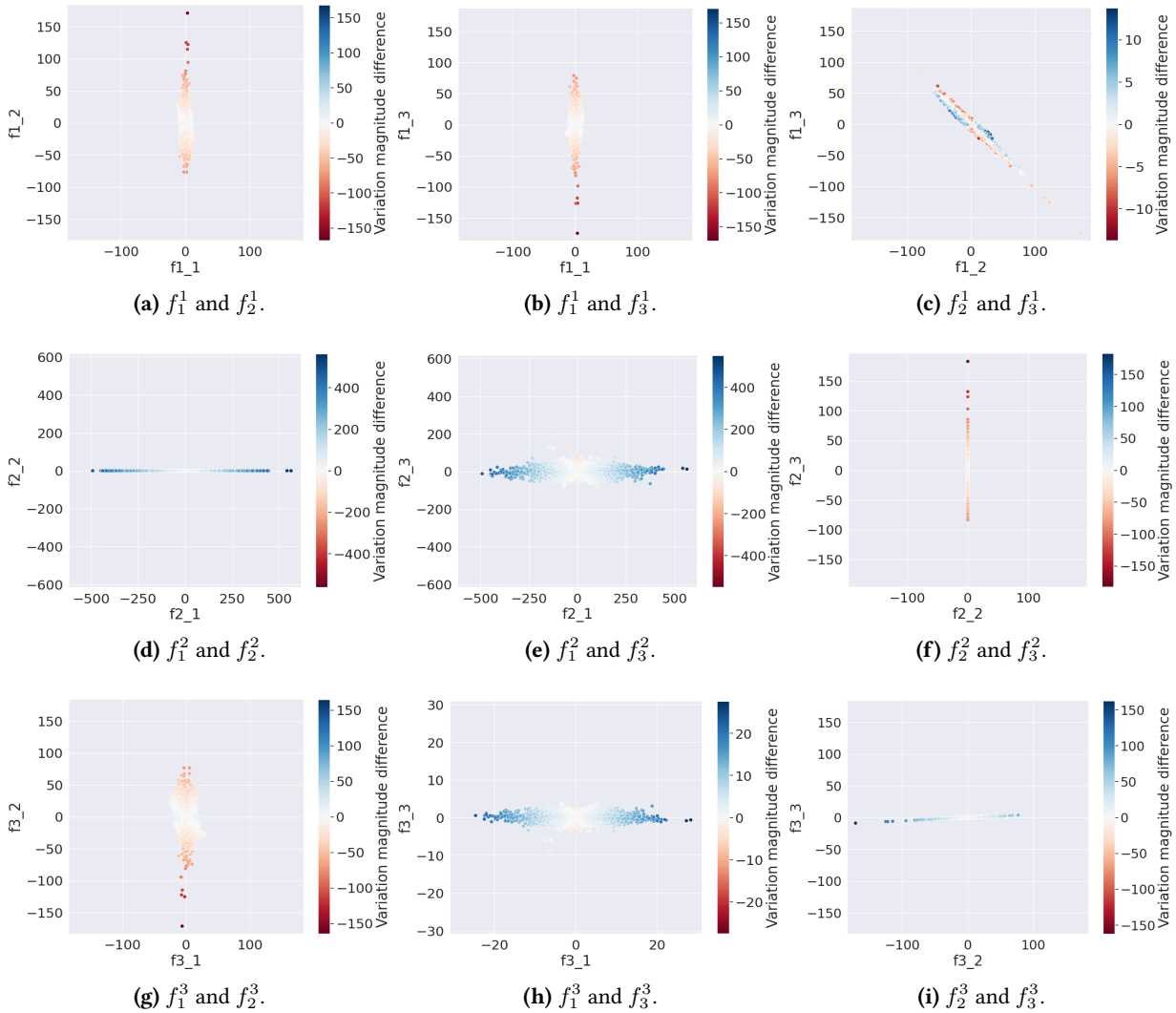
(g) f_1^3. **(h)** f_2^3. **(i)** f_3^3.

