The Distributed p-Median Problem in Computer Networks

Anas A. ALDABBAGH

Submitted in Partial Fulfilment of the Requirement of the Degree of

Doctor of Philosophy

Department of Computer Science

Supervisor

Dr. Giuseppe DIFATTA

UNIVERSITY OF READING

January 25, 2019

Abstract

The exponential growth of the Internet over the last decades has led to a significant evolution of the network services and applications. One of the challenges is to provide better services scalability by placing service replica in appropriate network locations.

Finding the optimal solution to the facility location problem is particularly complex and is not feasible for large scale systems. Locating facilities in near-optimal locations have been extensively studied in many works and for different application domains. This work investigates one of the most notable problems in facility location, i.e. the p-median problem, which locates p facilities with a minimum overall communication cost. All previous studies on the p-median problem used a centralised approach to find the near-optimal solution. In this case the required information needs to be collected in order to apply a sequential algorithm to find a solution.

The centralised approach is infeasible in large-scale networks due to the time and space complexity of the sequential algorithms as well as the large communication cost and latency to aggregate the global information. Therefore, this work investigates the p-median problem in a distributed environment.

To the best of the author's knowledge, this is the first work to study the distributed p-median problem for large-scale computer networks. Solving the p-median problem in a fully distributed way is a challenging task due to the lack of global knowledge and of a centralised coordinator.

Two new approaches for solving the p-median problem in a distributed environment are proposed in this thesis. Both are designed to be executed without any centralised collection of the data in a single node. These methods apply an iterative heuristic approach to improve a random initial solution and to converge to a final solution with a local minimum of the cost.

The first approach builds a global view of the system and improves the current solution by replacing a single facility at each iteration.

The second approach, is designed according to the well-known k-medoids clustering algorithm. At each iteration a local view of each cluster is generated and all facilities can be updated to optimise the solution.

Both approaches were implemented within the Java-based PeerSim network simulator for investigating the performance in large-scale systems and tested against different parameters such as the size of networks, number of facilities to be placed and different initial solutions. The results have shown that the first protocol is better at addressing locations for facilities since it converges to a lower total cost of the solution than the second protocol. However, the second one is faster in optimising the solution.

Acknowledgements

Thank you my God for all your blessing, and giving me the strength and confidence to achieve this important stage in my life.

I would like to express my sincere gratitude and appreciation to my supervisor, Dr Giuseppe Di Fatta, for his continuous support, effective co-operation, encouragement, insightful comments and valuable discussion which had a huge impact on this work.

My profound gratitude is given to my beloved parents, my father, professor A. Aldabbagh, and my mother, Shaimaa, whom they have a special place in my heart. Your unconditional love and continuous prayers have made me be as I am now. My extended love goes to my lovely sibling Dr Aws and Dr Marwa who provided me a spiritual and emotional support along the way.

A very special gratitude goes out to my second half, Baraa, for her enduring love, support, patience and encouragement throughout this scientific journey. My thanks and love are also extended to my dolls who introduced happiness, laugh to my life and let me get up active and energetic every morning Zinah, Amenah and Dinah.

I also would like to thank my lab mates Alex, Chris, Salwa, Perusha, Mahmood, Manal, Mark and my night study comrades Sami and Hassan for their endless support and for all the fun we have had over the years of study. An extraordinary thanks are due to Mosab for his unlimited collaboration and support during my work.

My sincere thanks are due to the Ministry of Higher Education and Scientific Research, Republic of Iraq, the Iraqi Cultural Attach in London, and the University of Mosul for the scholarship and the financial support. Thanks are also due to the University of Reading for offering all the necessities to prepare this thesis.

Declaration

I confirm that this is my own work and the use of all materials from other sources has been properly and fully acknowledged.

Signed: Anas Abdulmajeed Al-dabbagh

January 25, 2019

Certification

This is to certify that the thesis entitled **The Distributed p-Median Problem in Computer Networks** has been prepared under my supervision by Anas A. ALDABBAGH for the award of the Degree of Philosophy in Computer Science in the School of Mathematical, Physical and Computational Science, the University of Reading.

Dr. Giuseppe DIFATTA
Associate Professor
Department of Computer Science
University of Reading
Polly Vacher Building
Whiteknights
Reading
RG6 6AY

Contents

List of Figures

List of Tables

List of Abbreviations

DPM	Distributed P-Median protocol
f_i	Facility to Insert
f_r	Facility to Remove
FID	Facility IDentification
KM	distributed K-Medoid protocol
$\phi 1$(u)	closest open facility to the user node
$\phi 2$(u)	second closest open facility to the user node

Chapter 1

Introduction

1.1 Motivation

Over the last decades, communication among people has been significantly changed. Internet has become the main communication tool, mainly because many daily used devices such as mobile phones, PCs, laptops, and even televisions have become increasingly depend on the Internet. This has led to a dramatic increase of Internet users. Internet World States indicates that in 1995 the users of the Internet were only 16 million people (0.4% of the world population). This number changed to 4156 million (54.4 % of the world population) in 2018 and will be continuously increasing in next generations [1]. With these rapid changes in the number of Internet clients, it is important to provide the best services to users. One of the ways to provide good services is to locate facilities in the right places.

A facility in the network is a node such as a hub or a computer that provides services to other nodes. Facilities are considered in the best locations when they present a cheapest and fastest services to their users. This can be achieved if the spatial locations of the facilities are closest to most of the users.

Locating facilities, in their best locations to the users, have received significant research over the last decades [2]. In particular, the p-median problem aims at finding a specific number of facilities with the purpose of making the total cost of solution to be minimum [3] [4]. The cost in this work is defined as the sum of the number of hops between facilities and their connected users.

Studying the p-median problem had received a large consideration due to the wide range of its applications [5] [6] [7]. However, all the previous works on the p -median problem are in central approaches. The central approach is not feasible for the large-scale networks due to message load on the network, information privacy, a lot of memory and time is needed for information gathering and computations [8].

However, implementing a fully decentralized protocol is quite complicated, not only because of the need for synchronizing computations, but also due to the lack of the global knowledge about the network which is necessary for finding the best locations for the facilities.

This thesis is investigating this problem, and two protocols are suggested to solve the p-median problem in a distributed environment, the proposed protocols are tested on different topologies with different sizes (up to 500 K) using a simulator. The simulation results show an apparent reduction in the cost of the solution which means better locations for the candidate facilities are found.

1.2 The p-Median Problem

This thesis explores new ways to solve the p-median problem in a computer network. The p-median problem is one of the most important facility location problems [9] [10] [11] which has been extensively studied in different disciplines.

The p-median problem addresses the optimal location for p of predetermined m facilities, p $<$ m, in a way that the sum of the shortest path between the users and their closest facilities are minimised [12] [11] [13].

The p-Median problem is NP-hard [14] [15], so that heuristic and meta-heuristic methods are usually used to solve it [13] [16].

The existing approaches use centralised methods for solving the p-median problem, i.e. knowledge about the input topology should be available in a certain node or server. This knowledge can be used for determining the best p locations for the facilities. Existing approaches, however, can not address facilities locations in large-scale networked systems, due to the lack of global knowledge of topology and large memory required to collect information from the node, in addition to the requirement of vast time in order to collect information about the network topology.

Therefore, this work intends to solve the p-median problem in a large-scale network system. Two different approaches are proposed to solve this problem. Both of them are implemented in a fully distributed environment without a pre-defined for the network topology. The first approach (as will be discussed in Chapter 4) addresses location for facilities based on a global view of the network while the second approach (as will explained in Chapter 5) clusters the network according to the shortest path, based on classical k-medoid algorithm, then addresses the locations for facilities in each cluster simultaneously.

The experimental work presented in Chapter 6 provides one of the first investigations in solving the p-median problem in a distributed environment.

1.3 Applications of the p-Median Problem in Computer Network

Finding the best locations for the network facilities is very essential since it is needed for many network applications. Particularly, the p-median problem is crucial to a wide variety of different problems associated with distance minimisation. For example, the understanding of a computer or a topology behaviour based on communication network [17] [18]. Another example of the benefit from a minimum distance network clustering is to reduce the traffic and response time in a network through monitoring and control scalability of a network [19][20]. Wireless sensor networks, in which the data are periodically collected from sensors [21], can also be utilised in the distributed p-median solution by locating the servers in a way that the total distances to the sensors are minimised. Another application of the p-median problem is cluster analysis [22] [12] [23].

1.4 Aim and Contributions

As stated earlier, the main aim of this work is to investigate the p-median problem to address the near-optimal locations for facilities in distributed network systems.

To achieve the aim of this project, the following contributions are proposed:

- A novel Distributed p-Median protocol (DPM) to identify the best locations for the facilities in a network is proposed in this work. Despite that network topology and data are not gathered in a server, but they are remained intrinsically distributed in the network; the protocol builds a

distributed view of the network to identify the best locations among candidate locations.

- A second disributed K-Medoid protocol (KM) is proposed to solve the p-median problem by extending the classic k-medoids approach to a distributed environment for clustering the network topology.

The two distributed protocols apply a similar heuristic iterative approach to improve an initial random solution and converge to a final solution (local minimum). However, each applies a procedure of different granularity within each iteration. While DPM identifies adopt a finer granularity by identifying a single change to improve the solution, KM can apply changes to every component of the solution within each iteration, as shown in Figure 1.1.

The former is expected to be more accurate and requires more iterations to reach convergence. The latter is a straightforward extension of the most popular clustering method (k-medoid) and can provide an interesting comparative analysis.

1.5 Objectives

The work in this thesis contributes to the area of the facility location-allocation problem in computer networks. Specifically, it introduces novel protocols to solve a distributed p-median problem. The main objectives of the work in this thesis include:

1. Implementation of one of the sequential solutions of the p-median problem, the ”fast swap-based local search algorithm”. The obtained results from this implementation are used to validate the proposed distributed p-median approach, since it depends on the same logic of this sequential algorithm (Chapter 3).

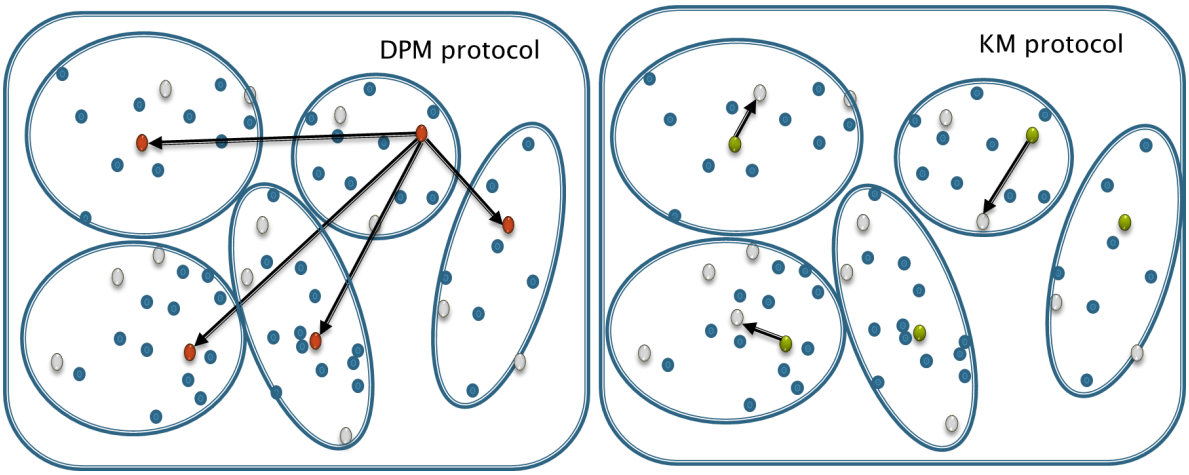


Figure 1.1: The strategies of the DPM and the KM protocols, while DPM identifies a single facility location based on the global information of the network to improve the solution, KM identifies facility locations in each component of the solution within each iteration until convergence

2. Distributed design and implementation of novel protocols to solve the distributed problem directly without the need to central information.

3. Investigation of the logic of the k-medoid algorithm, the classical model of network clustering, so as, in parallel, to find the best location for a facility in each cluster.

4. Design and implementation of a new distributed protocol to find the median locations for facilities in each cluster of the network.

5. An extensive experimental analysis that test the efficiency of the implementation of the proposed approaches. Besides to a comparative study for the implementation of both of them. The comparison analysis was carried out on various topologies with different configurations.

It is hoped that these contributions would lead to better improvement in computer networks services.

1.6 Summary of the Thesis

This thesis is composed of seven themed chapters which have been organised to incrementally describe the contribution. The remainder of the chapters of this thesis are structured as follows:

- **Chapter 2** Begins with laying out the p-median problem definition, a used terminology to identify this problem, as well as, a mathematical formulation of the p-median problem.

  This chapter also discusses the related work to the p-median problem and some previous techniques used to attempt to solve this problem.

- **Chapter 3** Is concerned with one of the common solutions of the centralised p-median problem, which is outperformed to other suggested solutions in this area [13].

  This solution is the centralised approach. Section 3.3 explains the implementation of the centralised algorithm while Section (3.4) shows the results of evaluating the algorithm.

- **Chapter 4** Explains the first proposed approach for solving the p-median problem in a distributed environment (DPM). It explains the design of the DPM approach in Sections 4.2 and 4.3. Moreover, the details of the DPM implementation is described in this chapter.

- **Chapter 5** Demonstrates the second proposed approach for solving the p-median problem in a distributed environment (KM). KM design is based on the classical k-medoid protocol which is defined in Section 5.2. An overview of the proposed approach and its design are illustrated in Sections 5.3 and 5.4 respectively, while Section5.6 presents a summary and conclusions about the chapter.

- **Chapter 6** An overview of the used simulator and the used dataset for simulation are described in Sections 6.1 and 6.2.

  Section 6.3.1 highlights the validation of the DPM approach against the results obtained from the centralised approach. Section 6.3.2 presents the evaluation of the metrics that are used to evaluate the performance of the DPM and KM approaches, while Section 6.4.1 shows a comparison analysis between the two approaches. Finally, Section 6.6 presents a summary of the chapter.

- **Chapter 7** Concludes the present work and discusses the improvements which could benefit from it in the future

Chapter 2

Background

This chapter reviews the general concepts related to the p-median problem and provides an overview of the p-median problem formulation. It also describes the fundamental techniques that are used for solving the p-median problem. Moreover, this chapter presents some algorithms that have been proposed in the literature.

2.1 Introduction

The operation of locating a set of facilities is a critical part of strategic planning for many applications because it affects their cost. Therefore, the location problem has been an issue of great interest in a wide range of fields.

Early work on the location problem was started in 1909 when Weber tried to locate a single facility in order to minimise the total distance between the facility and several users [24] [25] [26]. The problem was then extended to deal with a set of facilities in different applications domains. For example, Hakimi(1964) located a switching centre in a communications network [27]. Teitz and Bart(1968) also suggested an algorithm for locating multi facilities [28]. Thereafter, a growing trend towards studying the facility location problem has been noticed [29]. It is termed as 'p-median problem' [30] [31].

Section (2.2) defines the p-median problem with a brief list of the previous studies that have been carried out in this area.

2.2 Definition and History of the p-Median Problem

The p-median problem is intended to find the median locations for facilities among a given number of candidate locations in a way that the total cost from the user nodes to facility nodes is minimised [29][32] [33] [34] [35]. The best solution among the candidate solutions is found through an objective function. The p-median problem is considered as one of the most studied problems among location-allocation models [36] [37] [38].

The p-median problem has been used for many different applications to find the best locations for facilities and to provide satisfactory services to their clients for a wide range of domains [39] [40] [41] [42].

In the p-median problem, *p* represents the number of facilities that the solution is requested to have median places for them. A median place is the place of a facility which is closest to the most user nodes around it. For example, if p = 10, then ten median locations for facilities are addressed among the candidate locations. The facility capacity and the number of services provided by a facility are not considered in this problem [43].

Because of its importance, the p-median problem has been studied for more than a century [44]. Several strategies, which occupied a prominent position, for solving the p-median problem have been reported, the related literature among them are briefly reviewed below.

- The establishment of modern location theories started by Weber in 1909 [24], which can be considered as an extension to the rectangle distance minimisation of the mathematician Fermat in the 17th century [45].

- In 1964 the concept of a 1-median problem was generalised to the p-median problem by Hakimi [27] when switching centres were distributed in Telecommunication network [36].

- A neighbourhood search improvement algorithm was proposed by Maranzana [46], which divided the solution to p partitions for the sake of exploiting the ease 1-median problem by finding the optimal location in each partition, if the solution is updated, then the users are repartitioned and the process repeats itself until facility nodes remain without change [47].

- In 1968 an important heuristic algorithm for solving the p-median problem was proposed by Teitz and Bart [28], Section 2.6.1 illustrates the algorithm.

- A *first profit strategy* algorithm was proposed by Whitaker [48] which was found faster than Teitz and Bart algorithm [9]. Section 2.6.2 shows more details about Whitaker solution.

- Building up on the Whitaker study, Hansen proposed a *'variable neighbourhood search'* algorithm, which suggested the best improvement strategy to solve the p-median problem, as described in Section 2.6.3.

- Liotta *et al.* [19] [49] suggest to cluster the network into several partitions through exploiting the properties of mobile agents, then localise service facilities in each cluster according to p-median and p-centre problems.

- The *fast swap-based local search* algorithm was proposed by Resende and Werneck, is considered as one of the important algorithms in the p-median problem, in which Resende and Werneck proposed a tech-

nique to accelerate the algorithm. This algorithm is adopted in this work. Chapter 3 explains the details of this algorithm's implementation.

However, before explaining the details of the p-median problem and the suggested solutions, the terminology used in the p-median problem is explained in the following section.

2.3 Terminology Used in the p-Median Problem

The p-median problem has four main components [31] [36] [50]:

1. Facilities.
2. Users.
3. Cost of the solution.
4. Location or space.

The subsections below explain each of these components.

2.3.1 Facility

The facility is a node that provides services to user nodes. In a computer network, the facility could be servers, hubs, memory cache, sensors, etc. Some types of facilities are used as an information centre for monitoring the behaviour of the network, the monitored information could be analysed for the purpose of reducing the traffic of the network. The facilities are characterised by their status, number and costs [36].

In this work, a facility status is either '*open*' or '*closed*'; an open facility can serve the users, while a closed facility is a candidate location for an open facility. All the facilities are assumed to be identical, and the type of services that are presented by the facility is not taken in consideration of this work.

The number of facilities is an important parameter; it indicates the number of facilities needed to be opened in the network. Most of the location-allocation problems intend to open multi facilities in the solution together, taking into account the existing candidate locations [2].

Cost of a facility, is a numerical number refers to the distances of all users served by this facility. The cost of a facility is a critical attribute by which the location-allocation models are differentiated through it [43]. Facility cost can be classified into two types, fixed and variable costs. A variable cost has a relationship with the service delivery, while a fixed cost is associated with the facility setup. In this work, the cost of the facility is the sum of hops between the users and the facility. Therefore, it is variable, because it depends on the connected users to the facility.

2.3.2 User or Customer

The user node (also called demand or customer) is one of the main components of the p-median problem. The user is the node that needs to access the facilities services [43].

To provide satisfactory services to the user nodes, it is essential to know the distribution of the user nodes. Several scenarios for assigning the users to facilities are suggested, such as assigning each user to its closest open facility [51], assigning each user to the centroid of the area [52] and in some problems user nodes are randomly assigned to facilities for simulation purposes.

2.3.3 Cost of the Solution

The cost is the important objective in the solution of the location-allocation problem. The cost of the solution is the summation of all open facilities' cost whereas the cost of the open facility is the summation of all users' cost served

by this facility. A user cost is the shortest path in a number of hops to reach its closest open facility. The placement of facilities is differentiated according to its cost [43].

The open facility cost is not a fixed value; it depends on the number of the users served by the facility in addition to the cost of each served user.

The cost of the facility is also changed according to the placement of the other facilities in the system, because each facility serves the users around itself (a user is served by its closest open facility). Therefore, when a facility is open or closed, the cost of other open facilities in the system are changed, simply because the user nodes change their service source to their closest open facility.

2.3.4 Location

Location or space is the last main component of the location-allocation problem.

Discrete, *continuous* and *network-based* spaces are three types to represent the space in the location-allocation problem.

In discrete location models the best selected locations are chosen from preselected potential locations; therefore a preceding knowledge of the candidate sites has to be available. The decision is taken by choosing candidate sites depending on geographical or economic factors [43].

In continuous space [52][53][54] the possible site locations can be determined by one or more continuous coordinates. In continuous model there is no presumption of candidate sites; instead, they are generated as an output by this model. Continuous location coordinates are normally considered in Euclidean space [55].

Network-based is another type of location representation. In this representation model, the users and facilities are located in nodes (as adopted in this work). However, in some applications, the facilities can be located on the link between the nodes [56]. Graph network is adopted in many studies of the location-allocation problem [27][53] [57] [58][59] [60][61] [62] [63] [64] [45] [65].

2.4 Techniques for Solving the p-Median Problem:

Three techniques of solutions were used for solving the p-median problem. These methods are exact, heuristic and metaheuristic. The subsections below explain each method briefly focusing on the reason of the use of metaheuristic technique in this work.

2.4.1 Exact Technique

In order to find the optimal solution of the p-median problem using the exact technique; it needs to complete finding all the available solutions before choosing the optimal result. The exact technique may need enormous amounts of mathematical computation and a long time to be completed, in addition to a large memory for saving the temporary results [45].

Using the exact method, the total number of the solutions to find the best p facilities out of N facilities is computed according to the combinational formula 2.1.

$$\binom{n}{p} = \frac{n!}{p!(n-p)!} \tag{2.1}$$

Correa *et al.* [66] tried to allocate 26 university admission examinations out of 43 candidate facilities for 19710 students. For the exact solution, the number of candidate solutions are more than 421 billion, as shown by formula 2.2.

$$\binom{43}{26} = \frac{43!}{26!(43-26)!} = 421,171,648,758 \tag{2.2}$$

As can be noticed from this example, this simple problem takes a very large number of computations and a long time for finding the best solution. Therefore, heuristic and metaheuristic solutions are suggested to solve the p-median problem.

2.4.2 Heuristic Techniques

A heuristic is problem-solving technique that can be used to speed up the process of finding a satisfactory solution. Heuristic methods are used in approximation algorithms to find a solution. It is true that there is no guarantee to find the optimal solution. However, it is often a sufficient method that used for finding a solution close to an optimal solution for a non optimal ones [67] [45]. Arifin [45] mentioned that Cooper‘s algorithm [68] was considered as the first algorithm in a location-allocation problem which used a heuristic technique to solve the p-median problem. Other examples for solving p-median problems using heuristic technique methods are the greedy adding algorithm, the alternating algorithm and the vertex substitution algorithm [38].

2.4.3 Metaheuristic Techniques

Another formal technique for solving complex optimization problems is the metaheuristic. Blum and Roli [69] reported that metaheuristic is a high-level concept for exploring search spaces by using different strategies [70].

It is true that metaheuristics also produce approximate solutions. However, similarly to heuristics they can escape local optima, which is useful for solving the location-allocation problems.

Genetic algorithms, tabu search and ant colony are examples of metaheuristic algorithms, as well as the variable neighborhood search which is adopted in this work [45] [71].

2.5 Mathematical Formulation of the p-Median Problem

The classical p-median problem can be defined as follows [32] [38] [39] [72] [73] [74] [75]: for a graph or network G= (V, E), given a set F of m potential locations for facilities and an integer number p of required facilities to be opened. The aim of the p-median problem is to identify the near-optimal locations for p facilities from F, so that the summation of distances from all clients to all p facilities is minimised.

Before going to the details of the algorithm the following notation is defined:

- G= (V, E) is a graph or a network, where V is the set of nodes and E represents the links between nodes.
- F is the set of potential facilities, where F $\subset$ V.
- U is the set of users, where U $\subset$ V and U= V $\setminus$F.

The basic parameters of the problem are:

- $u = |U|$.
- $m = |F|$.
- p is the number of facilities to be opened. Where $1 < p \leq m$, also no relationship between u and m is assumed.
- Distance function $d{:}U\ X\ F \rightarrow$ ℝ, where d(u) is the cost in shortest path in number of hops between a user node and a facility. d_{ij} is the distance from user i to facility j.
- $$x_{ij} = \begin{cases} 1, & \text{if node } i \text{ is assigned to facility } j \\ 0, & \text{otherwise} \end{cases}$$
- $$y_j = \begin{cases} 1, & \text{if facility } j \text{ is opened} \\ 0, & \text{otherwise} \end{cases}$$

As shown in equation 2.3 the objective of the p-median is to minimise the summation of the distances between the users and the facilities.

$$Min \sum_{i}^{u} \sum_{j}^{m} u d_{ij} x_{ij} \tag{2.3}$$

The aim of this problem is implemented under the following constraints:

1. Each user node assigns to one open facility as in 2.4.

$$\sum_{j=1}^{m} x_{ij} = 1 \tag{2.4}$$

2. The total number of open facilities is p, as in 2.5 .

$$\sum_{j=1}^{m} y_j = p \tag{2.5}$$

3. No upper bound of the number of user nodes is assigned to each open facility.

2.6 Algorithms for the p-Median Problem

This section gives a brief overview of some proposed approaches for solving the p-median problem.

