PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
ABDELHAMID IBN BADIS UNIVERSITY OF MOSTAGANEM
FACULTY OF EXACT SCIENCES AND COMPUTER SCIENCE
DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

PhD. THESIS

Presented to obtain

The LMD PhD. Degree in Computer Science

Specialty: Distributed Computing and Networks

By:

Mohammedi Taieb Sabir

Use of the Multi-criteria Approach to Elect a Facilitator Within a Group Decision Support System

On September 15th 2025 in front of the committee composed of

President: Sehaba Karim	Research Professor	UMAB, Mostaganem
Reviewers:		
Dr. ROUBA Baroudi	Senior Lecturer	UMAB,Mostaganem
Dr BESSAOUD Karim	Senior Lecturer	UMAB, Mostaganem
Dr. MEROUFEL Bakhta	Senior Lecturer	UDL, Sidi Bel Abbes
Supervisor: Dr. Laredj Mohamed Adnane	Senior Lecturer	UMAB,Mostaganem

Academic Year: 2024 – 2025

Acknowledgement

First of all, I thank God for guiding me and making me succeed in finishing my thesis research.

I would also like to thank my very caring mother who is always kind to me and does all she can to support me and make me happy.

I also thank my father for supporting me and encouraging me to pursue my PhD since day one.

My thanks also go to my sisters and brother for their support and encouragement to keep on moving forward with my PhD research.

I want to thank my thesis supervisor Mr. Laredj Mohamed Adnane for his assistance, generosity, counsel, and constructive criticism throughout my PhD research; for giving me the information I needed; and for allowing me to continue my research in this area.

Figures List

Tables List

Contents

Publications and Communications

International Publications

- Taieb, S. M., & Laredj, M. A. (2024). Proposition of a new tool for the election of group decision support system facilitator based on the multi-criteria approach. STUDIES IN ENGINEERING AND EXACT SCIENCES, 5(2), e8850. https://doi.org/10.54021/seesv5n2-303
- S. Mohammedi Taieb and M. A. Laredj, "GFEA: Leader Election Algorithm for Choosing a GroupDecision Support System Facilitator", *Sci. Eng. Technol.*, vol. 5, no. 2, May 2025, doi: 10.54327/set2025/v5.i2.216.

International and National Conferences

- Comparison of MCDA Methods in the Coordinator Election Within Collaborative E-maintenance. Workshop "Preliminary Research Work in Computer Science" in 7th International Symposium on Modelling and Implementation of Complex Systems – MISC 2022. October 31st 2022. Mostaganem University.
- Comparative Study of MCDA Methods on GDSS Facilitator Election Problem. First National Conference in Computer Science Research and its Applications - RIA'23. Held online on May 10th 2023. Relizane University.
- Distributed Algorithm for Choosing a Facilitator within a Group Decision Support System. The First National Workshop on Computer Science and New Technologies CSNT'2024. December 08th 2024. Mostaganem University.

Abstract

The Breakdown of a modern industrial machine requires the intervention and collaboration of a group of experts in order to provide a diagnostic and a fix. This intervention can be done through the internet. This process is called collaborative e-maintenance. In order to provide these experts with a collaborative environment that works remotely, we use a Group Decision Support System (GDSS). Which improves the effectiveness of the process and reduces the loss in time and information. This system requires a human facilitator who coordinates and plans the meeting. His counterpart in the collaborative e-maintenance process is one of the experts called the coordinator. This leader is extremely important and has to satisfy certain criteria called the election criteria. A Delphi study was conducted to gather the facilitator election criteria. The criteria are classified into three categories: Experience, Network performance and Security. The election process can be implemented and automated by a machine through different solutions. Among the interesting solutions are Multi-Criteria Decision Aid (MCDA) methods and distributed election algorithms. This work proposes an approach in both solutions. Our multi-criteria approach starts by computing the election criteria weights using the Analytic Hierarchy Process method. Afterwards we applied four MCDA methods on a collaborative e-maintenance case study. The results of each method were compared based on several metrics. As a results, we found that MAUT selected the expert who satisfied most of the election criteria, while still having a better time complexity than PROMETHEE II. Additionally, a new election tool called GFET based on the multi-criteria approach was proposed. This work also proposes a model for the integration of GFET within the collaborative e-maintenance process. In contrast, since the multi-criteria approach requires inputting subjective parameters from decision-makers it is then subject to biases and behavioral problems. Consequently, the final decision or ranking can be impacted and thus the decision quality is also questionable. To fix the flaws present in MCDA methods, we also proposed a distributed election algorithm GFEA (GDSS Facilitator Election Algorithm) coupled with an objective weighting method called MEREC (Method based on the Removal Effects of Criteria). This eliminates possible biases since it doesn’t require subjective parameters from the decision-makers. The proposed algorithm is designed for unidirectional ring topology. Moreover, it considers multiple election criteria and uses a formal weighting method. Furthermore, it handles the failure and recovery of both the leader and election initiator. In addition, GFEA reserves a

backup leader to replace the leader in case of node or link failure. On top of this, the algorithm integrates a new tie-break mechanism that prioritizes the most important criteria rather than relying only on the UID (unique identifier).

Keywords: GDSS; MCDA; Facilitator; Leader Election Algorithm; Collaborative E-maintenance; Distributed Systems.

Résumé

La panne d'une machine industrielle moderne nécessite l'intervention et la collaboration d'un groupe d'experts afin de fournir un diagnostic et une solution. Cette intervention peut se faire via internet. Ce processus est appelé e-maintenance collaborative. Afin de fournir à ces experts un environnement collaboratif fonctionnant à distance, nous utilisons un Système d'Aide à la Décision de Groupe (GDSS). Ce qui améliore l'efficacité du processus et réduit la perte de temps et d'informations. Ce système nécessite un facilitateur humain qui coordonne et planifie la réunion. Son homologue dans le processus de e-maintenance collaborative est l'un des experts appelé le coordinateur. Ce leader est extrêmement important et doit satisfaire à certains critères appelés critères d'élection. Une étude Delphi a été menée pour recueillir les critères d'élection du facilitateur. Les critères sont classés en trois catégories : Expérience, Performance du réseau et Sécurité. Le processus d'élection peut être implémenté et automatisé par une machine grâce à différentes solutions. Parmi les solutions intéressantes figurent les méthodes d'Aide à la Décision Multicritère (MCDA) et les algorithmes d'élection distribués. Ce travail propose une approche dans les deux solutions. Notre approche multicritère commence par calculer les poids des critères d'élection en utilisant la méthode du processus analytique hiérarchique. Ensuite, nous avons appliqué quatre méthodes MCDA sur une étude de cas de maintenance électronique collaborative. Les résultats de chaque méthode ont été comparés sur la base de plusieurs métriques. En conséquence, nous avons constaté que MAUT a sélectionné l'expert qui satisfaisait la plupart des critères d'élection, tout en ayant une meilleure complexité temporelle que PROMETHEE II. De plus, un nouvel outil d'élection appelé GFET basé sur l'approche multicritère a été proposé. Ce travail propose également un modèle pour l'intégration de GFET dans le processus de maintenance électronique collaborative. En revanche, comme l'approche multicritère nécessite la saisie de paramètres subjectifs de la part des décideurs, elle est alors sujette à des biais et à des problèmes comportementaux. Par conséquent, la décision finale ou le classement peuvent être impactés et donc la qualité de la décision est également discutable. Pour corriger les défauts présents dans les méthodes MCDA, nous avons également proposé un algorithme d'élection distribuée GFEA (GDSS Facilitator Election Algorithm) couplé à une méthode de pondération objective appelée MEREC (Method based on the Removal Effects of Criteria). Cela élimine les biais possibles car il ne nécessite pas de paramètres subjectifs de la part des décideurs. L'algorithme proposé est

conçu pour une topologie en anneau unidirectionnel. De plus, il prend en compte plusieurs critères d'élection et utilise une méthode de pondération formelle. De plus, il gère la défaillance et la récupération du leader et de l'initiateur de l'élection. De plus, GFEA réserve un leader de secours pour remplacer le leader en cas de défaillance d'un nœud ou d'un lien. De plus, l'algorithme intègre un nouveau mécanisme de départage qui priorise les critères les plus importants plutôt que de s'appuyer uniquement sur l'UID (identifiant unique).

Mots-clés: GDSS; MCDA; Facilitateur; Algorithme d'élection du leader; e-Maintenance collaborative; Systèmes distribués.

Arabic Abstract

يتطلب تعطل أي آلة صناعية حديثة تدخل وتعاون مجموعة من الخبراء من أجل تقديم التشخيص والإصلاح. ويمكن إجراء هذا التدخل من خلال الإنترنت. وتسمى هذه العملية الصيانة الإلكترونية التعاونية. ومن أجل توفير بيئة تعاونية لهؤلاء الخبراء تعمل عن بعد، نستخدم نظام دعم القرار الجماعي (GDSS). وهو ما يحسن فعالية العملية ويقلل من الخسارة في الوقت والمعلومات. ويتطلب هذا النظام وجود ميسر بشري ينسق ويخطط للاجتماع. ونظيره في عملية الصيانة الإلكترونية التعاونية هو أحد الخبراء المسمى بالمنسق. وهذا القائد مهم للغاية ويجب أن يفي بمعايير معينة تسمى معايير الانتخاب. وقد أجريت دراسة دلفي لجمع معايير انتخاب الميسر. وتصنف المعايير إلى ثلاث فئات: الخبرة وأداء الشبكة والأمان. ويمكن تنفيذ عملية الانتخاب وأتمتتها بواسطة آلة من خلال حلول مختلفة. ومن بين الحلول المثيرة للاهتمام أساليب مساعدة القرار متعددة المعايير (MCDA) وخوارزميات الانتخابات الموزعة. ويقترح هذا العمل نهجًا في كلا الحلول. يبدأ نهجنا متعدد المعايير بحساب أوزان معايير الانتخابات باستخدام طريقة عملية التسلسل الهرمي التحليلي. بعد ذلك، طبقنا أربع طرق MCDM على دراسة حالة صيانة إلكترونية تعاونية. تمت مقارنة نتائج كل طريقة بناءً على العديد من المقاييس. ونتيجة لذلك، وجدنا أن MAUT اختار الخبير الذي استوفى معظم معايير الانتخابات، مع الحفاظ على تعقيد زمني أفضل من PROMETHEE II. بالإضافة إلى ذلك، تم اقتراح أداة انتخاب جديدة تسمى GFET تعتمد على نهج متعدد المعايير. يقترح هذا العمل أيضًا نموذجًا لدمج GFET ضمن عملية الصيانة الإلكترونية التعاونية. على النقيض من ذلك، نظرًا لأن نهج متعدد المعايير يتطلب إدخال معلمات ذاتية من صناع القرار، فإنه يكون عرضة للتحيزات والمشكلات السلوكية. وبالتالي، يمكن أن يتأثر القرار النهائي أو الترتيب وبالتالي تكون جودة القرار موضع تساؤل أيضًا. لإصلاح العيوب الموجودة في طرق MCDA، اقترحنا أيضًا خوارزمية انتخاب موزعة GFEA (خوارزمية انتخاب مُيسِّر GDSS) مقترنة بطريقة ترجيح موضوعية تسمى MEREC (طريقة تعتمد على تأثيرات إزالة المعايير). هذا يقضي على التحيزات المحتملة لأنه لا يتطلب معلمات ذاتية من صناع القرار. تم تصميم الخوارزمية المقترحة لطوبولوجيا الحلقة أحادية الاتجاه. علاوة على ذلك، فهي تأخذ في الاعتبار معايير انتخاب متعددة وتستخدم طريقة ترجيح رسمية. علاوة على ذلك، فهي تتعامل مع فشل واستعادة كل من القائد ومبادر الانتخاب. بالإضافة إلى ذلك، تحتفظ GFEA بقائد احتياطي ليحل محل القائد في حالة فشل العقدة أو الرابط. علاوة على ذلك، تدمج الخوارزمية آلية جديدة لكسر التعادل تعطي الأولوية للمعايير الأكثر أهمية بدلاً من الاعتماد فقط على UID (المعرف الفريد).

الكلمات المفتاحية: GDSS؛ MCDA؛ الميسر؛ خوارزمية انتخاب القائد؛ الصيانة الإلكترونية التعاونية؛ الأنظمة الموزعة.

Introduction

Collaborative E-maintenance process is the combination of remote maintenance and collaborative maintenance while using new information and communication technologies. This process involves groups of experts and technicians. Each group of experts collaborate to make decisions about solving a particular breakdown occurring in the company site where the technicians are available. Then the orders are passed to the technicians to apply the solution and thus treating the dysfunction. In order for these experts to work effectively and in harmony with the technicians, one of the experts has to be chosen as a coordinator. Which will be responsible for managing the entire process [1].

Group decision processes like the collaborative e-maintenance process can be improved and assisted by a system called Group Decision Support System (GDSS) [2]. This system includes a human part which includes the decision makers and the facilitator. The Facilitator has a crucial role in this system. As he holds high-level exchanges between group members while fostering cohesion and enhancing the meeting process as a whole [3, 4]. He is the equivalent of the coordinator.

Problematic

Our Problematic is the election of a facilitator within a GDSS. Given a set of decision makers, we select one of them to be the facilitator. Choosing different facilitators can lead to different outcomes in terms of time taken to reach a consensus, money spent and the quality of the decisions made. This is the reason why this issue has to be treated carefully.

The problematic treated in this work is formulated using the following questions: Among the set of DMs, how can we select one of them and assign him the role of GDSS facilitator? and how are DMs evaluated to find the most suitable DM for the facilitator role? To answer these questions, first, we need to list the possible solutions. Based on the problematic's nature there are 4 possible solutions: Leader election algorithms [5], multi-objective optimization [6], voting procedures or MCDA methods [7, 8].

MCDA methods are employed when facing a problem involving multiple stakeholders with different objectives. This type of decision problem requires selecting the alternative that satisfies

all the stakeholders. Since these problems are usually unstructured and involve some subjective parameters. MCDA methods are best suited for having a decision that decision makers can all agree on. Previous related work compared the experts based on five criteria which are: Experience as a DM, experience as a coordinator, download speed, distance from breakdown site and the expert's response time to a maintenance request [9]. There are other criteria to be considered, mainly in the categories of network performance and security. The security criterion has always been disregarded. Since it is hard to measure and involves several aspects like vulnerabilities and open ports.

Due to the similarities between the problematic of electing a leader in distributed systems and the problematic of electing a GDSS facilitator, distributed leader election algorithm represents a potential solution to the problematic of this work. Despite that, the solution must satisfy five requirements in order to be suitable for the problematic on hand. These requirements are: Consideration of multiple election criteria, inclusion of human related criteria, formal weighting method, backup leader and the ability to handle the failure and recovery of both the leader and the election initiator.

Objectives and Contributions

The objective of this work is to solve the problematic of facilitator election within a GDSS using two approaches: Multi-criteria approach and the Distributed election algorithms. Experimenting multiple MCDA methods would allow us to possibly view different results. We will use these results to compare between these methods and see each one's pros and cons.

This work's contributions are summarized as follows:

- Fixing GDSS facilitator election criteria using DELPHI method.
- Use of AHP (Analytic Hierarchy Process) to fix the election criteria weights.
- Application of MAUT, SAW, PROMETHEE II and TOPSIS on the problematic of electing a GDSS facilitator.
- Proposition of a new election tool called GFET (GDSS Facilitator Election Tool).
- Comparison of the utilized methods based on performance, ease of use and decision quality.

- Comparison between recent election algorithms designed for distributed systems, and spotting similarities between the problematic of electing a GDSS facilitator and the problematic of electing a leader for a distributed system. By exploiting these similarities, it is demonstrated that the distributed election algorithms can be effectively adapted to solve the problematic of electing a GDSS facilitator.
- Proposition of the GFEA (GDSS Facilitator Election Algorithm) algorithm optimized for the problematic of electing a GDSS facilitator based on human experience, security and network performance criteria.
- Use of the objective weighting method MEREC to fix the election criteria weights.
- GFEA robustness stems from the integration of several features like reserving a backup leader, improved tie-breaking mechanism that prioritizes criteria, leader recovery and tolerance to both initiator and leader failure.
- The three correctness properties are proven to be satisfied by the proposed election algorithm, and for the evaluation GFEA was tested on the collaborative e-maintenance process.
- Comparison of the proposed algorithm with other recent algorithms in terms of required functionalities for the problematic of GDSS facilitator election.

Thesis Structure

This thesis is divided into four chapters and is organized according to the following outline:

- Chapter 1: “Group Decision-making and GDSS”

  This chapter explains the concept of group decision-making and defines what a GDSS is. Additionally, it presents the strengths and limitations of it. Next, we focus on the GDSS facilitator by listing his responsibilities and missions.

- Chapter 2: “Multi-criteria Decision-making Methods”

  In the third chapter, we define, classify and explain MCDM methods. Then, some related works are presented. Succeeding this, we present the proposed multi-criteria approach

which is used later on to solve our problematic. The proposed approach consists of a subjective weighting method and a set of four MCDM methods.

- Chapter 3: “Distributed Election Algorithms”

  This chapter delves into the second approach we used to elect the GDSS facilitator. Distributed systems are defined and the problematic of electing a leader node is also explained and compared with our problematic. This chapter also presents and discusses the proposed distributed election algorithm GFEA by considering multiple failure and recovery cases of different types of nodes.

- Chapter 4: “Case Study: Collaborative E-maintenance”

  The field of collaborative e-maintenance which we chose for conducting our case study is defined and discussed thoroughly. In addition, we present the tasks required by the collaborative e-maintenance coordinator. His missions are illustrated using a detailed BPMN model. After that, the proposed multi-criteria approach is applied on this case study and the used MCDA methods are compared. This chapter also proposes an integration of the proposed election tool GFET into the collaborative e-maintenance process. The GFEA algorithm is then applied on a case study and is compared to other election algorithms based on functionality, performance and results.

Chapter 1

1. Group Decision-Making and GDSS

1.1. Introduction

This chapter starts by introducing the group decision-making process. Next, different group decision-making techniques are presented. Then, we move to GDSS and point out the importance, benefits and limitations of these systems. Afterwards, we cite some GDSSs types and explain the facilitator's missions.

1.2. Group Decision-making

A decision-making group is made up of two or more people who work together to identify an issue, determine its cause, develop potential solutions, evaluate potential solutions, and carry out plans to put the chosen solutions into action [10].

Problems involving several decision makers and alternatives are the subject of group decision making. There are multiple criteria to the alternatives. To put it another way, the decision-makers have to consider multiple criteria. A conflict may rise during group decision making, since every decision maker has his own preferences and objectives [11].

It is more likely that a problem will be identified during a group decision-making process compared to individual decision-making, and the decision-makers can then collaborate to develop a solution. As a result, collaborative decision-making becomes quicker and more efficient. Sharing responsibilities, utilizing decision-making expertise, gaining support from stakeholders, or assisting less seasoned group members [10].

1.3. Group Decision-Making Techniques

1.3.1. Brainstorming

A method of decision-making called "brainstorming" is used to describe a group's verbal idea generating. The main tenet of the brainstorming process is that everyone should be able to freely share their thoughts without worrying about humiliation or negative feedback from others. The objective is to come up with as many original and inventive ideas as possible [12].

1.3.2. Nominal Group Technique

The goal of NGT, an organized brainstorming method, is to generate a lot of ideas for a problem while guaranteeing that each group member contributes equally to the idea generating process [13]. NGT consist of six steps. First of all, the meeting session is started. Secondly, participants write their ideas quietly. Third, ideas are recorded using round-robin scheduling. In the fourth step, sequential discussions are made about the proposed ideas. The fifth step is about choosing the most important ideas through voting. In the final and sixth step, the participants discuss the chosen ideas [13].

NGT reduces the issues that frequently arise in face-to-face meetings. Moreover, the engagement of every participant is balanced in a sense that everyone gets a fair chance to generated ideas. Additionally, the generated ideas are more original than those generated in interactive group sessions. Furthermore, more ideas can be generated when compared to conventional interactive group sessions [13].

1.3.3. Delphi Technique

The Delphi method involves five steps [14]. The first one is about preparing the first questionnaire iteration. In the second step, the experts are chosen and invited to the session. They represent the experts' panel. The following step consist of gathering and evaluating the responses obtained in the first step. Next, feedback is given to the members. After that, design, distribute, and analyze the succeeding questionnaire round. The final step is to keep on running iterations until an agreement is reached [14]. Once the members arrive at a consensus, we present the final results to them.

1.3.4. Voting Procedures

Voting procedures are usually used when making social choices [15]. The process of voting entails selecting a winner for a vote using a method. Consequently, voting processes take on the role as decision-making instruments in a social choice framework, serving the dual purposes of objectively constructing a collective choice and selecting a winner or winners [16]. There are several voting procedures in the literature, such as: Simple majority rule, Copeland's method, Maxmin method, Plurality method and Borda count [15].

1.3.5. Game Theory

The goal of game theory is to identify the strategies that a group of players will agree upon in order to maximize their individual rewards [17].

1.4. Group Decision Support System

A GDSS is a type of CSCW (Computer Assisted Collaborative Work) [18], which can be used for group meetings where the participants communicate in different times and from different locations [19]. The GDSS usually engages a group of decision-makers and facilitator who have access to computers, decision models, database and a viewing screen [20]. Using integrated hardware, software and decision support tools, the GDSS helps the group decision making process [21]. The GDSS provides facilitation for finding more satisfying solutions within the limited solutions set. It is used as a tool to enhance the effectiveness of the group's search within the solution space [21]. This system includes tools with the purpose of focusing on the group decision process gains and decreasing its losses [22]. Figure. 1 shows the GDSS model based on the definitions included in papers [2] and [23]. There are several GDSS implementations, such as: Kindling, ThinkTank, ExpertChoice, Webcouncil and Facilitate Pro [24].

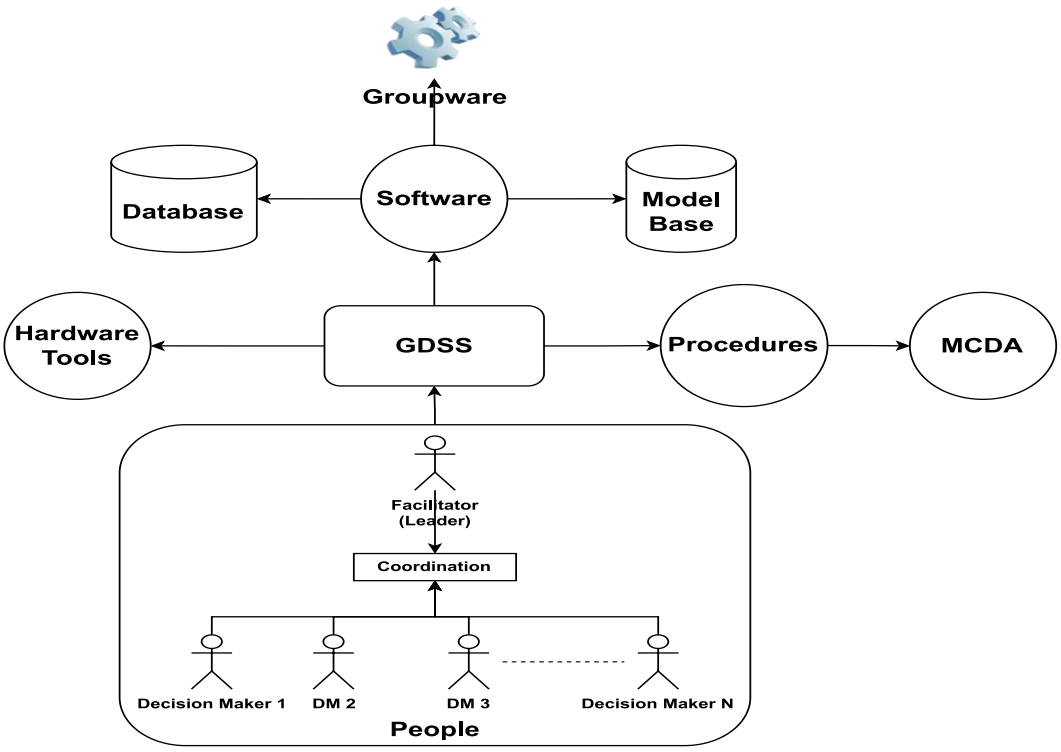

Figure. 1. GDSS Model [2, 23]

GDSS is a DSS (Decision Support System) that's designed to be used by a group of DMs (Decision Makers), who communicate by the mean of a communication subsystem [23]. It's a combination of a group of humans, hardware and software [23]. The software part includes a Groupware, which is a software designed to facilitate organized and methodical group work [25]. The GDSS enhances the group decision-making process of organizations [23]. The humans using the GDSS are grouped into two roles: the DMs and the facilitator [4]. It is necessary to have a human facilitator within a GDSS [23]. The traditional decision room is composed of multiple computers connected using a local network [5]. Nonetheless, it can also be extended to support connecting DMs who are geographically dispersed through internet [6]. One of the DMs is handed an important role called the facilitator role.

1.4.1. Advantages and limitations of GDSS

GDSS is said to have advantages such as anonymity to improve communication's equality and openness, parallel input to boost communication's speed and volume, and group memory to enhance information storage and use [26]. Furthermore, the usage of GDSS improves reciprocal comprehension and group creativity, among other things [26]. Figure. 2 shows the benefits of using a GDSS based on paper [27].

Among the limitations of the GDSS is that it does not enhance the satisfaction of the decision-makers. Moreover, GDSS lacks interactivity because roles and tasks are not assigned dynamically [19].

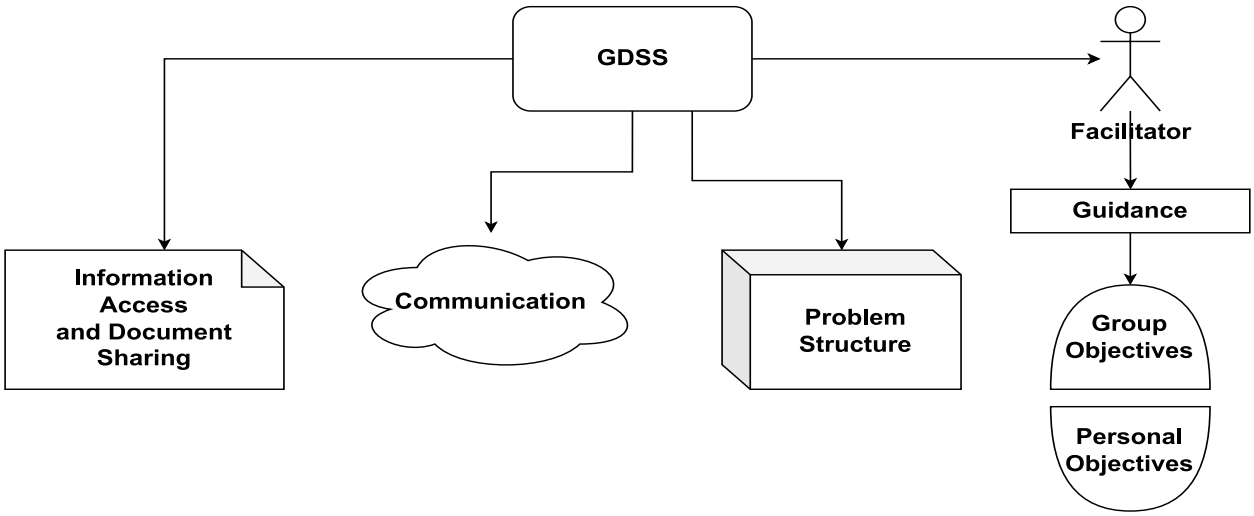

Figure. 2. GDSS Benefits [27]

1.4.2. GDSS Classification

When taking into account the location and time of the meeting, Barkhi et *al.* classified GDSS into two groups. Decision-makers in the same room participating in a meeting at the same time is considered a face-to-face GDSS. On the other hand, when each decision-maker is located at a different place yet they still communicate in the same time, then the system is considered a Distributed GDSS [28]. Additionally, another type of GDSS that is designed to have simple and common graphical user interface is called Web-based Multi-criteria GDSS. This is a GDSS that can be accessed through a web browser [28].

1.5. GDSS Facilitator Responsibilities

Group Facilitator provides key process guidance to help groups who may be struggling to navigate tasks and technology tools [29]. In [29], the authors described 26 skills required for online group facilitators. These tasks were arranged into seven classes: First, he develops a common group objective. Secondly, the facilitator establishes and maintains a common culture for the group. Moreover, he makes a plan for the meeting and does the preparation. In addition, the facilitator needs to have the ability and knowledge to use multiple online collaborative software. Another required competence is the communication with online participants who have different time zones, different geographical locations, and therefore different cultural backgrounds. The Final class is "Reflective Practitioner". This category includes enhancing the facilitation process by receiving feedback from participants about the positive aspects and the negative ones which need to be fixed. It also includes putting all facilitators at the same level of learning to ensure coherence in facilitation [29]. The facilitator must perform a role in which they assist the group in identifying their common goals, potential solutions, and an agreement [30]. Figure. 3 summarizes the GDSS facilitator missions based on paper [4].

The GDSS facilitator is responsible for various tasks. He walks the DMs through the meeting's agenda and starts the group conversation. Additionally, the facilitator can bring up new ideas if the DMs agree with that. Moreover, he can enhance the performance and effectiveness of the group. Furthermore, the facilitator helps the DMs using the technologies and doing their tasks. On top of this, he has to clarify the results of the meeting [7].

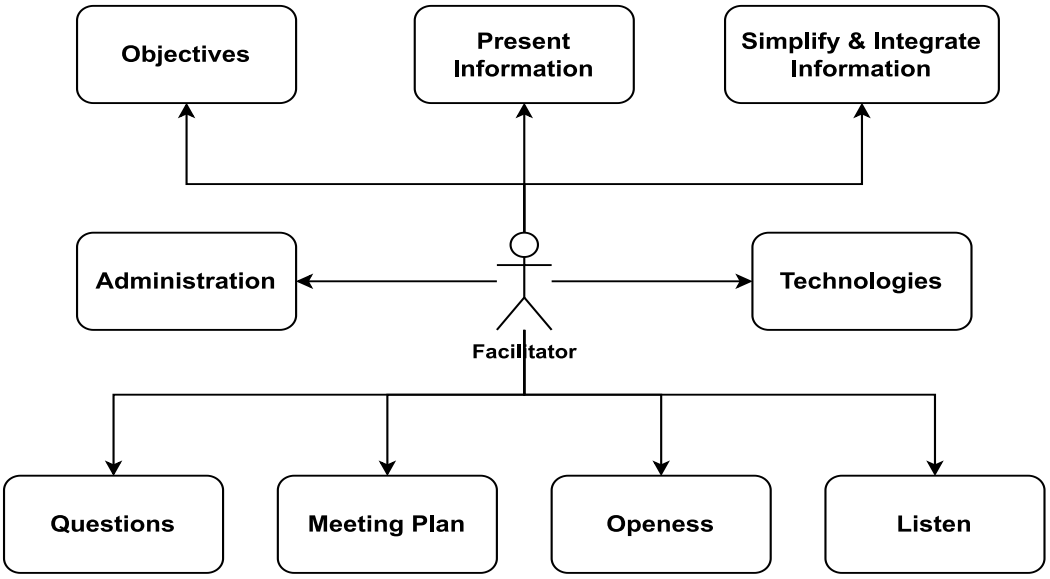

Figure. 3. Facilitator Missions Within GDSS[4]

1.6. Conclusion

In this chapter, we presented the concept of group decision-making along with some group-decision techniques that are used to reach a consensus and a final decision. We also explained what a GDSS is and discussed its benefits and limitations. Finally, we explained, detailed and emphasized the importance of the GDSS facilitator.

The next chapter explores MCDA methods and proposes a new multi-criteria approach to solve the problematic of electing a GDSS facilitator.

Chapter 2

2. Multi-criteria Decision-Making Methods

2.1. Introduction

This chapter presents an approach that can be used to solve our problematic, which is the multi-criteria approach. We present the MCDA methods and explain why they are used on a GDSS.

Next, related works that used MCDA methods with a GDSS are cited. The subsequent section lists some popular methods and compares them in a brief manner. The next section starts by explaining the steps involved in applying AHP method to attribute weights to the election criteria. In addition, the AIP aggregation procedure is described. After that, we present the steps and formulas of MAUT, SAW, PROMETHEE II and TOPSIS. Next, we present the strengths and weaknesses of the proposed multi-criteria approach. Moreover, the proposed approach is compared with related works in a concise manner. Finally, we end the chapter with our conclusion regarding the multi-criteria approach.

2.2. Multi-criteria Decision Aid Methods

MCDA is a subfield of operational research which deals with sophisticated decision-making problems. These problems involve several objectives that conflict with each other as well as a high level of uncertainty [31]. Given a set of potential courses of action, MCDA methods assess those actions and provide either a ranking of options, classification or a choice [8]. The use of these methods gives the decision makers an improved understanding of the decision problem [31]. Usually, problems involving several objectives and options require the intervention of multiple decision makers often known as stakeholders [32]. These DMs constitute a group that's going to choose an alternative that benefits in the attainment of goals [32].

2.3. Multiple Criteria Decision Aid Approach in GDSS

There are several reasons to use the MCDA approach in GDSS such as: The management of complexity, Transparency, Learning, Audit trail and legitimacy [32]. In addition, MCDA methods grant the DMs a structured methodology for solving problems involving several points of view and objectives [32]. Moreover, the multi-criteria approach aids in minimizing the problem's unstructured nature by allowing participants to collectively establish a framework for exchanging information [33]. Furthermore, when dealing with such interpersonal disagreements, MCDA approach may be an effective tool. The goal is to reach an agreement among the group members or at the least, try to decrease the level of conflict by making compromises [33].

The group process assisted by MCDA methods starts by identifying the decision makers and clarifying the session context. Next, the decision objectives are identified in order to elaborate comprehensive criteria and measure the attainment of objectives. After that, we generate the

alternatives. Following this, we reach the phase of preferences elicitation in which we gather the criteria weights. Furthermore, the alternatives are evaluated based on the decision criteria. Finally, the obtained results and final decision is analyzed and presented to the DMs [32].