Figure 4.5: Evolution of the sub-functions of the proposed decomposition during the 10 runs of the TS and VFS algorithms for the *tai45e01* instance. The solid lines indicate the mean of the sub-functions in each iteration, while the shaded areas represent the corresponding standard deviations.

searches, at least according to the analyzed instances.

All these experimental results are in line with the conclusions drawn during the theoretical analysis. Moreover, they can be confirmed if we look at the shape of the line plots in Figures 4.4 and 4.5. If we focus on the plots that correspond to f^2, we can see that the evolution of f_1^2 has a very similar shape to the evolution of f^2 shown in Figure 3.5. Something similar happens if we look at the f_2^3 and f^3 evolution plots. Thus, this confirms that the behaviour of local search algorithms in each of the elementary functions is predominantly defined by one of the sub-functions.

Finally, another interesting property that can be seen in Figures 4.6 and 4.7 is that the variations of the f_2^m and f_3^m sub-functions are highly correlated, with the only exception of the f^2 case due to f_2^2 being always 0. This suggests that there is an important objective value relationship between the combinations of facilities and locations that meet the conditions of the γ~ϵ and ζ cases. This phenomenon should be further studied in the future in order

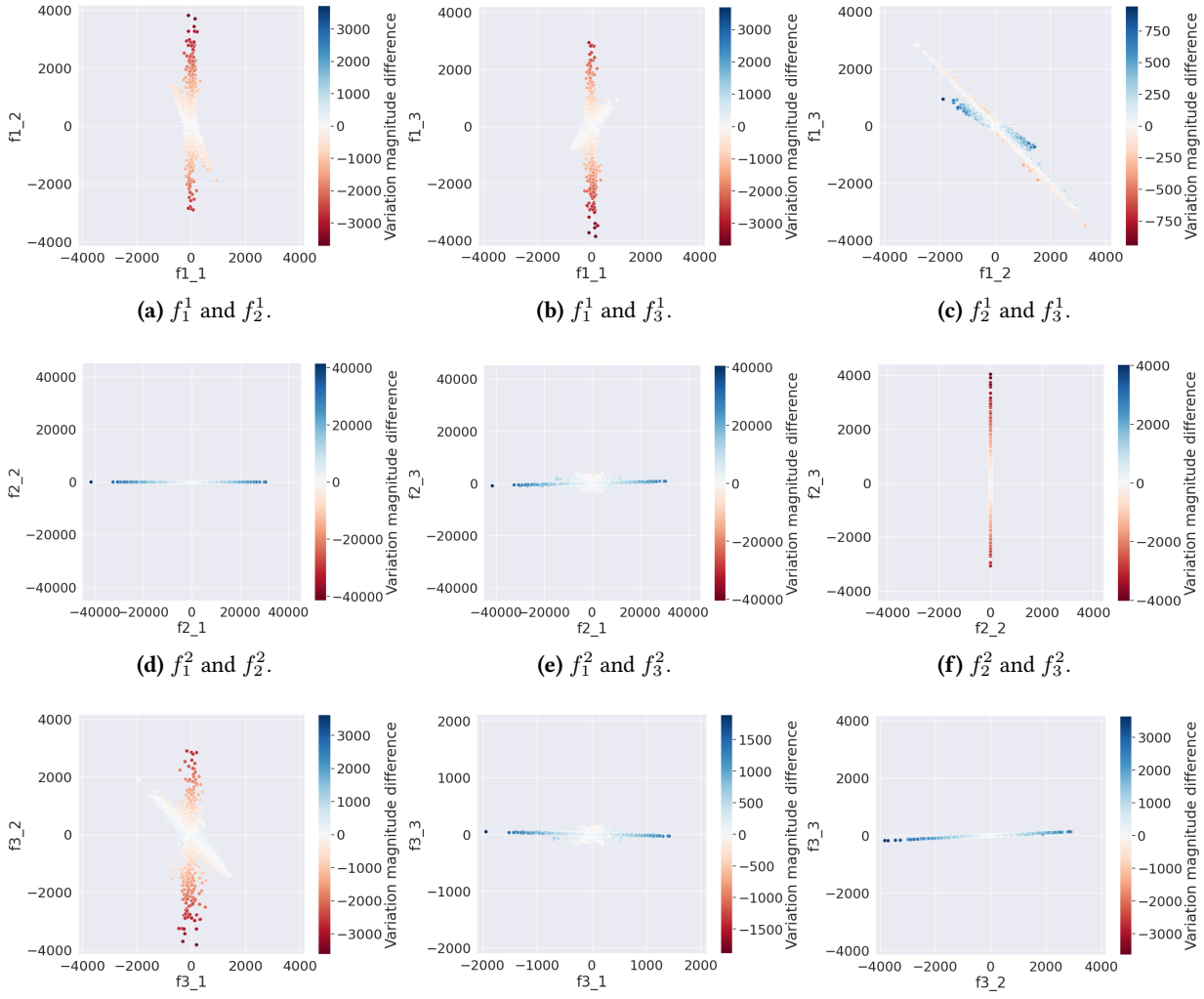


(a) f_1^1 and f_2^1. **(b)** f_1^1 and f_3^1. **(c)** f_2^1 and f_3^1.

(d) f_1^2 and f_2^2. **(e)** f_1^2 and f_3^2. **(f)** f_2^2 and f_3^2.

(g) f_1^3 and f_2^3. **(h)** f_1^3 and f_3^3. **(i)** f_2^3 and f_3^3.

Figure 4.6: Comparison of the sub-function value variations in 10,000 random pairs of (swap) neighbor solutions of the $dre42$ instance. Each axis represents the variation of a different sub-function, each point represents a pair of neighbor solutions, and the colors of the points indicate the magnitude difference between the variations of the corresponding sub-functions (absolute value in the X axis minus absolute value in the Y axis).

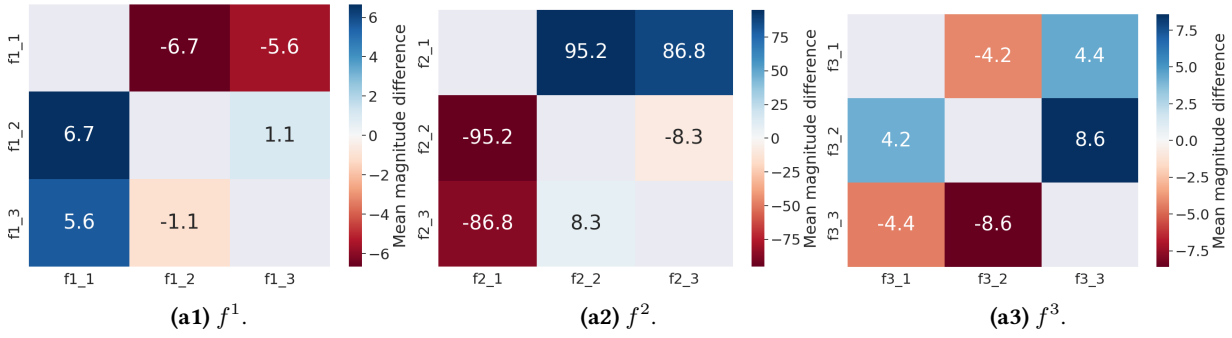


(a) f_1^1 and f_2^1. **(b)** f_1^1 and f_3^1. **(c)** f_2^1 and f_3^1.

(d) f_1^2 and f_2^2. **(e)** f_1^2 and f_3^2. **(f)** f_2^2 and f_3^2.