2.6.1 Vertex Substitution Heuristic Algorithm

A vertex substitution heuristic is one of the well-known algorithms, which was proposed by Teitz and Bart in 1968 [28]. It is considered as one of the standards and successful algorithms for solving the p-median problem [9] [39]. In this algorithm, an arbitrary set of facilities (opened and closed) are chosen to start the algorithm as shown in Figure 2.1. The algorithm is substituted by one of the candidate facilities to open, whenever the substitution can reduce the cost. If more than one candidate achieves the cost reduction, then the one that leads to the minimum cost is selected.

When no more swaps to reduce the cost are available, as explained in Algorithm 1, the best solution is reached and the algorithm is terminated [76]. The termination means that all the final open facilities are local medians allocated for the user nodes.

Figure 2.2 shows that the best locations of p open facilities are determined and each user node is assigned to its closest open facility.

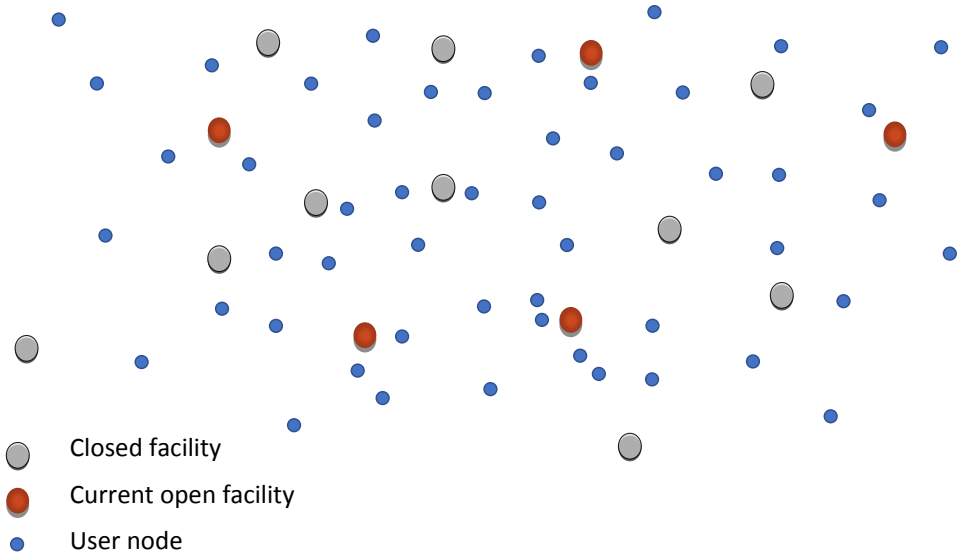


Figure 2.1: A graph of N user nodes with a set F of m potential facilities (open and closed) are arbitrary chosen (N=50, m =10, p=5)

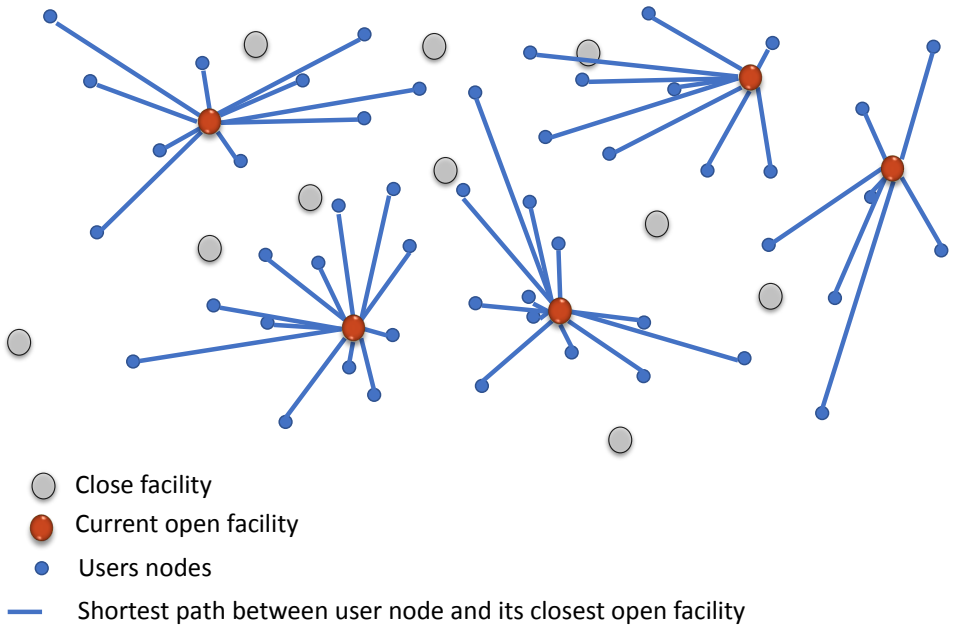


Figure 2.2: The near optimal locations for the facilities are found. Each user nodes assigned to its closest open facility via the shortest path.

Algorithm 1: Vertex Substitution Heuristic

```
Input: p value, set M of candidate facilities to open
Randomly select p facilities to open from the set of m
Assign each user node to its closest open facility
while There is a cost == TRUE do
    Compute the cost of the solution according to the equation (2.3)
    forall p facilities do
        forall m facilities do
            find a best facilitis pair to swap
        end
    end
    Re-assign each user node to its closest open facility
    Compute the new-cost of the solution
    if new-cost < cost then
        Set cost = new-cost
        else
            There is a cost = FALSE, a local optimum has been identified, and no
             improvement can be found
        end
    end
end
```

2.6.2 Fast Interchange Heuristics Method

Because of the velocity challenge of finding the best possible locations for candidate facilities, a *fast interchange heuristics* has been described by Whitaker based on the concept of Teitz and Bart algorithm [72]. Fast interchange heuristics is considered to be more efficient than vertex substitution by local search heuristic which was proposed by Teitz and Bart [77].

In vertex substitution algorithm, a profit from candidate facility to insert is computed for all possible facilities to remove, the facility that returns the best profit will be swapped out. To speed up the profit computation, a fast interchange method makes the profit composed of two components, gain and netloss, as stated in equation 2.6.

$$profit(f_i, f_r) = gain(f_i) - netloss(f_i, f_r) \tag{2.6}$$

When a new facility is candidate to be inserted, a certain number of nodes will be utilized from the added new facility, because it will be closer to them than their current assigned open facility; this is called gain. It can be calculated for all utilized nodes from the facility insertion. As shown in equation 2.7; for each node, if the inserted facility is closer than its current assigned facility, the difference between distances is considered as the gain from that insertion.

$$gain(f_i) = \sum_{u \in U} \max(0, d1(u) - d(u, f_i)) \tag{2.7}$$

The netloss, which is the second component of the profit equation, is computed for all other nodes that would not be utilized from the insertion of the facility (as in equation 2.8).

In case of the currently assigned facility to the node is removed; the node is reassigned to either the second closest facility $\phi 2$(u) or the newly inserted facility (if it is closest than $\phi 2$ to the node). In both cases, the cost to serve this node is increased.

$$netloss(f_i, f_r) = \sum_{u:[\phi_1(u)=f_r] \Lambda [d(u,f_i) > d_1(u)]} \min(d(u, f_i), d_2(u)) - d_1(u) \tag{2.8}$$

Whitaker algorithms were implemented by the findOut function, which takes a candidate facility to insert as an input and returns the best facility to remove, as explained in Algorithm 2 [72][78]. Whitaker used a first improvement strategy, in which swap facilities are made whenever a profit is available[72]. This can be done by invoking the findOut function for a particular f_i and check if there is a profit from the insert of this facility; the following steps are implemented:

1. Swap the facilities.
2. Update the first and the second closest facility to the user node.
3. Calculate the new gain and netloss.
4. Mark the inserted facility as add, since each facility is considered to be added only once.
5. Call findOut again, and if there is a profit goto step 1.

Algorithm 2: Function findOut to determine the best candidate for removal (f_r) by given the best candidate for insert (f_i)

```
gain = 0; // gain resulting from the addition of f_i
forall f ∈ S do
    netloss(f)= 0;
end
forall u ∈ U do
    if d(u,f_i) < d1(u) then
        // gain if f_i is close enough to u
        gain += [d1(u)−d(u,f_i)]
        else
            netloss(Φ1(u))+= min [d(u,f_i),(d2(u)-d1(u))];
        end
    end
end
f_r = argminf ∈ S (netloss(f));
profit = gain - netloss(f_r);
return (f_r, profit)
```

2.6.3 Variable Neighborhood Search (VNS)

Despite the importance of Whitaker study [28], it was not widely used (might be because that paper was not precise) [77] until it was applied as a subroutine of a *variable neighborhood search* heuristic by Hansen and Mladenovic in 1997 [78].

Best improvement strategy was adopted. In this strategy, evaluations are made to all possible swaps, then the most profitable ones are executed. Moreover, once the facility adding restriction is removed, it gives the flexibility to

find the best choices of open facility positions. Besides, the way of finding the best facility to open is evaluated as to be less complicated [77][78].

2.6.4 Fast Swap-based Local Search

Depending on the work that has been carried out by Whitaker 1983 and Hansen 1997, another algorithm to solve the p-median problem was suggested by Resend and Werneck in 2003 [72]. It was based on swapping facilities heuristics. However, before a candidate facility inserted, it is assessed with each candidate facility to remove to find the exchange that makes the best profit. After assessing of all candidate facilities to remove, a pair of facilities that achieves the best profit is selected to swap on the solution.

It is true that this implementation is considered as the worst complicated one; however, it is still the faster in practice, particularly for large applications [72].

To accelerate the algorithm, in each iteration, Resend and Werneck used the information gathered from the previous iterations instead of computing everything from scratch (other researchers calculate their entities from scratch). However, an additional memory space is used to achieve this acceleration [79]. The details of this model are discussed in Chapter 3.

2.7 Limitations of the Current p-Median Solutions

Despite that the p-median problem has been an object of research since 1960's, no studies have been published which tried to solve this problem in the distributed environment. However, the current centralised solution approaches have also some notable drawbacks which can be summarised as follows [80]:

- The number of facilities to open p must be given as an input. It is true that in some applications p can be easily determined, based on the fixed factors. For example, installing a known number of sensors in a network. However, it is a hard task to determine the optimal number of facilities in a given dataset.

- The candidate locations for facilities to be opened is also one of the input parameters, not only the size of F is a hard task to determine but also the elements of the set F.

- The cost of the solution is based on the distribution of the facilities which are arbitrarily selected.

- The algorithm converges to a local minimum.

- It is difficult to implement with an large-scale dataset because a lot of memory is required for the implementation. Moreover, it is infeasible to implement it on a network due to its decentralisation property and the difficulty of collecting the required information of the network in a central node.

This project aims to improve the last drawback by proposing solutions able to implement the p-median problem in a distributed environment, as assumed in Chapters 4 and 5.

2.8 Summary

In this chapter, the p-median problem has been defined and the terminology and formulation of this problem are presented. Besides, the explanation of the methods used for solving the p-median problem.

Some of the proposed algorithms for solving the p-median problem have also presented. Finally, the limitations of the current solutions were stated.

The various solutions techniques, described in this chapter, help the applications to address facilities locations with minimising their overall cost. However, all of these proposed strategies are sequential solutions, which presumed that all the required input data are available in a central place.

However, this work endeavors to specify such an interface in computer networks, which is discussed in Chapters 4 and 5.

The next chapter explains, in detail, the centralised approach of the p-median problem which is considered as a logic-based approach of the proposed distributed approaches.

Chapter 3

Centralised Approach for the p-Median Problem

3.1 Introduction

Over the last decades, many algorithms have been developed to solve the p-median problem [28] [54] [72]. Commonly, the p-median problem addresses the near-optimal location for p number of facilities to open from a pre-defined number (m) of candidate locations such that the total distances between users and the open facilities are minimised.

The key aspect is to find the solution within as less time as possible. After the successful development of the vertex substitution approach; several studies have been carried out to enhance it [28] [81] [82]. One of the important proposed approaches is called *fast swap-based local search* [10] [13] [48]. This chapter explains, in details, this approach implementation since its logic is used in the proposed distributed approaches (as illustrated in Chapters 4 and 5). Moreover, this chapter shows the results obtained from this approach.

3.2 Fast Swap-based Local Search

Building up on the Whitaker 1983 [81] and Hansen 1997 [82] procedures, Resende and Werneck [77] suggested a *Fast Swap-based Local Search* algorithm to solve the p-median problem. The primary goal is to find the near-optimal locations among the candidate facilities locations as fast as possible.

As shown in flowchart 3.1; the algorithm starts with N nodes topology and a set F of m candidate locations for facilities. p facilities from F are randomly picked to open as an initial solution. The algorithm iterates to find the best p locations among the m candidate locations until convergence. At each iteration, the algorithm finds a pair of facilities, (one to be closed and one to be opened). Each swap provides the best local improvement of the cost solution.

Remove a facility from the solution will cause loss to the user nodes join it, and insert a facility to the solution will gain the user nodes close to it . Finding the best pair of facilities to swap is done by subtracting the loss of a candidate facility to be removed from the gain of a candidate facility to insert. Among all possible pairs, the one that provides the highest improvement (called profit) is chosen for the next swap. Therefore, equation 3.1 is applied to all candidate pairs.

The best f_i is the closed facility that achieves maximum gain among all candidates if it is opened. On the other hand, the candidate open facility f_r to be removed is the one that leads to the least possible loss when it is removed from the solution.

$$profit(f_i, f_r) = gain(f_i) - loss(f_r) \tag{3.1}$$

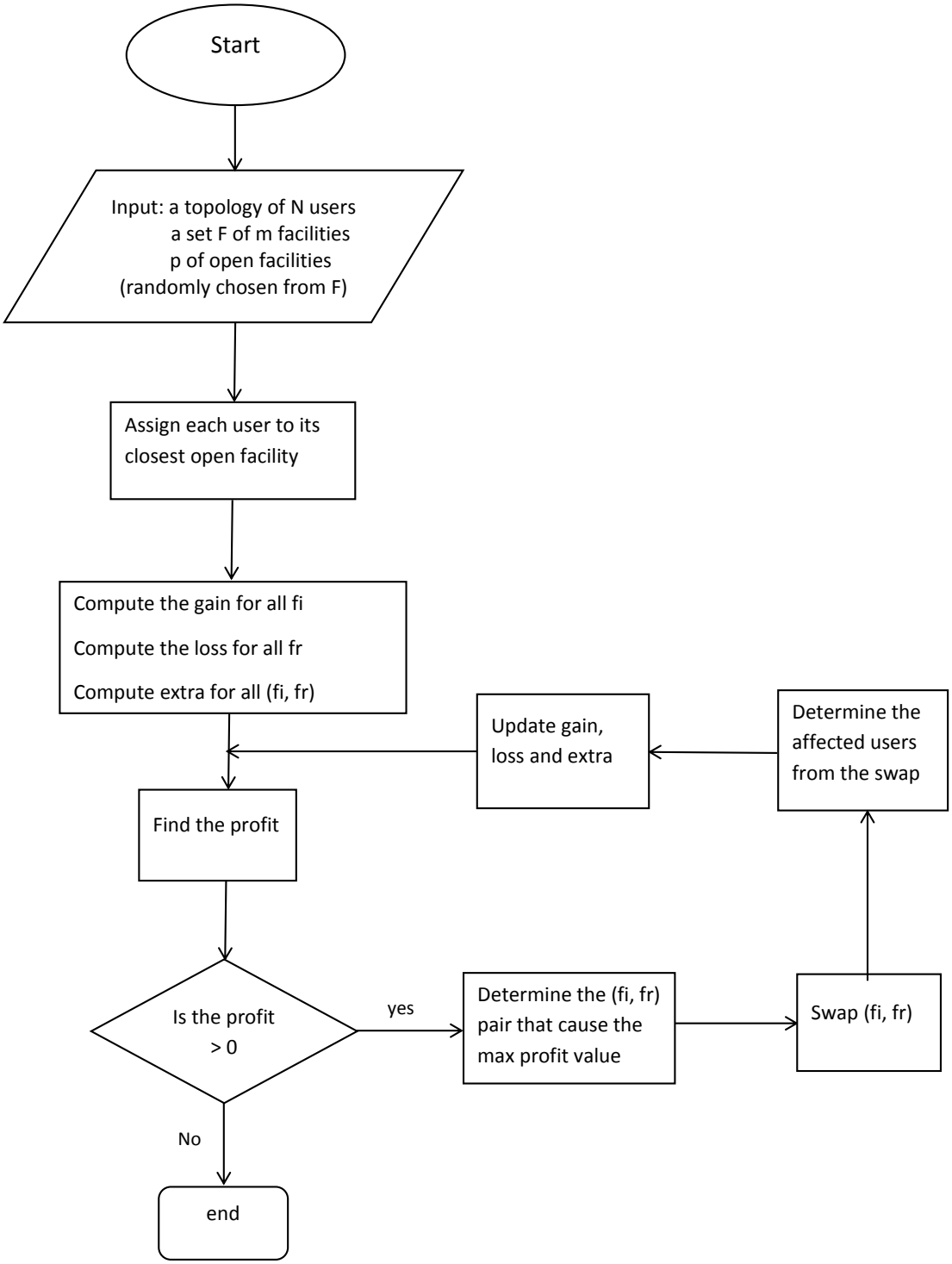


Figure 3.1: Fast swap-based local search flow chart

Equation 3.1 is composed of two components:

- The **gain** which is the total distance saved by transferring nodes to the facility f_i, if it is closer to them than their currently assigned facility.
- The **loss** which is the additional cost (extra distance) of transferring each node currently assigned to f_r to its second closest facility.

Gain is calculated for all candidates f_i as in the equation 2.7. It is represented by a vector of size (number of f_i). Each value of the vector represents the potiential gain if the candidate f_i is inserted.

On the other hand, the loss component in equation 3.1 is computed for each candidate f_r according to the equation 3.2. All users nodes assigned to f_r need to be reassigned to their second closest facility.

$$loss(f_r) = \sum_{u:\phi_1(u)=f_r} [d_2(u) - d_1(u)] \tag{3.2}$$

Actually, not each user node currently assigned to the facility f_r is being assigned to its second closest facility, it may be assigned to f_i if it is closer to it than the second closest facility. In this case, equation 3.1 of profit has to be corrected to deal with these cases. Two possible scenarios may occur if the current facility assigned to the node is removed:

The first one is that the inserted facility is closer to the node than its current $\phi_1(u)$ as shown in figure 3.2 , the user node in this case will be reassigned to f_i instead of its second closest facility. So that the prediction of loss to assign the node to its second closest facility is not valid, and is overestimated by $[d(u,\phi_2(u)) - d(u,f_r)]$.

The second scenario is when the inserted facility is closer than $\phi_2(u)$ but faraway from the user node than $\phi_1(u)$ as shown in figure 3.3. In this case,

Figure 3.2: Case1: f_i is inserted closer to the user node than its $\phi_1(u)$. Therefore, the user node will connect to f_i instead of its $\phi_1(u)$.

the user node will remain assigned to its current $\phi_1(u)$. However, the loss value is overestimated by $[d(u,\phi_2(u))\text{-}d(u,f_i)]$.

This overestimation must be treated by adding a correction factor *'extra'* to the profit equation 3.1. The extra value can be determined as follows:

$$extra(f_i,f_r) = \sum_{u:[\phi_1(u)=f_r]\Lambda[d(u,\phi_1(u))<=d(u,f_i)<d(u,\phi_2(u))]} [d(u,\phi_2(u)) - d(u,f_i)$$

$$+ \sum_{u:[\phi_1(u)=f_r]\Lambda[d(u,f_i)<d(u,\phi_1(u))<=d(u,\phi_2(u))]} [d(u,\phi_2(u)) - d(u,f_r)]$$

which can be simplified to:

$$extra(f_i,f_r) = \sum_{u:[\phi_1(u)=f_r]\Lambda[d(u,f_i)<d_2(u)]} [d_2(u) - \max(d(u,f_i),d_1(u))] \quad (3.3)$$

The profit equation for each predicted swap will become as in the equation 3.4 :

$$profit(f_i,f_r) = gain(f_i) - loss(f_r) + extra(f_i,f_r) \quad (3.4)$$

Figure 3.3: Case2: f_i is inserted and $\phi_1(u)$ is removed, however, the f_i closer to the user than its $\phi_2(u)$. In this case, the user node will connect to f_i instead of its $\phi_2(u)$.

3.3 The Centralized p-Median Problem Implementation

The centralized approach has been implemented using Java programming language and it has been thoroughly tested on different small topologies (up to 5000 nodes) with different initial sets of candidate facilities (as explained in section 3.4).

The algorithm is implemented as follows:

- **The inputs:**
  - A topology G= (V, E), where V is the set of nodes and E represents the links between nodes.
  - A set U of user nodes.
  - A set F of candidate facilities (size m), where V = F ∪ U.
  - p Number of open facilities randomly picked from F, where: p < m.
- **The output:**
  - The best locations for the candidate p facilities.

The program is started by computing the shortest path matrix between the m facilities and the user nodes according to the Dijkstra algorithm [83] [84],

$d_1(u)$ and $d_2(u)$ are determined from the sort path matrix. Accordingly, gain, loss and extra values are computed as in section 3.3.1

3.3.1 Computation of Gain, Loss and Extra Parameters

As shown in the Algorithm 3, at each iteration the values of gain, loss and extra are computed to find the best pair of facilities to swap and improve the current solution. *Gain(fi)* is the amount of distances obtained by transferring nodes from their current assigned facilities to the closer inserted facility (f_i). Gain is calculated for all possible candidates f_i; therefore, it is represented by a vector of size equals to number of candidate facilities to insert (m-p).

Loss is the amount of lost distances caused by transferring the user node from its current assigned facility to the second closest open facility. Loss is computed for all current open facilities (f_r) and saved in a vector of size equals to the number of the open facilities (p).

The correction factor *extra* is the overestimation in the gain or loss values in the case that the user node does not connect to its second closest facility as explained in section 3.2. *Extra* is computed for all f_i and f_r so that it will be saved in two dimensioned array of size (number of f_i, number of f_r). After computing these three objects, the best pair to swap is determined through the computation of the maximum profit as seen in section 3.3.2.

3.3.2 Find the Profit and Implement the Swap

Finding the profit from a swap between all candidates f_i and f_r facilities is essential to make the highest improvement in the solution. As shown in algorithm 4, the profit equation 3.4 is applied to all candidate facilities to find the best pair to swap. The pair of facilities (f_i,f_r) that makes the maximum positive profit is selected.

Algorithm 3: The gain, loss and extra parameters computation

Input: Short distance matrix between users and facilities
Gain vector values:
forall f_i **do**
  forall *user nodes(u)* **do**
    gain(f_i) += max(0, $d_1(u)$-d(f_i,u)
  end
end

Loss vector values:
forall *user nodes u* **do**
  forall f_r **do**
    if $\phi 1(u) == f_r$ **then**
      loss(f_r) += $d_2(u)$-$d_1(u)$
    end
  end
end

Extra matrix values
forall f_i **do**
  forall f_r **do**
    if *(*$\phi 1(u) == f_r$ *) AND (* $d(u,f_i) < d_2(u)$ *)* **then**
      Extra(f_i, f_r) += $d_2(u)$- max[(f_i,u), $d_1(u)$]
    end
  end
end

Algorithm 4: Compute the profit and find the best pair of facilities to swap

Input: gain(f_i) vector, loss(f_r) vector and extra(f_i,f_r) matrix
max_profit = 0
forall f_i **do**
  forall *fr* **do**
    profit(f_i,f_r)= gain(f_i)-loss(f_r) + extra(f_i,f_r)
    if *profit*(f_i,f_r) > *max-profit* **then**
      Set max_profit = profit
      Set facility to insert = f_i
      Set facility to remove = f_r
    end
  end
end

To implement the swap; the candidate f_r is removed from the open facilities group and replaced by the candidate f_i. At the same time, the candidate f_i is removed from the closed facilities group and replaced by the candidate f_r.

At this point, a new solution is configured, The values of the $d_1(u)$ and the $d_2(u)$ are updated for some user nodes. This led to changes in the gain, loss and extra values. This means that another pair of facilities might improve the solution if they are swapped. Therefore, gain, loss and extra values have to be recomputed to find the new pair of facilities to swap.