Some of the things that could cause bias or subjective non-optimal choice is the way that the DM is influenced by his society, cultural background, other DMs and even his emotions. The location and physical setup of the meeting room can even influence the DM decision [32].

The use of MCDA methods in group decision process has been applied in multiple fields. Examples of this include: Assessing alternative strategies in response to nuclear emergencies, nutrient management, prioritization of research themes, assessing environmental and energy policies consequences, strategic action plans for hospital trust and treating problems of water and environmental management like water regulation [32].

2.2. Related Works

Omari Youcef *et al* integrated the TOPSIS method and the Softmax function within WIM-GDSS in order to assign weights to the decision makers [34]. First, the DMs were ranked using TOPSIS based on their contribution to the decision problem. Then, the Softmax function was used to make the DMs weights add up to one.

Another work of Omari Youcef *et al* consisted of developing a new GDSS based on WIM-GDSS [35]. This newer system uses PROMETHEE II for the prediction module compared to TOPSIS in the previous version. AHP was used to fix the criteria weights. The developed system integrates a coordination protocol in which the initiator of the decision problem generates the criteria and alternative solutions. He verifies if the problem is valid by comparing the number of accepted invites to the acceptance threshold. Then, each DM evaluates the criteria through pair-wise comparisons and inputs them into AHP method to obtain the criteria weights, and each one enters his preference parameters. Next, the smart agent assigned to the DM, uses a prediction model that matches the DM preferences. If the model doesn't exist, PROMETHEE II is used to calculate the objective vector. In this instance, PROMETHEE II will also be used to train the new prediction model for later use. Following this phase, the initiator aggregates all individual objective vectors into the global matrix of performances. Succeeding this, PROMETHEE II is used again on the GMP to obtain the final objective vector. The authors compared Multi-Layer Perceptron (MLP)

and Linear Regression (LR) models in predicting the PROMETHEE II results. The results showed that MLP has a better accuracy than LR.

Amin Miftakul *et al* treated the problematic of choosing supervisor lecturers for the student creativity programs in the computer engineering department of the Sriwijaya State Polytechnic [36]. They proposed a new GDSS model which allows each DM to rank individually the alternatives using the Weighted Product method. Then, the BORDA method is implemented to aggregate the individual rankings into a final global ranking of the alternatives. The authors created a web application which uses this GDSS model in order to give recommendations when electing a supervisor lecturer within their department.

Laredj A. *et al* tackled the problematic of electing a coordinator within a collaborative e-maintenance process [9]. They used the ELECTRE I MCDA method to elect one of the experts as the coordinator. This method resulted in a partial ranking of the experts. Meaning, some experts were incomparable. The experts were evaluated based only on five criteria: experience as an expert, network speed, distance to breakdown site, coordinator experience and response time. In addition, the weights were fixed without using a formal weighting method. Moreover, using a MCDA method that doesn't provide a complete ranking of the experts resulted in the absence of a backup leader who would handle the process in case the connection between the coordinator and technicians is lost. This requires running another iteration of the election process just to replace the former leader [9]. Furthermore, their work didn't contain visual charts that should help in analyzing and comparing between the different experts. Finally, this paper didn't evaluate the performance and quality of using the ELECTRE I method on their example.

2.3. Classification of MCDM Methods

There are three main problem natures: Choice, Ranking and Sorting. Choice problems consist of choosing the smallest subset of alternatives considered as the best alternatives. Sorting problems aim at classifying alternatives into groups. Ranking problems are about ordering alternatives from best to worst [37].

Methods that treat ranking problems can also be used for choice problems. Because after ranking the alternatives from the best to the worst, we can select the best one as a final choice. The

leader election problematic is a choice problem. Some of the well-known MCDA methods are classified according to Table 1 [38].

Table 1. Classification of MCDA methods

MCDA Method	Method Type	Problem Nature	Ordering	Incomparability
AHP	Full Aggregation	Ranking or Choice	Full	No
MAUT	Full Aggregation	Ranking or Choice	Full	No
ELECTRE I	Outranking	Choice	Partial	Possible
PROMETHEE I	Outranking	Choice	Partial	Possible
PROMETHEE II	Outranking	Ranking or Choice	Full	No
TOPSIS	Reference Level	Ranking or Choice	Full	No
SAW	Full Aggregation	Ranking or Choice	Full	No

In this work we tried to avoid methods that may include incomparabilities like ELECTRE I and PROMETHEE I. Which leads to partial ordering of the alternatives, since there might be experts who we can't compare between them, meaning no preference or indifference relation is established between two experts. Instead, we chose MAUT, SAW, PROMETHEE II and TOPSIS as they provide full ordering of the experts. This allows us to have a backup expert that would replace the facilitator's position in case the connection between him and the breakdown site is lost [9].

2.2. Proposed Multi-criteria Approach

From what we have seen in previous sections, the coordinator of experts' group has several missions and the group ability to troubleshoot the breakdown by cooperating and uniting the efforts relies heavily on choosing the appropriate coordinator. Which leads us to our problematic: How can we elect one of the experts to be the coordinator and leader of the group? And how can we choose the most suitable expert for the position?

Multi criteria decision aid methods are a suitable solution for this kind of problematic, as it takes several criteria as an input, and outputs a consensual decision that satisfies the DMs. This chapter focuses on the multi-criteria approach and aims at finding the appropriate MCDA method

for electing a facilitator by trying four methods and comparing the obtained results. Figure. 4 summarizes the conducted approach in this work to tackle our problematic.

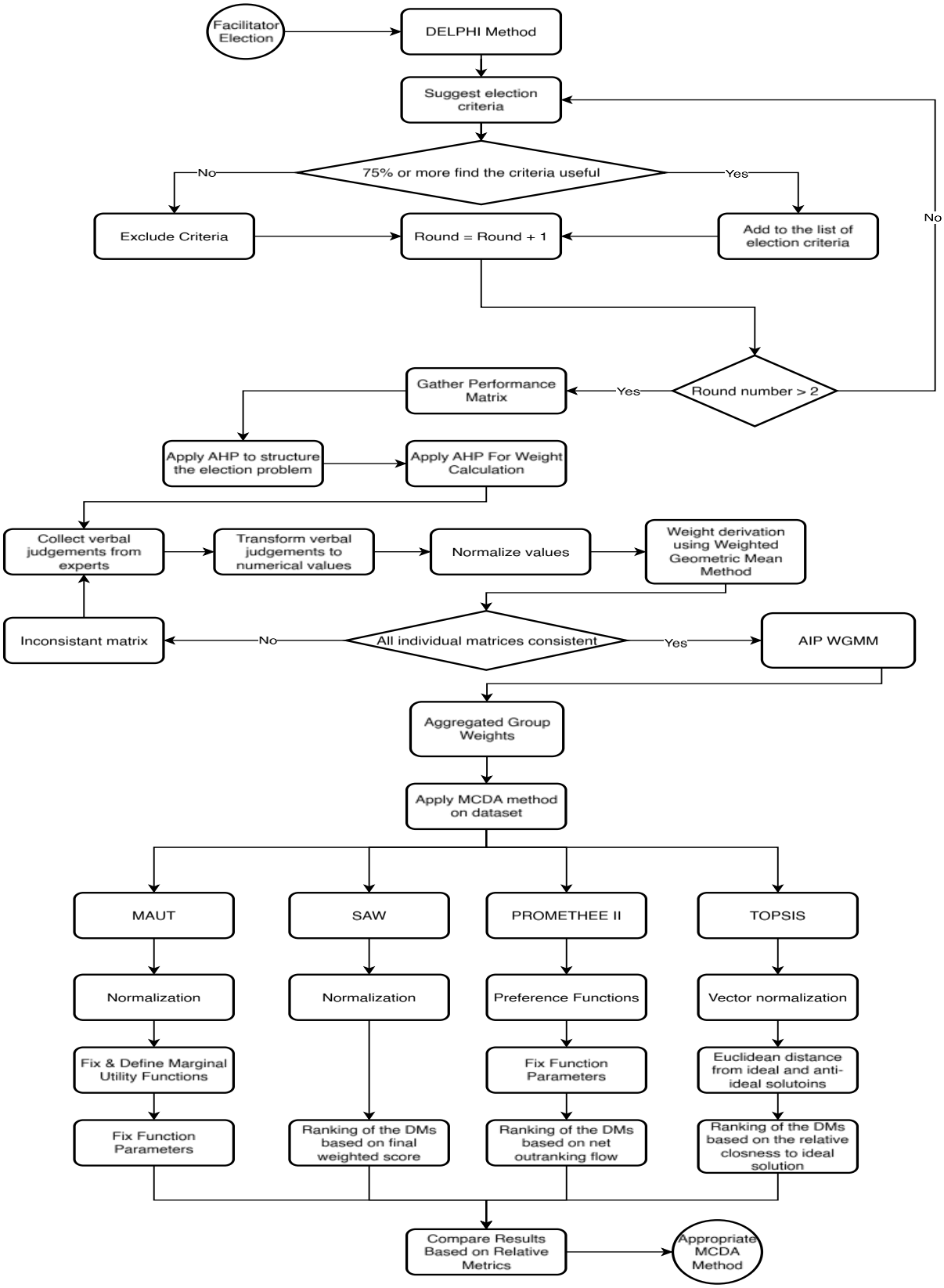

Figure. 4. Flow Chart Summarizing the Used Approach

2.3. Analytic Hierarchy Process (AHP)

Analytic Hierarchy Process is a multicriteria method used in decision making. It uses pairwise comparisons to rank alternatives or criteria based on their relative importance. First of all, the problem is structured in the form of levels. Starting from the main objective (highest level), then to criteria and sub-criteria if available. Ending with the lowest level which contains the alternatives. Next, for each pair of the same level, the decision maker is asked which alternative is more important and how much important is it. The subjective judgements are then translated into numerical values using Saaty's scale of relative importance [39]. Table 2 shows this scale in the context of comparing pairs of decision criteria.

Table 2 Saaty's Relative Scale of Importance [39] for Criteria Comparison

Degree of Relative Importance	Verbal Judgement
1	The two criteria are of same importance
3	A criterion is slightly more important than the other one
5	A criterion is strongly more important than the other criterion
7	A criterion has been shown in practice to be more important than the other one
9	A criterion is extremely important when compared to the other one
2, 4, 6, 8	If a compromise is required between a pair of criteria
Reciprocals	Inversing the comparison

The values are then normalized. Finally, there are multiple methods for obtaining the weights. The most common ones are the normalized column sum method, the geometric mean method, the eigenvector method and the normalized mean (average) method [40]. The normalized geometric mean method is shown in Eq. (1). This gives us the relative importance of that alternative compared to others. In the context of criteria, relative importance represents the weights of the criteria [39]. Performance wise, the time complexity of AHP is $O(n^3)$ [41].

$$w_j = \frac{\left(\prod_{j'=1}^{m} x_{jj'}\right)^{\frac{1}{m}}}{\sum_{j=1}^{m}\left(\prod_{j'=1}^{m} x_{jj'}\right)^{\frac{1}{m}}}. \quad (1)$$

Where w_j is the weight of the jth criterion, $x_{jj'}$ is the comparison value between criterion j and j', and m is the number of criteria.

In order to evaluate the solution's consistency, we calculate the Consistency Ratio (CR). First, we get the eigenvector of the pairwise comparison matrix using Eq. (2) [40].

$$Vj = \sum_{j'=1}^{m} x_{jj'} * w_{j'} \quad (2)$$

Where V_j is the eigenvector of jth criterion.

Secondly, we calculate the eigenvalue λmax by calculating the average of the eigenvector, with m being the number of criteria [40]:

$$\lambda max = \frac{\sum Vj}{m} \quad (3)$$

Next, we compute the Consistency Index (CI) using the formula [40]:

$$CI = \frac{\lambda max - m}{m - 1} \quad (4)$$

The formula for calculating the CR is [40]:

$$CR = \frac{CI}{RI_m} \tag{5}$$

With RI_m being the Random Consistency Index obtained from an array containing the consistency indices of randomly generated pairwise comparison matrices [40]. We select the RI corresponding to the number of criteria in our dataset.

2.3.1. Aggregation of Individual Preferences (AIP)

AIP [42] allows us to aggregate multiple individual weight vectors that are computed using AHP into a single group weight vector. This method is represented by Eq.(6) [42]. In this procedure, we aggregate the individual weight vectors of the criteria coming from each DM. Every DM should have a weight.

$$w_j = \left[\prod_{e=1}^{n} (wv_e(c_j))^{w_e}\right]^{1/\Sigma w_e} \tag{6}$$

Where wv_e is the weight vector of DM e. w_e is the weight of DM e, and c_j is the individual weight of criterion j.

2.4. Multi Attribute Utility Theory

Multi Attribute Utility Theory is a multi-criteria decision aid method. This method can be used in ranking and choice problems. It provides a complete ranking of the alternatives since it is an outranking approach without incomparability cases [38]. The absence of incomparability is because MAUT ranks alternatives based on a utility function which returns a real number, and two real numbers are always comparable [43]. Either one is preferred to another or they are indifferent. A software called “RightChoice” is based on MAUT and is free to use. The desirability of an alternative based on a criterion is measured by a marginal utility function. There are four types of marginal functions in the MAUT method: Linear, Logarithmic, Exponential, Step and Quadratic. The function is decreasing in the case of non-beneficial criterion and increasing when dealing with beneficial criterion. First, the evaluation matrix has to be normalized. The normalization of non-

beneficial and beneficial criteria is different. The non-beneficial criteria normalization leads to negative values. Consequently, the marginal function produced will be decreasing [43, 44]. The function can be modified by the decision maker in order to reflect his preferences.

The normalization formula for beneficial criteria is as follows [43, 44]:

$$g'_j(a_i) = \frac{g_j(a_i) - min(g_j)}{ma\,x(g_j) - mi\,n(g_j)} \tag{7}$$

Where $g_j(a_i)$ is the performance of alternative i according to criterion j.

The normalization formula for non-beneficial criteria is the following [43, 44]:

$$g'_j(a_i) = 1 + \left(\frac{\min(g_j) - g_j(a_i)}{\max(g_j) - \min(g_j)}\right) \tag{8}$$

The exponential marginal utility function u_1 is defined as follows [43, 44]:

$$u_1 = \frac{e^{\left(g'_j(a_i)\right)^2} - 1)}{1.72}; i = 1, \dots, n; j = 1, \dots, m \tag{9}$$

The final utility score S_i is calculated using the following additive model [44]:

$$S_i = \sum_{j=1}^{m} u_{ij} * w_j\,; i = 1, \dots, n \tag{10}$$

Where u_{ij} is the utility score of alternative i according to criterion j.

2.5. Simple Additive Weighting

Simple Additive Weighting is a MCDA method that can be used to rank the alternatives based on a score. The evaluations of the alternatives need to be normalized in order to obtain the normalized decision matrix. The normalization for beneficial and non-beneficial criteria is different. The final score for each alternative is obtained by multiplying each normalized value by the criterion weight and then adding up all the weighted values of all the criteria [45].

Normalization of Non-beneficial Criteria [45]:

$$\boldsymbol{g'_j(a_i) = \frac{\min(g_j)}{g_j(a_i)}} \tag{11}$$

Normalization of Beneficial Criteria [45]:

$$\boldsymbol{g'_j(a_i) = \frac{g_j(a_i)}{\max(g_j)}} \tag{12}$$

Final SAW score S_i for alternatives is obtained by adding up the normalized values multiplied by the corresponding criteria weights [45].

$$\boldsymbol{S_i = \sum_{j=1}^{m} g'_j(a_i) * w_j} \tag{13}$$

Where $g'_j(a_i)$ is the normalized value of alternative i according to criterion j.

2.6. PROMETHEE II

Preference Ranking Organization Method for Enrichment of Evaluations is an MCDA method designed for ranking and choice decision problems [44]. PROMETHEE I gives a partial ordering, whereas PROMETHEE II outputs a value called the net outranking flow which allows us to obtain a complete ordering of the alternatives without any incomparabilities. PROMETHEE requires only two additional sets of information: the weights of criteria and the preference functions for each criterion [46].

First, we compute the evaluative differences D_j which are the differences between pairs of alternatives i and k for a certain criterion j (see Eq. (14)) [46]:

$$\boldsymbol{D_j(a_i, a_k) = g_j(a_i) - g_j(a_k); i, k \in 1, \dots, n, j = 1, \dots, m} \tag{14}$$

Next, the generalized criteria are obtained by applying one of the six preference functions on the evaluative difference between two alternatives i and k [46]. For each criterion, the DM has to choose the appropriate preference function. Concerning beneficial criteria, we use Eq.(15):

$$\boldsymbol{P_j(a_i, a_k) = f[D(a_i, a_k)]\ with\ 0 \leq P_j(a_i, a_k) \leq 1} \tag{15}$$

For non-beneficial criteria we use Eq. (16) [46]:

$$\boldsymbol{P_j(a_i, a_k) = f[-D(a_i, a_k)]\ with\ 0 \leq P_j(a_i, a_k) \leq 1} \tag{16}$$

Furthermore, the preference index shows how much an expert i is preferred to an expert k:[46]

$$\boldsymbol{\pi(a_i, a_k) = \sum_{j=1}^{m} Pj(a_i, a_k)\,.\,w_j;\ i, k\ \in \{1, \dots, n\}} \tag{17}$$

Moreover, the positive outranking flow (Leaving Flow) indicates how much the expert i is outranking expert k [46]:

$$\boldsymbol{\varphi^{+}(a_i) = \frac{1}{n-1} \sum_{e_i \in E} \pi(a_i, a_k);\ i, k \in 1, \dots, n} \tag{18}$$

While the negative outranking flow (Entering Flow) indicates how much the expert i is outranked by expert k [46]:

$$\boldsymbol{\varphi^{-}(a_i) = \frac{1}{n-1} \sum_{a_i \in A} \pi(a_k, a_i);\ i, k \in 1, \dots, n} \tag{19}$$

.

In order to obtain a full ordering of the alternatives, we calculate the net outranking flow according to Eq. (20) [46]. The alternatives will be ranked from best to worst based on it. The alternative with the maximum net outranking flow will be the best one.

$$\boldsymbol{\varphi(a_i) = \varphi^{+}(a_i) - \varphi^{-}(a_i);\ i = 1, \dots, n} \tag{20}$$

2.7. TOPSIS

Technique of Ordering Preferences by Similarity to the Ideal Solution (TOPSIS) is a MCDA method which has been applied on several fields including manufacturing systems [40]. TOPSIS provides a complete order by ranking the alternatives based on how close they are to the ideal solution which has the maximum values. This method is divided into 6 steps.

First, we need to normalize the values of the performance matrix. There are multiple normalization methods which could lead to different results. The most common ones are the vector and linear normalization. In this work, we used the vector normalization which is computed for both beneficial and non-beneficial criteria according to Eq. (21) [40]:

$$r_{ij} = \frac{g_j(x_i)}{\sqrt{\sum_{i=1}^{n} g_j(x_i)^2}} \tag{21}$$

Secondly, we need to multiply the normalized values by the criteria weights [40]. In the third step, we have three ways to fix the ideal and anti-ideal solutions. One way is to choose the two vectors of best and worst values for all criteria, which is what we did in this work. Another method, is to define fictitious alternatives which have the best and the worst theoretical values on all criteria. The most subjective way is to allow the DM to fix those reference solutions. The fourth step, consist of calculating the distance of each alternative from the ideal and anti-ideal solutions. This can be done by using the Euclidean formula or any other distance formulas (see Eq. (22) and Eq. (23)) [40]:

$$D^{+} = \sqrt{\sum_{j=1}^{m} \left(v_{ij} - v_j^{+}\right)^2} \tag{22}$$

$$D^{-} = \sqrt{\sum_{j=1}^{m} \left(v_{ij} - v_j^{-}\right)^2} \tag{23}$$

With i indicating the index of the alternative, and j is the index of the criterion.

The relative closeness of each alternative is calculated in the fifth step according Eq. (24) [40]:

$$RC^{+} = \frac{D_i^{-}}{D_i^{+} + D_i^{-}} \tag{24}$$

The final step provides a ranking of the alternatives based on their relative closeness to the ideal solution. This ranking is done in descending order from the best alternative to the worst.

2.8. Qualities and Limitations of the Used Approach

Fixing the election criteria weights using the AHP method is a structured approach when compared to using random predefined weights. Moreover, it enables the decision makers to compare between each pair of criteria and indicate which one is more important and to what degree. Furthermore, AHP translates verbal judgements into numerical values that can be used for further calculations. In addition, AHP uses the consistency ratio to verify the consistency of the weights. However, the main drawback of this method is the high number of pair-wise comparisons which can be taxing in time and effort [47]. On top of that, when the weights aren't consistent, the DMs need to redo the verbal judgements and recalculate the consistency ratio to verify the consistency of the weights. In contrast, these calculations can be automated using a computer software. This is seen in the M-MACBETH software which implements the MACBETH MCDA method and provides recommendations to fix the inconsistent values [48].

Concerning the multi-criteria approach in general, it has several advantages. Including satisfying each DM preferences. Moreover, it allows the DMs to reach an agreement. In addition, MCDA methods give the participants flexibility by allowing them to choose the preference functions and parameters. On the other side, these methods are very subjective and informal. As they are mostly based on human judgements which can be biased and inaccurate.

To summarize, the qualities of using the proposed approach are:

- Multi-criteria approach aims at satisfying each DM preferences.
- Allows the DMs to reach an agreement.

- MCDA methods give the participants flexibility by allowing them to choose the preference functions and experiment with different parameters.
- Fixing the election criteria weights using the AHP method is a structured approach when compared to using random predefined weights.
- It enables the decision makers to compare between each pair of criteria and indicate which one is more important and to what degree.
- AHP translates verbal judgements into numerical values that can be used for further calculations.
- AHP uses the consistency ratio to verify the consistency of the weights.
- Calculations can be automated using a computer software. This is seen in the M-MACBETH software which implements the MACBETH MCDA method and provides recommendations to fix the inconsistent values [48].

The limitations of the used approach are:

- High number of pair-wise comparisons which can be taxing in time and effort [47].
- When the weights aren't consistent, the DMs need to redo the verbal judgements and recalculate the consistency ratio to verify the consistency of the weights.
- MCDA methods are subjective and informal. As they are mostly based on human judgements, which can be biased and inaccurate.

Many works proposed new GDSS models that make use of multi-criteria methods such as [35, 36, 49]. However, they didn't use these methods to elect the facilitator, and they didn't specify how the facilitator is selected. Laredj et *al.* [9] used the multi-criteria method ELECTRE I within the collaborative e-maintenance process to select an expert as the coordinator. The method has partial ranking, resulting in the absence of a backup coordinator. Additionally, their work didn't use a formal weighting method. Furthermore, election criteria weren't comprehensive as they only used 5 criteria for the evaluation of the experts.

2.9. Conclusion

This chapter introduced the multi-criteria decision-making methods. We showed how these methods can be used within a GDSS and then presented some related works. Afterwards, we classified some MCDM methods to identify the methods that are applicable on our problematic. Following this, we presented the proposed multi-criteria approach which consist of using AHP alongside AIP for fixing election criteria weights. Additionally, this approach employs a ranking MCDM method in order to rank the DMs from best to worst based on the method's score. Four MCDM methods were considered, which are: MAUT, SAW, PROMETHEE II and TOPSIS. Subsequently, we discussed the strength and limitations of the proposed multi-criteria approach, and evaluated it against other relatable research works.

One of the main drawbacks of MCDA methods are biases and behavioral problems. Since MCDA methods are subjective to the decision maker's preferences and depend on his input and parameters. If a DM is biased towards one side, it could lead to him preferring some alternatives over others, which influences the final ranking [32].

There are other approaches which can be applied on the facilitator election problematic. An interesting option is leader election algorithms. Because of the high similarity between the problematic of choosing a controller node and our problematic. An appropriate algorithm would take the criteria and the weights as inputs and return the chosen leader as an output. This algorithm needs to be optimized by trying to reduce time complexity and execution time as low as possible. The main advantage of using this approach is its formality, as it doesn't involve subjective parameters and is optimized for machine automation.

The next chapter compares different leader election algorithms and proposes a new algorithm optimized for human facilitator election.

Chapter 3

3. Distributed Election Algorithms

3.1. Introduction

Various election algorithms have been suggested in the literature [3, 8–19]. Some works integrate some of the required features for electing a GDSS facilitator. But no algorithm satisfies all the requirements of this problematic. Currently, no distributed election algorithm optimized for this problematic has been proposed. Moreover, the existing algorithms are optimized for distributed machines and not humans. The work in [8] proposes an election algorithm based on the load and failure rate of the nodes. While this algorithm involves multiple criteria, still it doesn't consider any human related criterion, plus it doesn't specify a formal weighting method. Consequently, it is required to either adapt an existing algorithm for our problematic, or design a new one entirely.

In this chapter, a new distributed election algorithm designed for the GDSS facilitator election is proposed. The proposed algorithm takes multiple election criteria into consideration including human experience, while using a proper weighting method to indicate the importance of each criterion, thus, influencing the final results. Furthermore, the algorithm elects not only the facilitator but also a backup leader. Moreover, the GFEA (GDSS Facilitator Election Algorithm) satisfies all three correctness properties, and adds a new tie breaking mechanism which considers the most important criteria instead of just relying on the UID (Unique Identifier). On top of that, the algorithm is fault tolerant and considers the disconnection of DMs, Leader, and even the initiator. This algorithm can also be applied on machines by changing the election criteria according to the problem on hand.

This chapter starts by defining what distributed systems are, and presents their different use cases. After that, we will see the interaction models. Next, distributed are classified into three main categories which are also split into other subcategories. Subsequently, we present some network topologies that are used in these systems. Following this, we explain the communication paradigms used in distributed systems among which we have the Message Passing Interface standard. This standard is then described, and its python binding mpi4py is presented. Then, failure models and failure detector theories are explained. Afterwards, we discuss two failure detector

implementations based on timeout. The succeeding section spots the similarities between our problematic and the problematic of leader election within distributed systems. Next, we present the comparative study that was conducted to compare between different distributed election algorithms that exist in the literature. Succeeding this, we explain why we opted to use the distributed election algorithms approach to solve the problematic of electing a GDSS facilitator. The following section introduces the objective weighting method MEREC used to fix the election criteria weights. Next, we give information about the system's model, including the topology, assumptions and notations. Next, the proposed algorithm GFEA is detailed, and is analyzed in terms of time and message complexity. Finally, GFEA is compared to other recent related works based on functionalities.

3.2. Distributed System Definitions

A distributed system is a group of computers or mobile devices connected through a network. These devices work together, appearing as a single unit to users, to achieve a common goal and deliver a service [1]. According to Van Steen and Tanenbaum [50], a group of independent computing devices that the users interact with as if it was a single, cohesive system is known as a distributed system.

According to Min Huang and Bode a distributed system is composed of multiple computers that don't share a synchronized clock nor do they share a common memory. These computers are linked using a communication network, by which each computer can access remote resources [51].

3.3. Distributed System Use Cases

These systems are employed in various fields, they are integrated in banking networks, smart container systems, smart plants, industry 4.0, IoT, smart cities and many other application cases [52].

3.4. Interaction Model Variants

3.4.1. Synchronous Systems

A synchronous distributed system is a system that has known time bounds. These times bounds exist for the time required for a process to execute a step, message transmission time and local clock drift rate of processes [53].

3.4.2. Asynchronous Systems

Asynchronous systems don't have time bounds. Meaning that transmitting a message, executing a program or clock drift rates all have an arbitrary time. Thus, sometimes certain operations can be done in a short time, but other times the same operations can take a long time [53].

3.4.3. Partially Synchronous Systems

In this variant of distributed systems, at the beginning, the system is asynchronous for an unknown period of time. But eventually, the system becomes synchronous [54].

3.5. Classification of Distributed Systems

Distributed systems can be classified into three main categories: Pervasive, High Performance and distributed information systems [50]. Each category includes multiple subgroups.

3.5.1. High Performance Computing

High Performance Computing (HPC) refers to the methods and techniques used to solve complicated and large-scale problems more quickly and inexpensively [55]. It combines supercomputers and parallel processing to create sophisticated and potent applications [55]. HPC is used in several fields such as finance, aerospace, medical field and urban planning [55].

- **Grid Computing**

  Grid computing is a type of computing infrastructure where multiple geographically dispersed computer resources are combined for a shared goal. It devotes all the shared resources to achieve a single job that cannot be done using a single computer [56].

- **Cluster Computing**

  Computers of homogenous hardware and software are linked together as a unit using a fast local area network [56].

- **Cloud Computing**

  Cloud computing is based on Utility computing which allows a user to pay for access of using resources from a datacenter [50]. It can provide a Software, Platform or Infrastructure as a service [50].

3.5.2. Pervasive Distributed System

Pervasive systems seek to give people constant, transparent access to services through the use of devices carried by the user or integrated in their immediate physical environment [57].

- **Mobile Computing**

In mobile computing, accessing and processing information as well as carrying out tasks is done through portable devices like tablets and smartphones while on the go [58].

- **Sensor Networks**

Wireless sensor networks are made up of geographically dispersed autonomous devices that use sensors to track various environmental and physical parameters, such as temperature, pressure, sound, vibration, movement, pollution, and in different locations. These gadgets use a wireless connection to talk to one another. These sensors might only have so much memory, processing power, energy, and bandwidth available to them. They are easily positioned in physical spaces and are exhibited in modest physical dimensions [59].

- **Ubiquitous Computing**

A model of human interaction with computing devices known as ubiquitous computing envisions an extensive number of tiny electronic devices subtly incorporated into day-to-day activities [60].

3.5.3. Distributed Information Systems

- **Transaction Processing**

In the context of database transaction processing, the client can send a nested transaction which consists of multiple requests that can be targeting different databases. Then the nested transaction is distributed across parallel machines to enhance performance [50].

- **Enterprise Application Integration**

If applications become decoupled from the database, there should be a way to enable the interoperability between different applications. In order to allow these applications to communicate with each other, a middleware is used. Applications can communicate through RPC (Remote Procedure Call), RMI (Remote Method Invocation) and MOM (Message-Oriented Middleware) [50].

Table 3 Distributed Systems Classification [50]

Category	Subclasses
Pervasive	• Mobile Computing • Sensor Networks • Ubiquitous Computing
High Performance	• Cloud Computing • Grid Computing • Cluster Computing
Distributed Information Systems	• Transaction Processing • Enterprise Application Integration

3.6. Distributed System Topologies

Minar [61] evaluated the effectiveness of multiple distributed system topologies based on seven properties which are: Coherence, Scalability, Management, Fault-tolerance, Security, Lawsuit-proof and Extensibility. Based on how nodes are linked and organized, they form several network topologies. In the following, we will present the main ones:

3.6.1. Centralized

In this topology, every slave node is only connected directly to the master node. Centralized systems are not fault tolerant as other nodes depend on a single central node. However, these systems are much easier to manage when compared to decentralized systems [61].

3.6.2. Bus

Bus topology connects multiple computers using a single main cable which can be a fiber optic cable. A terminator is connected to both the right and left ends of the cable. Each computer has a unique node address and is connected to the main cable using a tee connector [62].

3.6.3. Ring

In this topology, each node has two adjacent nodes and all the nodes form a closed circle shape [51]. However, the physical layout of the nodes doesn’t have to be circular [62].

3.6.4. Hierarchical (Tree)

Tree topology, sometimes referred to as hierarchical topology, is similar to a collection of star networks arranged hierarchically [63].

3.6.5. Complete Graph

Each node is connected with every other node in the network. This topology is good for avoiding partitioning, because even if a link or a node fails, the other nodes remain connected in a single component.

3.7. Distributed Systems Communication Paradigms

There are multiple communication paradigms in distributed systems. These paradigms can be divided into three main types: Inter-process communication, Remote Invocation and Indirect communication [64].

3.7.1. Message Passing

Two distributed processes can use message passing to communicate with each other. This type of communication mainly uses two operations which are *send* and *receive*. A process can use the operation *send* to send a message which is essentially a sequence of bytes to another process. The destination process receives the message using the *receive* operation. If the communication is synchronous then the blocking variants of *send* and *receive* are used. The sending process is stopped whenever a send is made until the matching receive is made. A process blocks until a message is received whenever it issues a receive [53]. Otherwise, if the communication is asynchronous, the sending is non-blocking. This means that a process can send a message without waiting for the destination process to issue a *receive* operation. The destination process can either use blocking or non-blocking communication. If non-blocking variant of *receive* is used, then a the receiving process can carry on with its execution while its buffer is being filled with data [53].

3.7.2. Sockets

This belongs to the interprocess communication paradigm. Processes can communicate via sockets by first creating a socket, then associating a port number and an IP address to it. A socket can either use TCP or UDP protocol. It cannot use both protocols at the same time. When a process wants to send a message using a socket, it has to specify first the destination port and IP address.

The process who has a socket with that destination port and address will be the only process to receive that message [53].

3.7.3. Multicast

The *multicast* operation allows a process to send a message to multiple processes that consist a group. The message is sent to every member of that group. Among the implementations of multicasting is the IP Multicast. Using the IP multicast, a process can transfer one IP Packet to a multicast group. A multicast group has an IP address that belongs to class D [53].

3.7.4. Remote Procedure Call (RPC)

RPC is a form of communication which allows a process to call a procedure that is present on another process. This is done as if the procedure is in the local address space [53, 64].

3.7.5. Remote Method Invocation (RMI)

RMI is like RPC but in the context of object-oriented programming. A local object (calling object) can invoke a method on a remote object using the object's reference and the method's name [53, 64].

3.7.6. Publish-Subscribe Architecture

This is a form of indirect one-to-many communication. In this system, processes that publish messages and classify them into topics are called publishers. Processes that receive messages are called subscribers. Subscribers are interested in one or more topics. As a consequence, subscriber only receive messages regarding topics they are subscribed to [53].

3.7.7. Message Queues

Another type of indirect communication are message queues. They are also known as Message-oriented middleware. This system is for point-to-point communication. Producers create messages and send them to messages queues, while consumers remove messages from the queue to process them [53].

