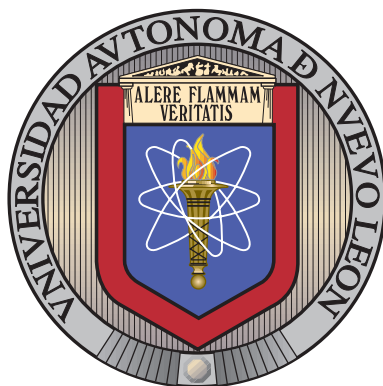


UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

SUBDIRECCIÓN DE ESTUDIOS DE POSGRADO



A HEURISTIC FOR THE α -NEIGHBOR p -CENTER
PROBLEM

POR

ERNESTO ANDRÉS ORTIZ LÓPEZ

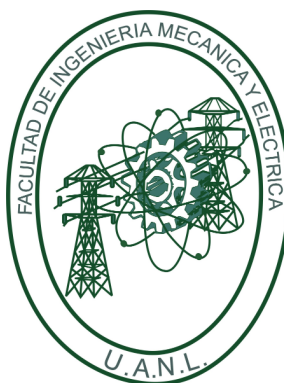
COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE
INGENIERO EN TECNOLOGÍA DE SOFTWARE

MAYO 2023

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

SUBDIRECCIÓN DE ESTUDIOS DE POSGRADO



A HEURISTIC FOR THE α -NEIGHBOR p -CENTER
PROBLEM

POR

ERNESTO ANDRÉS ORTIZ LÓPEZ

COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE
INGENIERO EN TECNOLOGÍA DE SOFTWARE

MAYO 2023



UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA
SUBDIRECCIÓN DE ESTUDIOS DE POSGRADO

Los miembros del Comité de Tesis recomendamos que la Tesis “A Heuristic for the α -Neighbor p -Center Problem”, realizada por el alumno Ernesto Andrés Ortiz López, con número de matrícula 1804204, sea aceptada para su defensa como requisito para obtener el grado de Ingeniero en Tecnología de Software.

El Comité de Tesis

Dr. Roger Z. Ríos Mercado
Director

Dra. María Angelica Salazar Aguilar
Revisora

Dra. Iris Abril Martínez Salazar
Revisora

Vo. Bo.

Dr. Fernando Banda Muñoz
Subdirector Académico

San Nicolás de los Garza, Nuevo León, mayo 2023

To Science ∴

CONTENTS

List of Figures	vii
List of Tables	viii
Agradecimientos	ix
Resumen	x
Abstract	xi
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	2
1.3 Contribution	2
2 Background	4
2.1 Literature review	4
2.2 Problem description	5
2.3 Mathematical model	6
2.4 Illustrative example	6
3 Proposed solution method	9
3.1 GRASP	10

3.2	Constructive heuristic	11
3.2.1	Greedy Center (GC)	12
3.2.2	Greedy Dispersion (GD)	13
3.2.3	Randomized Greedy Dispersion (RGD)	14
3.3	Local search	15
3.3.1	Fast Algorithm for the Greedy Interchange (FAGI)	16
3.3.2	Fast Vertex Substitution (FVS)	20
3.3.3	Alpha Fast Vertex Substitution (A-FVS)	22
4	Computational results	32
4.1	Performance of A-FVS	33
4.2	Performance of GRASP	38
4.2.1	Calibrating i_{\max}	38
4.2.2	Calibrating β	47
4.2.3	Final results	50
5	Conclusions	52
5.1	Future work	53
A	Detailed GRASP final results	55
	Bibliography	67

LIST OF FIGURES

2.1	Comparison between ANPCP and PCP solutions.	7
2.1	Comparison between ANPCP and PCP solutions.	8
3.1	After removing $\phi_{\alpha-1}(u)$, a facility closer to u than its current center, the new center becomes $\phi_{\alpha+1}(u)$	18
3.2	The two options for the new center when u is attracted to f_i (Equation (3.2)).	19
3.3	α -neighbors of u (A_u).	23
3.4	If an α -neighbor is f_r and u is attracted to f_i , its center remains unchanged (no penalty). In both of the above cases, ϕ'_α remains as the same facility as ϕ_α , even when the attraction of u to f_i varies. . .	25
3.5	If an α -neighbor is f_r and u is not attracted to f_i , its new center will be farther (penalty). In both of the above cases, ϕ'_α is farther than ϕ_α , regardless of how far f_i is.	26
4.1	Visualization of $x(S)$ from Table 4.6.	40
4.2	Visualizations of $x(S)$ from Table 4.7.	42
4.3	Distribution of nodes in sample instances.	45
4.4	Visualizations of $x(S)$ from Table 4.8.	46

LIST OF TABLES

3.1	Example of an M memory matrix where $\alpha = 2$	31
4.1	GD solution improved with FI strategy.	34
4.2	GD solution improved with BI strategy.	35
4.3	Random solution improved with FI strategy.	36
4.4	Random solution improved with BI strategy.	36
4.5	Average results of the local search comparison.	37
4.6	Improvements in <code>r11323_882_441_0</code> , $p = 20$	40
4.7	Improvements in <code>rat783_522_261_0</code> , $p = 20$	41
4.8	Improvements in <code>anpcp_882_441_0</code> , $p = 44$	43
4.9	Average results of the experiment, grouped by instance and parameters.	48
4.10	Means of Table 4.9.	48
4.11	Summarized results of GRASP experiments.	51
A.1	Results for <code>pr439_293_146</code>	56
A.2	Results for <code>rat575_384_191</code>	58
A.3	Results for <code>rat783_522_261</code>	60
A.4	Results for <code>dsj1000_667_333</code>	61
A.5	Results for <code>r11323_882_441</code>	63
A.6	Results for <code>r11889_1260_629</code>	65

AGRADECIMIENTOS

Me siento altamente agradecido hacia mi principal asesor y director de tesis, el Dr. Roger Z. Ríos Mercado, por la experiencia compartida de tan alto valor, por su paciencia durante el proceso, y por el impulso que me ha dado a presentar aquí un trabajo científico de calidad. Agradezco a la Dra. Diana Lucía Huerta Muñoz quien, a pesar de ser externa al comité, estuvo dedicando su tiempo en apoyarme en cada una de las revisiones de la tesis, desde el inicio de su realización. Así mismo, doy las gracias a las revisoras Dra. María Angélica Salazar Aguilar y Dra. Iris Abril Martínez Salazar, por aportar su tiempo para mejorar el presente trabajo.

Agradezco a la Facultad de Ingeniería Mecánica y Eléctrica (FIME) de la Universidad Autónoma de Nuevo León (UANL) por haber brindado recursos y mentoría para completar la ingeniería y la presente tesis, y al Consejo Nacional de Ciencia y Tecnología (CONACyT) por su apoyo económico con la beca de Ayudante de Investigador Nivel III del Sistema Nacional de Investigadores (SNI).

Es menester reconocer a mis padres, Ernesto Antonio Ortiz Rojas y Anabel López Leal, por haber hecho posible la realización de mi carrera profesional y haber estado presentes hasta la culminación de mis estudios.

Por último, no olvido que todos y cada uno de los autores que cito en el presente, a quienes admiro, me ayudaron indirectamente a su realización, mediante los excelentes trabajos de investigación que publicaron. Si esta tesis tiene algún valor relevante dentro de la literatura, es gracias a las aportaciones científicas de estos investigadores.

RESUMEN

Ernesto Andrés Ortiz López.

Candidato para obtener el grado de Ingeniero en Tecnología de Software.

Universidad Autónoma de Nuevo León.

Facultad de Ingeniería Mecánica y Eléctrica.

Título del estudio: A HEURISTIC FOR THE α -NEIGHBOR p -CENTER PROBLEM.

Número de páginas: 68.

OBJETIVOS Y MÉTODO DE ESTUDIO: El problema del p -centro de α -vecinos, del inglés *α -neighbor p -center problem* (ANPCP), es un problema de localización cuyo objetivo es seleccionar p centros que minimicen la distancia máxima entre cualquier usuario y su α -ésimo centro más cercano. El propósito de esta tesis es diseñar heurísticas para resolver el ANPCP eficientemente.

CONTRIBUCIONES Y CONCLUSIONES: En esta tesis se propone una adaptación de la búsqueda local de sustitución de vértices para el ANPCP, llamada *Alpha Fast Vertex Substitution* (A-FVS), la cual es significativamente más rápida debido al uso de estructuras de datos que almacenan las asignaciones entre usuarios y centros después de un intercambio de instalaciones, permitiendo reusar computaciones costosas sin tener que recalcularlas. Esta forma inteligente de explotar la estructura de la función objetivo del problema se traduce en mejoras significativas en tiempo de cómputo para todos los casos. Integramos el A-FVS en un Procedimiento de Búsqueda Adaptativa Ávida Aleatorizada (GRASP, siglas del inglés *Greedy Randomized Adaptive Search Procedure*) y se muestra cómo ésta supera al A-FVS.

Firma del director: _____

Dr. Roger Z. Ríos Mercado

ABSTRACT

Ernesto Andrés Ortiz López.

As a candidate to obtain a degree in Software Engineering.

Universidad Autónoma de Nuevo León.

Facultad de Ingeniería Mecánica y Eléctrica.

Title of the study: A HEURISTIC FOR THE α -NEIGHBOR p -CENTER PROBLEM.

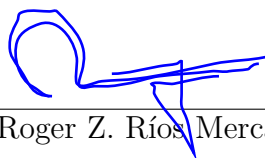
Number of pages: 68.

OBJECTIVES AND STUDY METHODS: The α -neighbor p -center problem (ANPCP) is a location problem in which the goal is to select p centers such that the maximum distance of a user to its α th closest open facility is minimized. The main goal of this thesis is to design efficient heuristic and metaheuristic procedures to solve the ANPCP.

CONTRIBUTIONS AND CONCLUSIONS: We present an enhanced implementation of the vertex substitution local search procedure for the ANPCP. The heuristic, called Alpha Fast Vertex Substitution (A-FVS), is significantly faster due to the use of data structures that store how the assignments between users and centers behave after a swap of facilities, allowing to reuse these expensive computations by accessing them instead of recalculating them. This intelligent way of exploiting the structure of the objective function led to significant computational speed-up times for all cases. The A-FVS is integrated as a local search phase into a Greedy Randomized Adaptive Search Procedure (GRASP). Empirical evidence shows that GRASP outperforms A-FVS.

Director signature: _____

Dr. Roger Z. Ríos Mercado



CHAPTER 1

INTRODUCTION

The α -neighbor p -center problem (ANPCP) is a generalization of the classical p -center problem (PCP), in which the goal is to select p centers such that the maximum distance of a customer or demand point to its α th closest open facility (its center) is minimized (Krumke, 1995). This problem arises in the modeling of the placement of emergency facilities, such as fire stations or hospitals, where the aim is to have a minimum guaranteed response time between a customer or demand point and its center by considering a notion of fault tolerance, i.e., providing backup centers in case one of them fails to respond to an emergency (Khuller et al., 2000). This ensures that even if up to $\alpha - 1$ facilities fail, every customer has an open facility close to it.

1.1 MOTIVATION

As explained in Chapter 2, the ANPCP has not received much attention in the literature, despite its closeness to the PCP. Furthermore, there was not any approach that considered a heuristic algorithm or a metaheuristic framework, until very recently. This led the existing literature to include some optimal and approximation algorithms. However, as technology grows in complexity and data grows in size by orders of magnitude, it becomes more and more common to find large instances in

classical problems. For example, a graph of thousands of nodes that requires some solution within a few minutes, even if it is not the best possible solution: time is essential. A problem such as the ANPCP, which can be used for delicate models such as placing vulnerable emergency facilities, needs to get up to date with the newest requests of the fast-evolving technology. As a consequence, it becomes essential to have appropriate algorithms to solve large instances of this problem quickly. Our hypothesis is that a metaheuristic designed to solve the ANPCP will be able to yield solutions of high quality in a relatively short amount of time. To achieve this, we will follow and show in detail the process of implementing a metaheuristic, from designing its individual components to adapting the quirks of the problem, providing empirical evidence of the efficiency of both the complete algorithm and its parts.

1.2 OBJECTIVES

- To design efficient heuristic algorithms for the ANPCP.
- To implement a robust metaheuristic framework for the ANPCP.
- To demonstrate the effectiveness of solving such a problem with a metaheuristic.
- To compare the results of solving the ANPCP using enhanced and special heuristic algorithms versus solving it with generic ones.

1.3 CONTRIBUTION

In this work, a Greedy Randomized Adaptive Search Procedure (GRASP) is proposed to solve the ANPCP. The key component of this metaheuristic is a fast local search algorithm, which provides reasonable quality solutions in less time than a greedy interchange due to the use of data structures that store how the assignments

between users and centers behave after a swap of facilities, allowing to reuse these expensive computations by accessing them instead of recalculating them. This algorithm is referred to as the *alpha fast vertex substitution* (A-FVS) method, which is an adaptation from an algorithm with the same name designed for the PCP, as explained in Chapter 3. In Chapter 4, we experimentally show that the A-FVS is significantly faster than a generic greedy interchange for all the tested cases (750 times faster for the largest instance), as well as the results of using our proposed GRASP to solve instances from the well-known TSPLIB library.

CHAPTER 2

BACKGROUND

There are some real-world situations in which a set of facilities, such as hospitals, malls, or emergency centers, requires to be located for servicing a set of demand points or users, such as patients, customers, or people at risk. This types of problems are commonly known as location problems. In Operations Research, location problems are discrete optimization problems that consist of determining the best location for one or several centers or facilities to serve a set of demand points (Laporte et al., 2019). The solution to a location problem is the set of those centers. A location problem is characterized by a specific objective function, which defines how a solution is better than another.

2.1 LITERATURE REVIEW

The first known mention of a location problem with unreliable facilities comes from Drezner (1987), where two continuous problems are solved by heuristic methods. The unreliable p -median problem (UPMP) is defined by introducing the probability that a facility becomes inactive, and the (p, q) -center problem (PQCP) is defined when p facilities need to be located but up to q of them may become unavailable at the same time. However, the ANPCP is formally introduced in Krumke (1995), where the problem is studied as a generalization of the PCP. The author showed

that the problem is \mathcal{NP} -hard and provided a 4-approximation algorithm for $\alpha \geq 2$. A better approximation algorithm was later proposed in Khuller et al. (2000), where an approximation factor of 2 was obtained for $\alpha < 4$, and a factor of 3 for any α .

Two optimal algorithms and one relaxation algorithm were proposed in Chen and Chen (2013) for both the continuous and the discrete versions of the ANPCP, although only the continuous version was tested. Here, the authors treated the problem as one of locating p circles so that each demand point is covered α times, where the centers of the p circles are the locations of the p service facilities, and the goal is to minimize the radius of the maximal circle. The problem is referred to as the r -all-neighbor p -center problem (RANPCP) in Medal et al. (2014), in which all nodes are demand nodes, rather than defining a demand node as a node that does not have a facility located on it. The authors provide an empirical bi-objective analysis of the problem.

Recently, a metaheuristic framework that combines a GRASP with strategic oscillation was presented in Sánchez-Oro et al. (2022) to solve the discrete version of the ANPCP, where the local search is a greedy interchange algorithm that only swaps the critical center, i.e., one of the centers whose distance to their users is the maximum, to break the objective function value. The heuristic was enhanced by Tabu Search, and the overall metaheuristic yielded better results than those of Chen and Chen (2013). One observation is that the authors consider the set of potential facility locations and the set of users as the same set. In our work, we treat both sets separately, so in a sense, our approach is more general as it can be applied to their version but not the other way around.

2.2 PROBLEM DESCRIPTION

The α -neighbor p -center problem (ANPCP) is defined as follows: given a set F of m facilities, a set U of n users or customers, a distance function $d(u, f)$ between any

user u and any facility f , an integer $p < m$, and an integer $\alpha \leq p$, find a subset $S \subseteq F$ of size p that minimizes the maximum distance x between any user and its α th closest open facility. The problem is known to be \mathcal{NP} -hard (Krumke, 1995). The ANPCP is a generalization of the p -center problem (PCP). Note that setting $\alpha = 1$ corresponds to the PCP, making it identical to the 1NPCP.

2.3 MATHEMATICAL MODEL

Let A be any subset of α facilities from a given solution S . The distance $\mathcal{D}^{(\alpha)}$ between any user $u \in U$ and the solution set S is defined as:

$$\mathcal{D}^{(\alpha)}(u, S) = \min_{\substack{A \subseteq S \\ |A| = \alpha}} \{ \max_{f \in A} d(u, f) \} \quad (2.1)$$

The objective function for the ANPCP is then defined as:

$$x^{(\alpha)}(S) = \max_{u \in U} \mathcal{D}^{(\alpha)}(u, S) \quad (2.2)$$

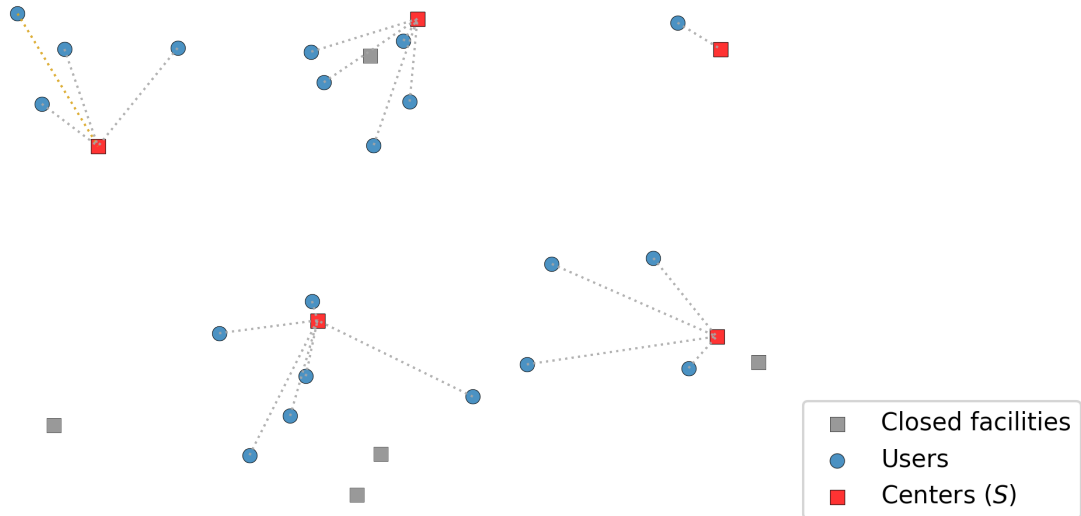
Let Π^p be the collection of subsets of F of cardinality p , that is, $\Pi^p = \{S \subseteq F \mid |S| = p\}$. Hence, the combinatorial optimization problem can be cast as:

$$\min_{S \in \Pi^p} x^{(\alpha)}(S) \quad (2.3)$$

2.4 ILLUSTRATIVE EXAMPLE

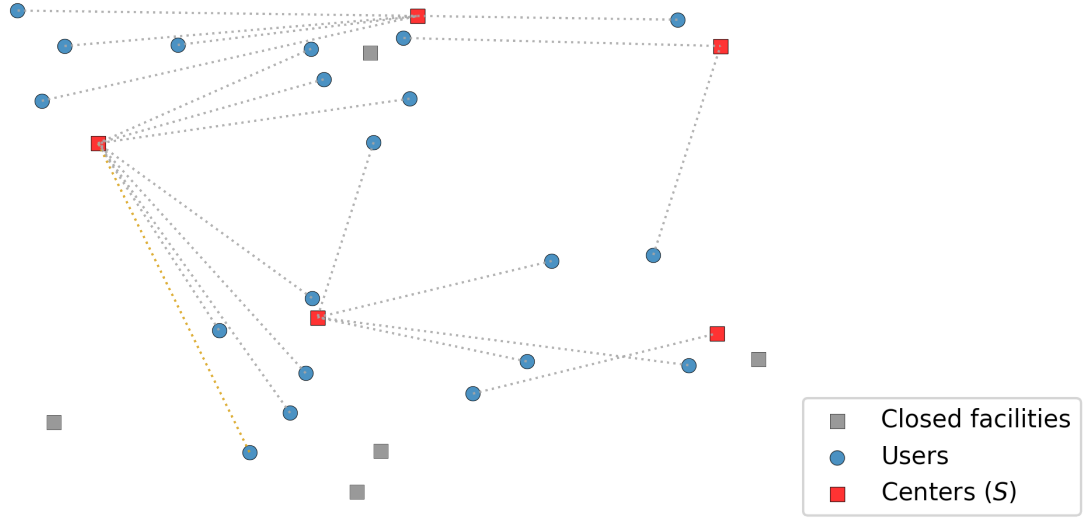
Figure 2.1a shows a visualization of the optimal solution S_1 for a random instance of the PCP, whose value is 268. The dotted lines represent the corresponding as-

signments, and the orange line is the largest distance between any user and its center, that is, the objective function value $x^{(\alpha)}(S)$. One way to visually differentiate between PCP and ANPCP solutions is that in the former, the assignment lines never cross each other: the centers and their assigned users look like clusters. In Figure 2.1b, note that if this same solution S_1 is used for the 2NPCP, i.e., setting $\alpha = 2$ and reassigning the users to their second nearest open facility from S_1 without modifying the solution, the objective function results in a greater value (607). On the other hand, if the 2NPCP is solved instead, i.e., running an algorithm to modify S_1 and obtain a better quality solution, the objective function value of this new solution S_2 is smaller (483), as shown in Figure 2.1c, which is actually the optimal solution for this particular example. This visual comparison justifies the relevance of solving the ANPCP rather than solving the PCP and just using that solution for different values of α , since that would yield poor quality solutions for the ANPCP. In general, as α increases, the objective function value is likely to decrease: $x^{(1)}(S) \leq x^{(2)}(S) \leq \dots \leq x^{(\alpha)}(S) \leq \dots \leq x^{(p)}(S)$ for any $S \subset F$ (Krumke, 1995). In the three figures, the same instance is used, with parameters $n = 20$ (users), $m = 10$ (total facilities), and $p = 5$ (size of solution S).