Practically, swapping between f_i and f_r will affect only on the users surrounding this pair of facilities. As shown in figure 3.4, a small number of users are affected by the swap. Therefore, instead of recalculating the gain, the loss and the extra values from scratch, they can be updated as seen in Section 3.3.3. Updating gain, loss and extra values instead of recomputing them saves computation time which is essential aspect.

3.3.3 Determine the Affected Users

As explained in section 3.3.2, finding the affected users from the swap of f_i and f_r facilities is useful to reduce the number of computational processes for updating gain, loss and extra values.

Because gain, loss and extra values are the summations of each user node contribution (as explained in section 3.3.1), their values are changed according to the affected users contribution. To update gain, loss and extra values the affected users must be determined first. It can be said that a user node is affected by a swap of f_i and f_r facilities if and only if (see Algorithm 5):

- The inserted facility f_i is closer to the user node than its current $d_1(u)$.
- The inserted facility f_i is closer to the user node than its current $d_2(u)$.

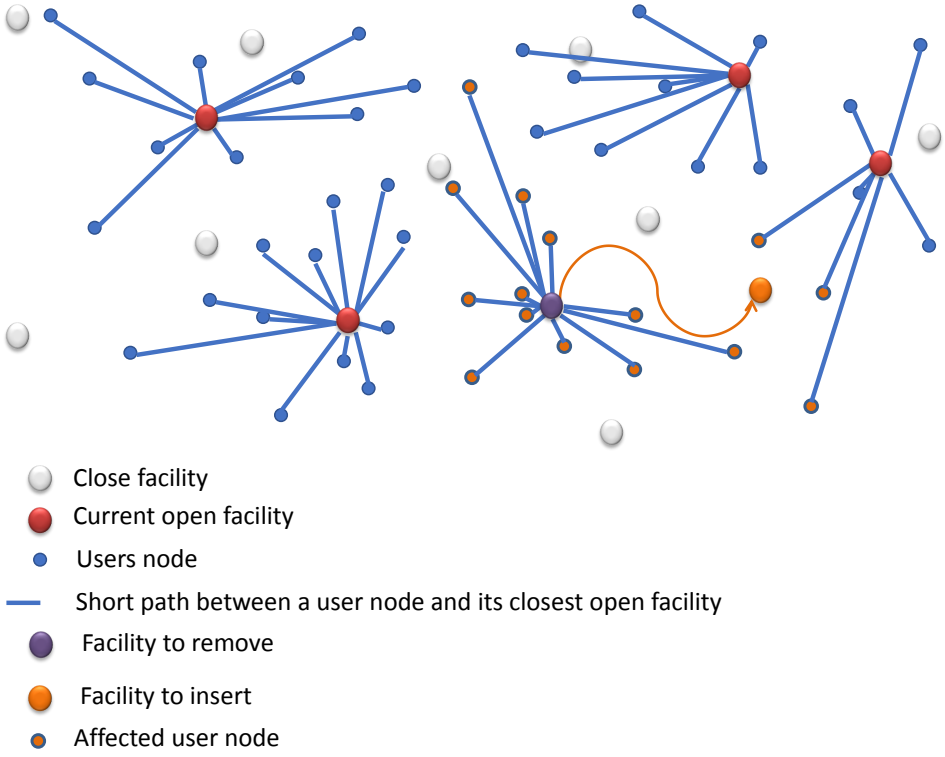


Figure 3.4: The affected users from the swap of f_i and f_r facilities. Little number from the total user nodes are affected.

- The removed facility f_r is the current $d_1(u)$ of the user node.

- The removed facility f_r is the current $d_2(u)$ of the user node.

3.3.4 Updating the Gain, the Loss and the Extra Values

From Algorithm 3 it can be noticed that the computation of gain, loss and extra are dependent on $d_1(u)$ and $d_2(u)$. Since the $d_1(u)$ and $d_2(u)$ are changed for the affected users only; the recalculation of gain, loss and extra for the not affected users will lead to the same results and it takes extra time to be completed.

Algorithm 5: The determination of affected users

Input: Set of f_i , Set of f_r.
output: Determine the affected users.
forall *user nodes(u)* **do**
  if *($\phi 1(u)$ == f_r) OR ($\phi 2(u)$ == f_r)* **then**
    mark the user node as affected
    continue
    else if *($d(u,f_i) < d_1(u)$) OR ($d(u,f_i) < d_2(u)$)* **then**
      mark the user node as affected
      continue
    end
  end
end

Updating the values of gain, loss and extra for the affected users is carried out through the following steps:

1. Subtract the contribution of each affected node from gain, loss and extra values.

2. Update the first and second closest p-median facility to the node (Section 3.3.5).

3. Add the new contribution of affected nodes to gain, loss and extra values.

The above steps have to be implemented in the same consequent order; because it is impossible to find the old $\phi 1(u)$ and $\phi 2(u)$ that contribute with the value of the gain, loss and extra after the update of the first and second closest facilities for the affected user node. As explained in Algorithm 6; the contribution of the affected user nodes is deleted as in the follwoing steps.

1. Update the loss value by subtracting the contribution of the user nodes which there ϕ_1 is removed.

2. If the inserted facility is closer to a user node than its current ϕ_1 then update the gain value by subtracting the diffrence between the distances of its current ϕ_1 and f_i from the current gain value.

3. If the inserted facility is closer to a user node than its current ϕ_2, then the extra value should be updated by subtracting the diffrence between the distances of f_i and ϕ_2 from the current extra value.

Algorithm 6: Subtract the old contribution of affected users from gain, loss and extra values

Input: List of fi , list of fr, list of affected users.
output: Delete the previous cycle contribution of the affected users.
forall *user nodes(u)* **do**
  if $\phi 1(u) == f_r$ **then**
    loss(f_r) -= $d_2(u)$ -$d_1(u)$
  end
  if d_1*(u,fi)* $< d_1(u)$ **then**
    gain(f_i) -= max(0, $d_1(u)$-d(fi,u))
  end
  if d_1*(u,fi)* $< d_2(u)$ **then**
    extra(f_i,f_r) -= $d_2(u)$ - max($d_1(u, fi)$,$d_1(u)$)
  end
end

3.3.5 Updating First and Second Closest Open Facilities:

As explained in section 3.3.2 the swap of the facilities makes the $\phi 1(u)$ and $\phi 2(u)$ of the affected user nodes invalid. Therefore, the closest and the second closest open facilities have to be updated to enable the program from iterate to find a better solution (if it exists). The information needed to update the first and second closest open facilities, $d_1(u)$ and $d_2(u)$, are obtained from the shortest path matrix that has been computed at the beginning of the implementation.

Resende and Werneck [77] had proven that the update of the $\phi 1(u)$ and $\phi 2(u)$ facilities to the node is a cheap operation for all the mentioned cases of the affected users (see section 3.3.2). The following steps are implemented to update the $\phi 1(u)$ and $\phi 2(u)$ of a user node as explained in Algorithm 7.

1. If an open facility is closer to the user node than other open facilities, then set it as $\phi 1(u)$.

2. If an open facility is closer to the user node than other open facilities and does not equal $\phi 1(u)$, then set it as $\phi 2(u)$.

Algorithm 7: Update the first and second closest open facility for a user

```
Input: List of open facilities, shortest path matrix.
output: Update the first and second closest open facility for the user nodes.
forall affected user nodes(u) do
    forall open facilities (f) do
        if d1(f,u) < d1(u) then
            φ1(u) = f
        end
    end
    forall open facilities (f) do
        if (d2(f,u) < d2(u) AND f != φ1(u) then
            φ2(u) = f
        end
    end
end
```

3.4 Testing the Centralized Approach

The main purpose for implementing this successful and interesting centralised solution of the p-median problem is to validate the proposed distributed protocols (described in Chapter 4).

An extensive set of trials has been carried out using different parameters. These parameters are different topologies with different number of m and p facilities arbitrarily selected.

Internet-like topology BRITE (Boston university Representative Internet Topology gEnerator), which is a parametrised topology generation tool [85], is used to generate the tested topologies.

In figure 3.5, a topology of 1000 nodes in a single Autonomous System (AS) is used in the trials of observing the cost of the solution, 100 candidate facilities are randomly selected (m = 100) and 25 facilities of them are opened

(p = 25). 10 trials with different random seeds are executed. The mean cost and the standard deviation of these trials shows reduction in the cost of the solution at each swap of facilities pair, until the best locations of p facilities among the m potential locations of facilities are found.

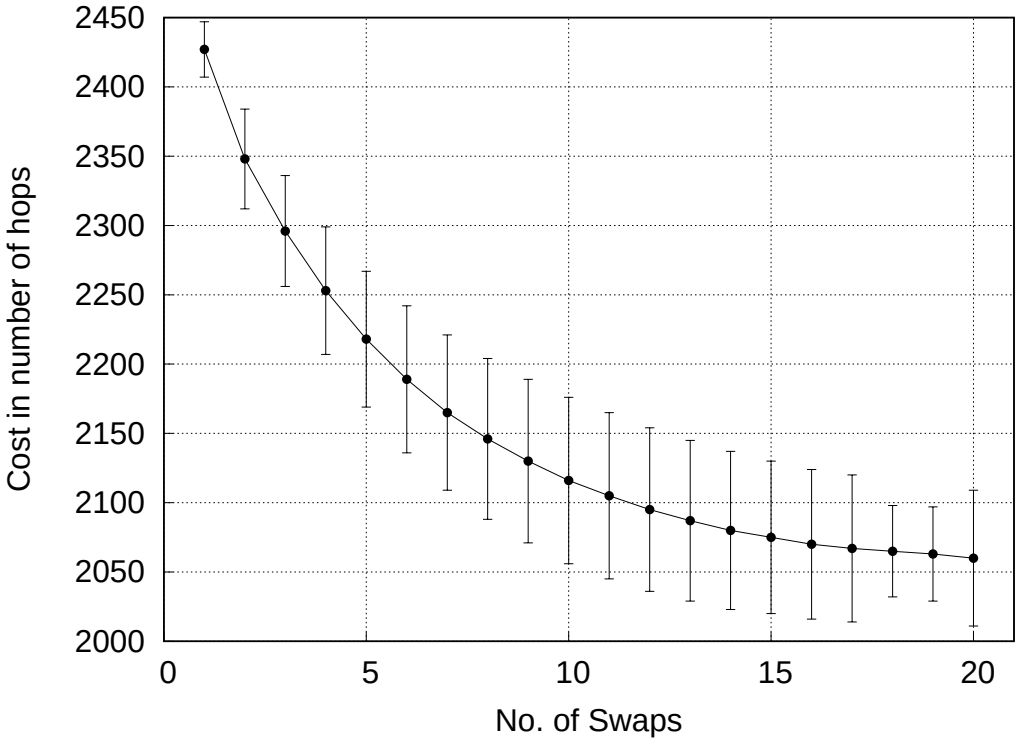


Figure 3.5: Cost of the solution (mean and standard deviation) over 10 trials for N=1000, m=100, p=25

The other set of trials is implemented on a different number of m (10 trials on each number (m)).This means different ranges of search spaces are given to find the optimal p locations within it. Many trials have been implemented with each value of m. The mean cost and the standard deviation of these trials show that the total cost of the solution is better (lower) when the area of choices is more comprehensive (see figure 3.6).

On the other hand, on a fixed range of search (N=1000, m=100) a set of trials is implemented with different number of open facilities (p = 10 to 50). As shown in figure 3.7; the cost of the solution (mean and standard deviation)

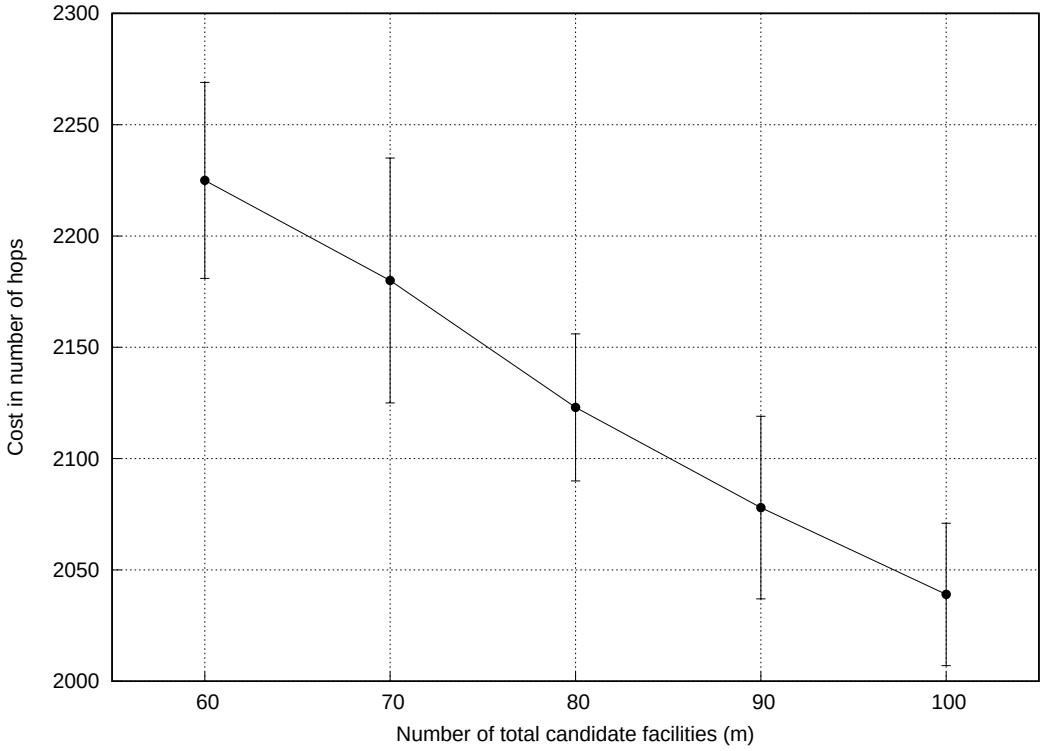


Figure 3.6: Cost of the solution (mean and standard deviation) for different ranges area of search (m) and N=1000, p=25

is reduced whenever the number of open facilities is increased. This is because the user nodes find open facilities closer to them rather than taken longer distances when the open facilities number is smaller.

However, on the same set of trials it is noticed that the number of sequenced swaps to convergence is increased whenever the number p is increased which means longer time is needed to converge as seen in figure 3.8.

3.5 Summary

This chapter explained one of the essential approaches for solving the p-median problem. The different parameters that used for solving the problem were also revealed. The chapter also showed the results obtained from running the centralised approach which has been used later to validated the proposed distributed approach (DPM).

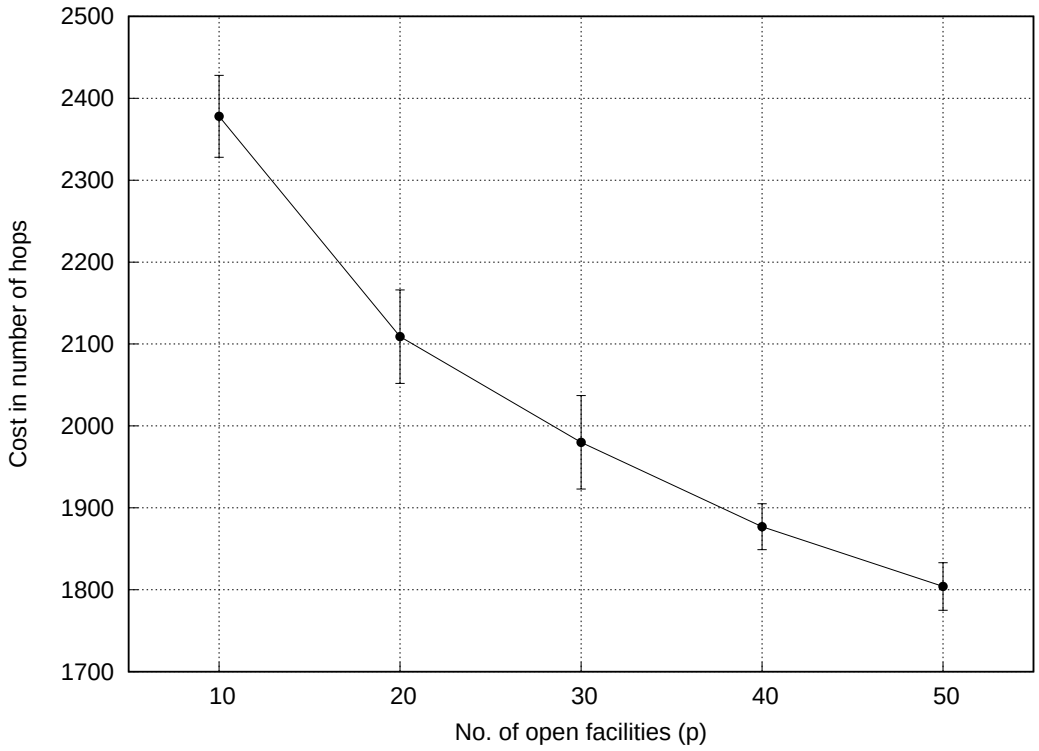


Figure 3.7: Cost of the solution (mean and standard deviation), 10 trials for each number of open facilities (p) and N=1000, m=100

However, the algorithm experience bottleneck at the presence of data in one place which infeasible to be implemented in a computer network. Therefore, distributed solution for solving the p-median problem was proposed as explained in Chapter 4.

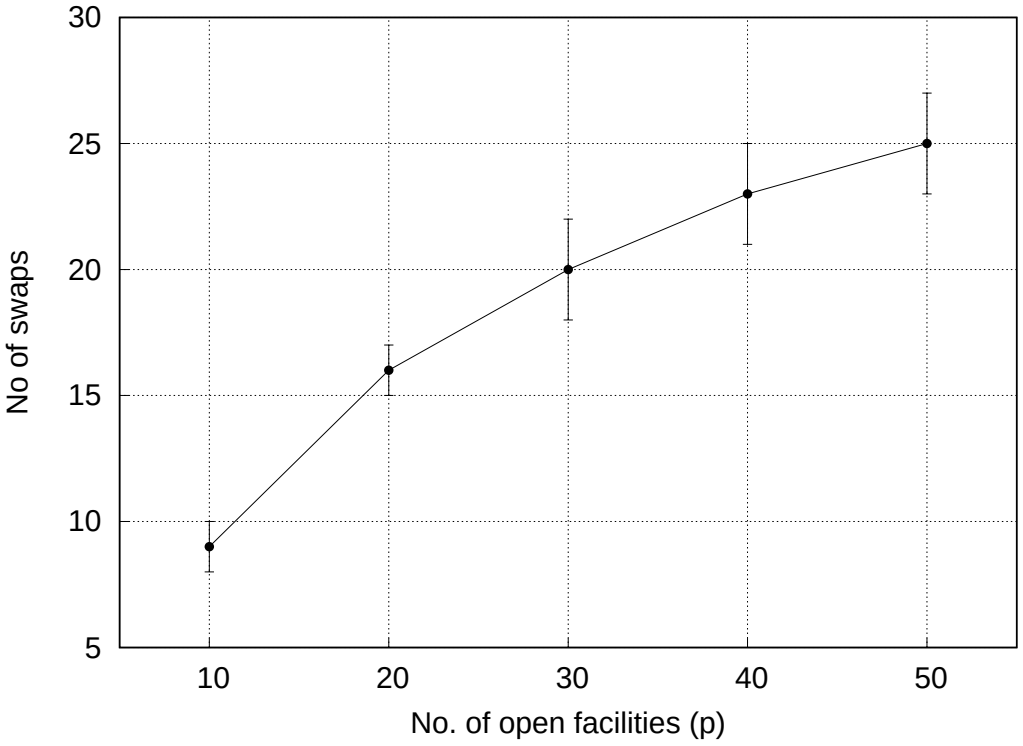


Figure 3.8: Number of swaps in the solution in different cases of the open facilities, N=1000, m=100, p = [10 to 50]

Chapter 4

The Distributed p-Median protocol (DPM)

As it was discussed in Chapter 2, many solutions for the p-median problem had been suggested. However, all the solutions were achieved via a centralised approach. This chapter explains the novel proposed distributed approach to find the p-median locations in computer networks. The proposed distributed formulation is based on the logic of the centralised approach which is described in (Chapter 3).

4.1 Introduction

This chapter explains, in details, the design and implementation of distributed p-median protocol to find the near-optimal p locations in a computer network that optimise a cost function. During the protocol implementation, the network topology and the data stay intrinsically distributed in the network. The proposed protocol is executed in parallel at all nodes to disseminate and compute the required information for determining the near-optimal locations for facilities.

Given a set U of n users, a set F of m candidate closed facilities and p integer number of facilities which are randomly selected from F to be opened. The proposed protocol applies an iterative heuristic approach to improve the initial solution and converges to a local optimum final solution. At each iteration, the protocol gradually builds a distributed view of the network which agrees for a swap operation in order to enhance the solution. This is enhanced by inserting a new facility into the solution and removing an open facility out of it. This operation is repeated by the protocol for a number of cycles until no improvement can be achieved.

The next sections discuss, in details, the work on the distributed p-median protocol (DPM).

4.2 An Overview of the DPM Protocol

The main challenge of implementing the protocol in a distributed environment is the lack of central data that can be used to find the best locations. Therefore, the DPM protocol is designed to find the required information from the user nodes and to process that locally and in parallel at each open facility without collecting the information on a central server. However, before explaining the working mechanism of the DPM protocol, the notation used in this protocol are are explained in table 4.1

G(N,E)	A topology, where N is the set of nodes and E represents the links between nodes
F	Set of potential locations for facilities, where F $\subset$ V
U	Set of users, where U $\subset$ V and U= V $\setminus$F
p	The number of facilities to be opened
m	$\lvert F \rvert$
N	Set of nodes in a network (user nodes and facilities nodes)
d1(u)	Shortest path in number of hops between a user node and its ϕ1(u)
d2(u)	Shortest path in number of hops between a user node and its ϕ2(u)

Table 4.1: Notation adopted in DPM protocol

Suppose a set U of n user nodes, a set F of m facilities and an integer p, where F $\subset$ U and p $<$ m. p facilities are randomly picked up from F to be opened. Initially, each node does not have any information about the system. The protocol starts by broadcasting messages from all the facilities in the network to all nodes. Once each user node receives the broadcast message; it first stores the message payload, then forwards it to its neighbors until all the network receives the broadcast information.

From the broadcast messages, the user nodes build a list of the available facilities with short distances to them and their status (opened or closed), then the following steps are implemented:

1. Determining the closest open facility.
2. Computes its weight in the network. The weight of the user node is its cost to join the closest open facility in addition to a list of its cost to join other facilities in the solution, as will be discussed in section (c).
3. Join the closest open facility by sending the JOIN message with its weight as a payload.

Each open facility gradually builds a local view from the received join messages from user nodes, then they perform the following:

1. Collect information about their clusters.
2. Compute its weight.
3. Compute a partial weight for the closed facilities in their clusters.
4. Exchange the required information with other open facilities.

After exchanging the open facilities information, the same global view of all network clusters is configured at each open facility. Thus, the same

decision is taken in all open facilities to swap one of the open facilities with one of the candidate facilities to open in a way that the overall cost of the solution is reduced.

After the facilities swap, the user nodes find a new open facility (if it is changed) and join it. The open facilities, update their local information accordingly in order to find another pair of facilities to swap.

These processes (facilities swap, join of user nodes to their closest open facilities, update the facility information and exchange the local information among the open facilities) are consequently repeated in order to improve the solution until the solution converges and the final median locations for the facilities are found.

4.3 The DPM Protocol

To simulate the DPM protocol, a topology G= (N, E) is given to the simulator as an input. The DPM protocol also assumes the following inputs:

1. A set U.
2. A set F of m candidate facilities.
3. An integer number p.

p open facilities are randomly chosen from F to be opened as an initial solution, where $p <$ m. The DPM protocol is implemented in a network environment, where nodes work asynchronously. Therefore, the implementation is made in consecutive phases to prevent overlapping in their functions and to make sure that the correct decision is taken. Three phases are designed to implement the algorithm as follows (figure 4.1 shows the graphical view of the protocol phases).