3.8. Message Passing Interface

MPI is a specification for the message passing library. Its main purpose is to give developers and computer manufacturers a single standard that is meant to be used in parallel systems. Message

passing allows processes that don't share the same address space to communicate with each other through sending and receiving messages [65].

3.8.1. MPI for Python

Developed since 2005, mpi4py which stands for (MPI for Python) is a well-known tool that offers Python bindings for MPI. Cython serves as a high-performance middleware between Python and C/C++ in MPI for Python [66].

3.9. Failure Models in Distributed Systems

There are multiple models of failure that can occur in distributed systems, in the following we list the main ones [54]:

- **Omission Failure:** This model concerns issues that occur when sending or receiving messages. For instance, a process may not send all the messages intended to be sent, or may not receive all the messages that it is supposed to receive.
- **Temporal Failure:** This concerns the time bound violations for sending and receiving messages or for other time bounded operations. Additionally, it also concerns drift rate violations.
- **Byzantine Failure:** Also known as arbitrary failure. A process can either add additional operations or remove certain operations randomly. Moreover, a process may use incorrect values when responding or may input wrong data. In certain cases, signatures can be used to authenticate the operations, thus identifying faulty nodes [53, 54].
- **Fail-Stop Model:** When a process stops running, and can be detected by other processes in the system.

3.10. Failure Detectors

Every process has a local failure detector module. The failure detector can either be reliable or unreliable. A failure detector can consider another process suspected or unsuspected. Suspected process is a process that is considered failed by the failure detector. While unsuspected process is considered to be a correct process (up and running). Reliable failure detectors can only exist in synchronous systems, because in these systems the delay time is bounded. On the other hand, asynchronous systems can only have unreliable failure detectors, since these systems are more

realistic, in a sense that the transmission delay time is unknown. When the delay time is unknown or variable, a failure detector can suspect another that's not really down but just too slow [53].

3.10.1. Failure Detector Properties and Classes

A failure detector has two properties which are: completeness and accuracy. Completeness indicates whether correct processes are aware of failed processes, and accuracy indicates that correct processes are not suspected by other correct processes. Completeness is set to prevent false negatives (crashed processes thought to be alive), and accuracy is set to prevent false positives (Correct processes thought to have failed). However, each property has two possible types based on its strength. Additionally, there is another variant of the accuracy property which is eventual accuracy. We present the different types below [54]:

- **Weak Completeness** states that every failed process is detected (suspected) by at least one correct process.
- **Strong Completeness** states that every failed process is detected by all correct processes.
- **Weak Accuracy** states that at least one correct process is not suspected by other correct processes.
- **Strong Accuracy** states that all correct processes are not suspected by the failure detectors of all other correct processes.
- **Eventual Weak Accuracy** After a certain time, eventually at least one correct process will not be suspected by all other correct processes.
- **Eventual Strong Accuracy** After a certain time, all correct processes will not be suspected by all correct processes.

Multiple classes of failure detectors can be formed based on combinations of the previously mentioned properties [54].

Table 4 Failure Detector Classes

		Accuracy			
		Weak	Strong	Eventual Weak	Eventual Strong
Completeness	Weak	W	ϑ	♦W	♦ϑ
	Strong	S	P (Perfect)	♦S	♦P

3.11. Failure Detectors Implementations

Common failure detector implementations usually use timeouts. The two most popular implementations are Heartbeat and Ping-ack.

3.11.1. Heartbeat

In this implementation the monitored process sends periodic messages every Δ time to another process indicating that it still alive [54]. If the monitoring process doesn't receive a heartbeat message from the other process after a timeout interval Δ has elapsed, then it adds the monitored process to its set of suspected processes. The timeout interval Δ is determined as the worst round-trip time of exchanging a message with that process. Thus, the timeout interval is different for every process.

3.11.2. Ping-Ack

A monitoring process sends a series of ping messages to the monitored process during a probing period. A ping message has a timeout interval Δ. When the monitored process receives a ping message it sends back an acknowledgement message to inform the monitoring process that its alive. If the monitoring process doesn't receive an ack message after sending the ping messages, then it considers the monitored process a failed process [67].

3.12. Leader Node Tasks

Coordinating the nodes within the system is the leader's responsibility. The system's leader is responsible for allocating resources, balancing the load on the different nodes, coordination of the consensus regarding replicated data and handling deadlock situations [52].

3.13. Leader Election Algorithms

A distributed election algorithm is an algorithm designed to select one node or one process among a set of processes to be the controlling node of that distributed system. Election algorithms must consider at least one election criterion. The most common criterion is the node's UID. Other election criteria include: CPU load, remaining battery life, available memory and network bandwidth.

3.13.1. Election Algorithm Correctness Conditions

A distributed election algorithm is valid if it satisfies the following properties [68]:

- **Termination:** The election algorithm has to stop in a finite time.
- **Agreement:** All nodes in the distributed system must be aware of the leader's UID.
- **Uniqueness:** There must be only one controller node in every connected component of the system.

3.14. Comparative Analysis of Distributed Election Algorithms

This section reviews multiple leader election algorithms designed for distributed systems. One of the newest works is a paper written by Sperling and Kulkarni [69], which proposed an election algorithm designed for asynchronous distributed systems. This algorithm presents a new voting procedure called shallow ranked voting, which allows the processes to vote for two processes. This algorithm guarantees the privacy of voters. The votes are encrypted using CKKS (Cheon, Kim, Kim and Song) method, which is a homomorphic encryption method. Meaning, that we can make approximate calculations on the encrypted data without having to decrypt them. This hides the identity of the voters as well as that of the top candidates which are most likely to win the election. If the primary choice doesn't win the election, the secondary choice gets the vote. This is also used for tie-breaking when two processes have the same majority of votes. When a process has the smallest number of votes it is eliminated and his voters' second choices take his votes. Since the ID is not used to break the tie, then this ensures privacy of the top candidates. Nevertheless, this work didn't consider multiple election criteria nor the recovery of the leader.

Jiang, F. et *al.* proposed a leader election approach based on node weight in the case of split brain [70], which is special case of partition when a network is divided into two partitions only. Election starts when the leader doesn't receive a heartbeat signal from the other servers or finds a node with higher weight. The weight indicates the service level of a node. The leader is the one who gets the majority of votes. The nodes with minimum weight will ensure the high availability of the system. This approach has less unavailable time than detection node-based and region leader-based approaches. The arbitration program also uses only 2% of the CPU full capacity. On the other hand, the authors didn't write a formal algorithm nor did they analyze the time and message complexity of their approach.

Luo, Y. *et al.* proposed an algorithm for the election of the block generator in the consensus mechanism of DPoS (Delegated Proof of Stake) [71]. They modified the Chang & Roberts ring algorithm by adding Stake Value. During the election this value will be multiplied by a random

value. But if the candidate who sent an election message has already been a leader before. Then, his Stake Value will be multiplied by zero, to ensure equality and avoid monopoly. Its message complexity is: 2n. Nonetheless, this algorithm doesn't have a tie break mechanism nor does it consider a backup leader.

Haddar proposed a scalable and energy aware k-leaders election algorithms designed for IoT wireless sensor networks [72]. Election starts from the initiators who broadcast an election message that helps to create a tree whose root is the initiator. The initiator receives the possible leaders of its neighbors and sends a Winner message to the k-highest weight nodes and a Looser message to the remaining nodes. The authors compared their algorithm to the other two top K-leader algorithms (WiLE and Top-K). Their algorithm gave better results in the number of messages and bytes transmitted in GRID and fully connected graph topologies with various network sizes. The residual energy was almost the same in the three algorithms. But the authors say their algorithm also consumes less power due to the reduced number of exchanged messages. On the other hand, they did not specify how the weights were calculated for the election criteria. Instead, they used random weights in the experiments.

Cahng and Lo proposed a consensus-based leader election algorithm for wireless Ad Hoc networks [73], which is based on Bully and Paxos algorithms. It has fault detection mechanism through finder nodes for detecting if the leader has left or crashed. The criteria used for election are residual battery power & node degree. The identifier is called "Vote" and is calculated based on these 2 criteria. The consensus consists in accepting only higher priority proposal and denying others. The proposed algorithm's message complexity is: O(n). As future work, they proposed to assure message integrity. Despite considering more than one election criterion, weights were not assigned.

Raychoudhury et *al.* proposed an algorithm that elects the K-highest weighted nodes as leaders in each connected component of mobile ad hoc networks [74]. The weight of a node indicates its available resources. There are 3 types of nodes: White nodes which are the normal nodes, Green nodes which are backups for the Red nodes, and Red nodes which are the highest weight neighbors. Red nodes are considered as local coordinator nodes in the sense that they help collecting the nodes weights and forward them to the highest weighted red node. The red node with the highest weight in a connected component is going to select the top-k nodes based on their

weight and elect them as leaders. This algorithm is fault tolerant and message efficient. It is also designed with topological changes in mind that could lead to network partitions. Moreover, it reserves a backup leader in case a red node crashes. However, this algorithm is based only on one election criterion, and doesn't consider the recovery of failed leader.

DRLEF (Distributed and Reliable Leader Election Framework) proposed in [75] by Elsakaan and Amroun consists of choosing an authentication server from a set of gateways. These gateways coordinate the network of wireless sensors. There are two types of nodes: Gateways and Sensor Nodes. There will be local elections in each area of the WSN (Wireless Sensor Network). The centrality here was measured by the deviation method. If the deviation exceeds a certain threshold, then the GW will not participate in the election. The gateway with maximum number of GWs as direct neighbors is called CGW (Central Gateway). Gateways send candidacy messages to the CGW. The candidate GWs are ranked based on the centrality criterion, and the best one is going to be elected as the leader. The other GWs enter hibernation mode, they are kept as backups in case the leader fails. This eliminates the need to redo the election process again. However, the DRLEF algorithm does not guarantee election in severe mobility circumstances. Additionally, it doesn't consider multiple election criteria nor the leader recovery.

Julian & Marian Jose used fuzzy analytic hierarchy process to elect a cluster head in ad hoc networks. The leader is elected based on his weight [76]. The node's weight is calculated on the basis of 7 criteria which are: node degree, transmission range, mobility, residual energy, trust value, status of the node, fairness of the node. Using this approach has several advantages. First, it eliminates inconsistencies in selection criteria. Secondly, the fuzzy variation of AHP removes duplicate weights. Additionally, it has better performance than standard WCA (Weighted Clustering Algorithm). On top of that, the nodes' mobility is taken into consideration. Finally, we obtain a ranking of the nodes from best to worst. In contrast, the authors didn't consider failure of the leader, recovery of the failed nodes and addition of new nodes.

Kadjouh et al. presented a dominating tree-based leader election algorithm (DoTRo) designed for smart cities IoT networks [77]. It uses the local minima finding algorithm (MinFind) to discover the local minimum values within the network. Afterwards, each local minimum is going to be the root that initiates a spanning tree. When two spanning trees get in contact, the tree with the smaller value continues the flooding process while the other one stops. Next, the local

minima will wait a maximum duration so their flooding processes can end. After this maximum time, if a local minimum node doesn't receive a message, then it becomes the leader. This algorithm is energy efficient, fault tolerant and reduces the number of sent and received messages when compared to MinFind. In contrast, this work didn't deal with security issues. Besides, no backup leader was considered and the election is solely based on one value.

Favier et al. introduced a novel centrality-based eventual leader election algorithm that works in dynamic networks [78]. The leader in this algorithm has to be in the center of the network. Each node has knowledge of its neighbors and the neighbors of its neighbors. A leader is elected in each component. When a node detects a change in its neighborhood, it updates its knowledge and emits its new view of the network. Nonetheless, there is a number of drawbacks for this algorithm. Firstly, if the nodes do not have the same knowledge, then they can choose different leaders. Secondly, this algorithm only takes into consideration the criterion of distance between the nodes. In addition, the size of the messages can be important since each node uses map structures and the message must respect the MTU (Maximum Transmission Unit) of the network packet. This requires the use of compression algorithms in order to reduce the size of the exchanged messages. It is possible to use collaborative calculations to calculate the centralities and thus save time. Moreover, this work didn't take into account multiple election criteria and leader recovery.

Biswas et al. proposed a new resource-based leader election algorithm [79], which selects the leader based on resource strength. Resource strength is calculated on the basis of 3 criteria: CPU, memory and remaining battery for mobile nodes. Each node has a queue with all the nodes present on the system. After that, the queue is sorted in descending order to place the node with the highest resource strength at the beginning of the queue and thus choosing it as the leader. This algorithm also takes into consideration the addition and removal of nodes from the system using update messages. However, the authors did not consider the security aspects for nodes joining the system. Moreover, the frequent addition and removal of nodes slows down the system. Plus, no formal weighting method was specified to determine the importance of each election criterion.

Another work by [68] A. Biswas et al. presented a novel failure rate and load based leader election algorithm (FRLLE) for bidirectional ring topology in synchronous distributed systems. This algorithm selects the node with minimum leader coefficient to be the leader. The leader

coefficient is computed based on several criteria which are: average CPU usage, memory usage, bandwidth usage and failure rate. The elected leader is the node with the minimum failure rate and minimum load, in order to ensure a stable leader with high performance. The proposed algorithm is faster and exchanges less messages than other classical ring-based algorithms. However, the authors didn't specify a formal method to fix the leader coefficient criteria weights and didn’t assign a backup for the leader.

Other classical and popular algorithms like LCR, Bully, HS and LeLann [80–83] rely solely on the UID to determine the leader and don’t reserve a backup leader.

Table 5. Comparison Between Existing Election Algorithms

Algorithm\criteria	Privacy	Tie-Break	Backup Leader	Partionning Tolerability	Homogeneous	Weighting Method	Optimized for Humans	Multi Criteria	Leader Recovery	Leader Failure Detection
LeLann [80]	No	No possible tie	No	No	Not Specified	No	No	No	No	Yes
LCR[81]	No	No possible tie	No	No	Not Specified	No	No	No	No	No
HS [82]	No	No	No	No	Not Specified	No	No	No	No	No
Bully [83]	No	No possible tie	No	No	Not specified	No	No	No	No	Yes
Weight Based[70]	No	No	No	2 partitions	Not Specified	No	No	No	No	Heartbeat
Fuzzy AHP [76]	No	No	No	No	Not Specified	Yes, Fuzzy Pair-wise comparison	No	Yes	No	No
DPoS [71]	No	No	No	No	Not Specified	No	No	No	No	No
Centrality-based [78]	No	Yes, Highest UID	No	Yes	Not Specified	No	No	No	No	Yes
Resource-based [79]	No	No	No	No	Not Specified	No	No	Yes	Yes	Yes
FRLLE [68]	No	Yes, Max UID	No	No	Yes	No	No	Yes	Yes	Yes
DotRo [77]	No	No	No	No	Not specified	No	No	No	No	Not specified
DRLEF [75]	No	No	Yes, Vice gateway	No	Not specified	No	No	No	No	Yes
Consensus [73]	No	No	No	No	Not specified	No	No	Yes	No	Yes, Finder nodes
Privacy-Preserved [69]	Yes	Yes, secondary votes	No	Yes	Not specified	No	No	No	No	No
Top-K [74]	No	Yes, Max UID	Yes	Yes	Not specified	No	No	No	No	Yes (Heartbeat)
SEALEA [72]	No	Yes, highest UID	Yes	No	Not specified	No	No	No	No	Not specified

Algorithm\criteria	Time complexity	Message Complexity	Topology	Synchronization	Fault tolerance	Structure	Mobility Support	Election Criteria	Year
LeLann [80]	O(n)	$O(n^2)$	Unidirectional Ring	Asynchronous	Yes	Static	No	UID	1977
LCR[81]	O(n)	$O(n^2)$	Unidirectional Ring	Asynchronous	Yes	Static	No	Max or Min UID	1979
HS [82]	O(n)	O(nlog(n))	Bidirectional Ring	Asynchronous	No	Static	No	Largest UID	1980
Bully [83]	O(1)	O(n^2)	Complete Mesh	Asynchronous	Yes	Static	No	Highest priority	1982
Weight Based[70]	Not mentioned	Not mentioned	Mesh	Not specified	Only leader	Static	No	Majority weighted votes	2020
Fuzzy AHP [76]	Not mentioned	Not mentioned	Mesh	Not specified	No	Static	Yes	7 criteria	2021
DPoS [71]	O(n)	Not mentioned	Unidirectional Ring	Not specified	Yes	static	No	Stake Value	2018
Centrality-based [78]	Not mentioned	Not mentioned	Arbitrary	Partially Synchronous	Yes	Dynamic	Yes	Closeness Centrality	2021
Resource-based [79]	Not mentioned	O(n)	Unidirectional Ring	Not specified	Yes	Dynamic	No	Resource strength value = CPU, Memory and Energy (Remaining Battery Life)	2018
FRLLE [68]	O(n)	$O(n^2)$	Bidirectional Ring	Synchronous	Yes	Dynamic	No	Smallest Leader Coefficient = Failure Rate + Load	2021
DotRo [77]	Not mentioned	kn(m+1)	Arbitrary	Not specified	Yes	Dynamic	No	Minimum Value	2020
DRLEF [75]	Not mentioned	Not mentioned	WSN	Synchronous	Yes	Dynamic	No	Deviation (Centrality)	2022
Consensus [73]	Not mentioned	O(n)	Partial Mesh/ WADHOC	Asynchronous	Yes	Dynamic	Yes	Remaining Battery & Node Degree	2012
Privacy-Preserved [69]	Not mentioned	O(diameter(G)*N*p)	Graph (Mesh)	Asynchronous	Yes	static	No	Most Votes	2023
Top-K [74]	Not mentioned	Not mentioned	Arbitrary	Asynchronous	Yes	Dynamic	Yes	Node Weight	2013
SEALEA [72]	Not mentioned	Not mentioned	Grid & Complete	Asynchronous	Leader only	Static	No	Node Weight	2022

3.15. Why Use Distributed Election Algorithms?

Distributed leader election algorithms are designed to solve the problem of choosing a unique node to be the leader of a connected network [72]. Similarly, the problematic treated in this research consist of choosing a single DM to be the facilitator of a group of DMs utilizing the GDSS. In both problematics there are multiple entities (nodes and DMs) and one controlling entity (leader/facilitator). Furthermore, the entities in both fields are geographically distant and connected via a network. Moreover, in both cases, they can communicate with each other via messages. Additionally, the same network protocols can be used in both cases. Figure 5 summarizes the similarities between the two problematics.

Election algorithms and the election of a facilitator have the same goal, which is to agree on a single leader. Both cases require considering certain criteria during the election. However, the criteria are mostly different since one side concerns machines while the other involves humans. Election algorithms have to be fault tolerant, the same as a facilitator needs a backup DM in case he loses connection with the other group members.

These similarities make the election algorithms seem like a potential solution to the problematic of electing a GDSS facilitator. However, due to the different nature of entities and context, an election algorithm designed for machines has to be modified in order to support human election.

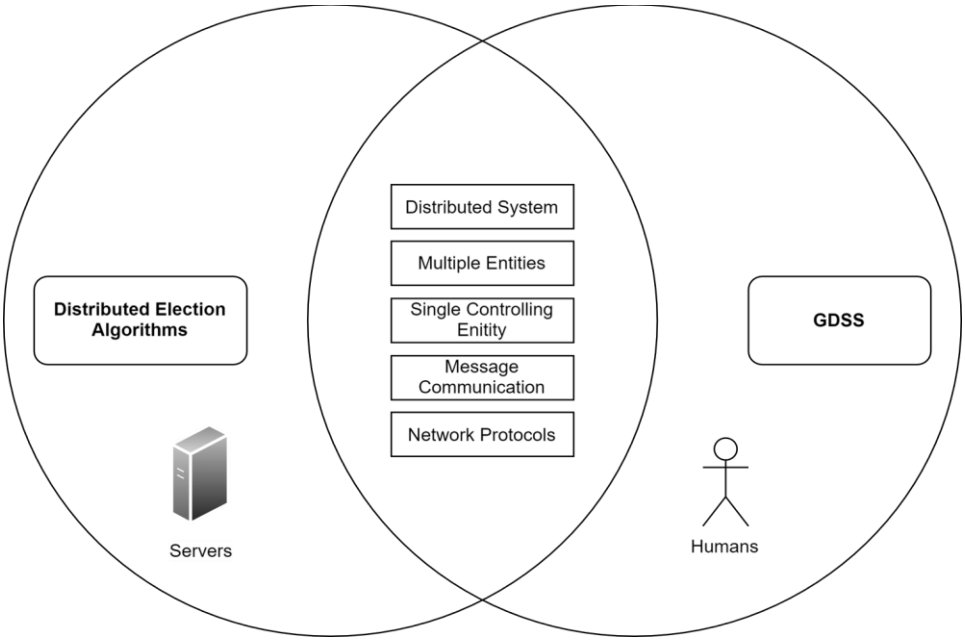

Figure 5 Venn Diagram Showing the Similarities Between Distributed Election Algorithms and GDSS

The election algorithms that exist in the literature [68–82] didn't include all the features required for the GDSS facilitator election in one single algorithm. Consequently, a potential solution is to combine each feature from each algorithm into a new election algorithm designed specifically for the problematic of GDSS facilitator election. A fulfilling algorithm should have a backup leader, and should consider multiple criteria which are relatable or applicable on humans. Additionally, it should use a proper weighting method to indicate the importance of each election criterion. Furthermore, it should consider the recovery of failed nodes, because realistically, DMs can lose their connection to the GDSS at any time, and recover their connection later on. Finally, in case a tie happens at the end of the election, a tie break mechanism is necessary in order to satisfy the uniqueness property.

3.16. System Model and Problem Definition

3.16.1. Problem Definition

Given a set D of DMs connected using a unidirectional ring network with |D| = n, one of the DMs has to take the role of the GDSS facilitator [84], while another one is reserved as a backup. The latter replaces the failed facilitator when he loses his connection with the network. The election

algorithm has to consider multiple election criteria and a formal weighting method is needed to distinguish the importance of each criterion. Additionally, the system has to be fault tolerant and must be capable of handling the potential recovery of failed nodes. Finally, the election algorithm must satisfy the 3 correctness conditions (uniqueness, termination and agreement) [68].

3.16.2. System Model

The network is a synchronous static unidirectional ring composed of n nodes. Each node represents a DM. There has to be at least 2 decision makers in the network (n>=2) [84]. Message passing is used for communication. Message delay between two nodes is based on the distance between the two DMs. If a DM loses his connection with the network, then his node is considered a failed node.

3.16.3. Assumptions

In order to simplify the implementation and analysis of the proposed algorithm, a set of assumptions was considered. First, each node has a unique identifier (0 < UID <= n) [68], which also indicates the order of joining the decision-making session (First DM to join the session has the smallest UID). Moreover, nodes are homogenous and not mobile. Furthermore, the maximum number of nodes n is fixed prior to starting the election. Additionally, each node knows the UID of the previous and next adjacent nodes [74]. Another assumption is that each node receives a heartbeat from the previous node and sends a periodic heartbeat message to the next adjacent node in order to detect if the previous node fails [74]. In addition, the communication direction is clock-wise. On top of that, the DM's performance is still used in calculating the election criteria weights even if the DM loses his connection, because this gives more input data to the objective weighting method. Furthermore, there are no hops between each pair of adjacent nodes. Moreover, no new DMs are added to the network other than the preselected n DMs. Finally, the ring topology is only used for the election and failure tolerance, it is not used for the actual group meeting communication.

3.16.4. Notations

There are seven types of messages in this algorithm. The description of each type is presented below:

- **Message(uid, value[j], j, init_uid):** General message object containing the UID of the message creator, his value of the jth criterion and the criterion's index.

- **InitiationMsg(init_uid, value[1], 1):** Initiation message created by the initiator which includes the UID of the initiator node. In addition to its value of the least important criterion (1^{st} criterion) and the first round number.
- **LeaderMsg(leader_uid, backup_uid):** Message announcing the new elected leader and the backup leader to all other nodes.
- **FailureMsg(uid):** Message announcing the failure of a node by sending its UID.
- **LeaderFailureMsg(new_leader_uid):** Message announcing the leader failure, and informing other nodes that the backup has become the new leader.
- **RecoveryMsg(uid):** Message announcing the recovery of a previously failed DM by sending his UID.
- **LeaderRecoveryMsg(failed_leader_uid):** Message announcing the recovery of the previously failed leader, thus updating the nodes with his new state, and informing them that the leader has become a backup.
- **InitiatorRecovery(rec_init_uid, value[j], j):** Message indicating the recovery of the previously failed initiator. This message includes the UID of the recovered initiator, its value of the j^{th} criterion in which it failed and the round j in which it failed.
- **TieMessage(ties[k], uid, $value_a$[m]):** Message containing the UIDs of the tied DMs holding the maximum score, the UID of the message creator, and their values of the jth criterion.

Variables used within GFEA are detailed below:

- **state[n]:** List containing the combination of role (DM, Initiator, Leader) and state (Failed or Connected) of each node in the network. Each node has its own local *state* list which gets updated when receiving messages. This allows each node to be aware of the current state of every other node in the same network.
- **value[m]:** List containing the evaluation of a DM in each election criterion.

- **criterion_type[m]:** List containing the type of each criterion, either a beneficial (maximization) criterion or non-beneficial (minimization) criterion.
- **r:** Received message.
- **score[n]:** List containing the algorithm score for each DM.
- **best_dms:** List of DMs with the maximum score.
- **ranking[n]:** List containing the ranking of DMs based on the final score.

Procedures integrated within the proposed algorithm are as follows:

- **send(Message msg):** Procedure for passing a message to the next adjacent node.
- **broadcast(Message msg):** Procedure implying that each node should keep forwarding the contained message to the next node, until it reaches its original node.

3.17. Proposed Election Algorithm

In this work, a new election algorithm called GFEA (GDSS Facilitator Election Algorithm) is proposed. This algorithm is inspired by the FRLLE election algorithm [68] and MCDM (multi-criteria decision-making) methods. This algorithm is optimized for the problematic of electing a GDSS facilitator. It uses the ring topology with unidirectional communication channels. The nodes communicate via message passing. Each DM has a unique identifier that indicates the order in which the DM has joined the decision-making session. Furthermore, each node is in one of seven states: Initiator, DM, leader, backup, failed leader, failed initiator or failed DM. Figure 6 illustrates the proposed leader election algorithm in a concise manner.

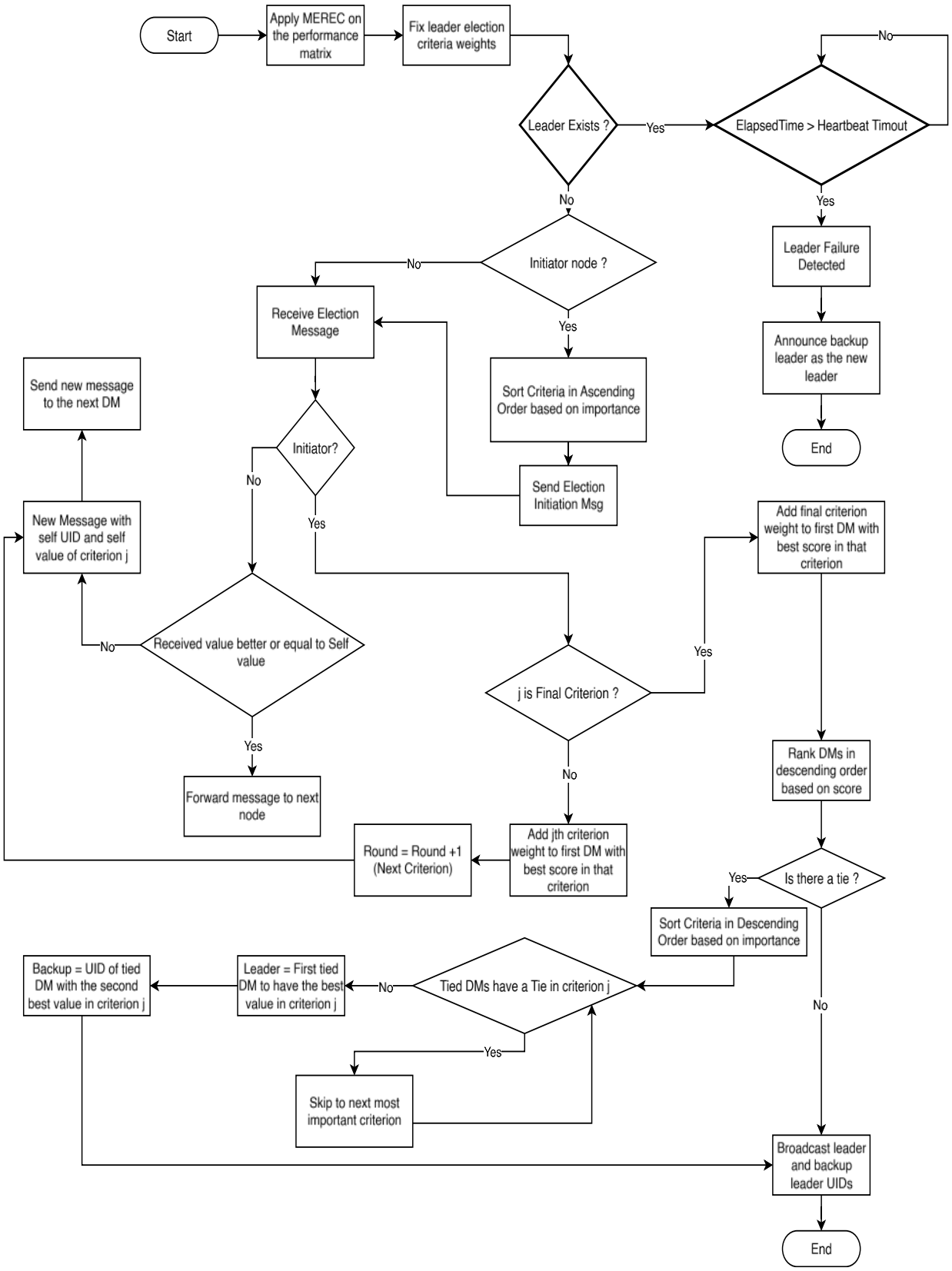

Figure 6. Flowchart of the Proposed GFEA Algorithm

3.17.1. Weighting Method

Election criteria are not necessarily equally important; thus, a formal weighting method is needed to fix the election criteria weights. There are three types of weighting methods in the literature [85]: objective, subjective and combined. When using objective weighting methods, the DMs subjective preferences are not considered [85]. In this work, we opted for objective methods to keep the election algorithm formal and unbiased. Among the popular objective methods are: the entropy, mean weight, CRITIC, standard deviation and statistical variance methods [86]. There are newer methods like CILOS, IDOCRIW and MEREC. In MEREC (Method of Removal Effects of Criteria) the effect of removing a criterion on the evaluation of alternatives is used to fix the criteria weights [87]. The MEREC weighting method was used in this approach in order to fix the election criteria weights.

Among the reasons MEREC was chosen instead of other objective methods is that it is easy to understand and use, since it simply involves applying a set of formulas to the values which are already present in the performance matrix. In addition, it has a strong mathematical foundation [88]. However, this doesn't mean that other objective weighting methods aren't valid. It is possible to use other methods with the proposed algorithm.

This method takes the performance matrix and the criteria types as input. The performance matrix should only contain positive non-zero values. The values corresponding to non-beneficial criteria are normalized according to formula (25). While in the case of beneficial criteria the values are normalized by applying formula (26) [87].

$$x'_{ij} = \frac{x_{ij}}{\max_{i}(x_{ij})} \tag{25}$$

$$x'_{ij} = \frac{\min_{i}(x_{ij})}{x_{ij}} \tag{26}$$

Where x_{ij} is the performance of alternative i according to criterion j, and x'_{ij} is the normalized performance of alternative i according to criterion j.

Next, the general performance GP_i of each alternative i is calculated based on formula (27) [87].

$$GP_i = \ln\left(1 + \left(\frac{1}{m}\sum_j \left|\ln\left(x'_{ij}\right)\right|\right)\right) \tag{27}$$

The following step consist in constructing m sets that contain the performance of alternatives when removing each criterion j. This performance is calculated using formula (28) [87], where k represents all criteria except criterion j.

$$GP'_{ij} = \ln\left(1 + \left(\frac{1}{m}\sum_{k,k\neq j} \left|\ln(x'_{ik})\right|\right)\right) \tag{28}$$

In the succeeding step, the removal effect RE_j of each criterion is calculated using formula (29) [87].

$$RE_j = \sum_{i=1}^{n} \left|GP'_{ij} - GP_i\right| \tag{29}$$

Finally, the criteria weights are obtained from the removal effects using formula (30) [87].

$$w_j = \frac{RE_j}{\sum_{k=1}^{m} RE_k} \tag{30}$$

Where RE_j is the Removal Effect of criterion j, and w_j is the weight of criterion j.

3.17.2. Election Algorithm Details

Now we are going to detail the different phases and how GFEA will handle different types of messages during its execution. Additionally, we will present the new tie-breaking mechanism.

Initiation Phase: This phase concerns the detection of the failed leader and the composition of the first election message. If a facilitator already exists and the node which is next to the leader doesn't receive a heartbeat message from him within a period of time called heartbeat

timeout [89], it announces that the leader has failed and the backup leader becomes the new leader. However, if there is no leader in the network, then the first DM to join the session (UID = 1) becomes the initiator of the facilitator election. The first-ever election only starts when all DMs have joined the session. The initiator starts by sorting the election criteria based on their importance in ascending order (From least important to most important). Next, he creates and sends the election initiation message containing his UID and its value of the 1st least important criterion. The initiation phase is detailed in Algorithm 1.

Algorithm 1. Leader failure detection and election initiation

```
1:  If(leader exists) Then
2:    If(previous node is the leader) Then
3:      If(no heartbeat received and elapsed_time > heartbeatTimeout) Then
4:        Leader = backup
5:        failedLeader_msg = new LeaderFailureMsg(backup);
6:        send(failedLeader_msg); // Announce the failure of leader and UID of new leader
7:      Else continue receiving heartbeat messages;
8:  Else If(self.uid == 1) Then
9:      initiatorUID = 1; // Initiator is first DM to join the group session.
10:     Sort criteria from least important to most important;
11:     i_msg = InitiationMsg(1, value1[1], 1);
12:     send(i_msg);
```

Scoring Phase: In this phase (Algorithm 2), when a node receives an election message, it checks its value in criterion j. If the received value is better than its own (greater in the case of maximization criteria and smaller in minimization criteria), and then forwards the message to the next node. But if the received value is worse than its own value, it creates a new message containing its UID and its value of criterion j. Then it sends the message to the next node. Once the message reaches the initiator node, it sends a new message for the next least important criterion j+1. The same process is repeated for all election criteria. If more than one DM has the best value in a criterion, then only the DM with the smallest UID gets the criterion weight added to his score.