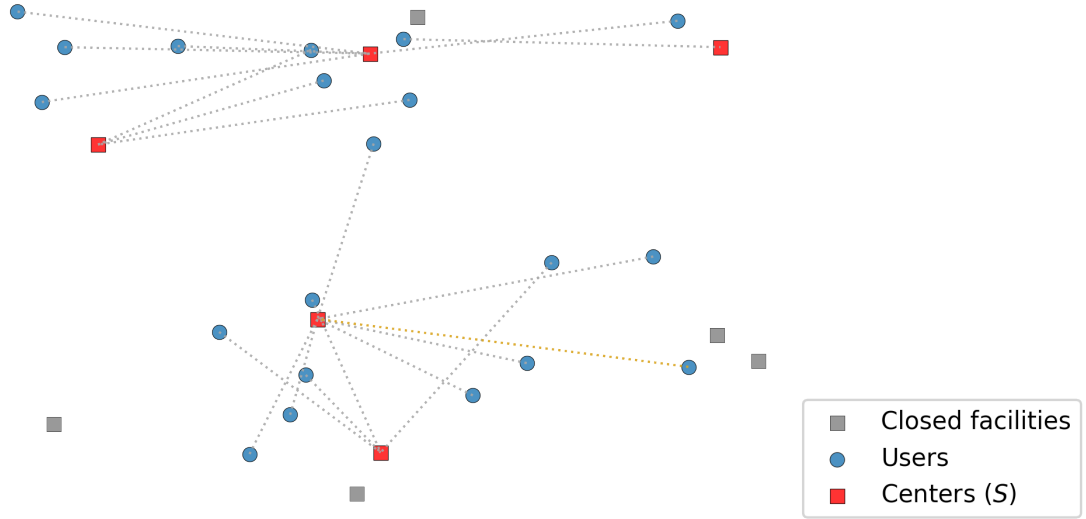


(a) Optimal solution for the 1NPCP (PCP), $x^{(1)}(S_1) = 268$.

FIGURE 2.1: Comparison between ANPCP and PCP solutions.



(b) Same solution from 2.1a applied to the 2NPCP, $x^{(2)}(S_1) = 607$.



(c) Optimal solution for the 2NPCP, $x^{(2)}(S_2) = 483$.

FIGURE 2.1: Comparison between ANPCP and PCP solutions.

CHAPTER 3

PROPOSED SOLUTION METHOD

The Greedy Randomized Adaptive Search Procedure (GRASP) (Feo and Resende, 1995) is a procedure that combines element of greedy construction, randomization, and local search to generate good quality solutions. It is a multi-start metaheuristic that has been successfully employed for many combinatorial optimization problems, including location problems (Delmaire et al., 1999; Klincewicz, 1992), and more specifically, center-based location problems (Quevedo-Orozco and Ríos-Mercado, 2015; López-Sánchez et al., 2019).

We propose a GRASP metaheuristic to solve the α -neighbor p -center problem (ANPCP), as introduced in Section 3.1. Each iteration of the metaheuristic is composed of two main phases, a construction and an improvement, which are discussed in Section 3.2 and Section 3.3, respectively. The first phase evaluates the objective function of the p -dispersion problem (PDP) to construct a feasible solution for the ANPCP. This algorithm, referred to in this paper as Greedy Dispersion (GD), was introduced by Erkut et al. (1994) and we explain it in Section 3.2.2. We also adapted it to use as a value-based restricted candidate list (RCL) within GRASP. We have called this variant, the Randomized Greedy Dispersion (RGD), and it is explained in Section 3.2.3. The constructed solution is then improved, in the second phase, by a local search heuristic whose move is a vertex substitution or interchange. We designed this algorithm specifically for the ANPCP by adapting the concept of

fast interchange, introduced in Whitaker (1983) and later adapted for the p -center problem (PCP) in Mladenović et al. (2003). This algorithm has not been studied for the ANPCP before, to the best of our knowledge, and we explain our adaptation in detail in Section 3.3.3. We named this local search the Alpha Fast Vertex Substitution (A-FVS) and it is significantly faster due to the use of data structures that store how the assignments between users and centers behave after a swap of facilities, allowing to reuse these expensive computations by accessing them instead of recalculating them.

The proposed GRASP combines the RGD constructive heuristic and the A-FVS local search heuristic, and accepts as input two parameters: a number of consecutive iterations without improvement before stopping the procedure, and a threshold value β to adjust the randomness of the GRASP restricted candidate list (RCL).

3.1 GRASP

A Greedy Randomized Adaptive Search Procedure (GRASP) is a metaheuristic for finding approximate (i.e. good sub-optimal, but not necessarily optimal) solutions to combinatorial optimization problems. It is based on the premise that diverse, good-quality starting solutions play an important role in the success of local search methods (Resende and González Velarde, 2003).

A GRASP is a multi-start, iterative randomized sampling technique in which each iteration provides a solution to the problem at hand. The incumbent solution over all GRASP iterations is kept as the final result. There are two phases within each GRASP iteration: the first intelligently constructs an initial solution via an adaptive randomized greedy function, and the second applies a local search procedure to the constructed solution attempting to find an improvement (Feo and Resende, 1995).

In this paper, we designed a GRASP framework to solve the ANPCP. The

construction phase is discussed in Section 3.2 and the heuristic used, called the Randomized Greedy Dispersion (RGD), is explained in Section 3.2.3. The local search phase is discussed in Section 3.3 and the proposed heuristic, the Alpha Fast Vertex Substitution (A-FVS), is explained in Section 3.3.3. The pseudocode of the GRASP is described in Algorithm 1.

Algorithm 1 GRASP($U, F, p, \alpha, D, \beta, i_{\max}$)

Input: U := set of users; F := set of facilities; p := number of open facilities; α := closeness position of centers; D := distance matrix $U \times F$; β := RCL threshold parameter; i_{\max} := maximum number of iterations without improvement.

Output: S^* := solution set of p centers

```

1:  $S^* \leftarrow \emptyset$ 
2:  $x^* \leftarrow \infty$ 
3:  $i \leftarrow 0$ 
4: while  $i < i_{\max}$  do
5:    $S \leftarrow \text{RGD}(F, \bar{D}, p, \beta)$ 
6:    $S' \leftarrow \text{A-FVS}(U, F, S, p, \alpha, D)$ 
7:   if  $x(S') < x^*$  then
8:      $x^* \leftarrow x(S')$ 
9:      $S^* \leftarrow S'$ 
10:     $i \leftarrow 0$ 
11:   else
12:      $i \leftarrow i + 1$ 
13:   end if
14: end while
15: return  $S^*$ 

```

3.2 CONSTRUCTIVE HEURISTIC

A solution to the ANPCP consists of a set of p selected facilities out of the m original candidates. A constructive heuristic obtains an initial feasible solution by adding one facility to the partial solution at each iteration until reaching a size of p open facilities.

We designed two constructive heuristics. One evaluates the objective function of the ANPCP, called the Greedy Center (GC) heuristic, and it is explained in Section 3.2.1. However, this objective function is rather computationally expensive,

so as an alternative, we implemented an algorithm that was originally designed for another problem, the p -dispersion problem (PDP). In this process, we managed to reduce the time complexity of the original algorithm by introducing an array that stores, for each facility not belonging to the the solution, its distance to the closest center (a facility in the solution). This allows accessing any of the distances in constant time, rather than iterating on the set of closed facilities over and over again. We have named this implementation the Greedy Dispersion (GD) heuristic and it is explained in Section 3.2.2. Furthermore, to adapt the GD as the construction phase of the GRASP metaheuristic, we modified it to use a value-based restricted candidate list (RCL). This variant is named the Randomized Greedy Dispersion (RGD) heuristic, which is explained in Section 3.2.3.

3.2.1 GREEDY CENTER (GC)

We first designed a greedy heuristic that evaluates the objective function of the ANPCP (2.2) when deciding which facility to add to the solution at each iteration. The pseudocode of the Greedy Center heuristic is shown in Algorithm 2.

Algorithm 2 $GC(U, F, D, p, \alpha)$

Input: U, F, D, p, α

Output: $S :=$ solution set of p centers

- 1: $S \leftarrow$ select α random facilities from F
 - 2: **while** $|S| < p$ **do**
 - 3: $f_i^* \leftarrow \arg \min_{f_i \in F \setminus S} x^{(\alpha)}(S \cup \{f_i\})$
 - 4: $S \leftarrow S \cup \{f_i^*\}$
 - 5: **end while**
 - 6: **return** S
-

The algorithm starts by initializing the solution set S with α random facilities. This is because, to evaluate the objective function of the ANPCP, S must have a size of at least α . Then, it enters to an iterative phase to select the remaining $p - \alpha$ facilities. At each iteration, the objective function is evaluated for each candidate facility in $F \setminus S$, denoted as f_i , and the one that minimizes the objective function

value is inserted to S . The objective function of the ANPCP is computationally expensive. Therefore, we proposed an enhanced constructive heuristic, which was originally designed for another problem.

3.2.2 GREEDY DISPERSION (GD)

The p -dispersion problem (PDP) consists of selecting p facilities from a given set of m candidates in such a way that the minimum distance between selected facilities is maximized (Erkut et al., 1994). This problem is of interest because the dispersion pattern of the chosen facilities is comparable to that of the ANPCP. It was decided to implement a constructive heuristic based on the objective function of the PDP, and see if it can yield starting solutions of good quality for the ANPCP, even if it ignores its actual objective function. This algorithm evaluates the set F as the set of facilities for the PDP. We identify this algorithm as the Greedy Dispersion (GD), and the idea is based on the algorithm originally proposed by Erkut et al. (1994) for the PDP. Its pseudocode can be seen in Algorithm 3.

Algorithm 3 $\text{GD}(F, \bar{D}, p)$

Input: $F :=$ set of facilities; $\bar{D} := |F| \times |F|$ distance matrix; $p :=$ number of centers

Output: $S :=$ solution set of p centers

```

1:  $f_l \leftarrow$  random facility from  $F$ 
2:  $S \leftarrow \{f_l\}$ 
3: while  $|S| < p$  do
4:    $\delta(f_i) \leftarrow \min\{\delta(f_i), \bar{d}(f_i, f_l)\}, \forall f_i \in F \setminus S$ 
5:    $f_i^* \leftarrow \arg \max_{f_i \in F \setminus S} \delta(f_i)$ 
6:    $S \leftarrow S \cup \{f_i^*\}$ 
7:    $f_l \leftarrow f_i^*$ 
8: end while
9: return  $S$ 
```

The procedure receives F , d_{ij} , and p as input, and it returns a solution set S . S is initialized with a random facility selected from F (line 2). Then, it iterates until exactly p facilities have been selected. For the following $p - 1$ iterations of

this loop, a facility from $F \setminus S$, denoted as f_i , must be inserted to S . The facility whose distance to the current S is the greatest among all the candidates in $F \setminus S$ is denoted as f_i^* , which maximizes the objective function of the PDP and, as a result, will be inserted to S . Because S is a set, the distance between a candidate facility f_i and S is defined as the minimum distance from f_i to any of the centers, i.e., $\delta(f_i, S) = \min_{f_s \in S} \{d(f_i, f_s)\}$. This function performs $O(p)$ operations in the worst case just by itself, and the worst-case time complexity of the complete algorithm would be $O(mp^2)$. As a contribution to this work, we reduce this overall complexity to $O(mp)$ by making use of an array that stores $\delta(f_i, S)$ for every f_i . This array is denoted as $\delta(\cdot)$. Because S gets updated during the algorithm, the array $\delta(\cdot)$ is updated at the same time as f_i^* is selected by comparing its current value with the last inserted facility, denoted as f_l , and choosing the minimum of both distances (line 4).

The GD heuristic is better than GC in terms of computational time complexity, and thus we use it as the construction phase of the metaheuristic proposed in this work, which is explained next.

3.2.3 RANDOMIZED GREEDY DISPERSION (RGD)

We adapted the GD algorithm from the previous section for the construction phase of the GRASP. We have called this adaptation the Randomized Greedy Dispersion (RGD) (see Algorithm 4) and was modified to accept a parameter to determine the size of the restricted candidate list (RCL). The RCL contains a set of candidate facilities with high greedy function values or costs. The next facility to be inserted into the solution is selected at random from this list (Resende and González Velarde, 2003). We identify this parameter with β and we use a value-based RCL. Let β be a real number such that $0 \leq \beta \leq 1$, and let δ_{\min} and δ_{\max} denote, respectively, the minimum and the maximum costs for the candidate facilities. Then, the RCL consists of the facilities whose cost $\delta(f_i)$ is such that $\delta(f_i) \geq t$, where t is the

threshold that restricts the facilities that can enter the candidate list, formulated as $t = \delta_{\max} - \beta(\delta_{\max} - \delta_{\min})$. Note that setting $\beta = 0$ leads to $t = \delta_{\max}$, making this selection scheme almost purely greedy (the only exception is the first random facility in the solution), whereas $\beta = 1$ makes $t = \delta_{\min}$, resulting in a totally random selection. Hence, it becomes important to find an appropriate value for β since it balances the grade of intensification and diversification of the constructed solution.

The RGD uses the same data structure $\delta(\cdot)$ and the same evaluation function that was described in Section 3.2.2 the GD. It is also very similar to the constructive procedure used in Sánchez-Oro et al. (2022), where it is implemented without the array $\delta(\cdot)$.

Algorithm 4 RGD(F, \bar{D}, p, β)

Input: F := set of facilities; $\bar{D} := |F| \times |F|$ distance matrix; p := number of centers; β := RCL threshold parameter.

Output: S := solution set of p centers

```

1:  $f_l \leftarrow$  random facility from  $F$ 
2:  $S \leftarrow \{f_l\}$ 
3: while  $|S| < p$  do
4:    $\delta(f_i) \leftarrow \min\{\delta(f_i), \bar{d}(f_i, f_l)\}, \forall f_i \in F \setminus S$ 
5:    $\delta_{\min} \leftarrow \min_{f_i \in F \setminus S} \delta(f_i)$ 
6:    $\delta_{\max} \leftarrow \max_{f_i \in F \setminus S} \delta(f_i)$ 
7:    $t \leftarrow \delta_{\max} - \beta(\delta_{\max} - \delta_{\min})$ 
8:    $L \leftarrow \{f_i \in F \setminus S : \delta(f_i) \geq t\}$ 
9:    $f_l \leftarrow \text{random}(L)$ 
10:   $S \leftarrow S \cup \{f_l\}$ 
11: end while
12: return  $S$ 

```

3.3 LOCAL SEARCH

A solution to the ANPCP can be improved by applying a local search procedure, i.e., modifying one or more components of the solution based on some rule, looking for a better objective function value among the space of candidate solutions or neighborhoods, until a local optimum is reached. A well-known neighborhood used

in local search for this type of location problems is based in the vertex swap move, which consists of replacing one of the facilities in the solution, denoted as f_r , by a closed facility, denoted as f_i . This move is repeated until no more improvements are found (local optimum reached).

A straightforward local search algorithm was first implemented for the ANPCP, identified as Naive Interchange (NI). This algorithm simply iterates over the set of open facilities and, for each of them, iterates the set of closed facilities. At each iteration, it temporarily removes the current open facility, adds the closed facility, and recalculates the objective function value. This procedure does $O(p^2mn)$ operations, resulting in inefficient performance, based on preliminary experiments. For this reason, we propose a faster vertex substitution local search algorithm to obtain solutions faster. We first introduce the concept of the *fast interchange* and its background in Section 3.3.1, then we analyze its application for the PCP in Section 3.3.2, and finally, we explain in detail how we adapted it and produced a fast local search heuristic that can solve the ANPCP, which we call the Alpha Fast Vertex Substitution (A-FVS) heuristic, in Section 3.3.3. To the best of our knowledge, this local search algorithm has not been proposed for this problem before.

3.3.1 FAST ALGORITHM FOR THE GREEDY INTERCHANGE (FAGI)

The *fast algorithm for the greedy interchange* (FAGI) was first proposed for the p -median problem (PMP) by Whitaker (1983). The key aspect of this implementation is its ability to find the most profitable candidate facility for removal (f_r), given a certain candidate facility for insertion (f_i), while partially evaluating the objective function. What makes this procedure fast is the observation that the profit that would result from applying a swap can be decomposed into two components. The first one, called *gain*, identifies the users who would benefit from the insertion of f_i

into the solution because each user is closer to it than its current closest facility. The second component, *netloss*, accounts for all other users that would not benefit from the insertion of f_i . If f_r is the current closest facility of a user, then it would have to be reassigned either to its second closest one or to f_i , whichever is the closest. In both cases, the cost of serving that user will either increase or remain constant. These two components are useful as they work as auxiliary data structures that make the algorithm fast, which has a worst-case time complexity of $O(mn)$.

The FAGI was implemented and tested for the ANPCP. Even though this algorithm uses the objective function of the PMP, no solutions were obtained successfully (the algorithm was getting stuck when $\alpha \geq 2$, even with small instances). After analyzing how the algorithm was taking decisions and updating the auxiliary data structures, we observed some key differences between the PMP and the ANPCP which were later crucial to propose an improved adaptation that can obtain solutions for any value of α (where $\alpha < p$).