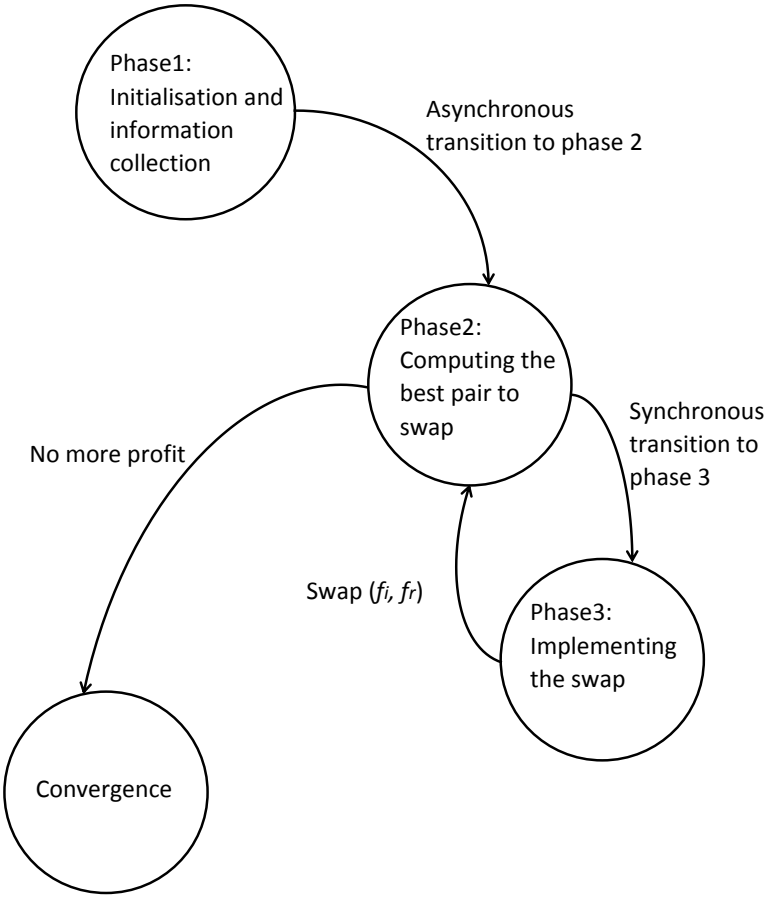


Figure 4.1: A graphical view of the DPM protocol phases. It consists of three main phases: in phase 1, facilities information are disseminated over the network and user nodes join their closest open facilities. Phase 2, finding the best pair of facilities $< f_i , f_r >$ to swap. Phase 3, implement the swap by closing f_r and opening f_i. Phase 2 and 3 are repeatedly executed until convergence to a local optimum and no more swaps can improve the solution.

1. **Phase 1: initialisation and information collection**

   The DPM protocol is started by making the open and closed facilities to advertise themselves via sending broadcast messages holding the required information to the nodes surrounding, as shown in Figure 4.2.

   On the other hand, the user nodes determine their closest facilities from these messages. The user nodes are clustered around the open facilities by joining up the closest open facility. The open facilities collect the information from their join users and compute the local weight of the clusters.

2. **Phase 2: Computing the best pair to swap**

   In this phase, the current open facilities exchange the local information of the clusters to build a distributed global information of the network. Based on the global information; each open facility determines the best pair $< f_i , f_r >$ to swap, the same decision for swap is taken in all open facilities.

3. **Phase 3: Implementing the swap and synchronize**

   Once f_i and f_r are determined; the open facilities in this phase implement the swap as in the following steps:

   (a) f_r notifies the candidate facility (f_i) to be opened about the swap.

   (b) f_r notifies all the user nodes in its clusters about the swap.

   (c) All other open facilities notify their user nodes about the swap.

   (d) The candidate f_i changes its status to open and the candidate f_r changes its status to close.

However, when the user nodes are informed about the swap, one of these two possible actions may take place:

(a) If f_i is closer to the user node than to its current open facility; the user node leaves its current facility and joins f_i.

(b) If f_i is closer to the user node than to its second closest open facility; the user node updates its record. The user record contains information about all facilities (opened and closed) in the network.

At this point, the open facilities transit to phase 2 to determine another pair of facilities to swap.

4. **Convergence state:**

While there are no more swaps that can improve the solution, all the facilities transit to this state and the solution is converged in local optimum.

Section 4.4 explains the details of the algorithms that implement the DPM protocol phases.

4.4 Implementation of the DPM Protocol

This section outlines the implementation of the DPM protocol. The DPM protocol is implemented in a fully distributed and asynchronous environment.

To implement the DPM protocol; the PeerSim [86] network simulator is adopted. PeerSim is a Java-based discrete-event peer to peer simulation tool. It allows simulation of a large size of the network on different configurations. In addition to its ability to observe the needed information in the nodes of the simulated graph, it keeps track of the messages between the facilities and the user nodes; this helps to analyse the results of the suggested protocol.

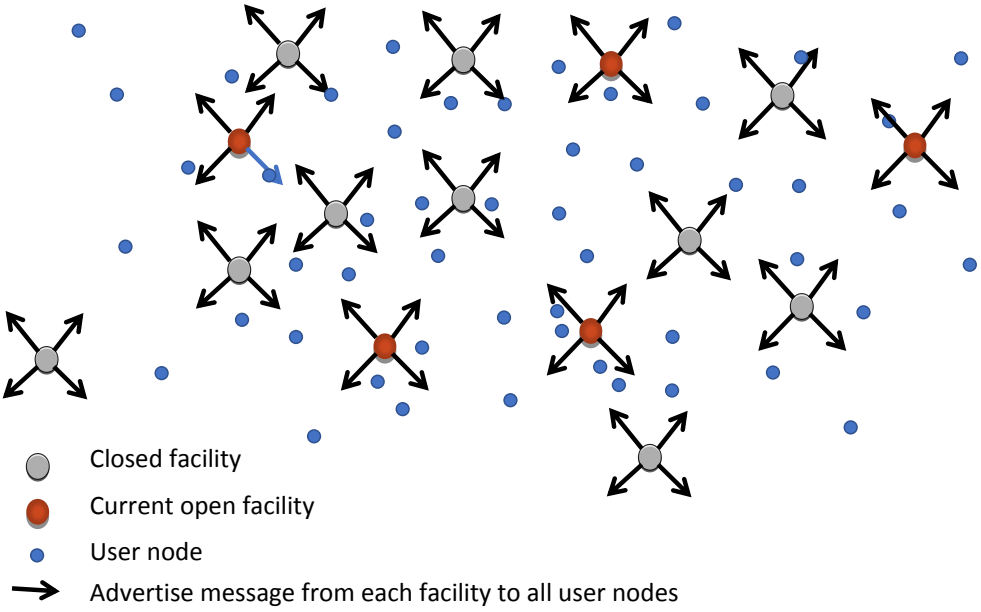


Figure 4.2: Facilities Advertisement: all the facilities (f_i and f_r) send a broadcast message to all nodes. Message payload:< FID, Distance and Status >

The sections below explain the details of the phases to implement the DPM.

4.4.1 Phase1: Initialisation and Information Collection

As mentioned previously; the main aim of the protocol is to minimize the cost of the solution. The first step to achieve this goal is to find the shortest path between the user nodes and the facilities.

As shown in algorithm 8, all the facilities (currently opened or candidate facility to be opened) broadcast an advertising message called a BROADCAST. Through this message, each facility announces itself. The message payloads are the facility ID, the facility status, the distance which is initialised at zero and source of the message (the node ID that the message comes from).

From the broadcast messages, each user node builds a view about the facilities in the network. The view about the facilities is not only about which

facilities are around the user node, but also about the shortest distance to the facility as well as the status of the facility (opened or closed) as shown in figure (4.3).

The shortest distance and the status of the facilities are essential for the user node to compute its weight, then to join the closest open facility. The details of these functions are described in the following.

(a) **Build a user partial view**

As aforesaid at the beginning of the DPM implementation, each node does not have any information about the network. To join the closest facility; the user nodes need to know the distances to the facilitates around them. Therefore, at the beginning of this phase, user nodes are kept waiting until receiving the facility broadcast messages.

As soon as a user node receives a broadcast message; it starts building its *local user record* from the broadcast message information.

As shown in figure 4.3, the *local user record* is built by extracting the information from the broadcast message payload. This information includes the ID of the facility that sends the message, the shortest distance to the facility and the status of the facility (opened or closed). The short distance to the facility is computed as in this section (b).

(b) **Building the short distance to a facility**

Finding the shortest distance from the facilities to the user node is computed during the travel of the broadcast message through the network. The BROADCAST message propagation is started from the facility, the facility sets the distance field to (one) and sends the message to all neighbors. As explained in algorithm 8, there are two possible recipients of

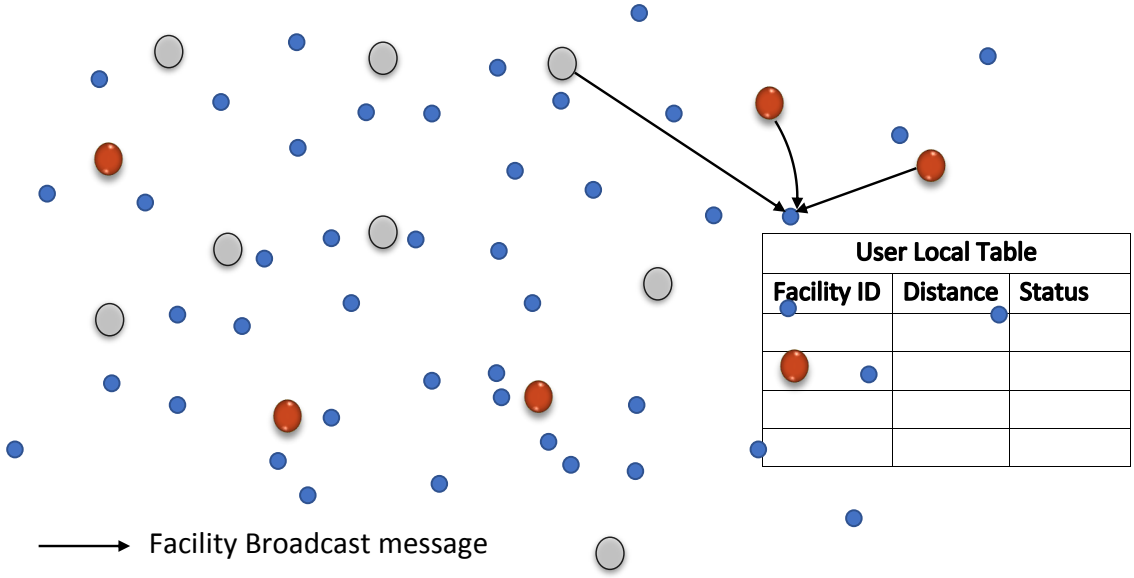


Figure 4.3: User local record, each user node build its local table from the receiving broadcast messages, which contain the facility ID, the short distance to the facility and the status of the facility

the message, while the message transmits over the network, either a user node or a facility node.

In the case of user node recipient, it implements the following steps:

(a) If this message comes from a new facility, (the first BROADCAST message reaches the user node from this facility), the user node records the message payload in its local record.

(b) Increments the distance field of the message by 1. This is in view of the fact that the distance is measured in the number of hops and the message becomes one node far from the advertised facility.

(c) Sends the message to the neighbors except the forthcoming source of the message.

The other case is when the recipient of the message is a facility node (opened or closed), it implements the following steps:

Algorithm 8: Phase1 initialisation and information collection, all the facilities send BROADCAST messages < FID, D, status > to all nodes. The user nodes build summarised view about the facilites in the network.

```
forall facilities (open and closed) do
    Send a facility broadcast mesaage m < FID, distance =1, status > to all
     neighbors
end

---

At the event a facility BROADCAST message is received at a user node:
if the m.fID is not in the user record then
    Add the message payload to the user local table
    Forward the message to all neighbors except the source
    else if the m.distance field < the current record.distance then
        Update the local table
        m.distance ++
        Forward the message to all neighbors except the source
        else
            Drop the message /* The same message come from a longer path*/
        end
    end
end

---

At the event a facility BROADCAST message is received at a facility node:
if the m.fID != the facility ID then
    m.distance
    Forward the message to all neighbors except the source
    else
        Drop the message /* The message of the same facility */
    end
end
```

(a) If the message facility ID equals the current ID facility (which means the same advertised message of the facility, but it has reached the node from a different neighbor). Then the facility drops the message. Otherwise, implement steps 2 and 3.

(b) Increments the distance field of the message by 1.

(c) Sends the message to all other neighbors except the source that the message from which it comes.

As noticed from the third point of both cases; the broadcast message is propagated to all user nodes around the facility. Therefore, the user node

may get the same advertised message from different paths. For this reason, the user node compares the distance field of the received message with the corresponding distance field of the same facility in its local record.

If the message comes from a shorter path, as shown in figure 4.4, then the distance field is updated before sending the message to the neighbors. Otherwise, the message is dropped.

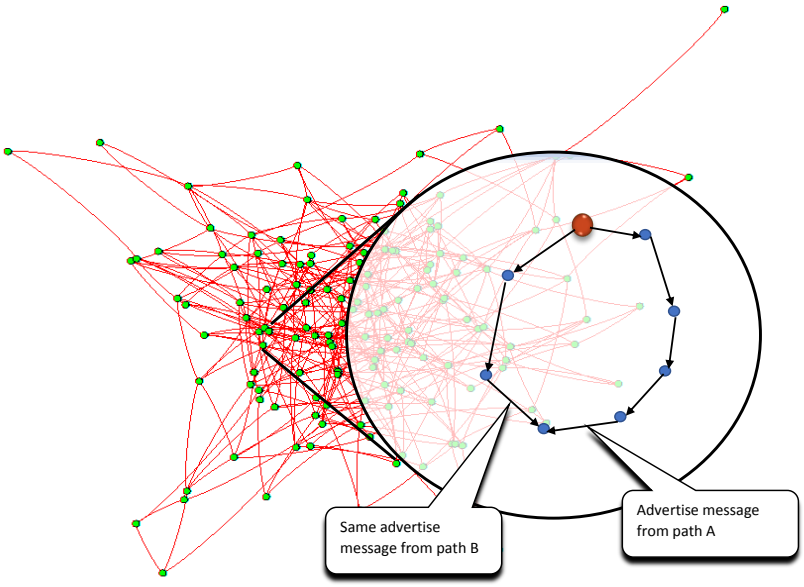


Figure 4.4: A user node receives the same broadcast message but from different path. It is true that path A is longer than path B, however, the message might reach the user from a longer path due to the network traffic

At the end of this stage, the user node has received different broadcasting messages from different facilities or from the same facility but from different paths. The local user record is built, and each user node has the following information:

(a) The list of the available facilities in the network.

(b) The shortest distance to reach each facility.

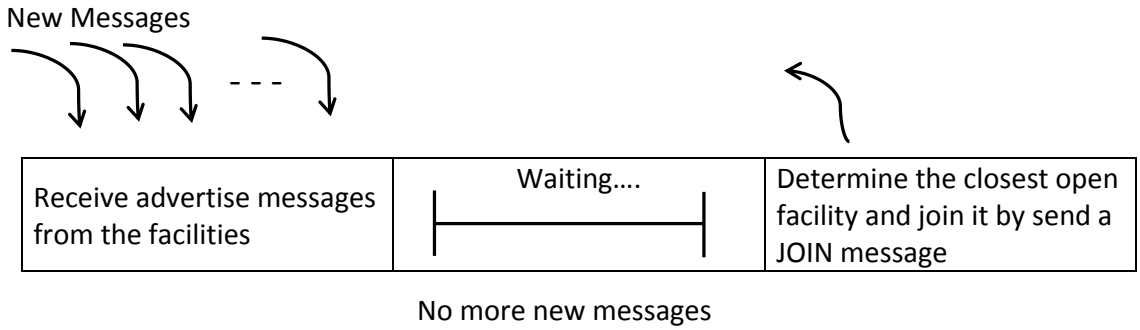


Figure 4.5: The user nodes receiving broadcast messages. While there are no more messages, the user nodes determine the closest open facility and join it.

(c) The status of the facilities (opened or closed).

The user node is now preparing to join the closest open facility as explain in this Section (c).

(c) **Join the closest open facility**

The user nodes are constantly updating their local record as they are receiving new broadcasting messages as explained in section (a). However, if there are no more broadcast messages, a user node will wait for number of cycles in order to decide that no more such cases are presented, see figure 4.5.

It is difficult to determine the end of the BROADCAST stage. All facility nodes need to determine the correct moment to stop flooding identification messages. Moreover, user nodes also need to regulate the same time point to transit to the closest facility selection procedure. Because of the decentralisation environment of the network; each node in the system must determine the end time point of the BROADCAST stage, based on local information and using a decentralised mechanism.

The endpoint of the BROADCAST stage is the time required when each node in the system has achieved a global convergence on the number of facility nodes and a shortest distance from a user node to every facility node. Ideally, each node has to detect global convergence using local information without any centralised server or coordinator. The local detection of global convergence requires a flexible and robust mechanism that can estimate the required time period for all nodes in a system to converge [87]. The DPM uses a cycle counter γ to express for the time period of convergence in each node.

The local value of γ_n in each node n is initially set to 0. Furthermore, γ_n is reset to 0 at each time the node n receives a BROADCAST message and finds out new information in it. An update becomes necessary for the local facility cache. However, in the case a user node receives a BROADCAST message, that does not carry any new information, γ_n is then increased by 1. In this way, the increment of γ_n is interrupted each time an update is applied until no new information is received. The continuous increment of γ_n over several consecutive cycles, in such a case, is interpreted as a local detection of convergence.

However, the global detection of convergence what must be obtained locally. The detection of global convergence at each node n is achieved locally when γ_n exceeds a predefined threshold of Γ. The threshold Γ is the time (in number of cycles) required, for all nodes in the system, to achieve global convergence. The determination of Γ depends on the network size, network topology, communication latency and other factors.

Although Γ can be computed during a network setup using decentralised routing protocols or traverse search algorithms; a simpler method is used to determine Γ for various system sizes in the simulations of the DPM. The method is a function of network diameter in the number of hops to cycle-length of the simulations. However, Γ is computed as follows:

$$\Gamma = D \times L \div CL, \tag{4.1}$$

where D is the network diameter in a number of hops, L is the maximum communication latency in a time unit, and CL is the cycle length in a time unit.

The network diameter D is the shortest distance between the two most distant nodes in the network. Typically, it is the longest distance in a set of shortest distances among nodes in a network. D represents the linear size of a network. Practically, various network sizes with a random topology are used in the simulations. The centralised Dijkstra procedure is used to approximate the average network diameter D for typical network sizes. Figure 4.6 illustrates examples of different values of D in different network sizes.

At this point, the user node is considered to have all the required information to compute its weight in the network.

The weight of the node is the cost of a node in the current state of the solution and its cost if its current open facility swapped with any of the candidate f_i. To cover the possible cases of the solution update; each user node computes the following:

1 The cost of a user node if $\phi 1$(u) is removed. Because the algorithm considers that the user node will connect to its second closest fa-

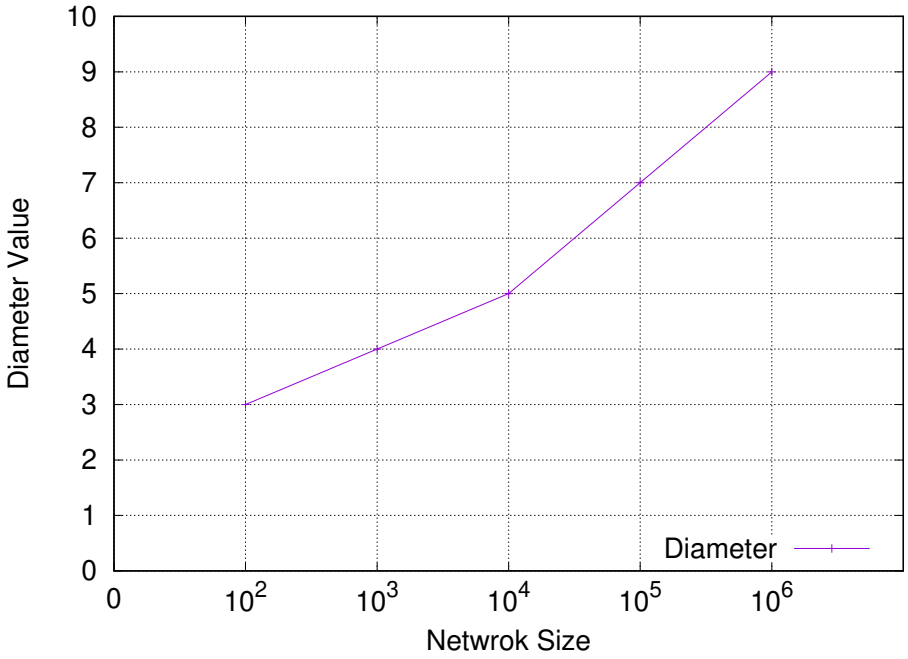


Figure 4.6: Typical diameter values in different network sizes

cility $\phi 2$(u) if $\phi 1$ is removed; the user node costs the solution the additional number of hops from $\phi 1$ to $\phi 2$. This loss in the cost is computed according to equation (4.2)

$$user_{loss}(f_r) = d2(u) - d1(u) \tag{4.2}$$

2 The cost of a user node when a candidate f_i is inserted closer to the user node than its current closest open facility, then the user node will gain a number of hops. This number represents the difference in distance between its current closest open facility and the candidate f_i. This gain can be computed according to equation (4.3)

$$user_{gain}(f_i) = \max[0, (d1(u) - d(u, f_i))] \tag{4.3}$$

3 The cost of a user node when a candidate f_i is inserted closer than the second closest facility while the first closest facility of the user is removed. In this case, the user node will join to f_i instead of

its second closest facility. Moreover, it gains the distance from the second closest facility to f_i. This is computed according to equation (4.4).

$$user_{extra}(f_i, f_r) = d2(u) - \max[d(u, f_i), d1(u)] \qquad (4.4)$$

After calculation of these three values; the user node sends a join message to the closest open facility. The join message payload is the loss value of the user, the gain vector of the user, and the extra vector of the user.

As shown in figure 4.7, at the event of receiving a join message; the open facility updates its local record as follows:

1 Add the loss value of the user to the local facility value.

2 Add the gain vector of the user to its corresponding facility gain vector.

3 Add the extra vector of the user to its corresponding facility extra vector.

After joining of the users; the open facility transits to phase2 and starts collecting the global information about the solution, as explained in section(4.4.2).

4.4.2 Phase2: Exchange the Necessary Information and Find the Best Pair to Swap

At the transition point of the open facility to phase 2; the cluster_loss(f_r) value becomes fully computed while both of cluster_gain(f_i) and cluster_extra(f_i,f_r) values are still partially computed. The partial computations of

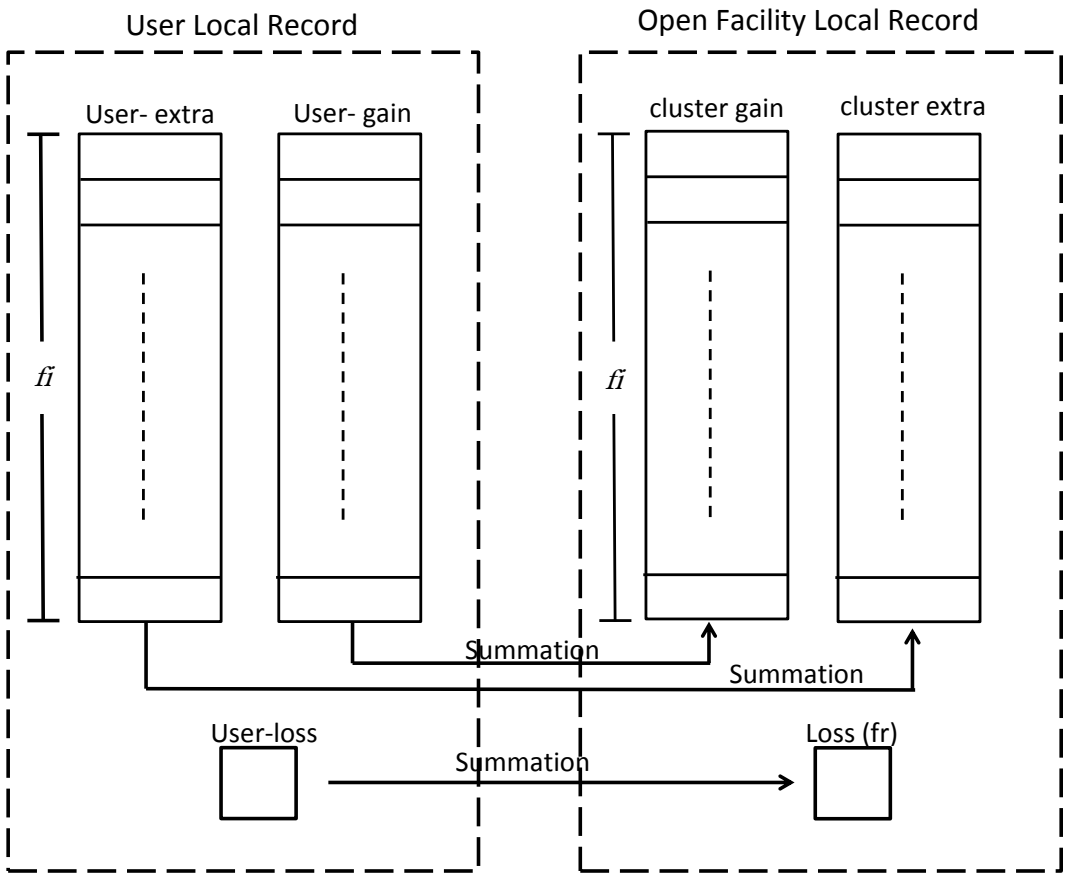


Figure 4.7: The open facility build a cluster knowledge from the JOIN messages of the user nodes

gain and extra are attributed to their values which consist of contributions of users from different clusters. Therefore, the open facilities exchange their local information to build the global information upon the network.

