P-HUB CENTER AND ROUTING NETWORK DESIGN PROBLEM AND SOLUTION ALGORITHMS

Abdul Kader KASSOUMEH

Master's Thesis

Department of Computer Engineering

Programme in Computer Software

Supervisor: Assoc. Prof. Dr. Ahmet ARSLAN

(Co-Supervisor: Asst. Prof. Dr. Zühal KARTAL)

Eskişehir

Eskişehir Technical University

Institute of Graduate Programs

August 2021

ABSTRACT

P-HUB CENTER AND ROUTING NETWORK DESIGN PROBLEM AND SOLUTION ALGORITHMS

Abdul Kader KASSOUMEH

Department of Computer Engineering

Programme in Computer Software

Eskişehir Technical University, Institute of Graduate Programs, August 2021

Supervisor: Assoc. Prof. Dr. Ahmet ARSLAN

(Co-Supervisor: Asst. Prof. Dr. Zühal KARTAL)

The p-hub center and routing problem deals with finding the location of hub facilities and allocating the demand nodes to these hub facilities, and forming the routes of the vehicles so as to minimize the maximum distance/time between any origin-destination pair under the assumption that one vehicle is dedicated between each non-hub node and hub node. In this study, we relax this assumption and present a mathematical programming model and combinations of algorithms for solving this challenging problem, namely, the single allocation p-hub center and routing problem. In order to generate initial solutions, we propose random, greedy and efficient mat-heuristic algorithms. Our mat-heuristic algorithm is based on a decomposition scheme in which we divide this hierarchical problem into two parts as the location of the hubs and allocations of non-hub nodes to the hubs; also as the routing part. Furthermore, we combine these initial solution algorithms with Variable Neighborhood Search and Simulated Annealing Algorithms. The algorithms are tested on instances on the Turkish network (TR) and Australia Post (AP) datasets. The results show that our initial solution algorithms combined with the metaheuristic algorithms could provide good solutions for medium and large-size instances within reasonable computing times.

Keywords: Hub Location Problems, p-Hub Center Problem, Single Assignment p-Hub Center and Routing Problem, Single Assignment p-Hub Center and Routing Problem Solution Algorithms.

ÖZET

P-ANA DAĞITIM ÜSSÜ MERKEZ VE ROTALAMA AĞ TASARIMI PROBLEMI VE ÇÖZÜM ALGORITMALARI

Abdul Kader KASSOUMEH

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Yazılımı Bilim Dalı

Eskişehir Teknik Üniversitesi, Lisansüstü Eğitim Enstitüsü, Ağustos 2021

Danışman: Doç. Dr. Ahmet ARSLAN

İkinci Danışman: Dr. Öğr. Üyesi Zühal KARTAL

p-ADÜ (Ana Dağıtım Üssü) merkezi ve rotalama problemi, ADÜ tesislerinin yerini bulmak ve bu ADÜ tesislerine talep düğümlerini tahsis etmek ve araçların rotalarını, herhangi bir başlangıç-varış çifti arasındaki maksimum mesafeyi/zamanı en aza indirecek şekilde oluşturmakla ilgilenir. Her bir ADÜ olmayan düğüm ile ADÜ düğümü arasında sadece bir aracın tahsis edildiği varsayılarak yapılmaktadır. Bu çalışmada, bu varsayımı gevşetiyoruz ve bu zorlu problemi, yani tek tahsisli p-ADÜ merkezi ve rotalama problemini çözmek için bir matematiksel programlama modeli ve algoritma kombinasyonları sunuyoruz. İlk çözümleri oluşturmak için rastgele, açgözlü ve mat-sezgisel tabanlı algoritmalar ve bu stratejilerin kombinasyonları kullanılarak oluşturulan algoritmalar öneriyoruz. Ayrıca, bu problemi çözmek için Değişken Komşuluk Arama ve Tavlama Benzetimi algoritmaları geliştirilmiş ve önerilmiştir. Algoritmalar, Türkiye veri seti (TR) ve Australia Post (AP) veri setlerindeki örnekler üzerinde test edilmiştir. Sonuçlar, metasezgisel algoritmalarla ticari çözücülerin birlikte kullanılması neticesinde ortaya çıkan mat-sezgisel algoritmaların, makul hesaplama süreleri içinde orta ve büyük boyutlu örnekler için iyi çözümler sağlayabileceğini göstermektedir.

Anahtar Sözcükler: ADÜ Yer Seçimi Problemi, p-ADÜ Merkez Problemi, Tek Atamalı p-ADÜ Merkez ve Rotalama Problemi, Tek Atamalı p-ADÜ Merkez ve Rotalama Problemi Çözüm Algoritmaları

ACKNOWLEDGEMENTS

I would like to thank my supervisors, Assoc. Prof. Dr. Ahmet Arslan and Asst. Prof. Dr. Zühal Kartal, for all the support and contributions that they provided to me throughout my thesis. I would like to thank them for their efforts in improving my thesis.

Thanks to my friend and mentor, Mr. Nicholas Anthony Ainsworth, for supporting me and reviewing my thesis linguistically.

I would like to thank my dear parents, Muhammed Kassoumeh and Mona Alhallak, my family, relatives, and friends, for believing in me, supporting me in every way and always being by my side.

Abdul Kader KASSOUMEH

05/08/2021

STATEMENT OF COMPLIANCE WITH ETHICAL PRINCIPLES AND RULES

I hereby truthfully declare that this thesis is an original work prepared by me; that I have behaved in accordance with the scientific ethical principles and rules throughout the stages of preparation, data collection, analysis and presentation of my work; that I have cited the sources of all the data and information that could be obtained within the scope of this study, and included these sources in the references section; and that this study has been scanned for plagiarism with "scientific plagiarism detection program" used by Eskişehir Technical University, and that "it does not have any plagiarism" whatsoever. I also declare that, if a case contrary to my declaration is detected in my work at any time, I hereby express my consent to all the ethical and legal consequences that are involved.

Abdul Kader KASSOUMEH

CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

GLOSSARY OF SYMBOLS AND ABBREVIATIONS

TR	:	Turkish network
CAB	:	Civil Aeronautics Board
PTT	:	Turkey Postal Telegraph Telephone
AP	:	Australia Post
pHCRP	:	p-Hub Center and Routing Problem
USApHCP	:	Uncapacitated Single Allocation p-hub Center Problem
RND	:	Random
GRD	:	Greedy
GRD-RND	:	Greedy-Random
RND-GRD	:	Random-Greedy
GRB	:	Gurobi
GRD-GRB	:	Greedy-Gurobi

1. INTRODUCTION

The "hub location problem" is generally the problem of choosing hub nodes and assigning other non-hub nodes to hubs. A general hub location problem concerns a network containing 'n' nodes.

Many-to-many transportation networks rely on hub facilities. Telecommunication systems, logistics systems, airline companies, postal companies and even social networks are among the most important application areas of hub location problems. Today, hubs are used in many areas such as maritime transport, cargo companies, public transport systems, and information distribution systems, and the advantages of hubs are utilized. In addition, other application areas of hub location problems are emergency services and humanitarian aid logistics.

Hubs are special facilities that serve as collection, consolidation, and distribution points on the network. The hubs in many-to-many transportation networks lower the overall transport cost by routing the flow between demand points. For example, in postal delivery networks, mail is picked up from demand points (postal codes), sorted and gathered at hubs (mail sorting centers), and then transferred in larger trucks to different hubs (mail sorting centers) before being distributed to the destination postal codes. By accumulating volume efficiencies, the hubs dramatically reduce costs. Thus, hub-based systems generally help organize flow between demand nodes and consolidate and sort them at hubs; they can then move the flow between hubs and distribute them to the destination points for a lower cost per unit flow. The main goal of the hub location problem is to determine the locations of hub facilities and assign demand nodes to hubs. As a result, hub systems conduct lower costs than directly serving each origin-destination node.

Figure *1.1* shows connections that provide inter-node flow in a non-hub network and connections that provide inter-node flow in a hub network. In (a), the number of connections is 10. In (b), 'k' and 'm' points are chosen as hubs. Connections are in the form of 3 hub-non-hub connections and 1 hub-hub connection. On the other hand, the path between the origin-destination pairs is longer. For example, if there is any flow between 'i' and 'j' in **Figure *1.1*** (a), the path is short since there is a direct connection

between 'i' and 'j', while in **Figure *1.1*** (b), the $i - k - m - j$ path will be followed over the hub connections and the path will be quite long.

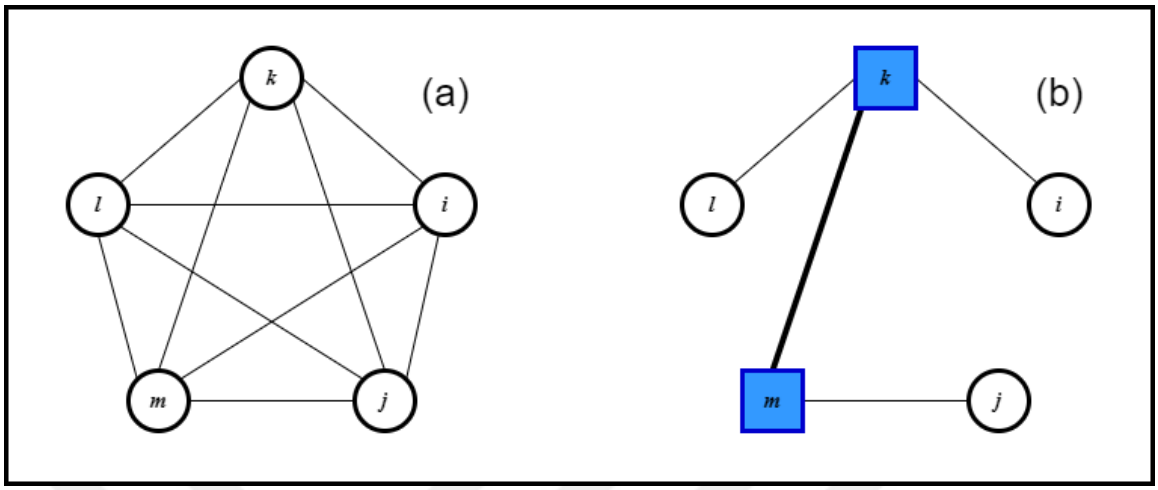

Figure 1.1: *a) inter-node connection in a non-hub network, b) inter-node connection in a hub network*

As seen in **Figure *1.1***, choosing some points as hubs and providing the transfer between the origin-destination pair through the selected hub points significantly reduces the number of connections. For example, in an unsymmetrical network with 'n' nodes, the total number of connections without a hub is $n * (n - 1)$. If one hub is selected and all other points are connected to this selected hub, the number of connections will be only $2 * (n - 1)$. If 'p' hubs are desired to be selected, there are as many alternatives as $\binom{n}{p}$ combinations.

As seen in **Figure *1.1*** (b), 2 out of 5 nodes ('k' and 'm') were selected as hubs, and other nodes were connected to these two hubs. For example, if there is a flow from node 'i' to node 'j', the route of this flow will be $i - k - m - j$. If there is flow from more than one starting point ('i' and 'l') to destination 'j', then these flows will be forwarded to node 'j' after being subjected to load collection at hub 'k' or 'm'.

In the following section, the literature research of hub location problems, and the hub location and routing problems are included. The third section, "Single Assignment p-hub Center and Routing Problem", includes partial mathematical models of the problem. In the fourth section, "Simulated Annealing Algorithm", the algorithm that is used in building the metaheuristic is explained. In the fifth section, "Variable Neighborhood Search Algorithm", the algorithm which is utilized separately in building

the metaheuristic is demonstrated with its components and variants. The sixth section, “Initial Solution Generation and a Mat-heuristic” includes six different initial solution strategies with a mat-heuristic algorithm. In the seventh section, parameters tuning and numerical results of the problem are presented. In the final section, conclusions and recommendations are given.

2. LITERATURE REVIEW

Hubs are facilities that serve as a consolidation, sorting, and transfer point in a many-to-many transportation network such as airlines, cargo distribution services, and telecommunications. The aim of hub location problems is to determine the locations of the hubs and to assign non-hub nodes to hubs in order to minimize total costs. Transfers between hubs can be done via larger vehicles. This reduces the unit transportation costs thanks to the economy of scale.

The first paper with a concept that is very close to the hub location problem was published by Hakimi (1964). In the problem, the idea of locating a node as a hub to ensure the flow between demand points is studied. It is seen that the problem of opening 'p' number of hubs was examined by the same author in the following year to meet the demands (Hakimi, 1965).

Toh and Higgins (1985) presented the study that first introduced the concept of the hub location problem in the airline industry. However, the first mathematical model of the hub location problem in the literature was presented by O'Kelly (1986a). In this study, O'Kelly modeled the hub location problems for one and two hubs.

O'Kelly (1986b) showed that for each origin-destination pair, flows through hubs to be established would result in a smaller number of connections, and the author worked by taking into account the cost parameters required to create a hub. In addition, this study showed that as a result of minimizing transportation costs, each node might not be assigned to the nearest hub.

O'Kelly (1987) discussed airline passenger networks as the p-hub median problem. He presented the first mathematical quadratic model for the solution of the problem. From the given 'n' origin-destination points, 'p' hubs should be selected, and these hubs minimize the total transportation cost.

Each demand point may not give the best solution by assigning it to the nearest hub. Therefore, Aykin (1990) discussed the flow-based hub location problem. Klincewicz (1991), on the other hand, generalized the model a little more by addressing the hub location problem, which considers flow and distance.

In the following years, Campbell (1994b; 1996) presented models for the multi-assignment version of the p-hub location problem, which has a similar objective function.

There are 3 datasets in the literature. O'Kelly (1987) evaluated the airline passenger transportation between 25 cities in the USA and created a dataset called the Civil Aeronautics Board (CAB for short). The CAB has formed the data of many hub location problems and research investigated in the following years. Another frequently used dataset is the Australian Postal (AP) dataset, first used by Ernst and Krishnamoorty (1996). AP is a dataset of nodes with 200 mail destinations in Sydney, and it also includes the amount of flow between nodes. The feature that makes CAB and AP datasets different from each other is that the flow quantities of the CAB dataset are symmetrical, but the flow quantities of the AP dataset are not symmetrical. The last dataset is the Tan and Kara (2007) dataset that is used for Turkey. The dataset is shown as 81 provinces from demand centers in Turkey. The actual data for the provinces from cargo flows were obtained for Turkey Postal Telegraph and Telephone (PTT), which has been benefiting from the average value of the datasets. All the three explained datasets are included in the Operations Research Library (Beasley, 1990).

Campbell (1994a) discusses the hub location problem as a comprehensive study for telecommunications, computer, and transportation systems. O'Kelly and Miller (1994) studied the real problems, slightly relaxing the assumptions of the standard hub location problem. Klincewicz (1998) conducted a study covering telecommunications, computer, and transportation systems in the hub location problem. O'Kelly (1998) examined airline passenger and airline cargo transportation and examined some different features between them. Bryan and O'Kelly (1999) examined and analyzed the problems studied on different hub networks in air transport. In recent studies, Alumur and Kara (2008) analyzed the hub location problems and divided them into 4 classes; p-hub median, fixed cost hub location, p-hub center and hub coverage problem. Farahani et al. (2013) included the models of the problems and their applications and classifications in their study.

The p-hub center problem is a bottleneck (minimum-maximum) type problem, just like the p-center problem. The p-hub center problem is the problem of which nodes the 'p' number of hubs to be opened will be assigned to, and the maximum distances of demand centers to these hubs are minimized. In the problem dealt with in this study,

literature studies have been included, which generally have a single assignment problem type in order to provide strategic management convenience.

The first study and mathematical model on the single assignment p-hub center problem in the literature was studied by Campbell (1994b). Campbell (1994b) defined the p-hub center problem in three different ways in his study:

1) Minimization of transportation cost between any origin and destination;

2) Minimization of the maximum cost of flow in any single connection (from an origin point to hub, from hub to hub, from hub to destination);

3) Minimization of the maximum movement cost of the flow between a hub and the origin/destination point.

Campbell (1994b) worked on mathematical models of single-assignment and multi-assignment versions of three types of p-hub center problems in his study.

Kara and Tansel (2000) presented various mathematical models for the single assignment p-hub center problem. In the study, Campbell (1994b) proposed new approaches for the linearization of the first type's model and presented a new model. In addition, they proved that the p-hub center problem is NP-difficult.

Kara and Tansel (2001) added a number of new constraints to the p-hub center problem. They named the problem they dealt with as the latest arrival hub location problem. In the addressed problem, there is the restriction that no aircraft takes off before all of the aircrafts which will bring products to a hub reach that hub. However, Wagner (2004) showed that this problem is not a p-hub center problem, since there is no waiting time constraint in hubs in the problems of minimization of the maximum cost/time in the problem discussed by Kara and Tansel (2001).

Ernst et al. (2002) developed a new formulation for the single assignment p-hub center problem in their study. The same formulation and solution techniques were published in 2009. The authors defined a new variable called 'r_k' (the radius of the hub 'k'), where it must be greater than or equal to the cost of traveling to any node assigned to hub 'k'. They tested the model using the CAB and AP datasets. By comparing the

model proposed by Ernst et al. (2009) to the model proposed by Kara and Tansel (2000), it has been observed that the model proposed by Ernst et al. (2009) gives better results.

Campbell et al. (2007) showed that the solution can be found in polynomial time for some types of single and multi-assignment types of the p-hub center problem.

Meyer et al. (2009) worked on a two-step algorithm to solve the single assignment p-hub center problem. In the first step, they used the branch-and-bound algorithm to find the best hub locations, and in the second step, they determined which hubs the demand centers would be assigned to. They used the ant colony algorithm to obtain a good upper limit.

Yaman and Elloumi (2012) studied a p-hub center problem based on the star network. There is one hub center in the star network and 'p' hubs are connected to this hub center. Demand points are connected to 'p' hubs. They used two mixed integer programming methods for the solution of the discussed problem.

Bashiri et al. (2013) studied the hub location problem by considering qualitative parameters alongside the quantitative parameters. They used a hybrid genetic algorithm to solve the problem using fuzzy systems.

The proposed 4-index basic formulation for the p-hub center problem (type 1) is included in **Model *2.1***.