In the PMP, as well as in the PCP, users are assigned to their closest open facility. Although their objective functions are evaluated differently, in both problems the centers have the same definition. Nonetheless, the definition of a center changes for the ANPCP, because users are assigned not to their closest open facility but their α th closest open facility, meaning that there are $\alpha - 1$ open facilities that are closer to any user u than its center. More formally, let $\phi_k(u)$ be the k th closest open facility of any user u ,

$$\phi_\alpha(u) \text{ is determined by all } \{\phi_k(u) \mid 1 \leq k < \alpha\} \quad (3.1)$$

If for u , one of the open facilities closer than its current center is considered to be f_r , then this center would no longer be the α th closest open facility. Hence, the new center of u denoted as $\phi'_\alpha(u)$ would, by definition, be moved to the next farther open facility, which is the current $\phi_{\alpha+1}(u)$. A visual representation is shown in Figure 3.1, where f_r is the gray facility (the one that will be removed from the

solution), the black straight line is the assignment of u to its center, and the red line and outline indicate which node will be the new center of u after removing f_r .

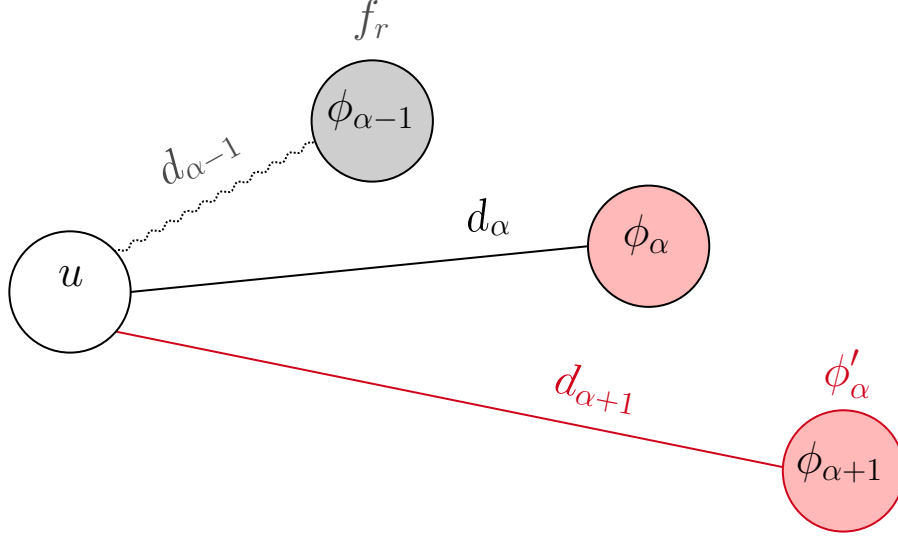


FIGURE 3.1: After removing $\phi_{\alpha-1}(u)$, a facility closer to u than its current center, the new center becomes $\phi_{\alpha+1}(u)$.

Similarly, the fact that u is attracted to f_i does not necessarily mean that f_i is going to be its new center: it depends on how close f_i is from u . This is represented visually in Figure 3.2. The light red nodes are open facilities, and the dotted lines represent the distances from u to the open facilities that are closer than its center. Mathematically, let $d_k(u) = d(u, \phi_k(u))$ and $d_i(u) = d(u, f_i)$. Then, $\phi'_\alpha(u)$ is determined by Equation (3.2).

$$\phi'_\alpha(u) \leftarrow \begin{cases} f_i, & \text{if } d_{\alpha-1}(u) \leq d_i(u) < d_\alpha(u) \\ \phi_{\alpha-1}(u), & \text{if } d_i(u) < d_{\alpha-1}(u) \end{cases} \quad (3.2)$$

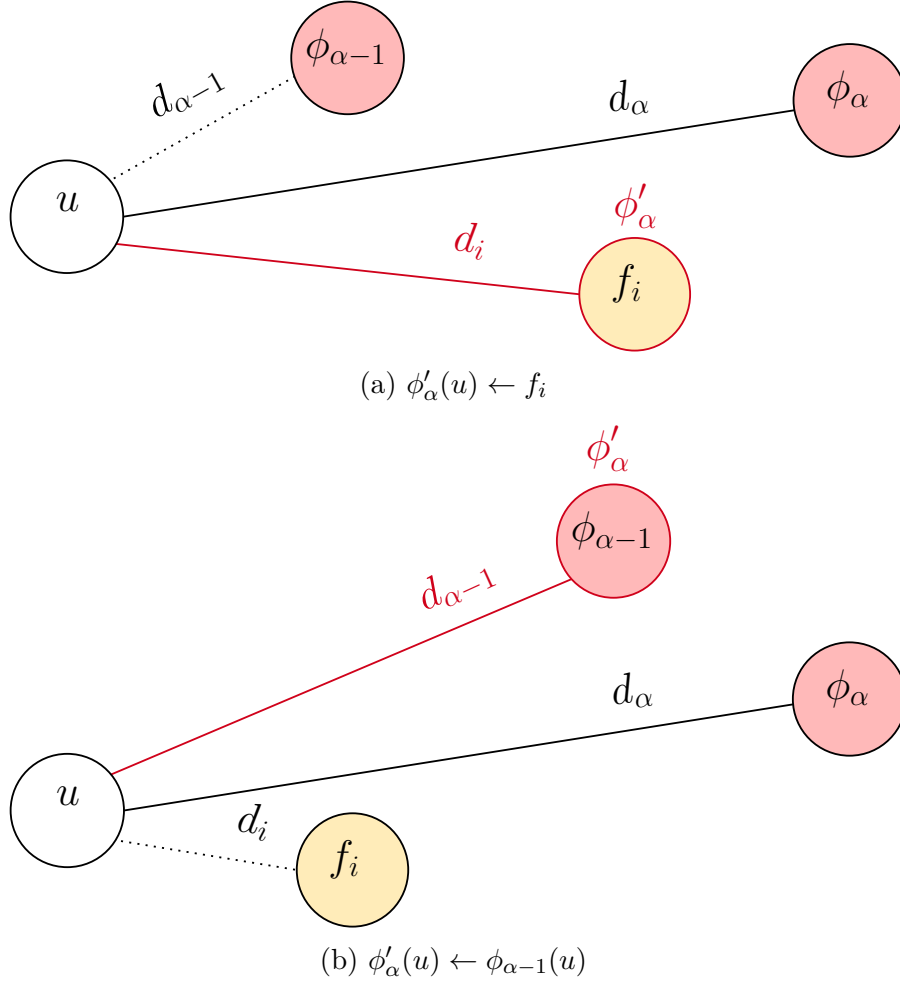


FIGURE 3.2: The two options for the new center when u is attracted to f_i (Equation (3.2)).

These observations resulted in modifying FAGI to update the data structures *gain* and *netloss* accordingly. However, after applying such modifications, the algorithm did not produce any solutions either. As a consequence, we decided to modify the greedy evaluation step by evaluating the ANPCP's objective function instead of the PMP's, focusing on how to calculate the objective function of our problem faster.

3.3.2 FAST VERTEX SUBSTITUTION (FVS)

We found that FAGI was later adapted to solve the PCP, named the *fast vertex substitution* (FVS), proposed by Mladenović et al. (2003). This algorithm also has a worst-case time complexity of $O(mn)$. Now, as mentioned earlier, the ANPCP is a variant of the PCP in which the difference is how the centers are defined. However, the observation that both problems aim to minimize a maximum allowed us to use FVS as a better basis algorithm to adapt for the ANPCP. The key to this adaptation is the use of three data structures that accelerate the search of f_r .

In the implementation for the PCP, the new objective function value or radius x' , that would result after the interchange, is kept only for users who are attracted by f_i , i.e., f_i is closer to them than their center. Besides these new assignments to f_i , a new maximum distance x^* could be found among existing assignments. To determine which of these existing connections would remain after a swap, it becomes necessary to store how the radiuses of old centers change when they are considered to be removed or not. The following conditions apply for users who are not attracted by f_i :

1. If a center will be removed (f_r), then its users must be reallocated. For each user u , its new center would be either its second-closest facility or f_i , whichever is the closest to u , but incurring a penalty because both options are farther than its old center. The overall penalty is stored in the array $z(\cdot)$, for each center.
2. Otherwise, the radius of that center would not change. This unchanged radius is stored in the array $r(\cdot)$, for each center.

These data structures have size $|S|$ because any open facility can become a center. They get updated by users who are not attracted by f_i and are crucial for the algorithm to decide the most profitable swap, i.e., the best facility to remove

given a candidate facility to insert. The pseudocode of this method is described in Algorithm 5.

Algorithm 5 PCP - move($f_i, U, S, D, \phi_1(\cdot), \phi_2(\cdot)$)

Input: f_i := facility to insert; U := set of users; S := solution set of centers; D := $|U| \times |F|$ distance matrix; $\phi_1(\cdot)$:= 1st closest facility of a user; $\phi_2(\cdot)$:= 2nd closest facility of a user

Output: (f_r, x^*) := pair of best facility to remove and the resulting objective function of swapping f_i and f_r

```

1:  $x' \leftarrow 0$ 
2:  $r(f) \leftarrow 0$ , for all  $f \in S$ 
3:  $z(f) \leftarrow 0$ , for all  $f \in S$ 
4: for  $u \in U$  do
5:   if  $d_i(u) < d_1(u)$  then
6:      $x' \leftarrow \max\{x', d_i(u)\}$ 
7:   else
8:      $z(\phi_1(u)) \leftarrow \max\{z(\phi_1(u)), \min\{d_i(u), d_2(u)\}\}$ 
9:      $r(\phi_1(u)) \leftarrow \max\{r(\phi_1(u)), d_1(u)\}$ 
10:  end if
11: end for
12:  $(x^*, f_r) \leftarrow \rho(S)$ 
13: return  $(f_r, x^*)$ 

```

Let $\sigma(S)$ be a function that returns the minimum possible objective function value that can result after applying the FVS to solution S . The formula can be accelerated by keeping track of the two greatest values from $r(\cdot)$ while updating the arrays.

$$\sigma(S) = \min_{f_j \in S} \{ \max\{x', z(f_j)\}, \max_{f_l \neq f_j} \{r(f_l)\} \} \quad (3.3)$$

Then let $\rho(S)$ be another function that returns a pair containing the result of calling $\sigma(S)$ and the corresponding facility that produced it, f_r .

$$\rho(S) = (\sigma(S), f_r) \quad (3.4)$$

The data structures altogether speed up the local search by partially eval-

uating the objective function, using the relations between f_i , U , and S for every possible swap, stored in $r(\cdot)$, $z(\cdot)$, and x' . This stored information allows the algorithm to keep track of multiple maximum distances, without repeating expensive computations, such as a complete revaluation of the objective function.

We observed that applying a modified FVS that considers Equation (3.2) solving the ANPCP, did not directly result in a good performance when $\alpha \geq 2$. This was mainly caused by the resulting objective function value after an interchange (a complete modification of the original solution) being unrelated to the information stored in the data structures, meaning that they were not enough to accurately evaluate the ANPCP. This was probably due to the need of keeping track of more relations for this problem than for the PCP. For this reason, we propose an adaptation of the FVS to obtain solutions for any value of $\alpha < p$. In the next subsection, we describe in more detail how we adapted the algorithm to solve the ANPCP.

3.3.3 ALPHA FAST VERTEX SUBSTITUTION (A-FVS)

According to the observations explained above, we adapted the FVS for the ANPCP, which from now on we will identify as the *Alpha Fast Vertex Substitution* (A-FVS), considering first the following modifications:

1. The group of users that is attracted to f_i was divided into two groups and, consequently, into two data structures as well. These two different groups came from the two possible cases for the new center in Equation (3.2). Therefore, a new variable was introduced for the objective function of the second case: x'' . The variable for the first case remained as x' .
2. The data structure $z(\cdot)$ was adapted to update all of the α -neighbors of every user, and not only its center. The reasoning for this is due to Equation (3.1). Let A_u be, for any user u , the set of open facilities closer than or equal to its

center, representing the α -neighbors of u . Note that in Figure 3.3, $\phi_{\alpha+1}$ is not part of A_u , which is shown over a purple background.

$$A_u \leftarrow \{\phi_k(u) \mid 1 \leq k \leq \alpha\} \quad (3.5)$$

$$A_u \subset S, \quad u \in U$$

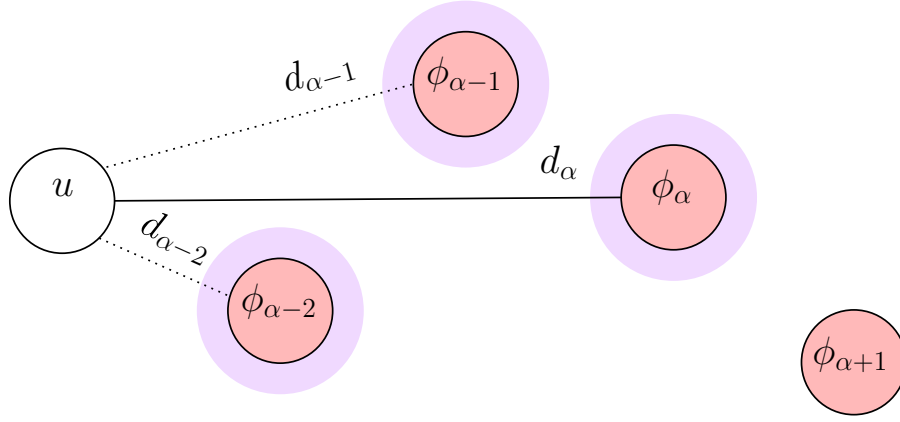


FIGURE 3.3: α -neighbors of u (A_u).

Each one of the α -neighbors is an open facility, therefore they can be removed from the solution. In other words, any of them can be selected to be f_r .

After evaluating the A-FVS in detail, we observed that the arrays did not accurately store the data between connections of users and centers, and the resulting objective function value was unrelated to them. During this evaluation, the following was noted:

1. The variables for the possible new objective function x' and x'' could be merged back, just as they were before. It is important to mention that the users still belong to two different groups; however, there is no need to use two different variables, because in any case, the equation applies a max function on both.

2. The data structure $z(\cdot)$ should be updated for all users, including those who are attracted by f_i because their α -neighbors were being considered when finding f_r , but these same users were not contributing to the array, so there was no stored information about what would happen to the centers that had any of their α -neighbors as f_r . This is why the objective function value after a swap and the data in the arrays were mismatched in previous evaluations.
3. As a consequence of the modification to $z(\cdot)$, data structure $r(\cdot)$ should be updated in the same way: for every user and all α -neighbors.

With these three observations, all users update and contribute to both arrays $z(\cdot)$ and $r(\cdot)$. This crucial difference from the original FVS is necessary for the ANPCP because both Equation (3.1) and Equation (3.2) are not properties of the PCP, and produced the following additional modifications:

1. If an α -neighbor will be removed (f_r), then its users must be reallocated unless they are attracted to f_i . In that case, their centers would not change, so the distances to them would remain constant (no penalty), independently of how close f_i is. Two examples of this case can be visualized in Figure 3.4,

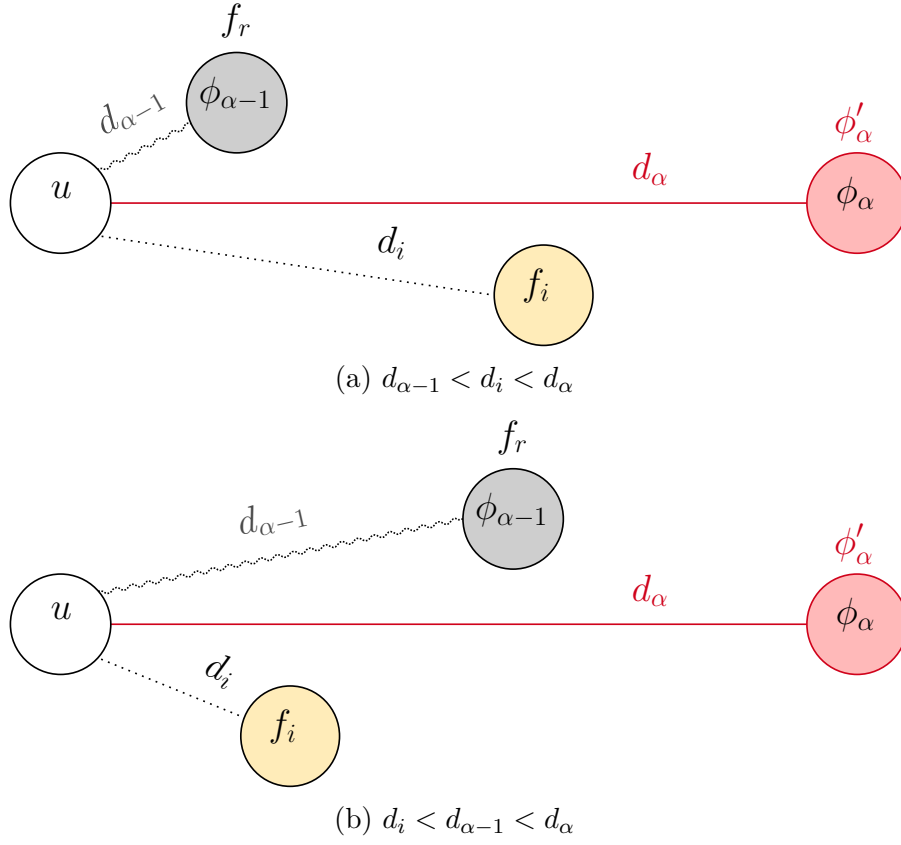


FIGURE 3.4: If an α -neighbor is f_r and u is attracted to f_i , its center remains unchanged (no penalty). In both of the above cases, ϕ'_α remains as the same facility as ϕ_α , even when the attraction of u to f_i varies.

However, in the worst case, if they were not attracted to f_i , then their new center would be either the $\alpha + 1$ closest facility or f_i , whichever is the closest, but still farther than their old center. Both possibilities are shown in Figure 3.5, where the red line is the new assignment of u as well as the penalty of interchanging f_r (the gray α -neighbor) with f_i . This penalty is stored in the array $z(\cdot)$, for each α -neighbor.

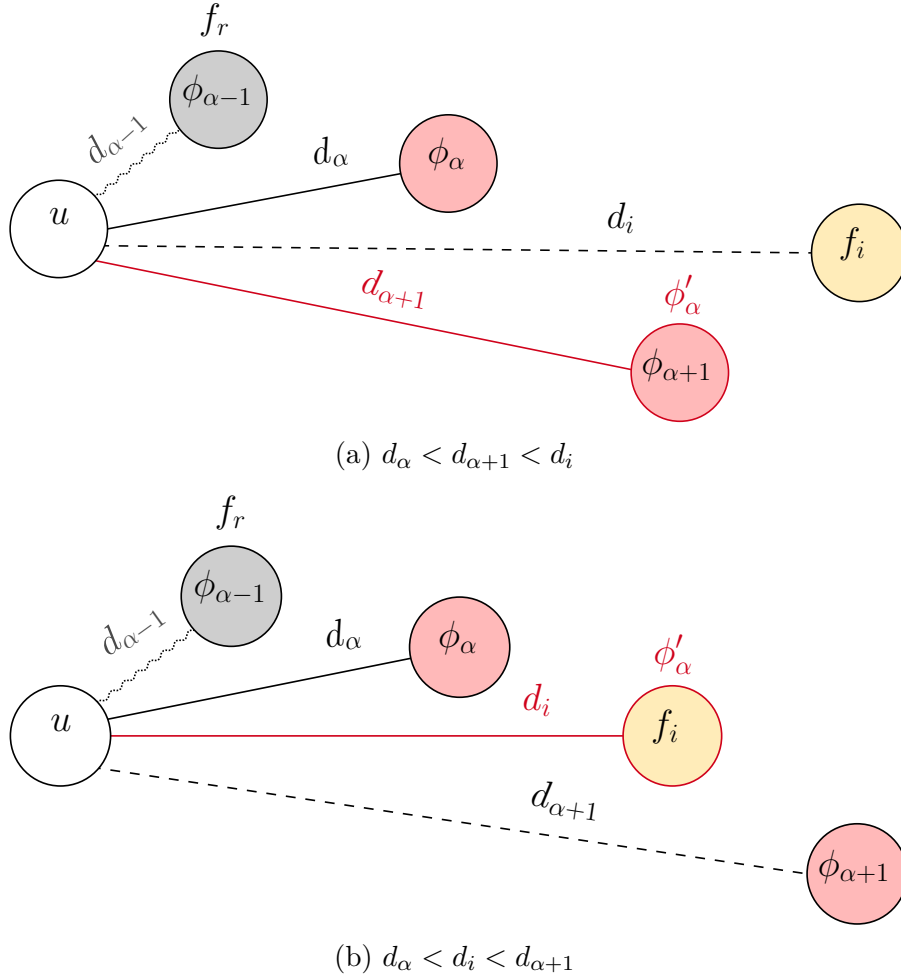


FIGURE 3.5: If an α -neighbor is f_r and u is not attracted to f_i , its new center will be farther (penalty). In both of the above cases, ϕ'_α is farther than ϕ_α , regardless of how far f_i is.