The exchange message payload is the cluster_loss, the cluster_gain, and the cluster_extra. As shown in figure 4.8; when the facility receives the exchange message, it implements the following steps:

1. Collects the cluster_loss(f_r) values to build a global loss vector of size p (number of open facilities).

2. Adds the local cluster_gain vector to its corresponding facility gain.

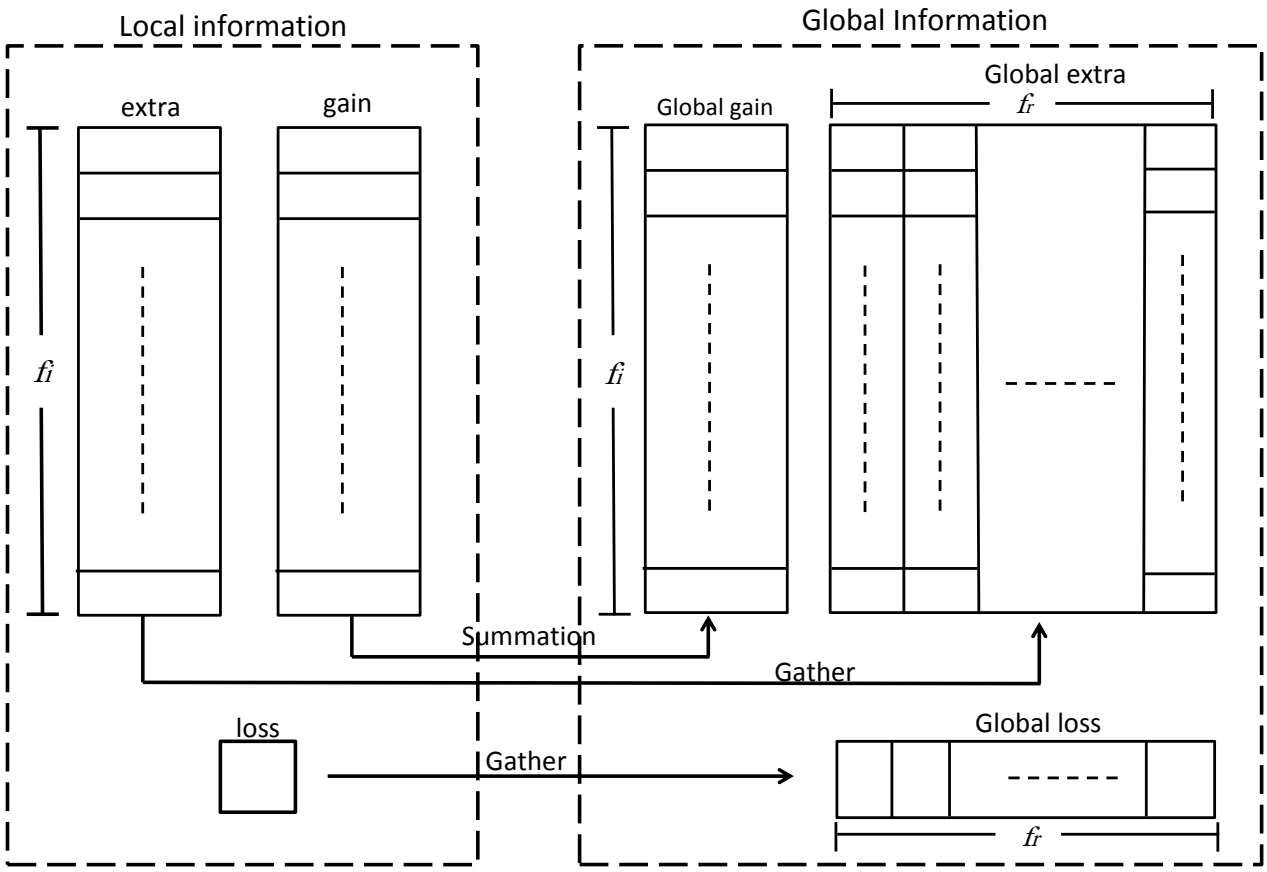


Figure 4.8: Exchange the open facilities local information and build a global view about the network

3. Collects the cluster_extra vectors of the exchange messages in order to make the global extra matrices of size(f_i,f_r).

When the number of the received exchange messages reaches to (p-1), as explained in figure 4.9, it means that the facility has received all the exchange messages, and all the required information to find the best available pair of facilities to swap.

The open facilities apply the profit equation (3.4) for all the candidate facilities. The pair of facilities (f_i, f_r) that make the highest profit is chosen to swap.

After completing the determination of f_i and f_r; the facilities transit to phase 3 to implement the swap.

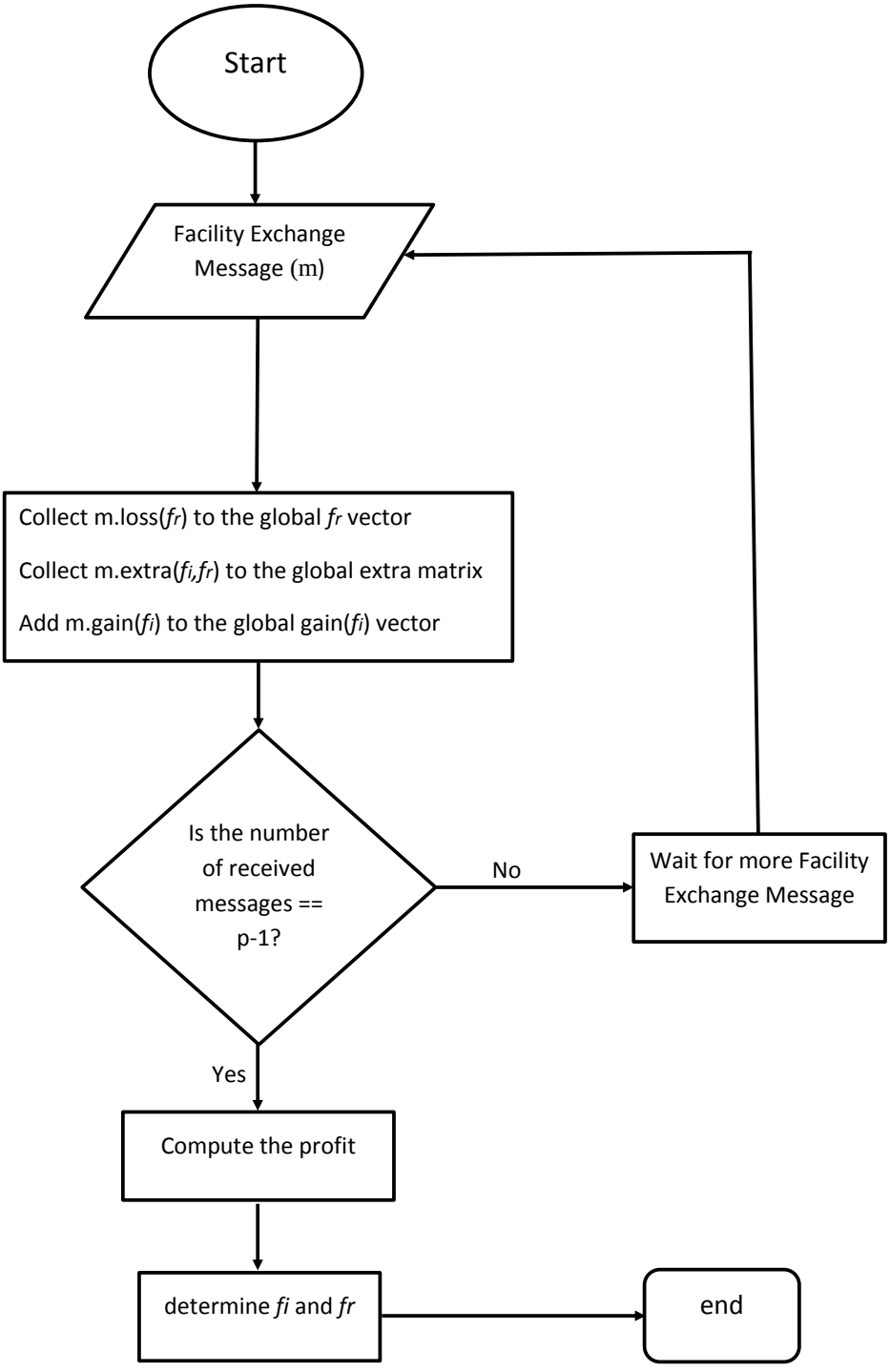


Figure 4.9: Facility exchange message flowchart

Algorithm 9: Phase 2, the open facilities exchange the local information and build a distributed global view about the solution. The same best pair of candidate facilities $< f_i , f_r >$ to swap are determined in all open facilitates.

```
Set counter to zero
p is the number of open facilities in the solution
max_profit = 0
candidate_f_r = null
candidate_f_i = null
forall open facilities do
    send the local information record to all other open facilities
end

---

At the event receive FACILITY EXCHANGE message
Add the message payload to the global table
counter++
if counter == (p-1) then
    for all f_i in the global facility record do
        for all f_r in the global facility record do
            compute profit (f_i , f_r)
            if profit > max_profit then
                candidate_f_r = f_r
                candidate_f_i = f_i
            end
        end
    end
end
```

4.4.3 Phase 3: Swap Implementation (if available)

After determination of f_i and f_r; all open facilities are transited to phase 3. As shown in algorithm 10, f_i and f_r are swapped in this phase through actions from both of them. The candidate f_r start the swap process through the implementation of the following steps:

1. Inform the candidate f_i to open by sending a CHANGE_STATUS message. When f_i receives this message; it changes its status to open and accepts the join messages from the user nodes.

2. Inform the users in its cluster about the swap by sending a SWAP $< f_i$, $f_r >$ message. As explained in algorithm 10, if the inserted facility is closer to the user node than its current $\phi 2$; then the user will join to f_i.

However, If the inserted f_i is closer to the user node than its current $\phi 2$; then the user node will update its *local record* to become correct, since it will be used in the next iterations.

Algorithm 10: Phase 3 implements the swap, the f_r sends two types of messages, the first one to tell f_i to open and the second one is to inform the user nodes in its cluster about the swap. The user nodes find the closest open facility and join it.

```
if This Facility == f_r then
    facilty.status = close
    send message CHANGE_STATUS to f_i
    forall Users in the f_r cluster do
        Send a SWAP < f_i, f_r > message
    end
end

---

At the event receive message CHANGE_STATUS:
facility.status = open

---

At the event receive a SWAP < f_i, f_r > message:
Update the local user record
if f_i closer than φ2 then
    Join the closest open facility
    else
        Join φ2
    end
end
```

The current open facilities in the system inform the user nodes in their clusters about the swap by sending an UPDATE_SOLUTION $< f_i, f_r >$ message. If f_i is closer to a user node than its current open facility, then the user is disconnected from their current $\phi 1$ and joins f_i.

When a user disconnects from a facility, the facility has to update its cluster local information by subtracting the user contribution from the cluster_loss, cluster_ gain and cluster_extra values. However, since the user contribution is directly added to the local cluster values of the facility; it is impossible for the facility to determine a specific user contribution. Therefore, the user sends its old contributions to the facility with the disconnection message payload, as explained in algorithm 11. Otherwise, the user node updates

its local record. The update of user local record is helpful in making the decision of the next iterations. At the end of phase3, all the open facilities are back to phase2.

Algorithm 11: Cluster information update, each open facility inform their user nodes about the swap to update their local table information and connect to their closest open facility (if it is changed) after disconnect from their current open facility.

```
if Facility.status == open then
    forall Users in the facility cluster do
        Send message UPDATE_SOLUTION < f_i,f_r >
    end
end
```

At the event receive message UPDATE_SOLUTION $< f_i, f_r >$:

```
if f_i closer than φ1 then
    send message DISCONNECT < user_gain, user_loss, user_extra >
    Join the f_i facility
    else
        update the user local table
    end
end
```

At the event receive a DISCONNECT $<$ user$_{gain}$, user$_{loss}$, user$_{extra}$ $>$ message:

cluster$_{gain}$ -= user$_{gain}$
cluster$_{loss}$ -= user$_{loss}$
cluster$_{extra}$ -= user$_{extra}$

The solution at this stage is reconfigured. Accordingly, the weight values of the clusters are changed. This might lead to find a different pair of facilities to swap in a way that can improve the solution. The process is repeated from phase 2, and the solution keeps losing cost until reaching the best status among the candidate locations of facilities.

4.4.4 Convergence State

The convergence is the termination state of the DPM protocol. It means that the solution can not find better locations among the candidate locations of facilities.

The open facilities are transit at this state. They stop trying to improve

the solution when they fail to find a profit by swapping any of the candidate facilities to open with one of the current open facility.

4.5 Summary

This chapter has presented the design and the implementation of a novel protocol (DPM) for addressing p-median locations in a computer network, not only without previous knowledge about the network but also without collecting the network data in a specific central node or a server.

The proposed protocol has been implemented in three phases to build a global view of the network and to address the near-optimal location for facilities.

During the first phase, all facilities broadcast an advertising message and the open facilities build up a local view about their clusters. In the second phase, the open facilities have exchanged the information to build a global view about the network and to determine the pair of facilities (open and closed) that can make the most profit when they swap. While the third phase implements the swap between the selected facilities. The second and the third phases are repeatedly implemented to enhance the solution until convergence.

Chapter 6 presents detailed analyses of the experimental results of the DPM protocol, and a comparison with the classical clustering method of networks (described in chapter 5).

Chapter 5

A Distributed Approach Based on the k-medoids Algorithm (KM)

5.1 Introduction

In this chapter, one of the main contributions of this work is presented, the KM approach. The inspiration for this approach comes from a fundamental logical concept of the k-medoid clustering algorithm.

Given a topology G(V,E), a set U of n users and a set F of m facilities, where F ∪ U = V. The protocol aims to place k facilities from F such that the total distances from the users to the placed facilitates are minimised.

Initially, k facilities are randomly chosen from F to be opened, the KM approach clustered the network to k clusters according to the short distance from the user and close facility nodes to the closest open facility (medoid). Thereafter, each cluster updates its medoids dependently and synchronised with the other clusters. After the update synchronisation, if the closest open facility to a node is changed, then the node joins the new closest one. The updating of both clusters medoid and user nodes connection are iteratively continued until no better locations for the medoids can be found in all clusters.

Because of its similarity to the DPM in applied of heuristic iterative approach to improve the solution and in forming clusters based on the short distance between the user nodes and the open facilities node; the KM protocol has provided an interesting comparative analysis for the obtained results from the DPM protocol.

This chapter begins with the definition of the k-medoids algorithm. Then it follows with the design of the KM protocol in the distributed network and the splits of the protocol functions in several phases to overcome the asynchronous implementation in the network. Finally, the details of the protocol algorithms and the steps of finding the best location for the medoids are described.

5.2 k-Medoid Algorithm Definition

Generally, the k-medoid is a clustering algorithm, which aims to cluster a set of objects into k number of clusters [88] [89] [90]. Same as the proposed p-median solutions, the target of the k-medoid algorithm is to cluster the network in a way that the overall cost of the solution is minimised, in addition, the solution of the k-medoid algorithm is actual data points. The other similarity with the p-median problem is that the k-medoid also use the heuristic to iteratively process until convergence [89].

In the k-medoid clustering algorithm, the objects are clustered around medoids, the most centrally located object of the cluster, such that the total distance for the travelling of user nodes is minimised [91][92]. To achieve this aim, the user nodes are clustered according to the shortest distance to the closest medoid (open facility node).

As described in the algorithm 12 [93], the classical k-medoid algorithm is started by arbitrarily selection of k number of nodes to form the initial clusters. The other nodes join its closest medoid forming k clusters based on the shortest path. After that, at each cluster, iteratively, a random node is selected and its overall cost is computed. If it is found less than the current medoid; then it will swap with the medoid. Clustering quality is progressively improved in each repetition until converging in a local optimum.

Algorithm 12: Classical k-medoids clustering algorithm.

Input:
N: a topology containing n nodes.
k: the number of clusters.
Output:
A set of k clusters.
Method:
Arbitrarily choose k nodes in N as the initial representative seeds (medoids).
repeat
  Assigning each node to the cluster with the nearest medoid
  Randomly select a non-representative object o_i
  Compute the total cost, S, of swapping the medoid m with o_i
  if $S < 0$ **then**
    swap m with o_i to form the new set of k medoids
  end
until *convergence*

5.3 An Overview of the KM Approach

The KM approach is proposed to address the near-optimal locations for facilities in a computer network based on the concept of the k-medoid clustering algorithm. The KM protocol is designed to be implemented in a distributed network environment, without a predefinition for the network topology. During the implementation of the KM protocol, the data of the network stays trifling in the nodes, and the user nodes or the closed facility nodes are gradually clustered around their closest open facilities (medoids) of the clusters.

Given a set N = { $node_1$, $node_2$, ..., $node_n$} of n networked nodes and a set F = { f_1, f_2, ..., f_m} of m candidate facilities. The KM protocol starts by randomly selecting k facilities from F to be opened. Thereafter, the user nodes and the closed facilities are connecting to the closest open facilities forming k clusters.

After clustering; the open facilities find the best location among the candidate locations within its cluster. The best location for a facility to be open is that can achieve the minimum total cost of the user nodes in the cluster.

All clusters are updating their medoids synchronously by swapping themselves with the best candidate closed facilities. The medoid informs the nodes in their cluster about the swap, and also informs the medoids of the other clusters about the cluster update. The user nodes and the closed facilities are connecting to their closest medoids if it is changed. Thereafter, all the new medoids are updating their information according to the new clusters.

Updating the medoids and reconfiguring the cluster are continued until converging to a local optimum and no more new set for the medoids in all clusters to be updated.

To implement the KM protocol, each facility starts sending a broadcast message to show their presence to the other nodes in the network. The closed facilities and the user nodes are gradually built a summarised view about the available facilities in the network. The summarised view is the list of the available facilities with their cost and the status of the facilities in the network, as shown in figure 5.1.

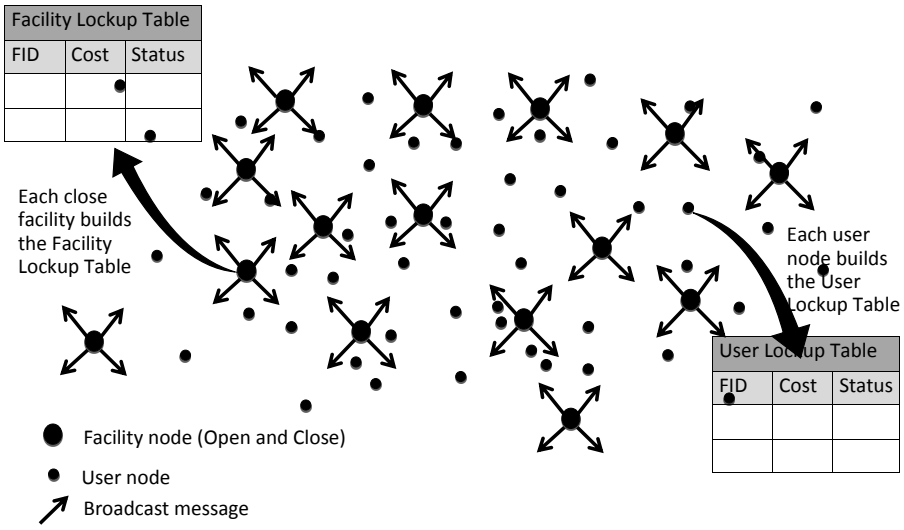


Figure 5.1: Phase1 in the KM protocol, closed facilities and user nodes build a summarised view about the solution from the broadcast messages

While there are no more broadcast messages, the user nodes and the closed facilities nodes send JOIN messages to their closest open facilities. From the JOIN messages, each open facility (medoid) computes the weight of its cluster. The weight of the cluster is the total cost of the joining users. Moreover, the medoid computes a list of weights of the cluster in case that any of the closed facilities in its cluster is opened. The closed facility that achieves the minimum cost for the cluster is chosen to be placed as a new medoid of the cluster.

The KM protocol is designed in three main phases in order to overcome the synchronisation problems of the network distributed feature. Each phase, implements a set of functions before synchronisation (if required) among nodes and transits it to the next phase. In section 5.4 the model of the KM protocol and its phases are described in more detail.

5.4 The KM Protocol

To simulate the KM protocol, a topology G= (N, E) is given to the simulator as an input. The KM protocol assumes the same inputs as of the DPM protocol (see Section 4.3), where k = p.

To overcome the asynchronous problem of the distributed environment; the KM protocol functions are separated into three phases. All the medoids are synchronised at the end of each phase. The functions of each phase are described as follows:

1. Phase 1: **Information dissemination and clusters configuration**

   This phase is started by the announcement of facilities about their locations. The facility announcement is to send its information through BROADCAST messages to the other nodes in the solution informing them about its location and its status (opened or closed).

   At the event of receiving a BROADCAST message; the user or closed facility node gradually builds a summarised view of the available facilities in the network. Thereafter, they join their closest open facility forming a cluster around it as shown in figure (5.2).

2. Phase 2: **Find the best pair to swap**

   In this phase, the medoid of each cluster evaluates all the candidate facilities to open in its cluster by computing a weights list of all closed facilities as shown in Figure 5.3. The weight of a closed facility in a cluster is its cost if it is opened instead of the current medoid of the cluster.

   The closed facility (f_i) that makes the minimum cost is compared with the current medoid cost, if the candidate facility to open (f_i) cost is lower

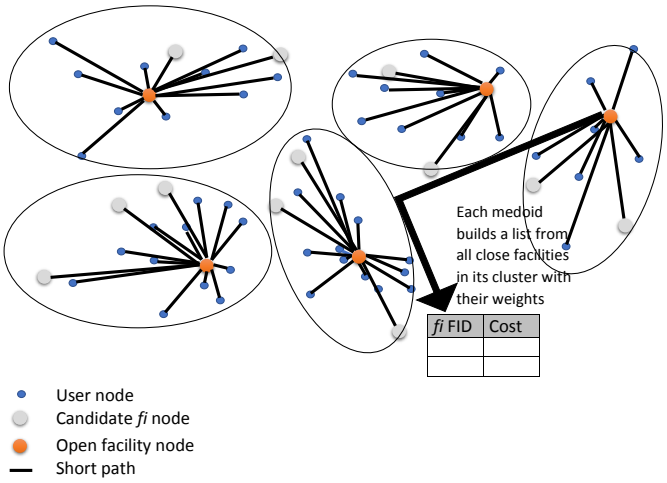


Figure 5.2: User and closed facility nodes are joining their closest open facility. The open facilities build a table from all closed facilities in their cluster in preparation for computing the closed facilities cost

than the medoid cost, then the solution is updated as described in phase 3. Otherwise, it may fall in local optima and can not find a better solution.

3. Phase 3: **Update the medoids of the clusters**

   This phase implements the swap between the current medoids and the determined facility to be medoid in phase2. The swap is implemented through the following steps:

   (a) *From the medoid site*

      i. Inform the candidate facility to insert about the swap.

      ii. Inform the other closed facilities in the cluster about the swap.

      iii. Inform the user nodes in the cluster about the swap.

      iv. Inform the other medoids in the network about the swap.