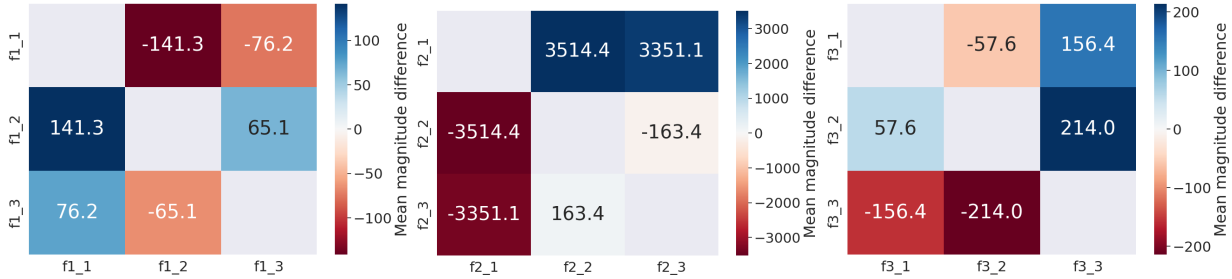
(g) f_1^3 and f_2^3. **(h)** f_1^3 and f_3^3. **(i)** f_2^3 and f_3^3.

Figure 4.7: Comparison of the sub-function value variations in 10,000 random pairs of (swap) neighbor solutions of the $tai45e01$ instance. Each axis represents the variation of a different sub-function, each point represents a pair of neighbor solutions, and the colors of the points indicate the magnitude difference between the variations of the corresponding sub-functions (absolute value in the X axis minus absolute value in the Y axis).



(a1) f^1. **(a2)** f^2. **(a3)** f^3.

(a) *Dre42* instance.



(b1) f^1. **(b2)** f^2. **(b3)** f^3.

(b) *Tai45e01* instance.

Figure 4.8: Mean differences between the magnitudes of the sub-function value variations in 10,000 random pairs of (swap) neighbor solutions of the *dre42* and *tai45e01* instances. Blue colors represent that the sub-function in the Y axis has larger magnitude variations, while red colors represent just the opposite.

to avoid potential redundancies when using the ELD to analyze the problem.

4.4 Implications

The previous theoretical and experimental analyses can help us understand some particularities of the elementary landscape decomposition of the QAP:

- In [45], they noticed that the f^2 elementary function is the one that generally has the highest influence on the variance of the original objective function of the QAP when the instance is symmetric. As we have just seen, this may happen because most of the objective value variation of f^2 comes from the f_1^2 sub-function. This sub-function measures the same combinations of locations and facilities as the f objective function ($\alpha \sim \beta$ case), so it ensures that the variations in the fitness of f^2 are similar to those in f. Moreover, f_1^2 appears to be the sub-function with the highest magnitude variations in the entire decomposition, which would guarantee that the f^2 elementary function has a high contribution to the general objective value variance. This is not the case if we consider the f^3 elementary function, since most of its fitness variation comes from the f_2^3 sub-function, which measures information that is not present in the original landscape of the problem ($\gamma \sim \epsilon$ case).

- Regarding the local search algorithms that we studied in Chapter 3, the proposed decomposition can also help us explain the behaviour of the TS and VFS algorithms in different types of instances. In the case of the *Dre* instances, for example, we observed that focusing on optimizing the f^2 elementary function seems to be the best strategy. That is, it seems that optimizing the α~β case is just enough to obtain good results. In the case of the *Tai-e* instances, however, optimizing just the α~β case causes the algorithm to get stuck in low quality local optima. In this situation, considering information that is not in the original formulation of the problem (that is, the γ~ϵ and ζ cases) may help to escape the local optima, and thus, optimizing the f^3 elementary function becomes particularly useful.

In short, it is easy to see that the proposed decomposition allows us to delve deeper into the ELD of the problem and its effects on local search processes. Taking this into account, future research lines should focus on finding the relationship between the decomposed landscapes and the characteristics of QAP instances. By doing so, we would be able to know which aspects of the solution should be optimized in order to find good solutions for any given instance. This information could then be used to select the most suitable algorithm for each situation.