At the end of the final round, if multiple DMs have the maximum score, then a tie-break mechanism is used to return a single facilitator (see Algorithm 2). Afterward, the initiator node sends the chosen leader and the backup leader UIDs in a broadcast message to inform all other DMs who is the current facilitator and who should replace him if he fails.

Algorithm 2. Scoring Phase

Input: pm, criteria_weights, criterion_type.
Output: leader, backup.
n: number of decision makers;
m: number of criteria;
r: received message;

```
1:  If(this node is the initiator node) Then
2:      If(this is the final round) Then
3:          score[r.uid] = score[r.uid] + criteria_weights[j];
4:          best_dms = UIDs of DMs with max score;
5:          If(Only one DM has the max score)Then
6:              leader = best_dms;
7:              ranked = sort DMs from highest score to smallest score;
8:              backup = ranked[2];
9:              leader_msg = new LeaderMsg(leader, backup);
10:             broadcast(leader_msg);
11:         Else tie_break(best_dms);
12:     Else Add the weight of the j^th criterion to the score of the received UID;
13:         j = j+1;
14:         msg = new Message(self.uid, value[j], j, init_uid);
15:         send(msg);
16: Else If(criterion_type[j]==”max”) Then
17:         If(r.value[j] < self.value[j]) Then
18:             msg = new Message(self.uid, self.value[j], j, init_uid);
19:             send(msg);
20:         Else  send(r);
```

```
21: Else If(criterion_type[j]==”min”) Then
22:         If(r.value[j] > self.value[j]) Then
23:             msg = new Message(self.uid, self.value[j] , j, init_uid);
24:             send(msg);
25:         Else  send(r);
```

Tie Break: In case two or more DMs have the same final score, then the initiator uses Algorithm 3 to break the tie and return one DM as the facilitator. First, the initiator starts by sorting the criteria in descending order based on their weights. Starting from the most important criterion (biggest weight) to the least important criterion (smallest weight) the first DM to have a better value than all other tied DMs in a criterion j is declared leader. The second-best value in the same criterion j is selected as the backup leader. If multiple DMs have the best value in a criterion j, then the algorithm continues to the next most important criterion and checks again. It is impossible to have a tie in all criteria among the tied DMs. The tie break mechanism is detailed in Algorithm 3.

Algorithm 3. Tie Break Between Multiple DMs

```
    ties = [a, b, …, k]; // k tied nodes
1:  sortDescending(criteria, criteria_weight);
2:  j = 1;
3:  If(this is the initiator node)Then
4:    While(j <= m) Do
5:      If(initiator is one of the tied DMs)Then
6:        tie_break_msg = new TieMessage(ties, self.uid, self.value[j]);
7:      Else
8:        tie_break_msg = new TieMessage(ties, self.uid, Null);
9:        send(tie_break_msg);
10: Else If(this is a DM and criterion_type[j]==”max”)
11:     If (this is a tied DM and r.value[j] < self.value[j] or r.value[j] == Null) Then
12:       self.backup = r.uid;
13:       self.leader = self.uid;
```

```
14:            found = True;
15:            tie_break_msg = new TieMessage(ties, self.uid,   self.value[j], found);
16:            send(tie_break_msg);
17:        Else send(r);
18:  Else If(this is a DM and criterion_type[j]==”min”)
19:        If (this is a tied DM and r.value[j] > self.value[j] or r.value[j] == Null) Then
20:            self.backup = r.uid;
21:            self.leader = self.uid;
22:            found = True;
23:            tie_break_msg = new TieMessage(ties, self.uid, self.value[j], found);
24:            send(tie_break_msg);
25:        Else send(r);
26:  If(Initiator receives TieMessage and found == True)
27:            leader = r.leader;
28:            backup = r.backup;
29:            leader_msg = new LeaderMsg(leader, backup);
30:            broadcast(leader_msg);
31:            BreakLoop;
32:  Else If(Initiator receives TieMessage and found == False)
33:            j = j + 1; // Continue to next criterion
34:           Go to step 5;
```

Handling Leader Announcement Message: When a DM node receives the leader announcement message, it updates its state list with the new leader and backup leader UIDs and passes the received message to the next node (see Algorithm 4). On the other hand, if the initiator node receives this type of message, it changes its state to DM. Except if it's already a leader or a backup leader, in that case, it doesn’t change its state.

Algorithm 4. Handling of Leader Announcement Message

```
Input: r: received leader announcement message.
If(Initiator receives leader announcement message) Then
    discard(r);
    If(Initiator is not the new leader or new backup) Then
            state[self.uid] = DM;
Else                                        // Other Nodes
    leader_exist = True;
    state[r.leader] = "Leader";
    state[r.backup] = "Backup";
    send(r);
```

Fault Tolerance: In case the facilitator gets disconnected from the network, the node next to the failed leader which detected his failure, sends a leader failure message containing the UID of the backup as the new leader (see Algorithm 5). The backup leader is the DM with the second-best score. Having a backup eliminates the time and resource cost of running the election another time when the leader fails [72]. If a DM loses his connection with the network, then, his state is changed to "Failed Node", and his previous node gets connected directly to his next node in order to keep the ring topology intact. This network doesn't support partitioning as it will always try to keep its logical ring topology intact.

Algorithm 5. Handling Leader and DM Failure

```
If(Time with no heartbeat from leader > Heartbeat_threshold) Then
    leader = backup
    leader_msg = new LeaderFailureMsg(leader);
    send(leader_msg);
If(DM node fails) Then
    His previous node becomes adjacent to his next node;
```

Whenever the initiator fails, the node next to it becomes the new initiator. After that, the new initiator starts a new election from the beginning (see Algorithm 6). Because the failed initiator had the list containing the scores. Hence, the new initiator creates the election initiation message starting with the least important criterion, and sends it to the next node.

Algorithm 6. Initiator Failure

```
1: If(Initiator node fails) Then
2:     Node next to failed initiator becomes the new initiator;
3:     Discard current election messages;
4:     i_msg = New InitiationMsg(new_initiator, value[1], new_initiator);
5:     send(i_msg); \\ New initiator sends the new initiation message.
```

Failure Recovery: When a previously failed node joins the network, it restores its previous status (DM or Leader). As a result, if the backup already replaced the leader and the previously failed leader gets reconnected to the session, then he becomes a leader again, and the backup becomes a backup again (see Algorithm 7). Next, the recovered leader sends a recovery message to the other nodes. Furthermore, if a DM recovers before the final round (criterion m) is finished, he is still considered a candidate. Because the criteria are sorted in ascending order, he can compensate for the previous rounds (less important criteria) by scoring in the more important criteria (later rounds). However, if he recovers after all the rounds have gone through, then he isn't considered a candidate, and will be removed from the ranking, as the initiator is already in the leader announcement phase and isn't aware of the DM recovery until the recovery message reaches him.

Algorithm 7. Recovery of failed nodes

```
1: If(This is the recovering leader) Then
2:         backup = leader;
3:         leader = previously failed leader UID;
4:         rec_msg = new LeaderRecoveryMsg(leader, backup);
5:         broadcast(rec_msg);
```

```
6:  Else If(This is the recovering initiator and current round == failure round) Then
7:          Recovered initiator restores his initiator state;
8:          Current initiator becomes a DM;
9:          continue_election();
10: Else If(This is the recovering initiator and current round ≠ failure round) Then
11:         recovered initiator becomes a DM;
12:         New initiator continues current election;
13: Else                                                    // This is a DM
14:         rec_msg = new RecoveryMsg(self.uid);
15:         broadcast(rec_msg);
```

If the failed initiator recovers during the same round in which it failed, then it restores its state as the initiator and the election continues. But if at least one round has passed since his failure, then it becomes a DM and the current initiator (node next to the recovered initiator) continues the current election.

When the leader receives his own recovery message, then he discards it [68]. However, if a DM node receives a leader recovery message, he updates his state list with the new state of the recovered leader and forwards the received message to the next adjacent node (see Algorithm 8).

Algorithm 8. Handling Leader Recovery Messages

```
    r: Leader recovery messages;
1:  If(This is the new leader) Then
2:          discard(r);
3:  Else If(This is the new backup) Then
4:          state[self.uid] = "Backup";
5:          state[r.leader] = "Leader";
6:          send(r);
7:  Else    state[r.leader] = "Leader";
8:          state[r.backup] = "Backup";
9:          send(r);
```

3.17.3. Election Algorithm Correctness

- **Uniqueness:** The DM with the highest score will be elected as the GDSS facilitator. However, if there are multiple DMs having the same max score, a tie-breaking mechanism is used. Starting from the most important criterion to the least important one, the first candidate to have an advantage in a criterion j will be selected as the leader. Which means that there will always be one single leader in the system.
- **Termination:** The algorithm takes m×n + n time steps when there is no tie. In contrast, in the worst-case scenario it takes $m(n^2/2 + n) + 4$ time steps. Consequently, the algorithm does terminate in a finite time.
- **Agreement:** At the end of the algorithm or after the tie breaking mechanism ends, an announcement message containing the elected leader and backup leader UIDs is sent to all DMs. Thus, every DM in the group is aware of the new elected facilitator.

3.17.4. Complexity Analysis

In this section, the proposed algorithm GFEA is analyzed based on the number of time steps required and the total number of exchanged messages in both best and worst cases.

3.17.4.1. Time Complexity

Time complexity is determined based on the number of time of steps that the election algorithm takes to complete.

- **Best case:** When the initiator doesn't fail and there is no tie in the scores, the proposed algorithm takes m×n+ n time steps to end. As a result, the best time complexity for the proposed GFEA algorithm is: O(m×n). Where m is the number of election criteria and n is the number of DMs.
- **Worst case:** The worst time complexity is when there are n-2 initiators fail after initiating all 12 rounds and before receiving the final round message. In addition to a tie in the score and the tied DMs have a tie in m-1 criteria. An additional n×m time steps are required to break the tie (n tied DMs). So, the total number of time steps is: $m\left((n-2)\times\left(\frac{n+2}{2}\right)\right) + n\times(m-1)+n = m\left(\frac{n^2-4}{2}+n\right)$. Thus, the worst time complexity is: O(m×n²).

3.17.4.2. Message Complexity

Message complexity is obtained based on the number of sent and received messages between all the nodes during the election.

- **Best case:** The best-case scenario is when there is no tie in the final score and the initiator doesn't fail. Starting from the initiator node, each node sends a message in each round (Criterion). In addition to the leader announcement message. Consequently, the algorithm exchanges 2(m×n + n) messages. Therefore, the message complexity of GFEA in the best case is O(m×n).
- **Worst case:** The worst theoretical case, is when there are n-2 initiators fail after initiating all 12 rounds and before receiving the final round message. In addition to a tie in the score of n DMs, and the tied DMs have the same value in m-1 criteria. As a result, the proposed algorithm exchanges $2m\left((n-2)\times\left(\frac{n+2}{2}\right)+nm+n\right)=m\left(\frac{2n^2-4}{2}+2nm+n\right)$ messages. Therefore, the message complexity of GFEA in the worst case is O(m×n²). O(m×n²) was selected instead of O(nm²) because a distributed system can scale to include hundreds of nodes, whereas the number of criteria is usually limited compared to the number of nodes. An advantage of this algorithm is that there is only one initiator at a time. Thus, avoiding multiple nodes initiating the election at the same time.

3.18. Functionality-based Comparison with GFEA

Table 6 compares the proposed GFEA algorithm with modern election algorithms that exist in the literature. The comparison is based of algorithm functionalities. The correct sign ✔ indicates that the algorithm integrates the functionality.

Table 6 Comparison of the Proposed Election Algorithm with Existing Algorithms

Functionality / Algorithm	Multiple criteria	Backup Leader	Criteria Weighting Method	Tie Break	Optimized for Humans	Fault Tolerance	Initiator Failure & Recovery
GFEA	✔	✔	✔	✔	✔	✔	✔
FRLLE [68]	✔			✔		✔	
Top-K [74]		✔		✔		✔	
SEALEA [72]		✔		✔			
FAHP [76]	✔		✔				
DRLEF [75]		✔				✔	
Privacy-Preserved [69]				✔		✔	

These algorithms are applicable on human election. However, they are not optimized for humans. Because the criteria considered in these algorithms are not human related. Which would result in choosing a facilitator solely based on his machine's performance, while ignoring experience and security criteria, which is not suitable for the facilitator role. The privacy-preserving election algorithm [69] uses votes to determine a leader. Which is a common approach used in human elections.

The GFEA algorithm showcases the strengths and weaknesses of each DM. Because, contrary to algorithms like FRLLE and Resource-based [68, 79], it doesn't combine all criteria into one single value before the election, but rather builds the score progressively during the

election. This is because the score can't be calculated solely based on the individual list, instead, the best value across all nodes is needed to determine the best node in each criterion, and then its weight is added to the score. This represents an advantage to GFEA, as it uses a global view instead of a local one. Additionally, no fictious reference values are used for defining what's the best value in each criterion. As a replacement, the actual best values present within the performance matrix are used as a reference. Another plus that puts GFEA a in more favorable position is that instead of using random weights or weights fixed by human judgement like in FRLLE and Resource-based [68, 79], it uses MEREC [87]. Which is a formal objective weighting method. The weights affect the final score of nodes; thus, they can alter the elected leader and backup leader. Furthermore, in contrast to the existing election algorithms, the proposed algorithm considers the failure and recovery of the initiator node. This aspect is neglected in most of the related works. Moreover, the tie break mechanism integrated within GFEA considers the election criteria instead of just relying on the UID. Plus, it gives priority to the most important criteria. This ensures that the proposed algorithm always elects the most suitable node for the leader position, by satisfying the election criteria.

3.19. Conclusion

This chapter introduces a new distributed leader election algorithm designed specifically for electing a human GDSS facilitator. The system considered is a fault tolerant unidirectional ring synchronous system, in which each node represents a DM, and they communicate via message passing. The proposed algorithm integrates multiple election criteria falling into security, experience and network performance categories. These criteria are weighted using the MEREC method. Furthermore, GFEA reserves a backup leader to replace the facilitator in case the connection fails between the facilitator and the GDSS. This saves time and avoids halting the group decision-making session. Additionally, this algorithm doesn't use the UID to break a possible tie. Instead, it uses a new tie breaking mechanism which searches for the DM who has an advantage in the most important election criteria. Moreover, the failure and recovery of both the initiator and leader are handled in an efficient way. No existing election algorithm has integrated all these features together. Finally, GFEA is flexible and can be applied on the classical leader election problem in distributed systems by changing the election criteria to machine related criteria, such as CPU and memory load.

The following chapter presents the case study that we used to test and evaluate both the multi-criteria approach and GFEA. Furthermore, a new election tool is proposed and integrated into the collaborative e-maintenance process.

Chapter 4

4.Case Study: Collaborative E-maintenance

4.1. Introduction

In this chapter, we will present the field of collaborative e-maintenance on which we tested and evaluated both the multi-criteria approach and the distributed election approach. First, we explain the steps involved in DELPHI studies, which can be used to elicit the election criteria. Next, we test the previously discussed MCDA methods on the first performance matrix. The obtained results and charts are analyzed thoroughly, leading us to a comparison between the application of every method based on different metrics related to the context of electing a facilitator. Additionally, we demonstrate the use of a new software tool called GFET designed to automate and facilitate the election of a GDSS facilitator. Then a model is proposed to integrate GFET into collaborative e-maintenance. Secondly, the proposed distributed election algorithm GFEA is evaluated in the same field.

4.2. Maintenance Types

4.2.1. Preventive Maintenance

By scheduling regular inspections and replacements, preventative maintenance aims to minimize unscheduled downtime. Lubrication, adjustments, oil changes, and sophisticated diagnostics are examples of regularly scheduled tasks [90]. This type of maintenance also includes breakdown and failure simulations. Examples of theses simulations in the context of software security maintenance include hiring a hacker to attack a company’s system.

4.2.2. Corrective Maintenance

Corrective maintenance is the process of fixing a machine's breakdown after it has happened. This method increases the chance of equipment unavailability (downtime) [90]. Among the negative aspects of this type of maintenance is that unplanned downtime cost money and time. This type should be avoided for critical equipment or for equipment that is hard to fix [90].

4.2.3. Predictive Maintenance

Predictive maintenance involves doing repairs before an anticipated breakdown happens. This method's main goal is to forecast a machine's health by using known characteristics or repeated study. Predictive maintenance is a kind of condition-based maintenance in which we forecast performance in the future using both historical and present data. Both scheduled and unplanned downtime are decreased when this strategy is used [90].

4.3. Collaborative E-maintenance

An industrial company's ability to succeed or fail depends on the priority it gives to maintenance. The maintenance burden accounts for a significant portion of the total operating costs of the production. Maintenance cost can go up to 40% of production cost. Its cost has increased by 400 million dollars between 1979 and 1989 [91].

With the new advancements in information and communication technologies the industry has seen a revolution and an upgrade to become Industry 4.0. Which integrates new technologies that facilitate the automation process of manufacturing and production [92]. Among those technologies are the sensors that can capture specific quantities like liquid temperature and pressure, and then send this information as signals to the controller. The controller then converts the signals and process the information. Based on the obtained information he sends signals that activate the actuators. These actuators do physical actions. Such as moving a robotic arm, control a motor or a valve. The maintenance of these modern industrial production machines requires the collaboration of multiple experts, each one coming from a different field. The collaboration in maintenance offers multiple repair approaches and multiple opinions from different experts. This gives more options and flexibility when fixing a breakdown. It also enables learning through the sharing and exchange of knowledge, insight and information between the experts. Other important benefits of collaborative maintenance are efficiency and minimizing repair time [93].

Emerging countries need the technologies, equipment and industrial systems required to extract oil and process it into usable gas and electricity. Therefore, they buy these industrial systems from foreign suppliers. Since these systems are not fabricated by local engineers, it makes sense that the supplier's experts and engineers have comprehensive knowledge of how their equipment work and how to maintain and repair them.

There are several reasons to call for remote experts. First, the interactions of the local technicians with the foreign experts allow them to level up their skills and technical knowledge. Secondly, they gain access to remote support and expertise [91]. Moreover, most companies' managers lower the priority of maintenance. Furthermore, the qualified local experts retired and didn't provide training nor knowledge transfer to the newer engineers. In addition, the isolated and remote nature of oil fields like the desert, is a big challenge for emerging countries [93]. On top of that, we cannot neglect the safety of the experts, as some maintenance processes require working with radioactive components or with high voltage electricity [1].

The use of new Information and Communication Technologies allows for distant experts to be involved in the maintenance process without having to move to the breakdown site. This eliminates the cost and time of transportation. Moreover, during the pandemic, this may be the only way to repair broken machines. As some countries forbid to leave or enter the country during the peak of pandemics. Furthermore, the foreign experts may not be available to travel to a remote location. Consequently, the collaborative e-maintenance shortens the maintenance time and thus decreases the machine's idle time [93]. This is very important since the breakdown could stop a service which will affect the client's quality of service, prevent access to a service or delay a product's manufacturing time.

Collaborative E-maintenance is the software infrastructure which connects geographically distant experts, technicians and subsystems. This infrastructure is established on a reachable network. Strong collaboration between various staff, organizational domains, and businesses is made possible by the collaborative e-maintenance software. A Groupware will allow for multiple geographically distant experts to collaborate on finding the optimal solution and repair approach for fixing the breakdown [93]. Since the GDSS includes a Groupware, then we can use a GDSS on collaborative e-maintenance process. On the other hand, the collaboration of several experts on

one objective requires coordination. This coordination is assured by the facilitator (coordinator) of the experts group.

4.4. Facilitator Missions in Collaborative E-maintenance

The coordinator's missions in the e-maintenance group include assigning a sequence number (order number) to each expert. This number is used to assign an expert to a group that is handling a certain breakdown. The sequence number assignment is used in Figures 5 and 6. The process of introducing additional experts to groups includes granting a sequence number. The group's coordinator also has the option of adding a new expert either by invitation or by request (see Figures 5 and 6). He can also transfer an idle expert to another group of experts. This is shown in Figure 8. The facilitator also controls the mutual exclusion of maintenance resources by determining if the critical section is unoccupied or not (see Figure 7). If it is, he permits the experts to access it; if not, he places their requests in a queue based on the request's importance. Additionally, he has the ability to split up the group if he notices that there are too many breakdowns (see Figure 9), in order to satisfy the demand for additional experts [1]. Figure 4 Shows a brief representation of the coordinator missions in collaborative E-maintenance process. While Figures 5 to 9 illustrate the BPMN model of the different collaborative E-maintenance sub-processes in which the coordinator is involved.

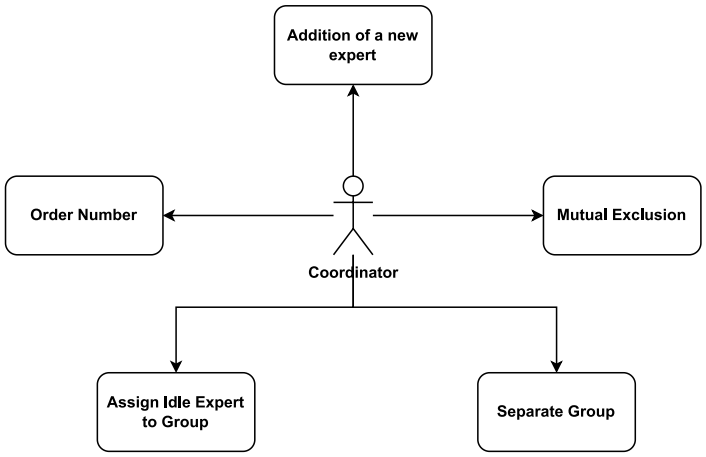

Figure. 7. Collaborative E-maintenance Coordinator Missions [1]

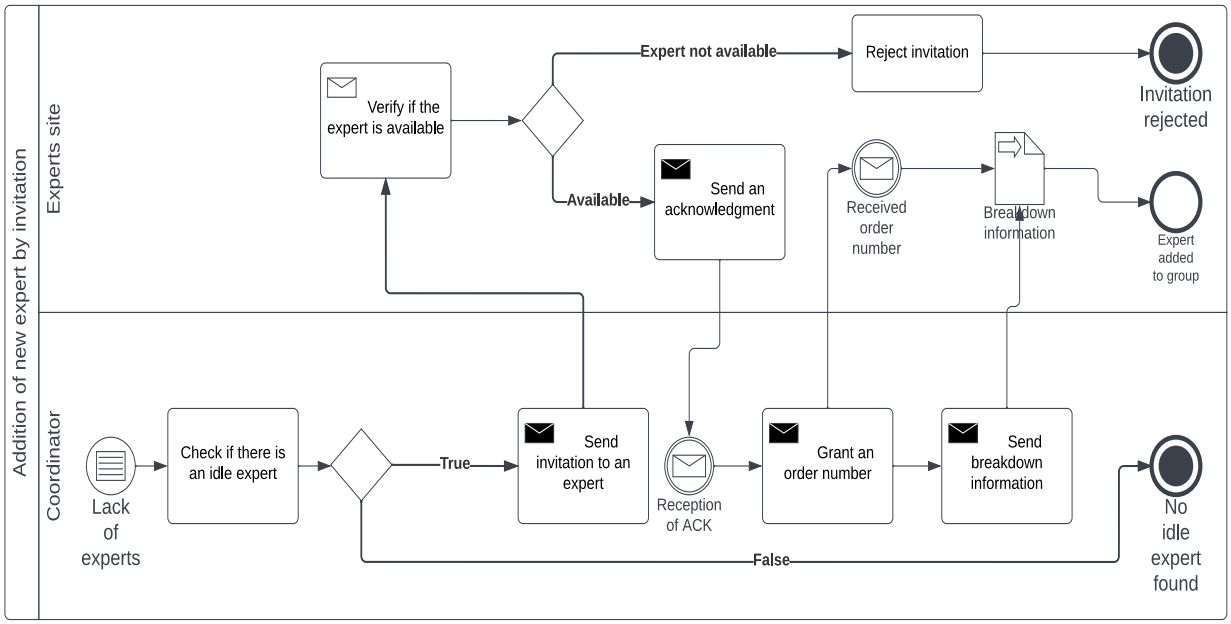

Figure. 8. BPMN Model of the Process of Addition of a New Expert by Invitation

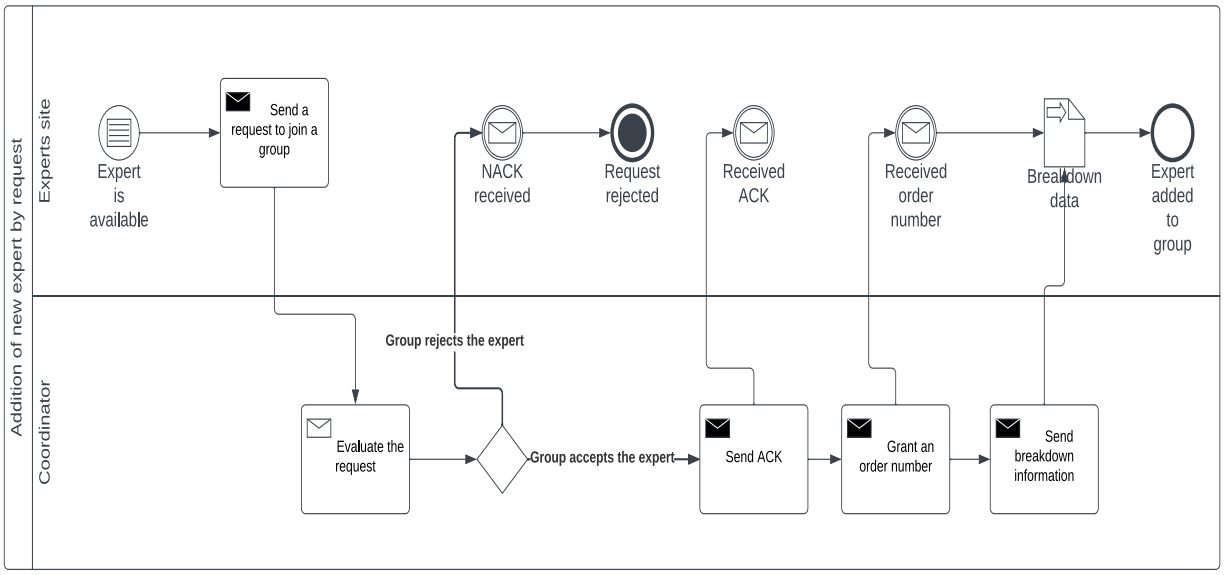

Figure. 9. The BPMN Model of the Process of Addition of New Expert by Request

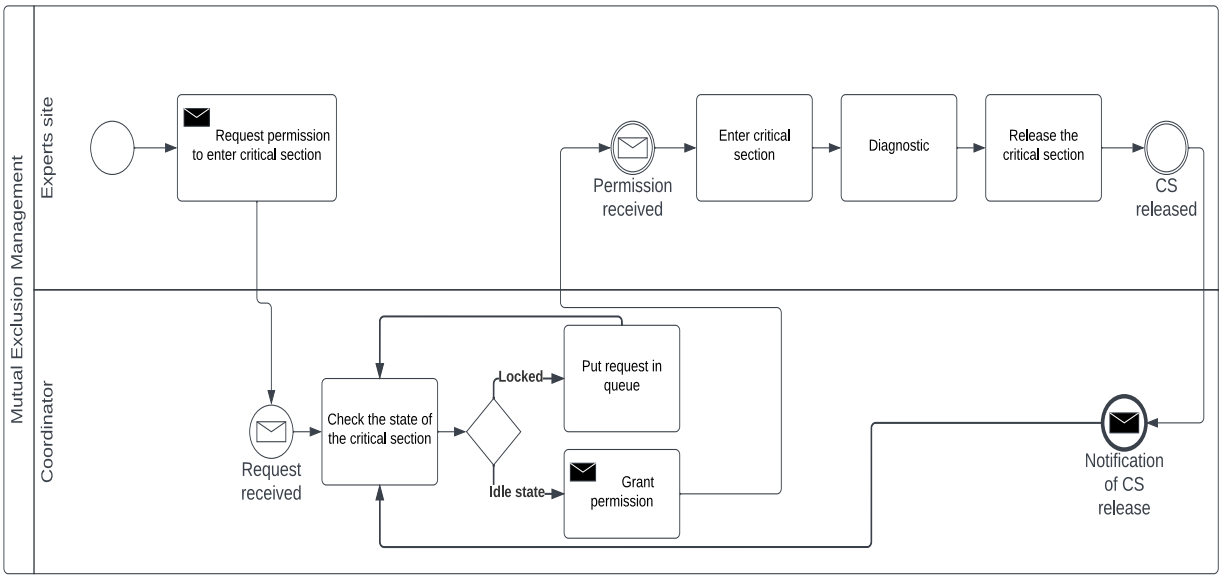

Figure. 10. BPMN Model of the Process of Mutual Exclusion Management

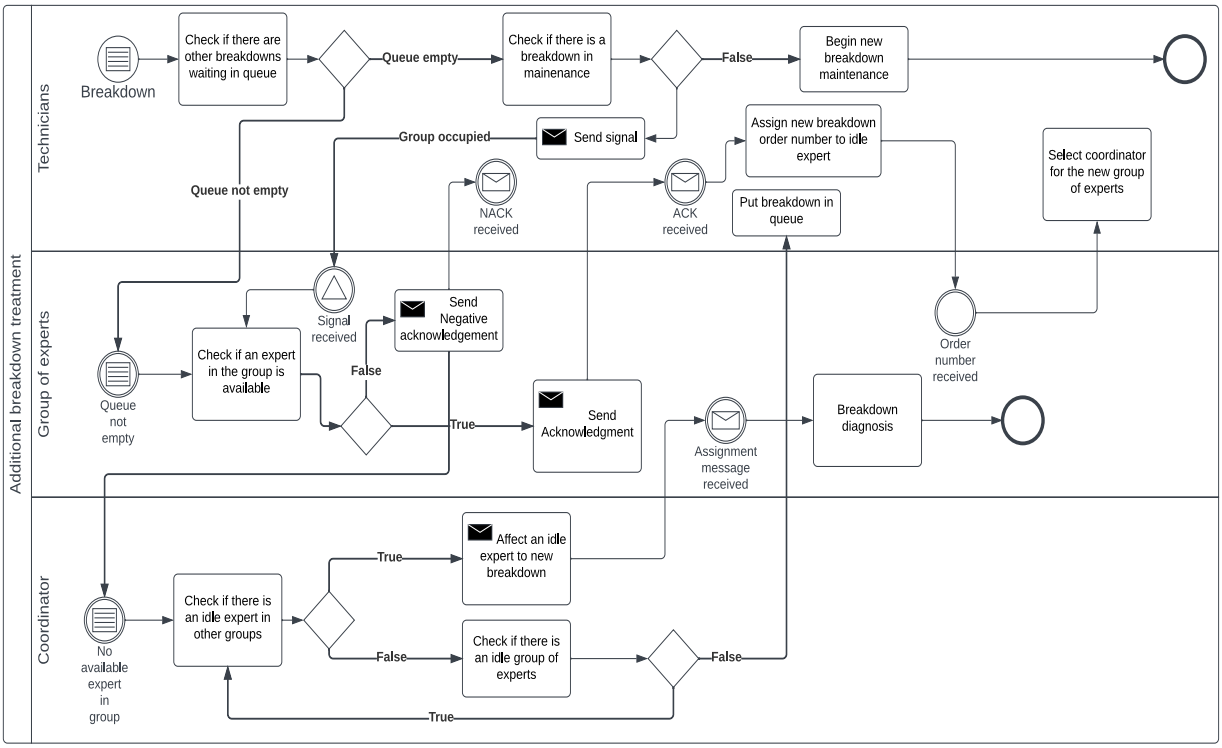

Figure. 11. BPMN Model of the Process of Additional Breakdown Treatment

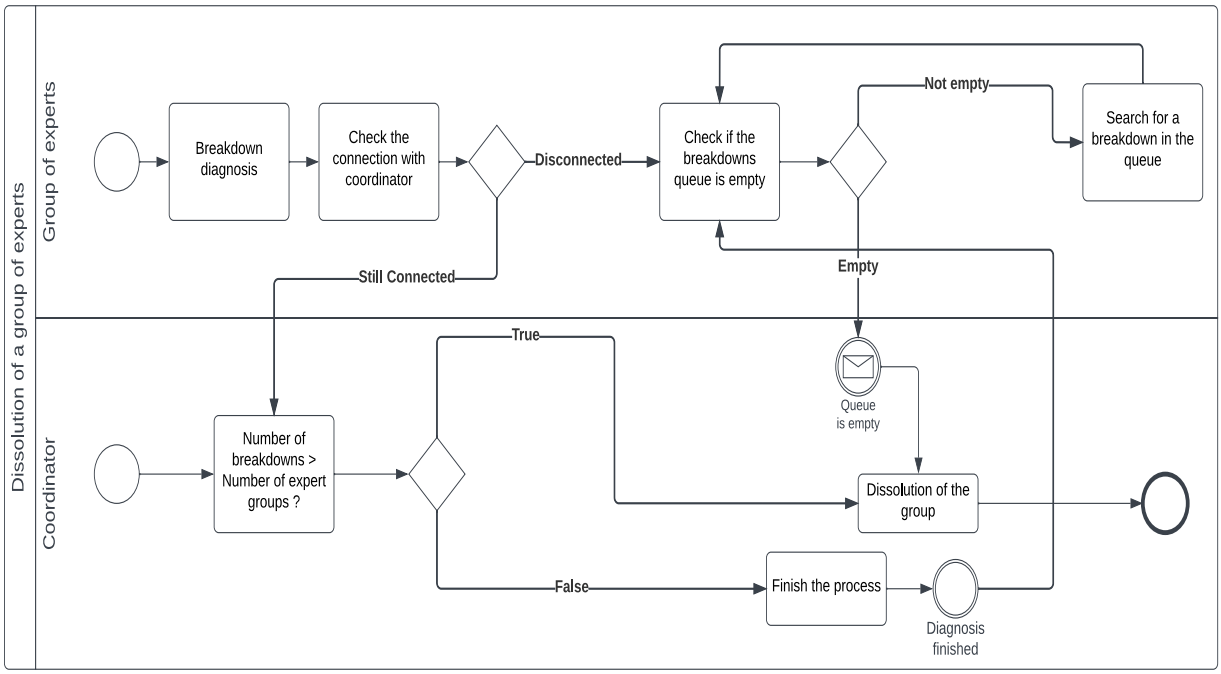

Figure. 12. BPMN Model of the Process of Dissolution of a Group of Experts

4.5. Multi-criteria approach Results

To evaluate the multi-criteria approach, we retrieved the performances of four maintenance experts based on the election criteria. Their performances are displayed in Table 7.