   (b) *The user node site:*

      When the user node has notified about a swap; it does not join the new medoid directly. However, it determines the closest medoid

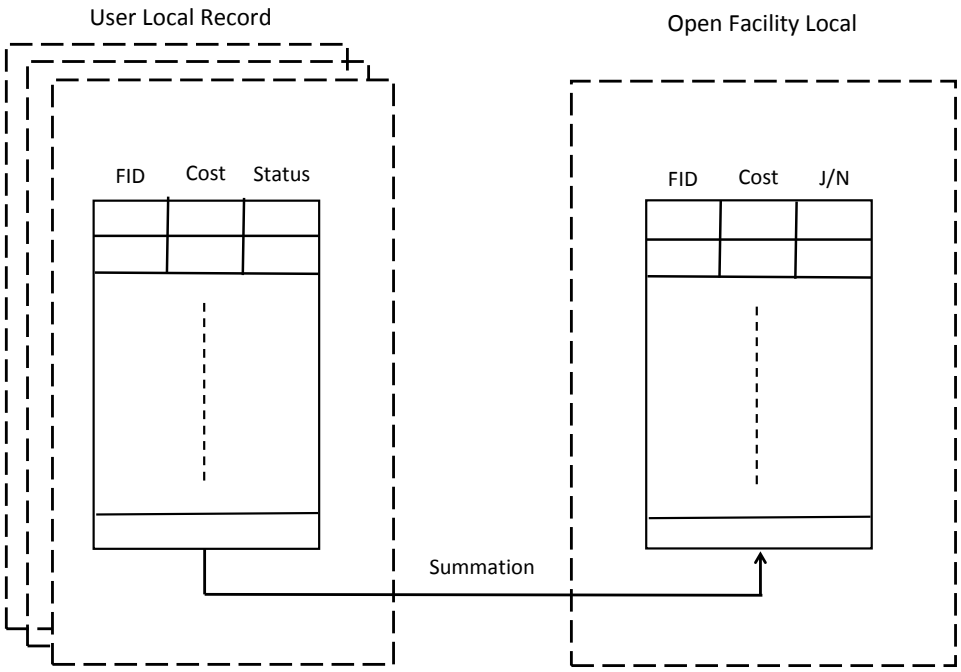


Figure 5.3: Computing the cost of all f_i in the cluster through the summation of f_i cost in all user nodes in the cluster

from the *local user record*, even if it is from another cluster, and joins it.

The medoid informs the user not only about the swap in its cluster, but also about the swaps occurred in other clusters.

(c) *from the closed facilities site:*

Same as for a user node; each closed facility waits for the swap messages, it determines the closest medoid then joins it.

(d) *The medoid of the other clusters:*

The medoid of a cluster informs its user about the update of the other clusters to enable them updating their local record tables and joining the closest available medoid to them.

All the medoids are back to phase 2. The steps in phase2 and phase3 are repeated until convergence.

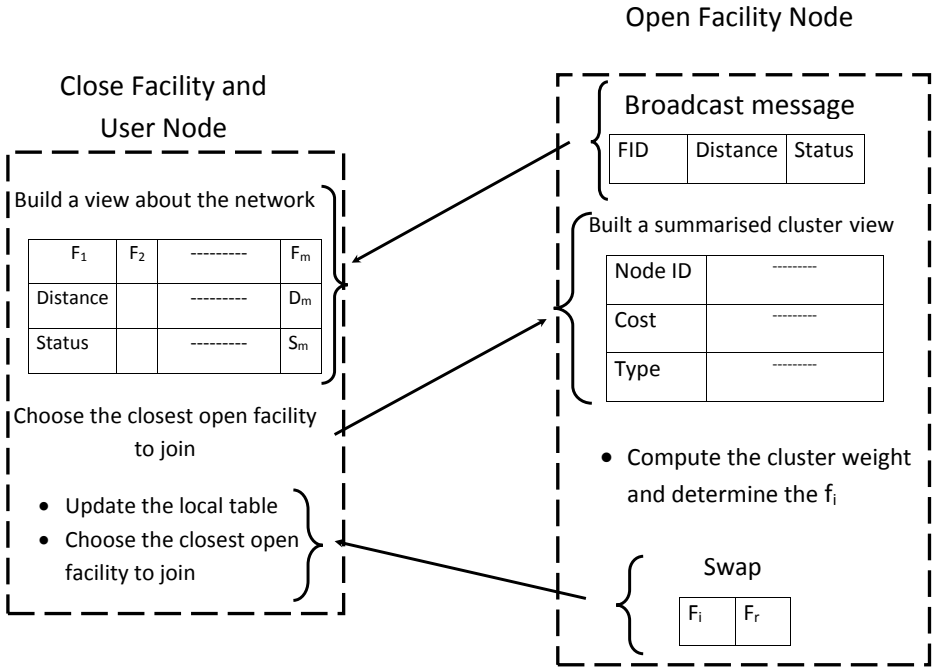


Figure 5.4: The main functions of the nodes during the phases of the KM approach

4. **Convergence**

   When a medoid does not find a better closed facility in its cluster to swap with it nor does it receives an update message from other clusters; the protocol is converged in a local optimum.

Figure 5.4 shows the main functions of all KM phases.

5.5 The Implementation of the KM Protocol

This section explains the details of the KM protocol implementation including the algorithms and the phases procedures. As mentioned in section 5.4, the KM protocol is implemented in three successive phases in order to control the asynchronous behaviour of the network nodes.

The PEERSIM simulator [86], which is a Java-based discrete-event P2P simulation tool, is used to test the protocol with different sizes of topologies and on different configurations as described in chapter 6.

The details of the phases implementation are described in the following sections:

5.5.1 Phase1: Information Dissemination and Clusters Configuration

In this phase, the user nodes and the closed facility nodes have to be clustered around their closest medoids. To implement the clustering; each user node and closed facility node must know the short distance to the medoids for joining the closest one.

Therefore, all the medoids and the candidate facilities to be medoids must announce about themselves at the beginning of this phase by sending a BROADCAST message to all nodes in the system as shown in algorithm(13). Within the BROADCAST message the short distance to the facility is built.

The KM protocol BROADCAST message is similar to the DPM protocol (see section (4.4.1)). However, building a view about the medoids from both the user nodes and the facility nodes, in addition to the joining procedure are differed from the DPM protocol, as described below.

(a) **Built a User View**

Since the KM protocol is implemented in a distributed environment; each user node will gradually discover the facilities around it from the received broadcast messages. As shown in figure 5.1, when the user node receives the broadcast message; it builds a local *user lookup table*. The user lookup table is a local table of each user that contains records for all facilities in the network. The record of each facility contains a summarised information about the facility ID, the short distance to the

Algorithm 13: The information about all facilities are disseminating to the network. User and closed facility nodes build a summarised view about the available facilities in the network

forall *facilities (open and closed)* **do**
  Send a facility broadcast message m <FID, distance =1, status> to all neighbors
end

At the event receive a facility broadcast message at user node:
if *the m.fID is not in the user local table* **then**
  Add the message payload <FID, distance, status> to the user local table record
  m.distance ++
  Forward the message m to all neighbors except the source
  else if *the m.distance < the current fID.distance* **then**
    Update the local table
    m.distance ++
    Forward the message to all neighbors except the source
    else
      Drop the message /* The same message has received from a longer path*/
    end
  end
end

At the event receive a broadcast message at an open facility node:
m.distance ++
Forward the message to all neighbors except the source

At the event receive a broadcast message at a closed facility node:
if *m.fID is not in the local facility record* **then**
  Add the message payload <FID, distance, status> to the user local table record
  Forward the message to all neighbors except the source
  else if *the m.distance < the current fID.distance* **then**
    Update the local table
    Forward the message to all neighbors except the source
    else
      Drop the message /* The same message has received from a longer path*/
    end
  end
end

facility and the status of the facility (opened or closed). See table (5.1)

Node ID	f ID	f ID	f1 ID	f ID	f2 ID	f ID	- - -	- - -	fn ID
	d	d	d	d	d	d	- - -	- - -	Short distance
	closed	closed	opened	closed	opened	closed	- - -	- - -	Status

Table 5.1: User node and closed facility lookup table: This table is built during the receiving of broadcast messages forming a summarised view at the nodes available as facilities in the network

(b) **Built a Closed Facility View**

The closed facilities on the other hand, do not only broadcast the advertised messages, but also they built a summarised view about the available facilities in the network. As shown in algorithm 13, each closed facility builds its *facility lookup table* which, as the user node, contains a list of open facilities in the solution with their short distances, in addition to their statuses as shown in table (5.1).

(c) **Join the Closest Medoid and Clusters Configuration** While the user nodes and the closed facilities nodes receive the advertised messages; they keep updating their local tables until no more new messages are coming. As mentioned earlier, because the implementation of the protocol is in a distributed network; the end of the advertising stage should be distinctly determined to enable all the nodes to make the right decision by connecting to their closest medoids.

If a message from the closest open facility to a node is delayed and the node receives an advertisement message from other facilities, then it may join the wrong medoid. To overcome this problem; all the nodes must wait for a number of cycles after receiving the last message. The same procedure in the DPM protocol for waiting a number of cycles to join (Section (b)) is also adopted in this protocol. The number of cycles to wait is based on several factors to make sure that all the messages

reached their destinations. These factors are the diameter of the network and the maximum communication latency and the cycle length. Based on these factors, equation 4.1 in Section 4.4.1 is applied to determine the number of cycles to wait before determining and joining the closest medoid.

While there are no more new advertisement messages; the user nodes and the closed facilities determine their closest medoid (as shown in algorithm 14) and send a JOIN message to it.

Algorithm 14: Select the closest open facility

```
Initialise cost =Max integer number
Initialise closest-open-facility =null
forall facilities in the facility lockup table do
    if facility.status == open then
        if facility.cost < cost then
            cost = facility.cost
            closest-open-facility = facility.FID
        end
    end
end
```

In the JOIN message each of the users node and the closed facility node sends the list of the facilities and their corresponding cost to the medoid.

At the event of receiving a JOIN message; the medoids builds a *medoid lookup table* which contains a join list information about both closed facilities and user nodes in its cluster.

While there are no more JOIN messages all the medoids transit to phase2 and determine the best pair to swap as described in section (5.5.2).

5.5.2 Phase2: Find the Best Pair to Swap

In this phase, each medoid computes the weight of the cluster as shown in Algorithm 15. From the *medoid lookup table* each medoid computes the

weight of its cluster by summation the cost of the joining users as in equation (5.1).

Algorithm 15: Compute the cost of the cluster

```
Initialise cluster_cost =0
forall users in the medoid lookup table do
    cluster_cost += user.cost
end
```

Moreover, the open facility computes the cost of all closed facilities in the cluster. As shown in the Algorithm 16, the cost of a closed facility is computed by summation of the cost of the facility from all joining users records in the *medoid lookup table*, as in the equation (5.2).

$$cost(cluster) = \sum_{u:joinUsers} cost(u) \tag{5.1}$$

Form the computed weights; each medoid determines the best candidate closed facility to be a medoid. The closed facility that makes the minimum cost among other facilities in the cluster will swap with the medoid, as can be seen in phase3.

Algorithm 16: Compute the cost of the closed facilities in the cluster

```
Initialise cost_vector =0
forall closed facilities fi in the medoid lookup table do
    forall users in the medoid lookup table do
        cost_vector[fi] += user.fi.cost
    end
end
```

After the determination of the new medoids in all clusters; all the facilities transit to phase3 to implement the swap (see section (5.5.3)).

$$cost(f_i) = \sum_{u:joinUsers} cost(f_i) \tag{5.2}$$

5.5.3 Phase3: Updating the Medoids of the Clusters

In this phase all clusters update their medoids. Obviously, the cluster is updated if and only if a candidate f_i's cost is less than the current cost of the medoid.

All clusters update their medoids dependently and in parallel. The update process is implemented as following:

1. The medoid inform the candidate f_i with the swap by sending a CHANGE_STATUS message. As shown in Algorithm 17, at the event of receiving the CHANGE_STATUS message; the f_i changes its status to open and be ready for receiving join messages from the user and closed facility nodes.

Algorithm 17: phase3 update the solution, the current medoid is closed and the determined f_i is opened. All the user nodes in the cluster are informed about the swap as well as the other medoids in the solution are informed

```
if This is facility == fr then
    facility.status = closed
    send CHANGE-STATUS message to the fi
    send SWAP <fi,fr> message to the join users and closed facilities to inform them
      about the close decision
    send CLUSTER-UPDATE<fi,fr> message to the other open facilities in the
      solution
    phase = phase2
end
```

At the event a SWAP<f_i,f_r> message:
update the user local table
determine the closest medoid
join the new closest medoid

At the event CHANGE-STATUS message
facility.status = open

At the event facility CLUSTER-UPDATE
update the facility local table
inform the user nodes about the update

2. The medoid inform the users and other closed facilities in its cluster with the swap by sending them a SWAP(f_i, f_r) message. The user and the closed facility nodes update their local table by changing the status of f_i to open and the status of f_r to closed, determine a new closest open facility to them and join it.

3. The medoid informs the other open facilities in the system with the swap by sending CLUSTER_UPDATE(f_i, f_r) message to the medoids of all other clusters.

   At the event of receiving CLUSTER_UPDATE(f_i, f_r) message; the medoid informs the user and the closed facility nodes in its clusters about the swap.

   As described in Algorithm 18, when the user or the closed facility nodes receive the CLUSTER_UPDATE(f_i, f_r) message; They update their local lookup table and join their closest medoid.

4. Finally, the f_r medoid node releases all resources, change its status to closed, and runs the protocol using closed facility procedure.

Algorithm 18: CLUSTER_UPDATE (f_i, f_r) message. The user node updates their local lookup table and join their closest medoid.

```
Local record .f_i. status = opened
Local record .f_r. status = closed
if f_i. distance < medoid.distance then
    Disconnect from the medoid
    Join f_i
end
```

Due to the dissemination of swap information of each cluster to all other clusters, it becomes possible for a clusters nodes to choose and join an open facility that belongs to another cluster if it is closer to them than the current

medoid and, therefore, forming a new cluster with the newly selected medoid node. From that, the system clusters may be dynamically reconfigured to address the changes in one cluster.

All the medoids (open facilities) are then transited back to phase 2 as shown in Algorithm 17 to find another best solution in its new clusters.

5.5.4 Phase 4: Convergence

Phase2 and phase3 are repeatedly implemented and the cluster medoids are updated continuously until no more new update messages are sent; the facilities are transited to convergence phase, and the algorithm is terminated in a local optima solution.

5.6 Summary of the chapter

This chapter has introduced a proposed solution (KM protocol) for addressing the near-optimal location of facilities in a computer network. The KM protocol clusters the network into k clusters according to the short distance between the user nodes and the k open facilities. The KM protocol using a meta-heuristic methods to deal with each cluster separately and in parallel finds the medoids of the clusters.

The KM protocol inspiration comes from the k-medoid clustering algorithm which is found similar to the proposed distributed p-median protocol (DPM) in clustering the network and some other parameters. Therefore, it is found to be an interesting environment for comparison with the proposed DPM. It is true that the KM protocol also uses the meta-heuristic methods

for finding the medoids of the network as the DPM. However, KM uses completely different procedures to cluster the network and to address the best locations for the facilities.

The next chapter will present a detailed analyses of the experimental work for each algorithm described in Chapters 4 and 5.

Chapter 6

Experimental Results Analyses and Discussion

The DPM and KM protocols are designed to address the near-optimal locations for facilities in large-scale and complex hierarchical topologies such as the Internet. Therefore, they are tested on big sizes of different artificial internet-like topologies, in addition to snapshots of real networks.

This chapter presents and discusses the results of the experimental work that has been carried out to validate and evaluate the efficiency of the proposed work.

As would be seen in the next sections, both DPM and KM protocols are implemented in the Java-based P2P simulator PeerSim with different types and sizes of topologies.

The remaining sections in this chapter are organised as follows. 6.1 describes the environment of the experiments. Section 6.1.1 describes the simulator used to simulate the protocols. The dataset and the topologies are explained in section 6.2. In section 6.3, the results for validation and evaluating the DPM protocol are demonstrated, while section 6.4 demonstrates the results of the proposed KM approach. Moreover, a comparative analysis between the DPM and KM protocols are discussed, which involves testing

of the proposed approaches against various assumptions of a size for search spaces in a graph, as well as, different sizes of topologies. The final section 6.6 has summarised the discussed issues in this chapter.

6.1 Experimental Overview

All algorithms of the centralised approach and the distributed approaches were written in Java. The DPM and the KM approaches were implemented in a fully distributed way and without a central node or global knowledge using PeerSim [86] [94]. PeerSim is a Java-based network simulator tool, and is adopted to test DPM and KM protocols. Peersim allows loading for topologies from graph files.

The DPM and KM approaches are executed in parallel in both types of nodes (facilities and user nodes) and the results are collected in files using the observer (collection agent). The simulations were repeated using different random seeds until sufficient statistics had been collected. Various types of tests were run using different sizes of topologies and different configurations, as can be seen in the next sections.

6.1.1 The Simulator (PeerSim)

PeerSim is a Java-based network simulator tool; it was developed at University of Bologna and Trento, Italy as a tool for the university researchers, thereafter it was released under LGPL open source license [86] [94].

The network in PeerSim is formed as a list of nodes, and each node has its protocol, for example, in DPM or KM the open facility nodes have a different protocol than the close facilities nodes. In the DPM and KM approaches the specific protocol of each node might be changed during the real-time

implementation, for example, when the *fr* is closed the protocol in the node is changed from the open facility protocol to the closed facility protocol.

PeerSim has initialisers and controls. The initialisers are executed before the simulation to prepare the environment for the implementation, such as wire the nodes of the graph, determine the type of each node and load the specific protocol to the nodes. While controls can be used to monitor the simulation, for example, the observer which observes the demanded results and prints them on an output console such as a file.

The simulation is specified by a plain text configuration file which defines the implantation of the experiment such as the size of the network, the file path to load the topology, the initialisers and the controls. Figure 6.1 shows a configuration text file example for one of the experiments.

6.2 Datasets

Two main types of datasets were used in the experimental work, as the basis of the PeerSim simulator: artificially generated datasets and real-world datasets.

- **Artificial Topologies:** This kind of topologies was generated to enable the analysis of the proposed approaches with a variety of configurations, such as different size of topologies, different number of open facilities and different initial open facilities. Furthermore, the number of cycles to converge for both protocols is measured under different configurations.

  The used artificial topologies were Internet-like topologies in order to test the ability of implementing the proposed protocols in such environment. Section (6.2.1) explains the tool for the topologies generator.

```
# constant randomization seed, comment for a random seed
random.seed 1528227902201

# experiment parameters
SIZE 100000                    # network size
MINDELAY 50                    # minimum transmission delay
MAXDELAY 250          # maximum transmission delay
CYCLES 800                     # maximum number of cycles
CYCLELENGHT 1000.0             # pseudo cycle length in time unit
DIAM 9.0                       # network diameter

# declare network parameters
network
{
   size SIZE
   node peersim.core.GeneralNode
}

# declare simulation parameters
simulation
{
   endtime CYCLES*CYCLELENGHT
}

# declare protocols
protocol.lnk peersim.core.IdleProtocol                    # link protocol

protocol.urt peersim.transport.UniformRandomTransport     # transport protocol

{
   mindelay MINDELAY
   maxdelay MAXDELAY
}

protocol.tr peersim.transport.UnreliableTransport         # transport protocol
{
   transport urt
   drop 0.0
}

# P-median abstract protocol
protocol.pmp uk.ac.rdg.sse.anas.dist.pmedian.DistributedPmedianProtocolED
{
   step CYCLELENGHT
   linkable lnk
   transport tr
   netsize SIZE
   maxSafetyDelay (DIAM*MAXDELAY)/CYCLELENGHT
}

# initialisers
# network linking initialiser
init.wirb uk.ac.rdg.sse.anas.util.WireBriteFile
{
   protocol lnk
   filename ./topologies/brite/100K.brite \\ The path of the input graph

}

# p-median protocol initialiser
init.pmi uk.ac.rdg.sse.anas.util.PmedianInitializer
{
   protocol pmp
   m 100                       # number of facilities
   p (m * 0.25)     # close to open facilities ratio
}
```

Figure 6.1: A configuration example of one of the tests: It determines the size of tested topology, m and p as well as other required instructions to implement the trail.

- **Real-World Topologies:** Snapshots of real-world networks, from Stanford Large Network Dataset Collection **(SNAP)** [95] [96], are also tested with various configurations. In particular, the directed **P2P** Gnutella network datasets from SNAP with different sizes and connectivity quality are loaded into simulations. However, Gnutella networks forms, in some cases, have a number of disconnected communities [97]. Hence, the proposed protocols will not perform well due to the disconnection. In order to overcome this limitation of Gnutella typical snapshots, edges are added to the dataset to make it fully connected and ready for simulation.

6.2.1 Brite Topology

BRITE is an internet-like topology generator tool [98]. It was used to generate several different topologies. Brite tool can generate topologies using different connectivity models such as Barabasi-Albert model [99]. In which a topology is built with preference towards higher degree [98]. The created topologies were in different sizes (size range: 1000 - 500k nodes) that simulated the multi Autonomous System (AS) hierarchical networks.

6.3 Simulation and Experimental Results of the DPM Protocol

The obtained results from the simulations of the DPM protocol have been used for three main purposes:

1. Make certain that the DPM is valid by comparing its results with the obtained results from the centralised approach.

2. Investigation of the main purpose of this work, which is addresses the near-optimal locations for facilities in computer network.

3. Comparing with the KM approach results which addresses the near-optimal locations for facilities based on the k-medoids concept.

The subsections bellow describe, in details, each of the above points.

6.3.1 The DPM Validation

Because the DPM approach is based on the logic of the centralised p-median solution, which has been presented in chapter 3; collected results from the DPM approach are verified against the centralised approach to ensure that it is working correctly.

The verification process was inspected by going through tests of many topologies in both the centralised and distributed approaches, their output should become identical in order to confirm the validation of the proposed DPM approach.

The validation tests have been implemented with a different set of candidate facilities and also by varying the sizes of candidate facilities. To make the right comparison; the input of both approaches (centralised and distributed) were identical in both of their topology and their initial open facilities. Table (6.1) shows the test results are precisely the same.

6.3.2 Evaluation Metrics

The metrics that are used to evaluate the performance of the DPM and KM protocols are:

1. The cost of the solution.

Trail No.	m	p	Initial cost	Final cost	Number of swaps
1	60	25	2424	2180	13
2	60	25	2478	2175	12
3	70	25	2441	2208	14
4	70	25	2594	2207	18
5	100	25	2340	2028	19
6	100	25	2640	2089	19
7	100	20	2546	2180	16
8	100	20	2429	2057	17
9	100	30	2204	1909	23
10	100	30	2376	2043	21

Table 6.1: Cost and number of swaps to converge for the centralised and DPM protocol, 10 trails on different sets of facilities with m and p. It shows the exact output in all trails (N = 1000)

2. The space or area to search for candidate facilities to be opened.

3. The number of swaps between facilities to converge.

6.3.3 The Cost of the Solution

The cost of the solution, which is the main aim of the proposed algorithm, is affected by some factors such as the distribution of the candidate facilities, the size of space to search for the best locations for the facilities to open and the number of open facilities. This section demonstrate the collected results for different topologies and on different factors.

The DPM protocol has extensively been tested over a vast size of the network, and many trails have been implemented with different sets of facilities. As shown in Figure 6.2, on the same topology, the mean and standard deviation of the collected results from 10 trails have demonstrated an apparent reduction on the overall cost of the solution in each pair of facilities swap until convergence.