Decision Variables:

$$x_{ijkm} = \begin{cases} 1, \text{if the flow is from point } i \text{ to point } j \text{ uses } k \text{ and } m \text{ hubs} \\ 0, \text{otherwise} \end{cases}$$

$$y_k = \begin{cases} 1, \text{if } k \text{ is a hub} \\ 0, \text{otherwise} \end{cases}$$

Parameters:

c_{ij} = cost of a unit flow from node i to node j

$$\boldsymbol{c_{ijkm} = c_{ik} + c_{mj} + \alpha\, c_{km}}$$

Model 2.1: *p-hub Center Problem Type-1*

$$Min\ Z = Max\ \{x_{ijkm}\ c_{ijkm}\}, \qquad (1)$$

$$\sum_k y_k = p, \tag{2}$$

$$0 \le y_k \le 1, \ \forall k \in N \tag{3}$$

$$0 \le x_{ijkm} \le 1, \ \forall i \in N, j \in N, k \in N, m \in N \tag{4}$$

$$\sum_k \sum_m x_{ijkm} = 1, \ \forall i \in N, j \in N \tag{5}$$

$$x_{ijkm} \le y_k, \ \forall i \in N, j \in N, k \in N, m \in N \tag{6}$$

$$x_{ijkm} \le y_m, \ \forall i \in N, j \in N, k \in N, m \in N \tag{7}$$

The objective function in the p-hub center formulation minimizes the largest transportation cost value between origin-destination points. Constraint (2) in **Model *2.1*** indicates that the number of hubs will be 'p'. Constraint (3) ensures that the variable 'y_k' is between 0 and 1. Constraint (4) indicates the limitation of the values that the variable 'x_{ijkm}' will take. Constraint (5) ensures that the flow is routed through the hub pair for each origin-destination pair. Constraints (6) and (7) ensure that flows are routed through locations with a hub.

The 2nd basic formulation with 4 indices proposed for the p-hub center problem (type 2) is presented in Model 2.2.

Decision Variables:

x_{ijkm}

$= 1,$ the rate of use of k and m hubs of the flow going from point i to point j

$$y_k = \begin{cases} 1, \text{if } k \text{ is a hub} \\ 0, \text{otherwise} \end{cases}$$

Parameters:

c_{ij} = cost of a unit flow from node i to node j

Model 2.2: *p-hub Center Problem Type-2*

$$Min\ Z = Max\ \{(c_{ik}, c_{mi}, \alpha c_{km}) x_{ijkm}\}, \tag{1}$$

$$\sum_k y_k = p, \tag{2}$$

$$0 \le y_k \le 1, \ \forall k \in N \tag{3}$$

$$0 \leq x_{ijkm} \leq 1, \ \forall i \in N, j \in N, k \in N, m \in N \quad (4)$$

$$\sum_k \sum_m x_{ijkm} = 1, \ \forall i \in N, j \in N \quad (5)$$

$$x_{ijkm} \leq y_k, \ \forall i \in N, j \in N, k \in N, m \in N \quad (6)$$

$$x_{ijkm} \leq y_m, \ \forall i \in N, j \in N, k \in N, m \in N \quad (7)$$

The objective function of Model 2.2 is to minimize the largest cost of flows in a single connection. In addition, the constraints in the model are identical to the constraints in Model 2.1. Both formulations in this section assume that the flow between hubs cannot be 0 ($\alpha \neq 0$). However, the models allow for 0 flow between certain origin-destination points.

The p-hub center problem model is given in Model 2.3 with the 2-index formulation of the variable they call 'r_k' proposed by Ernst et al. (2009).

Decision Variables:

$r_k = \text{radius of hub } k$

$$x_{ik} = \begin{cases} 1, \text{if node } i \text{ is assigned to hub } k \\ 0, \text{otherwise} \end{cases}$$

Parameters:

c_{ik}: distance between node i and k

Model 2.3: *Ernst et al. (2009) p-hub Center Problem*

$$Min\ Z = Max\ (r_k + r_l + \alpha c_{kl}), \ \forall k \in N, l \in N \quad (1)$$

$$\sum_k x_{ik} = 1, \ \forall i \in N \quad (2)$$

$$\sum_k x_{kk} = p, \quad (3)$$

$$x_{ik} \leq x_{kk}, \ \forall i \in N, k \in N \quad (4)$$

$$r_k \geq c_{ik} x_{ik}, \ \forall i \in N, k \in N \quad (5)$$

$$x_{ik} \in \{0,1\}, \ \forall i \in N, k \in N \quad (6)$$

$$r_k \geq 0, \ \forall k \in N \quad (7)$$

The objective function of the model uses the hub radius to minimize the largest cost between any two nodes. Constraint (2) in Model 2.3 ensures that each node is assigned to

only one hub, while Constraint (3) determines the number of hubs to be opened as 'p'. Constraint (4) ensures that assignments are made only to hubs. Constraint (5) ensures that node 'i' is assigned if node 'k' is a hub, and node 'i' is assigned such that it does not exceed the radius of 'k'. Constraint (6) shows that the variable 'x_{ik}' is a 0-1 integer variable, and Constraint (7) shows that the variable 'r_k' is a positive variable.

Decision variables, parameters and 4-index basic formulation for the single assignment p-hub coverage problem are presented in Model 2.4.

Decision Variables:

$$V_{ijkm} = \begin{cases} 1, \text{if } k \text{ and } m \text{ hubs contain origin} - \text{destination pairs} \\ 0, \text{otherwise} \end{cases}$$

$$x_{ijkm} = k \text{ and } m \text{ hubs delivery rate from origin } i \text{ to center } j$$

$$y_k = \begin{cases} 1, \text{if } k \text{ is a hub} \\ 0, \text{otherwise} \end{cases}$$

Parameters:

F_k = cost of installing hub k

Model 2.4: *p-hub Coverage Problem*

$$Min\ Z = \sum_k F_k y_k, \tag{1}$$

$$x_{ijkm} \le y_k, \ \forall\, i \in N, j \in N, k \in N, m \in N \tag{2}$$

$$x_{ijkm} \le y_m, \ \forall\, i \in N, j \in N, k \in N, m \in N \tag{3}$$

$$\sum_k \sum_m V_{ijkm}\, x_{ijkm} \ge 1, \ \forall\, i \in N, j \in N \tag{4}$$

$$0 \le y_k \le 1, \ \forall\, k \in N \tag{5}$$

The objective function of the mathematical model minimizes the cost of opening a hub. Constraints (5), (2) and (3) in Model 2.4 are identical to the constraints in Model 2.1. Constraint (4) ensures that each origin-destination pair is assigned once.

If the fixed cost 'F_k' of opening hubs to node 'k' takes the same value for all hubs, this issue becomes a problem of opening a minimum number of hubs.

If the cost of covering all origin-destination pairs is higher than the specified limit, it can be resolved in two ways. Option 1 is to drop some of the origin-destination pairs, and option 2 is to loosen the set limit value a bit.

The translation of Ernst's proposed model for the p-hub center into the model of the 2-index p-hub coverage problem, decision variables and parameters are included in Model 2.5.

Decision Variables:

r_k = radius of hub k

$$x_{ik} = \begin{cases} 1, \text{if node } i \text{ is assigned to hub } k \\ 0, \text{otherwise} \end{cases}$$

Parameters:

c_{ik}: distance between node i and k

F_k = cost of installing hub k

β = Coverage distance

Model 2.5: *Ernst et al. (2009) to p-hub Coverage Problem*

$$Min\ Z = \sum_k F_k x_{kk} \quad (1)$$

$$\sum_k x_{ik} = 1, \ \forall\, i \in N \quad (2)$$

$$\sum_k x_{kk} = p, \quad (3)$$

$$x_{ik} \le x_{kk}, \ \forall\, i \in N, k \in N \quad (4)$$

$$r_k \ge c_{ik} x_{ik}, \ \forall\, i \in N, k \in N \quad (5)$$

$$r_k + r_m + \alpha c_{km} \le \beta, \ \forall\, k \in N, m \in N \quad (6)$$

$$x_{ik} \in \{0,1\}, \ \forall\, i \in N, k \in N \quad (7)$$

$$r_k \ge 0, \ \forall\, k \in N \quad (8)$$

The objective function of this mathematical model ensures that the cost of opening a hub is minimal. Constraints (2)-(8) in Model 2.5, except Constraint (6), are identical to the constraints in Model 2.3. Constraint (6) is the coverage constraint.

Hub location problems require that each pair of hubs has a direct hub connection to each other. Therefore, there are many hub connections in the hub network. Thanks to this flexibility in the delivery of flows, a new area of research has arisen. When comparing the hub network design with complete hub network connections and the hub network design that allows lack in hub connections, the service quality in terms of service time/cost is almost the same (Alumur, Kara, & Karasan, 2009).

There are few studies in the literature examining the poorly linked hub location problem. O'Kelly and Miller (1994) proceeded by referencing three critical questions in the design of the hub network in their studies. These questions are as follows:

1) Are the nodes in the network assigned to only one hub?

2) Are connections between nodes allowed to ignore a hub?

3) Are the established hubs completely interconnected?

The authors took into account these questions in their studies and exemplified 8 different network designs.

Nickel et al. (2001) proposed a model about the location problem in urban public transport networks in their study. The authors considered the fixed cost required to establish hub connections in the problem. They have minimized the total transportation costs and the fixed cost required to establish connections.

Podnar et al. (2002) examined a different hub network design problem where lower-cost links were selected. However, they did not examine in which locations the hubs should be placed in the problem of designing a hub network.

Campbell et al. (2005a; 2005b) presented the model they developed for the hub arc location problem in their study. These studies were also carried out by Podnar et al. (2002); they did not aim to locate the discrete hub facilities, but to find hub arcs that reduce the flow costs between hubs. In the study, the model has been tested on examples and their optimal solutions.

Alumur et al. (2012) studied the problem of designing a hub network using different transportation modes in addition to the hub location and allocation decisions in the

literature. In this problem, transportation costs and travel times, which are generally examined separately, are evaluated together. A mixed integer programming model has been developed for solving the problem, and the model was tested on Turkey and CAB datasets.

Serper and Alumur (2016) addressed the problem of designing a hub network in which different transportation models and different vehicle types are taken into account. The problem aims to find out the locations and capacities of hubs, which hubs will be assigned to non-hub nodes, which mode of transportation will be used in hubs and the number of vehicle types to be used in the hub network. A mixed integer model has been developed in the study. For solving the problem, a variable neighbor search algorithm has been proposed. The algorithm was applied to CAB and TR datasets.

It is assumed that there is a direct connection between the demand points and the hubs in many of the hub location problems. Using different vehicles for each hub-demand point link can cause high vehicle costs and traffic congestion. However, through correct vehicle routing, a vehicle visiting a few nodes and going to the hub will reduce vehicle costs and traffic congestion (Kartal, Krishnamoorthy, & Ernst, 2019).

Within the hub location and routing problem, how many hubs will be opened, where the locations of these hubs are, which routes should be followed by the determined vehicles and the minimization of total transportation costs are problematic areas.

There are few studies on this topic in the literature. The first study belongs to Nagy and Salhi (1998). The authors took into account several constraints to establish the hub and to determine the route between origin and destination. They developed a mixed integer model to solve the problem.

Bruns et al. (2000) worked on the problem of redesigning the postal delivery service in Switzerland. The purpose of the problem was to determine the number and location of hubs to be installed, and to which hubs the demand nodes will be assigned.

Wasner and Zapfel (2004) determined the collection and distribution routes of vehicles in a network of cargo distribution companies in Austria. In the study, transportation between hubs is directed through a central hub. The model has considered the locations of the warehouses and hubs, which hub the demand centers will be assigned

to, and the vehicle routes between the warehouses. While establishing the model, it was assumed that there is a direct transportation between warehouses and the policy of delivery and then distribution. The authors proposed a nonlinear mixed integer model for the problem.

Çetiner et al. (2010) worked on the multi-assignment hub location and vehicle routing problem on the TR dataset. In this study, flows are collected and distributed simultaneously.

Camargo et al. (2013) worked on a problem similar to the single assignment hub location and vehicle routing problems. The difference with other studies in the literature is that the number of hubs to be established was determined by the authors. The authors assumed that each hub can only receive service from one vehicle and that there is a limit to the number of hubs a vehicle can visit.

Kartal et al. (2017) studied the single assignment p-hub median and routing problem that includes simultaneous collection and distribution. The aim of the problem is to minimize the transportation cost and the routing costs of the flow in the network and between hubs.

Rieck et al. (2014) worked on the opening of hub facilities and the problem of vehicle routing. A linear model with mixed integers was proposed for solving the problem.

3. SINGLE ASSIGNMENT P-HUB CENTER AND ROUTING PROBLEM

Our mathematical models determine hubs, assign non-hub nodes to hubs, and create the routing structure associated with them. In order to make sure that all non-hub nodes are served, the routes must be constructed so that all non-hub nodes can have their required demand and flow reached through the hubs to which they are assigned. Furthermore, in order to best minimize the maximum cost (distance) between any origin-destination pair, routes will have multiple stopovers.

The model is designed to incorporate separate collection and distribution routes for vehicles in order to determine the maximum cost (distance) between any pair of origin-destination. Vehicles complete their trips in a wide variety of real-world applications by returning via the same route, visiting the same nodes on the way back; the forward route is considered the collection route, while the backward route is considered the distribution route. To provide an example and show the distribution and collection routes, consider the hub node as node 1. Vehicle 1 is assigned to that hub, and the allocated nodes to vehicle 1 are nodes 2, 3, and 4. Assume that the collection route that has the minimum cost (distance) is represented by $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 1$. To put it another way, starting from hub node 1, vehicle 1 makes its way to nodes 2, 3, and 4 in that order then comes back to the hub node 1. As a result, $1 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 1$ is the distribution tour.

3.1. Mathematical Formulation for the Single Allocation Fixed p-Hub Center and Routing Problem

The mathematical formulation of the single allocation fixed p-hub center and routing problem was proposed by Uçar (2020). The problem seeks to allocate non-hub nodes to the predetermined hubs and forms the routes according to the assigned number of vehicles per hub. While forming our mat-heuristic algorithms, we have inspired from this formulation. Therefore, we will be given the decision variables, parameters and constraints of the single assignment fixed location hub center routing problem mathematical model with 4 indices with O(n^3) complexity are included in Model 3.1.

Decision Variables:

$collect_{kij}^{v} =$

$\begin{cases} 1, \text{if the vehicle } v \text{ starts its route from hub } k \text{ visits nodes } i \text{ and } j \text{ respectively} \\ 0, \text{otherwise} \end{cases}$

$distribute_{ijk}^{w} =$

$\begin{cases} 1, \text{if vehicle } w \text{ visits nodes } i \text{ and } j \text{ respectively completes its tour at hub } k \\ 0, \text{otherwise} \end{cases}$

$x_{ij}^{v} = \begin{cases} 1, \text{if vehicle } v \text{ visits nodes } i \text{ and } j \text{ respectively on the route} \\ 0, \text{otherwise} \end{cases}$

$h_i = \begin{cases} 1, \text{if node } i \text{ is a hub} \\ 0, \text{otherwise} \end{cases}$

$z_{iv} = \begin{cases} 1, \text{if } v \text{ vehicle is assigned to hub } i \\ 0, \text{otherwise} \end{cases}$

$collect\_sum_{kv}$ = Maximum collection distance of the route of vehicle *v* starting its route from hub *k*

$distribute\_sum_{iw}$ = Maximum distribution distance of the route of vehicle *w* completing its route at its hub

Parameters:

d_{ij} = distance between nodes *i* and *j*

n_i = number of vehicles assigned to hub *i*

p = number of hubs to open

Model 3.1: *Single Assignment Fixed Location Hub Center Routing Problem*

$$Min\ Z, \tag{1}$$

$$\sum_{i} h_i = p \tag{2}$$

$$\sum_{v} \sum_{j \neq i} x_{ij}^{v} - \sum_{v} z_{iv} = 1 - hi, \ \ \forall\, i \in N \tag{3}$$

$$z_{iv} \leq \sum_{j \neq i, j \in NH} x_{ij}^{v}, \ \ \forall\, i \ \in H, v \in V \tag{4}$$

$$\sum_{i \in H} z_{iv} = 1, \ \ \forall\, v \in V \tag{5}$$

$$\sum_{v} z_{iv} = h_i * n_i, \ \forall i \in H \tag{6}$$

$$\sum_{j \neq i} x_{ij}^{v} - \sum_{j \neq i} x_{ji}^{v} = 0, \ \forall i \in N, v \in V \tag{7}$$

$$\sum_{j \neq i, j \in NH} x_{ij}^{v} - 1 + h_i \leq z_{iv}, \ \forall i \in N, v \in V \tag{8}$$

$$u_i - u_j + L * \sum_{v} x_{ij}^{v} + (L-2) * \sum_{v} x_{ji}^{v} \leq L - 1, \ \forall i \in NH, j \in NH: i \neq j \tag{9}$$

$$collect\_sum_{kv} \geq \sum_{i} \sum_{j} \left(d_{ij} * collect_{kij}^{v}\right), \ \forall k \in N, v \in V \tag{10}$$

$$\sum_{k \in H} collect_{kij}^{v} = x_{ij}^{v}, \ \forall i \in N, j \in NH, v \in V: j \neq i \tag{11}$$

$$collect_{kij}^{v} \leq x_{ij}^{v}, \ \forall i \in N, j \in NH, k \in H, v \in V \tag{12}$$

$$collect_{kij}^{v} \leq z_{kv}, \ \forall i \in N, j \in NH, k \in H, v \in V \tag{13}$$

$$distribute\_sum_{iw} \geq \sum_{i} \sum_{j} \left(d_{ij} * distribute_{ijk}^{w}\right), \ \forall i \in H, w \in V \tag{14}$$

$$\sum_{k \in H} distribute_{ijk}^{w} = x_{ij}^{w}, \ \forall i \in NH, j \in N, w \in V: j \neq i \tag{15}$$

$$distribute_{ijk}^{w} \leq z_{kw}, \ \forall i \in NH, j \in N, k \in H, w \in V \tag{16}$$

$$distribute_{ijk}^{w} \leq x_{ij}^{w}, \ \forall i \in NH, j \in N, k \in H, w \in V: j \neq i \tag{17}$$

$$\sum_{j \neq i} \sum_{v} x_{ij}^{v} = 1 + (n_i - 1) * h_i, \ \forall i \in N \tag{18}$$

$$\sum_{j \neq i} \sum_{v} x_{ji}^{v} = 1 + (n_i - 1) * h_i, \ \forall i \in N \tag{19}$$

$$Z \geq (collect\_sum_{kv} + distribute\_sum_{iw} + \alpha * d_{ik}), \ \forall i \in H, k \in H, v \in V, w \in V: i \neq k, v \neq w \tag{20}$$

$$Z \geq \left(collect_{sum_{kv}} + distribute_{sum_{kw}}\right), \ \forall k \in N, v \in V, w \in V: v \neq w \tag{21}$$

$$Z \geq \sum_{i} \sum_{j \neq i} \left(d_{ij} * x_{ij}^{v}\right) - d_{kl} * x_{kl}^{v} - M * (1 - x_{kl}^{v}), \ \forall k \in N, l \in N, v \in V \tag{22}$$

$$z_{iv}, x_{ij}^{v}, h_i, distribute_{ijk}^{w}, collect_{kij}^{v} \in \{0,1\}, \tag{23}$$

$$u_i, collect\_sum_{kv}, distribute\_sum_{iw}, Z \geq 0, \tag{24}$$

The objective function of the model is to minimize the longest total distance between all nodes 'i' and 'j'. Constraints (20), (21) and (22) in **Model *3.1*** have been developed to ensure this situation in the mathematical model. Objective function value:

the longest distance between two nodes can be equal to the sum of the maximum collection ($collect\_sum$) and maximum distribution ($distribut\_sum$) and the distance between hubs for vehicles assigned to different hubs. This situation is given in Constraint (20). The longest distance between any pair of nodes 'i' and 'j' can also be obtained from two different vehicle routes assigned to the same hub. In this case, this distance is the maximum collection distance plus the maximum distribution distance for two separate vehicles. The realization of the objective function in this way is provided in Constraint (21). In Constraint (22), The situation where the distance between two nodes on the resulting design network can be the farthest is due to a single route. In this constraint, the longest distance is calculated for all routes by subtracting the shortest distance between the nodes of that route from the total length of the route.