Table 7. First Performance Matrix

Actions/ Criteria	Experience as Expert (Integer)	Treated Breaks (Integer)	Distance (Km)	Coordinator Experience (Integer)	Expert Response time (Min)	Open Ports (Int)	Vulners (Int)	Severity Sum	Connect Type	Net Latency (ms)	Down Speed (Mbps)	Upload Speed (Mbps)
E1	15	21	2500	2	15	12	13	5.3	0.6 (ADSL)	150	20	5
E2	13	27	750	2	5	10	11	17.4	1 (Fiber)	65	1500	750
E3	18	32	4000	0	10	15	18	6.1	0.3 (4G)	222	50	25
E4	10	30	1800	1	10	9	10	12.9	0.8 (Sat)	95	100	50

4.5.1. Criteria Elicitation Technique

To make sure that all pertinent objectives are found, a variety of methodologies can be employed, including in-depth interviews, workshops, and questionnaires [32]. There are numerous methods

for gathering criteria and preferences that lead to decisions. These methods include: NGT (Nominal Group Technique), Focus Group Discussion, Interviews and Q methodology [94]. One of the methods based on questionnaires is the Delphi method, which can be used for criteria elicitation. Mukherjee N. *et al.* [94] advise employing the Q approach and Delphi technique to elicit judgments in high-conflict scenarios. The Nominal Group Technique may be more appropriate when there is less debate and a consensus is required quickly [94]. When compared to traditional face-to-face meetings, the Delphi technique enhances decision-making [94]. In our paper, we used Delphi method to gather the criteria from maintenance experts, as utilizing expertise does not mean giving up objectivity [95].

Twenty maintenance professionals were contacted individually. These professionals have been chosen for their skills and their experience in the diagnosis and maintenance of breakdowns. They did not know the identity of the other professionals contacted. A questionnaire was sent to each expert. We proposed 27 criteria in the questionnaire and asked the experts to evaluate them with the possibility to suggest other criteria. Each expert had to check a box saying weather a criterion is useful or not useful for electing the most suitable expert for the coordinator role. They could transmit the questionnaire only if it was entirely finished, and they could only respond once every iteration. Each iteration was expected to last between two and three weeks. Every expert had to send a response within a maximum period of four weeks. The results were sent individually and they were aggregated into percentages. Each expert could compare their answers with those of the group anonymously. After that, they had the chance to change their preferences and send them again. A criterion was accepted if it had a positive consensus. Positive consensus was made if at least 75% of the experts found the criterion useful. A criterion was excluded if it had a negative consensus. Negative consensus was made if at least 75% of the experts found the criterion not useful [96].

Using the Delphi method, we managed to gather 12 criteria from 20 experts. The criteria agreed on by the experts are:

- **Experience as a DM:** The experience of the expert as a decision maker in previous group decision processes.

- **Number of Treated Breakdowns:** The number of treated breakdowns during previous sessions in which the expert was involved. This serves as an indicator of the experience in the field of collaborative E-maintenance which is a favorable criterion.
- **Distance:** A non-beneficial criterion which represents the distance between the expert's current location and the breakdown site.
- **Coordination Experience:** The number of times the expert has been a coordinator in previous group sessions. This indicates the coordination experience of the expert.
- **Response Time:** The time it takes an expert to respond to a meeting request.
- **Number of Open Ports:** We used the free open-source software "Nmap" to scan the decision makers' machines for open ports [97]. Both TCP and UDP ports were scanned. We need to minimize this criterion, because a machine's open ports are thought to be a good indicator of how many potential system doors are open, putting them vulnerable to attacks [98].
- **Number of Vulnerabilities:** In order to scan each decision maker machine for vulnerabilities we used a vulnerability scanner called "Nessus Essentials" [97]. Which is a free version of Nessus that allows to scan a limited number of machines. This is a non-beneficial criterion, since a big number of vulnerabilities indicates that there is a big number of possible exploits.
- **Sum of Vulnerabilities Severities:** This shows how dangerous are the vulnerabilities present on the expert's machine. Nessus detects vulnerabilities and retrieves their severity [97]. The severity is measured by a score called Common Vulnerability Scoring System (CVSS) which is a quantitative technique [99, 100]. We use CVSS v3.0 base score and add up all discovered vulnerabilities' CVSS ratings [99]. This is a non-beneficial criterion that has to be minimized.
- **Type of Internet Connection:** Mobile Data/ADSL/Satellite/Fiber-optic. For each type we give a score indicating the performance. 4G gets a 0.3. ADSL: 0.6. Cable: 0.7. Satellite: 0.8. Fiber: 1. Mobile data is the worst option as it has a limited data plan and is not reliable. Fiber optic is the best option since it allows for very high internet speeds, lower network latency and is much more reliable and stable than other options. This criterion indicates the stability of the network.

- **Average Network Latency:** The average amount of time it takes a packet to travel from its source to its destination is known as network latency or response time [101]. It is measured in milliseconds. This is a non-beneficial criterion. The response time is responsible for the delay during video and audio calls, as well as in sending and receiving messages and files.
- **Download Speed:** The download speed of the expert's network measured in Mbps. This criterion affects the video and audio quality as well as the time necessary to download files.
- **Upload Speed:** The speed of which the data are sent from the user to the server measured in Mbps. The voice or video call relies heavily on this criterion as it determines the quality of audio and video that the user can output and send to other participants in real time. It also affects the time needed to send files to other users.

4.5.2. AHP for Fixing Criteria Weights

In our work, we used AHP [39] for two purposes:

(i) Structure our problematic into three layers (global objective, local objectives and alternatives) as shown in Figure. 13.

(ii) Calculate the weight of each criterion based on verbal judgements obtained from the experts.

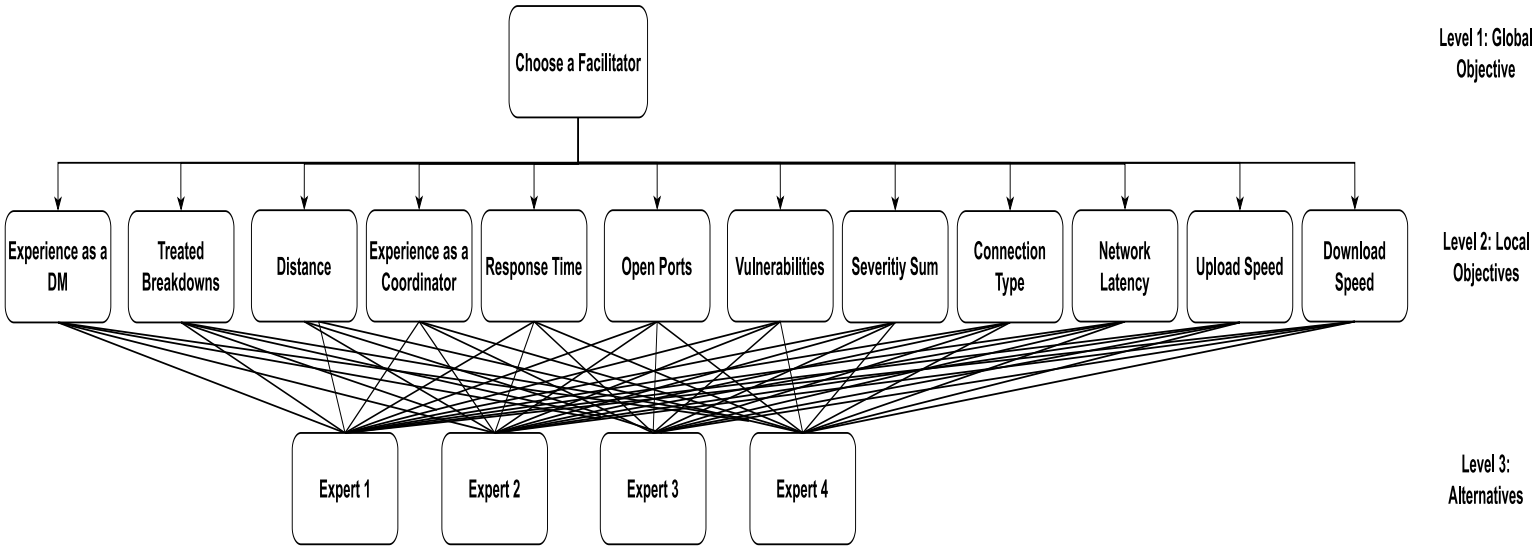

Figure. 13. Hierarchy of Facilitator Election Problematic

Every expert compared each pair of election criteria in an individual pair-wise comparison matrix. Tables 6 to 9 show the individual pair-wise comparison matrices of the four experts. To compute the individual weight vectors, AHP with normalized GMM (Geometric Mean Method) was used. All matrices are consistent as all of their consistency ratios are under 0.1.

Table 8 Pair-wise Comparison Matrix of Expert 1

Criteria/Criteria	Experience as Expert	Treated Breaks (Integer)	Distance	Coordinator Experience	Expert Response time	Open Ports (Int)	Vulnerabilities	Severity	Connect Type	Net Latency (ms)	Download Speed	Upload Speed (Mbps)
Experience as expert	1	1/5	3	1/3	3	1	1/2	1/3	3	3	4	2
Treated Breaks	5	1	5	1/3	3	3	2	1	5	3	7	3
Distance	1/3	1/5	1	1/3	1	5	3	2	7	3	3	1
Coordinator Exp	3	3	3	1	5	7	5	1	7	6	7	4
Response Time	1/3	1/3	1	1/5	1	3	2	1/2	3	5	2	1
Open Ports	1	1/3	1/5	1/7	1/3	1	1/2	1/6	5	1	1	1/2
Vulnerabilities	2	1/2	1/3	1/5	1/2	2	1	1/5	3	2	2	1
Severity	3	1	1/2	1	2	6	5	1	7	5	9	5
Connect Type	1/3	1/5	1/7	1/7	1/3	1/5	1/3	1/7	1	1/3	2	1
Net Latency	1/3	1/3	1/3	1/6	1/5	1	1/2	1/5	3	1	2	1
Down Speed	1/4	1/7	1/3	1/7	1/2	1	1/2	1/9	1/2	1/2	1	1/2
Up Speed	1/2	1/3	1	1/4	1	2	1	1/5	1	1	2	1
Weights	0.075	0.155	0.086	0.231	0.067	0.034	0.054	0.17	0.022	0.035	0.023	0.048
Consistency Ratio	0.095											

Table 9 Pair-wise Comparison Matrix of Expert 2

Criteria/Criteria	Experience as Expert	Treated Breaks (Integer)	Distance	Coordinator Experience	Expert Response time	Open Ports (Int)	Vulnerabilities	Severity	Connection Type	Net Latency (ms)	Download Speed	Upload Speed (Mbps)
Experience as expert	1	1/4	3	1/4	3	1	1/2	1/2	3	3	4	2
Treated Breaks	4	1	5	1/5	3	4	2	1	5	3	7	3
Distance	1/3	1/5	1	1/3	1	5	3	2	7	3	3	1
Coordinator Exp	4	5	3	1	5	7	5	1	7	6	7	4
Response Time	1/3	1/3	1	1/5	1	3	2	1/2	3	5	2	1
Open Ports	1	1/4	1/5	1/7	1/3	1	1/2	1/6	5	1	1	1/2
Vulnerabilities	2	1/2	1/3	1/5	1/2	2	1	1/5	3	2	2	1
Severity	2	1	1/2	1	2	6	5	1	7	5	9	5
Connection Type	1/3	1/5	1/7	1/7	1/3	1/5	1/3	1/7	1	1/3	2	1
Net Latency	1/3	1/3	1/3	1/6	1/5	1	1/2	1/5	3	1	2	1
Download Speed	1/4	1/7	1/3	1/7	1/2	1	1/2	1/9	1/2	1/2	1	1/2
Upload Speed	1/2	1/3	1	1/4	1	2	1	1/5	1	1	2	1
Weights	0.076	0.149	0.086	0.245	0.067	0.033	0.054	0.163	0.022	0.034	0.023	0.047
Consistency Ratio	0.097											

Table 10 Pair-Wise Comparison Matrix of Expert 3

Criteria/Criteria	Experience as Expert	Treated Breaks (Integer)	Distance	Coordinator Experience	Expert Response time	Open Ports (Int)	Vulnerabilities	Severity	Connection Type	Net Latency (ms)	Download Speed	Upload Speed (Mbps)
Experience as expert	1	1/6	3	1/3	3	1	1/2	1/3	3	3	4	2
Treated Breaks	6	1	4	1/3	3	3	2	1	5	3	7	3
Distance	1/4	1/5	1	1/3	1	5	3	2	7	3	3	1
Coordinator Exp	3	3	3	1	5	7	5	1	7	6	7	4
Response Time	1/3	1/3	1	1/5	1	3	2	1/2	3	5	2	1
Open Ports	1	1/3	1/5	1/7	1/3	1	1/2	1/6	5	1/2	1	1/2
Vulnerabilities	2	1/2	1/3	1/5	1/2	2	1	1/5	3	2	2	1
Severity	3	1	1/2	1	2	6	5	1	6	5	9	5
Connection Type	1/3	1/5	1/7	1/7	1/3	1/5	1/3	1/6	1	1/3	2	1
Net Latency	1/3	1/3	1/3	1/6	1/5	2	1/2	1/5	3	1	2	1
Download Speed	1/4	1/7	1/3	1/7	1/2	1	1/2	1/9	1/2	1/2	1	1/2
Upload Speed	1/2	1/3	1	1/4	1	2	1	1/5	1	1	2	1
Weights	0.074	0.155	0.085	0.232	0.067	0.033	0.055	0.169	0.022	0.037	0.023	0.048
Consistency Ratio	0.096											

Table 11 Pair-wise Comparison Matrix of Expert 4

Criteria/Criteria	Experience as Expert	Treated Breaks (Integer)	Distance	Coordinator Experience	Expert Response time	Open Ports (Int)	Vulnerabilities	Severity	Connect Type	Net Latency (ms)	Download Speed	Upload Speed (Mbps)
Experience as expert	1	1/5	3	1/4	2	1	1/2	1/4	3	3	4	2
Treated Breaks	5	1	5	1/3	3	3	2	1	5	3	7	3
Distance	1/3	1/5	1	1/3	1	5	3	2	6	3	3	1
Coordinator Exp	4	3	3	1	5	7	5	1	7	6	6	4
Response Time	1/2	1/3	1	1/5	1	3	2	1/2	3	5	2	1
Open Ports	1	1/3	1/5	1/7	1/3	1	1/2	1/6	5	1	1	1/2
Vulnerabilities	2	1/2	1/3	1/5	1/2	2	1	1/4	3	2	2	1
Severity	4	1	1/2	1	2	6	4	1	7	5	7	5
Connect Type	1/3	1/5	1/6	1/7	1/3	1/5	1/3	1/7	1	1/3	1	2
Net Latency	1/3	1/3	1/3	1/6	1/5	1	1/2	1/5	3	1	2	1
Down Speed	1/4	1/7	1/3	1/6	1/2	1	1/2	1/7	1	1/2	1	1/2
Upload Speed	1/2	1/3	1	1/4	1	2	1	1/5	1/2	1	2	1
Weights	0.069	0.156	0.086	0.234	0.07	0.035	0.056	0.168	0.022	0.035	0.026	0.045
Consistency Ratio	0.097											

We applied the AIP procedure with the WGMM (Weighted Geometric Mean Method) in order to aggregate the individual weight vectors produced from the pair-wise comparison matrices of the 4 experts into one group weight vector [42]. Since the experts are not already ranked, we gave an equal weight to each expert's weight vector (see Eq. (31)).

Table 12 shows the individual weight vectors of each expert and the aggregated group weight vector in the final row.

$$w_e = \frac{1}{n} = \text{¼} = 0.25 \tag{31}$$

Where n is the number of experts and w_e is the weight of expert e.

Table 12. Aggregation of Individual Weight Vectors into a Single Group Weight Vector

Criteria Weights	Experience as Expert (Integer)	Treated Breaks (Integer)	Distance (Km)	Coordinator Experience (Integer)	Expert Response time (Min)	Open Ports (Int)	Vulners (Int)	Severity Sum	Connect Type	Net Latency (ms)	Down Speed (Mbps)	Upload Speed (Mbps)
E1	0.075	0.155	0.086	0.231	0.067	0.034	0.054	0.17	0.022	0.035	0.023	0.048
E2	0.076	0.149	0.086	0.245	0.067	0.033	0.054	0.163	0.022	0.034	0.023	0.047
E3	0.074	0.155	0.085	0.232	0.067	0.033	0.055	0.169	0.022	0.037	0.023	0.048
E4	0.069	0.156	0.086	0.234	0.07	0.035	0.056	0.168	0.022	0.035	0.026	0.045
Group Weights	0.073	0.154	0.086	0.235	0.068	0.034	0.055	0.167	0.022	0.035	0.024	0.047

4.5.3. MAUT

For our paper, we aimed at changing the utility functions so as to be limited between 1 and 0. The functions must give 0 at worst and 1 at best. We chose the following formula for the logarithmic based marginal utility function:

$$u_2\left(g'_j(a_i)\right) = \log\left(9 * g'_j(a_i) + 1\right). \quad (32)$$

The linear function is defined as follows:

$$u_4 = g'_j(ai);\ u_4 \in [0,1]. \quad (33)$$

The proposed quadratic function is:

$$u_3 = \left(2 * g'_j(a_i) - 1\right)^2. \quad (34)$$

The used step function is defined in the following manner:

$$u_5 = \frac{ceiling\left(op * g'_j(a_i)\right)}{op}. \quad (35)$$

Where “op” indicates the number of evaluation options for criterion j.

Table 13 shows the utility functions that the DMs chose for each criterion:

Table 13. Marginal utility functions applied on each criterion

Criteria	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
Utility Functions	log	log	exp	log	exp	exp	log	log	Step; op=4	lin	log	exp

Applying MAUT on our performance matrix gave the scores presented in Table 14. The ranking of the experts based on these scores is shown in Figure. 14.

Table 14. MAUT Final Utility Scores

Expert	E1	E2	E3	E4
MAUT Final Utility Score	0.552	0.754	0.403	0.601

Figure. 14. Outranking Graph Obtained from MAUT

4.5.4. SAW Results

We applied SAW on our dataset and got the scores for each expert as shown in

Table 15. After that, the alternatives were ranked based on the SAW final score as presented in Figure. 15.

Table 15. SAW Final Score for each alternative

Expert	E1	E2	E3	E4
SAW Final Score	0.709	0.831	0.492	0.576

Figure. 15. Outranking Graph obtained from SAW

4.5.5. PROMETHEE II Results

In this work, the experts chose the preference functions and the parameters shown in Table 16.

Table 16. Preference functions and their parameters

Criteria	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
Function Type	t3	t4	t5	t3	t5	t6	t3	t6	t2	t1	t6	t6
Parameters	p=2	q=1; p=5	q=15; p=100	p=1	q=2; p=20	s=5	p=10	s=9	q=0.1	None	s=2	s=1

The application of PROMETHEE II on our data resulted in giving each expert the corresponding net outranking flow shown in Table 17. This allows us to rank the experts from the best one having the highest net flow to the worst one having the smallest net flow. This ranking is illustrated in the outranking graph presented in Figure. 16.

Table 17. PROMETHEE II Net Outranking Flow

Expert	E1	E2	E3	E4
Net Outranking Flow	-0.06	0.314	-0.231	-0.023

Figure. 16. Outranking Graph Obtained from PROMETHEE II

Visual PROMETHEE Analysis

Visual PROMETHEE is a software based on the PROMETHEE methods which has both academic and business editions. We used Visual PROMETHEE academic edition to analyze how sensitive is the final outranking of the experts to changes in criteria weights. This can be done using the Walking Weights window (Figure. 17). Which allows us to change the weight of any criterion and see how the outranking of alternatives changes in real time. In Figure. 17 you can see that when we change the weight of the criterion Treated Breakdowns to 75%, the ranking of the experts changes. An easier way to analyze the sensitivity is by using the Visual Stability Intervals (see Figure. 18). This window shows the interval in which the weight of a criterion can vary without affecting the final outranking of the alternatives. Table 18 shows the weight stability interval of every criterion obtained from the Visual PROMETHEE software. We can see that changing the

weight of Network Latency or Connection Type doesn't change the ranking of the experts. Another observation is that the criteria: Distance, Number of Open Ports and Number of Vulnerabilities, have a wide range of weight values that can be used while still keeping the same ranking.

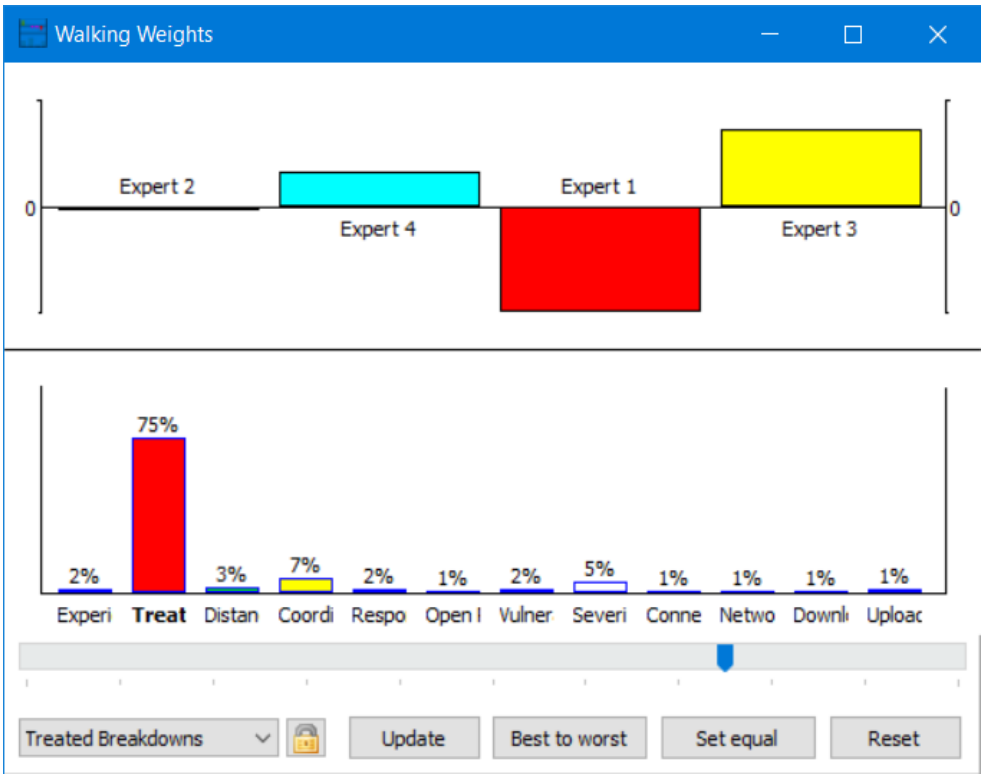

Figure. 17. Walking Weights Window of Treated Breakdowns Criterion

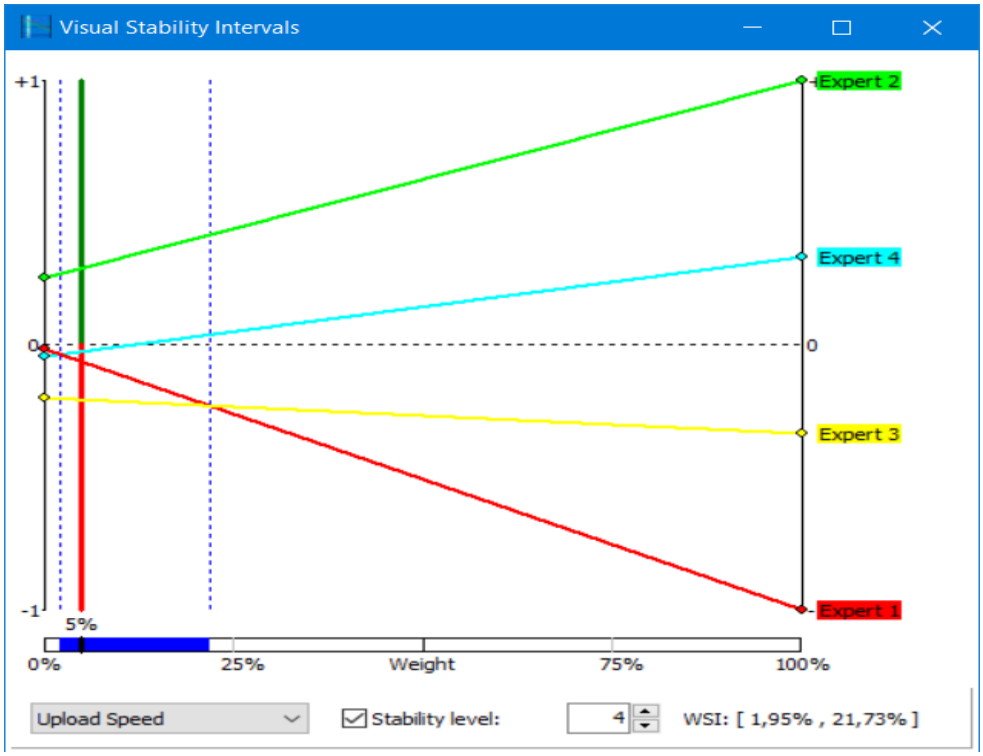

Figure. 18. Visual Stability Intervals Window of Upload Speed Criterion

Table 18. Weight Stability Interval for Each Criterion

Criterion	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
Weight Stability Interval	[0%, 9.83%]	[12.96%, 21.60%]	[3.17%, 100%]	[16.21%, 26.26%]	[0%, 40.24%]	[0%, 79.75%]	[0%, 71.63%]	[0%, 23.18%]	[0%, 100%]	[0%, 100%]	[0%, 19.84%]	[1.95%, 21.73%]

Figure. 19 illustrates a tool called PROMETHEE Rainbow. In this window each color corresponds to a criterion. The upper criteria are the good features of an expert. While the lower criteria are the weaknesses of an expert.

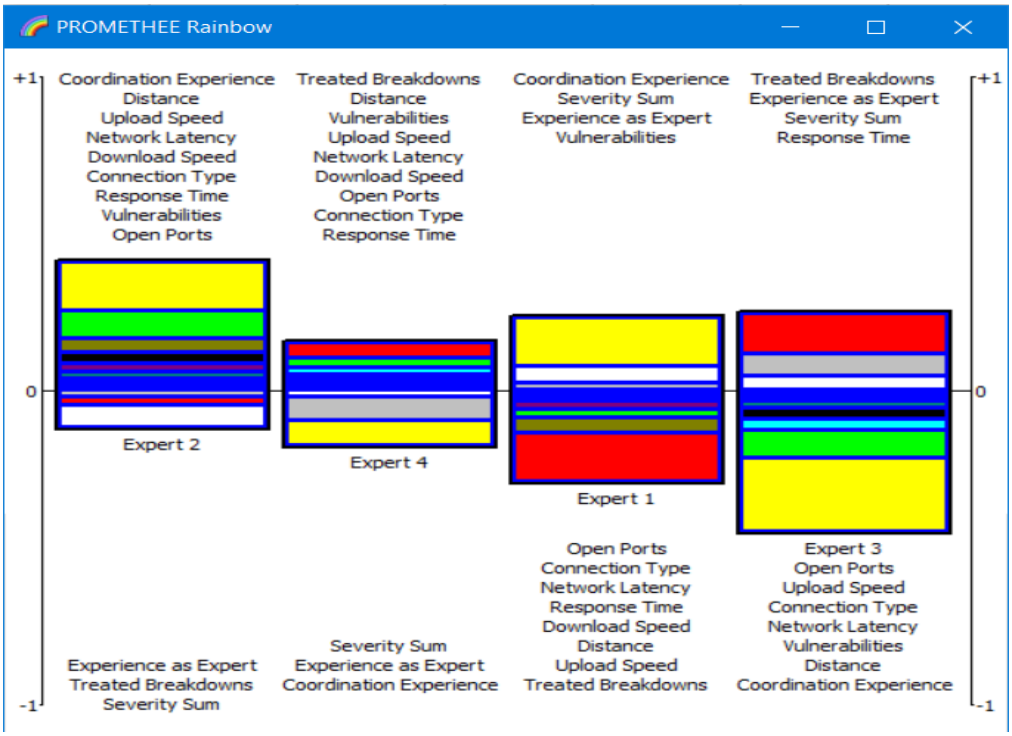

Figure. 19. PROMETHEE Rainbow Window

The PROMETHEE Diamond window in Figure. 20 allows us to see both the partial and complete rankings of the experts obtained from both PROMETHEE I and PROMETHEE II respectively. The vertical Phi net flow axis is used to demonstrate PROMETHEE II complete ranking. While the cones in the 2D plane represent the experts. The cone corresponding to expert 2 covers all other cones. So even in PROMETHEE I partial ranking, expert 2 is still the best option.

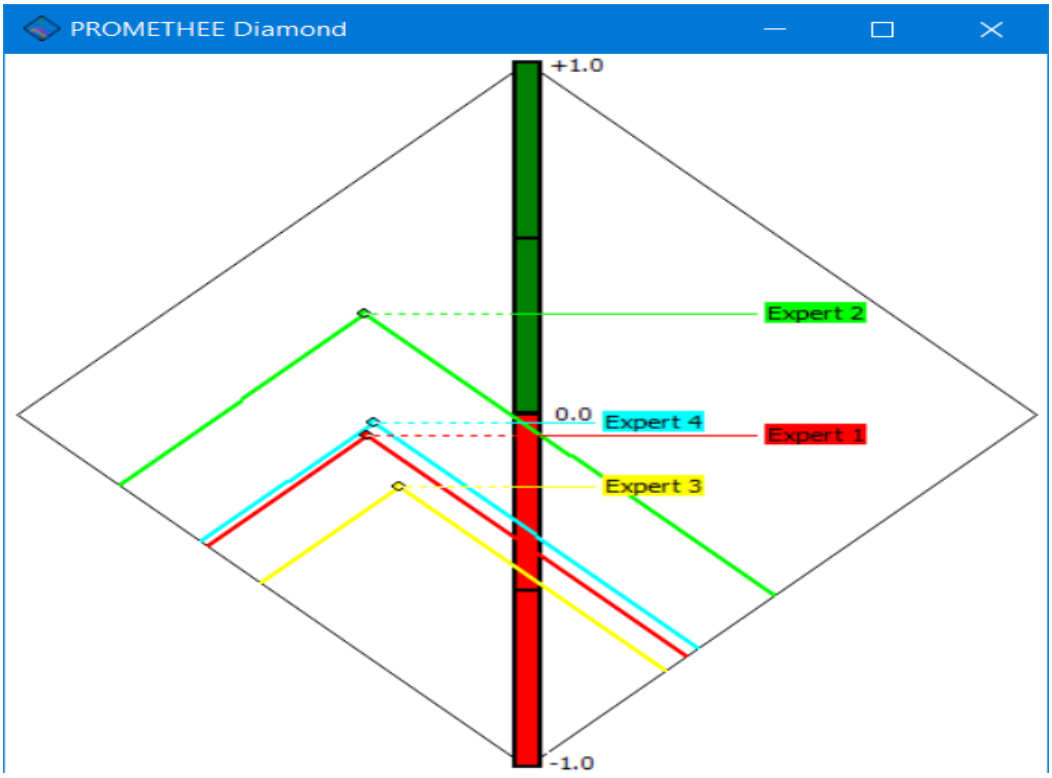

Figure. 20. PROMETHEE Diamond Window

4.5.1. TOPSIS Results

After applying TOPSIS to our performance matrix, we obtained the relative closeness of each expert to the ideal solution. These values are presented in Table 19. Figure 21 shows the outranking graph of the experts based on their relative closeness.

Table 19. TOPSIS Relative Closeness for Each Expert

Experts	Expert 1	Expert 2	Expert 3	Expert 4
Relative Closeness	0.705	0.668	0.34	0.472

Figure 21. Outranking Graph Obtained from TOPSIS

4.5.2. Metrics and Criteria Used for Comparison

In this section, we are going to compare between the used MCDA methods and their results based on 9 metrics.

- **Time Complexity:** The worst time complexity for the MCDA method represented in big O notation. Where n is the number of experts and m is the number of criteria.
- **Best Execution Time:** The least amount of time it takes to run the Python implementation of the MCDA method on our dataset without generating the outranking graph. It is measured in milliseconds and the program is run several times to obtain the best possible execution time for each method.
- **Memory Cost:** This metric shows the highest temporary memory occupied by the Python implementation of the method during its execution. This metric was obtained using the "memory_profiler" library.
- **Number of Satisfied Criteria by Leader:** The number of criteria in which the elected coordinator has the maximum value compared to other experts.
- **Objectives Attainment Percentage:** This shows how much the elected coordinator is close to reaching the best weighted values in all criteria. It is computed based on Eq. (36):

$$\boldsymbol{P_i = \frac{\sum_{j=1}^{m} g_j(e_i) * w_j}{\sum_{j=1}^{m} g_j(e_{best}) * w_j} \times 100} \quad (36)$$

Where P_i is the objectives attainment percentage of the elected DM. e_{best} is the DM with the best value in criterion j that's available in the performance matrix, and e_i is the elected DM UID.