2. Otherwise, the radius of that α -neighbor would only change if its users are attracted to f_i . In that case, their new center would be either the $\alpha - 1$ closest facility or f_i , whichever is the farthest, as stated in Equation (3.2) and visualized in Figure 3.2, resulting in an improved assignment because this new distance would be shorter.

In the worst case, if they were not attracted to f_i , then their center would not change, remaining its radius unchanged. This possible gain is stored in array $r(\cdot)$, for each α -neighbor.

These two final modifications constitute the core step of the A-FVS, which we identify as the **move** procedure. The objective of this procedure is, in summary, given a candidate facility to insert, f_i , to store the greatest improvement of the objective function in x' , and for each α -neighbor of every user, to store the greatest penalty of removing it in $z(\cdot)$, and the greatest gain of keeping it in $r(\cdot)$. Then, it finds the facility that minimizes the maximum of all the distances from the three data structures and returns it as f_r together with that minimum distance as x^* .

The formula used in the **move** procedure to evaluate the objective function that results after a swap is the same used in the FVS (3.3). This is because the data structures of both algorithms do the same thing: they are just adapted to be updated differently. The pseudocode of this procedure is shown in Algorithm 6. This algorithm was the first local search modification that yielded an accurate result after evaluating it step by step.

The pseudocode of the complete local search procedure, including the move that uses the data structures, is shown in Algorithm 7.

NEIGHBORHOOD REDUCTION

The A-FVS uses an additional acceleration that increasingly reduces the size of the neighborhood $N(S)$ to explore during the iterations, by considering only facilities that would decrease the maximum distance between any user and its center, or “break” the objective function, to be inserted into the solution. More formally, let the current objective function value $x(S)$ be determined by the distance between a user u^* and its center $f^* = \phi_\alpha(u^*)$, that is, $x(S) = d(u^*, f^*)$. We call *critical allocation* to the assignment of the user-center that determines the objective function, *critical user* the u^* , and *critical center* the f^* . Note that there will be no improvement of x in $N(S)$ if f_i is farther from u^* than its center (f^*) because the critical allocation would not change. Therefore, selecting facilities closer to u^* than f^* as candidates

Algorithm 6 $\text{move}(f_i, U, S, D, \phi_k(\cdot))$

Input: f_i := facility to insert; U := set of users; S := solution set of centers; $D := |U| \times |F|$ distance matrix; $\phi_k(\cdot)$:= k th closest facility of a user

Output: (f_r, x^*) := pair of best facility to remove and the resulting objective function of swapping f_i and f_r

```

1:  $x' \leftarrow 0$ 
2:  $r(f) \leftarrow 0$ , for all  $f \in S$ 
3:  $z(f) \leftarrow 0$ , for all  $f \in S$ 
4: for  $u \in U$  do
5:   if  $d_i(u) < d_\alpha(u)$  then
6:      $m_z \leftarrow d_\alpha(u)$ 
7:      $m_r \leftarrow \max\{d_i(u), d_{\alpha-1}(u)\}$ 
8:      $x' \leftarrow \max\{x', m_r\}$ 
9:      $A'_u \leftarrow \{\phi_k(u) \mid 1 \leq k < \alpha\}$ 
10:  else
11:     $m_z \leftarrow \min\{d_i(u), d_{\alpha+1}(u)\}$ 
12:     $m_r \leftarrow d_\alpha(u)$ 
13:     $A'_u \leftarrow \{\phi_k(u) \mid 1 \leq k \leq \alpha\}$ 
14:  end if
15:  for  $f \in A'_u$  do
16:     $z(f) \leftarrow \max\{z(f), m_z\}$ 
17:     $r(f) \leftarrow \max\{r(f), m_r\}$ 
18:  end for
19: end for
20:  $(x^*, f_r) \leftarrow \rho(S)$ 
21: return  $(f_r, x^*)$ 

```

Algorithm 7 A-FVS($U, F, S, p, \alpha, D, \phi_k(\cdot)$)

Input: U := set of users; F := set of facilities; S := set of open facilities; p := number of open facilities; α := closeness position of centers; D := distance matrix $U \times F$; $\phi_k(\cdot)$:= k th closest facility of a user

Output: S := best solution set found

```

1:  $u^* \leftarrow \arg \min_{u \in U} \{x(S)\}$ 
2:  $improved \leftarrow \mathbf{true}$ 
3: while  $improved$  do
4:    $x^* \leftarrow x$ 
5:   for  $f_i \in F \setminus S$  do
6:      $d_i^* \leftarrow d(u^*, f_i)$ 
7:     if  $d_i^* < x$  then
8:        $(f_r, x') \leftarrow \text{move}(f_i, U, S, D, \phi_k(\cdot))$ 
9:       if  $x' < x^*$  then
10:         $x^* \leftarrow x'$ 
11:         $f_i^* \leftarrow f_i$ 
12:         $f_r^* \leftarrow f_r$ 
13:      end if
14:    end if
15:  end for
16:  if  $x^* < x$  then
17:     $x \leftarrow x^*$ 
18:     $S \leftarrow S \setminus \{f_r^*\} \cup \{f_i^*\}$ 
19:     $improved \leftarrow \mathbf{true}$ 
20:  else
21:     $improved \leftarrow \mathbf{false}$ 
22:  end if
23: end while
24: return  $S$ 

```

to insert into S avoids considering options that wouldn't improve the current S . This technique is used in Mladenović et al. (2003) for the PCP, and in Sánchez-Oro et al. (2022) for the ANPCP.

UPDATES

The concept of the α -neighbors is mathematically defined in Equation (3.5) using $\phi_k(u)$, which represents the k th closest open facility of a user u . This means that the algorithm needs to store what facilities are close to every user and to what degree, which is represented by k . We achieve this by implementing a matrix M of n rows and m columns, where each cell indicates the degree of closeness between a user u and a facility f . For each user, the assigned value in M for the open facilities in $\{\phi_k(u) | 1 \leq k \leq \alpha + 1\}$ is k . For the rest of the facilities, including the closed ones, the value is 0. Since every cell in M is updated, the time complexity to fill it is $O(mn)$.

This matrix serves as a memory that allows to calculate of the objective function value in $O(pn)$ time. For each user, its center can be found by iterating its row and looking for the column that has α as a value. To avoid visiting the closed facilities, we filter the search by using only the indexes of the open facilities (the solution).

Table 3.1 shows how an M matrix would look like for a small instance of $n = 6$ and $m = 5$ with a solution $S = \{f_2, f_3, f_4, f_5\}$ where $\alpha = 2$, from which the following can be observed:

- f_4 is the nearest open facility of u_1 and u_4 ($k = 1$).
- f_5 is the second nearest open facility of u_1 and u_4 ($k = 2$, their α th center).
- f_2 is the third nearest open facility of u_1 and u_4 ($k = 3$).

- The degree of closeness between f_3 and u_1 or u_4 is unknown ($k = 0$) because it is greater than $\alpha + 1$ and therefore unnecessary to store ($k > 3$).
- The degree of closeness between f_1 and any user is unknown ($k = 0$) because it is a closed facility and therefore unnecessary to store.

TABLE 3.1: Example of an M memory matrix where $\alpha = 2$.

	f_1	f_2	f_3	f_4	f_5
u_1	0	3	0	1	2
u_2	0	1	3	2	0
u_3	0	1	3	2	0
u_4	0	3	0	1	2
u_5	0	0	3	2	1
u_6	0	1	3	2	0

This updating step is used in both the NI and the A-FVS algorithms. However, while in the former the matrix is updated every time a swap is even considered, in the latter it is only updated after a swap is applied to the solution (line 18), drastically reducing the number of times that M needs to be filled.

CHAPTER 4

COMPUTATIONAL RESULTS

In this section, we present the results of the computational experiments of the proposed algorithm. In particular, we highlight the advantages of using the A-FVS described in Section 3.3.3 to solve the ANPCP. Additionally, the best configuration for the GRASP metaheuristic is tested and the results are analyzed.

We have generated several instances with random coordinates between 1 and 1000 on both axes, which were used to compare two local search algorithms in Section 4.1. We have also used instances of the TSPLIB library¹ to measure the efficiency of GRASP and calibrate its parameters. These instances use the same set of nodes to represent both the set of potential facilities and the set of user nodes. In this thesis, we define the ANPCP using two different sets of nodes: one representing the potential facilities (F), and another one the demand points or users (U), in which both can differ on distribution and cardinality. Thus, to use the TSPLIB to solve the ANPCP according to our definition, we have split some of the instances into 2 sets, by randomly choosing a fraction of the total nodes to be F , and the rest of them to be U . The variations of each instance are named as follows: `<name>_<n>_<m>_<index>`. For example, we split the instance `r11323` by randomly setting a third of the nodes to be potential facilities. Then, the name of this variation is `r11323_882_441_0` ($1323/3 = 441$). We can split the same instance again with the same fraction and

¹<http://comopt.ifl.uni-heidelberg.de/software/TSPLIB95/>

it would yield a different variation because the nodes for F are selected at random. The name of this other variation is `r11323_882_441_1`, notice the increase in the index at the end of the name.

For the experiments in this chapter, we have only considered $\alpha = 2, 3$ because $\alpha = 1$ represents the PCP, which is out of the scope of this work, and larger values would represent unrealistic situations. All algorithms were implemented in CPython 3.9 and executed over an Intel i7-4790K (4 GHz) with 32 GB RAM. The computer runs Windows 10 Pro 21H1. The open source code is released under the MIT license, publicly available at GitHub².

4.1 PERFORMANCE OF A-FVS

To compare the A-FVS against a simple interchange algorithm, identified as NI in this section, we performed multiple experiments using randomly generated instances. For these experiments, we considered two different strategies. One is the First Improvement (FI), which applies the first interchange that improves the current solution. The other is the Best Improvement (BI), which evaluates all possible interchanges within the current neighborhood and then applies the one that would yield the best solution. We also considered two types of initial solutions for the local search algorithms: on one hand, random starting solutions; and on the other, solutions greedily constructed with the GD heuristic from Section 3.2.2. The former is equivalent to $\text{RGD}(\beta = 1)$, and the latter to $\text{RGD}(\beta = 0)$.

The results, for each tested instance and strategy, are shown in Tables 4.1 to 4.4, and the averaged values among all instances and strategies, in Table 4.5. We tested five different instances for each configuration of the smallest size ($n = 50, m = 50$), four for the medium ($n = 100, m = 100$), and three for the biggest ($n = 500, m = 500$). The configurations are combinations of each size with $p =$

²<https://github.com/netotz/alpha-neighbor-p-center-problem>

$0.1m, 0.25m, 0.5m$ and $\alpha = 2, 3$. The contents of the tables are the average values of grouping the results of all the experiments, where $x(S)$ is the objective function value of the final solution, Imp. (%) is the relative improvement in $x(S)$, Time (s) is the computing time required by the algorithm to finish in seconds, and Moves is the number of times that a vertex interchange was applied and resulted in a better quality solution, i.e., the number of improvements before the local search reached a local optimum. The first four columns from left to right are the parameters of the instances and the configurations. The fifth column is the objective function value of the initial solution, either constructed by the GD or randomly generated. The next four columns are the results of the solution improved by NI, and the last four are those improved by the A-FVS. Then, the results of these four tables are grouped by strategy and constructive heuristic, showing and comparing their mean values in Table 4.5.

TABLE 4.1: GD solution improved with FI strategy.

n	m	p	α	GD	NI				A-FVS			
				$x(S)$	$x(S)$	Imp. (%)	Time (s)	Moves	$x(S)$	Imp. (%)	Time (s)	Moves
50	50	5	2	682.40	524.60	22.15	0.73	6.20	502.60	25.49	0.03	8.20
			3	870.40	636.20	26.34	0.87	8.20	636.40	26.30	0.04	8.40
		12	2	409.00	316.40	22.89	0.58	6.00	309.20	24.70	0.02	7.20
			3	510.60	396.40	21.72	0.29	5.60	401.80	20.77	0.01	4.40
		25	2	324.20	281.40	12.80	0.05	1.20	276.00	14.87	0.00	1.80
			3	373.60	325.60	12.58	0.05	2.80	325.60	12.58	0.01	2.40
100	100	10	2	439.50	376.75	14.11	4.42	6.25	366.00	16.47	0.08	8.00
			3	590.00	499.25	14.48	7.27	4.50	485.75	16.87	0.10	6.25
		25	2	277.25	222.50	18.75	2.63	8.00	205.00	25.30	0.11	13.50
			3	324.25	268.75	17.02	5.92	8.75	254.25	21.45	0.13	13.00
		50	2	211.50	180.75	14.53	0.38	4.00	180.75	14.53	0.02	3.50
			3	251.25	214.00	14.12	1.02	4.50	206.75	17.08	0.04	6.00
500	500	50	2	191.00	161.33	15.36	623.47	6.67	142.67	25.11	3.41	28.00
			3	231.00	198.00	13.91	559.60	4.33	184.00	19.91	1.61	13.00
		125	2	135.00	108.00	19.48	294.20	3.33	100.67	25.30	1.35	12.00
			3	159.67	121.67	22.12	327.13	8.33	119.00	23.73	1.10	10.00
		250	2	92.33	89.33	3.30	15.06	1.33	89.33	3.30	0.20	1.67
			3	115.33	102.67	10.37	16.09	2.00	102.67	10.37	0.21	2.33

The time difference is noticeable even in small instances, but increases in larger instances, especially when using the BI strategy. The differences in the improvement of solution quality are also relevant and show that A-FVS is not only a much faster

TABLE 4.2: GD solution improved with BI strategy.

n	m	p	α	GD	NI				A-FVS			
				$x(S)$	$x(S)$	Imp. (%)	Time (s)	Moves	$x(S)$	Imp. (%)	Time (s)	Moves
50	50	5	2	682.40	516.20	23.36	2.26	4.40	506.60	24.83	0.05	4.80
			3	870.40	643.00	25.50	3.11	4.00	643.00	25.50	0.07	4.00
		12	2	409.00	315.80	23.05	1.32	3.40	312.20	23.93	0.02	4.20
			3	510.60	406.20	19.88	2.01	3.00	405.80	19.97	0.03	3.20
		25	2	324.20	281.40	12.80	0.22	1.20	276.00	14.87	0.01	1.60
			3	373.60	325.60	12.58	0.66	2.20	320.60	14.18	0.01	3.00
		100	2	439.50	375.50	14.35	19.39	3.75	339.50	22.70	0.25	9.00
			3	590.00	504.50	13.70	23.88	2.75	484.75	17.03	0.27	5.75
100	100	25	2	277.25	225.50	17.83	12.14	3.75	206.25	24.66	0.15	9.00
			3	324.25	263.50	18.61	25.70	6.00	255.50	21.08	0.18	8.50
		50	2	211.50	180.75	14.53	5.06	3.00	180.75	14.53	0.02	2.50
			3	251.25	214.00	14.12	7.32	3.50	205.00	17.71	0.05	5.50
		500	2	191.00	158.00	17.03	3834.00	6.67	142.33	25.32	5.10	17.67
			3	231.00	198.00	13.91	3134.87	4.00	183.00	20.28	5.11	13.00
		125	2	135.00	108.00	19.48	1459.31	3.33	100.33	25.60	2.39	13.67
			3	159.67	122.00	21.94	1529.05	4.67	119.00	23.73	1.83	9.67
500	500	250	2	92.33	89.33	3.30	159.19	1.33	89.33	3.30	0.14	1.33
			3	115.33	102.67	10.37	197.16	2.00	102.67	10.37	0.28	2.33

algorithm than NI but also more efficient by yielding better objective function values. Furthermore, it can be observed that A-FVS applies much more moves as the size of the instance increases, which indicates that it explores more neighborhoods and therefore increases the diversification of the local search, without incurring a penalty in time. For these reasons, we decided to use A-FVS as the local search step in the GRASP framework.

In Table 4.5, by averaging out the results of all sizes, it can be observed that A-FVS outperforms NI in both solution quality and computing time. Note that using GD to greedily construct an initial solution allows A-FVS to yield slightly better quality in the objective function than starting from a random solution, while taking less time. This shows that GD is useful to get upper bounds for the ANPCP, despite not evaluating its objective function. A-FVS coupled with the BI strategy improves the initial solution by 0.3% more than the FI; however, it takes around double the time. As a consequence, the GD heuristic is worth using in the construction phase for GRASP as the RGD, and the FI strategy not only improves the solution quality produced by A-FVS, but it also takes less time.

TABLE 4.3: Random solution improved with FI strategy.

n	m	p	α	<i>Random</i>	NI				A-FVS			
				$x(S)$	$x(S)$	Imp. (%)	Time (s)	Moves	$x(S)$	Imp. (%)	Time (s)	Moves
50	50	5	2	847.80	514.20	38.10	0.50	10.60	512.80	38.28	0.02	10.20
			3	888.40	636.00	28.14	0.58	6.40	633.60	28.43	0.02	6.40
		12	2	522.80	323.20	37.23	0.19	9.20	314.00	39.00	0.01	9.40
			3	687.00	397.20	42.08	0.20	9.80	380.20	44.15	0.02	10.80
		25	2	355.60	282.00	19.77	0.04	2.80	276.00	21.69	0.01	4.20
			3	435.80	323.00	26.24	0.06	6.20	322.00	26.49	0.01	6.40
100	100	10	2	574.00	357.00	35.19	3.42	9.75	349.50	36.37	0.04	11.00
			3	644.50	475.75	25.38	4.98	9.50	481.75	24.17	0.06	9.25
		25	2	387.75	220.00	40.19	2.82	13.25	217.25	42.15	0.06	13.50
			3	415.25	258.00	37.62	4.04	14.50	257.00	37.79	0.06	13.25
		50	2	235.75	189.50	19.98	1.04	5.75	182.50	23.05	0.03	8.25
			3	288.25	230.50	19.88	0.76	4.25	210.50	26.71	0.04	10.75
500	500	50	2	298.67	188.33	36.87	428.03	7.00	163.33	45.43	1.90	24.33
			3	339.67	230.67	31.44	605.43	8.67	182.67	45.11	2.34	29.67
		125	2	182.33	120.00	34.09	406.94	9.00	103.67	43.06	1.88	25.67
			3	229.67	144.67	36.16	364.20	10.67	130.67	42.60	2.18	29.33
		250	2	119.33	94.33	20.50	42.83	8.00	89.33	25.13	1.07	14.33
			3	144.67	108.00	24.74	78.57	7.33	108.00	24.74	0.61	7.33

TABLE 4.4: Random solution improved with BI strategy.

n	m	p	α	<i>Random</i>	NI				A-FVS			
				$x(S)$	$x(S)$	Imp. (%)	Time (s)	Moves	$x(S)$	Imp. (%)	Time (s)	Moves
50	50	5	2	847.80	524.00	36.70	1.50	4.20	515.60	37.99	0.03	4.40
			3	888.40	633.60	28.43	1.53	3.80	633.60	28.43	0.04	3.20
		12	2	522.80	326.00	36.78	2.03	6.60	314.00	39.00	0.02	7.80
			3	687.00	387.20	43.21	2.25	8.00	377.60	44.53	0.05	10.20
		25	2	355.60	282.00	19.77	0.34	2.20	276.00	21.69	0.01	3.20
			3	435.80	325.40	25.62	0.71	4.80	322.00	26.49	0.01	6.00
100	100	10	2	574.00	353.75	35.77	19.75	9.25	322.25	42.17	0.15	10.00
			3	644.50	486.00	23.73	20.09	5.50	479.00	25.00	0.15	6.00
		25	2	387.75	226.50	39.22	25.29	10.25	210.50	43.17	0.12	13.50
			3	415.25	268.50	35.00	28.73	8.75	255.25	38.24	0.18	14.25
		50	2	235.75	186.75	21.19	9.24	6.00	182.50	23.05	0.04	7.25
			3	288.25	230.50	19.88	7.29	3.75	210.50	26.71	0.07	8.75
500	500	50	2	298.67	182.00	39.03	3642.07	6.67	147.67	50.48	4.85	24.67
			3	339.67	233.33	30.55	4706.46	7.00	179.67	46.05	7.40	28.00
		125	2	182.33	120.00	34.09	3420.12	8.33	99.00	45.82	4.40	31.00
			3	229.67	142.33	37.23	5668.66	11.00	146.00	37.13	4.25	22.67
		250	2	119.33	98.33	17.19	762.28	4.67	93.33	21.82	1.17	9.67
			3	144.67	108.00	24.74	1212.94	6.33	108.00	24.74	0.90	6.67

TABLE 4.5: Average results of the local search comparison.