Constraint (2) in **Model *3.1*** allows a 'p' number of hubs to be opened. It should be noted here that the model can work without this constraint. Constraint (3) ensures that if node 'i' is a hub, 'v' vehicles are assigned to this hub, and if node 'i' is not a hub, only 1 vehicle visits that node. Constraint (4) ensures that if node 'i' is a hub, vehicle 'v' is assigned to that hub, the vehicle passes from node 'i' to node 'j'. While constraint (5) ensures that each vehicle is assigned to only one hub. Constraint (6) ensures that if any node 'i' is a hub, 'n_i' vehicles are assigned to that node. Constraint (7) is the conservation of flow constraint; It ensures that the number of vehicles arriving and leaving a node is equal. With Constraint (8), it is ensured that the variable 'z_{iv}' takes value only when node 'i' is a hub. Constraint (9) is the sub-tour elimination constraint. The sub-tour blocking constraints were first tried with the classical MTZ (Miller, Tucker, & Zemlin, 1960) constraints, but it was noticed that the current formulation partially accelerated the solution results, so they were included in the mathematical programming model through this version of MTZ. Constraint (10) calculates the maximum collection distance for each route. Constraints (11) and (12) were developed together to describe the relationship between the variable collect and 'x_{ij}^{v}'. Constraint (13) associates the variable collect with the variable 'z_{kv}', which indicates that a 'v' vehicle is assigned to each hub. Constraint (14) calculates the maximum distribution distance for each route. Constraints (15) and (16) were developed together to describe the relationship between $distribute$ (distribution) variable and 'x_{ij}^{v}'. Constraint (17) associates the variable distribute with the

variable 'z_{kv}', which indicates that a 'v' vehicle should be assigned to each hub. Constraints (18) and (19) together ensure that if node 'i' is a hub, the number of links leaving hub 'i' and entering node 'i' is the same as the number of vehicles assigned to that hub. In addition, these constraints ensure that if node 'i' is not a hub, where vehicle 'v' visiting node 'i' will only visit the 'i-j' link once.

3.2. Partial Mathematical Models

The mathematical models which are used in creating mat-heuristic algorithms are given in this section. Ernst et al.'s (2009) mathematical model is given in section 3.2.1, which gives the fastest result for the p-hub center problem in the literature. It is a radius-based formulation. In section 3.2.2, firstly, the objective function of the single depot multi traveling salesman problem is adapted (reduced) to the problem of this study. Then, section 3.2.3 focuses on special cases with one vehicle per hub. After finding the locations of the hubs, greedily or by using the p-hub center radius based formulation, according to the number of vehicles, we use either the formulation in **Model *3.3*** (more than one vehicle per hub case) or **Model *3.4*** (one vehicle case).

3.2.1. p-hub Center Problem formulation

Uncapacitated single allocation p-hub center problem (USApHCP) defined on a complete, symmetric network $G = (N, \mathcal{E})$ with a node-set $N = \{1, 2, \ldots, N\}$ and an arc-set $\mathcal{E}$. Every arc $[i, j]$ has an infinite capacity and cost $c_{ij} = c_{\text{ji}}$ which satisfies the triangle inequality. We also suppose that $c_{ij} \geq 0$ and sometimes $c_{ii} = 0$; however, this is not mandatory. USApHCP requires the complete subnetwork selection of p hubs, each of which has an infinite node capacity for flow collection, transport and distribution (Ernst, Hamacher, Jiang, Krishnamoorthy, & Woeginger, 2009).

The purpose of the p-hub center problem is to find the location of the hubs and allocation of the non-hub nodes to these hubs in order to minimize the maximum distance between any origin-destination pair. In this study, the results of the p-hub center problem were used in finding the hub locations and before routing, in the decision making of the assignments, in other words, the clusterings.

In the following, we formally present the mathematical formulation of the p-hub center problem. The decision variables and parameters used are as follows:

Decision Variables:

$$X_{ik} = \begin{cases} 1, \text{if node } i \text{ is allocated to hub } k \\ 0, \text{otherwise} \end{cases}$$

$$X_{kk} = \begin{cases} 1, \text{if node } k \text{ is a hub} \\ 0, \text{otherwise} \end{cases}$$

Parameters:

z: represents all origin-destination maximum transportation costs

α: the discount number.

Model 3.2: *p-hub Center Problem Formulation*

$$Min\ Z, \tag{1}$$

$$z \geq r_k + r_i + \alpha * d_{ki}, \ \ \forall\ k \in N, i \in N \tag{2}$$

$$\sum_{k \in N} x_{kk} = p, \tag{3}$$

$$\sum_{k \in N} x_{ik} = 1, \forall\ i \in N \tag{4}$$

$$x_{ij} \leq x_{ji}, \forall\ i \in N, j \in N \tag{5}$$

$$r_k \geq \sum_{i \in N} d_{ik} * x_{ik}, \ \ \forall\ k \in N \tag{6}$$

$$\sum_{i \in N} x_{ij} \geq (n_i + 1) * x_{jj}, \ \ \forall\ j \in N \tag{7}$$

$$x_{ij} \in \{0, 1\}, \tag{8}$$

$$r_k > 0 \tag{9}$$

The objective is represented by a free variable Z. USApHCP's objective is achieved by reducing the maximum of the total unit costs between any pair of nodes i and j to the smallest possible amount. Constraints (2)-(5) in **Model *3.2*** are standard. Constraint (6) affirms that the radius of a hub must be greater than or equal to the cost of traveling to any node assigned to the hub, while the radius of a non-hub node may be as small as zero (see above). Constraint (7) ensures that a sufficient number of non-hub nodes are assigned to each hub.

As mentioned above, this problem was exploited in the developed mat-heuristics in finding hub locations and finding non-hub nodes assigned to hubs for routing.

3.2.2. Multiple traveling salesman problem formulation with center objective

The Multiple Traveling Salesman Problem (mTSP) is a general form of the Traveling Salesman Problem (TSP) that allows multiple salesmen to exist. A set of cities is provided, with one depot where there exist m salesmen, and a cost metric. The objective of our problem is to assign each salesman a tour such that the maximum distance between any pair of nodes is minimized and that every city is visited by one salesman only.

The mathematical model presented in this section is based on **Model *3.1***. In (Uçar, 2020), the problem is to minimize the maximum distance between any origin-destination while the number and the location of the hubs are predetermined. There is also n_i number of vehicles which are assigned to each hub. Analogously in this formulation, we consider just one depot (hub) and the vehicles that are assigned to this hub. This formulation is used to minimize the maximum distance between any i-j node pairs in the routes after determining the hub locations and allocations via p-hub center formulation. Furthermore, if there is more than one vehicle assigned to each hub, this formulation is used to obtain an initial solution based on just mathematical model formulations.

Additional decision variables and the mathematical model formulation of mTSP (**Model *3.3***) are provided in the following:

Decision Variables:

u_i: auxiliary variable

$$collect_{kij}^{v} = \begin{cases} 1, \text{if the vehicle } v \text{ starts its route from hub } k \text{ visits nodes } i \text{ and } j \text{ respectively} \\ 0, \text{otherwise} \end{cases}$$

$$distribute_{ijk}^{w} = \begin{cases} 1, \text{if vehicle } w \text{ visits nodes } i \text{ and } j \text{ respectively completes its tour at hub } k \\ 0, \text{otherwise} \end{cases}$$

$collect\_sum_{kv}$ = Maximum collection distance of the route of vehicle v starting its route from hub k

$distribute\_sum_{iw}$ = Maximum distribution distance of the route of vehicle *w* completing its route at its hub

Model 3.3: *Multiple Traveling Salesman Problem with Center Objective*

$$Min\ Z,$$

$$\sum_{j\in N:j\neq i} x_{ijv} - \sum_{j\in N:j\neq i} x_{jiv} = 0,\ \ \forall\, v \in V, i \in N \tag{1}$$

$$u_j \geq u_i + 1 + L * \sum_{v\in V} x_{ijv} - 1,\ \ \forall\, i \in N, j \in NH \tag{2}$$

$$\sum_{k\in H} collect_{kijv} = x_{ijv},\ \ \forall\, i \in N, j \in NH, v \in V: i \neq j \tag{3}$$

$$collect_{kijv} \leq x_{ijv},\ \ \forall\, k \in H, i \in N, j \in NH, v \in V \tag{4}$$

$$distribute_{ijkv} = x_{ijv},\ \ \forall\, i \in NH, j \in N, k \in H, v \in V: i \neq j \tag{5}$$

$$\sum_{k\in H} distribute_{ijkv} = x_{ijv},\ \ \forall\, i \in NH, j \in N, v \in V: i \neq j \tag{6}$$

$$\sum_{j\in N} \sum_{i\in H:j\neq i} x_{ijv} = 1,\ \ \forall\, v \in V \tag{7}$$

$$\sum_{j\in N} \sum_{v\in V:j\neq i} x_{ijv} = n_i,\ \ \forall\, i \in H \tag{8}$$

$$\sum_{j\in N} \sum_{v\in V:j\neq i} x_{jiv} = n_i,\ \ \forall\, i \in H \tag{9}$$

$$\sum_{j\in N} \sum_{v\in V:j\neq i} x_{ijv} = 1,\ \ \forall\, i \in NH \tag{10}$$

$$\sum_{j\in N} \sum_{v\in V:j\neq i} x_{jiv} = 1,\ \ \forall\, i \in NH \tag{11}$$

$$distribute\_sum_{kv} \geq \sum_{i\in NH} \sum_{j\in N:i\neq j} d_{ij} * distribute_{ijkv},\ \ \forall\, k \in H, v \in V \tag{12}$$

$$collect\_sum_{kv} \geq \sum_{i\in N} \sum_{j\in NH:i\neq j} d_{ij} * collect_{kijv},\ \ \forall\, k \in H, v \in V \tag{13}$$

$$Z \geq distribute\_sum_{kv} + collect\_sum_{kw},\ \ \forall\, k \in H, v \in V, w \in V: v \neq w \tag{14}$$

Constraint (1) in **Model *3.3*** is flow balance constraint. Constraint (2) is sub-tour elimination constraint. Constraint (3)-(4) are related to collection of a route. Constraint (5) and (6) are related to the distribution value of a route. Constraint (7)-(11) guarantee that if the node is a non-hub node, then this node is visited by just one vehicle. If the node is a hub node, then there is as n_i vehicles arriving and leaving from this hub node.

Constraint (12) calculates the maximum distribution and Constraint (13) calculates the maximum collection of the routes. Constraint (14) ensures that the maximum distance distance between any node pair i and j can be equal to the sum of the maximum collection ($collect\_sum$) and maximum distribution ($distribute\_sum$) and the distance between hubs for vehicles assigned to different hubs.

3.2.3. Traveling salesman problem formulation

In this section, the traveling salesman problem formulation based on the 3-index vehicle-flow formulation, which is used to find the hub's route when the number of vehicles assigned to each hub is 1, is given. The goal here is to minimize the total route length, starting with the hub and visiting each city only once. We note here that center type objective is used in our models, since there is only one vehicle. So, this objective function is equivalent to the center type objective function.

Additional decision variables and the mathematical model formulation are provided in the following:

Decision Variables:

$$x_{ij} = \begin{cases} 1, \text{if the traveling salesman travels from node } i \text{ to node } j \\ 0, \text{otherwise} \end{cases}$$

c_{ij}: the cost of traveling from i to j

Model 3.4: *Traveling Salesman Problem Formulation*

$$Min\ Z, \tag{1}$$

$$min \sum_{i \in N} \sum_{j \in N: j \neq i} c_{ij}\, x_{ij}, \tag{2}$$

$$\sum_{i \in N: i \neq j} x_{ij} = 1, \ \ \forall\, j \in N \tag{3}$$

$$\sum_{j \in N: j \neq i} x_{ji} = 1, \ \ \forall\, i \in N \tag{4}$$

$$u_i - u_j + n x_{ij} \leq n - 1, \ \ \forall\, 2 \leq i \neq j \leq n \tag{5}$$

Constraint (2) and (3) in **Model *3.4*** ensure that each city is only visited once and by just one vehicle. Constraint (4) assures that only one tour covers all cities, rather than two or more disconnected tours that cover all cities collectively.

4. SIMULATED ANNEALING ALGORITHM

One of the most common heuristic approaches of finding solutions for combinatorial optimization problems is the Simulated Annealing (SA) algorithm. It is inspired by the metalwork annealing process, and presented by Kirkpatrick et al. (1983). The annealing process determines the ideal molecular structure of metal particles where the potential energy of the mass is decreased and relates to the incremental cooling of the metals steadily following its high-temperature state. SA algorithm typically uses the iterative motion of the variable temperature parameter to mimic the annealing of the metals (Ingber, 1993).

The outputs of the objective functions are compared with the current and adjacent points within the domain, in a simple optimization algorithm, so that when the neighboring point provides a better solution than the present one, it is saved as the current solution for the next iteration. The algorithm otherwise terminates the process without aiming for better outcomes in the broader domain. This means that the algorithm is vulnerable to be stuck in local minimum or maximum solutions. SA algorithm proposes an efficient solution to this problem by integrating two iterative loops, the cooling protocol for the annealing process and the Metropolis criteria. The fundamental principle behind the criteria of Metropolis is to be executed arbitrarily to further investigate the candidate's neighborhood in order to avoid being stuck at local optima. In particular, let the minimization problem be considered and an objective function $f(x_i)$ that fits the argument set of $x_i = \{x_1, x_2, ..., x_n\}$ where $n \in R$ is defined. So, if $f(x_{i+1}) < f(x_i)$, take x_{i+1} as a new optimum point of the candidate to move as a neighbor. Contrariwise, $w = \exp[-\{f(x_{x+1}) - f(x_i)\}/T_c]$ is defined, where T_c presents the current temperature parameter. Also, a random number s is generated where $0 < s < 1$ (Metropolis, Rosenbluth, Rosenbluth, Teller, & Teller, 1953). After that, if $w > s$ relation returns true, the algorithm will consider x_{i+1} as a new minimum candidate as well, otherwise it will deny and return to the last step and produce a new random number: s. The Metropolis criterion therefore permits movement of the current phase to some degree; even the objective direction of the function is converged through the potential local minimum point.

The Metropolis algorithm, on the other hand, solves the constant temperature (getting stuck at a local minimum) issue. This means that the algorithm requires a broader search zone for large T_c values. This condition increases the likelihood of reaching the global minimum or at least a very good local minimum. In addition, since the iterative movement of the algorithm is randomly initialized, global minima may be skipped, or the algorithm follows the extreme point in an inaccurate approach in the inappropriate areas. Oppositely, SA needs smaller search areas for smaller values of T_c, which will result in a trap at a local minimum in this case. To eliminate this drawback, SA provides an iterative approach to change the temperature parameter and the solution point by integrating nested loops.

The SA algorithm's movement starts with a greater temperature value to perform the iteration of the internal loop and, within the current internal loop, assign the solution with the best objective function as the new candidate solution. After that, in the outer loop, the temperature is decreased by α, which is the cooling factor, and the starting temperature is updated. This iterative procedure proceeds until the lowest temperature limit is met, or a defined number of iterations is reached. **Algorithm *4.1*** shows the steps of the SA algorithm.

Notice that changing the temperature in the outer loop and random choice step sizes in the inner loop are very important. If T_{max} is initially set to be very high, $w > s$ relation is fulfilled with every iteration such that the algorithm in the domain of the function to be optimized is probably terminated at a random minimum. In addition, the relation $w > s$ is not satisfied if T_{max} is initially set to be very small and the movement is terminated at the first minimum. Also, choosing the rate of decrease in temperature has a significant impact on the efficiency of SA (Eren, Kucukdemiral, & Ustoglu, 2017). Therefore, SA algorithm overcomes these limitations and deals with the optimization problem by setting T_{max} to a large value then decrease the temperature gradually by traversing through a great number of points in the objective function's domain. The temperature is decreased steadily in the outer loop; meanwhile, the search zone is narrowed by setting the local minimum in the former internal loop. So, it excludes the local minimum points from the search zone, and therefore reaches the global minimum point accordingly. It can therefore be observed that, among other heuristic techniques, SA is a highly preferred strategy as

it gives functional randomness in the search to escape local optimum points (Dueck & Scheuer, 1990). Contrastingly, SA includes a balance between solution sensitivity and computational time.