- **Number of Parameters:** The number of thresholds and preference functions that the DM has to fix before calculating the final score. It depends on the number of criteria and the number of functions' parameters.
- **DM Involvement (Input Level)** [38]**:** Based on the average of 4 parameters: Number of parameters and data quantity, the knowledge and skill level required from the DM within the decision problem context, the requirement to use other techniques and the definition time. It shows the modeling effort required by the method in order to reach the wanted results [38].

- **Elected Leader:** The first ranked expert from the ranked list of experts based on their final score. He will be responsible for coordinating the group of experts.
- **Next Best Leader (Backup Coordinator)** [9]**:** The second-best ranked expert from the list of ranked experts, who will take over the coordinator's role in case the current coordinator leaves or if he loses connection with the breakdown site.

4.5.3. Used Tools

In order to use different MCDA methods on our dataset we used the open-source library "PyDecision" version 2.8.7 [102]. Which contains several MCDA methods implementations written in Python. This allowed us to avoid doing calculation manually by automating them. Thus, reducing calculation time and allowing the experts to test different parameters to find the most suitable ones and view the results for each method quickly. It also provides an outranking graph based on "Matplotlib" library. Which allows the experts to easily understand the ranking of alternatives and view the potential coordinator as well as the backup leader (2^{nd} best expert). We also added the MAUT implementation to the "PyDecision" Github repository. The system used for these experimentations is a Windows 10 64bit on a core i5 6300HQ with maximum clock speed of 3.2 GHz, 8 GB of 2133 MHz DDR4 RAM and M.2 SATA SSD. The used version of Python is 3.11.0.

In order to illustrate the obtained results in an easy and simple way for understanding and analyzing, we used bar charts and radio charts. The Bar chart (Figure 22) was rendered using the "Matplotlib" library and the Radio charts (see Figure 23 - Figure 27) were done through the "Plotly" python library. All of the charts used normalized values of the performance matrix. The values were normalized by dividing them by the maximum value of each criterion.

4.5.4. Discussion and Interpretation of Results

Table 20 shows the performance of each method and the results obtained after applying it on the dataset. In this subsection, the results are compared and discussed thoroughly in order to reach a conclusion on which MCDA method is more suitable for electing a GDSS facilitator.

Table 20. Comparison of MCDA Methods in Leader Election

Method/ Metric	Time Complexity	Best Execution Time (ms)	Memory Cost (MB)	Number of Satisfied Criteria by Leader	Objectives Attainment Percentage	Number of Parameters	Input Level	Elected Leader	Next best Leader
MAUT	O(nm) [103]	1,97	122,49	7	98.52%	13	High	E2	E4
SAW	O(nm) [103]	0,98	122,46	7	98.52%	0	Low	E2	E1
PROMETHEE II	O(mn²) [104]	1,95	59,88	7	98.52%	15	Medium	E2	E4
TOPSIS	O(n²) [41]	0,97	122,53	2	7.14%	0	Low	E1	E2

The criteria satisfaction graphs show how close an expert is to reaching the best value in each criterion. The Bar chart (Figure 22) represent the criteria satisfaction for all experts. While (Figure 24 - Figure 27) show the criteria satisfaction for each expert alone. All of the charts used normalized values of the performance matrix. The values were normalized by dividing them by the maximum value reached in each criterion.

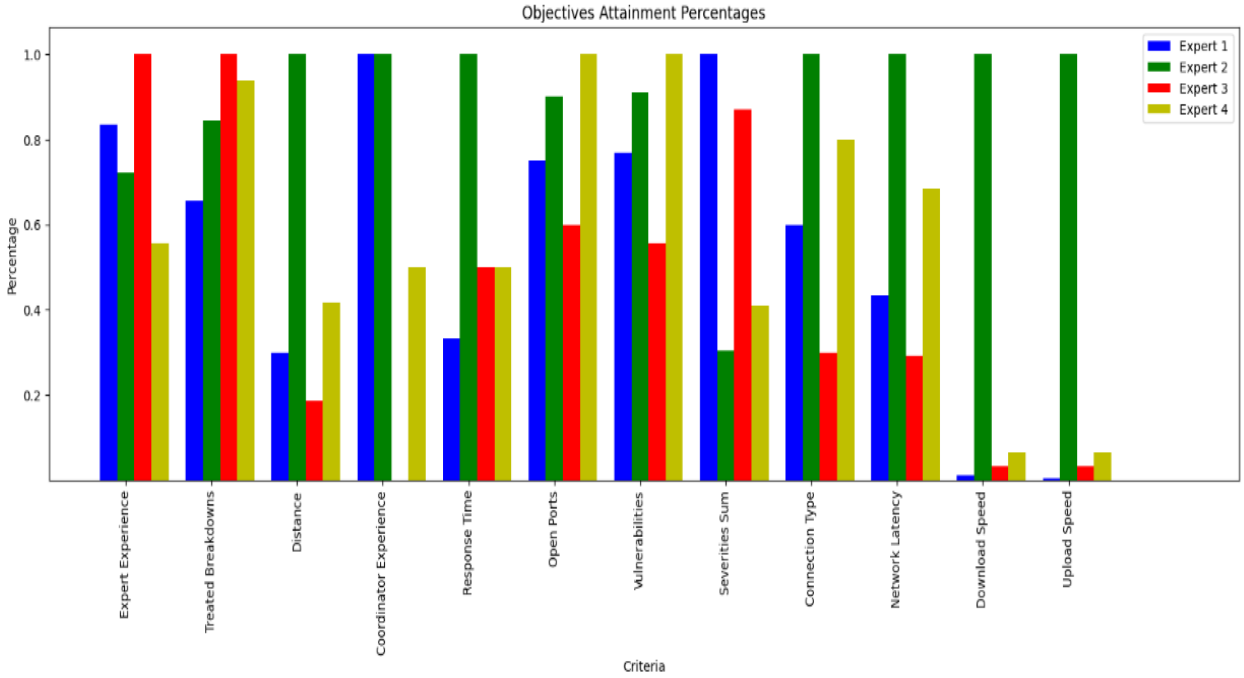

Figure 22. Objectives Attainment Percentage for Each Expert

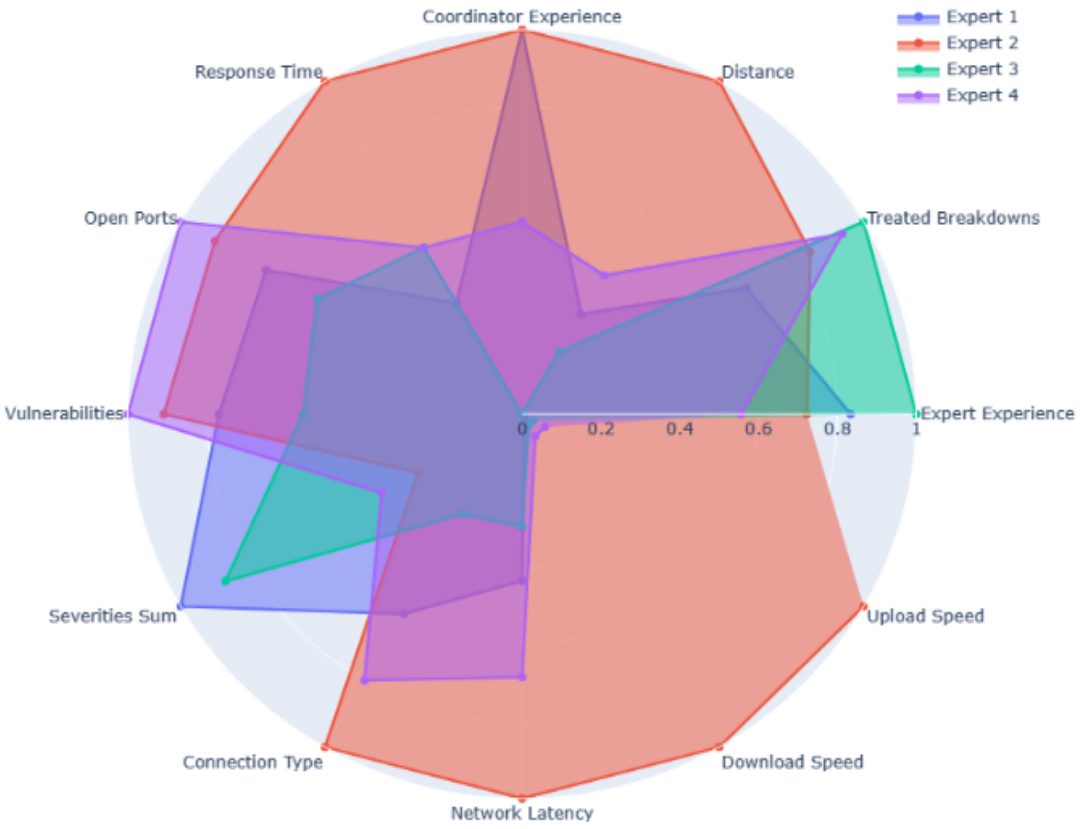

Figure 23. Radio Chart Criteria Satisfaction for Each Expert

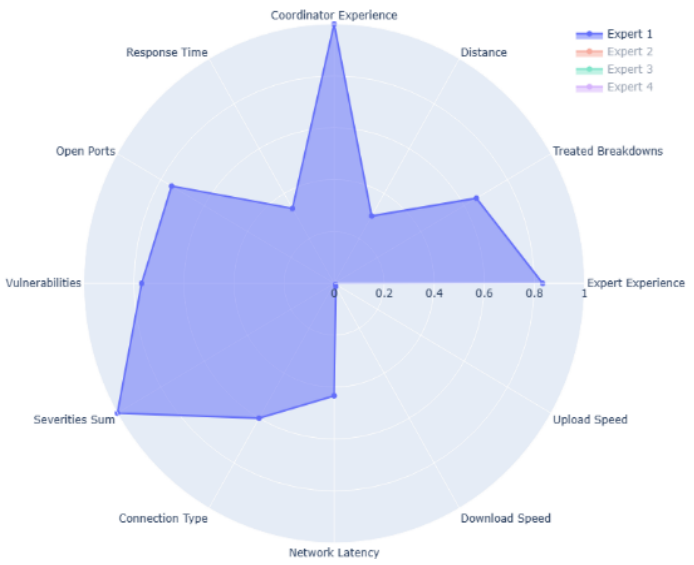

Figure 24. Criteria Satisfaction by Expert 1

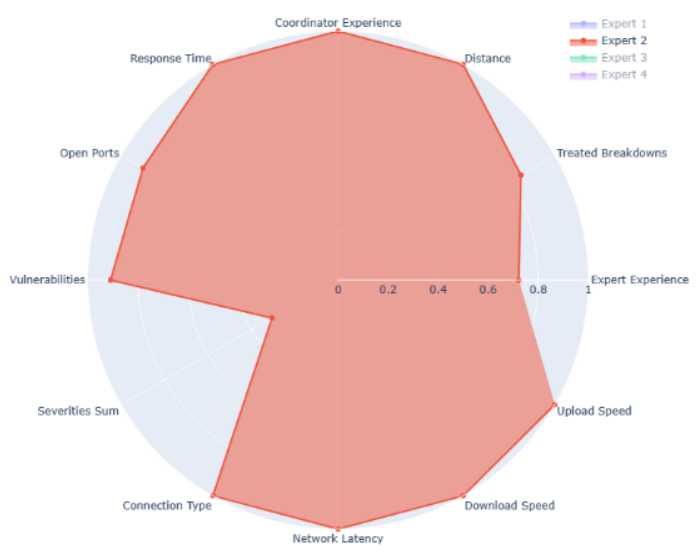

Figure 25. Criteria Satisfaction by Expert 2

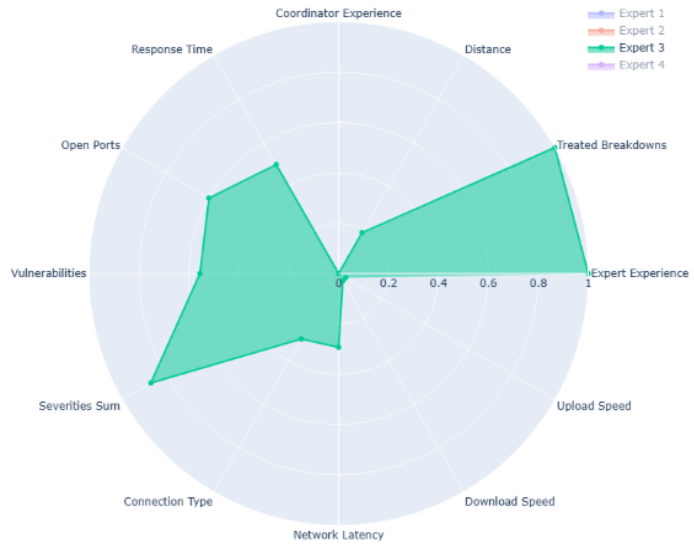

Figure 26. Criteria Satisfaction by Expert 3

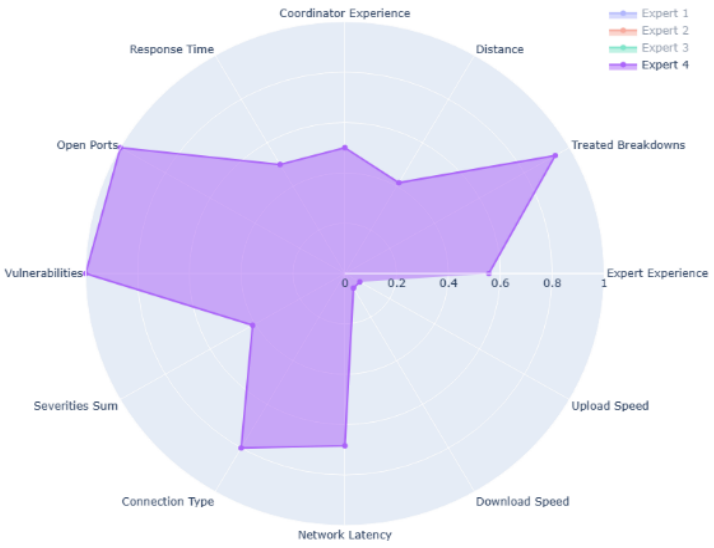

Figure 27. Criteria Satisfaction by Expert 4

From Table 20 we can see that most of the methods ranked expert 2 the highest, resulting in electing him as a coordinator for the experts' group. This is because expert 2 has the best values in 7 out of 12 criteria. Meaning, he satisfies the majority of the criteria including distance and coordinator experience (see Figure 22 and Figure 25) which is the most important criterion. On

the other hand, TOPSIS ranked expert 1 as the best expert for the role who also has the most experience as a coordinator and his machine has the least vulnerability severity (see Figure 24). Even though MAUT, PROMETHEE II and SAW all came up with the same leader, SAW selected a different backup leader expert 1 compared to expert 4 in the former methods. The backup leader chosen by MAUT and PROMETHEE II is more in favor of security and technical expertise than in network performance and coordination (see Figure 27). Furthermore, expert 4 is better than expert 1 in 9 criteria and better than expert 3 in 7 criteria. While SAW also offered a backup who prioritizes technical expertise and security, additionally, he also has the maximum coordination experience which is more relevant to the coordinator role. In contrast, TOPSIS ranked expert 2 as the next best leader, who is clearly the best expert overall.

Despite PROMETHEE II and MAUT gave the same results in terms of coordinator and backup leader, PROMETHEE II has worse theoretical time complexity than MAUT, as it includes calculating evaluative differences between each pair of alternatives for each criterion. But MAUT took more time to execute because the PyDecision implementation of PROMETHEE II is optimized to use matrices with dimensions smaller than the number of pairwise comparisons. Although, the execution time is variable and depends on the machine's state and background tasks. Moreover, MAUT requires the experts to be more involved in the decision process. Since the expert has to define the appropriate function for each criterion, but it also makes the method more flexible. PROMETHEE on the other hand, requires to fix several parameters depending on the preference function used and desired preference degree. Which can become very time consuming and complicated when the number of criteria is high. SAW doesn't require specifying preference functions. Its main advantage is its simplicity and ease of use. TOPSIS requires less CPU execution time than the other methods. Because of the vector normalization method. Which doesn't require a distinction between beneficial and non-beneficial criteria. Additionally, it doesn't need any parameters nor preference functions. All of these methods take almost the same amount of system memory as they are all based on matrices. So, their memory footprints depend heavily on the biggest matrix size and the normalization formula. Except for PROMETHEE II, which took less memory space because the implementation in "PyDecsion" is well optimized. Instead of declaring a matrix of (((n-1) × n) × m) that contains the distance between alternatives for all criteria, the author made a loop which uses a single matrix with (n×n) dimensions that gets distances of all alternatives for one criterion in one iteration from a distance function. After that, it accumulates

the obtained preference values to a preference matrix (n×n) with each criterion, and overwrites the same distance matrix for the next criterion. Additionally, PROMETHEE II doesn't require normalization like other methods.

After analyzing the obtained results and comparing the used methods, we concluded that PROMETHEE II is the most suitable method for electing the coordinator of the experts among the four tested methods. Even though it gave similar results to MAUT. First, it has better execution time than MAUT. In addition, PROMETHEE II selected the best expert overall to be the coordinator (expert 2) who satisfies most of the criteria including the most important criterion which is coordination experience. Moreover, the chosen backup leader (expert 4) outperforms the backup leader chosen by SAW (expert 1) in the majority of criteria (9 criteria). Additionally, we can see from Figure. 19 that expert 2 and expert 4 both have more good features and less weaknesses than expert 1 and 3. Furthermore, the backup expert (expert 4) has an average performance in other criteria except for network quality criteria (download and upload speeds), which have lower priority compared to experience and security related criteria. In addition, PROMETHEE II gives the DM some control over the method by choosing the preference functions and threshold parameters, which makes it more flexible. This enables PROMETHEE II to be a good balance between subjectivity and formality. On top of this, PROMETHEE II requires a lower input level of DM than that of MAUT (medium versus high). Finally, PROMETHEE II is simpler and easier to use when compared to MAUT. This is because MAUT requires the DMs to define the marginal utility functions, while in PROMETHEE II the DMs must specify predefined preference functions and specify threshold parameters. Thus, the DMs won't have to give or change the function's definition [105].

4.6. Proposed Facilitator Election Tool

In order to automate the election process based on the proposed approach, we created GFET tool for electing a GDSS facilitator within the context of collaborative e-maintenance. It is intended to be used by the expertise center [91]. GFET gives the users the option to experiment with four different MCDA methods.

4.6.1. Integration of GFET

Authors proposed a collaborative e-maintenance model integrating the GFET (see Figure 28). This is based on the model proposed by Hedjazi [91]. When the expertise center receives a breakdown notification from the technician, it invites experts specialized in fields that are relevant to the breakdown's nature. When an expert accepts an invite, the center asks him for his data regarding the election criteria. After the expert has sent his individual evaluation vector, the expertise center then asks the expert to fill an excel file holding the pair-wise comparison matrix in order to compare the election criteria. This process is repeated for every expert individually. When every expert has sent his individual evaluation vector, all the vectors are combined into a single performance matrix. Next, when every expert has sent his individual pair-wise comparison matrix, these matrices are then entered in GFET to compute the global criteria weights using AIP. Afterwards, the expertise center selects an MCDA method and enters its preference functions and parameters. Following this, GFET returns the UID of one expert as the coordinator. The center sends his UID to the experts. Succeeding this, the experts coordinated by the elected facilitator conduct the meeting and form a PRA (Plan of Repairing Actions). Next, the PRA is sent to the technician on site so he can fix the broken machine by following the provided actions.

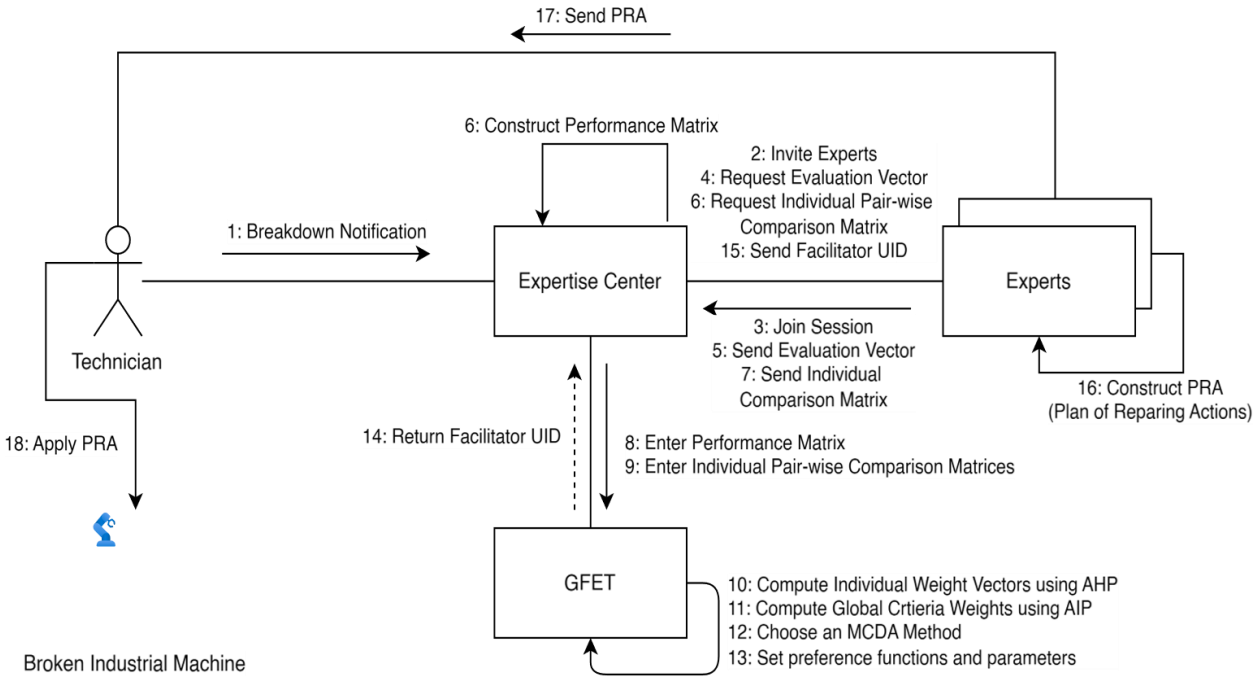

Figure 28 UML Collaboration Diagram Integrating GFET in Collaborative E-maintenance

4.6.2. Tool Presentation

When the user starts the tool, he sees the frame shown in Figure 29. This frame gives the user the option to choose between two modes: Simplified or Advanced mode. The simplified mode is designed for users who don't have strong knowledge in multi-criteria methods. In this mode the user enters the performance matrix and the pair-wise comparison matrices of the criteria. Following that, he can directly view the election results obtained using the PROMETHEE II method set with the default preference functions and thresholds presented in section 0. The advanced mode is designed for users with solid knowledge in MCDA methods. While this mode also requires to input the same data, it adds the possibility to choose and test different MCDA methods. Furthermore, this mode enables the user to change the default preference or utility functions as well as their parameters in both MAUT and PROMTHEE II methods.

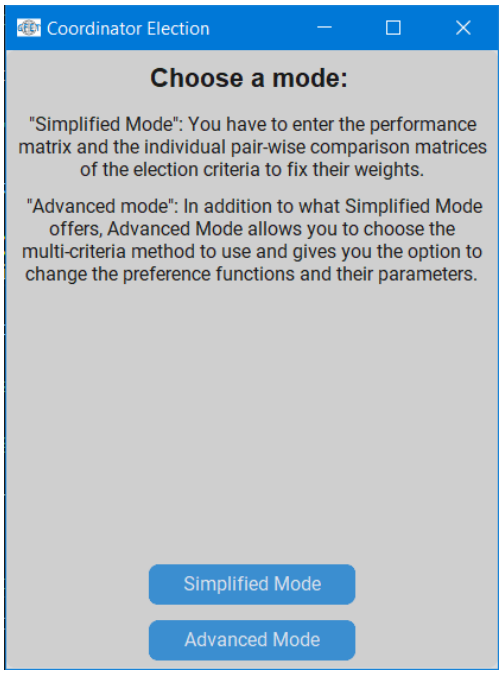

Figure 29 Mode Selection Frame

After choosing a mode, the main frame is shown (see Figure 30), in which the user has to specify the number of DMs before using any other functionality. The number of DMs has to be greater than 1 for it to be valid.

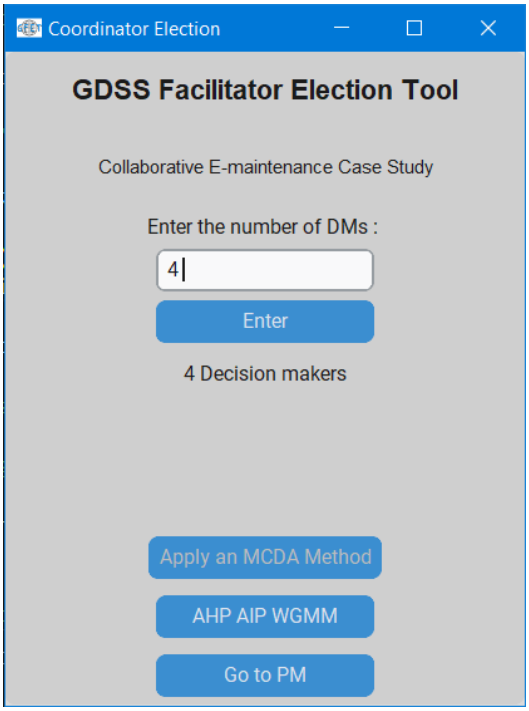

Figure 30 Main Frame of GFET

Figure 31 presents the frame in which the user can enter the performance matrix either by manually filling each cell or by importing an excel file containing an already filled performance matrix. Once the matrix is entered, the user submits it by clicking on the Submit button.

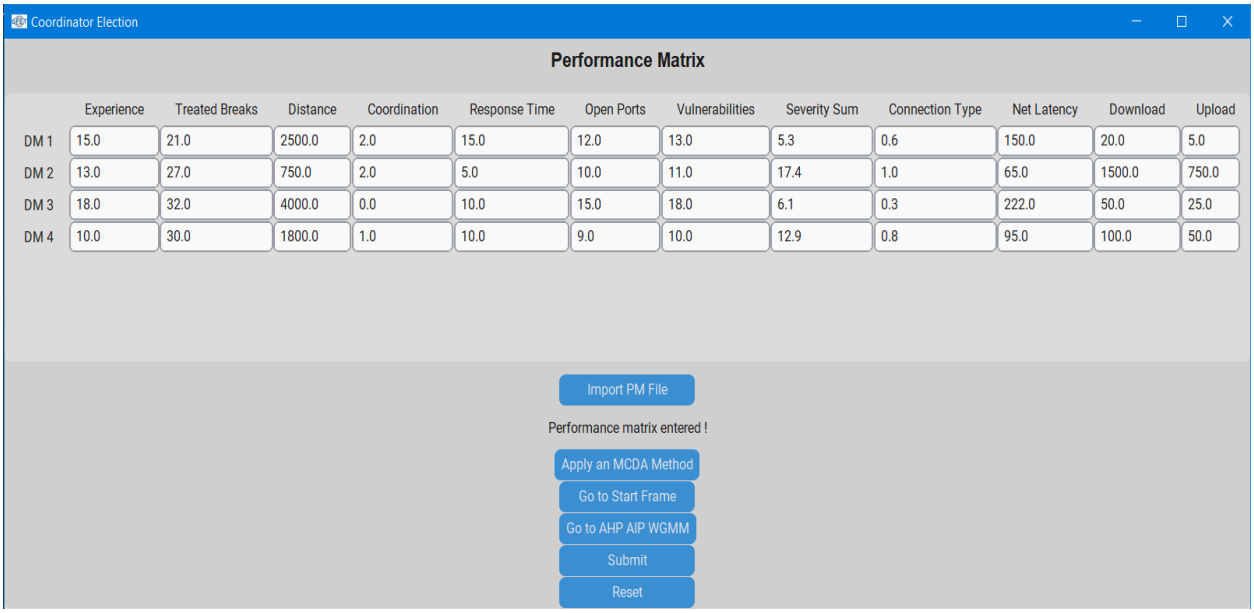

Figure 31 Performance Matrix Input Frame

Figure 32 shows the frame in which the user enters the individual pair-wise comparison matrices of the election criteria belonging to each DM in order to calculate the global weights of the criteria. The user can either import an excel file containing all the individual pair-wise comparison matrices. Each sheet of this file has the matrix of a single DM.

	Experience	Treated Breaks	Distance	Coordination	Response Time	Open Ports	Vulnerabilities	Severity Sum	Connection Type	Net Latency	Download	Upload	Consistency Ratio
Exp 1	0.075	0.155	0.086	0.231	0.067	0.034	0.054	0.17	0.022	0.035	0.023	0.048	0.095 Consistent
Exp 2	0.076	0.149	0.086	0.245	0.067	0.033	0.054	0.163	0.022	0.034	0.023	0.047	0.097 Consistent
Exp 3	0.074	0.155	0.085	0.232	0.067	0.033	0.055	0.169	0.022	0.037	0.023	0.048	0.096 Consistent
Exp 4	0.069	0.156	0.086	0.234	0.07	0.035	0.056	0.168	0.022	0.035	0.026	0.045	0.097 Consistent
Global Weights	0.073	0.154	0.086	0.236	0.068	0.034	0.055	0.168	0.022	0.035	0.024	0.047	

Figure 32 Global Weights Frame

Another option, is to enter manually the matrix of each DM in separate window. This is shown in Figure 33. In each window corresponding to a DM, the user can manually fill each cell or he can import an excel file containing the individual pair-wise comparison matrix of the DM. After the matrix is entered, the tool checks if the matrix is consistent or not. The matrix is only accepted if it is consistent. Next, the individual weight vector is calculated and added to the global weights frame alongside the consistency ratio (see Figure 32).

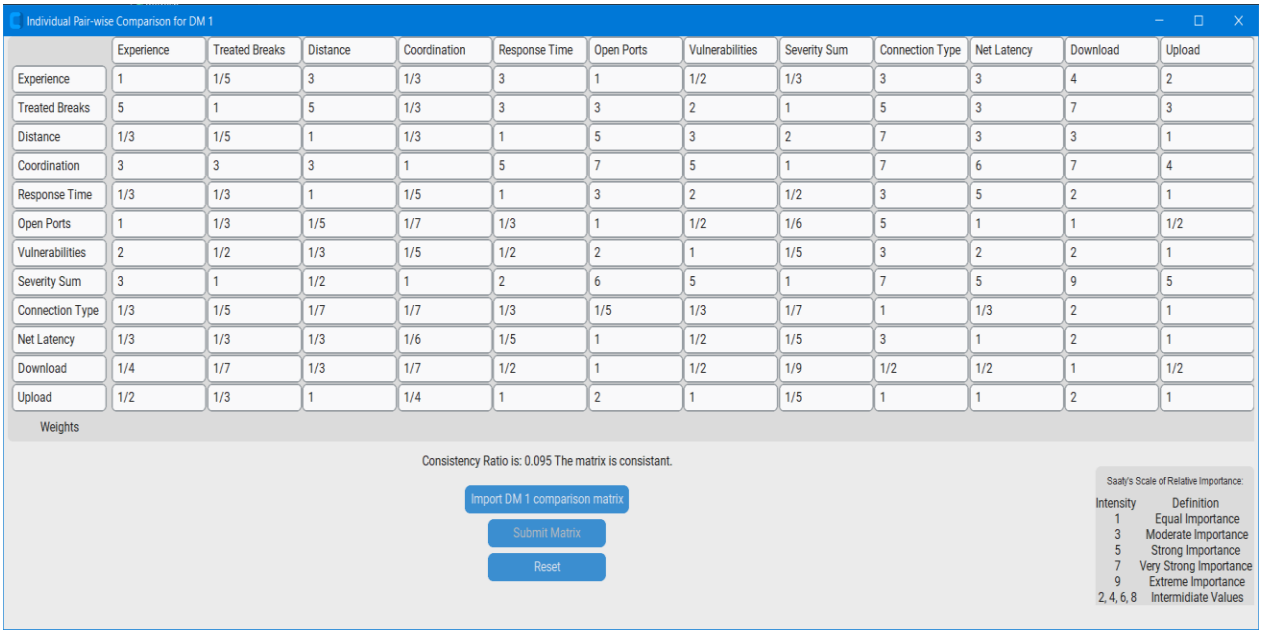

Figure 33 Individual Criteria Pair-wise Comparison Matrix Input Frame

Once all the individual weight vectors are added to the AHP AIP frame, the user can then press a button to compute the global weights using the AIP WGMM aggregation procedure.

When the election criteria global weights are computed and the performance matrix is entered, the user can now access the frame which enables him to select an MCDA method among four different methods (see Figure 34). The user has the option to choose either MAUT, SAW, PROMETHEE II or TOPSIS.

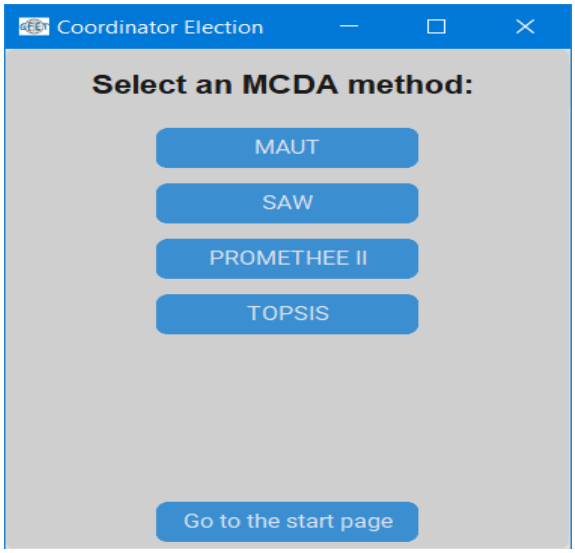

Figure 34 MCDA Method Selection Frame

The tool switches to the results frame when the user clicks on either SAW or TOPSIS button. However, when the MAUT button is clicked, the frame presented in Figure 35 is shown. This frame allows the user to change the default marginal utility functions and the Step function parameter for each criterion.