Initial	Strategy	NI			A-FVS		
		Imp. (%)	Time (s)	Moves	Imp. (%)	Time (s)	Moves
GD	FI	16.45	103.32	5.11	19.12	0.47	8.31
	BI	16.46	578.70	3.50	19.42	0.89	6.60
<i>Random</i>	FI	30.75	108.03	8.48	34.13	0.58	13.56
	BI	30.45	1085.07	6.51	34.58	1.32	12.07

4.2 PERFORMANCE OF GRASP

In this section, we show and analyze the experiments and their results to measure our proposed GRASP method. We first conduct two experiments to calibrate the input parameters of GRASP, i_{\max} and β , on a subset of selected instances. With the insight obtained from these calibrations, we select the most adequate values for the parameters, and then we proceed to run the GRASP on the whole set of instances.

4.2.1 CALIBRATING i_{\max}

This experiment was designed to calibrate the best value to use for the maximum number of iterations, i_{\max} . Here, we try to expose the algorithm to challenging instances with challenging parameters, such as setting p to be less than 10% of the total facilities ($p < 0.1m$), since the smaller the value for p , the more possible combinations of distinct solutions, and therefore the greater the size of neighborhoods. This implies a greater effort for the local search to find the best move, because there are more candidate moves to explore. Because the goal of this experiment is to calibrate i_{\max} , we fixed α and p , and for β we chose a random value between 0 and 1 in every construction, to minimize their influence as factors. We also fixed $i_{\max} = 5000$ and we ran GRASP once for two large instances from the TSPLIB and for one large instance randomly generated. The reasons why we selected these three instances are explained in the following experiments and their results.

The first instance we tried is `r11323_882_441_0`, which is one of the largest sizes from the subset of the TSPLIB, using $p = 44$ as (almost) the 10% of $m = 441$, and both $\alpha = 2, 3$. Since it is likely to find more diversity in the space of solutions as p gets smaller, we hypothesized that these two configurations would allow for more improvements. Nevertheless, in this test, there were no improvements during the 5000 iterations. In every iteration, the local search phase resulted in the same

objective function value, despite of the construction heuristic yielding varying values. We tried the second largest instance, `pr1002_668_334_4`, using the same values for p and α as before, yet without any single improvement. The reason why the two greatest instances did not report any improvement at all was found out to be the coordinates of the nodes, as we explain and show later on. As a workaround, we retried the experiment with `r11323_882_441_0` but lowering p to 20, now being the 4.53% of $m = 441$, and $\alpha = 2, 3$. As mentioned earlier, a smaller p implies that the space of possible solutions is greater, and thus for this experiment, it would be expected to see more improvements.

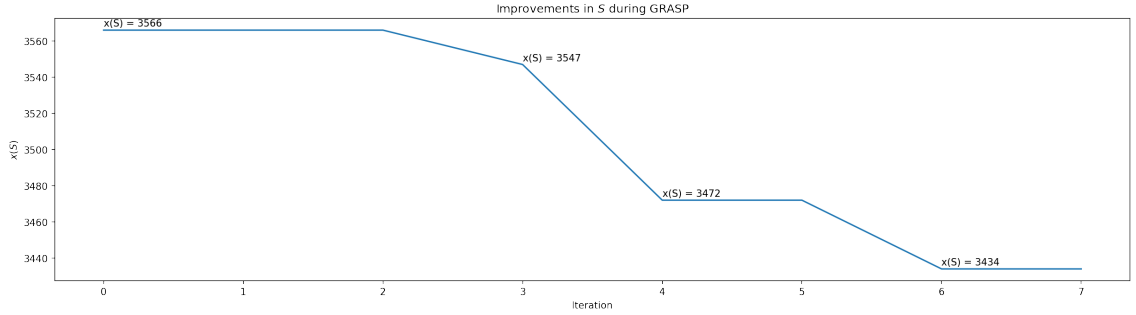
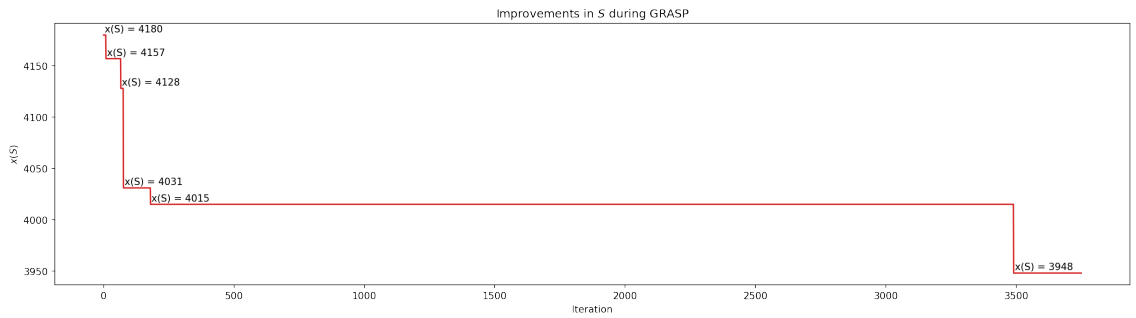
Table 4.6 includes more details about the iterations in which a new best objective function value was found (the improvements) for `r11323_882_441_0`. i is the iteration number in which the improvement occurred, where $i = 0$ is the first one; β is the value that was used for each construction; $x_c(S)$ is the objective function value of the constructed solution by RGD, while $x(S)$ is that of the improved solution by A-FVS and the value that is taken into account for the improvements; Imp. (\%) is the increase in the percentage of the previous best solution quality, taking the solution of the first iteration as the starting value; Time (s) is the total running time in seconds that GRASP had been using so far; and i.w.i. is the number of iterations without improvement from the previous best solution. Figure 4.1 shows how the solution evolved through the iterations of GRASP, up to the last improvement. In this case, the solution for $\alpha = 2$ stopped improving very early in the test, and there were 2 more improvements for $\alpha = 3$.

TABLE 4.6: Improvements in `r11323_882_441_0`, $p = 20$.

i	β	$x_c(S)$	$x(S)$	Imp. (%)	Time (s)	i.w.i.
0	0.49	5676	3566	-	2.31	-
3	0.51	4800	3547	0.53	10.58	3
4	0.25	5392	3472	2.11	15.47	1
6	0.69	6249	3434	1.09	23.87	2

(a) $\alpha = 2$

i	β	$x_c(S)$	$x(S)$	Imp. (%)	Time (s)	i.w.i.
0	0.22	5624	4180	-	6.56	-
9	0.94	8196	4157	0.55	40.00	9
66	0.85	7517	4128	0.70	239.76	57
76	0.38	5727	4031	2.35	275.02	10
180	0.22	5932	4015	0.40	621.83	104
3490	0.41	6261	3948	1.67	11237.42	3310

(b) $\alpha = 3$ (a) $\alpha = 2$ (b) $\alpha = 3$ FIGURE 4.1: Visualization of $x(S)$ from Table 4.6.

The next experiment was run with a smaller instance, `rat783_522_261_0`, using $p = 20$ and $\alpha = 2, 3$. For this instance, $p = 20$ represents the 7.66% of $m = 261$. Despite this percentage being larger than the one used in Table 4.6 (4.53%), and hence a potential reduction of the solutions space, there were actually more improvements (9) than in the other (6). These results are detailed in Table 4.7 and visualized in Figure 4.2. For both values of α , the solutions were improved almost the same number of times, being again $\alpha = 2$ the one that stopped improving earlier.

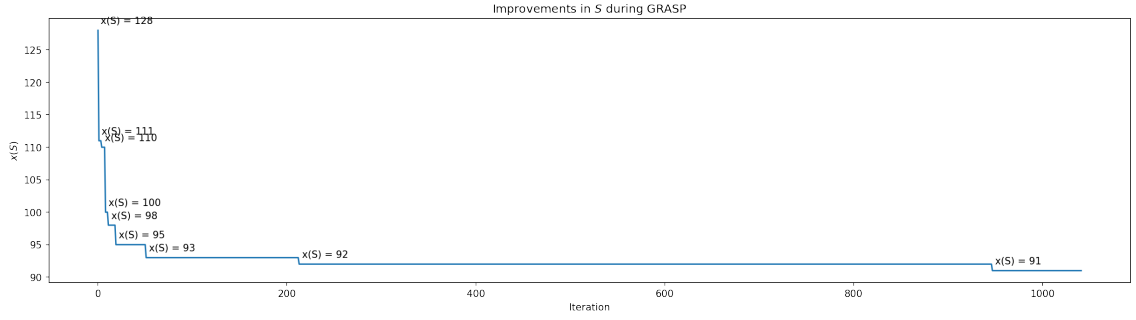
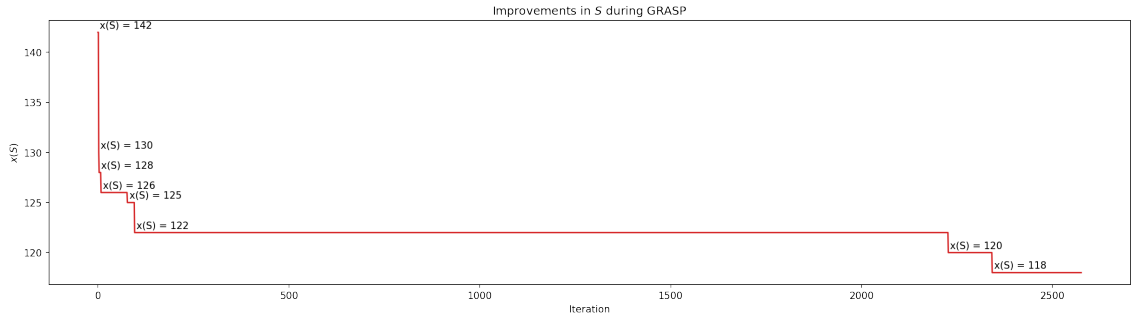
TABLE 4.7: Improvements in `rat783_522_261_0`, $p = 20$.

i	β	$x_c(S)$	$x(S)$	Imp. (%)	Time (s)	i.w.i.
0	0.76	130	128	-	0.24	-
1	0.22	127	111	13.28	0.53	1
4	0.51	150	110	0.90	3.49	3
8	0.14	130	100	9.09	6.08	4
11	0.31	157	98	2.00	8.68	3
19	0.12	160	95	3.06	17.51	8
51	0.19	121	93	2.11	42.77	32
213	0.53	140	92	1.08	152.64	162
947	0.62	131	91	1.09	633.43	734

(a) $\alpha = 2$

i	β	$x_c(S)$	$x(S)$	Imp. (%)	Time (s)	i.w.i.
0	0.43	163	142	-	0.36	-
2	0.45	165	130	8.45	2.13	2
3	0.38	165	128	1.54	2.96	1
8	0.31	174	126	1.56	7.60	5
77	0.10	164	125	0.79	44.87	69
96	0.59	161	122	2.40	57.60	19
2227	0.45	172	120	1.64	1472.83	2131
2342	0.32	172	118	1.67	1542.07	115

(b) $\alpha = 3$

(a) $\alpha = 2$ (b) $\alpha = 3$ FIGURE 4.2: Visualizations of $x(S)$ from Table 4.7.

After plotting both instances, we realized that the nodes in `r11323_882_441` are placed in several horizontal lines, while those in `rat783_522_261` are placed more randomly. This can be observed in Figure 4.3a and Figure 4.3b, respectively. The arrangement of the nodes in the larger instance makes it difficult to escape a local optimum as p increases. For this reason, we generated a random instance with a very similar shape (space of coordinates) as `r11323_882_441` and with the same number of users and facilities, named as `anpcp_882_441_0`, to run the same experiment and compare how the coordinates of the nodes affect the convergence of the objective function. The nodes of this new instance now have a very similar distribution to those of `rat783_522_261`, which can be seen in Figure 4.3c. Table 4.8 and Figure 4.4 display the results of this test, for both $\alpha = 2, 3$. Note that there were multiple improvements despite p being 10% of the total facilities, which confirms the hypothesis that the disposition of the nodes is a relevant factor in how the objective function converges to a local optimum. The solution stopped improving earlier for

$\alpha = 3$, but the total number of improvements is greater (11) than for $\alpha = 2$ (7).

TABLE 4.8: Improvements in `anpcp_882_441_0`, $p = 44$.

i	β	$x_c(S)$	$x(S)$	Imp. (%)	Time (s)	i.w.i.
0	0.41	3130	2275	-	4.45	-
3	0.90	3948	2224	2.24	22.32	3
5	0.75	4278	2209	0.67	33.35	2
7	0.42	3038	2206	0.14	41.99	2
10	0.30	3154	2185	0.95	57.05	3
110	0.30	3154	2166	0.87	452.61	100
1541	0.13	3140	2165	0.05	6040.19	1431
3710	0.69	3779	2161	0.18	14531.35	2169

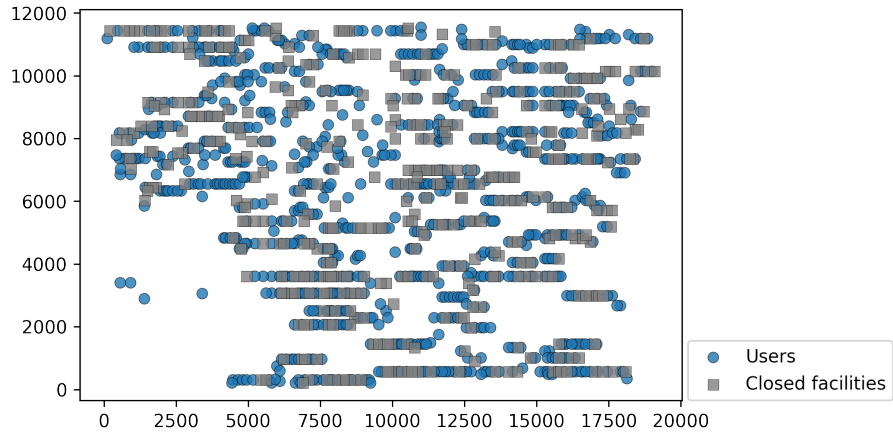
(a) $\alpha = 2$

i	β	$x_c(S)$	$x(S)$	Imp. (%)	Time (s)	i.w.i.
0	0.67	3952	2887	-	5.88	-
1	0.35	3714	2807	2.77	11.64	1
5	0.62	4315	2803	0.14	29.70	4
15	0.16	3894	2793	0.36	71.69	10
20	0.30	3763	2726	2.40	95.30	5
82	0.08	3763	2714	0.44	373.13	62
83	0.72	4471	2702	0.44	380.92	1
173	0.46	4001	2686	0.59	798.96	90
312	0.17	3686	2679	0.26	1438.55	139
647	0.42	4119	2642	1.38	2929.06	335
1131	0.43	4465	2637	0.19	4956.00	484
2050	0.45	4281	2629	0.30	8793.84	919

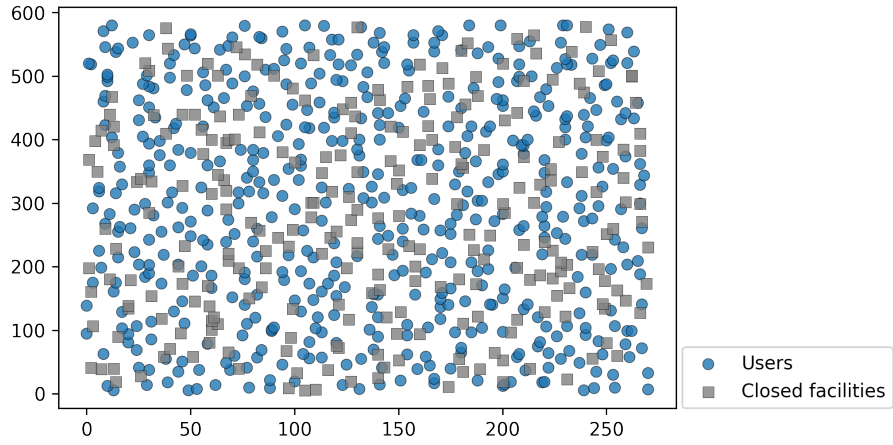
(b) $\alpha = 3$

With these three experiments, we know that a solution may improve even after 5000 iterations, especially with large instances and with nodes whose coordinates follow a random-like distribution. As a consequence, we have decided to modify the stopping criteria for GRASP. Instead of setting a fixed number of maximum iterations for the whole procedure, it now sets a maximum number of consecutive iterations *without improvements* to stop the procedure. According to the results of this subsection (which are biased by the subset of the 3 tested instances), we can estimate that most of the best improvements would take place before 100 consecutive

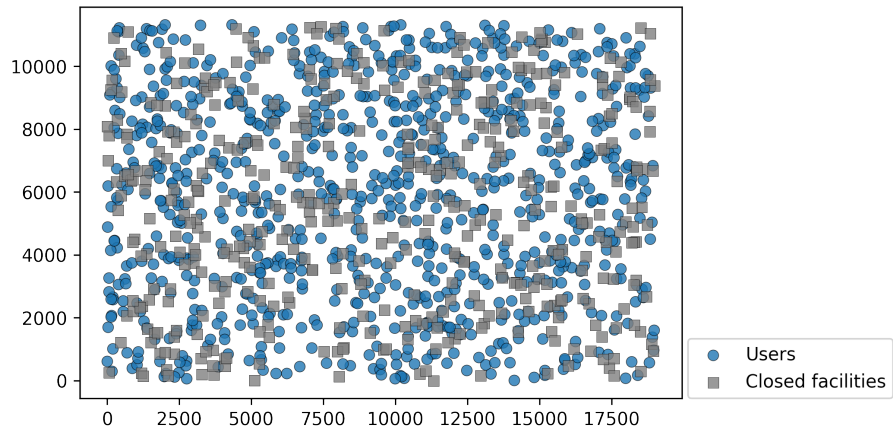
iterations without an improvement. However, in cases where a small improvement in the quality of the solution of 1% or even less could drastically impact the consequences of a decision to a problem, and where the application of a solution could wait a couple of hours, or even days depending on the size of the problem instance, then setting this parameter to numbers as large as 5000 or more could certainly yield better results. Note that, in general, the larger the instance, the more iterations it could take to reach a local optimum that is difficult to overtake. Therefore, we can say that a reactive or adaptable way to set i_{\max} based on both the size and the distribution of an instance would be ideal for this parameter. Even so, for the experiments in the next sections, we will fix $i_{\max} = 100$ in an attempt to reduce the overall execution time and continue with the experimentation.