Algorithm 4.1: *Simulated Annealing*

Input: x (initial solution), T_{max} (maximum temperature), T_{min} (minimum temperature), α (temperature decreasing rate), $iter_{max}$ (maximum iteration), N (operators set)
Output: x' (improved solution)
Function SA(x, T_{max}, T_{min}, $iter_{max}$);
$x' \leftarrow x$;
$iter \leftarrow 0$;
$T \leftarrow T_{max}$;
repeat
  repeat
    $iter \leftarrow iter + 1$;
    choose $x'' \in \mathrm{N}(x')$ randomly;
    $\Delta \leftarrow f(x'') - f(x')$;
    if $\Delta < 0$ **then**
      $x' \leftarrow x''$;
    else
      Take($s \in (0,1)$);
      if $s < e^{-\Delta/T}$ **then**
        $x' \leftarrow x''$;
      end
    end
  until $iter = iter_{max}$;
  $T \leftarrow T * \alpha$;
  $iter \leftarrow 0$;
until $T \leq T_{min}$;
return x';

5. VARIABLE NEIGHBORHOOD SEARCH ALGORITHM

Variable Neighborhood Search (VNS) is a meta-heuristic algorithm widely used to resolve problems related to combinatory and global optimization. This algorithm's key concept is to use local search to adjust the neighborhood of the search. VNS, as a form of structural neighborliness improvement based on repeated local searches, was first suggested by Mladenovic and Hansen in 1997 (Mladenovic & Hansen, 1997). VNS is a basic algorithm, and its value is that it has a few parameters (Hansen & et al., 2010). This gives it a very decent pace. VNS has strongly based its conclusions on the following:

i) A local optimum is not always a local optimum with respect to one neighborhood structure in another neighborhood structure. This is increasingly manipulated by using complex steps to locate a local optimum in all neighborhood structures.

ii) A global optimum for all feasible neighborhood structures is a local optimum. This means that many neighborhoods are in use if there is low efficiency in local optima.

iii) Local optima are comparatively close to each other with regards to one or more neighborhoods for several problems. This implies that the new incumbent solution can be exploited more successfully (Hansen, Mladenović, Todosijević, & et al., 2017).

5.1. Variable Neighborhood Search Components

An improvement process is used to evolve the solution, and a process called the shaking phase is used in optimistically solving local minima jams, which are the components of a VNS heuristic. The improvement and the shaking processes are carried out in accordance with the alteration in the neighborhood before a predefined stop criterion is met. That is to say, before a stop criterion is met, the following three steps are repeated:

(1) Shaking process

(2) Improvement process

(3) Neighborhood change

5.1.1. Shaking process

The goal of a VNS heuristic shaking process is to optimistically overcome local minima traps. VNS utilizes a finite number of pre-set operators, which is indicated by $N = \{N_1, \ldots, N_{k_{max}}\}$. Each operator N_k , $1 \leq k \leq k_{max}$ maps a given solution x to a solutions' set within the kth neighborhood under solution x that is denoted by $N_k(x)$. Notice that the order of operators in set N determines the order in which the neighborhood structures of a given solution x should be examined. The basic shaking process involves choosing a random solution in the kth neighborhood structure, $N_k(x)$. However, an entirely random jump in the kth neighborhood is too diverse for some problem instances. Therefore, so-called intensified shaking is better to be used sometimes as it takes into consideration how sensitive the objective function is to minor shaking (change) of the solution. Nevertheless, the VNS variants presented in this study use simple shaking processes based on a random solution from the kth neighborhood structure (see **Algorithm *5.1***).

Algorithm 5.1: *Shaking*

Input: x (initial solution), k (neighborhood number), N (operators set)
Output: x' (random solution)
Function Shake(x, k, N);
1 choose $x' \in N_k(x)$ randomly;
2 return x';

5.1.2. Improvement processes

VNS uses a local search process or a variable neighborhood descent (VND) process to reach $x \in X$ solution that is considered a local minimum.

5.1.2.1. Local search process

Local search heuristic is based on the $\mathcal{N}(x)$ neighborhood structure inspection of the existing incumbent solution x at each iteration. It starts from an initial solution x, then, in the predefined neighborhood structure $\mathcal{N}(x)$, it seeks a solution that is better than x' (if it exists) at each iteration, and it becomes the current solution x. A local search heuristic completes its work by reaching the local optimum solution in its neighborhood structure $\mathcal{N}(x)$ and set that as the incumbent solution x.

The most popular search methods used in local search heuristics are the first improvement (after the first improvement in the objective function is found within the $\mathcal{N}(x)$ neighborhood, it is set as the new incumbent solution) and the best improvement (the solution with the best objective function of all $\mathcal{N}(x)$ solutions (if it exists) is set as the new incumbent solution) (Hansen & et al., 2010).

This study utilizes the local search heuristic with the best improvement method where $f(x')$ is not possible to be less than $f(x)$ for $\forall x' \in \mathcal{N}_k(x) \subseteq X$. So, the initial solution x is chosen, a descent path from x is sought within a $\mathcal{N}(x)$ neighborhood, and that the minimum $f(x)$ in $\mathcal{N}(x)$ is taken in the same orientation. The heuristic ends if there is no path of descent; contrariwise, it iterates. In general, the steepest descent path is used, also known as the best improvement (Kirlik & Ceyda, 2012). **Algorithm *5.2*** shows the pseudocode of the best improvement step.

Algorithm 5.2: *Best improvement*

Input: x (initial solution), $\mathcal{N}$ (neighborhoods)
Output: x' (local minimum)
Function BestImprovement(x, $\mathcal{N}$);
1 repeat
2 $\quad x' \leftarrow x$;
3 $\quad x \leftarrow \operatorname{argmin}_{y \in \mathcal{N}_k(x')} f(y)$;
4 **until** $f(x') \leq f(x)$;
5 return x';

5.1.2.2. Variable neighborhood descent process

The variable neighborhood descent (VND) processes take advantage of the fact that the local optimum solution is more likely to be a global optimum, if it is the same in multiple neighborhoods than in one neighborhood structure alone. More specifically, in a VND method, a sequential or nested exploration process investigates many neighborhood structures that might enhance a given solution. It uses the first improvement or best improvement as a search strategy.

The basic sequential VND (BVND) procedure starts by ordering several neighborhood structures in a list, and then they are tested one by one (according to the initial order). A set of operators $\mathcal{N} = \{\mathcal{N}_1, \ldots, \mathcal{N}_{l_{max}}\}$ determines the neighborhood

structures with their order of test. The basic sequential VND process explores its neighborhood structures, as defined by operators $\mathcal{N}_l$, $1 \leq l \leq l_{max}$ one after another in accordance with the initial order, starting from a given solution x. When an enhancement in the incumbent solution occurs in any neighborhood structure, the basic sequential VND process resumes the search in the new incumbent solution's first neighborhood structure (according to the given order). **Algorithm *5.3*** shows the steps of the sequential VND using the best improvement search strategy.

Algorithm 5.3: *Sequential VND using the best improvement search strategy*

Input: x (initial solution), l_{max} (number of neighborhoods), $\mathcal{N}$ (neighborhoods)
Output: x' (local minimum)
Function VND(x, l_{max}, $\mathcal{N}$);
1 $l \leftarrow 1$;
2 **repeat**
3 $x' \leftarrow \text{argmin}_{y \in \mathcal{N}_l(x)} f(y)$;
4 SequentialNeighborhoodChange(x, x', l);
5 **until** $l = l_{max}$;
6 **return** x';

1.1.1. Neighborhood change

The goal of this process is to direct the VNS heuristic while discovering the solution space. More specifically, it determines which area to discover next and whether or not a solution is approved as a new incumbent solution. There are many widely used neighborhood change processes, e.g., sequential neighborhood change process, cyclic neighborhood change process, pipe neighborhood change process, and the skewed neighborhood change process (Hansen, Mladenović, Todosijević, & et al., 2017). For the sake of this study, the sequential neighborhood change process is the most appropriate one.

Algorithm *5.4* provides the steps to the sequential neighborhood change process. Firstly, the value $f(x')$ of a new solution x' is compared with the $f(x)$ value of the current solution x, which is in neighborhood k. If an enhancement of the current solution in some neighborhood structure has been made, k will be set to the initial value (first neighborhood structure), and the current solution will be replaced with the new one. If

not, the next neighborhood (according to the defined order) would be taken into consideration.

Algorithm 5.4: *Sequential neighborhood change process*

Input: x (current solution), x' (new solution), k (neighborhood number)
Output: x (current or new solution), k (neighborhood number)
Function SequentialNeighborhoodChange(x, x', k);
1 **if** $f(x') < f(x)$ **then**
2 $\quad x \leftarrow x'$;
3 $\quad k \leftarrow 1$;
4 else
5 $\quad k \leftarrow k + 1$;
6 End

5.2. Variable Neighborhood Search Variants

There are several VNS variants, e.g., fixed neighborhood search, Basic VNS, reduced VNS, general VNS, skewed VNS, and nested VNS. Also, there are some advanced VNS variants (Hansen, Mladenović, Todosijević, & et al., 2017). This study considers the Basic VNS experiments.

- Basic VNS

The Basic VNS variant starts by executing a shaking process presented in **Algorithm *5.1***, then a simple local search process using the best improvement search strategy presented in **Algorithm *5.2***. Lastly, a sequential neighborhood change step is presented in **Algorithm *5.4***. This continues until the predefined stop condition is met. Some common stopping conditions are the maximum allowed CPU time, the maximum number of iterations, and the maximum number of iterations without enhancement. The first two stopping conditions are used in this study. The steps of Basic VNS are given in **Algorithm *5.5***.

Algorithm 5.5: *Basic variable neighborhood search*

Input: x (initial solution), t (stopping condition), k_{max} (number of neighborhoods), $\mathcal{N}$ (neighborhoods), N (operators set)
Output: x (improved solution)
Function VNS(x, t, k_{max}, $\mathcal{N}$, N);

```
repeat
    k ← 1;
    repeat
        x′ ← Shake(x, k, N);
        x″ ← BestImprovement(x′, 𝒩); // local search
        SequentialNeighborhoodChange(x, x″, k);
    until k = k_max;
until t is fulfilled;
return x;
```

6. INITIAL SOLUTION GENERATION ALGORITHMS AND A MAT-HEURISTIC

The initial solution is the first step in solving the optimization problems. Particularly, the optimization problems' solver algorithms are fed by the initial solution and start with that as the incumbent solution. So, the initial solution's structure is one of the most effective factors on the movement towards the optimal solution and on the objective function of most of the algorithms that solve the optimization problems. Therefore, there are many common algorithms used in implementing the initial solution. This study focuses on the random, greedy, greedy-random, random-greedy, Gurobi, and greedy-Gurobi based initial solution generation algorithms which are explained in detail in the following subsections.

An initial solution generation begins by picking hub nodes then assigning non-hub nodes to hubs, so the resulted routes are combined to create the initial solution.

6.1. Random Initial Solution Generation Algorithm

The random initial solution generation algorithm has two sub-processes, as shown in Algorithm 6.1. Firstly, picking hubs randomly by choosing h random number of nodes with $0 \leq h < n$, where n is the number of nodes in the solution's space, then adding each $node$ to a $hubs$ list that is returned to the invoker at the end of this process. Algorithm 6.2 shows the steps of implementing the procedure of picking hubs randomly.

Algorithm 6.1: *Random initial solution*

Input: n (number of nodes), h (number of hubs), v (number of vehicles per hub)
Output: x (random solution)
Function RandomInitialSolution(n, h, v);
1 $hubs \leftarrow$ PickHubsRandomly(n, h);
2 $x \leftarrow$ AssignNodesRandomly(n, h, $hubs$, v);
3 **return** x;

Algorithm 6.2: *Pick hubs randomly*

Input: n (number of nodes), h (number of hubs)
Output: $hubs$ (list of hubs)
Function PickHubsRandomly(n, h);

1 $i \leftarrow 0$;
2 **repeat**
3 $i \leftarrow i + 1$;
4 Pick($node \in [0, n)$ randomly; // integer
5 add $node$ to $hubs$;
6 **until** $i = h$;
7 **return** $hubs$;

Secondly, the other sub-process is assigning non-hub nodes to hubs randomly. It starts by setting the minimum and maximum number of nodes assigned per vehicle route: $numOfNodesForVehicle_{min}$ to 1 and $numOfNodesForVehicle_{max}$ to $n - (h * (v + 1)) + 1$, where v is the number of vehicles for each hub node. Consequently, each route should have at least one non-hub node assigned. Then, the algorithm iterates through vehicles and picks a random $numOfNodesForVehicle$ (number of nodes for the current route) and tests it against the remaining number of nodes and vehicles to ensure that it does not run out of nodes. If the remaining nodes are not enough for the remaining vehicles, it re-picks $numOfNodesForVehicle$ again. Otherwise, the process continues by checking; if the current vehicle is the last one, all the remaining nodes will be assigned to it. After that, the algorithm loops $numOfNodesForVehicle$ times, and each time it picks a random node and checks if it is already assigned before, it re-picks another one and assigns it. Then, it completes the route by connecting the output vehicle to its hub. Lastly, the routes are added up to build the solution and return it. The process of assigning nodes randomly is explained in **Algorithm *6.3***.

Algorithm 6.3: *Assign nodes randomly*

Input: n (number of nodes), h (number of hubs), $hubs$ (list of hubs), v (number of vehicles per hub)
Output: x (random solution)
Function AssignNodesRandomly(n, h, $hubs$, v);
1 $numOfNodesForVehicle_{min} \leftarrow 1$;
2 $numOfNodesForVehicle_{max} \leftarrow n - (h * (v + 1)) + 1$;
3 $remainingNodes \leftarrow n - h$;
4 $i \leftarrow 0$;
5 **repeat**
6 Pick($numOfNodesForVehicle \in [numOfNodesForVehicle_{min}, numOfNodesForVehicle_{max})$ randomly;

```
    remainingVehicles ← h * v − i;
    if remainingNodes − numOfNodesForVehicle ≥ remainingVehicles − 1 then
        if remainingVehicles = 1 then
            numOfNodesForVehicle ← remainingNodes;
        end
        remainingNodes ← remainingNodes − numOfNodesForVehicle;
        j ← 0;
        repeat
            j ← j + 1;
            Pick (node ∈ [0, n) randomly ;
            if node already assigned then
                j ← j − 1;
                redo the iteration;
            end
            add node to vehicle;
        until j = numOfNodesForVehicle;
        connect vehicle to its hub in hubs
    else
        i ← i − 1;
    End
    add vehicle to x;
    i ← i + 1;
until i < h * v;
return x;
```

6.2. Greedy Initial Solution Generation Algorithm

The greedy initial solution generation algorithm has two sub-processes, as shown in **Algorithm *6.4***. Firstly, the picking hubs greedily step starts by calculating the averages of the distances for each node to the other nodes and storing them in $nodesDistanceAvgList$. Then, the algorithm sorts the averages list ascendingly. After that, it picks the first h number of nodes from $nodesDistanceAvgList$ as hubs and adds them to the $hubs$ list that is returned to the invoker at the end of this process. **Algorithm *6.5*** shows the steps of implementing picking hubs greedily.

Algorithm 6.4: *Greedy initial solution*

Input: n (number of nodes), h (number of hubs), v (number of vehicles per hub)

```
    Output: x (greedy solution)
    Function GreedyInitialSolution(n, h, v);
1   hubs ← PickHubsGreedily(n, h);
2   x ← AssignNodesGreedily(n, h, hubs, v);
3   return x;
```

Algorithm 6.5: *Pick hubs greedily*

```
    Input: n (number of nodes), h (number of hubs)
    Output: hubs (list of hubs)
    Function PickHubsGreedily(n, h);
1   i ← 0;
2   repeat
3       i ← i + 1;
4       sum ← 0;
5       j ← 0;
6       repeat
7           j ← j + 1;
8           sum ← sum + distance(i, j);
9       until j = n;
10      average ← sum/n;
11      add average to nodesDistanceAvgList;
12  until i = n;
13  nodesDistanceAvgList ← sortAscendingly(nodesDistanceAvgList);
14  i ← 0;
15  repeat
16      i ← i + 1;
17      add nodesDistanceAvgList(i) to hubs;
18  until i = h;
19  return hubs;
```

Secondly, the other sub-process is assigning non-hub nodes to hubs greedily. Like in the random initial solution algorithm, it starts by setting the minimum and maximum number of nodes assigned per route (vehicle): $numOfNodesForVehicle_{min}$ to 1 and $numOfNodesForVehicle_{max}$ to $n - (h * (v + 1)) + 1$. Consequently, each route should have at least one non-hub node assigned. Then, the algorithm iterates through vehicles and checks if this is the first time to visit $currentHub$, it gets the distances

between $currentHub$ and each non-visited node and stores them in $nodesToHubDistance$ list. After sorting the list ascendingly, it picks a random $numOfNodesForVehicle$ and tests it against the remaining number of nodes and vehicles to ensure that it does not run out of nodes. If the remaining nodes are not enough for the remaining vehicles, it re-picks $numOfNodesForVehicle$ again. Otherwise, the process continues by checking; if the current vehicle is the last one, all the remaining nodes will be assigned to it. After that, the algorithm loops $numOfNodesForVehicle$ times, and each time it picks the first node (the node which has the minimum distance to the $currentHub$) in $nodesToHubDistance$ list and checks if it is already assigned before, it re-picks again. Then, it completes the route by connecting the output vehicle to its hub. Lastly, the routes are added up to build the solution and return it. **Algorithm *6.6*** shows the process of assigning nodes greedily.

Algorithm 6.6: *Assign nodes greedily*

Input: n (number of nodes), h (number of hubs), $hubs$ (list of hubs), v (number of vehicles per hub)
Output: x (greedy solution)
Function AssignNodesGreedily(n, h, $hubs$, v);
$numOfNodesForVehicle_{min} \leftarrow 1$;
$numOfNodesForVehicle_{max} \leftarrow n - (h * (v + 1)) + 1$;
$remainingNodes \leftarrow n - h$;
$i \leftarrow 0$;
repeat
  if $currentHub$ visited for the first time **then**
    $j \leftarrow 0$;
    repeat
      if $node_j$ is not hub **then**
        add distance between $node_j$ and $currentHub$ to $nodesToHubDistance$ list;
      end
      $j \leftarrow j + 1$;
    until $j < n$;
    $nodesToHubDistance \leftarrow sortAscendingly(nodesToHubDistance)$;
  end
  Pick($numOfNodesForVehicle \in [numOfNodesForVehicle_{min}, numOfNodesForVehicle_{max}]$) randomly;
  $remainingVehicles \leftarrow h * v - i$;

```
    currentHub ← i / numVehiclesPerHub;
    if remainingNodes − numOfNodesForVehicle ≥ remainingVehicles − 1 then
        if remainingVehicles = 1 then
            numOfNodesForVehicle = remainingNodes;
        end
        remainingNodes = remainingNodes − numOfNodesForVehicle;
        j ← 0;
        repeat
            j = j + 1;
            Pick the first node in nodesToHubDistance list;
            if node already assigned then
                j = j − 1;
                redo the iteration;
            end
            add node to vehicle;
            set node as assigned;
            remove node from nodesToHubDistance list;
        until j = numOfNodesForVehicle;
        connect vehicle to its hub in hubs;
        add vehicle to x;
    else
        i = i − 1;
    end
    i = i + 1;
until i = h * v;
return x;
```

6.3. Greedy-Random Initial Solution Generation Algorithm

The greedy-random initial solution generation algorithm is implemented by using two sub-processes, as Algorithm 6.7 shows. It begins by picking hubs greedily then assigning non-hub nodes to hubs randomly, as shown in Algorithm 6.5 and Algorithm 6.3, respectively.