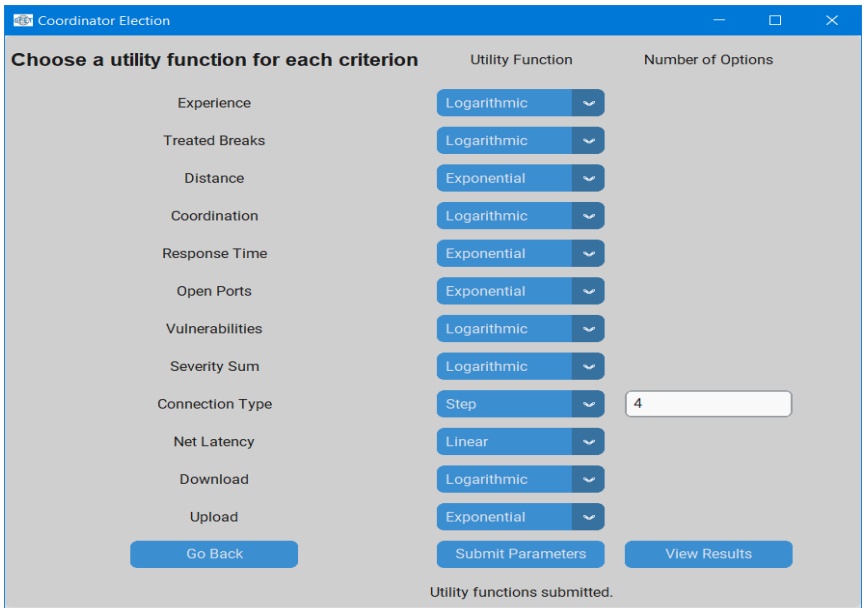

Figure 35 MAUT Utility Function and Parameters Selection Frame

The PROMETHEE II button takes the user to the frame shown in Figure 36. This frame allows the user to modify the default preference function and its parameters for each criterion.

By default, the functions and parameters used in our approach are already entered in both PROMETHEE II and MAUT frames. When the user submits the functions and their parameters, he can then view the results by clicking on the View Results button.

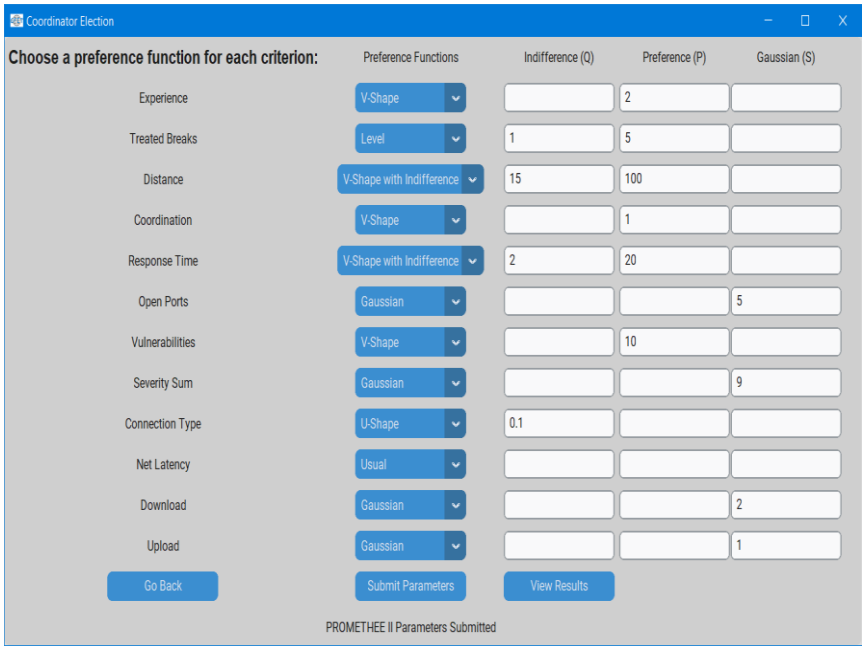

Figure 36 PROMETHEE II Preference Functions and Parameters Selection Frame

After submitting the parameters, results can be seen by clicking “View Results”. The results frame shown in Figure 37 displays the obtained results after applying the MAUT method on our dataset. The results frame of any MCDA method displays two graphs rendered using the Matplotlib library. In addition, the DMs scores are displayed in the bottom. The graph on the right shows the outranking of the DMs from best to worst based on the method score. The graph on the left presents the radar chart of the criteria satisfaction for the DM who ranked first using the chosen MCDA method. The user can go back and select a different MCDA method to see each method’s outcome using the same input data.

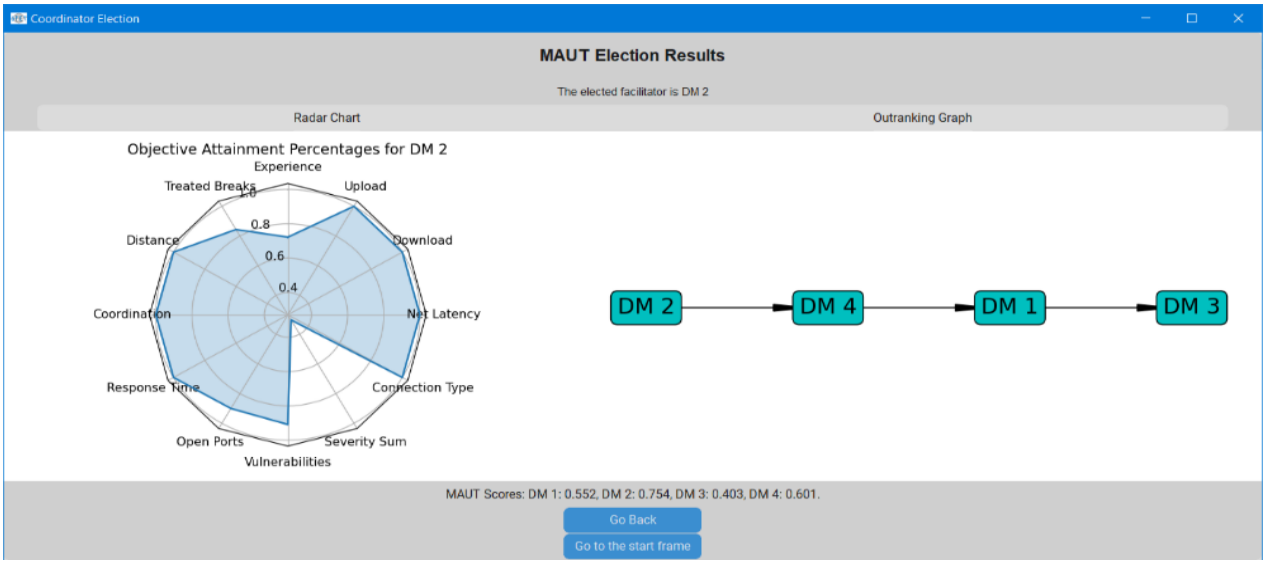

Figure 37 Results Frame After Applying MAUT

Figure 38 displays the results frame after applying the TOPSIS method. In this figure, we can see that TOPSIS gave a different ranking of the experts when compared to MAUT. Consequently, the left graph shows the performance of expert 1 instead of expert 2.

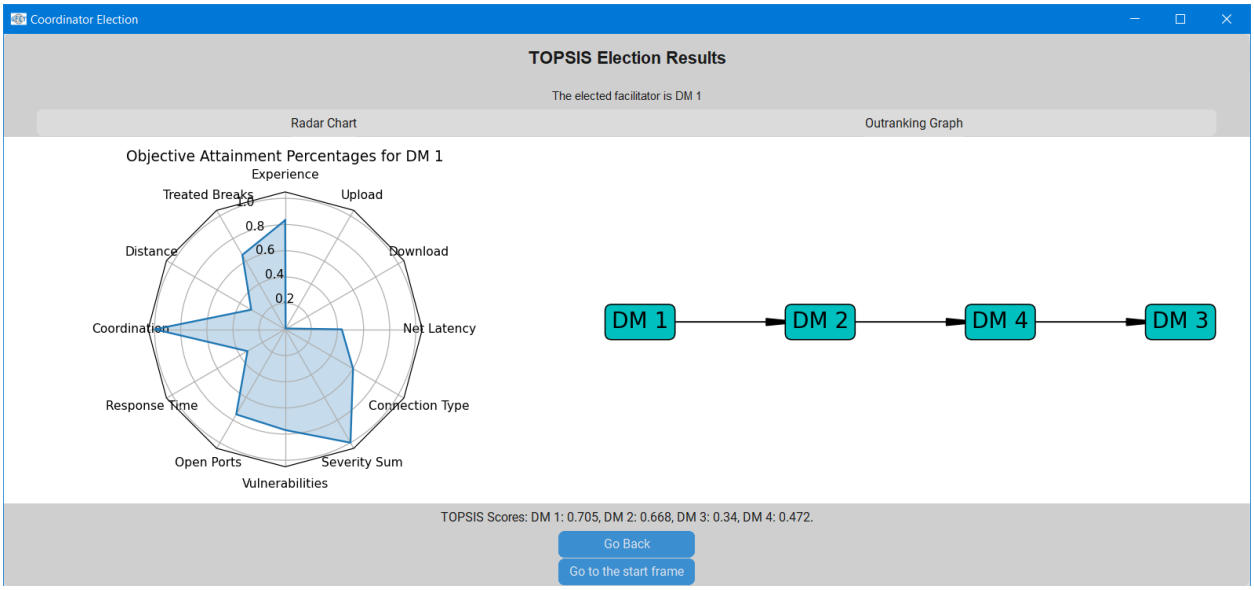

Figure 38 Results Frame after Applying TOPSIS

4.7. Empirical Assessment of GFEA

The proposed algorithm was tested on the case of collaborative e-maintenance process in industry. The collaborative e-maintenance system is a distributed system that connects multiple experts distant from each other with the aim to enable the proper coordination of their tasks, and output a solution in the form of a list containing the repairing actions [106]. Similarly, the GDSS also connects several DMs distributed geographically in order to reach a consensus. By analogy, GDSS can be used on the collaborative e-maintenance process to conduct the decision-making process resulting in choosing the corrective actions to repair a broken industrial machine. Furthermore, the collaborative e-maintenance requires choosing one the experts as the coordinator [106], who has similar tasks to the GDSS facilitator. Since they are both responsible for preparing and coordinating the group meeting.

When a breakdown happens, the technician on site notifies the expertise center. The center then selects a number of experts who are usually geographically distant from the site. The number of selected experts depends on the number of fields and the severity of the breakdown. During our case study, an industrial machine stopped functioning correctly. Consequently, the expertise center invited 6 experts to diagnose and provide a list of repairing actions to the technician on site. However, a coordinator is required to proceed with the maintenance process. Here the coordinator is also going to be the facilitator of the GDSS. The six experts were evaluated based on the 12 election criteria, which resulted in constructing the following performance matrix Table 21.

Table 21 Second Performance Matrix

	Experience as DM	Treated Breaks	Distance	Coordination	Response Time	Open Ports	Vulnerabilities	Severity Sum	Connection Type	Net Latency	Download	Upload
DM 1	12	8	500	2	17	20	16	32	1	13	100	10
DM 2	37	30	3938	19	20	15	14	10	0.8	366	25	2.5
DM 3	44	23	4401	13	19	18	20	22	0.6	486	50	5
DM 4	29	11	3477	5	18	9	22	31	0.3	198	20	2
DM 5	38	10	5029	3	17	11	10	19	1	103	500	200
DM 6	24	18	1846	9	21	13	18	28	0.8	232	20	2

4.7.1. Comparison of Objective Weighting Methods

Table 2 shows the execution time in milliseconds (ms) of five objective weighting methods on our performance matrix (Table 3) using their python implementations. We used the "objective_weighting" python package as an implementation of the methods. Figure 39 illustrates the obtained weights from every method using a bar chart.

Table 22 Execution time of objective weighting methods python implementation

Method	MEREC	CILOS	IDOCRIW	CRITIC	ANGLE	Entropy
Execution Time (ms)	0.97	0.96	0.97	7.01	0.96	1

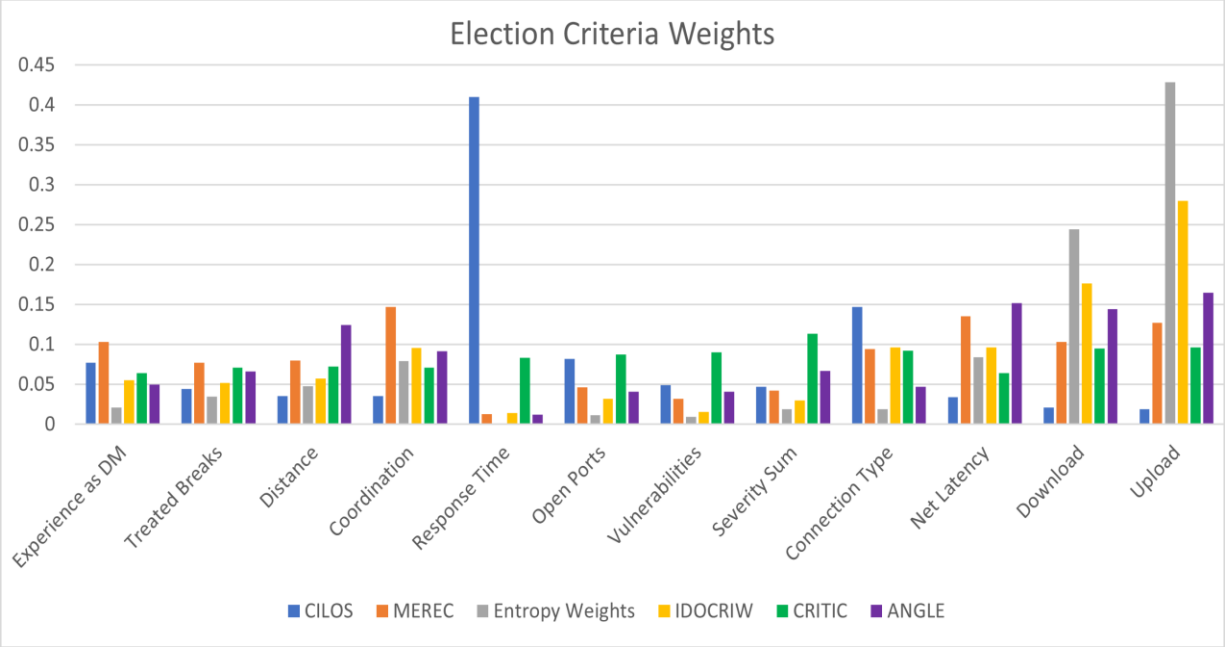

Figure 39 Bar chart showing election criteria weights using different objective weighting methods

MEREC was applied on the performance matrix. As a result, the weights corresponding to each election criterion were fixed. The importance of each criterion is presented in Table 23 and is illustrated in Figure 40 in the form of a bar chart.

Table 23 Election Criteria Weights Using MEREC

Criteria	Experience as DM	Treated Breaks	Distance	Coordination	Response Time	Open Ports	Vulnerabilities	Severity Sum	Connection Type	Net Latency	Download	Upload
MEREC Weights	0.103	0.077	0.08	0.147	0.013	0.046	0.032	0.042	0.094	0.135	0.103	0.127

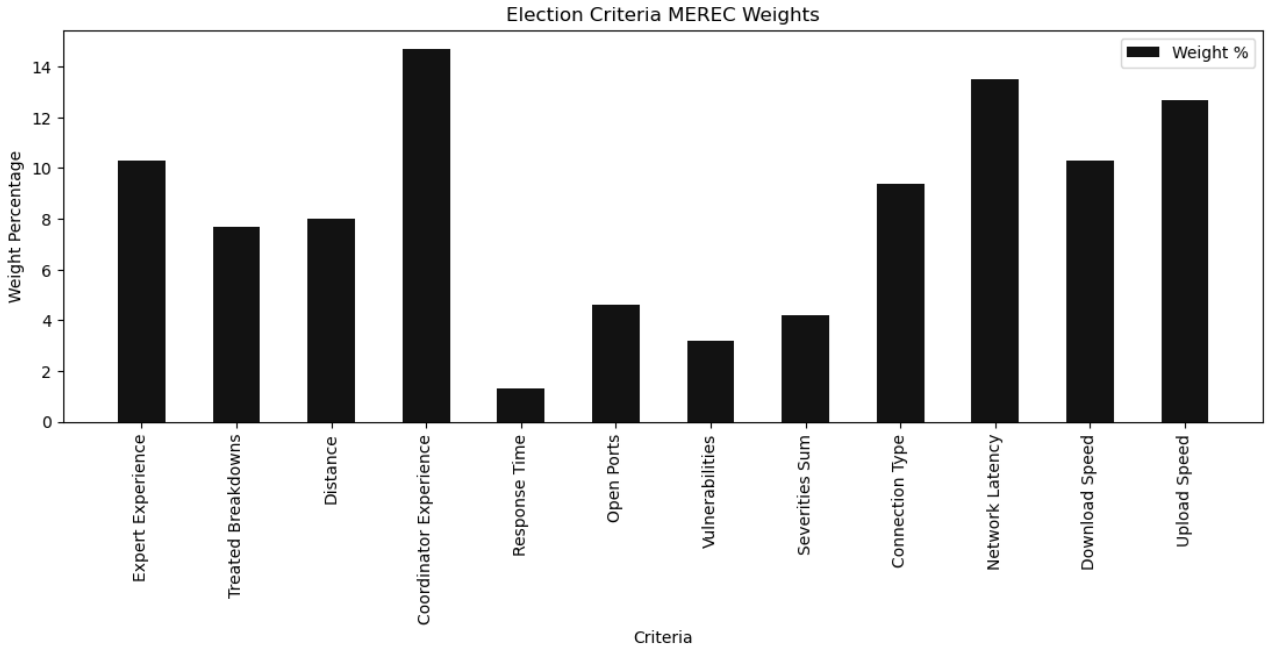

Figure 40 Election Criteria Importance Using MEREC

From Table 23 and Figure 40 it is observed that applying MEREC resulted in assigning the coordination experience criterion greater importance than all other election criteria.

4.7.2. Execution of GFEA

In this case study, it was the first time that the expert group was formed. Meaning that the coordinator (leader) role hasn't been assigned to any expert yet. Consequently, the DM who was the first one to join the group session is the initiator of the leader election. He is also given the smallest UID (DM 1). Here, each node represents the machine of a decision-maker. DM1 sorts the criteria in ascending order based on their weights. Next, he creates an initiation message InitMsg(1, 17, 1) and sends it to the adjacent node next to him following clockwise direction which is DM4. The original initiation message contains the UID of the initiator node 1 and his performance on the least important criterion, which is the expert's response time criterion.

When DM 4 receives the initiation message from DM 1, he checks whether the received value of the first criterion is smaller than or equal to his own value. DM 4 value is greater than the received one. So, he forwards the received message to the node next to him (DM 6). This process is repeated for all other nodes except the initiator. When DM 5 receives the election message from DM 2, he forwards the message to DM 1 despite it having the best value, because DM 1 precedes DM 5 in the ring. This can be observed from Figure 41.

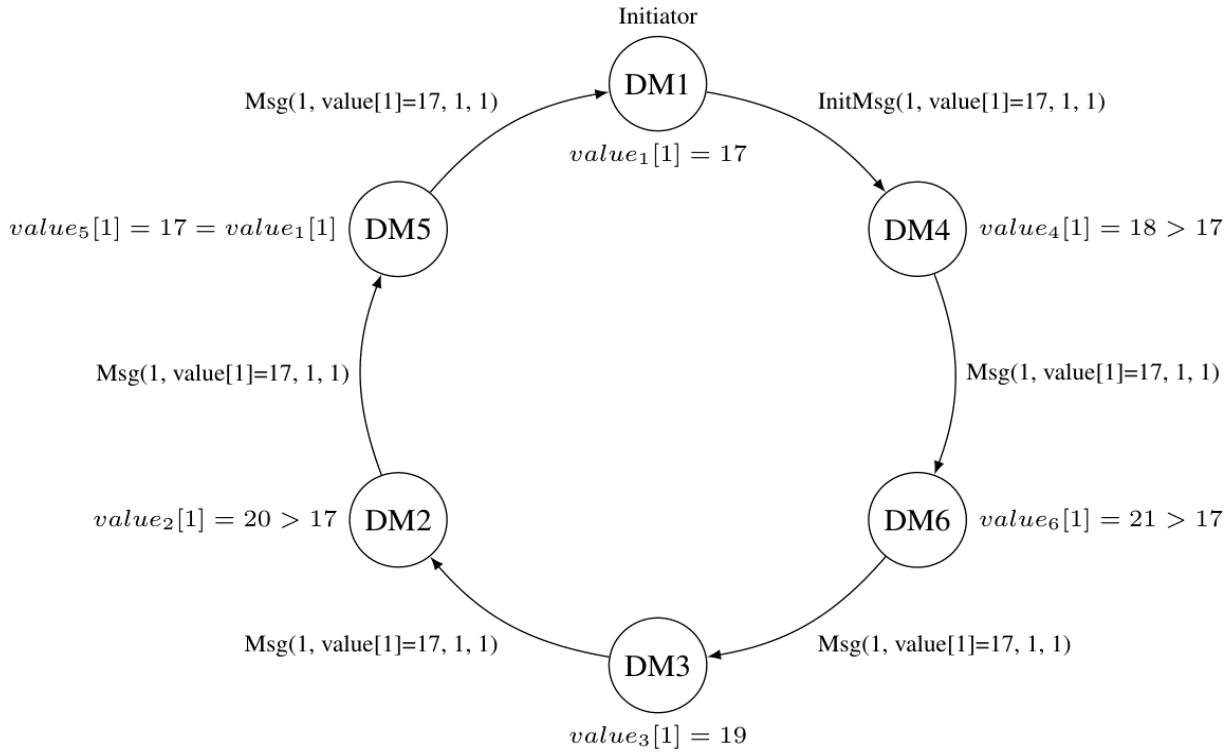

Figure 41: Election Initiation and First Election Round

Eventually, the message reaches DM 1, and he adds the weight of "Expert Response Time" 0.013 to his own score. The score of DM 1 then becomes 0.013. Afterwards, the initiator moves to the next more important criterion. Handling the second round of election which concerns the 2nd least important criterion (number of vulnerabilities) is illustrated in Figure 49. In this round, when DM 2 receives the message initiated by DM 1, it compares the value within the message with its own value of the second criterion. DM 2 has a better value so it creates a new message by setting the best value holder parameter to his own UID (2). He updates the best value for that criterion to 14. Then, he sends the updated message Msg (2, 14, 2) to the next node (DM 5). When DM 5 receives the new message from DM 2, it creates a new message with its own UID and value Msg(5,

10, 2) , then sends it to the initiator. At the end of this round DM 1 adds the weight of “number of vulnerabilities” to DM 5 score. This process is repeated for all 12 election criteria.

Figures Figure 42 to Figure 48 demonstrate the execution of GFEA using mpi4py on the second performance matrix. The algorithm was simulated on a local machine using mpi4py. The machine has a 64-bit Intel core i5 6600HQ processor and 16GB of RAM.

```
G:\Projects_SSD\MPI Py>mpiexec -np 6 py gfea_ring.py
Round: 1
0 sent a message to 3
3 received message from 0
3 sent a message to 5
5 received message from 3
5 sent a message to 2
2 received message from 5
2 sent a message to 1
1 received message from 2
1 sent a message to 4
4 received message from 1
4 sent a message to 0
0 received message from 4
Winner of round 1 is: 3
Round: 2
0 sent a message
3 received message from 0
3 sent a message to 5
5 received message from 3
5 sent a message to 2
2 received message from 5
2 sent a message to 1
1 received message from 2
1 sent a message to 4
4 received message from 1
4 sent a message to 0
0 received message from 4
Winner of round 2 is: 2
```

Figure 42 Mpi4Py Initiation of GFEA and Execution of Rounds One and Two

```
Round: 3
0 sent a message
3 received message from 0
3 sent a message to 5
5 received message from 3
5 sent a message to 2
2 received message from 5
2 sent a message to 1
1 received message from 2
1 sent a message to 4
0 received message from 4
Winner of round 3 is: 1
Round: 4
0 sent a message
3 received message from 0
3 sent a message to 5
4 received message from 1
4 sent a message to 0
5 received message from 3
5 sent a message to 2
2 received message from 5
2 sent a message to 1
1 received message from 2
1 sent a message to 4
0 received message from 4
Winner of round 4 is: 2
```

Figure 43 Mpi4Py Execution of Rounds Three and Four

```
Round: 5
0 sent a message
3 received message from 0
3 sent a message to 5
4 received message from 1
4 sent a message to 0
5 received message from 3
5 sent a message to 2
2 received message from 5
2 sent a message to 1
1 received message from 2
1 sent a message to 4
0 received message from 4
Winner of round 5 is: 1
Round: 6
0 sent a message
3 received message from 0
3 sent a message to 5
4 received message from 1
4 sent a message to 0
5 received message from 3
5 sent a message to 2
2 received message from 5
2 sent a message to 1
1 received message from 2
1 sent a message to 4
4 received message from 1
4 sent a message to 0
0 received message from 4
Winner of round 6 is: 4
```

Figure 44 Execution of rounds five and six

```
Round: 7
0 sent a message
3 received message from 0
3 sent a message to 5
5 received message from 3
5 sent a message to 2
2 received message from 5
2 sent a message to 1
1 received message from 2
1 sent a message to 4
0 received message from 4
Winner of round 7 is: 5
Round: 8
0 sent a message
4 received message from 1
4 sent a message to 0
3 received message from 0
3 sent a message to 5
5 received message from 3
5 sent a message to 2
2 received message from 5
2 sent a message to 1
1 received message from 2
1 sent a message to 4
0 received message from 4
Winner of round 8 is: 2
```

Figure 45 Mpi4Py Execution of Rounds Seven and Eight

```
Round: 9
0 sent a message
3 received message from 0
3 sent a message to 5
4 received message from 1
4 sent a message to 0
5 received message from 3
5 sent a message to 2
2 received message from 5
2 sent a message to 1
1 received message from 2
1 sent a message to 4
4 received message from 1
4 sent a message to 0
0 received message from 4
Winner of round 9 is: 1
Round: 10
0 sent a message
3 received message from 0
3 sent a message to 5
5 received message from 3
5 sent a message to 2
2 received message from 5
2 sent a message to 1
1 received message from 2
1 sent a message to 4
4 received message from 1
4 sent a message to 0
0 received message from 4
Winner of round 10 is: 1
```

Figure 46 Mpi4Py Execution of Rounds Nine and Ten

```
Round: 11
0 sent a message
3 received message from 0
3 sent a message to 5
5 received message from 3
5 sent a message to 2
2 received message from 5
2 sent a message to 1
1 received message from 2
1 sent a message to 4
4 received message from 1
4 sent a message to 0
0 received message from 4
Winner of round 11 is: 5
Round: 12
0 sent a message
3 received message from 0
3 sent a message to 5
5 received message from 3
5 sent a message to 2
2 received message from 5
2 sent a message to 1
1 received message from 2
1 sent a message to 4
4 received message from 1
4 sent a message to 0
0 received message from 4
Winner of round 12 is: 5
```

Figure 47 Mpi4Py Execution of Rounds Eleven and Twelve

```
GFEA Score of every DM:
 [0.322 0.266 0.103 0.046 0.262 0.   ]
Leader is: 1 and Backup is: 2
5 received leader message from 3
5 sent a leader message to [[2]]
6 Leader is: 1 Backup is: 2
1 received leader message from 2
1 sent a leader message to [[4]]
2 Leader is: 1 Backup is: 2
2 received leader message from 5
2 sent a leader message to [[1]]
3 Leader is: 1 Backup is: 2
3 received leader message from 0
3 sent a leader message to [[5]]
4 Leader is: 1 Backup is: 2
4 received leader message from 1
4 sent a leader message to [[0]]
5 Leader is: 1 Backup is: 2
0 received message from 4
Election time : 140.6947998329997 ms
Number of messages : 154
Attempting to use an MPI routine after finalizing MPI
```

Figure 48 Leader Announcement Phase

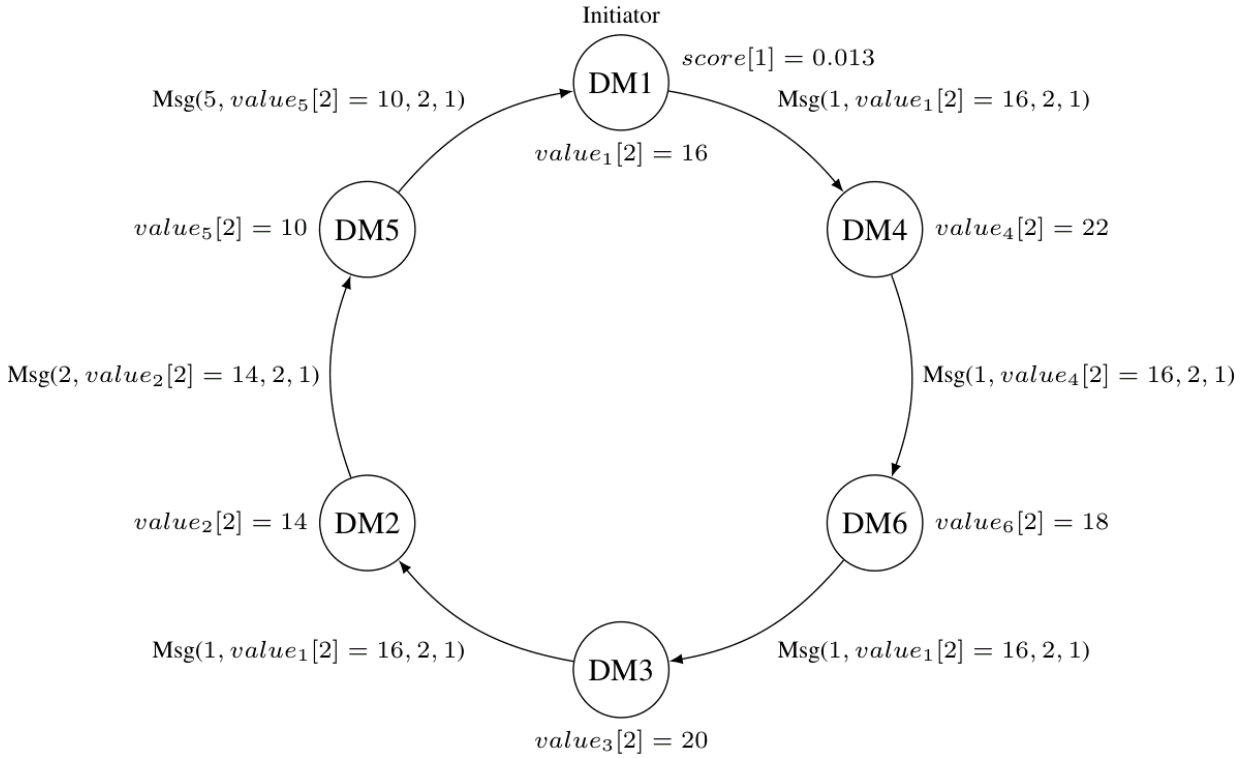

Figure 49: Second Round of the Election

When the initiator receives the message of the final criterion Msg(2, 19, 12) which is the most important criterion, it adds the criterion weight 0.147 to the score of DM 2. In addition, it selects itself as the leader and DM 2 as the backup leader because they have the highest and second-highest scores. Next, it sends their UIDs in a leader announcement message leader_msg(1, 2). This phase is illustrated in Figure 50.

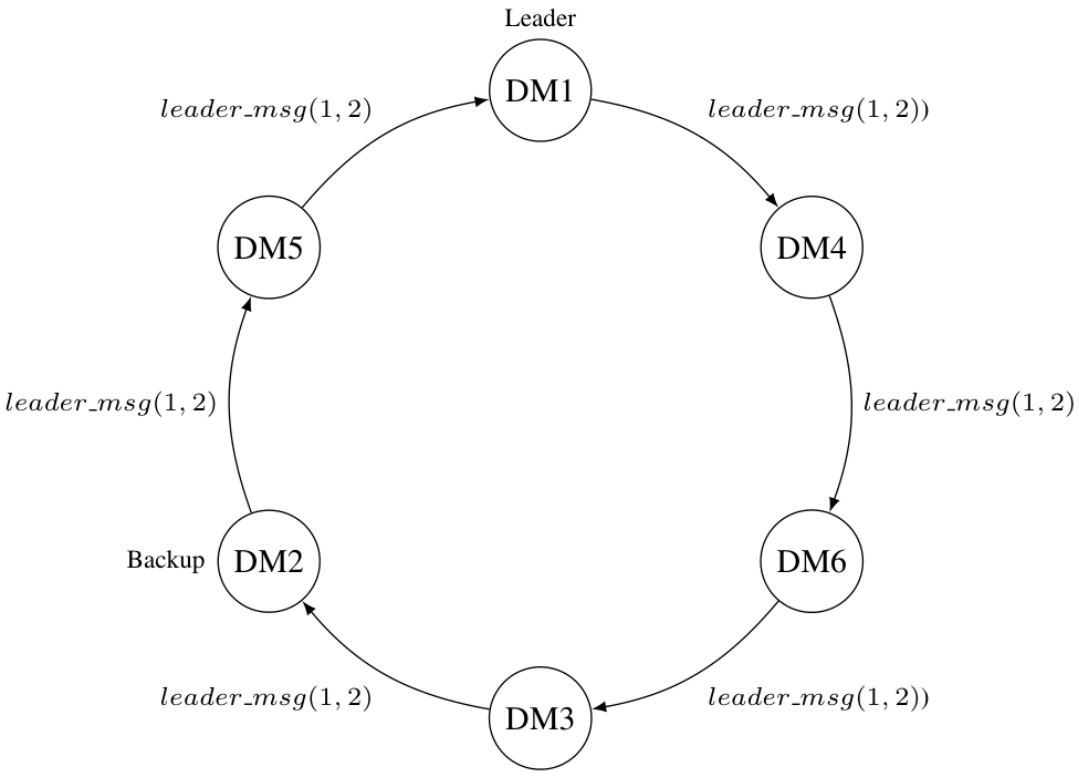

Figure 50: Leader & Backup Leader Announcement

The final score of each DM is shown in Table 24. The elected facilitator is DM 1 who was initially the initiator, and the backup leader is DM 2 who got the second highest score. The initiator ranked the DMs including itself based on their score. This ranking is presented in Table 25.