(a) r11323_882_441_0

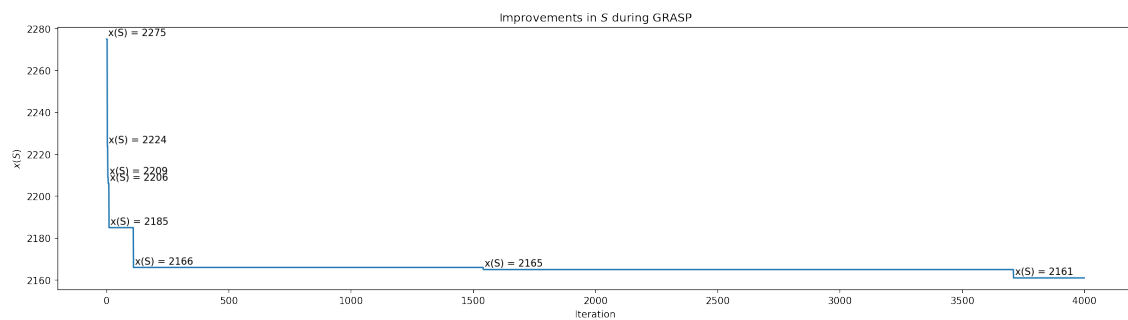
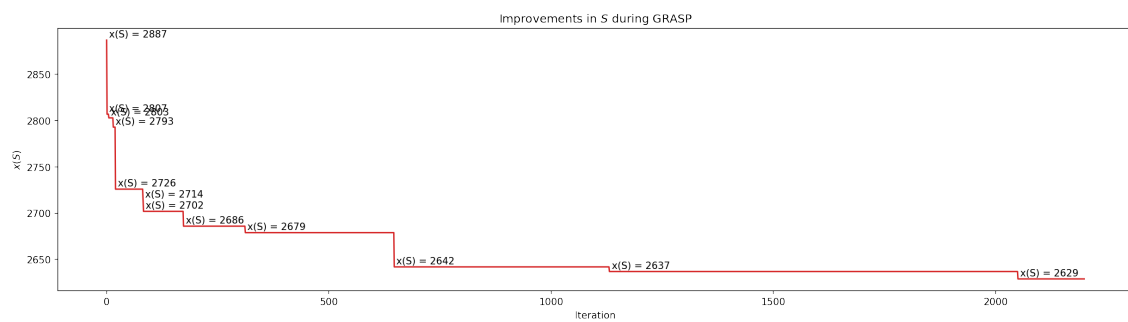


(b) rat783_522_261_0



(c) anpcp_882_441_0

FIGURE 4.3: Distribution of nodes in sample instances.

(a) $\alpha = 2$ (b) $\alpha = 3$ FIGURE 4.4: Visualizations of $x(S)$ from Table 4.8.

4.2.2 CALIBRATING β

In this experiment, we calibrate β , the parameter that determines the randomness of the RCL. We tested 7 values: $\beta = \{0, 0.2, 0.4, 0.6, 0.8, 1, R\}$, where R means that a random number between 0 and 1 is used as β in each construction. This is different from $\beta = 1$, which means that every construction yields a random solution. Values 0 and 1 are included to see how a completely greedy and a completely random (respectively) construction influences the quality of the results. We fixed $i_{\max} = 100$ for each call to GRASP. We used a subset of 3 instances from the TSPLIB and split them by setting a third of the total nodes to be considered as potential facilities: `rd400_267_133`, `rat783_522_261`, and `nrv1379_920_459`; identified with the letters S (small), M (medium), and L (large), respectively. These instances were split 10 times, giving 10 different variations for each one. We set $p = \{0.05m, 0.1m, 0.2m\}$ and $\alpha = \{2, 3\}$ for every test, which provided 180 distinct configurations. Each configuration was then tested for the 7 values of β , resulting in a total of 1260 conducted tests.

Table 4.9 groups all the tests and shows the average results of the 10 variants for every unique configuration of parameters, while the means of these results are in Table 4.10. Table 4.9 includes 3 groups of columns: from left to right, the first group is just the identifier of the TSPLIB instance (sub-column I) and the parameters of the solver (sub-columns p and α); the second one, Deviation (%), is the average deviation with respect to the best solution found in the experiment; and the third group, Time (s), is the average computing time in seconds required by GRASP. Each of these 2 last groups are divided in 7 sub-columns, which are the tested values of β . A deviation of 0% would mean that a particular value of β yielded the best solution in all of the tests. Therefore, the smaller the deviation a β has, the better quality it produces. These minimum values are shown in bold, for each row. Note that we refer to the best solution found in this experiment, and not to the optimal solution of a particular instance. The minimum time taken in each row is also shown in bold.

TABLE 4.9: Average results of the experiment, grouped by instance and parameters.

I	p	α	Deviation (%)							Time (s)						
			0	0.2	0.4	0.6	0.8	1	R	0	0.2	0.4	0.6	0.8	1	R
S	6	2	0.96	0.73	0.60	0.58	0.21	0.33	0.29	26.16	28.77	26.10	28.21	29.51	27.98	26.11
		3	0.52	0.14	0.14	0.00	0.00	0.09	0.09	41.38	39.53	38.22	37.69	38.58	39.59	38.91
	13	2	2.33	1.03	1.16	1.33	0.68	1.31	1.34	27.54	37.18	32.61	30.14	38.17	41.20	37.02
		3	1.74	0.23	0.44	1.12	0.32	0.65	0.62	32.35	42.97	40.15	41.56	43.37	38.27	34.93
	26	2	1.19	0.57	1.03	0.98	0.41	0.57	0.88	29.57	28.82	27.53	29.83	32.09	36.74	29.92
		3	1.58	0.87	0.75	0.33	0.89	0.29	1.00	24.24	31.62	34.86	36.65	36.60	45.48	38.04
M	13	2	6.23	2.78	2.16	1.90	2.78	2.95	3.38	105.49	90.91	118.73	93.29	115.99	135.81	105.73
		3	1.63	1.29	1.57	0.92	1.70	2.86	2.09	108.18	105.57	123.30	143.48	135.61	122.79	104.90
	26	2	1.88	2.38	2.00	2.25	1.13	3.37	1.63	78.37	76.09	81.07	93.06	90.99	124.44	85.66
		3	1.61	2.52	0.92	3.11	3.43	3.34	3.24	83.89	74.79	96.51	98.93	92.75	106.80	91.62
	52	2	1.28	1.29	0.92	1.47	2.38	4.40	1.29	75.29	67.05	73.97	81.31	99.15	92.36	86.00
		3	1.61	1.46	2.65	2.50	3.51	3.09	1.03	67.00	80.66	69.56	70.89	101.51	101.11	76.05
L	22	2	2.85	1.63	1.46	1.00	2.42	1.56	2.17	435.81	412.44	535.88	535.93	426.15	519.43	496.27
		3	1.51	0.86	1.07	1.92	1.58	1.25	1.65	410.03	552.72	538.26	509.24	641.91	781.19	578.80
	45	2	1.57	1.15	1.18	2.27	1.88	3.17	1.25	424.41	414.26	448.11	530.66	547.57	551.31	526.36
		3	1.07	1.81	1.05	0.68	1.97	3.05	1.44	412.98	384.18	539.40	521.55	521.66	530.16	482.52
	91	2	0.26	0.00	0.15	0.31	0.10	0.51	0.36	266.55	259.36	298.87	344.16	470.86	525.01	306.01
		3	0.25	0.25	0.87	1.12	1.00	1.83	0.88	345.95	340.45	361.07	385.74	444.35	591.48	439.43

Table 4.10 shows the overall average results of Table 4.9. This table provides an insight on the behaviour of the algorithm when it is subjected to diverse cases of varying sizes and parameters. The columns represent a value of β and the rows represent the following metrics: #Bests is the number of times (out of 18) that the average deviation was the minimum of a unique configuration; Avg. Dev. (%) is the mean of the average deviations; and Avg. Time (s) is the mean of the average times in seconds. Bold values in the #Bests row are the maximum, while bold values in the other 2 rows are the minimum.

TABLE 4.10: Means of Table 4.9.

	β						
	0	0.2	0.4	0.6	0.8	1	R
#Bests	1.00	5.00	1.00	5.00	5.00	1.00	1.00
Avg. Dev. (%)	1.67	1.17	1.20	1.32	1.47	1.92	1.37
Avg. Time (s)	166.40	170.41	193.57	200.68	217.05	245.06	199.13

It can be derived from these results that the best value for the β parameter is $\beta = 0.2$ in terms of both the number of best solutions found and the average deviation. There are two other values that match the number of best solutions

found (5), $\beta = 0.6$ and $\beta = 0.8$, but their deviations indicate a 13.29% and 25.6% decrease in the quality of the solutions, respectively. Also, $\beta = 0.2$ is the second best option in computing time, only 2.4% slower than the fastest option, $\beta = 0$. The reason why the completely greedy constructions ($\beta = 0$) resulted, in average, in the fastest GRASP executions, is probably due to a decrease in the number of possible better solutions, and thus the local search performs less moves and finishes in less time. At the same time, this full intensification leaves little or no room to improve the constructed solutions, and therefore results in worse quality: $\beta = 0$ is the second worst option in terms of the average deviation, followed by $\beta = 1$, which is in fact the one that took the longest average time. As a consequence of the showed evidence, we have decided that the best value is $\beta = 0.2$, and it is the one used in the next section.

4.2.3 FINAL RESULTS

The main objective of this section is to observe and analyze the performance of the proposed GRASP. To the best of our knowledge, this is the first attempt to solve with a heuristic approach the discrete version of the ANPCP that uses two different sets of nodes for the potential facilities and the users.

For this experiment, we considered using $p = \{0.05m, 0.1m, 0.15m\}$ and $\alpha = \{2, 3\}$ on a subset of 6 instances from the TSPLIB. As in previous experimentation, each of these instances was randomly split 10 times to set a third of the total nodes to be considered as potential facilities. Therefore, we used 60 different configurations for each instance, giving a total of 360 tests.

Table 4.11 is a summary of the results grouped by instance of running GRASP($\beta = 0.2$, $i_{\max} = 100$) once for each of the instances in the selected subset, showing the following metrics: Instance, the name of the split TSPLIB instance that was solved; $x(S)$ Differences, the minimum, maximum, and average (sub-columns Min. (%), Max. (%), and Avg. (%), respectively) relative differences in the quality between the best improved solution by the local search heuristic and the best constructed one (regardless of the iterations), and the number of times (out of 60) that the local search provided a better solution than the best constructed one (sub-column $\# > 0$); Avg. Time (s), the average of the 60 computing times in seconds required to run GRASP on each configuration with a limit of 1800 seconds; and $\# i_l > 0$, the number of times (out of 60) that the last iteration in which a new best solution was found occurred after the first iteration (0), i.e., the number of times that GRASP found at least one better solution than the initial one. The details of these summarized results can be found in Appendix A, where a table is reported for each TSPLIB instance.

An $x(S)$ Difference of 0% means that the A-FVS algorithm could not yield a better solution than the best constructed solution, which would make it useless as

TABLE 4.11: Summarized results of GRASP experiments.

Instance	$x(S)$ Differences				Avg. Time (s)	# $i_l > 0$
	Min. (%)	Max. (%)	Avg. (%)	# > 0		
pr439_293_146	0	36.02	13.58	57	31.35	34
rat575_384_191	6.43	21.36	14.63	60	51.85	59
rat783_522_261	4.79	21.95	13.69	60	96.68	59
dsj1000_667_333	19.05	34.48	26.46	60	240.53	59
r11323_882_441	0	37.46	13.86	53	255.95	21
r11889_1260_629	13.15	28.49	20.36	60	1337.72	60

the local search component of GRASP. This situation occurred in only 10 out of the 360 conducted tests (2.77%), demonstrating the efficiency of the A-FVS. The only two instances that reported at least one occurrence of this situation (where Min. (%) = 0) are the only two that reported less than 60 better solutions by the local search. As previously explained in Section 4.2.1 while calibrating i_{\max} , the distribution of the nodes in **r11323** makes it a challenging instance to solve, similarly for **pr439**. These two instances also reported the least number of cases where GRASP was able to find better solutions multiple times. It can be then concluded that the distribution of the nodes highly affects the performance of the A-FVS. The average time increases as the size of the instance increases as well. In conclusion, the evidence shows that the A-FVS algorithm as the local search heuristic can provide a better quality in the solutions of virtually all the tests and that the multi-start approach of GRASP allows the exploration of high-quality solutions in a reasonable time (less than 25 minutes for the largest instance, in average). To the best of our knowledge, **r11889** is the largest instance from the TSPLIB that has been used to solve the ANPCP, specifically splitting it as 1260 users and 629 potential facilities (1889 nodes in total), while **r11323** is the largest in Sánchez-Oro et al. (2022), where all of the 1323 nodes are used as both users and facilities.

CHAPTER 5

CONCLUSIONS

We have GRASP metaheuristic to solve the discrete version of the α -neighbor p -center problem (ANPCP), which consists of selecting p centers out of m facilities such that the maximum distance from any user to its α th center is minimized. This problem is a generalization of the p -center problem (PCP) for which $\alpha = 1$ is implicitly assumed. In real-world applications, the ANPCP aims to have a minimum guaranteed response time between a customer or demand point and its center by considering the notion of providing backup centers in case one of them fails to respond to an emergency, ensuring that even if up to $\alpha - 1$ facilities fail, every customer has a functioning facility close to it.

We proposed a metaheuristic framework that implements GRASP, using a constructive heuristic that uses the objective function of the p -dispersion problem (PDP) as a surrogate function because the original objective of the ANPCP is computationally expensive to calculate during the construction phase. This algorithm was then adapted to use a value-based restricted candidate list (RCL) to be coupled within GRASP, and identified as the Randomized Greedy Dispersion (RGD). We reduced the time complexity of the original algorithm by introducing an array that stores the distances between the facilities outside of the solution and their closest center. The constructed solution is then improved by a vertex substitution or interchange local search heuristic. We designed an algorithm specifically for the ANPCP

by adapting the concept of the *fast interchange*, naming it the Alpha Fast Vertex Substitution (A-FVS). This algorithm is significantly faster due to the use of data structures that store how the assignments between users and centers behave after a swap of facilities, allowing to reuse these expensive computations by accessing them instead of recalculating them. Results of computational experiments on this meta-heuristic approach were reported and analyzed, proving its efficient performance to solve the ANPCP.

5.1 FUTURE WORK

An idea for future research that is worthwhile exploring could be to develop a fast construction heuristic that follows similar ideas from A-FVS, such as the use of specific data structures since the construction algorithm used in this work does not evaluate the objective function of the ANPCP. Another interesting idea could be to develop a Tabu Search version of the A-FVS to exploit the local search and increase diversification.

A natural area for future work is the use of an exact method to obtain lower bounds and compare them with our proposed heuristic. The ANPCP is a very computationally demanding problem and so finding optimal solutions is difficult, but one can attempt to get at least lower bounds on the objective function value that could give an estimate of the quality of the solutions delivered by the heuristic.

Finally, an extension of this work would be to compare our proposed heuristic with that from Sánchez-Oro et al. (2022), where the authors developed a GRASP metaheuristic improved by Strategic Oscillation and a Tabu Search; however, they consider only one set of nodes for both users and facilities, while we considered two different sets. To fix this for our algorithms, users and facilities can be forced to share the same set of nodes, with an extra restriction that ensures that once a node is chosen as a center, it cannot longer be considered as a user. This would be

necessary since the algorithms presented in this work assume that any node is either a user or a potential facility, never both, and if this assumption is maintained when solving an instance that considers only one set of nodes, then the minimum distance would always be 0 because the nearest node of any node would be itself.

APPENDIX A

DETAILED GRASP FINAL RESULTS

Tables A.1 to A.6 show the results of running GRASP($\beta = 0.2$, $i_{\max} = 100$) once on each of the instances from the selected subset. The first group of columns, from left to right, consists of the identifier or index of the variation of the split instance (sub-column v) and the parameters of the solver (sub-columns p and α , respectively). The next group, $x(S)$, includes the objective function value of the best solution of all iterations given by the constructive heuristic (sub-column RGD), the objective function value of the best improved solution of all iterations given by the local search heuristic (sub-column A-FVS), and the relative improvement of the local search solution (sub-column Diff. (%)). Note that both objective function values can be independent of each other: the best A-FVS could have happened in a different iteration than the best RGD. This group gives insight into how much the quality of the solutions improves when the A-FVS algorithm is used as the local search phase of GRASP. The third group, A-FVS Imp. (%), consists of the minimum, maximum, and average (sub-columns Min., Max., and Avg., respectively) relative improvement of the solution yielded by the local search, calculated from all the iterations. The last group, UPC, consists of the minimum, maximum, and average (sub-columns Min., Max., and Avg., respectively) assigned users per center in the best solution found in the experiment. The next column, Time (s), is the computing time in seconds required to run GRASP with a limit of 1800 seconds (30 minutes), and the

last column, i_l , is the last iteration that resulted in a better solution within GRASP.

In some of the individual executions, the best improved solution was not better than the best constructed solution ($x(S)$ Diff. (%) = 0). At the same time, in all of these cases, there were no improvements in 100 iterations ($i_l = 0$). This means that in some instances with some parameters there is a local optimum which is easy to arrive at but difficult to escape from.

Note that several solutions reported a minimum of zero users assigned per center (UPC Min. = 0), meaning that there was at least one center that was not serving any user. We refer to this type of center as an *empty center*. From a decision-making perspective, the fact that the solution includes empty centers does not make sense: in that case, it would be better to simply remove them from the solution. However, according to the mathematical model, S must contain exactly p centers, and also there is no constraint about a minimum number of users to serve. The decision to include or not a facility in the solution is made solely on the objective function, which cares only about the distances between users and facilities. Therefore, a solution with empty centers is still feasible under this mathematical model.