Algorithm 6.7: *Greedy-random initial solution*

Input: n (number of nodes), h (number of hubs), v (number of vehicles per hub)

Output: x (greedy-random solution)

Function GreedyRandomInitialSolution(n, h, v);

1 $hubs \leftarrow$ PickHubsGreedily(n, h);

2 $x \leftarrow$ AssignNodesRandomly(n, h, $hubs$, v);

3 **return** x;

6.4. Random-Greedy Initial Solution Generation Algorithm

The random-greedy initial solution generation algorithm is opposite to the greedy-random algorithm, as shown in Algorithm 6.8. This algorithm picks the hubs randomly, as seen in Algorithm 6.2, and assigns non-hub nodes to hubs greedily, as in Algorithm 6.6.

Algorithm 6.8: *Random-greedy initial solution*

Input: n (number of nodes), h (number of hubs), v (number of vehicles per hub)

Output: x (random-greedy solution)

Function RandomGreedyInitialSolution(n, h, v);

1 $hubs \leftarrow$ PickHubsRandomly(n, h);

2 $x \leftarrow$ AssignNodesGreedily(n, h, $hubs$, v);

3 **return** x;

6.5. A Mat-heuristic Approach for Initial Solutions Generation

Alongside metaheuristics, Gurobi commercial optimization solver is used in generating mat-heuristic initial solutions.[1] The following sub-sections explain two different initial solutions: one built using Gurobi alone and another one that is a combination of greedy algorithm and Gurobi.

6.5.1. Gurobi based initial solution generation algorithm

Algorithm 6.9 shows the Gurobi-based initial solution generation algorithm implemented with two sub-processes. It starts by picking hubs and assigning non-hub nodes to hubs using the p-hub center model (**Model *3.2***) that shows the steps and constraints of this sub-process as explained in section 3.2.1. After that, the algorithm creates and optimizes routes using Gurobi Optimizer, as shown in **Algorithm *6.10***.

[1] Gurobi's website: https://www.gurobi.com

The nodes assignment process starts by allocating non-hub nodes to hubs with the p-hub center using **Model *3.2***, but this does not create routes. The stage of creating and optimizing routes is done by using TSP algorithm (**Model *3.4***) if $v = 1$. Otherwise, if $v > 1$, mTSP algorithm (**Model *3.3***) is used to create and optimize the vehicles' routes. TSP and mTSP models are explained in sections 3.2.3 and 3.2.2. After the assignment phase is finished, the initial solution becomes ready.

Algorithm 6.9: *Gurobi-based initial solution*

Input: n (number of nodes), h (number of hubs), v (number of vehicles per hub)
Output: x (Gurobi-based solution)
Function GurobiBasedInitialSolution(n, h, v);
1 $hubs \leftarrow$ PickHubsWithGurobiPHubCenter(n, h);
2 $x \leftarrow$ CreateRouteWithGurobi(n, h, v);
3 **return** x;

Algorithm 6.10: *Assign nodes with Gurobi*

Input: n (number of nodes), $hubs$ (list of hubs), v (number of vehicles per hub)
Output: x (Gurobi solution)
Function CreateRouteWithGurobi(n, h, v);
1 **repeat for each** hub in $hubs$
2 $nodes \leftarrow$ Assigned nodes to hub with Gurobi p-hub center;
3 **if** $v = 1$ **then**
4 $optimizedHub \leftarrow$ CreateAndOptimizeRouteWithGurobiTSP($nodes$, hub, v);
5 **else**
6 $optimizedHub \leftarrow$ CreateAndOptimizeRoutesWithGurobimTSP($nodes$, hub, v);
7 **end**
8 $x \leftarrow$ append $optimizedHub$;
9 **end**
10 **return** x;

6.5.2. Greedy-Gurobi initial solution generation algorithm

Algorithm 6.11 demonstrates the greedy-Gurobi initial solution generation algorithm, which has two stages. It begins by picking hubs greedily, as shown in Algorithm 6.5. Once that is done, the algorithm uses Gurobi Optimizer to assign non-hub

nodes to hubs as presented in Algorithm 6.10. This phase (non-hub nodes assignment with Gurobi) is explained in the previous section (section 6.5.1).

Algorithm 6.11: *Greedy-Gurobi initial solution*

Input: n (number of nodes), h (number of hubs), v (number of vehicles per hub)
Output: x (greedy-Gurobi solution)
Function GreedyGurobiInitialSolution(n, h, v);
1 $hubs \leftarrow$ PickHubsGreedily(n, h);
2 $x \leftarrow$ CreateRouteWithGurobi(n, h, v);
3 **return** x;

7. COMPUTATIONAL RESULTS

We carried out the solution algorithms of pHCRP on the TR and AP dataset and gave the results in tables. The TR dataset contains the data on the distances between 81 demand centers (nodes) in Turkey. The AP dataset is the Australian postal service dataset. It contains Sydney's postal districts and consists of 200 different nodes. The distance between the nodes in the AP dataset is not symmetrical.

SA and VNS metaheuristics were used separately to deal with the pHCRP, and Gurobi Optimizer version 9.0.2 was utilized to develop two different initial solutions. We have performed the experimental tests on a server that is running UBUNTU operating system and has a 3.5 GHz Intel Xeon E5-2643 v2 processor and 128,863 GB RAM in total, but we assign 30 GB to the heap. All the proposed metaheuristics (SA and VNS) are coded in Java, while Gurobi parts are coded in Python.

7.1. Parameter Tuning for Simulated Annealing and Variable Neighborhood Search

There are some shared parameters between the two metaheuristics (SA and VNS). Although the neighborhood moves order, the neighborhood moves which are used in the algorithms are $insertInRoute$ (insertion within one route), $insertWithinRoutes$ (insertion within two different vehicle routes), $edgeOptWithinRoutes$ (making an edge-opt within two different vehicle routes), $swapInRoute$ (swap within one route), $swapWithinRoutes$ (swap within two different vehicle routes), $edgeOptInRoute$ (making edge-opt within one route), and $hubMove$ (swap hub with one node in route) with the probabilities of 0.125, 0.125, 0.125, 0.125, 0.125, 0.125 and 0.25, respectively. Preliminary results show that giving a greater probability to $hubMove$ move will result in better solutions under reasonable time.

- Simulated Annealing:

SA has the initial temperature T which is set to 1,000,000, and the minimum temperature is set at .0000001. The temperature is decreased by $\alpha = 0.99$; that is better than 0.9 and 0.95 after conducting a comparison on a decent sample. The maximum time for an instance is considered as the primary stopping condition for the algorithm.

SA was tested on four different combinations of 2 binary parameters. The first is $useBest$, which controls if the algorithm will continue the next iteration with the best

solution used as the current solution or not. The second parameter is $force$, which decides either to keep the algorithm moving until it reaches the maximum time or increments the inner-loop counter when a substandard solution is ignored until it reaches L number of bad solutions which are equal to n or $n * 10$. After making some tests on the 25 and 50 nodes instances, we found that setting $useBest$ to false and $force$ to true gives better results than the other combinations.

- Variable Neighborhood Search:

VNS starts with the shaking step to shuffle the solution by making a random move (bad move accepted) then a random number (0-20) of good moves.

7.2. Initial Solutions CPU Times

The average CPU times needed for generating ten initial solutions of each of the six strategies on the TR and AP datasets are shown in **Table *7.1***. It is presented that all methods of initial solutions take less than one second to be created, except the GRB and GRD-GRB initial solutions of the problem instances with 25 nodes or more. The initial solutions instances that take more than a second are marked in bold in **Table *7.1***.

Table 7.1: *Initial solutions CPU times*

Prob	Initial Solutions CPU Times (seconds)					
	RND	**GRD**	**GRD-RND**	**RND-GRD**	**GRB**	**GRD-GRB**
TR.10.2.1	0.00036	0.0023	0.0002	0.0002	0.09	0.05
TR.10.2.2	0.00008	0.0003	0.0002	0.0002	0.08	0.04
TR.10.3.1	0.00006	0.0003	0.0001	0.0002	0.08	0.03
TR.15.2.1	0.00008	0.0003	0.0002	0.0011	0.23	0.05
TR.15.2.2	0.00109	0.0029	0.0002	0.0011	0.32	0.10
TR.25.2.1	0.00008	0.0005	0.0002	0.0002	0.75	0.37
TR.25.2.5	0.00017	0.0006	0.0003	0.0004	**8.12**	**211.51**
TR.25.5.1	0.00005	0.0005	0.0003	0.0004	**1.71**	0.26
TR.25.5.2	0.00007	0.0007	0.0002	0.0005	**3.12**	0.31
TR.50.2.1	0.00008	0.0008	0.0005	0.0003	**9.87**	**6.79**
TR.50.2.5	0.00015	0.0011	0.0005	0.0015	**14411.21**	**14400.90**

TR.50.5.1	0.00008	0.0007	0.0004	0.0007	**51.11**	0.74
TR.50.5.2	0.00011	0.0009	0.0005	0.0008	**176.11**	**1.07**
TR.81.2.1	0.00013	0.0010	0.0008	0.0003	**40.92**	**10.61**
TR.81.2.5	0.00028	0.0016	0.0010	0.0011	**14436.73**	**14402.25**
TR.81.5.1	0.00015	0.0025	0.0009	0.0011	**5235.67**	**68.81**
TR.81.5.2	0.00027	0.0026	0.0011	0.0015	**3127.45**	**479.68**
TR.81.9.1	0.00020	0.0030	0.0007	0.0012	**7200.57**	**7200.82**
TR.81.9.2	0.00039	0.0025	0.0008	0.0022	**7201.09**	**7201.41**
TR.81.9.3	0.00053	0.0030	0.0009	0.0026	**7202.20**	**7202.77**
TR.81.9.4	0.00049	0.0037	0.0011	0.0024	**7209.52**	**7214.80**
TR.81.9.5	0.00048	0.0035	0.0010	0.0039	**7207.49**	**7271.79**
AP.100.5.1	0.00076	0.00852	0.00248	0.00738	**961.81**	**6.56**
AP.100.5.2	0.00160	0.00544	0.00112	0.00226	**1238.07**	**13.04**
AP.100.5.5	0.00079	0.00602	0.00154	0.00223	**15444.06**	**14914.35**
AP.200.10.1	0.00043	0.00946	0.00416	0.00528	**7210.11**	**7210.74**
AP.200.10.2	0.00112	0.00966	0.00232	0.00795	**7208.36**	**7305.79**
AP.200.10.5	0.00186	0.01406	0.00363	0.01054	**43043.05**	**51332.27**
Avg	0.00043	0.00316	0.00097	0.00213	**5308.21**	**5230.64**

In conclusion, the order of the initial solutions in terms of the least CPU time needed is RND, GRD-RND, RND-GRD, GRD, GRD-GRB, then GRB initial solutions, with averages of 0.00043, 0.00097, 0.00213, 0.00316, 5308.21, and 5230.64 seconds, respectively. So, it is evident that GRB and GRD-GRB initial solutions take a lot of CPU time, especially for problem instances with a large number of nodes n. That is because the time limit is 2 hours for the p-hub center model (Model 3.2) and 2 hours for each hub routes optimization using mTSP or TSP (Model 3.3 or Model 3.4). For instance, TR.50.2.5 and TR.81.2.5 instances take more than 4 hours each. Specifically, $2\ hubs * 2\ hours = 4\ hours$. So, 4 hours for optimizing the two hubs. So, in both instances, it reached the time limit while optimizing the two hubs' routes using mTSP (as the number of vehicles per hub $v > 1$) and the rest of CPU time was for p-hub center.

7.3. Results of Simulated Annealing and Variable Neighborhood Search for the Single Allocation p-Hub Center and Routing Problem

We generated 28 problem instances to test the metaheuristics (SA and VNS) by setting the number of nodes n to 10, 15, and 25 for the TR dataset. In order to test the algorithms on larger instances, we include the 50-node and 81-node instances from the TR dataset and 100-node and 200-node from the AP dataset. The number of hubs p is set to be 2, 3, 5, 9, and 10, depending on the number of nodes n of such an instance. The number of vehicles v for each hub is assumed to be 1, 2, 3, 4, and 5.

Each problem instance was run ten times. The attributes of the problem instance ($DS.n.p.v$) in the first column of **Table *7.2*** to **Table *7.7*** can be comprehended as DS represents the dataset short form, the 81-node Turkish dataset (TR) or the 200-node Australia Post (AP) dataset. Attribute n is the number of nodes, p is the number of hubs, and v is the number of vehicles which are assigned to each hub. The least cost (objective) that is found for each instance among all the implemented metaheuristics (SA and VNS) is presented in the second column. Found underneath the heading of SA and VNS are five columns where the minimum gap (of the heuristics), the average gap, the coefficient of variation, the average of initial solutions gap, and the average of iterations, where the algorithm's runs reached their best solution, are shown. The last column in **Table *7.2*** to **Table *7.9*** presents the running time (CPU) for each run of a problem instance. Where the CPU time for an instance depends on the number of nodes n of that instance. For n equals 10, 15, 25, 50, 81, 100, and 200 nodes, we set the maximum time of the run to 10, 30, 60, 270, 1,000, 3,600, and 7,200 seconds respectively. The problem instances' parameters (DS, n, p, v, and time limits for each instance) were set depending on Kartal et al.'s (2019) parameters, in order to compare the results with the previous researches on single assignment pHCRP.

The coefficient of variation is calculated by dividing the standard deviation (the standard deviation of the costs of a problem instance in 10 runs) by the mean of the population (the average of the costs of a problem instance in 10 runs). Thus, it indicates the range of variability according to the mean. We utilize this calculation because it enables us to compare how much variation there is between the 10 solutions objectives for a problem instance, even if the means differ significantly.

The average initial solutions gap is used to see how close the initial solutions are to the least (best) objective solution and the gap between the best and initial solutions. However, the average of iterations is listed to compare the number of iterations, on average, needed for the algorithm to keep improving and reaching the best solutions. Generally, an SA's complete improvement iteration takes less time than a VNS improvement iteration.

Naturally, not in a systematic way, the average of best cost solutions' iterations for SA and VNS gets larger and larger as the instance's number of nodes n increases, as shown in **Table *7.2*** to **Table *7.7***.

The metaheuristics (SA and VNS) results are divided and analyzed into tables according to the initial solution methods (**Table *7.2*** to **Table *7.7***). Each table shows and compares the results of the two metaheuristics according to specific initial solution method instances. In these tables, the number of nodes n, which is used for different instances in the TR and AP datasets, is set to 10, 15, 25, 50, 81, 100, and 200 nodes, where 1,2, 3, 4, or 5 vehicles v per hub are used. The last table (**Table *7.8***) shows the final results out of the 6 tables (**Table *7.2*** to **Table *7.7***).

Table 7.2 presents and compares the results of SA and VNS metaheuristics where the RND initial solution (Algorithm 6.1) is implemented on the TR and AP datasets. We can figure out from Table 7.2 that SA gets the least cost in all the cases in both datasets, except for TR.25.5.1 instance, which is reached by VNS. On the other hand, VNS does not reach the best solutions in most instances, especially those with more than 25 nodes. It moves far away from the minimum cost as the number of nodes n increases. As seen in the minimum gap column, VNS starts at 0.0% then reaches 24.97% and 153.56% in the TR and AP datasets, at TR.81.5.2 and AP.200.10.5 problem instances, respectively. Moreover, the average of minimum gaps of SA is 0.01% ≈ 0.0%, which is much less than the VNS average, which is 20.50%.

The average gaps of VNS in **Table *7.2***, **Table *7.3***, Table 7.4, and **Table 7.5** start at 1.71%, 2.53%, 2.16%, and 3.41%, respectively, at TR.10.2.2, TR.10.3.1, TR.10.2.2, and TR.10.2.2 problem instances, respectively. These are acceptable averages, but, in the same tables (RND, GRD, GRD-RND, and RND-GRD based problem instances), for the

larger instances in the TR dataset, VNS hits 115.63%, 101.16%, 89.20%, and 120.16%, respectively, at TR.81.9.3, TR.81.5.2, TR.81.9.4, and TR.81.9.2 problem instances, respectively, which are quite high. The same low performance is in the AP dataset instances; as VNS reaches 257.99%, 255.85%, 299.72%, and 313.46% for RND, GRD, GRD-RND, and RND-GRD based problem instances, respectively. In contrast, comparing to VNS, SA has much lower (better) average gaps for large number of nodes n, specifically, which are between 0.0% - 4.45%, 0.0% - 4.29%, 0.0% - 6.47%, and 0.0% - 4.05% in **Table *7.2***, **Table *7.3***, Table 7.4, and **Table 7.5**, respectively. These results mean that the SA solutions' objectives are generally closer to the least costs (best solutions) than the VNS solutions.

The coefficient of variation values in **Table *7.2***, **Table *7.2***, **Table *7.3***, Table 7.4, and **Table 7.5** are better in SA than VNS, even though they are very low in VNS, where the averages of coefficient of variation are 0.16% and 0.17% for RND and GRD-RND based instances, respectively, and 0.18% for GRD and RND-GRD based instances. On the other hand, they are 0.01% ≈ 0.0% for SA. Therefore, the VNS objectives are close to each other around the mean values in all the cases while SA minimum costs are condensed more, so they are more reliable. GRB and GRD-GRB coefficient of variation results are mentioned later in this section.

Regarding the average of initial solutions gaps, there is a noticeable difference between SA and VNS in **Table *7.2***. The SA average of initial solutions average gaps is 1036.03%, but it is 772.81% for VNS. So, for SA and VNS, the RND initial solution objectives, on average, are larger than the reached best costs by more than 10 and 7.5 times, respectively. So, SA surpasses VNS here too, where it improves the initial solutions more. In general, it is noticed that the SA metaheuristic improves the initial solutions in the TR dataset by more than 100% for all instances. However, the VNS metaheuristic improves them by 110% or more. So, regarding RND initial solution minimum percentage of improvement, VNS outperforms (**Table *7.2***).