The DPM protocol is also tested on different sizes of topologies (100k - 200k) and also shown a successful addressing for the p number of facilities among the candidate facilities to open as shown in Figure (6.3)

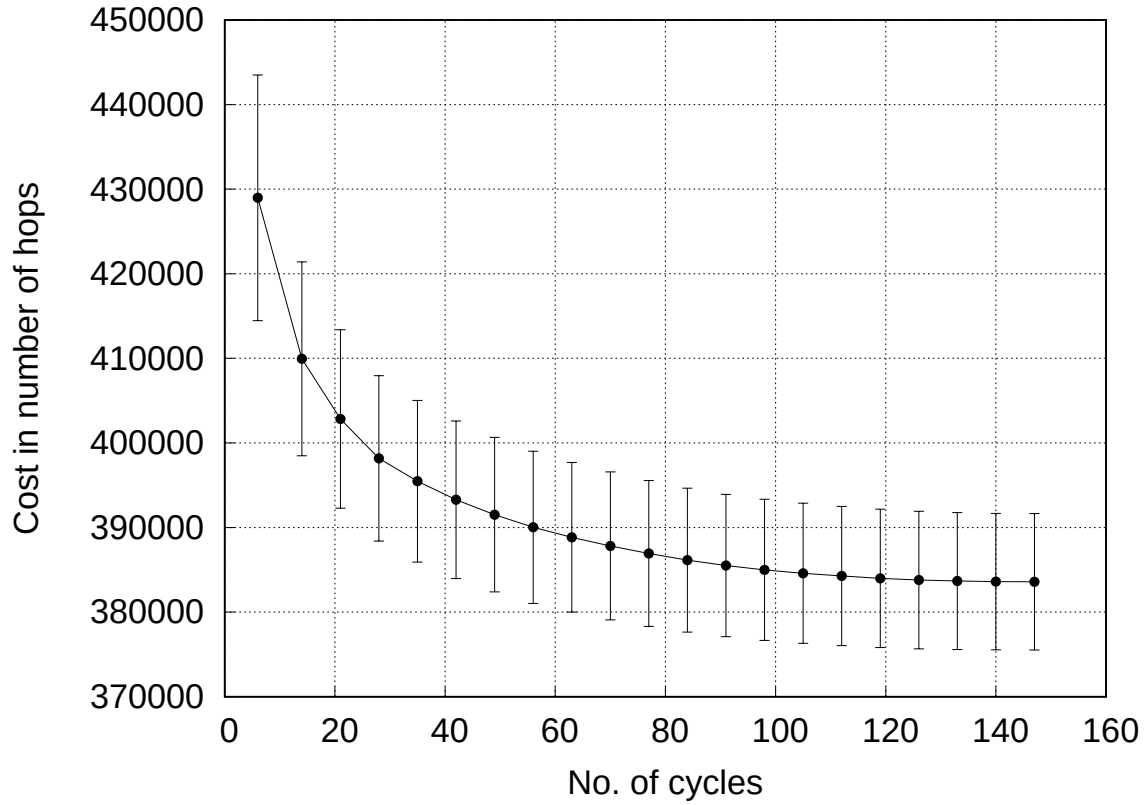


Figure 6.2: Cost of the solution (mean and standard deviation), 10 trails with different random seeds (N= 100K, m=100, p=25)

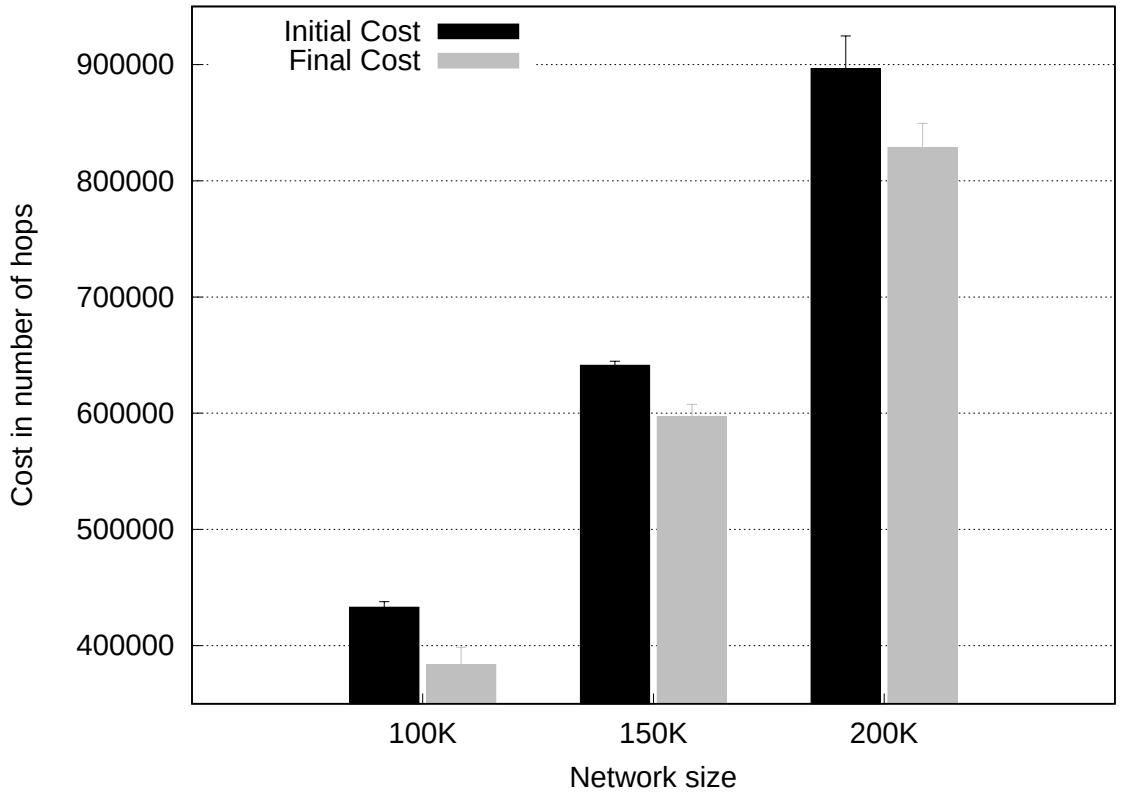


Figure 6.3: Initial and final cost of the solution (mean and standard deviation) for 10 trails with different sizes of topologies with different initial open facilities (m= 1% of N, p = 25% of m).

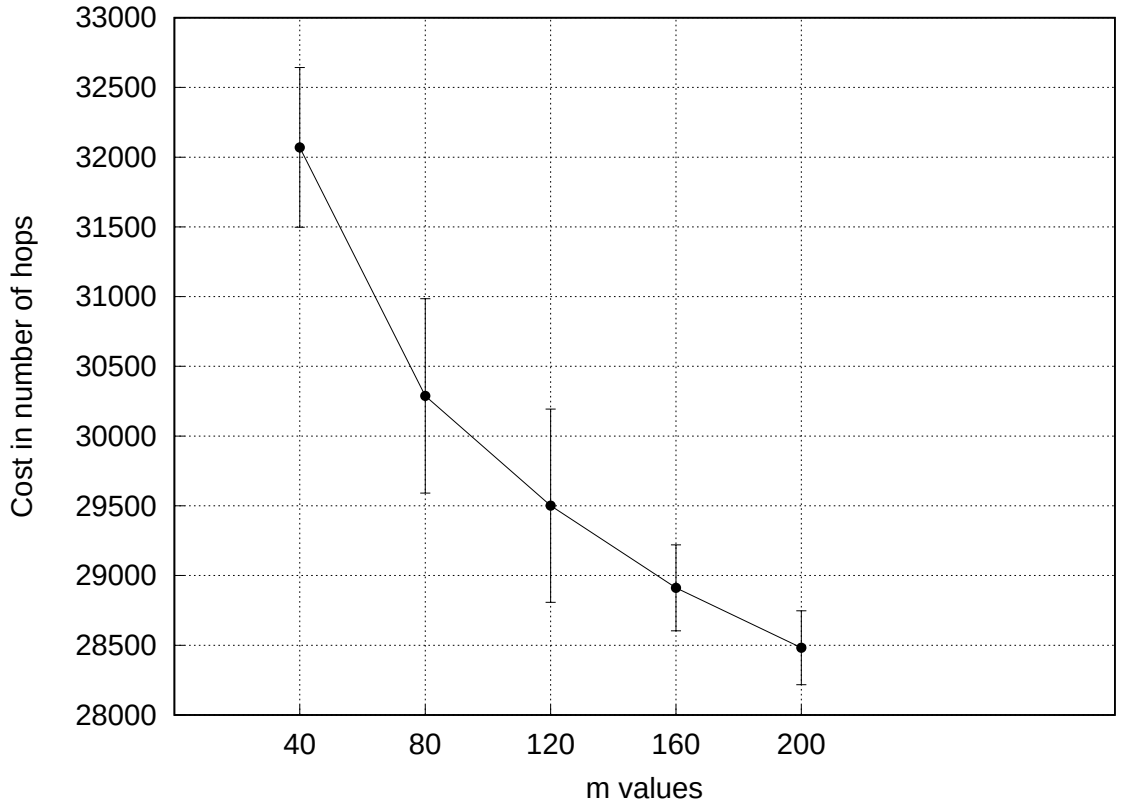


Figure 6.4: Cost of the solution (mean and standard deviation), 10 trails with various sizes of search spaces (N= 100K, p=25)

6.3.4 The Size of Space to Search

The second evaluation metric is the size of space to search which is the number of candidate facilities to open (m). As shown in Figure (6.4), the total cost of the solution has been decreased as the size of m is increased. However, this decrease in the total cost of the solution is offset by an increase in the time to converge. The next part of this section will explain the time to converge.

6.3.5 The Number of Swaps to Converge

As mentioned earlier in chapter (4), the DPM is converged in local optimal when there are no more swaps to reduce the cost of the solution. Not only the choice of the initial open facilities is affected by the number of swaps, but also the number of candidate locations to open.

With a different set of initial open facilities on a graph of size 100K; most of trails have shown different number of swaps (see table 6.2)

Moreover, what can be clearly seen in the table (6.2) is the difference between the number of swaps with the increase of available search space. For example, the maximum number of swaps on 10 trails was 11 when m= 40, while, on the same topology, this number is increased to 23 when the size of m =120.

Trail No.	Number of swaps		
	m= 40	m= 80	m= 120
1	6	15	23
2	7	16	20
3	9	19	19
4	11	15	20
5	7	13	15
6	11	21	18
7	11	19	19
8	6	18	20
9	8	16	21
10	9	18	20

Table 6.2: The number of swaps to converge on different sets of facilities with different size (m) and p, (N=100k, p=25)

6.4 KM protocol results

As explained in chapter 5, the proposed KM protocol is based on the logic of the classical k-medoid algorithm which is found similar to DPM protocol in term of clustering the network according to the shortest distance. However, the experimental results show that the DPM protocol tends to perform better than KM protocol on values of final costs of the solution. The subsection below will show the comparative analysis between both protocols.

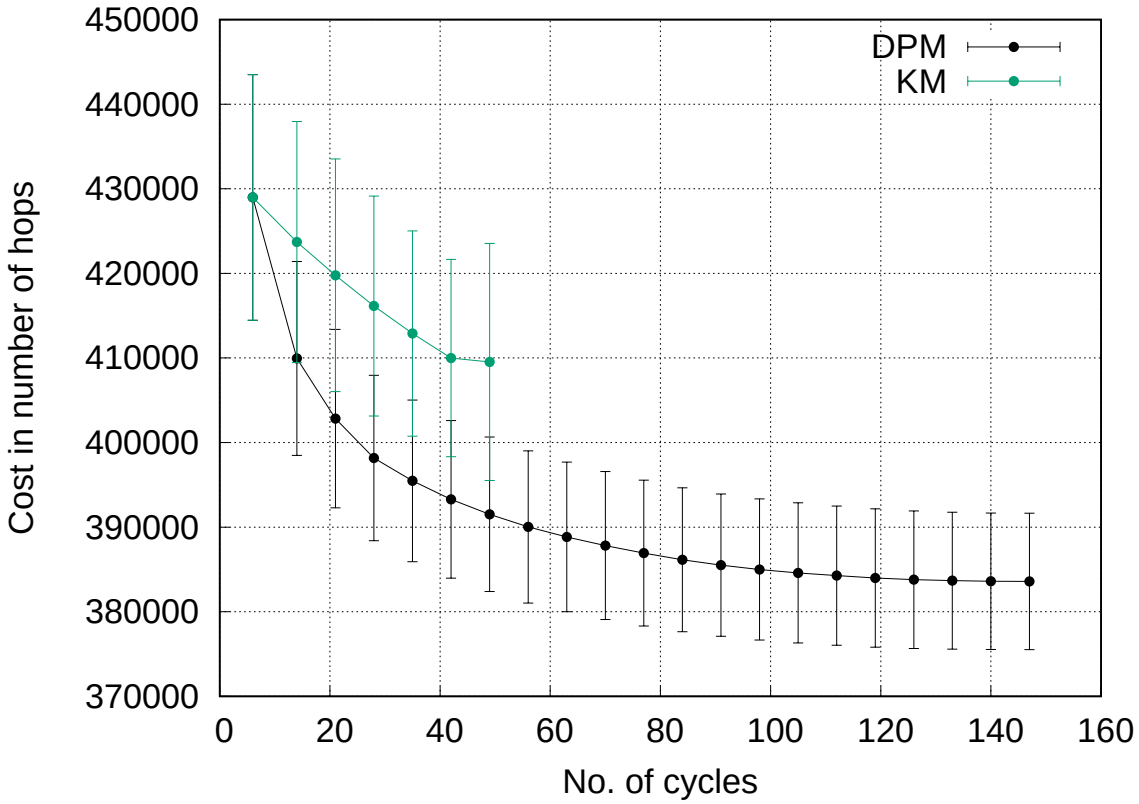


Figure 6.5: Cost of the solution (mean and standard deviation) for DPM and KM, 10 trails with (N= 100K, m=100, p=25)

6.4.1 Convergence Analysis for DPM and KM Protocols

The first important comparison between DPM and KM protocols is concerned with the achievement of the primary aim of the work; it addressed the near-optimal locations for facilities from the candidate locations. Both protocols have been evaluated through an extensive set of simulated topologies, each with different inputs and on different configurations.

As an example, in Figure 6.5 ten trails, with different sets of facilities, have been executed. In each test, the same initial p opened facilities are used for each protocol. The mean cost and the standard deviation for both protocols have demonstrated an apparent reduction on the overall cost of the solution in each swap until convergence. However, the KM protocol shows a higher cost of solution during all cycles of the protocol life.

The demonstrated results in Figure (6.5), means that the swap decision,

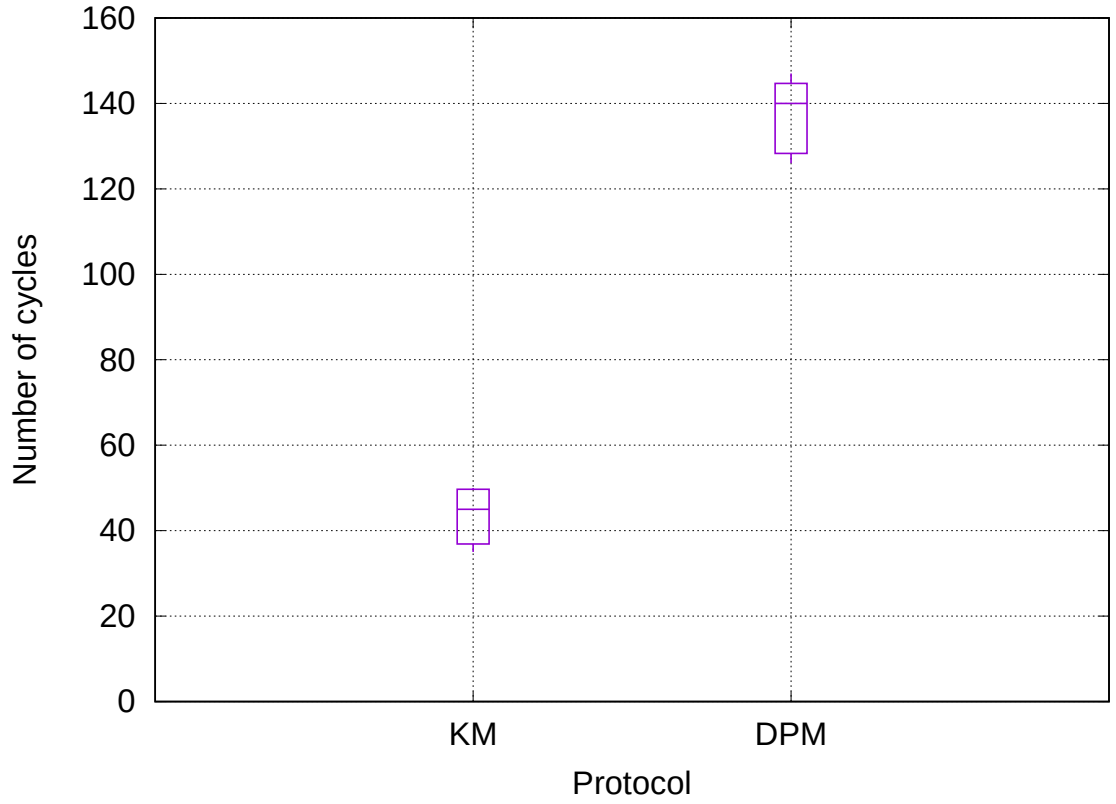


Figure 6.6: Convergence time in number of protocol cycles for DPM and KM, 10 trails with (N= 100K, m=100, p=25)

based on the global knowledge of the solution, has found better locations for facilities than taken decisions, based on local cluster information.

The second significant comparison between the DPM and KM protocols is the time to converge. Ten trails on a topology of size 100k, as shown in Figure (6.6), the KM protocol has converged before the DPM protocol. However, this means that KM cannot continue to find better locations for the candidate facilities and it falls into a local optimum before the DPM.

As indicated previously, the DPM and KM protocols have also been tested on different sizes of graphs. As shown in Figure (6.7), the mean and standard deviation of final costs for ten trails in each graph is computed. All the trails have indicated a distinguishable progress of the DPM protocol from the KM protocol concerning the final solution cost.

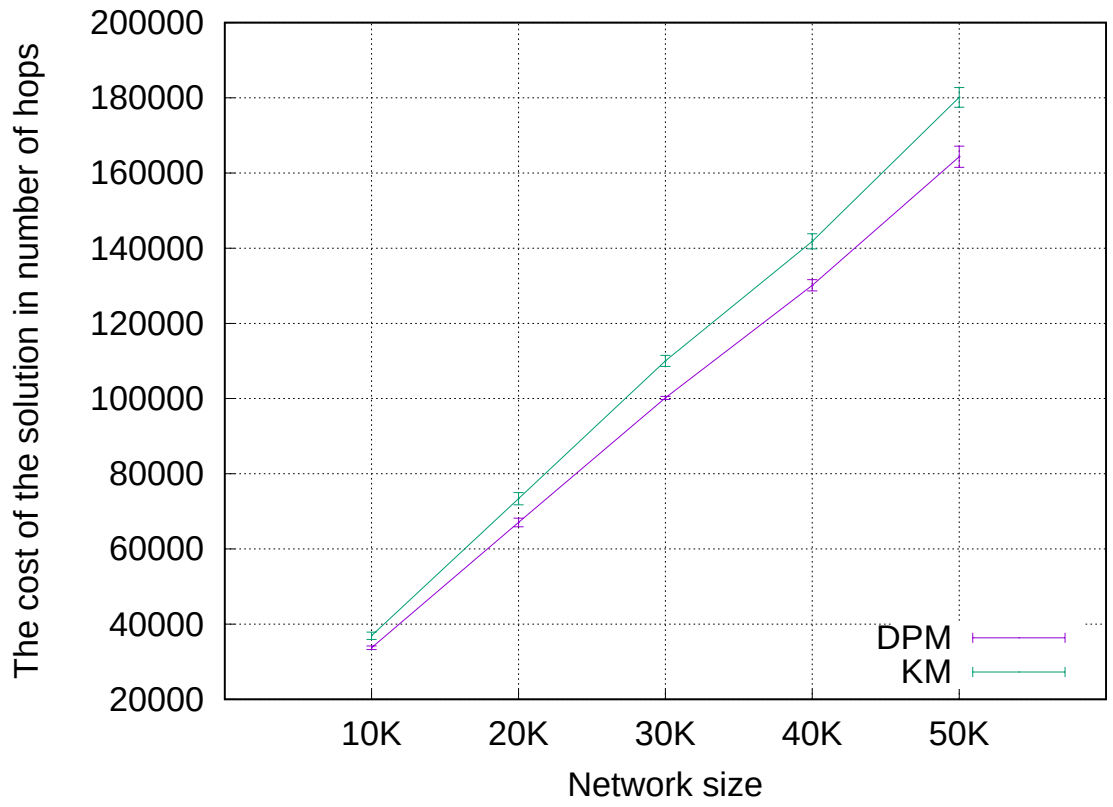


Figure 6.7: Convergence time in number of protocol cycles for DPM and KM, 10 trails with (N= 100K, m=100, p=25)

The protocols are also tested on different ranges of search spaces (see Figure 6.8). The KM protocol shows a higher cost than the DPM, despite a wider range of choices to search for the best locations of facilities are given.

Figure 6.8 has also shown that the cost of the solution is decreased as much as the size of candidate m facilities is increased. However, the KM solution also falls in local optima and can not find a better solution as the DPM.

6.5 Analysis of the Messages Number in both DPM and KM Protocols

As explained in chapters 4 and 5, the DPM and KM approaches depend on the messages (events) for finding the p-median locations. The number of messages are analysed against the type of messages, shown in Figure 6.9, the significant number of messages among nodes during the running cycles is the

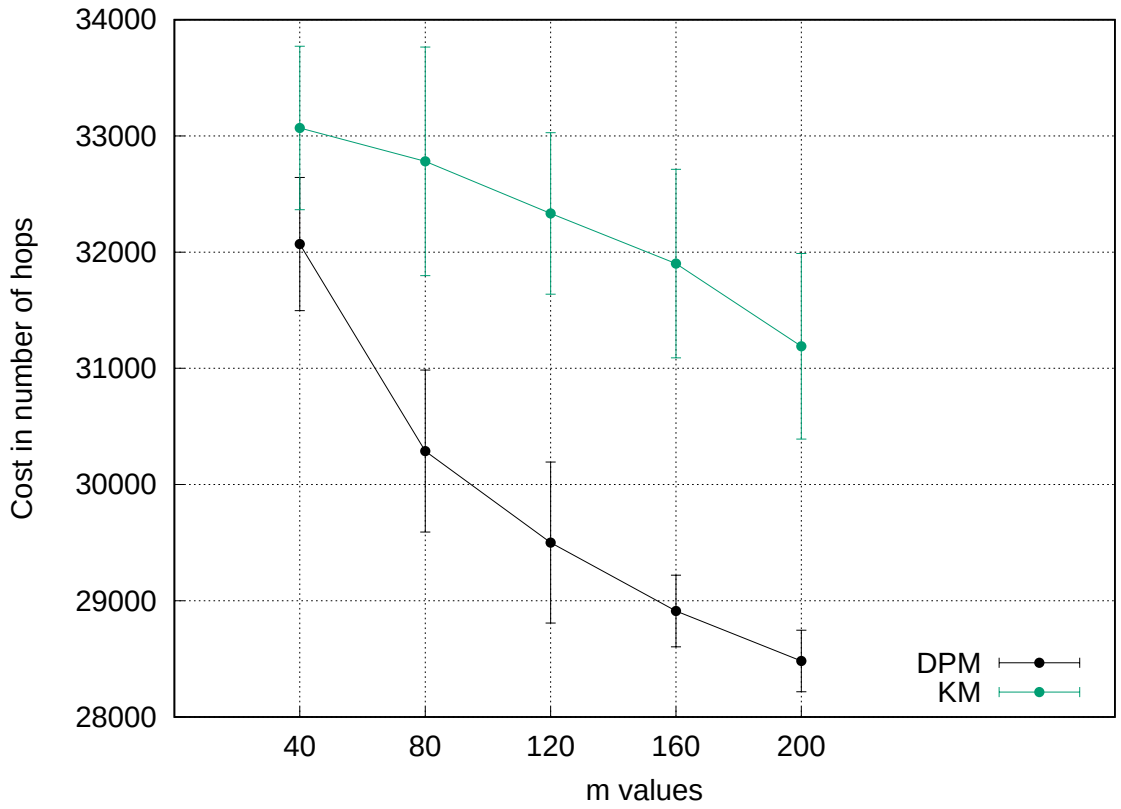
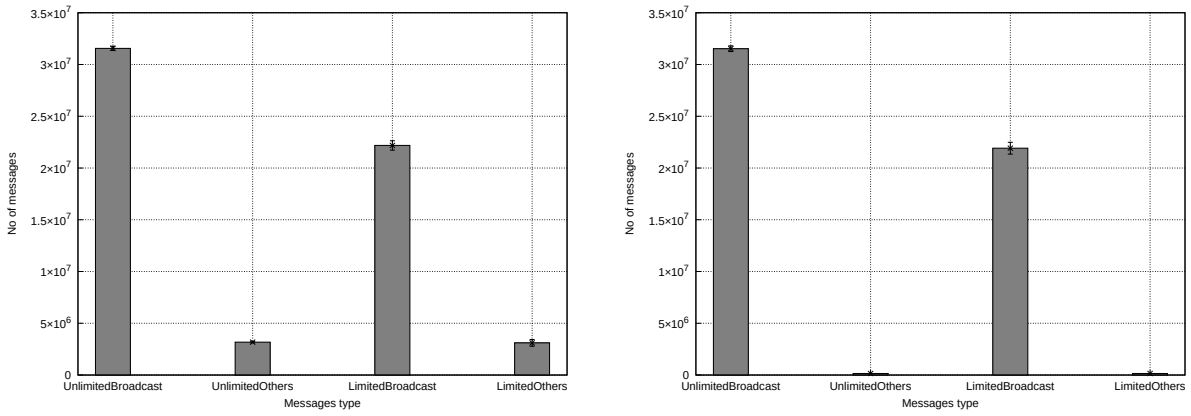


Figure 6.8: Convergence time in number of protocol cycles for DPM and KM, 10 trails with (N= 100K, m=100, p=25)

facility broadcast messages. This is due to the travel from all facilities to all user nodes in the solution.