CHAPTER 5

Conclusions and future work

As observed in this work, the elementary landscape decomposition is a useful tool to better understand the underlying components that form a particular combinatorial optimization problem. In the case of the quadratic assignment problem, we have used the elementary landscape decomposition to try to understand the behaviour of two local search based algorithms: the Tabu Search (TS) and the Variable Function Search (VFS). Through this analysis, we have been able to point out the performance differences between the algorithms in different types of instances, and we have successfully linked those differences with the elementary landscapes that compose the problem.

In order to go a step further on the analysis of the ELD, we have also proposed an additional decomposition that allows us to have a deeper insight into the characteristics of the elementary landscapes. Given a symmetric instance with null main diagonals, we have proved that the elementary functions of the QAP are composed of three sub-functions that measure different combinations of locations and facilities in the solutions. The elementary functions only differ in the coefficients that measure the contribution of each type of combination of locations and facilities. Therefore, analyzing the properties of the sub-functions can help us understand which are the components of the solution that have a larger impact on each of the elementary functions. With this information, the elementary landscape decomposition could be effectively used to optimize specific characteristics of a QAP instance.

However, there is still much work to be done in order to fully understand the elementary landscapes that form the QAP. For example, the proposed additional decomposition is only valid for a particular type of QAP instances. Although it covers the vast majority of the benchmark instances that are available online, it would be interesting to propose a more general decomposition scheme. Moreover, the carried out analysis only focuses on some general characteristics of the sub-functions and their relationship with the corresponding elementary function. Thus, a more complete analysis should be performed from both theoretical and practical point of views. Just as an example, finding closed expressions for the average fitness variation of the sub-functions when moving between solutions in the search space could help us understand the optimization process of the elementary landscapes.

Another topic that has been left unexplored in this work is the reasons why the characteristics of the ELD vary depending on the instance type. Although previous works have already discovered that the symmetry of the instances is a key factor that determines the structure of the ELD [45], we still do not know which other aspects of the input instance should be considered when analyzing the problem decomposition. For example, in this work, we have encountered some evidences that suggest that having a block distance or flow matrix may hinder the optimization of the f^2 elementary function. However, we have not been able to discover why this happens. Thus, it would be interesting to perform additional comparative studies in order to clarify the relationship between the characteristics of the instance matrices and the ELD of the problem.

Finally, regarding the practical applications of the elementary landscape decomposition of the QAP, all the information gathered during the analysis of the elementary landscapes could be used to create new meta-heuristic algorithms that efficiently solve the problem. For example, the VFS algorithm could be modified to focus the optimization process on one landscape or another, depending on the search status of the algorithm. This decision could be carried out by applying Neural Combinatorial Optimization (NCO) techniques [66, 67] that automatically decide which elementary landscape is more promising at each step of the algorithm. Additionally, this idea could be applied to other combinatorial optimization problems with a known ELD in order to check if this strategy can be extended to different types of contexts.

Bibliography

[1] B. H. Korte, J. Vygen, B. Korte, J. Vygen, Combinatorial optimization, Vol. 1, Springer, 2011. See page 1.

[2] C. H. Papadimitriou, K. Steiglitz, Combinatorial optimization: algorithms and complexity, Courier Corporation, 1998. See page 1.

[3] D. Du, P. M. Pardalos, Handbook of combinatorial optimization, Vol. 4, Springer Science & Business Media, 1998. See page 1.

[4] T. L. Magnanti, Combinatorial optimization and vehicle fleet planning: Perspectives and prospects, Networks 11 (2) (1981) 179–213. See page 1.

[5] A. A. Bakhtiari, H. Navid, J. Mehri, D. D. Bochtis, Optimal route planning of agricultural field operations using ant colony optimization, Agricultural Engineering International: CIGR Journal 13 (4) (2011). See page 1.