Table 7.2: *Random initial solution problem instances results*

Prob		SA	VNS

	Least Cost	Min gap	Avg gap	Co. of Var.	Avg Init. Obj gap	Avg Iteration	Min gap	Avg gap	Co. of Var.	Avg Init. Obj gap	Avg Iteratio
TR.10.2.1	3597	**0.00**	0.44	0.01	103.40	18866	**0.00**	2.61	0.03	113.04	48
TR.10.2.2	2331	**0.00**	**0.00**	**0.00**	166.83	19039	**0.00**	1.71	0.03	159.67	64
TR.10.3.1	2651	**0.00**	**0.00**	**0.00**	130.38	16829	**0.00**	7.50	0.07	113.63	119
TR.15.2.1	4128	**0.00**	1.59	0.02	157.71	49867	**0.00**	7.61	0.03	180.85	517
TR.15.2.2	2769	**0.00**	0.26	0.01	234.85	93189	**0.00**	6.97	0.08	217.65	942
TR.25.2.1	5247	**0.00**	1.28	0.02	276.21	173390	5.13	14.09	0.06	244.25	9352
TR.25.2.5	2136	**0.00**	0.25	0.01	416.23	287946	**0.00**	10.36	0.06	419.59	2994
TR.25.5.1	2710	0.41	2.62	0.02	367.49	198719	**0.00**	7.49	0.06	379.71	3037
TR.25.5.2	2026	**0.00**	1.38	0.01	351.92	318794	3.16	10.27	0.04	366.34	3198
TR.50.2.1	7861	**0.00**	2.82	0.02	416.32	974202	11.54	20.35	0.06	334.34	10907
TR.50.2.5	2609	**0.00**	1.69	0.02	992.62	7026864	8.05	15.88	0.06	855.24	11526
TR.50.5.1	3737	**0.00**	1.48	0.02	714.23	1581300	13.46	20.45	0.05	560.58	11903
TR.50.5.2	2536	**0.00**	2.16	0.01	826.76	1947525	13.37	38.15	0.17	748.97	10150
TR.81.2.1	9980	**0.00**	2.63	0.02	509.33	2822916	17.67	22.70	0.03	414.48	15294
TR.81.2.5	2952	**0.00**	2.42	0.01	1498.39	15564091	14.13	88.72	0.37	1366.27	14093
TR.81.5.1	4514	**0.00**	2.55	0.02	999.22	6725910	17.74	46.78	0.16	793.17	11808
TR.81.5.2	2895	**0.00**	2.22	0.02	1523.66	17333433	24.97	95.37	0.34	1205.57	10878
TR.81.9.1	2985	**0.00**	2.83	0.01	1385.81	13383998	21.04	76.88	0.30	1084.03	6048
TR.81.9.2	2231	**0.00**	1.46	0.01	1699.58	12569225	18.92	96.10	0.28	1220.64	4046
TR.81.9.3	2110	**0.00**	**0.00**	**0.00**	1423.35	2678419	23.22	115.63	0.27	1193.06	3304
TR.81.9.4	2048	**0.00**	1.51	0.01	1169.25	3220798	18.99	91.90	0.25	914.55	2423
TR.81.9.5	2048	**0.00**	0.39	0.01	819.26	1766960	14.60	54.33	0.19	800.76	2224
AP.100.5.1	165546	**0.00**	3.50	0.03	921.97	25082641	24.07	52.13	0.21	773.95	22000
AP.100.5.2	102758	**0.00**	3.13	0.01	1530.19	86099359	10.78	61.80	0.29	1241.09	16151
AP.100.5.5	69161	**0.00**	0.26	**0.00**	1638.45	50401389	6.87	14.45	0.06	1680.30	8724
AP.200.10.1	132701	**0.00**	4.45	0.03	2240.03	119360795	70.54	226.04	0.34	1374.02	3427
AP.200.10.2	90055	**0.00**	3.09	0.01	3188.47	79650470	82.23	225.08	0.43	1633.18	2514
AP.200.10.5	74624	**0.00**	1.33	0.02	3306.81	13877348	153.56	257.99	0.28	1249.58	1327
	Avg	0.01	1.70	0.01	1036.03		20.50	60.33	0.16	772.81	

We obtain from Table 7.3 the results of SA and VNS metaheuristics where the GRD initial solution (Algorithm 6.4) is applied on the TR and AP dataset instances. It is shown

that in all the cases where GRD, GRD-RND, RND-GRD, GRB, and GRD-GRB initial solution strategies are used, in Table 7.3, Table 7.4, Table 7.5, Table 7.6, and Table 7.7, respectively, SA reaches the least cost (0.0% minimum gaps) in all the cases in the TR and AP dataset. In contrast, VNS does not find the least cost in most of the instances, particularly for the instances with more than 15 nodes, as it becomes larger than the minimum costs as the number of nodes n gets bigger. VNS starts by 0.0% then it reaches 23.57% and 116.32% at TR.81.9.1 and AP.200.10.5 problem instances in the TR and AP datasets, respectively. Moreover, the average of minimum gaps of SA is 0.00%, which is much less than the VNS's average (18.04%). Even though the average of minimum gaps of VNS for instances built with GRD initial solution method within the TR and AP datasets (Table 7.3) is better than the average of minimum gaps for instances that have RND initial solutions (Table 7.2), it does not reach as many best solutions as in the RND initial solution instances.

Concerning the average of initial solutions gaps related to the TR dataset GRD based instances, there is a slight variation between SA and VNS. This is shown in Table 7.3. The SA mean μ of initial solutions average gaps is 729.53%, but it is 554.50% for VNS. So, for SA and VNS, the GRD initial solution objectives, on average, are more than 7 and 5.5 times greater, respectively, than the discovered best costs. In other words, SA has a slight advantage here, where it tends to slightly improve the initial solutions. In general, it is noticed that the SA metaheuristic improved the initial solutions by more than 66% for all instances. However, the VNS metaheuristic improved them by at least 62%. In conclusion, VNS and SA have almost the same initial solution minimum improvement percentages (**Table *7.3***).

SA and VNS show less improvement percentages for GRD initial solutions minimum in **Table *7.3*** than RND initial solutions in **Table *7.2***. That is because the GRD initial solution is a deterministic algorithm in most of its parts. It improves the initial solution for a slight improvement or sometimes it provides significant improvement. So, most of the time, GRD strategy needs less improvement to reach the best solutions than RND initial solutions needs.

Table 7.3: *Greedy initial solution problem instances results*

Prob	Least Cost	SA					VNS				
		Min gap	Avg gap	Co. of Var.	Avg Init. Obj gap	Avg Iteration	Min gap	Avg gap	Co. of Var.	Avg Init. Obj gap	Avg Iteration
TR.10.2.1	3597	**0.00**	0.87	0.02	66.73	21101	**0.00**	5.99	0.04	62.40	162
TR.10.2.2	2331	**0.00**	**0.00**	**0.00**	87.53	20310	**0.00**	2.93	0.06	87.83	40
TR.10.3.1	2651	**0.00**	**0.00**	**0.00**	76.85	16769	**0.00**	2.53	0.03	68.24	38
TR.15.2.1	4128	**0.00**	1.04	0.02	96.04	43107	**0.00**	6.79	0.06	79.54	382
TR.15.2.2	2769	**0.00**	0.39	0.01	134.47	56007	**0.00**	6.32	0.05	140.25	567
TR.25.2.1	5247	**0.00**	2.41	0.02	172.75	145965	5.22	7.76	0.02	155.27	8333
TR.25.2.5	2136	**0.00**	0.69	0.01	216.54	253220	3.18	6.45	0.02	219.84	4473
TR.25.5.1	2721	**0.00**	1.77	0.01	239.86	200990	0.70	18.46	0.18	210.74	6330
TR.25.5.2	2031	**0.00**	1.45	0.01	178.04	341019	3.74	6.47	0.01	148.72	2592
TR.50.2.1	7797	**0.00**	2.92	0.02	242.53	1411573	11.86	19.68	0.05	231.33	8622
TR.50.2.5	2597	**0.00**	1.88	0.01	640.07	6760667	5.43	21.77	0.09	607.66	13298
TR.50.5.1	3709	**0.00**	3.78	0.03	427.52	1599012	11.08	28.38	0.15	427.10	11190
TR.50.5.2	2569	**0.00**	1.42	0.02	564.29	3873652	7.67	30.72	0.18	540.48	8684
TR.81.2.1	9823	**0.00**	4.29	0.02	302.23	2632723	8.48	23.32	0.05	286.94	12936
TR.81.2.5	2906	**0.00**	2.80	0.02	1213.53	17689600	17.31	61.83	0.32	894.02	13675
TR.81.5.1	4530	**0.00**	2.76	0.02	693.29	6009446	17.48	54.32	0.24	590.89	14048
TR.81.5.2	2896	**0.00**	3.23	0.02	1031.61	11885840	23.48	101.16	0.22	912.63	10847
TR.81.9.1	2978	**0.00**	3.37	0.01	899.08	18767338	23.57	97.39	0.34	849.44	6871
TR.81.9.2	2231	**0.00**	1.44	0.01	1273.33	20467172	20.98	104.38	0.39	913.83	4034
TR.81.9.3	2110	**0.00**	**0.00**	**0.00**	931.33	2608851	15.88	60.38	0.24	708.17	3048
TR.81.9.4	2048	**0.00**	2.09	0.01	747.26	3502655	20.12	81.91	0.25	617.28	2545
TR.81.9.5	2048	**0.00**	0.78	0.01	455.27	1776120	3.03	51.92	0.25	473.50	1941
AP.100.5.1	170123	**0.00**	2.68	0.03	578.46	41504738	18.37	54.64	0.19	651.69	22076
AP.100.5.2	103525	**0.00**	1.80	0.01	1153.64	74583544	8.17	47.94	0.27	991.22	17271
AP.100.5.5	69161	**0.00**	0.17	**0.00**	1167.52	62355574	5.95	34.91	0.27	1211.68	11476
AP.200.10.1	132714	**0.00**	3.49	0.02	1945.59	130086810	60.25	188.18	0.39	1158.33	3352
AP.200.10.2	93044	**0.00**	2.24	0.03	2482.48	82455551	96.93	255.85	0.35	1297.45	2402
AP.200.10.5	74624	**0.00**	0.40	0.01	2408.96	17472829	116.32	244.21	0.35	989.54	1315
	Avg	**0.00**	1.79	0.01	729.53		18.04	58.09	0.18	554.50	

Table 7.4 compares SA and VNS metaheuristics results where the GRD-RND initial solution methodology (Algorithm 6.7) was carried out on the TR and AP datasets problem instances. It is shown that VNS cannot find the least cost in all of the instances with $n > 15$, where it gives higher minimum costs than the best solutions' objectives as the number of nodes n increases. As seen in the minimum gap's column, VNS starts with 0.0% and hits 43.21% and 187.42% at TR.81.9.4 and AP.200.10.5 problem instances in the TR and AP datasets, respectively. Moreover, the VNS average of minimum gaps is 23.38%, while the SA average is still 0.0% which is much better.

According to Table 7.4, there is a big difference between SA and VNS; when we examine the average initial solutions gap for VNS, it is 757.75%, whereas it is 1041.90% for SA. So, for SA and VNS, the GRD-RND initial solution-based instances objectives are more than 10 and 7.5 times, respectively, than the discovered best costs. That makes it very close to the average of the RND solution instances (**Table *7.2***). As a result, SA yields a more significant advantage, as it helps in improving the initial solutions more than VNS. Generally, it is noticed that the SA metaheuristic improves the initial solutions by more than 105% for all instances. However, the VNS metaheuristic improves them by 87% or more. So, regarding GRD-RND initial solution minimum percentage of improvement, SA has higher (better) percentages of improvement for most of the problem instances (Table 7.4).

Table 7.4: *Greedy-Random initial solution problem instances results*

Prob	Least Cost	SA					VNS				
		Min gap	Avg gap	Co. of Var.	Avg Init. Obj gap	Avg Iteration	Min gap	Avg gap	Co. of Var.	Avg Init. Obj gap	Avg Iteration
TR.10.2.1	3597	**0.00**	0.96	0.02	105.09	22861	4.36	7.77	0.02	87.43	592
TR.10.2.2	2331	**0.00**	**0.00**	**0.00**	121.66	15797	**0.00**	2.16	0.04	120.64	40
TR.10.3.1	2651	**0.00**	**0.00**	**0.00**	136.27	20428	**0.00**	5.46	0.05	121.34	20
TR.15.2.1	4128	**0.00**	0.73	0.01	162.11	49083	**0.00**	5.57	0.03	183.79	251
TR.15.2.2	2769	**0.00**	0.39	0.01	224.19	53250	**0.00**	7.78	0.07	221.85	1048
TR.25.2.1	5266	**0.00**	0.86	0.01	272.57	144391	3.55	7.93	0.03	256.61	3373
TR.25.2.5	2136	**0.00**	0.87	0.01	404.82	329202	2.29	5.77	0.03	389.59	5169
TR.25.5.1	2710	**0.00**	1.35	0.01	422.39	206755	2.55	15.34	0.08	347.13	1339

TR.25.5.2	2031	**0.00**	1.40	0.01	356.04	249783	5.02	7.76	0.02	290.68	3613
TR.50.2.1	7875	**0.00**	1.81	0.01	374.16	700357	6.02	18.26	0.06	344.34	8567
TR.50.2.5	2604	**0.00**	1.63	0.01	946.41	5233344	15.86	30.92	0.20	801.39	10565
TR.50.5.1	3713	**0.00**	3.93	0.03	666.17	1958966	14.09	29.61	0.10	573.45	10676
TR.50.5.2	2525	**0.00**	3.55	0.02	882.63	2693155	7.68	19.68	0.11	845.65	7796
TR.81.2.1	9931	**0.00**	3.75	0.02	515.93	2889165	13.89	27.72	0.06	450.35	16105
TR.81.2.5	2962	**0.00**	2.02	0.01	1455.78	19001259	18.80	75.26	0.35	1160.91	12424
TR.81.5.1	4498	**0.00**	2.18	0.02	1082.61	9008565	18.36	65.00	0.22	811.05	13845
TR.81.5.2	2893	**0.00**	3.04	0.02	1467.78	7562753	21.09	86.34	0.33	1262.21	10712
TR.81.9.1	3024	**0.00**	1.79	0.01	1240.05	16528538	25.53	84.56	0.23	1079.74	7123
TR.81.9.2	2232	**0.00**	1.28	0.01	1559.12	17648705	24.19	76.64	0.36	1279.45	4089
TR.81.9.3	2107	**0.00**	0.13	0.00	1501.09	4259082	38.35	96.82	0.32	1027.83	3334
TR.81.9.4	2048	**0.00**	2.34	0.01	1129.47	4536458	43.21	89.20	0.13	858.06	2587
TR.81.9.5	2048	**0.00**	0.34	0.01	856.31	1828865	9.57	55.32	0.17	752.49	2155
AP.100.5.1	166863	**0.00**	6.47	0.04	945.05	41037191	18.28	39.59	0.20	717.12	19875
AP.100.5.2	102583	**0.00**	5.07	0.03	1513.53	65151743	12.29	75.25	0.42	1288.00	17454
AP.100.5.5	69161	**0.00**	0.35	**0.00**	1766.65	72173283	8.32	17.22	0.05	1601.88	8732
AP.200.10.1	136682	**0.00**	3.13	0.04	2192.99	127470429	64.29	211.13	0.42	1402.49	3326
AP.200.10.2	90400	**0.00**	3.66	0.02	3393.13	82254142	89.71	299.72	0.36	1767.13	2427
AP.200.10.5	74624	**0.00**	0.07	**0.00**	3479.17	15642566	187.42	248.67	0.20	1174.50	1319
	Avg	**0.00**	1.90	0.01	1041.90		23.38	61.16	0.17	757.75	

It is seen in Table 7.5 that VNS does not find the least cost in the majority of the instances, which are based on RND-GRD initial solution and implement Algorithm 6.8. Specifically, for those with more than 15 nodes, except for TR.25.2.5 instance, VNS starts at 0.0% and reaches 32.04% and 130.59% at TR.81.9.3 and AP.200.10.1 problem instances in the TR and AP datasets, respectively. Furthermore, the average of minimum gaps of SA is 0.00%, which is incomparable to the VNS's average of 18.81%. Therefore, even though the average of minimum gaps of VNS in Table 7.5 is better than the average of minimum gaps for instances based on RND initial solutions in Table 7.2, it does not reach as many least costs as the RND-based initial solution instances.

In the TR dataset instances where RND-GRD initial solution was used, there is a considerable disparity between SA and VNS in the average initial solutions gap in Table

7.5. The SA average of initial solutions average gaps is 582.60%, but it is 434.80% for VNS. So, for SA and VNS, the RND-GRD initial solution objectives, on average, are more than 5.5 and 4 times greater, respectively, than the reached best costs. As a result, SA is better in this case, where it improves the initial solutions more than VNS by 100% or more, on average. In general, it is noticed that the SA metaheuristic improves the initial solutions in the TR and AP datasets by more than 58% for all instances. However, the VNS metaheuristic improves them by at least 56%. So, regarding the RND-GRD initial solution minimum percentage of improvement, both have almost the exact values, as shown in **Table 7.5**.