TABLE A.1: Results for pr439_293_146

v	p	α	$x(S)$			A-FVS Imp. (%)			UPC			Time (s)	i_l
			RGD	A-FVS	Diff. (%)	Min.	Max.	Avg.	Min.	Max.	Avg.		
0	7	2	4621	3868	16.30	10.08	47.30	28.78	19	81	41.86	38.42	5
		3	5930	5433	8.38	0	33.04	19.04	6	102	41.86	45.96	13
	14	2	3222	2719	15.61	0	38.60	19.24	1	70	20.93	15.14	7
		3	4090	3401	16.85	5.43	38.06	16.98	1	74	20.93	27.93	45
	21	2	2620	2620	0	0	40.47	14	0	70	13.95	7.55	0
		3	3278	3222	1.71	0	27.24	15.88	0	48	13.95	10.14	0
1	7	2	4646	4204	9.51	2.63	39.25	23.31	6	123	41.86	28.98	4
		3	5942	5205	12.40	1.73	42.52	21.57	9	102	41.86	76.58	1
	14	2	3506	2794	20.31	12.87	43.96	27.41	0	80	20.93	23.63	3
		3	4204	3630	13.65	4.48	33.06	23.40	2	52	20.93	29.70	1

Continues in next page.

v	p	α	$x(S)$			A-FVS Imp. (%)			UPC			Time (s)	i_l
			RGD	A-FVS	Diff. (%)	Min.	Max.	Avg.	Min.	Max.	Avg.		
	21	2	3506	2714	22.59	22.59	43.60	26.39	0	53	13.95	13.22	0
		3	4204	3506	16.60	16.60	29.82	20.93	0	44	13.95	15.10	0
2	7	2	4500	4201	6.64	2.27	38.98	25.02	0	153	41.86	33.98	2
		3	6026	5330	11.55	10.79	36.89	22.53	11	95	41.86	63.27	5
	14	2	3316	2876	13.27	6.06	38.25	23.22	1	102	20.93	22.25	12
		3	3951	3586	9.24	3.23	31	22.24	2	51	20.93	29.76	0
	21	2	3115	2571	17.46	17.46	38.09	24.79	1	67	13.95	12.96	0
		3	3483	2987	14.24	11.43	35.89	22.07	0	75	13.95	30.70	10
	7	2	4646	4201	9.58	7.53	42.70	22.64	10	74	41.86	30.69	0
		3	6275	5381	14.25	7.56	39.86	24.84	2	146	41.86	65.45	2
3	14	2	3265	2774	15.04	0.83	35.33	18.59	0	99	20.93	28.62	64
		3	4153	3595	13.44	0	33.23	20.15	1	69	20.93	53.85	95
	21	2	2841	2277	19.85	15.49	40.98	28.24	1	39	13.95	17.88	1
		3	4153	2815	32.22	7.10	43.35	31.03	0	84	13.95	32.85	11
	7	2	4750	4226	11.03	10.17	39.43	24.55	10	117	41.86	29.21	1
		3	6110	5155	15.63	1.73	41.33	21.39	15	94	41.86	98.11	46
4	14	2	3757	3450	8.17	8.17	31.87	21.11	2	78	20.93	13.60	0
		3	4201	3757	10.57	10.57	35.24	24.59	1	50	20.93	25.72	1
	21	2	3745	3450	7.88	7.88	28.02	15.95	1	42	13.95	8.95	0
		3	4201	3499	16.71	16.71	31.98	22.82	2	48	13.95	20.94	0
	7	2	4652	4250	8.64	7.35	36.91	22.76	0	111	41.86	38.56	1
		3	6122	5087	16.91	1.40	40.63	25.51	12	110	41.86	90.66	0
5	14	2	4250	2719	36.02	0	45.58	34.90	0	77	20.93	26.86	6
		3	4386	4250	3.10	3.10	29.31	14.19	3	48	20.93	21.36	0
	21	2	4250	2719	36.02	36.02	45.58	39.09	0	60	13.95	13.38	0
		3	4310	4250	1.39	1.39	14.93	11.12	0	57	13.95	15.74	0
	7	2	4662	4025	13.66	4.16	42.47	24.12	3	87	41.86	36.70	0
		3	6100	5064	16.98	6.86	42.52	26.30	2	143	41.86	65.43	2
6	14	2	3552	3115	12.30	12.30	38.77	25.28	1	56	20.93	14.79	0
		3	4265	3588	15.87	2.64	41.16	24.16	1	100	20.93	25.38	14
	21	2	3390	3115	8.11	8.11	29.27	16.70	1	38	13.95	6.48	0
		3	3839	3261	15.06	11.85	34.45	24.06	1	51	13.95	16.61	0
	7	2	4567	4201	8.01	8.01	40.65	25.12	0	100	41.86	32.58	2
		3	6110	5235	14.32	12.58	39.52	25.71	8	179	41.86	80.86	0
7	14	2	3399	3070	9.68	9.68	40.68	20.79	0	61	20.93	14.41	0
		3	4296	3650	15.04	13.32	47.03	23.01	2	59	20.93	27.83	6
	21	2	3070	3070	0	0	18.29	10.75	0	50	13.95	5.67	0
		3	4201	3316	21.07	21.07	36.55	25.04	0	49	13.95	25.31	0
	7	2	4688	4004	14.59	2.72	37.45	23.53	0	147	41.86	34.21	0
		3	6042	5256	13.01	0	36.68	20.85	7	166	41.86	65.24	1

v	p	α	$x(S)$			A-FVS Imp. (%)			UPC			Time (s)	i_l
			RGD	A-FVS	Diff. (%)	Min.	Max.	Avg.	Min.	Max.	Avg.		
	14	2	3221	2642	17.98	4.56	38.27	18.35	1	75	20.93	14.61	0
		3	4049	3483	13.98	1.11	33.12	20.05	2	54	20.93	28.72	3
	21	2	2498	2452	1.84	1.84	33.77	19.27	0	76	13.95	9.43	0
		3	3070	3070	0	0	37.99	21.72	0	39	13.95	13.22	0
	7	2	4702	4339	7.72	7.39	36.77	22.22	9	83	41.86	33.84	2
		3	6108	5318	12.93	1.28	32.52	17.69	8	105	41.86	47.71	2
9	14	2	3272	2735	16.41	8.13	43.13	22.69	0	91	20.93	23.49	10
		3	4311	3473	19.44	4.56	41.18	26.94	0	89	20.93	40.94	8
	21	2	2700	2095	22.41	12.04	42.75	28.17	1	55	13.95	27.43	22
		3	3483	2735	21.48	17.02	45.68	28.89	0	69	13.95	32.70	13

TABLE A.2: Results for rat575_384_191

v	p	α	$x(S)$			A-FVS Imp. (%)			UPC			Time (s)	i_l
			RGD	A-FVS	Diff. (%)	Min.	Max.	Avg.	Min.	Max.	Avg.		
0	9	2	150	131	12.67	0	41.33	19.64	25	55	42.67	50.66	26
		3	195	161	17.44	0	37.55	14.63	19	72	42.67	55.11	27
	19	2	103	81	21.36	4.85	44.97	27.87	13	34	20.21	56.57	42
		3	122	105	13.93	0	42.63	17.68	4	40	20.21	40.45	40
	28	2	81	66	18.52	0	47.29	27.26	4	22	13.71	50.50	59
		3	102	82	19.61	0	44.52	24.73	3	22	13.71	64.38	85
1	9	2	146	127	13.01	0	41.10	19.69	26	69	42.67	48.60	14
		3	195	158	18.97	0	35.08	16.80	12	98	42.67	56.09	6
	19	2	93	85	8.60	0	40.82	21.64	7	35	20.21	45.23	56
		3	126	105	16.67	0	37.50	16.96	11	30	20.21	48.10	38
	28	2	77	67	12.99	4.44	38.53	20.79	6	24	13.71	37.55	15
		3	96	82	14.58	1.83	35.43	20.93	4	22	13.71	33.71	22
2	9	2	148	129	12.84	0	40.81	20.29	23	62	42.67	52.48	27
		3	193	162	16.06	0	36.47	17.39	7	95	42.67	70.14	25
	19	2	102	83	18.63	0	37.76	18.12	8	35	20.21	28.48	3
		3	122	106	13.11	0	34.16	15.88	9	38	20.21	32.33	0
	28	2	78	66	15.38	2.41	45.90	24.72	7	23	13.71	51.28	86
		3	98	83	15.31	0	36.55	17.15	8	23	13.71	41.48	50
3	9	2	145	129	11.03	1.18	39.17	20.89	29	73	42.67	39.53	5
		3	195	157	19.49	0	33.47	18.15	21	61	42.67	73.16	22
	19	2	95	82	13.68	0	39.47	20.37	11	30	20.21	54.49	86

Continues in next page.

v	p	α	$x(S)$			A-FVS Imp. (%)			UPC			Time (s)	i_l
			RGD	A-FVS	Diff. (%)	Min.	Max.	Avg.	Min.	Max.	Avg.		
	28	3	123	106	13.82	0	35.26	16.22	11	39	20.21	34.50	9
		2	75	66	12	0	43.65	20.08	5	23	13.71	39.19	37
		3	99	83	16.16	0	35.71	18.33	4	31	13.71	34.62	19
4	9	2	140	131	6.43	0	37.50	18.55	26	57	42.67	49.98	24
		3	193	160	17.10	0	33.74	16.80	24	74	42.67	93.57	56
	19	2	98	80	18.37	0	37.59	20.07	6	28	20.21	50.27	58
		3	120	108	10	1.40	31.36	16.08	8	39	20.21	30.19	4
	28	2	78	67	14.10	0	39.47	22.81	3	23	13.71	43.98	51
		3	98	86	12.24	0	38.57	17.18	3	24	13.71	30.17	2
5	9	2	150	125	16.67	0	37.38	19.06	27	82	42.67	48.33	5
		3	188	157	16.49	0.48	32.96	15.96	5	79	42.67	123.22	108
	19	2	97	80	17.53	0	41.26	18.62	12	34	20.21	53.11	56
		3	122	105	13.93	0	34.73	16.55	8	34	20.21	85.58	151
	28	2	78	65	16.67	2.56	43.31	21.50	7	22	13.71	31.05	5
		3	98	82	16.33	0	42.18	16.12	6	25	13.71	38.65	37
6	9	2	148	124	16.22	1.24	38.57	20.86	26	60	42.67	49.95	3
		3	194	160	17.53	0	33.07	14.57	10	71	42.67	99.75	80
	19	2	99	81	18.18	0	42.76	19.88	4	35	20.21	43.25	29
		3	119	106	10.92	0	38.20	18.11	5	36	20.21	45.73	54
	28	2	81	68	16.05	0	38.94	20.39	5	22	13.71	33.18	12
		3	96	83	13.54	0	37.76	17.74	4	26	13.71	38.64	36
7	9	2	151	129	14.57	1.71	37.50	19.64	25	67	42.67	44.42	9
		3	193	159	17.62	0	38.58	17.91	16	95	42.67	86.68	42
	19	2	96	83	13.54	0	37.84	18.41	6	38	20.21	40.98	39
		3	120	108	10	0	33.51	17.57	5	35	20.21	35.89	5
	28	2	79	65	17.72	0	42.48	20.59	5	21	13.71	32.56	9
		3	98	83	15.31	0	37.68	15.77	3	27	13.71	48.09	86
8	9	2	149	129	13.42	0	41.78	19.27	11	101	42.67	78.17	88
		3	195	156	20	0	37.50	16.31	15	91	42.67	100.55	88
	19	2	94	86	8.51	0.99	37.67	17.05	5	34	20.21	27.91	9
		3	121	106	12.40	0	33.54	17.40	3	33	20.21	50.80	55
	28	2	76	67	11.84	0	39.64	19.61	6	27	13.71	54.10	65
		3	97	87	10.31	0	36.11	14.28	7	31	13.71	23.94	3
9	9	2	142	126	11.27	0	38.99	19.90	17	68	42.67	93.41	133
		3	194	157	19.07	0	33.86	17.16	8	75	42.67	78.82	49
	19	2	94	81	13.83	0	42.25	19.60	9	32	20.21	46.81	49
		3	120	106	11.67	0	34.94	17.82	11	30	20.21	55.78	77
	28	2	76	67	11.84	0	36.80	17.74	3	23	13.71	42.86	59
		3	92	82	10.87	0	37.96	16.58	2	30	13.71	42.24	45

TABLE A.3: Results for rat783_522_261

v	p	α	$x(S)$			A-FVS Imp. (%)			UPC			Time (s)	i_l
			RGD	A-FVS	Diff. (%)	Min.	Max.	Avg.	Min.	Max.	Avg.		
0	13	2	144	121	15.97	0	37.25	19.90	13	65	40.15	87.36	8
		3	181	161	11.05	0	34.85	15.71	10	77	40.15	107.38	52
	26	2	100	84	16	0	42.76	23.83	7	34	20.08	70.38	31
		3	125	103	17.60	0	34.15	19.45	8	36	20.08	65.12	13
	39	2	79	63	20.25	1.15	44.09	21.31	5	21	13.38	63.28	41
		3	90	85	5.56	0	37.86	20.39	6	31	13.38	50.97	14
1	13	2	138	117	15.22	0	33.50	16.15	18	61	40.15	68.44	17
		3	177	160	9.60	0	35.09	15.71	9	82	40.15	70.36	1
	26	2	92	81	11.96	0	45.12	17.84	10	37	20.08	64.52	13
		3	117	102	12.82	0	36.78	16.87	9	35	20.08	59.76	8
	39	2	73	64	12.33	2.47	48.06	21.29	7	21	13.38	74.69	31
		3	97	81	16.49	0	34.68	17.57	2	26	13.38	99.43	80
2	13	2	140	120	14.29	0	39.91	17.48	20	60	40.15	90.44	44
		3	174	159	8.62	0	31.97	16.32	25	57	40.15	96.28	29
	26	2	99	83	16.16	0	42.48	18.34	7	39	20.08	73.61	38
		3	116	104	10.34	0	38.82	16.03	6	36	20.08	102.13	102
	39	2	82	64	21.95	0	39.50	19.65	5	27	13.38	59.51	11
		3	91	83	8.79	0	37.67	16.40	5	23	13.38	111.08	137
3	13	2	141	122	13.48	0	38.07	15.65	15	75	40.15	119.78	105
		3	174	155	10.92	0	32.91	14.35	15	53	40.15	158.30	116
	26	2	93	81	12.90	0.79	39.29	15.91	6	33	20.08	91.47	78
		3	120	103	14.17	0	34.97	14.45	10	43	20.08	96.05	69
	39	2	76	65	14.47	0	46.40	20.63	6	25	13.38	117.28	52
		3	98	81	17.35	0	34.13	15	5	25	13.38	61.84	22
4	13	2	145	120	17.24	1.99	40.38	16.21	25	66	40.15	59.53	0
		3	175	155	11.43	0	35.04	15.55	23	65	40.15	170.26	118
	26	2	95	83	12.63	0.99	40	20.12	6	30	20.08	58.64	2
		3	121	104	14.05	0	34.25	16.02	4	36	20.08	109.85	80
	39	2	76	64	15.79	0	46.28	17.46	5	29	13.38	130.92	115
		3	89	83	6.74	0	39.73	17.50	5	22	13.38	93.24	89
5	13	2	141	117	17.02	0	38.91	18.77	25	59	40.15	113.41	55
		3	175	155	11.43	0	37.25	15.98	12	73	40.15	98.75	29
	26	2	94	83	11.70	0	45.12	18.96	7	31	20.08	75.24	19
		3	122	102	16.39	0	41.76	18.89	10	32	20.08	139.57	85
	39	2	75	64	14.67	0	47.24	20.03	7	24	13.38	116.60	70
		3	93	81	12.90	0.98	40.28	19.51	4	23	13.38	66.24	4
	13	2	139	116	16.55	0	39.09	17.06	19	67	40.15	155.14	118
		3	172	153	11.05	0	37.35	14.70	12	79	40.15	235.03	196

v	p	α	$x(S)$			A-FVS Imp. (%)			UPC			Time (s)	i_l
			RGD	A-FVS	Diff. (%)	Min.	Max.	Avg.	Min.	Max.	Avg.		
	26	2	91	79	13.19	0.97	40.60	19.70	9	34	20.08	70.02	23
		3	118	102	13.56	0	35.40	16.75	4	37	20.08	82.74	37
	39	2	76	63	17.11	0	42.19	18.78	6	22	13.38	61.75	11
		3	93	82	11.83	0	35.77	16.70	4	21	13.38	103.73	93
	13	2	141	119	15.60	0	42.86	19.09	18	62	40.15	243.02	230
		3	180	154	14.44	1.56	32.52	15.40	10	68	40.15	95.97	19
7	26	2	98	81	17.35	1.96	35.92	20.98	6	35	20.08	78.13	33
		3	119	100	15.97	0	37.06	15.14	7	38	20.08	95.69	56
	39	2	76	67	11.84	2.15	44.80	19.84	4	24	13.38	87.87	60
		3	101	83	17.82	0	37.41	19.05	3	23	13.38	112.08	97
	13	2	142	122	14.08	0	41.63	18.23	15	77	40.15	115.64	66
		3	167	159	4.79	0	32.91	17.49	14	100	40.15	113.49	25
8	26	2	94	82	12.77	0	38.26	18.39	5	34	20.08	97.17	69
		3	120	101	15.83	0	39.41	17.89	7	31	20.08	83.99	42
	39	2	76	64	15.79	1.15	37.07	20.55	2	31	13.38	91.16	36
		3	92	81	11.96	0	36.36	17.19	3	30	13.38	138.21	159
	13	2	138	116	15.94	0	39.59	17.54	20	61	40.15	75.82	16
		3	181	154	14.92	0	32.16	14.19	22	72	40.15	82.89	22
9	26	2	95	83	12.63	0	35.42	16.61	8	34	20.08	67.85	32
		3	116	106	8.62	0	35.76	12.86	8	35	20.08	61.38	29
	39	2	76	64	15.79	0	36.79	16.59	5	23	13.38	65.72	29
		3	93	82	11.83	0	34.62	14.69	6	28	13.38	94.38	110
	13	2	138	116	15.94	0	39.59	17.54	20	61	40.15	75.82	16
		3	181	154	14.92	0	32.16	14.19	22	72	40.15	82.89	22

TABLE A.4: Results for dsj1000_667_333

v	p	α	$x(S)$			A-FVS Imp. (%)			UPC			Time (s)	i_l
			RGD	A-FVS	Diff. (%)	Min.	Max.	Avg.	Min.	Max.	Avg.		
0	16	2	308261	240066	22.12	14.26	53.43	33.41	12	120	41.69	141.38	12
		3	392892	301403	23.29	2.99	45.40	29.02	6	90	41.69	244.75	80
	33	2	194691	135457	30.42	16.12	61.52	40.96	6	52	20.21	279.35	92
		3	276399	195519	29.26	19.50	51.58	41.36	0	52	20.21	297.10	28
	49	2	162719	119988	26.26	20.82	57.20	37.46	0	27	13.61	222.11	16
		3	198560	137900	30.55	24.97	57.89	40.17	1	31	13.61	402.99	96
1	16	2	319477	232986	27.07	16	55.88	40.49	5	68	41.69	200.95	2
		3	424921	341431	19.65	16.95	43.28	33.87	2	101	41.69	155.73	4
	33	2	208811	138022	33.90	24.95	59.71	42.19	4	41	20.21	353.30	56

Continues in next page.