[6] X.-S. Zhang, Z. Li, R.-S. Wang, Y. Wang, A combinatorial model and algorithm for globally searching community structure in complex networks, Journal of combinatorial optimization 23 (4) (2012) 425–442. See page 1.

[7] G. Naseri, M. A. Koffas, Application of combinatorial optimization strategies in synthetic biology, Nature communications 11 (1) (2020) 1–14. See page 1.

[8] E. Alba, G. Luque, A new local search algorithm for the dna fragment assembly problem, in: European Conference on Evolutionary Computation in Combinatorial Optimization, Springer, 2007, pp. 1–12. See page 1.

[9] S. Arora, B. Barak, Computational complexity: a modern approach, Cambridge University Press, 2009. See page 1.

[10] M. Jünger, G. Reinelt, G. Rinaldi, The traveling salesman problem, Handbooks in operations research and management science 7 (1995) 225–330. See page 1.

[11] J. Ceberio, A. Mendiburu, J. A. Lozano, The linear ordering problem revisited, European Journal of Operational Research 241 (3) (2015) 686–696. See page 1.

[12] T. S. Caetano, J. J. McAuley, L. Cheng, Q. V. Le, A. J. Smola, Learning graph matching, IEEE transactions on pattern analysis and machine intelligence 31 (6) (2009) 1048–1058. See page 1.

[13] E. L. Lawler, D. E. Wood, Branch-and-bound methods: A survey, Operations research 14 (4) (1966) 699–719. See page 1.

[14] J. E. Mitchell, Branch-and-cut algorithms for combinatorial optimization problems, Handbook of applied optimization 1 (1) (2002) 65–77. See page 1.

[15] S. Voß, Meta-heuristics: The state of the art, in: Workshop on Local Search for Planning and Scheduling, Springer, 2000, pp. 1–23. See page 1.

[16] E.-G. Talbi, A taxonomy of hybrid metaheuristics, Journal of heuristics 8 (5) (2002) 541–564. See page 1.

[17] E. Aarts, E. H. Aarts, J. K. Lenstra, Local search in combinatorial optimization, Princeton University Press, 2003. See page 1.

[18] H. R. Lourenço, O. C. Martin, T. Stützle, Iterated local search, in: Handbook of metaheuristics, Springer, 2003, pp. 320–353. See page 1.

[19] F. Glover, Tabu search: A tutorial, Interfaces 20 (4) (1990) 74–94. See pages 1, 2.

[20] P. J. Van Laarhoven, E. H. Aarts, Simulated annealing, in: Simulated annealing: Theory and applications, Springer, 1987, pp. 7–15. See page 1.

[21] Z. Beheshti, S. M. H. Shamsuddin, A review of population-based meta-heuristic algorithms, Int. J. Adv. Soft Comput. Appl 5 (1) (2013) 1–35. See page 1.

[22] S. Sivanandam, S. Deepa, Genetic algorithms, in: Introduction to genetic algorithms, Springer, 2008, pp. 15–37. See page 1.

[23] M. Dorigo, M. Birattari, T. Stutzle, Ant colony optimization, IEEE computational intelligence magazine 1 (4) (2006) 28–39. See page 1.

[24] P. Larrañaga, J. A. Lozano, Estimation of distribution algorithms: A new tool for evolutionary computation, Vol. 2, Springer Science & Business Media, 2001. See page 1.

[25] T. C. Koopmans, M. Beckmann, Assignment problems and the location of economic activities, Econometrica: journal of the Econometric Society (1957) 53–76. See pages 1, 5.

[26] E. L. Lawler, The quadratic assignment problem, Management science 9 (4) (1963) 586–599. See page 1.

[27] A. N. Elshafei, Hospital layout as a quadratic assignment problem, Journal of the Operational Research Society 28 (1) (1977) 167–179. See page 1.

[28] R. E. Burkard, J. Offermann, Entwurf von schreibmaschinentastaturen mittels quadratischer zuordnungsprobleme, Zeitschrift für Operations Research 21 (4) (1977) B121–B132. See page 1.