Table 7.5: *Random-Greedy initial solution problem instances results*

Prob	Least Cost	SA					VNS				
		Min gap	Avg gap	Co. of Var.	Avg Init. Obj gap	Avg Iteration	Min gap	Avg gap	Co. of Var.	Avg Init. Obj gap	Avg Iteration
TR.10.2.1	3597	**0.00**	0.44	0.01	58.29	19626	**0.00**	4.96	0.03	56.20	32
TR.10.2.2	2331	**0.00**	**0.00**	**0.00**	97.17	19286	**0.00**	3.41	0.04	80.02	65
TR.10.3.1	2651	**0.00**	**0.00**	**0.00**	83.12	19889	**0.00**	6.01	0.06	75.37	87
TR.15.2.1	4128	**0.00**	2.99	0.02	87.59	44082	**0.00**	6.29	0.04	88.01	7046
TR.15.2.2	2769	**0.00**	0.92	0.02	135.88	57351	3.86	9.47	0.04	138.79	1826
TR.25.2.1	5247	**0.00**	0.86	0.01	110.68	162182	4.63	8.97	0.04	131.08	10743
TR.25.2.5	2136	**0.00**	1.15	0.01	278.81	384528	**0.00**	14.87	0.12	242.27	2004
TR.25.5.1	2710	**0.00**	2.00	0.01	189.51	233931	13.47	18.19	0.03	173.55	2568
TR.25.5.2	2042	**0.00**	1.12	0.01	172.93	313575	2.40	9.57	0.07	162.95	2839
TR.50.2.1	7919	**0.00**	2.37	0.02	161.53	701317	4.13	15.33	0.06	178.58	10535
TR.50.2.5	2571	**0.00**	2.22	0.01	445.02	5385418	7.58	31.57	0.20	477.35	10344
TR.50.5.1	3696	**0.00**	2.58	0.02	312.88	1498131	10.23	36.05	0.19	298.65	9467
TR.50.5.2	2547	**0.00**	1.83	0.02	442.64	3307866	9.23	36.73	0.15	401.47	7760
TR.81.2.1	9922	**0.00**	3.68	0.02	257.95	3759052	14.36	20.84	0.04	222.12	13270
TR.81.2.5	2946	**0.00**	1.76	0.01	911.54	37548078	13.85	79.25	0.34	785.41	13818
TR.81.5.1	4495	**0.00**	2.71	0.02	573.87	8360778	10.26	50.89	0.19	483.54	14487
TR.81.5.2	2913	**0.00**	2.14	0.01	775.36	20706028	18.09	59.30	0.30	609.14	11045
TR.81.9.1	3013	**0.00**	2.69	0.01	704.43	12569149	12.08	51.21	0.32	599.06	6528
TR.81.9.2	2228	**0.00**	1.12	0.01	864.53	21476415	29.49	120.16	0.38	685.58	3771

TR.81.9.3	2107	**0.00**	0.13	0.00	796.31	4019653	32.04	88.98	0.26	534.94	3144
TR.81.9.4	2048	**0.00**	1.56	0.01	635.18	2043729	15.48	83.40	0.26	525.25	2708
TR.81.9.5	2048	**0.00**	0.69	0.01	488.05	2317567	4.64	36.64	0.16	374.71	1864
AP.100.5.1	168201	**0.00**	4.05	0.03	552.74	49502005	12.18	35.92	0.17	492.01	21979
AP.100.5.2	102130	**0.00**	3.35	0.02	877.00	114033193	16.63	54.57	0.32	743.18	17208
AP.100.5.5	69161	**0.00**	0.36	**0.00**	1105.23	64250431	10.26	26.79	0.17	1005.50	10201
AP.200.10.1	131719	**0.00**	3.63	0.02	1425.77	114793741	130.59	223.16	0.26	606.55	3478
AP.200.10.2	91092	**0.00**	3.12	0.02	1910.36	77148908	64.85	226.77	0.43	1075.28	2462
AP.200.10.5	74624	**0.00**	0.03	**0.00**	1858.55	17009826	86.33	313.46	0.43	927.95	1332
	Avg	**0.00**	1.77	0.01	582.60		18.81	59.74	0.18	434.80	

Table 7.6 and Table 7.7 show and compare SA and VNS metaheuristics results. The GRB-based (Algorithm 6.9) and GRD-GRB (Algorithm 6.11) initial solutions were implemented on the TR and AP datasets. It is shown in both tables that VNS does not reach the least cost in most of the problem instances, particularly for instances with $n > 15$, except for the last two instances (TR.81.9.4 and TR.81.9.5). However, some of the other instances' minimum gaps are tolerable enough (less than 2% for GRB-based instances and less than 4% for GRD-GRB based instances).

VNS starts at a 0.0% minimum gap and reaches 14.76% and 15.44% at TR.81.5.2 problem instance for GRB and GRD-GRB based instances, respectively, in the TR dataset instances in **Table *7.6*** and **Table *7.7***. While with the AP dataset instances, the minimum gap reaches 21.29% and 28.93% for GRB and GRD-GRB based instances, respectively. AP dataset instances have bigger maximums of minimum gaps because they have more nodes than TR instances. Thus, VNS does not get the best solutions as n increases, for GRB and GRD-GRB initial solutions-based instances, as shown in the minimum gap column in **Table *7.6*** and **Table *7.7***. Nonetheless, it has a very acceptable average of minimum gaps (5.69% and 6.33% for GRB and GRD-GRB based initial solutions, respectively) compared to the instances built on other methods of initial solutions. However, the best average of minimum gaps in the previous results tables (**Table *7.2*** to **Table 7.5**) is 18.04%, in **Table *7.3*** (problem instances based on GRD initial solution strategy). This is around triple the average of minimum gaps of GRB and GRD-GRB

based initial solution instances. In other words, instances with GRB and GRD-GRB initial solution methods are closer to their best solutions than the other initial solution strategies.

VNS shows a huge improvement in the average gap in **Table *7.6*** and **Table *7.7*** compared to the previous results of instances built on RND, GRD, GRD-RND, and RND-GRD initial solutions. In **Table *7.6*** and **Table *7.7***, the averages of gaps of VNS start from 0.1% and 0.0%, respectively, at TR.81.9.5 problem instance, which are quite tolerable averages gaps. However, for other instances in the TR dataset, they reach 22.83% and 22.91%, respectively, at TR.81.5.2 problem instance. While, within AP dataset instances, they reach 27.03% and 34.33% for GRB and GRD-GRB based initial solutions, respectively. Although these are considered high averages in both datasets, compared to the maximum gap averages of results for problem instances built with RND, GRD, GRD-RND, and RND-GRD initial solutions, 257.99%, 255.85%, 299.72%, and 313.46%, respectively, they are deemed to be very low. Also, in **Table *7.6*** and **Table *7.7***, the means μ of average gaps are 9.93% and 10.25%, which are pretty low compared to 60.33%, 58.09%, 61.16%, and 59.74%, for the instances with the previously mentioned initial solution strategies.

On the other hand, as usual, SA has a much lower average gaps for instances with a large number of nodes n, specifically between 0.0% - 5.6% for GRB-based instances and 0.0% - 4.93% for GRD-GRB based instances. This means that the SA solutions' objectives are generally closer to the least costs than the VNS solutions. Notice in **Table *7.6*** that the mean μ of average gaps for SA, which is 2.05%, is the worst (largest) mean μ of average gaps between all the tested problem instances for SA.

In **Table *7.6*** and **Table *7.7*** (GRB and GRD-GRB based initial solutions), the coefficient of variation values are slightly better in SA than VNS. They are $0.01\% \approx 0$ in SA, similar to the averages of instances with RND, GRD, GRD-RND, and RND-GRD initial solutions in **Table *7.2*** to **Table 7.5**. In contrast, VNS shows a major improvement where the averages of coefficient of variation values in GRB and GRD-GRB based initial solutions are 0.03% and 0.02%, respectively. These are lower than the averages for instances built with the above-mentioned initial solutions (RND, GRD, GRD-RND, and RND-GRD), where they are between 0.16% and 0.18%. As a result, in the GRB and

GRD-GRB cases, the SA and VNS best costs for each instance are more condensed around the mean (very close to each other around the mean μ).

The instances that conduct GRB and GRD-GRB initial solution strategies in **Table *7.6*** and **Table *7.7*** show that SA and VNS averages of initial solutions gaps are close to each other. The SA means μ of initial objectives average gaps are 45.42% and 48.77% for GRB and GRD-GRB based initial solutions, respectively. For VNS, they are 37.72% and 39.26%, respectively. So, SA outperforms VNS, where it improves the initial solutions more.

Since GRB and GRD-GRB initial solutions are extremely improved by Gurobi Optimizer when picking hubs and assigning non-hub nodes to hubs, SA and VNS only improve the solutions by a small percentage to reach their least (better) costs. In general, it is noticed that the SA metaheuristic improves the initial solutions by more than 14.9% and 10% for the problem instances based on GRB and GRD-GRB, respectively. However, the VNS metaheuristic improves them by at least 6.5% and 5%, respectively. So, regarding these minimum percentages of improvement, SA outperforms (see Table 7.6 and Table 7.7).

Table 7.6: *Gurobi-based initial solution problem instances results*

Prob	Least Cost	SA					VNS				
		Min gap	Avg gap	Co. of Var.	Avg Init. Obj gap	Avg Iteration	Min gap	Avg gap	Co. of Var.	Avg Init. Obj gap	Avg Iteration
TR.10.2.1	3597	**0.00**	**0.00**	**0.00**	47.71	20786	**0.00**	2.52	0.02	47.71	39
TR.10.2.2	2331	**0.00**	**0.00**	**0.00**	71.13	17661	**0.00**	4.03	0.05	71.13	78
TR.10.3.1	2651	**0.00**	**0.00**	**0.00**	51.83	17426	**0.00**	2.50	0.04	51.83	113
TR.15.2.1	4128	**0.00**	0.95	0.01	34.35	46585	**0.00**	5.00	0.04	34.35	486
TR.15.2.2	2769	**0.00**	**0.00**	**0.00**	50.23	51277	**0.00**	9.11	0.10	50.23	366
TR.25.2.1	5266	**0.00**	2.99	0.02	28.96	136261	3.49	7.24	0.03	24.61	8036
TR.25.2.5	2136	**0.00**	0.60	0.01	23.36	333694	1.17	4.26	0.02	21.93	1102
TR.25.5.1	2721	**0.00**	1.86	0.01	31.20	217771	1.73	5.55	0.03	28.97	1624
TR.25.5.2	2031	**0.00**	0.66	0.01	54.26	356891	0.79	3.88	0.02	53.05	5198
TR.50.2.1	7912	**0.00**	2.57	0.02	16.29	2762613	4.27	5.95	0.01	11.53	7408

TR.50.2.5	2572	**0.00**	3.24	0.02	30.87	2992098	6.03	8.77	0.02	23.43	5234
TR.50.5.1	3696	**0.00**	3.53	0.03	37.61	2027856	11.01	14.34	0.02	23.96	5777
TR.50.5.2	2521	**0.00**	2.53	0.02	64.22	2319834	7.38	11.29	0.04	52.94	6025
TR.81.2.1	9846	**0.00**	5.60	0.03	14.91	3774488	7.87	9.41	0.01	6.52	11130
TR.81.2.5	2905	**0.00**	3.55	0.02	33.29	21561828	9.81	14.76	0.02	21.38	5827
TR.81.5.1	4502	**0.00**	2.42	0.01	39.94	12002209	10.55	16.65	0.03	26.58	10271
TR.81.5.2	2853	**0.00**	4.42	0.02	103.33	19103802	14.76	22.83	0.05	77.18	9432
TR.81.9.1	3002	**0.00**	2.91	0.01	48.30	8767812	10.93	15.51	0.03	33.69	3847
TR.81.9.2	2239	**0.00**	1.43	0.01	63.78	13771069	6.92	10.70	0.02	53.17	3592
TR.81.9.3	2105	**0.00**	0.20	**0.00**	37.58	3357996	1.24	6.83	0.03	35.90	1097
TR.81.9.4	2048	**0.00**	1.90	0.01	29.05	2699321	**0.00**	2.90	0.03	29.05	1310
TR.81.9.5	2048	**0.00**	0.11	**0.00**	34.67	1848715	**0.00**	0.10	**0.00**	34.67	605
AP.100.5.1	169648	**0.00**	5.31	0.06	35.14	19385362	7.89	9.57	0.01	25.26	17070
AP.100.5.2	103859	**0.00**	1.95	0.01	89.53	127540223	4.25	13.51	0.05	81.81	15788
AP.100.5.5	69161	**0.00**	0.26	**0.00**	30.23	69824287	6.46	8.48	0.01	22.33	3067
AP.200.10.1	132504	**0.00**	4.69	0.04	53.84	116468231	21.29	27.03	0.04	26.83	2690
AP.200.10.2	91399	**0.00**	3.56	0.02	67.23	70960405	20.31	26.84	0.03	39.00	2296
AP.200.10.5	74624	**0.00**	0.07	**0.00**	48.95	15993970	1.29	8.46	0.04	47.05	318
	Avg	**0.00**	2.05	0.01	45.42		5.69	9.93	0.03	37.72	

Table 7.7: *Greedy-Gurobi initial solution problem instances results*

Prob	**Least Cost**	**SA**					**VNS**				
		Min gap	**Avg gap**	**Co. of Var.**	**Avg Init. Obj gap**	**Avg Iteration**	**Min gap**	**Avg gap**	**Co. of Var.**	**Avg Init. Obj gap**	**Avg Iteration**
TR.10.2.1	3597	**0.00**	**0.00**	**0.00**	33.69	19815	4.36	5.79	0.02	28.10	330
TR.10.2.2	2331	**0.00**	**0.00**	**0.00**	23.55	17798	**0.00**	9.98	0.05	23.55	21
TR.10.3.1	2651	**0.00**	**0.00**	**0.00**	46.28	22051	**0.00**	0.60	0.01	46.28	27
TR.15.2.1	4128	**0.00**	2.19	0.01	63.74	25277	**0.00**	2.06	0.03	63.74	590
TR.15.2.2	2769	**0.00**	0.65	0.01	50.45	59428	**0.00**	2.81	0.04	50.45	617
TR.25.2.1	5247	**0.00**	3.14	0.02	37.16	150802	3.87	6.72	0.01	32.06	5751
TR.25.2.5	2136	**0.00**	0.34	**0.00**	11.28	334309	3.89	4.94	0.01	7.12	1033
TR.25.5.1	2722	**0.00**	1.79	0.01	72.63	171685	4.81	9.69	0.03	64.70	1953
TR.25.5.2	2031	**0.00**	1.70	0.01	40.47	328994	2.95	5.78	0.02	36.44	1323

TR.50.2.1	7797	**0.00**	3.17	0.02	19.66	1318939	3.63	5.26	0.01	15.47	5639
TR.50.2.5	2604	**0.00**	1.54	0.01	13.67	8469048	7.76	7.94	**0.00**	5.49	2979
TR.50.5.1	3713	**0.00**	2.57	0.02	39.11	1522896	4.47	7.77	0.02	33.15	7915
TR.50.5.2	2528	**0.00**	2.26	0.01	76.23	5162362	6.76	12.54	0.03	65.06	6403
TR.81.2.1	9972	**0.00**	4.93	0.03	13.09	3175869	3.06	4.95	0.01	9.73	7270
TR.81.2.5	2925	**0.00**	2.16	0.01	12.68	14499875	6.22	7.72	0.01	6.08	4544
TR.81.5.1	4518	**0.00**	2.20	0.02	68.33	12860845	4.43	13.39	0.04	61.19	9674
TR.81.5.2	2862	**0.00**	3.22	0.02	87.95	22447460	15.44	22.37	0.04	62.80	6931
TR.81.9.1	2990	**0.00**	3.05	0.02	133.38	14033395	13.31	22.91	0.05	105.96	6607
TR.81.9.2	2231	**0.00**	1.13	0.01	64.90	18306178	12.42	17.23	0.03	46.69	3262
TR.81.9.3	2107	**0.00**	0.13	**0.00**	42.57	4655139	6.07	9.30	0.02	34.41	1848
TR.81.9.4	2048	**0.00**	1.20	0.01	35.84	2819831	**0.00**	3.40	0.02	35.84	1060
TR.81.9.5	2048	**0.00**	1.13	0.01	10.64	1922937	**0.00**	**0.00**	**0.00**	10.64	854
AP.100.5.1	169865	**0.00**	2.61	0.03	41.98	60240663	4.63	11.47	0.03	35.69	18593
AP.100.5.2	102191	**0.00**	2.52	0.01	87.66	114382719	14.68	17.84	0.02	63.63	14897
AP.100.5.5	69161	**0.00**	0.27	**0.00**	37.02	85636981	6.08	8.78	0.03	29.17	6163
AP.200.10.1	134637	**0.00**	3.88	0.02	69.74	126820027	19.47	28.62	0.04	42.07	3107
AP.200.10.2	88997	**0.00**	4.86	0.03	113.74	84661097	28.93	34.33	0.03	65.78	2479
AP.200.10.5	74624	**0.00**	0.73	0.01	18.16	17234408	0.05	2.73	0.02	18.11	901
	Avg	**0.00**	1.91	0.01	48.77		6.33	10.25	0.02	39.26	

The results of SA and VNS metaheuristics, alongside the 6 different initial solution methodologies (RND, GRD, GRD-RND, RND-GRD, GRB, and GRD-GRB) on the TR and AP datasets, are shown and compared in Table 7.8. SA reaches the best solution in all instances. However, VNS found the best solution only for 9 instances (all the 10-node and 15-node instances, TR.25.2.5, TR.25.5.1, TR.81.9.4, and TR.81.9.5), all of them are in the TR dataset. It did not reach a single least cost within AP instances. VNS starts at 0.0% minimum gap and reaches 14.76% at TR.81.5.2 problem instance in TR dataset and 23.56% in AP dataset. Even though VNS does not reach the best solutions in most of the instances, as shown in the minimum gap column in **Table *7.8***, it has a very tolerable average of minimum gaps (5.08%). So, according to the summarized results, VNS performed well but SA generally outperformed.

Initial Solutions columns in **Table *7.8*** show the average of each initial solution strategy's costs. As shown, it is clear how large these initial costs are compared to the least reached costs. The initial costs in bold mean that the initial solution strategy (RND, GRD, GRD-RND, RND-GRD, GRB, or GRD-GRB) was used by either SA or VNS metaheuristic to find the best solution for a problem instance, e.g., TR.10.2.1, and AP.100.5.1, etc. An initial solution cost with an asterisk (*) means that the algorithm/s (SA and/or VNS) with this initial solution strategy has/have reached the best solution more than once (in more than one run).

As seen in Table 7.8, within both TR and AP datasets, for problem instances with a small number of nodes n (10 and 15), in order to reach the least costs by using SA and/or VNS metaheuristics, any of the initial solution methods can be used. However, as the number of nodes of the instance becomes larger, fewer initial solution methods were able to reach the least costs, with some anomalies (TR.81.9.4 and TR.81.9.5). It is noticeable that the algorithm was able to reach the best solution by using the RND-based initial solutions in all the instances with a small number of nodes n (10, 15, and 25). Whereas, for instances with a large n (50 and 81 from the TR dataset and 100 and 200 from the AP dataset), GRB and RND-GRD were the most used initial solutions to reach the least cost. Moreover, the developed metaheuristics (SA and VNS) used the RND-GRD based initial solutions to reach the least costs in 18 different instances (marked in bold) out of 28 problem instances in the TR and AP datasets, and it is used more than once in 13 cases (marked in bold with an asterisk). So, RND-GRD is the most reliable initial solution strategy even for instances with a large n. Then comes GRB as the next one. Another advantage of RND-GRD based initial solution method is that: its building CPU time is less than a second (milliseconds). On the other hand, it takes hours to build a GRB-based initial solution with large n, as shown in Table 7.1.