v	p	α	$x(S)$			A-FVS Imp. (%)			UPC			Time (s)	i_l
			RGD	A-FVS	Diff. (%)	Min.	Max.	Avg.	Min.	Max.	Avg.		
	49	3	262976	191871	27.04	15.79	55.97	34	0	69	20.21	232.65	38
		2	170353	129112	24.21	24.21	55.10	37.25	1	30	13.61	142.40	0
		3	213660	140168	34.40	26.07	52.18	39.85	2	33	13.61	263.76	22
2	16	2	304718	236454	22.40	6.89	49.15	30.20	11	116	41.69	197.30	62
		3	421141	319532	24.13	7.30	42.69	25.72	6	88	41.69	128.75	1
	33	2	210481	138022	34.43	22.07	57.67	38.31	4	48	20.21	239.25	51
		3	261446	193633	25.94	19.15	53.82	36.45	4	57	20.21	213.75	4
	49	2	173650	116619	32.84	29.91	54.42	43.23	2	37	13.61	263.42	10
		3	215846	141419	34.48	27.28	54.42	40.56	2	29	13.61	217.39	2
3	16	2	330175	234863	28.87	12.41	54.94	35.15	9	108	41.69	185.68	41
		3	419161	319571	23.76	13.87	45.45	31.64	1	96	41.69	174.21	16
	33	2	195792	142085	27.43	17.05	54.89	37.70	3	48	20.21	278.35	75
		3	257576	191358	25.71	17.56	47.23	33.77	1	58	20.21	262.68	18
	49	2	160673	116711	27.36	24.70	53.72	39.35	2	29	13.61	234.65	4
		3	208439	150629	27.73	19.78	49.39	33.66	0	37	13.61	298.87	48
4	16	2	323605	249358	22.94	12.54	47.59	32.28	6	99	41.69	144.33	3
		3	411597	307951	25.18	13.58	45.47	31.55	9	103	41.69	218.16	2
	33	2	199042	139954	29.69	16.17	59.98	43.79	3	47	20.21	213.13	19
		3	274260	199545	27.24	18.49	50.20	36.36	0	68	20.21	194.34	14
	49	2	156110	126368	19.05	17.32	54	34.09	2	34	13.61	135.58	1
		3	198125	144134	27.25	20.27	55.69	40.11	2	28	13.61	394.83	62
5	16	2	296483	234631	20.86	0	44.87	25.43	9	109	41.69	173.56	50
		3	412850	327486	20.68	12.25	44.99	28.82	2	103	41.69	154.72	8
	33	2	196487	136793	30.38	15.37	54.92	38.77	4	60	20.21	247.54	24
		3	254095	189368	25.47	12.58	50.01	32.25	3	55	20.21	203.34	21
	49	2	153593	118189	23.05	18.33	57.99	36.68	2	40	13.61	259.41	40
		3	197313	139429	29.34	20.42	59.87	36.79	1	34	13.61	223.14	3
6	16	2	307683	235038	23.61	10.40	48.19	29.74	15	109	41.69	153.90	3
		3	406856	310159	23.77	16.98	46.83	30.91	6	117	41.69	175.33	22
	33	2	201424	136984	31.99	18.75	50.39	34.35	7	42	20.21	335.62	81
		3	241554	193710	19.81	6.71	51.30	26.41	0	61	20.21	432.50	135
	49	2	153576	115727	24.65	16.45	47.08	35.44	1	34	13.61	258.04	58
		3	206856	141258	31.71	18.75	44.30	31.05	1	30	13.61	481.95	127
7	16	2	318914	232986	26.94	14.98	55.53	35.17	6	103	41.69	178.03	17
		3	412272	324330	21.33	16.93	45.55	33.26	8	105	41.69	195.15	8
	33	2	187000	139044	25.64	19.56	53.84	36.24	6	60	20.21	429.26	135
		3	254670	193624	23.97	11.82	51.27	28.03	0	68	20.21	259.54	56
	49	2	163942	118490	27.72	26.39	53.05	36.15	2	33	13.61	230.50	8
		3	206567	142574	30.98	22.21	49.60	35.65	0	35	13.61	414.58	76
16		2	324875	237927	26.76	15.60	49.40	32.15	10	81	41.69	128.31	3

Continues in next page.

v	p	α	$x(S)$			A-FVS Imp. (%)			UPC			Time (s)	i_l	
			RGD	A-FVS	Diff. (%)	Min.	Max.	Avg.	Min.	Max.	Avg.			
	33	3	395901	307951	22.22	4.38	49.83	24.16	6	114	41.69	159.80	4	
		2	198236	142012	28.36	18.38	47.84	34.55	4	53	20.21	205.32	5	
		3	237051	185564	21.72	10.99	51.05	30.84	1	81	20.21	358.17	79	
	49	2	164865	114758	30.39	24.62	49.02	35.64	2	30	13.61	203.34	8	
		3	187154	144494	22.79	12.91	47.98	32.91	0	35	13.61	293.84	8	
	16	2	314337	236946	24.62	15.02	51.54	34.25	6	76	41.69	131.75	16	
		3	396710	319234	19.53	7.76	42.07	31.12	11	113	41.69	190.38	25	
	9	33	2	206856	142530	31.10	24.91	54.10	39.12	6	49	20.21	218.08	20
			3	257474	194134	24.60	17.54	48.60	30.85	0	49	20.21	240.96	13
	49	2	159515	114218	28.40	17.72	50.86	35.68	2	34	13.61	322.92	113	
3		206856	143102	30.82	26.02	54.10	38.73	0	36	13.61	241.57	9		

TABLE A.5: Results for r11323_882_441

v	p	α	$x(S)$			A-FVS Imp. (%)			UPC			Time (s)	i_l
			RGD	A-FVS	Diff. (%)	Min.	Max.	Avg.	Min.	Max.	Avg.		
0	22	2	4158	3434	17.41	17.41	41.91	30.59	7	68	40.09	191.08	0
		3	5172	3900	24.59	16.32	41.50	28.72	7	67	40.09	375.52	18
	44	2	3853	3434	10.87	10.87	35.60	20.70	0	45	20.05	55.74	0
		3	4162	3547	14.78	14.78	38.86	26.23	0	43	20.05	97.73	0
	66	2	3715	3434	7.56	7.56	26.29	15.91	0	30	13.36	45.46	0
		3	3908	3547	9.24	9.24	28.89	19.38	0	35	13.36	66.94	0
1	22	2	3597	3124	13.15	12.16	43.43	29.84	10	71	40.09	324.25	0
		3	4785	3775	21.11	10.11	41.36	25.55	12	62	40.09	509.68	70
	44	2	3124	3124	0	0	28.81	13.92	2	41	20.05	43.73	0
		3	3853	3365	12.67	12.67	31.54	22.21	3	48	20.05	99.28	0
	66	2	3124	3124	0	0	22.96	7.90	1	35	13.36	28.72	0
		3	3620	3365	7.04	7.04	25.40	14.95	0	35	13.36	61.12	0
2	22	2	3667	2992	18.41	10.42	48.92	28.99	7	82	40.09	1342.66	245
		3	4445	3953	11.07	7.81	38.01	23.33	14	88	40.09	330.50	1
	44	2	2752	2752	0	0	34.86	12.03	0	38	20.05	50.47	0
		3	3275	2870	12.37	11.47	46.35	26.38	0	38	20.05	232.62	0
	66	2	2752	2752	0	0	19.18	3.29	0	30	13.36	24.01	0
		3	2870	2870	0	0	28.20	13.30	0	33	13.36	60.21	0
	22	2	4068	3475	14.58	12.67	37.65	26.55	15	75	40.09	205.39	1
		3	4763	3918	17.74	5.84	40.91	25.98	10	67	40.09	618.10	95

v	p	α	$x(S)$			A-FVS Imp. (%)			UPC			Time (s)	i_l
			RGD	A-FVS	Diff. (%)	Min.	Max.	Avg.	Min.	Max.	Avg.		
	44	2	3918	3475	11.31	11.31	32.01	16.29	5	44	20.05	73.05	0
		3	3983	3550	10.87	10.87	30.73	22.26	7	44	20.05	132.63	0
	66	2	3649	3475	4.77	4.77	27.57	12.80	1	31	13.36	60.71	0
		3	3918	3550	9.39	6.30	26.30	15.26	3	32	13.36	83.92	0
	22	2	3856	3043	21.08	14.68	48.66	31.13	0	76	40.09	358.64	32
		3	4837	3861	20.18	12.03	41.56	27.70	0	87	40.09	473.06	48
4	44	2	2689	2580	4.05	4.05	45.28	25.51	1	44	20.05	74.58	0
		3	3959	3290	16.90	16.90	36.97	23.37	0	49	20.05	103.81	0
	66	2	2580	2580	0	0	29.85	13.28	1	42	13.36	31.23	0
		3	3448	3290	4.58	4.58	31.01	16.60	0	33	13.36	49.17	0
	22	2	3736	3208	14.13	13.46	44.75	29.66	1	70	40.09	340.13	22
		3	4560	3853	15.50	9.85	41.27	26.82	14	75	40.09	390.58	31
5	44	2	3233	2058	36.34	19.83	51.08	38.93	0	50	20.05	880.30	135
		3	3630	3233	10.94	10.94	41.16	26.35	0	45	20.05	104.77	0
	66	2	3233	2022	37.46	37.46	52.60	42.33	0	30	13.36	112.81	0
		3	3630	3233	10.94	10.94	32.43	18.08	0	30	13.36	62.11	0
	22	2	3860	3036	21.35	10.55	46.44	29.14	0	61	40.09	681.22	92
		3	4817	3817	20.76	2.14	44.04	28.16	16	82	40.09	994.99	177
6	44	2	2753	2022	26.55	17.54	47.63	34.73	0	37	20.05	552.51	59
		3	3454	2621	24.12	17.77	47.93	31.78	0	42	20.05	632.61	73
	66	2	2489	2022	18.76	18.76	44.30	31.86	0	30	13.36	100.15	0
		3	3275	2489	24	23.34	44.36	31.79	0	33	13.36	129.34	0
	22	2	4024	3531	12.25	12.25	41.96	26.76	12	75	40.09	146.36	0
		3	4774	3775	20.93	6.37	38.83	25.87	4	60	40.09	387.64	34
7	44	2	3550	3531	0.54	0.54	29.18	15	2	53	20.05	52.26	0
		3	4068	3550	12.73	12.73	31.31	22.85	1	49	20.05	120.73	0
	66	2	3550	3531	0.54	0.54	26.04	11.07	4	32	13.36	41.66	0
		3	3967	3550	10.51	10.51	26.15	17.24	1	36	13.36	83.23	0
	22	2	3729	3054	18.10	3.24	48.06	30.51	12	74	40.09	384.35	6
		3	4751	3749	21.09	9.58	43.56	25.17	1	72	40.09	569.40	102
8	44	2	2954	2948	0.20	0.20	36.95	17.27	7	49	20.05	57.20	0
		3	3657	3124	14.57	14.57	35.88	26.90	2	44	20.05	163.81	0
	66	2	2948	2948	0	0	19.39	8.53	2	29	13.36	28.99	0
		3	3448	3124	9.40	9.40	31.89	17.26	1	36	13.36	69.15	0
	22	2	3852	3069	20.33	11.73	45.66	30.50	1	75	40.09	273.51	2
		3	4679	3738	20.11	10.01	44.23	28.24	14	64	40.09	887.88	158
9	44	2	2948	2085	29.27	19.08	48.25	35.30	0	43	20.05	408.35	12
		3	3481	2948	15.31	15.31	41.70	26.71	0	46	20.05	223.92	0
	66	2	2948	1905	35.38	34.94	51.20	41.22	0	29	13.36	179.07	0
		3	3417	2948	13.73	13.73	27.14	19.73	0	35	13.36	93.86	0

TABLE A.6: Results for r11889_1260_629

v	p	α	$x(S)$			A-FVS Imp. (%)			UPC			Time (s)	i_l
			RGD	A-FVS	Diff. (%)	Min.	Max.	Avg.	Min.	Max.	Avg.		
0	31	2	3058	2530	17.27	0	48.04	22.92	9	78	40.65	1110.82	46
		3	3612	3137	13.15	8.35	43.38	24.68	14	81	40.65	1625.43	78
	62	2	2293	1800	21.50	15.30	38.43	25.66	1	51	20.32	858.60	8
		3	2679	2198	17.95	11.71	43.88	22.85	2	41	20.32	1179	28
	94	2	1821	1376	24.44	11.25	46.36	32.98	2	31	13.40	1770.33	96
		3	2331	1759	24.54	19.07	40.44	29.33	0	34	13.40	1807.71	121
1	31	2	3085	2511	18.61	0	46.15	26.52	10	68	40.65	1501.56	96
		3	3790	3173	16.28	12.20	43.16	26.26	14	73	40.65	929.89	19
	62	2	2192	1767	19.39	8.66	45.12	27.42	1	46	20.32	1547.76	105
		3	2632	2204	16.26	6.49	36.55	22.96	2	48	20.32	1120.83	40
	94	2	1739	1426	18	13.51	47.85	34.73	1	30	13.40	1612.61	95
		3	2366	1767	25.32	9.47	37.47	28.10	0	33	13.40	1806.09	144
2	31	2	3109	2518	19.01	12.67	44.91	24.04	19	63	40.65	828.17	23
		3	3765	3176	15.64	9.23	41.78	26.26	10	70	40.65	1146.66	53
	62	2	2245	1790	20.27	12.96	37.82	26.64	3	46	20.32	1071.89	47
		3	2656	2223	16.30	10.24	48.31	24.51	1	45	20.32	1541.20	93
	94	2	1849	1396	24.50	15.22	48.25	34.64	1	30	13.40	1801.08	116
		3	2295	1758	23.40	6.51	39.44	28.26	0	32	13.40	1804.43	91
3	31	2	3131	2478	20.86	10.64	43.26	26.86	16	80	40.65	975.57	25
		3	3783	3160	16.47	5.47	40.84	27.36	14	84	40.65	1065.21	37
	62	2	2276	1783	21.66	6.75	43.66	26.51	1	49	20.32	1092.30	57
		3	2680	2223	17.05	13.73	42.35	26.36	1	54	20.32	769.92	15
	94	2	1891	1399	26.02	15.38	46.11	35.45	0	35	13.40	1511.96	96
		3	2355	1766	25.01	4.97	37.33	27.07	0	32	13.40	1166.84	43
4	31	2	3160	2563	18.89	6.93	45.98	25.81	21	71	40.65	748.86	4
		3	3664	3149	14.06	11.14	42.60	26.43	13	82	40.65	1806.51	240
	62	2	2256	1796	20.39	9	35.65	26.64	3	41	20.32	875.56	17
		3	2708	2224	17.87	10.76	47.52	26.56	1	41	20.32	1803.60	126
	94	2	1860	1411	24.14	16.88	44.08	33.55	1	38	13.40	1140.19	34
		3	2312	1767	23.57	0	46.06	29.84	0	30	13.40	1670.71	95
5	31	2	3077	2487	19.17	3.72	44.85	25.56	6	75	40.65	1702.53	134
		3	3845	3147	18.15	9.80	40.48	26.35	15	69	40.65	889.77	9
	62	2	2247	1783	20.65	13.19	37.99	26.47	3	41	20.32	1810.34	130
		3	2653	2168	18.28	6.72	38.50	25.50	1	50	20.32	1800.33	111
	94	2	1878	1386	26.20	0	48.20	34.10	3	29	13.40	1466.30	69
		3	2336	1797	23.07	9.30	44.97	27.81	0	40	13.40	1127.89	19
	31	2	3088	2509	18.75	5.33	43.52	25.15	19	71	40.65	1431.65	89
		3	3804	3163	16.85	4.10	37.57	24.54	5	73	40.65	1121.64	20

v	p	α	$x(S)$			A-FVS Imp. (%)			UPC			Time (s)	i_l
			RGD	A-FVS	Diff. (%)	Min.	Max.	Avg.	Min.	Max.	Avg.		
	62	2	2266	1784	21.27	12	38.89	27.03	3	42	20.32	920.72	21
		3	2756	2229	19.12	9.62	43.44	23.18	3	39	20.32	1517.51	88
	94	2	1994	1426	28.49	16.59	45.14	33.69	2	34	13.40	983.04	20
		3	2283	1763	22.78	10.08	41.05	28.71	0	29	13.40	1155.46	30
	31	2	3086	2464	20.16	9.31	43.43	25.38	14	93	40.65	1109.10	53
		3	3848	3122	18.87	8.18	43.78	26.76	11	70	40.65	1806	209
7	62	2	2193	1768	19.38	12.36	44.73	26.96	4	37	20.32	916.31	27
		3	2622	2223	15.22	10.76	36.05	25.06	2	42	20.32	881.17	24
	94	2	1891	1401	25.91	18.35	48.92	33.57	1	34	13.40	1710.89	128
		3	2384	1721	27.81	11.48	44.09	29.02	0	37	13.40	1310.36	48
	31	2	3056	2484	18.72	5.15	43.14	24.93	19	68	40.65	845.65	6
		3	3851	3147	18.28	8.20	43.51	26.69	13	88	40.65	1269.12	80
8	62	2	2223	1780	19.93	13.45	44.08	27.62	0	48	20.32	1808.27	159
		3	2735	2209	19.23	11.44	46.26	27.45	0	49	20.32	740.24	6
	94	2	1763	1391	21.10	10.45	45.15	33.93	1	30	13.40	1699.70	114
		3	2290	1763	23.01	12.45	49.72	29.24	0	29	13.40	1810.69	105
	31	2	3065	2497	18.53	3.51	44.76	25.63	18	73	40.65	1661.05	134
		3	3860	3174	17.77	1.51	43.03	26.44	9	88	40.65	1507.35	88
9	62	2	2173	1797	17.30	10.08	36.44	26.92	2	48	20.32	929.92	13
		3	2719	2211	18.68	9.75	43.28	27.06	3	45	20.32	1144.25	39
	94	2	1954	1414	27.64	18.42	45.71	34.15	1	34	13.40	1689.41	107
		3	2312	1763	23.75	4.50	47.50	28.87	0	36	13.40	1805.01	136
	31	2	3065	2497	18.53	3.51	44.76	25.63	18	73	40.65	1661.05	134
		3	3860	3174	17.77	1.51	43.03	26.44	9	88	40.65	1507.35	88

BIBLIOGRAPHY

- D. Chen and R. Chen. Optimal algorithms for the α -neighbor p -center problem. *European Journal of Operational Research*, 225(1):36–43, 2013.
- H. Delmaire, J. A. Díaz, E. Fernández, and M. Ortega. Reactive GRASP and tabu search based heuristics for the single source capacitated plant location problem. *INFOR*, 37(3):194–225, 1999.
- Z. Drezner. Heuristic solution methods for two location problems with unreliable facilities. *Journal of the Operational Research Society*, 38(6):509–514, 1987.
- E. Erkut, Y. Ülküsal, and O. Yenicerioğlu. A comparison of p -dispersion heuristics. *Computers & Operations Research*, 21(10):1103–1113, 1994.
- T. A. Feo and M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133, 1995.
- S. Khuller, R. Pless, and Y. J. Sussmann. Fault tolerant k -center problems. *Theoretical Computer Science*, 242(1-2):237–245, 2000.
- J. G. Klincewicz. Avoiding local optima in the p -hub location problem using tabu search and grasp. *Annals of Operations research*, 40(1):283–302, 1992.
- S. O. Krumke. On a generalization of the p -center problem. *Information Processing Letters*, 56(2):67–71, 1995.
- G. Laporte, S. Nickel, and F. Saldanha-da Gama. Introduction to location science.

- In *Location Science*, chapter 1, pages 1–21. Springer, Cham, Switzerland, 2nd edition, 2019.
- A. D. López-Sánchez, J. Sánchez-Oro, and A. G. Hernández-Díaz. GRASP and VNS for solving the p -next center problem. *Computers & Operations Research*, 104: 295–303, 2019.
- H. R. Medal, C. E. Rainwater, E. A. Pohl, and M. D. Rossetti. A bi-objective analysis of the r -all-neighbor p -center problem. *Computers & Industrial Engineering*, 72: 114–128, 2014.
- N. Mladenović, M. Labbé, and P. Hansen. Solving the p -center problem with tabu search and variable neighborhood search. *Networks*, 42(1):48–64, 2003.
- D. R. Quevedo-Orozco and R. Z. Ríos-Mercado. Improving the quality of heuristic solutions for the capacitated vertex p -center problem through iterated greedy local search with variable neighborhood descent. *Computers & Operations Research*, 62:133–144, 2015.
- M. G. C. Resende and J. L. González Velarde. GRASP: Greedy Randomized Adaptive Search Procedures. *Inteligencia Artificial*, 19(1):61–76, 2003.
- J. Sánchez-Oro, A. D. López-Sánchez, A. G. Hernández-Díaz, and A. Duarte. GRASP with strategic oscillation for the α -neighbor p -center problem. *European Journal of Operational Research*, 303(1):143–158, 2022.
- R. A. Whitaker. A fast algorithm for the greedy interchange for large-scale clustering and median location problems. *INFOR*, 21(2):95–108, 1983.