[29] N. W. Brixius, K. M. Anstreicher, The steinberg wiring problem, in: The sharpest cut: the impact of Manfred Padberg and his work, SIAM, 2004, pp. 293–307. See page 1.

[30] A. M. Geoffrion, G. W. Graves, Scheduling parallel production lines with changeover costs: Practical application of a quadratic assignment/lp approach, Operations Research 24 (4) (1976) 595–610. See page 1.

[31] F. Chicano, G. Luque, E. Alba, Elementary landscape decomposition of the quadratic assignment problem, in: Proceedings of the 12th annual conference on Genetic and evolutionary computation, 2010, pp. 1425–1432. See pages 1, 2, 8, 9, 10, 29, 33, and 35.

[32] S. Sahni, T. Gonzalez, P-complete approximation problems, Journal of the ACM (JACM) 23 (3) (1976) 555–565. See page 2.

[33] G. A. E.-N. A. Said, A. M. Mahmoud, E.-S. M. El-Horbaty, A comparative study of meta-heuristic algorithms for solving quadratic assignment problem, arXiv preprint arXiv:1407.4863 (2014). See page 2.

[34] F. Neri, C. Cotta, Memetic algorithms and memetic computing optimization: A literature review, Swarm and Evolutionary Computation 2 (2012) 1–14. See page 2.

[35] N. Krasnogor, J. Smith, A tutorial for competent memetic algorithms: model, taxonomy, and design issues, IEEE transactions on Evolutionary Computation 9 (5) (2005) 474–488. See page 2.

[36] U. Benlic, J.-K. Hao, Memetic search for the quadratic assignment problem, Expert Systems with Applications 42 (1) (2015) 584–595. See pages 2, 13.

[37] A. Misevicius, An improved hybrid genetic algorithm: new results for the quadratic assignment problem, in: International Conference on Innovative Techniques and Applications of Artificial Intelligence, Springer, 2003, pp. 3–16. See pages 2, 13.

[38] F. Glover, Tabu search—part i, ORSA Journal on computing 1 (3) (1989) 190–206. See page 2.

[39] É. Taillard, Robust taboo search for the quadratic assignment problem, Parallel computing 17 (4-5) (1991) 443–455. See pages 2, 10.

[40] F. Chicano, F. Daolio, G. Ochoa, S. Vérel, M. Tomassini, E. Alba, Local optima networks, landscape autocorrelation and heuristic search performance, in: International Conference on Parallel Problem Solving from Nature, Springer, 2012, pp. 337–347. See page 2.

[41] F. Daolio, S. Verel, G. Ochoa, M. Tomassini, Local optima networks of the quadratic assignment problem, in: IEEE Congress on Evolutionary Computation, IEEE, 2010, pp. 1–8. See page 2.

[42] P. Merz, B. Freisleben, Fitness landscape analysis and memetic algorithms for the quadratic assignment problem, IEEE transactions on evolutionary computation 4 (4) (2000) 337–352. See page 2.

[43] M.-H. Tayarani-N, A. Prügel-Bennett, Quadratic assignment problem: a landscape analysis., Evol. Intell. 8 (4) (2015) 165–184. See page 2.

[44] F. Chicano, L. D. Whitley, E. Alba, A methodology to find the elementary landscape decomposition of combinatorial optimization problems, Evolutionary Computation 19 (4) (2011) 597–637. See pages 2, 8.

[45] X. Benavides, J. Ceberio, L. Hernando, On the symmetry of the quadratic assignment problem through elementary landscape decomposition, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2021, pp. 1414–1422. See pages 2, 9, 10, 11, 13, 16, 20, 30, 40, and 44.

[46] J. Ceberio, B. Calvo, A. Mendiburu, J. A. Lozano, Multi-objectivising combinatorial optimisation problems by means of elementary landscape decompositions, Evolutionary computation 27 (2) (2019) 291–311. See pages 2, 11, 13, and 23.