Each initial solution strategy's results are shown above in Table 7.2 to Table 7.7 separately. In conclusion, from the overall results and the comparisons between the two metaheuristics (SA and VNS), we observe that SA outperformed VNS in terms of solution quality. Finally, after all the previous results and analysis, the best combination between an initial solution strategy and a metaheuristic that may produce an efficient and high-

performance algorithm and provide the best results compared to the other combinations is the RND-GRD initial solution methodology with the SA metaheuristic.

Table 7.8: *Final results*

Prob	Least Cost	Initial Solutions Costs						SA	VNS	CPU (s) per run
		RND	GRD	GRD-RND	RND-GRD	GRB	GRD-GRB	min gap	min gap	
TR.10.2.1	3597	**7490***	**5919***	**7207***	**5656***	**5313***	**4809***	**0.00**	**0.00**	10
TR.10.2.2	2331	**6136***	**4375***	**5155***	**4396***	**3989***	**2880***	**0.00**	**0.00**	10
TR.10.3.1	2651	**5885***	**4574***	**6066***	**4752***	**4025***	**3878***	**0.00**	**0.00**	10
TR.15.2.1	4128	**11116***	**7752***	**11268***	**7752***	**5546***	**6759***	**0.00**	**0.00**	30
TR.15.2.2	2769	**9034***	**6573***	**8944***	**6700***	**4160***	**4166***	**0.00**	**0.00**	30
TR.25.2.1	5247	**19364***	**14202***	19533	**11870***	6791	**7197**	**0.00**	3.87	60
TR.25.2.5	2136	**11063***	**6905***	**10740***	**7701***	**2635***	**2377***	**0.00**	**0.00**	60
TR.25.5.1	2710	**12861**	8881	**13291***	**8129**	3570	4699	**0.00**	**0.00**	60
TR.25.5.2	2026	**9451**	5444	8798	5536	3133	2853	**0.00**	1.04	60
TR.50.2.1	7797	39336	**27803**	37219	21841	9201	**9330**	**0.00**	3.63	270
TR.50.2.5	2571	27717	19298	27222	**14991***	3366	2960	**0.00**	4.82	270
TR.50.5.1	3696	29218	20641	28488	**15750**	**5086**	5165	**0.00**	4.95	270
TR.50.5.2	2521	23956	17391	25262	13886	**4140**	4455	**0.00**	7.06	270
TR.81.2.1	9823	60615	**40372**	61706	36033	11314	11277	**0.00**	4.62	1000
TR.81.2.5	2905	48292	36029	45227	29748	**3872**	3296	**0.00**	6.95	1000
TR.81.5.1	4495	48545	36353	50849	**29605**	6300	7605	**0.00**	4.96	1000
TR.81.5.2	2853	47120	34492	46537	24947	**5801**	5379	**0.00**	14.76	1000
TR.81.9.1	2978	43566	**32346**	42653	23922	4452	6978	**0.00**	11.82	1000
TR.81.9.2	2228	37593	29001	37635	**22077***	3667	3679	**0.00**	7.45	1000
TR.81.9.3	2105	32881	20760	33306	18275	**2896**	3004	**0.00**	1.24	1000
TR.81.9.4	2048	**25359***	**17498***	**26640***	**14922***	**2643***	**2782***	**0.00**	**0.00**	1000
TR.81.9.5	2048	**19984***	**11736***	**19358***	**11108***	**2758***	**2266***	**0.00**	**0.00**	1000
AP.100.5.1	165546	**1743414**	1333960	1678262	1107497	229265	241168	**0.00**	7.36	3600
AP.100.5.2	102130	1600916	1259892	1627005	**1001073**	196846	191768	**0.00**	6.01	3600
AP.100.5.5	69161	**1259118***	**918882***	**1282976***	**838272***	**90071***	**94763***	**0.00**	5.95	3600
AP.200.10.1	131719	3220506	2695473	3254028	**2077842**	203840	228531	**0.00**	22.01	7200
AP.200.10.2	88997	2902837	2481692	3179936	1798056	152843	**190222**	**0.00**	23.56	7200

AP.200.10.5	74624	**2547982***	**1815563***	**2702243***	**1445456***	**111153***	**88178***	**0.00**	0.05	7200
							Avg	**0.00**	5.08	

Since pHCRP is an NP-hard problem, the optimality of instances with only a small number of nodes can be found. Thus, in order to evaluate this study's (metaheuristics with the initial solutions) performance and results, we compare it to the previous studies that have the results of single assignment pHCRP on the TR and AP datasets instances. So, we compared them with the best results found by Kartal et al. (2019). They made their tests on the same datasets (TR and AP) using CPLEX Optimizer only for 10-node (found optimal costs) and 15-node instances. While they developed two heuristics based on Ant Colony System (ACS) and Discrete Particle Swarm Optimization (DPSO) that ran on all the 10, 15, 25, 50, and 81-node problem instances from the TR dataset and 100-node and 200-node problem instances from the AP dataset that we used in our study as seen in **Table *7.8***, except TR.81.9.1, TR.81.9.2, TR.81.9.3, TR.81.9.4, TR.81.9.5; therefore, we did not include them in the comparison table (**Table *7.9***). However, indicate that, for the AP dataset, they took the coefficients as 3.0 for collection, 2.0 for distribution and 0.75 for transfer. For the sake of simplicity, we took all the coefficients as 1.0 so the results are not comparable. Kartal et al. (2019) used the same time limits per run; thus, the comparison with their results is presented in **Table *7.9***.

The developed SA and VNS metaheuristics combined with the proposed initial solutions have produced high-quality results compared to Kartal et al.'s best results (see **Table *7.9***). Firstly, our best solutions have got the optimal costs of the 10-node instances as in theirs. We also found the same costs as their least costs for the 15-node instances. Then, for instances with a large n (50 and 81), this study's least costs are lower (better) than Kartal et al.'s results, starting from TR.25.2.5. They are marked in bold in **Table *7.9***.

The percentage of improvement over Kartal et al.'s least costs increases as the number of nodes increases. Therefore, the TR dataset instances with 81 nodes have the highest improvement percentage among the other instances in the TR dataset, which get 25% at TR.81.5.2. Then come the 50-nodes instances that attain 18% at TR.50.5.2. After that are the 25-node instances with a maximum improvement of 4%. Nonetheless, there are some anomalies, specifically, with the instances that have two hubs p and one vehicle

v in TR.25.2.1, TR.50.2.1, and TR.81.2.1 instances, as they have less percentage of improvement, 0%, 2%, and 2%, respectively. Also, the best metaheuristic and initial solution combinations are shown. Finally, it is seen that this study's results have shown a quite significant improvement over the previous research studies.

Table 7.9: *Comparison with Kartal et al.'s results*

Prob	**Least Cost**	**Kartal et al.** (2019) **Least Cost**	**Improvement**	**Best Metaheuristic&Init. Sol. Combination/s**	**CPU (s) per run**
TR.10.2.1	3597	3597 (optimal)	0%	All	10
TR.10.2.2	2331	2331 (optimal)	0%	All	10
TR.10.3.1	2651	2651 (optimal)	0%	All	10
TR.15.2.1	4128	4128	0%	All	30
TR.15.2.2	2769	2769	0%	All	30
TR.25.2.1	5247	5247	0%	SA&all except GRD-RND and GRB	60
TR.25.2.5	**2136**	2219	**4%**	SA&all, VNS&RND, VNS&GRD	60
TR.25.5.1	**2710**	2779	**3%**	SA&GRD-RND, RND-GRD, VNS&RND	60
TR.25.5.2	**2026**	2112	**4%**	SA&RND	60
TR.50.2.1	**7797**	7919	**2%**	SA&GRD	270
TR.50.2.5	**2571**	3003	**17%**	SA&RND-GRD	270
TR.50.5.1	**3696**	4233	**15%**	SA&RND-GRD, GRB	270
TR.50.5.2	**2521**	2970	**18%**	SA&GRB	270
TR.81.2.1	**9823**	10,020	**2%**	SA&GRD	1000
TR.81.2.5	**2905**	3515	**21%**	SA&GRB	1000
TR.81.5.1	**4495**	5568	**24%**	SA&RND-GRD	1000
TR.81.5.2	**2853**	3575	**25%**	SA&GRB	1000

8. CONCLUSIONS AND FUTURE WORKS

Today, businesses that serve using a hub network structure try to keep their customer satisfaction at a high level while trying to reduce their costs. In this study, solution algorithms for single assignment p-hub center and routing network design problem have been developed and tested for businesses using a hub network structure. The developed algorithms include SA and VNS metaheuristics running on top of 6 different initial solution strategies developed randomly, greedily, using Gurobi Optimizer, and a mix of those. The objective function of the algorithms developed for the pHCRP is the minimization of the maximum transportation cost. The metaheuristics are carried out on the TR and AP datasets.

In this study, we compared all the proposed initial solutions (RND, GRD, GRD-RND, RND-GRD, GRB, and GRD-GRB) combined with the SA and VNS metaheuristics. When we compared the two algorithms, we observed that SA performed much better than VNS. From the numerical results, it was seen that the proposed algorithms are able to find feasible solutions with decent costs for problem instances with a small number of nodes as well as large-sized problems. After all the analysis that has been done on the results, it was shown that RND-GRD initial solution methodology with the SA metaheuristic form the best combination that gives an efficient and high-performance algorithm and provides the best results compared to the other combinations.

This study has outperformed the previous researches which were carried out using different models and metaheuristics on the same problem instances and datasets. So, this study has important findings that can be used by the managers of cargo companies in the decision-making process.

Future studies may focus on implementing other metaheuristics on different datasets and comparing the results with the initial solutions, and SA and VNS results. Furthermore, future research will seek to work on the incomplete p-hub center and routing problem modeling and developing mat-heuristics. Finally, multi assignment pHCRP solution algorithms using SA and VNS metaheuristics combined with multiple initial solution strategies could be considered in the future.

REFERENCES

Alumur, S., & Kara, B. (2008). Network Hub Location Problems: The State of the Art. *European Journal of Operational Research, 190*(1), 1-21.

Alumur, S., Kara, B., & Karasan, O. (2009). The design of single allocation incomplete hub networks. *Transportation Research Part B, 43*, 936-951.

Alumur, S., Kara, B., & Karasan, O. (2012). Multimodal hub location and hub network design. *Omega, 40*, 927-939.

Aykin, T. (1990). On a Quadratic İnteger Program for the Location of Interacting Hub Facilities. *European Journal of Operational Research, 46*(3), 409–411.

Bashiri, M., Mirzaei, M., & Randall, M. (2013). Modeling fuzzy capacitated p-hub center problem and a genetic algorithm solution. *Applied Mathematical Modelling, 37*(5), 3513-3525.

Beasley, J. (1990). OR-Library: distributing test problems by electronic mail. *Journal of the Operational Research Society, 41*(11), 1069-1072.

Bruns, A., Klose, A., & Stahly, P. (2000). Restructuring of swiss parcel delivery services. *OR-Spektrum, 22*(2), 285–302.

Bryan, D., & O'Kelly, M. (1999). Hub-and-spoke networks in air transportation: An analytical review. *Journal of Regional Science, 39*(2), 275-295.

Camargo, R., Miranda, G., & Lokketangen, A. (2013). A new formulation and an exact approach for the many-to-many hub location-routing problem. *Appl Math Model, 37*(1213), 7465–7480.

Campbell, A., Lowe, T., & Zhang, L. (2007). The p-hub center allocation problem. *European Journal of Operational Research, 176*(2), 819-835.

Campbell, J. (1994a). A survey of network hub location. *Studies in Locational, 6*, 31–49.

Campbell, J. (1994b). Integer Programming Formulations of Discrete Hub Location Problems. *European Journal of Operational Research, 72*, 387–405.

Campbell, J. (1996). Hub Location and the P-Hub Median Problem. *Operations Research, 44*(6), 923-935.

Campbell, J., Ernst, A., & Krishnamoorthy, M. (2005a). Hub arc location problems: Part I–Introduction and Results. *Management Science, 51*(10), 1540–1555.

Campbell, J., Ernst, A., & Krishnamoorthy, M. (2005b). Hub arc location problems: Part II–formulations and optimal algorithms. *Management Science, 51*(10), 1556–1571.

Cetiner, S., Sepil, C., & Sural, H. (2010). Hubbing and routing in postal delivery systems. *Ann Oper Res, 181*(1), 109–124.

Dueck, G., & Scheuer, T. (1990). Threshold Accepting: A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing. *J. Comput. Phys., 90*(1), 161–175.

Eren, Y., Kucukdemiral, I. B., & Ustoglu, I. (2017). Introduction to optimization. In O. Erdinc, *Optimization in Renewable Energy Systems: Recent Perspectives* (pp. 27-74). Butterworth-Heinemann.

Ernst, A., & Krishnamoorthy, M. (1996). Efficient algorithms for the uncapacitated single allocation p-hub median problem. *Location Science, 4*(3), 139–154.

Ernst, A., Campbell, J., & Krishnamoorthy, M. (2002). *Facility location : application and theory.* Berlin: Springer.

Ernst, A., Hamacher, H., Jiang, H., Krishnamoorthy, M., & Woeginger, G. (2009). Uncapacitated single and multiple allocation p-hub center problems. *Computers and Operations Research, 36*(7), 2230-2241.

Farahani, R., Hekmatfar, M., Arabani, A., & Nikbakhsh, E. (2013). Hub location problems: A review of models, classification, solution techniques, and applications. *Computers and Industrial Engineering, 64*(4), 1096-1109.

Hakimi, S. (1964). Optimum location of switching centers and the absolute centers and medians of a graph. *Operations Research, 12*, 450-459.

Hakimi, S. (1965). Optimum location of switching centers in a communications network and soma related graph theoretic problems. *Operations Research, 13*, 462-475.

Hansen, P., & et al. (2010). Variable Neighbourhood Search: Methods and Applications. *Annals of Operations Research, 175*(1), 367–407.

Hansen, P., Mladenović, N., Todosijević, R., & et al. (2017). Variable neighborhood search: basics and variants. *EURO Journal on Computational Optimization, 5*(3), 423-454.

Ingber, L. (1993). Simulated Annealing: Practice Versus Theory. *Mathematical and Computer Modelling, 18*, 29-57.

Kara, B. a. (2000). On the single-assignment p-hub center problem. *European Journal of Operational Research, 125*(3), 648-655.

Kara, B. a. (2001). The latest arrival hub location problem. *Management Science, 47*(10), 1408-1420.

Kartal, Z., Hasgul, S., & Ernst, A. (2017). Single allocation p-hub median location and routing problem with simultaneous pick-up and delivery. *Transp Res Part E Logist Transp Rev, 108*, 141–159.

Kartal, Z., Krishnamoorthy, M., & Ernst, A. (2019). Heuristic algorithms for the single allocation p-hub center problem with routing considerations. *OR Spectrum, 41*, 99-145.

Kirkpatrick, S., Gelatt, C., & Vecchi, M. (1983). Optimization by Simulated Annealing. *Science (New York, N.Y.), 220*, 671-80.

Kirlik, G., & Ceyda, O. (2012). A Variable Neighborhood Search for Minimizing Total Weighted Tardiness with Sequence Dependent Setup Times on a Single Machine. *Computers & Operations Research, 39*(7), 1506-1520.

Klincewicz, J. (1991). Heuristics for the P-Hub Location Problem. *European Journal of Operational Research, 53*(1), 25-37.

Klincewicz, J. (1998). Hub location in backbone/tributary network design: A review. *Location Science, 6*(4), 307–335.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., & Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics, 21*, 1087-1092.

Meyer, T., Ernst, A., & Krishnamoorthy, M. (2009). A 2-phase algorithm for solving the single allocation p-hub center problem. *Computers and Operations Research, 36*(12), 3143-3151.

Miller, C., Tucker, A., & Zemlin, R. (1960). Integer programming formulation of traveling salesman problems. *Journal of the ACM, 7*, 326-329.

Mladenovic, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research, 24*(11), 1097.

Nagy, G., & Salhi, S. (1998). The many-to-many location-routing problem. *Top, 6*(2), 261–275.

Nickel, S., Schöbel, A., & Sonneborn, T. (2001). Hub location problems in urban traffic networks. *Mathematics Methods and Optimization in Transportation Systems*, 1–12.

O'Kelly, M. (1986b). Activity Levels at Hub Facilities in Interacting Networks. *Geographical Analysis, 18*(4), 343-356.

O'Kelly, M. (1986a). The Location of Interacting Hub Facilities. *Transportation Science, 20*(2), 92-106.

O'Kelly, M. (1987). A Quadratic Integer-Program for the Location of Interacting Hub Facilities. *European Journal of Operational Research, 32*(3), 393-404.

O'Kelly, M. (1998). Hub location with flow economies of scale. *Transportation Research Part B-Methodological, 32*(8), 605-616.

O'Kelly, M., & Miller, H. (1994). The hub network design problem: A review and synthesis. *Journal of Transport Geography, 2*(1), 31–40.

Podnar, H., Skorin-Kapov, J., & Skorin-Kapov, D. (2002). Network cost minimization using threshold-based discounting. *European Journal of Operational Research, 137*, 371–386.

Rieck, J., Ehrenberg, C., & Zimmermann, J. (2014). Many-to-many location-routing with inter-hub transport and multi-commodity pickup-and-delivery. *European Journal of Operational Research, 236*(3), 863–878.

Serper, E., & Alumur, S. (2016). The design of capacitated intermodal hub networks with different vehicle types. *Transportation Research Part B, 86*, 51-65.

Tan, P., & Kara, B. (2007). A hub covering model for cargo delivery systems. *Networks, 49*(1), 28-39.

Toh, R. S., & Higgins, R. C. (1985). The impact of hub and spoke network centralization and route monopoly on domestic airline profitability. *Transportation Journal, 24*, 16–27.

Uçar, D. (2020). Tek atamali sabit yerleşimli p-ana dağitim üssü merkez ve rotalama problemi; ve ağ tasarimi. Master's degree thesis, Eskişehir Technical University, Eskişehir.

Wagner, B. N. (2004). A note on "the latest arrival hub location problem". *Management Science, 50*(12), 1751-1752.

Wasner, M., & Zapfel, G. (2004). An integrated multi-depot hub-location vehicle routing model for network planning of parcel service. *Int J Prod Econ, 90*(3), 403–419.

Yaman, H., & Elloumi, S. (2012). Star p-hub center problem and star p-hub median problem with bounded path lengths. *Computers and Operations Research, 39*(11), 2725-2732.