Practically, each facility serves nodes in its cluster, the trails have shown that it is unnecessary to forward the broadcasting messages to the whole network due to the waste of time and load increase on the network. Instead; the number of broadcast messages is restricted. The restriction is implemented by sending the broadcast messages to a D diameter of hopes around the facility. The implementation trails have shown the same results in the final cost of the solution with a distinct reduction for the elapsed time of the implementation.



a. DPM protocol b. KM protocol

Figure 6.9: Comparison of broadcast messages and other types of messages number using maximum hops number of messages transmit, without limitation for messages in both protocols, (N= 50K, m=100, p=25)

6.6 Summary

This chapter presents the experimental work that is aimed to validate and evaluate the proposed DPM approach to solve the p-median problem in a distributed environment. It also successfully presents the implementation of the proposed KM approach for addressing the facility locations on a network based on the k-medoid clustering algorithm.

The experimental work is demonstrated on different types of topologies and different sizes. Moreover, the results on the same topology with different numbers of available facilities are also presented on both proposed protocols (DPM and KM). Comparisons between the DPM and KM protocols are also presented under different configurations.

The last section (6.5) has presented an analysis study about the type of messages travelling in the network during the protocol run. Moreover, a solution is suggested by setting a maximum number of hops for a message to transfer among hops to reduce the number of messages.

Chapter 7

Conclusions and Future Works

This chapter briefly describes the journey of this work. A summary of the main contributions and how the results have met the objectives is presented. It also discusses some conclusions and future directions.

7.1 Synopsis

Due to the enormous increase of computer network size, it has become infeasible to solve problems about facility location-allocation in a centralised approach, since it requires a long time to collect the data and a large memory, as well as the communication overhead. This work has proposed two paradigms for solving the facility location-allocation problem in a distributed environment using the p-median problem concept.

The lack of the central data and the coordinator in a distributed environment are the main challenges that were faced. To overcome the central data problem, the approaches which built a distributed global view to find the near-optimal locations for facilities in a network, without collecting the data of the network in a central server or node, were proposed. The coordination problem was also solved by making the proposed approaches in three main phases and the nodes were synchronised at the end of each phase to assure

the correct decision to choose facilities' locations.

The proposed approaches clustered the network according to the shortest distances between the users and the facilities. However, their strategies to address the near-optimal locations for the facilities were different.

The first approach (as described in Chapter 4) built the global information about the network in order to address the best locations for facilities among all the available candidate locations in the system (determined only one better location for a facility in each iteration).

While the second proposed approach (as described in Chapter 5) built a local information about each cluster in order to address the best facility to be opened in each cluster. All the clusters were able to update their open facilities simultaneously.

Using a simulation-based experiment, it was shown that the first proposed approach (DPM) could address more accurate locations which led to a lower cost of a solution than the second approach (KM). However, it required more cycles to reach convergence.

7.2 Conclusion

This thesis has looked at the p-median problem as a means of locating facilities in large distributed networks. The findings of this study make three noteworthy contributions to the current literature:

1. The first contribution, which is described in Chapter 3, is the execution and analysis of the centralised p-median problem to identify the factors that affect the performance of the p-median problem. The obtained results are used in the validation of a novel proposed distributed p-median approach.

2. The second contribution, which met the second objective of Section 1.4. The proposed distributed approaches are implemented without predefining of the network topology. Moreover, the data of network are remained intrinsically distributed in the nodes without collecting them in a central node or server for manipulation. These are described in, in details, Chapters 4 and 5.

3. The third contribution, which is described in Chapter 5 has met the third and fourth objectives of Section 1.4, designed to cluster the network to a number of clusters according to a concept of the k-medoid clustering approach, then within each cluster, in parallel, the median location for a facility is determined.

The last objective of Section 1.4 is met by Chapter 6 which provides an comparative analysis of the proposed approaches against different parameters. It confirms that the decision for locating a facility in the system based on the information of the whole network is more accurate in terms of cost than the strategy of clustering a network then placing a facility in each cluster base on the local information of the cluster. However, the latter one takes less iteration to reach convergence.

7.3 Key Findings

To evaluate each of the proposed approaches, both were tested against various numbers of parameters that effect on their performance. Such as testing the implementation against different number of open facilities, various number of candidate facilities to be opened, different initial data points and different sizes of topologies.

The comparative analysis of DPM and KM has shown a significant cost reduction in the solution using both approaches. However, the results of the DPM are more accurate than the KM which shows higher costs for results in different configurations.

Evaluation carried out on both protocols have shown their ability to address the near-optimal locations for facilities in a computer network. Over many simulations analysis have revealed the following.

- Addressing locations for facilities based on a global view of the network (as in DPM) lead to more accurate optimisation results than clustering the network and optimising a facility location based on the local view of each cluster separately (as in KM solution). However, the former solution takes more cycles to converge.

- When more candidate facilities are available, the protocols can find better solutions in terms of cost. However, as the bigger as the size of available candidate facilities; the more cycles are required to converge.

7.4 Future Works

This section discusses the future work that could lead to valuable contributions to this research.

As aforementioned, the number of the candidate locations to be facilities are affected by the final cost of the solution; however, the effect of the number of the candidate locations is concentrated on the number of cycles to converge; since the increasing of search space will increase the required time to converge. Therefore, it would be interesting to afford more information on the scope of the search which would help us to establish a greater degree of accuracy of this item.

The distributed p-median solutions presented in this work are implemented on static networks. Due to the dynamism feature of the Internet; further studies regarding the dynamic conditions of the network would be worthwhile.

One of the inputs to the p-median problem is the number of the candidate facilities to be opened p. As it has been discussed earlier in this thesis, it is easy to determine the p for network applications. However, some network applications concentrate on the cost of the solution regardless of p open facilities. Further research on p is suggested which would help to establish a greater degree of accuracy on this matter.

Due to the asynchronous behaviour of the network nodes and the decentralised implementation of the proposed protocols (DPM and KM), as described earlier (Chapters 4 and 5); they are synchronised at the end of each phase to make certain that the correct decision to open a facility is taken. However, this means extra time is required. The issue of synchronisation is an intriguing one which could be usefully explored in future research.

Further research might explore the p-centre problem in which the goal is to reduce the maximum response time among the facilities and the user nodes [100] [101].

References

[1] I. W. Stats, "World internet users and 2017 population stats," 2017.

[2] K. Carling, M. Han, J. Håkansson, and P. Rebreyend, "Distance measure and the p p-median problem in rural areas," *Annals of Operations Research*, vol. 226, no. 1, pp. 89–99, 2015.

[3] O. Alp, E. Erkut, and Z. Drezner, "An efficient genetic algorithm for the p-median problem," *Annals of Operations research*, vol. 122, no. 1-4, pp. 21–42, 2003.

[4] F. García-López, B. Melián-Batista, J. A. Moreno-Pérez, and J. M. Moreno-Vega, "The parallel variable neighborhood search for the p-median problem," *Journal of Heuristics*, vol. 8, no. 3, pp. 375–388, 2002.

[5] K. Jain and V. V. Vazirani, "Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation," *Journal of the ACM (JACM)*, vol. 48, no. 2, pp. 274–296, 2001.

[6] A. Al-khedhairi, "Simulated annealing metaheuristic for solving p-median problem," *Int. J. Contemp. Math. Sciences*, vol. 3, no. 28, pp. 1357–1365, 2008.

[7] Z. Drezner and S. Salhi, "Incorporating neighborhood reduction for the solution of the planar p-median problem," *Annals of Operations Research*, vol. 258, no. 2, pp. 639–654, 2017.

[8] H. Mashayekhi, J. Habibi, T. Khalafbeigi, S. Voulgaris, and M. Van Steen, "Gdcluster: a general decentralized clustering algorithm," *IEEE transactions on knowledge and data engineering*, vol. 27, no. 7, pp. 1892–1905, 2015.

[9] J. Reese, "Solution methods for the p-median problem: An annotated bibliography," *Networks*, vol. 48, no. 3, pp. 125–142, 2006.

[10] C. C. Fast, "Novel techniques for the zero-forcing and p-median graph location problems," Tech. Rep., 2017.

[11] V. Marianov and D. Serra, "Median problems in networks," in *Foundations of location analysis*. Springer, 2011, pp. 39–59.

[12] M. G. Resende and R. F. Werneck, "A hybrid heuristic for the p-median problem," *Journal of heuristics*, vol. 10, no. 1, pp. 59–88, 2004.

[13] N. Mladenović, J. Brimberg, P. Hansen, and J. A. Moreno-Pérez, "The p-median problem: A survey of metaheuristic approaches," *European Journal of Operational Research*, vol. 179, no. 3, pp. 927–939, 2007.

[14] O. Kariv and S. L. Hakimi, "An algorithmic approach to network location problems. i: The p-centers," *SIAM Journal on Applied Mathematics*, vol. 37, no. 3, pp. 513–538, 1979.

[15] P. Avella, A. Sassano, and I. Vasil'ev, "Computational study of large-scale p-median problems," *Mathematical Programming*, vol. 109, no. 1, pp. 89–114, 2007.

[16] M. G. Resende and R. F. Werneck, "On the implementation of a swap-based local search procedure for the p-median problem," in *Proceedings of the Fifth Workshop on Algorithm Engineering and Experiments (ALENEX03)*, 2003, pp. 119–127.

[17] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," *SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 4, pp. 251–262, Aug. 1999. [Online]. Available: http://doi.acm.org/10.1145/316194.316229

[18] R. Albert, H. Jeong, and A.-L. Barabási, "Internet: Diameter of the world-wide web," *nature*, vol. 401, no. 6749, p. 130, 1999.

[19] A. Liotta, C. Ragusa, and G. Pavlou, "Near-optimal service facility location in dynamic communication networks," *IEEE communications letters*, vol. 9, no. 9, pp. 862–864, 2005.

[20] R. Guimerà, A. Díaz-Guilera, F. Vega-Redondo, A. Cabrales, and A. Arenas, "Optimal network topologies for local search with congestion," *Physical review letters*, vol. 89, no. 24, p. 248701, 2002.

[21] T.-W. Kuo and M.-J. Tsai, "On the construction of data aggregation tree with minimum energy cost in wireless sensor networks: Np-completeness and approximation algorithms," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 2591–2595.

[22] P. Hansen and B. Jaumard, "Cluster analysis and mathematical programming," *Mathematical programming*, vol. 79, no. 1-3, pp. 191–215, 1997.

[23] P. Avella and A. Sassano, "On the p-median polytope," *Mathematical Programming*, vol. 89, no. 3, pp. 395–411, 2001.

[24] M. L. Brandeau and S. S. Chiu, “An overview of representative problems in location research,” *Management science*, vol. 35, no. 6, pp. 645–674, 1989.

[25] Y. Konforty and A. Tamir, “The single facility location problem with minimum distance constraints,” *Location Science*, vol. 5, no. 3, pp. 147–163, 1997.

[26] Z. Drezner and H. W. Hamacher, *Facility location: applications and theory*. Springer Science & Business Media, 2001.

[27] S. L. Hakimi, “Optimum locations of switching centers and the absolute centers and medians of a graph,” *Operations research*, vol. 12, no. 3, pp. 450–459, 1964.

[28] M. B. Teitz and P. Bart, “Heuristic methods for estimating the generalized vertex median of a weighted graph,” *Operations research*, vol. 16, no. 5, pp. 955–961, 1968.

[29] S. H. Owen and M. S. Daskin, “Strategic facility location: A review,” *European journal of operational research*, vol. 111, no. 3, pp. 423–447, 1998.

[30] M. Jamshidi, “Median location problem,” in *Facility Location*. Springer, 2009, pp. 177–191.

[31] Z. Azarmand and E. Neishabouri, “Location allocation problem,” in *Facility location*. Springer, 2009, pp. 93–109.

[32] S. Elloumi, “A tighter formulation of the p-median problem,” *Journal of combinatorial optimization*, vol. 19, no. 1, pp. 69–83, 2010.

[33] M. Charikar, S. Guha, É. Tardos, and D. B. Shmoys, "A constant-factor approximation algorithm for the k-median problem," *Journal of Computer and System Sciences*, vol. 65, no. 1, pp. 129–149, 2002.

[34] M. Dzator, J. Dzator *et al.*, "An efficient modified greedy algorithm for the p-median problem," 2015.

[35] G. Laporte, S. Nickel, and F. S. da Gama, *Location science*. Springer, 2015, vol. 528.

[36] R. Z. Farahani and M. Hekmatfar, *Facility location: concepts, models, algorithms and case studies*. Springer, 2009.

[37] M. Sevkli, R. Mamedsaidov, and F. Camci, "A novel discrete particle swarm optimization for p-median problem," *Journal of King Saud University-Engineering Sciences*, vol. 26, no. 1, pp. 11–19, 2014.

[38] M. Hekmatfar and M. Pishvaee, "Hub location problem," in *Facility Location*. Springer, 2009, pp. 243–270.

[39] E. Rolland, D. A. Schilling, and J. R. Current, "An efficient tabu search procedure for the p-median problem," *European Journal of Operational Research*, vol. 96, no. 2, pp. 329–342, 1997.

[40] J. Skorin-Kapov, "Extensions of a tabu search adaptation to the quadratic assignment problem," *Computers & Operations Research*, vol. 21, no. 8, pp. 855–865, 1994.

[41] E. L. Senne and L. A. Lorena, "Lagrangean/surrogate heuristics for p-median problems," in *Computing Tools for Modeling, Optimization and Simulation*. Springer, 2000, pp. 115–130.

[42] P. Krishnan, D. Raz, and Y. Shavitt, "The cache location problem," *IEEE/ACM Transactions on Networking (TON)*, vol. 8, no. 5, pp. 568–582, 2000.

[43] M. P. Scaparra and M. G. Scutellà, "Facilities, locations, customers: Building blocks of location models. a survey." 2001.

[44] P. Hansen, N. Mladenović, and J. A. M. Pérez, "Variable neighbourhood search: methods and applications," *Annals of Operations Research*, vol. 175, no. 1, pp. 367–407, 2010.

[45] R. Zurita-Milla and O. Huisman, "Md. shamsul arifin february, 2011," 2011.

[46] F. Maranzana, "On the location of supply points to minimize transport costs," *Journal of the Operational Research Society*, vol. 15, no. 3, pp. 261–270, 1964.

[47] L. Caccetta, M. Dzator *et al.*, "Heuristic methods for locating emergency facilities," in *Proceedings of the 16th International Congress on Modelling and Simulation, Melbourne, Australia*. Citeseer, 2005, pp. 1744–1750.

[48] G. Lim, J. Reese, and A. Holder, "Fast and robust techniques for the euclidean p-median problem with uniform weights," *Computers & industrial engineering*, vol. 57, no. 3, pp. 896–905, 2009.

[49] C. Ragusa, A. Liotta, and G. Pavlou, "An adaptive clustering approach for the management of dynamic systems," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 12, pp. 2223–2235, 2005.

[50] C. S. ReVelle and H. A. Eiselt, "Location analysis: A synthesis and survey," *European journal of operational research*, vol. 165, no. 1, pp. 1–19, 2005.

[51] M. S. Daskin and K. L. Maass, "The p-median problem," in *Location science*. Springer, 2015, pp. 21–45.

[52] M. N. Neema and A. Ohgai, "Multi-objective location modeling of urban parks and open spaces: Continuous optimization," *Computers, Environment and Urban Systems*, vol. 34, no. 5, pp. 359–376, 2010.

[53] M. Neema, K. Maniruzzaman, and A. Ohgai, "New genetic algorithms based approaches to continuous p-median problem," *Networks and Spatial Economics*, vol. 11, no. 1, pp. 83–99, 2011.

[54] S. Salhi and M. Gamal, "A genetic algorithm based approach for the uncapacitated continuous location–allocation problem," *Annals of Operations Research*, vol. 123, no. 1-4, pp. 203–222, 2003.

[55] E. Suomalainen, "Multicriteria analysis and visualization of location-allocation problems," *Master's thesis, Department of Engineering Physics and Mathematics, University of Helsinki*, 2006.

[56] M. S. Daskin, *Network and discrete location: models, algorithms, and applications*. John Wiley & Sons, 2011.

[57] T. Uno, S. Hanaoka, and M. Sakawa, "An application of genetic algorithm for multi-dimensional competitive facility location problem," in *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, vol. 4. IEEE, 2005, pp. 3276–3280.

[58] A. L. Medaglia, J. G. Villegas, and D. M. Rodríguez-Coca, "Hybrid biobjective evolutionary algorithms for the design of a hospital waste

management network," *Journal of Heuristics*, vol. 15, no. 2, p. 153, 2009.

[59] J. Yang, J. Xiong, S. Liu, and C. Yang, "Flow capturing location-allocation problem with stochastic demand under hurwicz rule," in *Natural Computation, 2008. ICNC'08. Fourth International Conference on*, vol. 7. IEEE, 2008, pp. 169–173.

[60] D. Gong, M. Gen, G. Yamazaki, and W. Xu, "Hybrid evolutionary method for capacitated location-allocation problem," *Computers & industrial engineering*, vol. 33, no. 3-4, pp. 577–580, 1997.

[61] X. Li, Z. Liu, and X. Zhang, "Applying genetic algorithm and hilbert curve to capacitated location allocation of facilities," in *Artificial Intelligence and Computational Intelligence, 2009. AICI'09. International Conference on*, vol. 1. IEEE, 2009, pp. 378–383.

[62] A. T. Murray, "Advances in location modeling: Gis linkages and contributions," *Journal of geographical systems*, vol. 12, no. 3, pp. 335–354, 2010.

[63] A. N. Sadigh and H. Fallah, "Demand point aggregation analaysis for location models," in *Facility Location*. Springer, 2009, pp. 523–534.

[64] S. L. Straitiff and R. G. Cromley, "Using gis and k= 3 central place lattices for efficient solutions to the location set-covering problem in a bounded plane," *Transactions in GIS*, vol. 14, no. 3, pp. 331–349, 2010.

[65] D. Gong, M. Gen, W. Xu, and G. Yamazaki, "Hybrid evolutionary method for obstacle location-allocation," *Computers & Industrial Engineering*, vol. 29, no. 1-4, pp. 525–530, 1995.

[66] E. S. Correa, M. T. A. Steiner, A. A. Freitas, and C. Carnieri, "A genetic algorithm for solving a capacitated p-median problem," *Numerical Algorithms*, vol. 35, no. 2-4, pp. 373–388, 2004.

[67] É. D. Taillard, "Heuristic methods for large centroid clustering problems," *Journal of Heuristics*, vol. 9, no. 1, pp. 51–73, 2003.

[68] L. Cooper, "Heuristic methods for location-allocation problems," *SIAM review*, vol. 6, no. 1, pp. 37–53, 1964.

[69] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM computing surveys (CSUR)*, vol. 35, no. 3, pp. 268–308, 2003.

[70] L. Bianchi, M. Dorigo, L. M. Gambardella, and W. J. Gutjahr, "A survey on metaheuristics for stochastic combinatorial optimization," vol. 8, no. 2, pp. 239–287, Jun. 2009. [Online]. Available: http://dx.doi.org/10.1007/s11047-008-9098-4

[71] F. Glover and M. Laguna, *Tabu search. In Modern heuristic techniques for combinatorial problems*. John Wiley & Sons, Inc., 1993.

[72] J. Alcaraz, M. Landete, and J. F. Monge, "Design and analysis of hybrid metaheuristics for the reliability p-median problem," *European Journal of Operational Research*, vol. 222, no. 1, pp. 54–64, 2012.

[73] D. Aksen, N. Aras, and N. Piyade, "A bilevel p-median model for the planning and protection of critical facilities," *Journal of Heuristics*, vol. 19, no. 2, pp. 373–398, 2013.

[74] Z. Drezner, J. Brimberg, N. Mladenović, and S. Salhi, "New heuristic algorithms for solving the planar p-median problem," *Computers & Operations Research*, vol. 62, pp. 296–304, 2015.

[75] A. Rahmani and S. MirHassani, "A hybrid firefly-genetic algorithm for the capacitated facility location problem," *Information Sciences*, vol. 283, pp. 70–78, 2014.

[76] M. T. Melo, S. Nickel, and F. Saldanha-Da-Gama, "Facility location and supply chain management–a review," *European journal of operational research*, vol. 196, no. 2, pp. 401–412, 2009.

[77] M. G. Resende and R. F. Werneck, "A fast swap-based local search procedure for location problems," *Annals of Operations Research*, vol. 150, no. 1, pp. 205–230, 2007.

[78] T. S. Hale and C. R. Moberg, "Location science research: a review," *Annals of operations research*, vol. 123, no. 1-4, pp. 21–35, 2003.

[79] S. Dantrakul, C. Likasiri, and R. Pongvuthithum, "Applied p-median and p-center algorithms for facility location problems," *Expert Systems with Applications*, vol. 41, no. 8, pp. 3596–3604, 2014.

[80] P. Berkhin, "A survey of clustering data mining techniques," in *Grouping multidimensional data*. Springer, 2006, pp. 25–71.

[81] R. Whitaker, "A fast algorithm for the greedy interchange for large-scale clustering and median location problems," *INFOR: Information Systems and Operational Research*, vol. 21, no. 2, pp. 95–108, 1983.

[82] P. Hansen and N. Mladenović, "Variable neighborhood search for the p-median," *Location Science*, vol. 5, no. 4, pp. 207–226, 1997.

[83] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[84] Y. Deng, Y. Chen, Y. Zhang, and S. Mahadevan, "Fuzzy dijkstra algorithm for shortest path problem under uncertain environment," *Applied Soft Computing*, vol. 12, no. 3, pp. 1231–1237, 2012.

[85] A. Medina, I. Matta, and J. Byers, "Brite: a flexible generator of internet topologies," 2000.

[86] A. Montresor and M. Jelasity, "Peersim: A scalable p2p simulator," in *Peer-to-Peer Computing, 2009. P2P'09. IEEE Ninth International Conference on*. IEEE, 2009, pp. 99–100.

[87] J. M. Bahi, S. Contassot-Vivier, and R. Couturier, "An efficient and robust decentralized algorithm for detecting the global convergence in asynchronous iterative algorithms," in *International Conference on High Performance Computing for Computational Science*. Springer, 2008, pp. 240–254.

[88] H. Song, J.-G. Lee, and W.-S. Han, "Pamae: Parallel k-medoids clustering with high accuracy and efficiency," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 1087–1096.

[89] C. C. Aggarwal and C. K. Reddy, *Data clustering: algorithms and applications*. CRC press, 2013.

[90] C. Sammut and G. I. Webb, *Encyclopedia of machine learning*. Springer Science & Business Media, 2011.

[91] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for k-medoids clustering," *Expert systems with applications*, vol. 36, no. 2, pp. 3336–3341, 2009.

[92] S. M. R. Zadegan, M. Mirzaie, and F. Sadoughi, "Ranked k-medoids: A fast and accurate rank-based partitioning algorithm for clustering large datasets," *Knowledge-Based Systems*, vol. 39, pp. 133–143, 2013.

[93] T. Velmurugan and T. Santhanam, "Computational complexity between k-means and k-medoids clustering algorithms for normal and uniform distributions of data points," *Journal of computer science*, vol. 6, no. 3, p. 363, 2010.

[94] I. Kazmi and S. F. Y. Bukhari, "Peersim: An efficient & scalable testbed for heterogeneous cluster-based p2p network protocols," in *Computer Modelling and Simulation (UKSim), 2011 UkSim 13th International Conference on*. IEEE, 2011, pp. 420–425.

[95] J. Leskovec, A. Krevl, and S. Datasets, "Stanford large network dataset collection, 2014," *URL: http://snap. stanford. edu/data/index. html*, 2014.

[96] J. Leskovec and R. Sosič, "Snap: A general-purpose network analysis and graph-mining library," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, no. 1, p. 1, 2016.

[97] S. Saroiu, K. P. Gummadi, and S. D. Gribble, "Measuring and analyzing the characteristics of napster and gnutella hosts," *Multimedia systems*, vol. 9, no. 2, pp. 170–184, 2003.

[98] A. Medina, A. Lakhina, I. Matta, and J. Byers, "Brite: Universal topology generation from a user's perspective," Boston University Computer Science Department, Tech. Rep., 2001.

[99] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[100] A. Suzuki and Z. Drezner, "The p-center location problem in an area," *Location science*, vol. 4, no. 1-2, pp. 69–82, 1996.

[101] B. C. Tansel, R. L. Francis, and T. J. Lowe, "State of the artlocation on networks: a survey. part i: the p-center and p-median problems," *Management science*, vol. 29, no. 4, pp. 482–497, 1983.