[47] D. Rockmore, P. Kostelec, W. Hordijk, P. F. Stadler, Fast fourier transform for fitness landscapes, Applied and Computational Harmonic Analysis 12 (1) (2002) 57–76. See page 2.

[48] C. M. Reidys, P. F. Stadler, Combinatorial landscapes, SIAM review 44 (1) (2002) 3–54. See page 6.

[49] P. F. Stadler, et al., Towards a theory of landscapes, in: Complex systems and binary networks, Springer, 1995, pp. 78–163. See pages 6, 7.

[50] L. K. Grover, Local search and the local structure of np-complete problems, Operations Research Letters 12 (4) (1992) 235–243. See pages 6, 7.

[51] D. Whitley, A. M. Sutton, A. E. Howe, Understanding elementary landscapes, in: Proceedings of the 10th annual conference on Genetic and evolutionary computation, 2008, pp. 585–592. See page 6.

[52] B. Codenotti, L. Margara, Local properties of some NP-complete problems, International Computer Science Institute, 1992. See page 7.

[53] A. Perez, J. Ceberio, J. A. Lozano, Are the artificially generated instances uniform in terms of difficulty?, in: 2018 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2018, pp. 1–8. See page 8.

[54] J. Skorin-Kapov, Tabu search applied to the quadratic assignment problem, ORSA Journal on computing 2 (1) (1990) 33–45. See page 10.

[55] S. Salhi, Defining tabu list size and aspiration criterion within tabu search methods, Computers & Operations Research 29 (1) (2002) 67–86. See page 10.

[56] M. Voorneveld, Characterization of pareto dominance, Operations Research Letters 31 (1) (2003) 7–11. See page 11.

[57] R. E. Burkard, S. E. Karisch, F. Rendl, Qaplib–a quadratic assignment problem library, Journal of Global optimization 10 (4) (1997) 391–403. See page 13.

[58] Z. Drezner, P. M. Hahn, É. D. Taillard, Recent advances for the quadratic assignment problem with special emphasis on instances that are difficult for meta-heuristic methods, Annals of Operations research 139 (1) (2005) 65–94. See pages 13, 18, 19, 23, and 35.

[59] A. Misevicius, An implementation of the iterated tabu search algorithm for the quadratic assignment problem, OR spectrum 34 (3) (2012) 665–690. See page 13.

[60] A. Benavoli, G. Corani, J. Demšar, M. Zaffalon, Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis, The Journal of Machine Learning Research 18 (1) (2017) 2653–2688. See page 16.

[61] B. Calvo, J. Ceberio, J. A. Lozano, Bayesian inference for algorithm ranking analysis, in: Proceedings of the genetic and evolutionary computation conference companion, 2018, pp. 324–325. See page 16.

[62] U. Dorndorf, A. Drexl, Y. Nikulin, E. Pesch, Flight gate scheduling: State-of-the-art and recent developments, Omega 35 (3) (2007) 326–334. See page 19.

[63] F. Daolio, S. Verel, G. Ochoa, M. Tomassini, Local optima networks of the quadratic assignment problem, in: IEEE Congress on Evolutionary Computation, IEEE, 2010, pp. 1–8. See page 23.

[64] S. L. Thomson, F. Daolio, G. Ochoa, Comparing communities of optima with funnels in combinatorial fitness landscapes, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2017, pp. 377–384. See page 23.

[65] S. Jukna, Linearity of expectation, in: Extremal Combinatorics, Springer, 2011, pp. 255–278. See page 33.

[66] I. Bello, H. Pham, Q. V. Le, M. Norouzi, S. Bengio, Neural combinatorial optimization with reinforcement learning, arXiv preprint arXiv:1611.09940 (2016). See page 44.

[67] A. I. Garmendia, J. Ceberio, A. Mendiburu, Neural combinatorial optimization: a new player in the field, arXiv preprint arXiv:2205.01356 (2022). See page 44.