If classical ring algorithms such as LeLann, LCR and HS that base their election on the UID were used on our case study, then DM 6 would be elected because he has the highest UID. However, DM 6 had a score of zero, and he is considered the worst expert based on GFEA. This is because he doesn't have the best value in any of the 12 election criteria. Consequently, this shows that relying solely on the UID can give the worst results.

Table 24 Algorithm Score for Each DM

Decision Maker UID	DM 1	DM 2	DM 3	DM 4	DM 5	DM 6
Algorithm Score	0.322	0.266	0.103	0.046	0.262	0

Table 25 Ranking of the DMs Based on GFEA Score

Rank	1	2	3	4	5	6
Decision Maker	DM 1	DM 2	DM 5	DM 3	DM 4	DM 6

4.7.3. Leader Failure Scenario

This subsection illustrates an example scenario in which the leader DM 1 loses his connection to the network. In this case the next adjacent node DM 4 doesn't receive a heartbeat message from the leader for a period of time exceeding the threshold. Consequently, DM 4 considers that the leader has failed. Next, DM 4 sends a leader failure message leader_failure_msg(2) containing the UID of the new leader DM 2 (backup) who replaces the facilitator automatically without executing the election algorithm again (see Figure 51). When DM4 receives the leader failure message, it discards it. Finally, each node keeps sending a periodic heartbeat to its next node (see Figure 52).

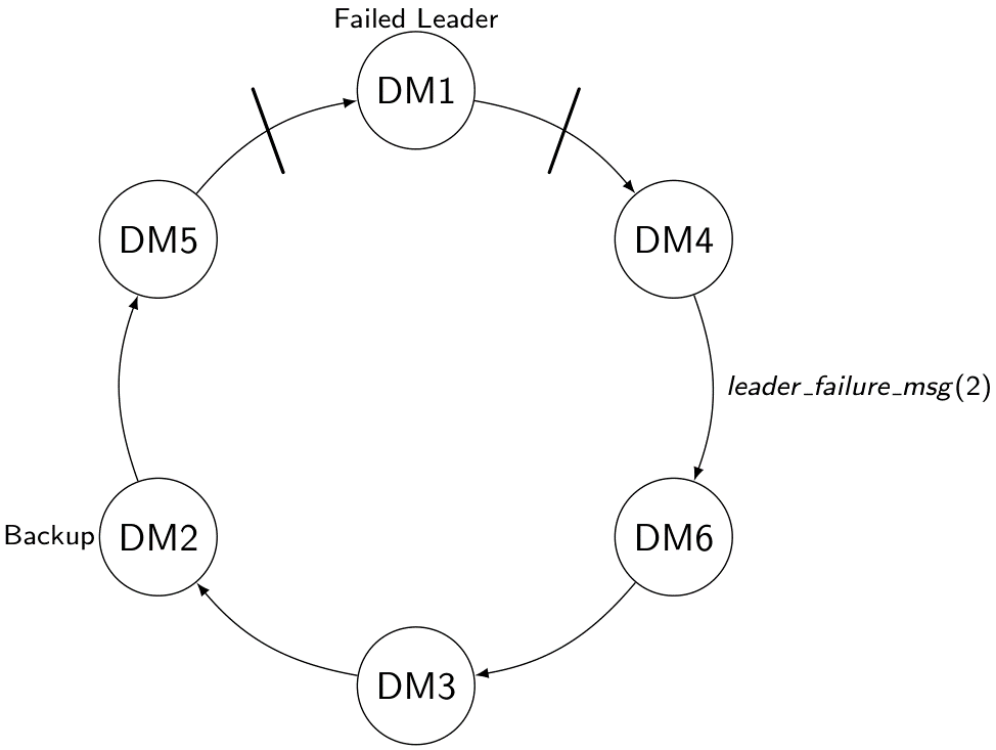

Figure 51 Leader Disconnects from the Network

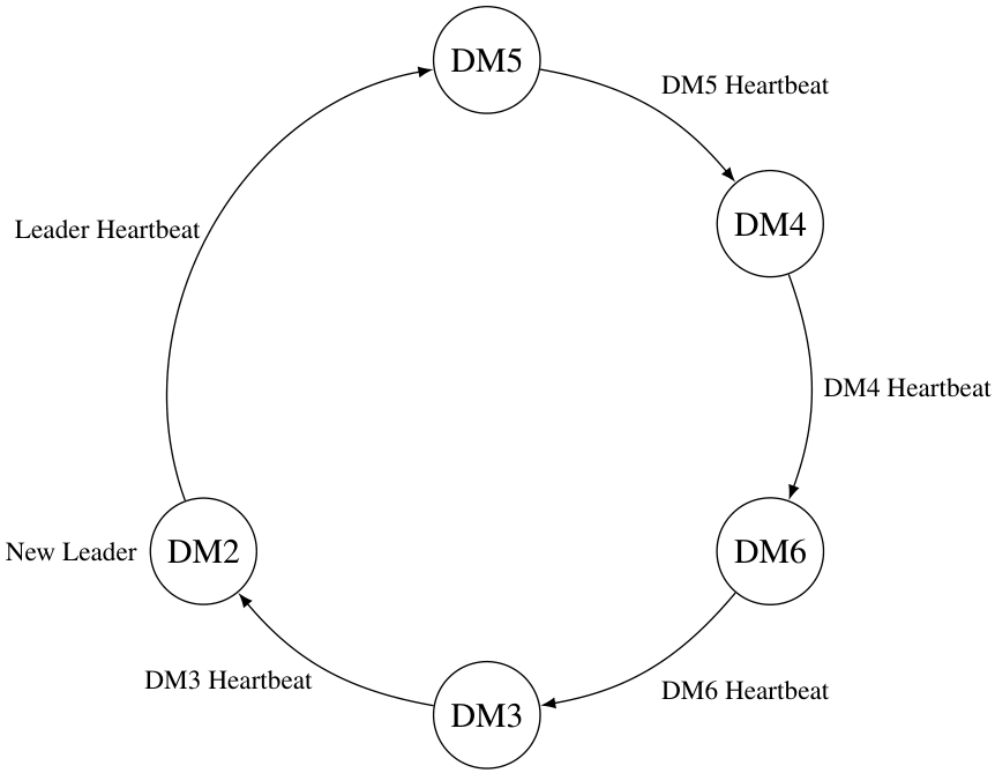

Figure 52 Backup Replaces the Failed Leader

4.7.4. Initiator Failure Scenario

In the case that the initiator DM 1 would fail during the election, then DM 4 would detect the initiator failure after not receiving its heartbeat for a period exceeding the heartbeat timeout period. Consequently, DM 4 changes its state to become the new initiator, and creates a new election initiation message containing its UID, its own value of the least important criterion and its UID as the election initiator, in order to inform other nodes. Figure 53 illustrates this scenario using our performance matrix.

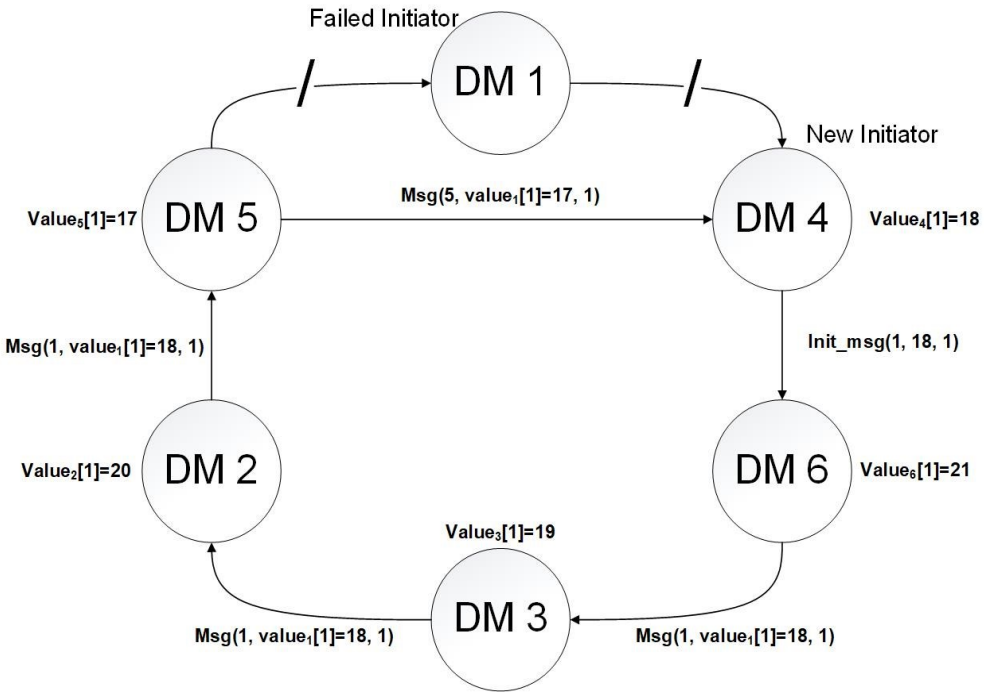

Figure 53 DM 4 Replaces the Failed Initiator

4.7.5. Initiator Recovery Scenario

In this scenario, we assume that the initiator (DM 1) failed during round six which concerns the distance. If DM 1 recovers during the same round in which it failed, then when the new initiator DM 4 receives the election message coming from the recovered initiator, it changes its state to DM and forwards the message to DM 6. Figure 54 illustrates this scenario.

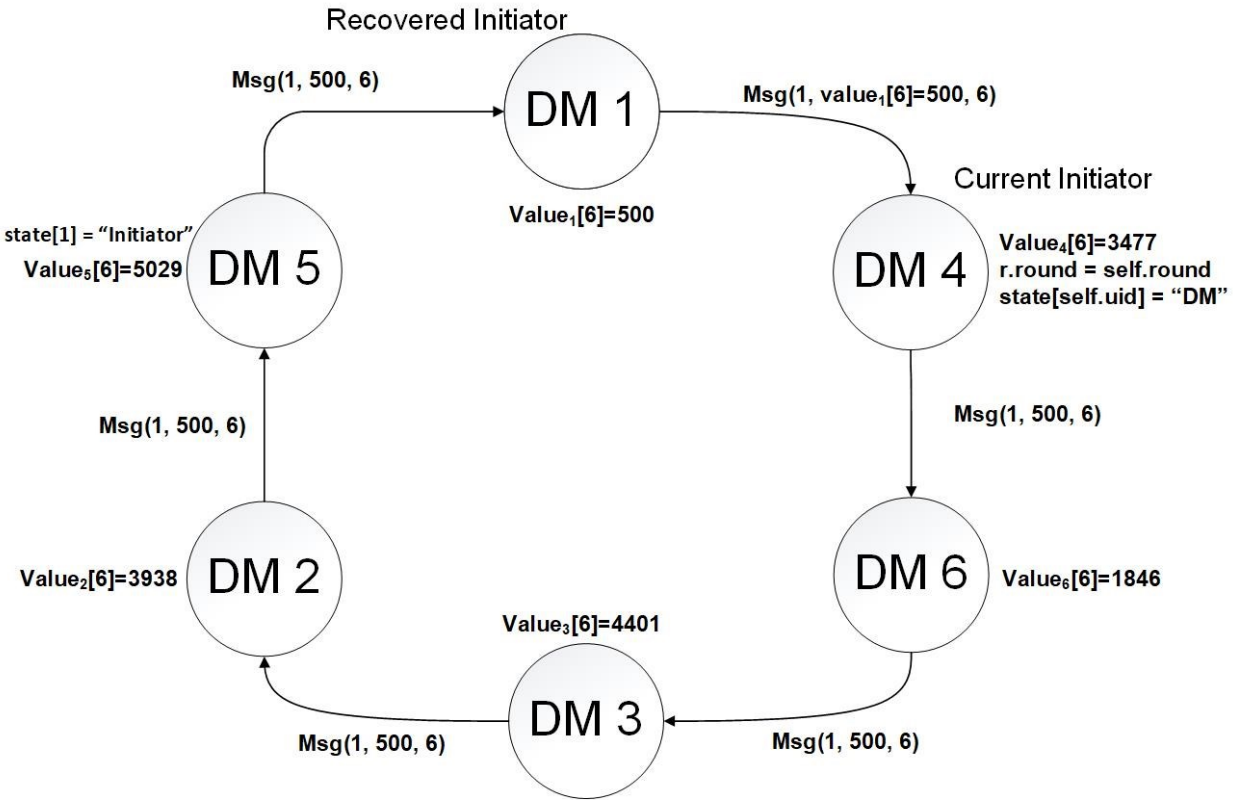

Figure 54 Initiator Recovers During the Same Failure Round

Now let's suppose that DM 1 has failed during the first round which concerns the response time criterion. If DM 1 recovers during round 6 that concerns the distance criterion. Then when the new initiator DM 4 receives the election message holding the round number 1 coming from DM 1, it discards that message and carry on with the current election. When DM 1 receives a message containing the initiator UID of DM 4, it changes its state to DM and continue to participate in the current election. Figure 55 illustrates this scenario.

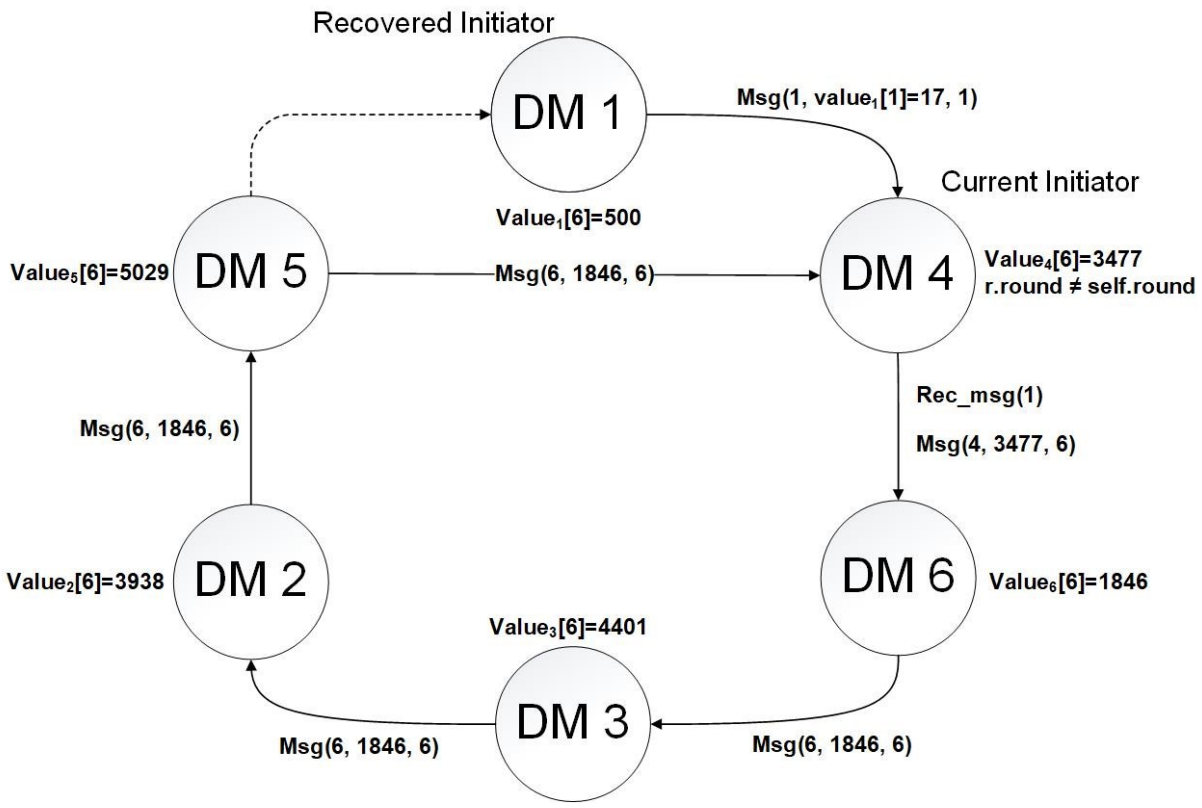

Figure 55 Initiator Recovery in Another Round

4.7.6. Results & Discussions

From Figure 56 and Table 21 it is seen that the elected leader (DM 1) is the closest DM to the breakdown site. This can come handy if an expert physical presence is required on site or if he loses his connection to the GDSS. Plus, it reduces the cost of time and travel fees for the expert to arrive at the site. Similar to DM 5, he also has the best response time. Additionally, DM 1 has the least network lag (13 ms), which allows him to work with other DMs almost in real-time. This is mainly because of two factors: he has the best type of internet connection (fiber optic) and he has the shortest distance to the breakdown site [107]. However, in the security category, DM 1 doesn't perform well compared to other DMs. Which means that his machine makes the GDSS vulnerable to confidential information leaks and malicious modifications.

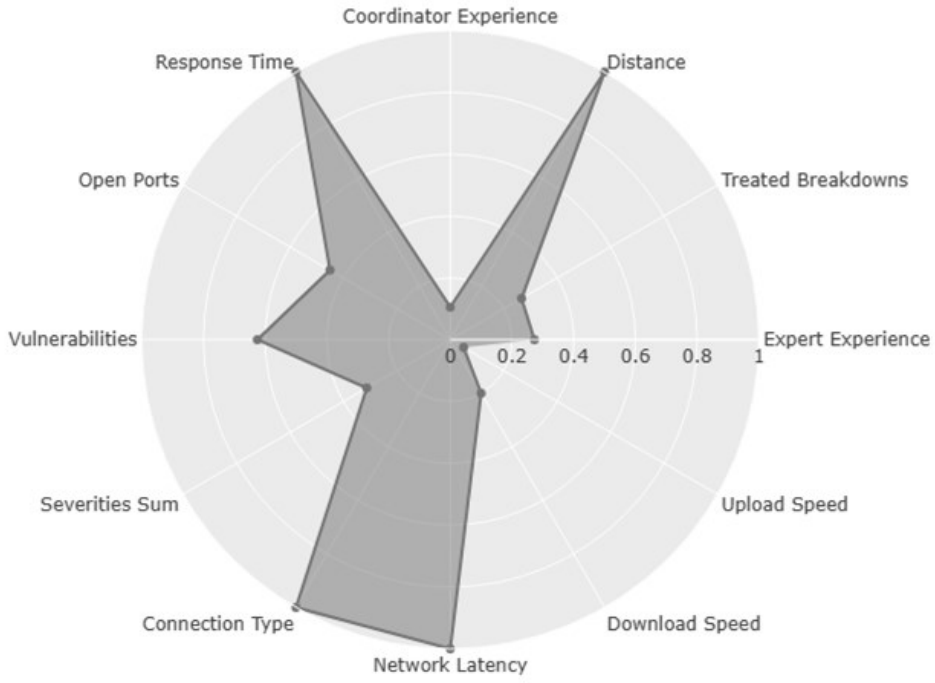

Figure 56 Radar Chart Showing Normalized Dataset Values of Expert 1

Based on Figure 57 and Table 21, the backup leader (DM 2) has the most coordination experience, ensuring a well-planned meeting, better handling of conflicts and highlighting each DM's opinion using the right questions. Furthermore, DM 2 has participated the most in treating industrial breakdowns and has been an industrial expert longer than DM 1, giving him an edge when it comes to technical questions, fetching required information from database and breakdown diagnosis. Additionally, in the security category, DM 2 performs better than DM 1. Since his machine has fewer vulnerabilities with less severity and fewer open ports. Which means that his machine makes the GDSS less vulnerable to malicious attacks. However, both his download and upload speeds are considerably slower when compared to other experts like DM 5. This can be time consuming when uploading or downloading files. On top of this, his network delay is slightly high, resulting in a lag during real time audio and video communications with the technician and with other DMs. Moreover, DM 2 has an average response time when asked to join a decision-making session.

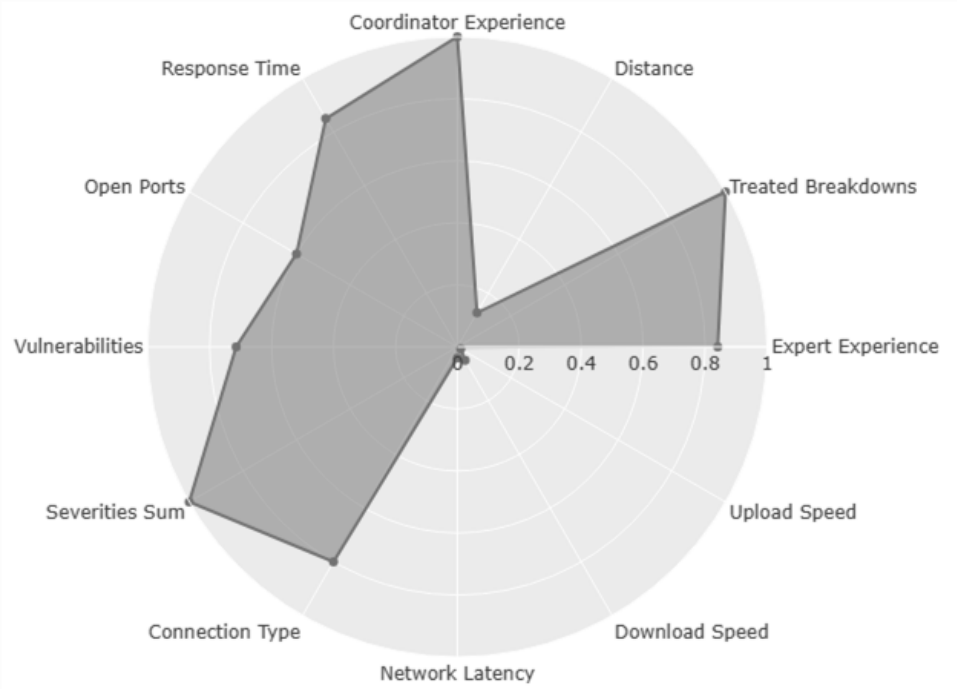

Figure 57 Radar Chart Showing Normalized Dataset Values of Expert 2

4.7.7. Brief Comparison with Other Ring Election Algorithms

Table 26 compares GFEA with other classic ring election algorithms in terms of execution time, number of exchanged messages and maximum resident memory. These algorithms were coded in Python using the library of mpi4py. These algorithms were run using Microsoft MPI implementation which is based on MPICH. The tests were conducted on a local machine composed of a 64-bit Intel 6600HQ processor, and 16 GB of RAM.

Table 26 Performance Comparison with Classic Ring Election Algorithms

Election Algorithm	Execution Time for Six Nodes (ms)	Number of Exchanged Messages	Max Resident Memory (MB)
GFEA	131	154	35
Ring	175	24	34.56
LCR	213	34	34.79
Bully	21	96	18.38

From Table 26 we can see that GFEA is faster than Ring and LCR, however, it is slower than Bully. GFEA also consumes more memory and exchanges more messages than other algorithms. This is mainly because GFEA goes through 12 election rounds as it considers multiple election criteria rather than just basing the election on the UID like the Ring, LCR and Bully.

Ring election algorithms have a similar memory consumption, while Bully consumes the least amount of RAM. It is noticeable that Bully's execution is much faster than other algorithms. The reason for this is that the implementation used for Bully didn't include probing or a timeout period to get a response from higher ranked nodes. In addition, the implementation didn't simulate any node failure, so all nodes were responding as soon as they receive a message. Moreover, Bully was implemented on a complete graph topology, thus a node can send multiple messages to multiple nodes at a time instead of forwarding a message from one node to the next in order to reach the destination.

Table 27 compares the proposed GFEA algorithm with modern election algorithms that exist in the literature. The comparison is based on the functionality it satisfies, time complexity of the worst case in Big O(n) notation, the message complexity in the worst case, UID of the elected leader and UID of the backup.

Table 27 Comparison of Results with Existing Algorithms

Algorithm	Satisfied Functionality	Worst Time Complexity	Worst Message Complexity	Elected Leader	Backup Leader
GFEA	All	$O(mn^2)$	$O(mn^2)$	DM1	DM2
Ring	None	$O(n^2)$	$O(n)$	DM6	No
LCR	None	$O(n^2)$	$O(n)$	DM6	No
Bully	None	$O(1)$	$O(n^2)$	DM6	No
FRLLE [68]	Recovery	$O(n)$	$O(n^2)$	DM1	No
Top-K [74]	Backup	Not Specified	Not Specified	DM6	DM3
SEALEA [72]	Backup	Not Specified	Not Specified	DM6	DM3
DRLEF [75]	Backup	Not Specified	Not Specified	DM3	DM5
Privacy-Preserved [69]	Tie-break	Not specified	O(d(G).n.p)	DM4	No
Multi-attribute [108]	Multi-criteria	O(n.D)	O(n.l)	DM2	No

If LeLann, LCR, Bully or HS algorithms [80–83] were applied on our case study, then DM 6 would have been elected as the leader. Despite DM 6 not having the best value in any criterion and having the worst score (zero score), he would still be elected in UID based algorithms, because he has the biggest UID. Additionally, if the smallest UID was considered as the election criterion, then DM 1 who's the most suitable expert for the role, would win the election. However, this approach is not reliable, as the UID in our case indicates the joining order. Meaning that if DM 6 was the first to join the session, then he would be elected as the leader despite him being the worst fit for the leader role. Consequently, the UID alone cannot be the determining factor for the suitability of a DM for the leader role. Additionally, in GFEA, only one node can detect the leader failure, and only one node can initiate the election at a time. This is a considerable advantage when knowing that in classical ring election algorithms multiple nodes can detect the leader failure and initiate multiple elections at the same time [81].

4.8. Conclusion

Having to transport foreign experts to the breakdown site costs both time and money. It can also be dangerous in some situations. With the advancements made in the information and communication technologies, the experts' interventions can be done remotely through internet. But the challenge relies in the necessity for a group of experts to collaborate together remotely. This issue can be solved by using a GDSS which will aid the experts in collaborating and making a decision without having to move to the breakdown site.

The GDSS requires organization and coordination, usually done by the means of a human facilitator who's one of the decision makers. His counterpart in the collaborative e-maintenance process is the coordinator, who has several important tasks on hand. This means that the chosen expert to be the coordinator has to be competent and qualified for the position. This qualification is evaluated based on several criteria which constitute the election criteria. In order to consider all criteria, we used the multi-criteria approach as a solution. We fixed the criteria by eliciting the experts' preferences through the DELPHI method.

In this chapter, we presented the process of collaborative e-maintenance in which the coordinator plays a similar role to that of the GDSS facilitator. The coordinator's role in collaborative e-maintenance involves around coordinating and managing multiple distant experts with the aim of leading them to reach a consensus about a plan of repairing actions. We illustrated the coordinator's missions through a BPMN model and we presented some related works.

The previously proposed multi-criteria approach was tested on a performance matrix composed of four experts. AHP with the aggregation procedure AIP were used to assign weights to the election criteria. Next, a comparison was made between MAUT, SAW, PROMETHEE and TOPSIS in the context of ranking the experts from best to worse by merit to take on the facilitator role. Authors found that PROMETHEE II was more suitable for this problematic when compared to the other methods. PROMETHEE II had a better execution time and is easier to use by the DMs when compared to MAUT. Additionally, we proposed the software tool GFET, which is designed to facilitate the election of the facilitator.

On the other hand, the proposed algorithm GFEA was also evaluated in the same field. The proposed algorithm was applied on another performance matrix consisting of six experts. MEREC

was applied on the second performance matrix to obtain the criteria weights. This approach resulted in electing the expert who excels in four criteria, and reserving a backup that excels in three criteria. If the election was solely based on the maximum UID like in the classical ring algorithms, then this would result in choosing the expert that doesn't satisfy any criterion.

Nonetheless, the proposed algorithm has certain limitations. Its robustness compromises performance and adds network overhead. GFEA is much slower and uses much more messages when compared to other ring algorithms such as Ring, LCR and FRLLE. Mainly because of considering multiple election criteria and taking into account the failure of the initiator. However, the performance can be improved by optimizing the algorithm and reducing its time and message complexity.

Conclusion and Perspectives

Certain decision problems require the intervention of multiple decision-makers or stakeholder that aren't necessary located in the same room. These decision-makers can be geographically dispersed in different countries. Such case is the process of collaborative e-maintenance. This process is started when an industrial machine stops functioning, thus requiring urgent repair to keep the production running. This usually requires calling remote experts from different fields. In order to support such cases, a solution would be to use a GDSS as cooperative work system. This system involves decision-makers who are coordinated and aided by a human facilitator. The GDSS facilitator is responsible for several tasks and is crucial to conduct the decision-making session.

As the problematic of electing a GDSS facilitator involves more than one criterion, it can be considered a multi-criteria decision problem. This led us to consider employing MCDA methods. Consequently, in this research we proposed a new multi-criteria approach to elect a GDSS facilitator. First, we conducted a Delphi study to gather criteria that are considered to be useful in choosing the appropriate DM for the facilitator role. The 12 election criteria fall into three categories: experience, security and network performance. Next, chose the collaborative e-maintenance field as a case study to test the proposed approach. We applied four MCDA methods to a performance matrix consisting of four experts that were involved in a breakdown repair. The four methods that were used are MAUT, SAW, PROMETHEE II and TOPSIS. After using each method, we compared their results based on nine metrics (time complexity, execution time, memory footprint, number of satisfied criteria, objectives attainment percentage, number of parameters, DM involvement, elected leader and backup leader). These metrics reflect the performance of the method and the quality of the decision. Based on the obtained results from our case study, PROMETHEE II seemed to be faster and simpler to use when compared to MAUT. Another contribution we made, is the development of a new election tool called GFET. This tool is intended to be used by the expertise center. We proposed to integrate this tool in the collaborative e-maintenance model using a UML collaboration diagram. GFET automates the proposed multi-criteria approach and makes it easier to use through a user-friendly graphical interface.

Despite the multi-criteria approach being suitable for this kind of problematic. It still has some flaws including being vulnerable to biases and human behavior problems. This flaws stem

from the subjective side of MCDA methods, as they require subjective parameters and input from human DMs. During this research, we found that the problematic of electing a leader node in a distributed system and electing a GDSS facilitator share multiple similarities. Due to these similarities and the formality of election algorithms, we considered using an election algorithm to elect the GDSS facilitator.

Existing leader election algorithms often consider one criterion which is the UID of the node. Other works that involves multiple election criteria don't use a formal weighting method. Consequently, we proposed a new algorithm called GFEA which is designed with the main objective of election a GDSS facilitator. The proposed algorithm chooses a leader based on multiple election criteria. These criteria are weighted using MEREC which is an objective weighting method that only takes the performance matrix and criteria type as an input. Moreover, GFEA always reserves a backup node in case the leader fails or gets disconnected. Furthermore, it can handle the initiator failure and recovery. This shows the robustness of the proposed algorithm. We compared GFEA with related works in terms of functionality. The comparison shows that no other distributed election algorithm integrates all the features required by our problematic.

Perspectives and Future Work

Regarding the multi-criteria approach, an interesting path would be to explore the use of fuzzy variations of MCDA methods. Another perspective, is to broaden the criteria elicitation study to include multiple companies and fields from different countries which would result in a more comprehensive and generalized set of election criteria. Furthermore, the GFET tool should validated by conducting usability tests with maintenance experts.

We identify several areas for future development of the GFEA algorithm. First, its performance should be evaluated against other algorithms, comparing factors like election time and number of exchanged messages, especially in large-scale networks. Second, a comparison should be conducted between distributed election algorithms and multi-criteria approach for selecting a GDSS facilitator. To further validate and refine the GFEA algorithm, real-world expert feedback is crucial. Additionally, the algorithm's adaptability to different network topologies warrants investigation. Moreover, future work should consider how to accommodate the addition

of new decision-makers within the network during the election process. Furthermore, the multi-criteria approach should be compared to the distributed election approach. Finally, other objective methods should be tested with GFEA in order to see how changing the weights affects the final ranking of the DMs.

References

1. Laredj, M.A., Bouamrane, K.: Workflow specification for interaction management between experts in a cooperative remote diagnosis process. Computer Science and Information Systems. 8, 573–590 (2011). https://doi.org/10.2298/csis1003260011.

2. DeSanctis, G., Gallupe, R.B.: FOUNDATION FOR THE STUDY OF GROUP DECISION SUPPORT SYSTEMS. Manage Sci. 33, 589–609 (1987). https://doi.org/10.1287/mnsc.33.5.589.

3. Gu, W., Moustafa, A., Ito, T., Zhang, M., Yang, C.: A Case-based Reasoning Approach for Supporting Facilitation in Online Discussions. Group Decis Negot. 30, 719–742 (2021). https://doi.org/10.1007/s10726-021-09731-4.

4. Clawson, V.K., Bostrom, R.P.: The importance of facilitator role behaviors in different face to face group support systems environments. In: Proceedings of the Twenty-Eighth Hawaii International Conference on System Sciences. pp. 181–190. IEEE Comput. Soc. Press (1995). https://doi.org/10.1109/HICSS.1995.375731.

5. Zhang, G., Chen, J., Zhang, Y., Liu, C.: Research of Asynchronous Leader Election Algorithm on Hierarchy Ad Hoc Network. In: 2009 5th International Conference on Wireless Communications, Networking and Mobile Computing. pp. 1–4. IEEE (2009). https://doi.org/10.1109/WICOM.2009.5301989.

6. Konak, A., Coit, D.W., Smith, A.E.: Multi-objective optimization using genetic algorithms: A tutorial. Reliab Eng Syst Saf. 91, 992–1007 (2006). https://doi.org/10.1016/j.ress.2005.11.018.

7. Nurmi, H.: Voting Procedures: A Summary Analysis. Br J Polit Sci. 13, 181–208 (1983). https://doi.org/10.1017/S0007123400003215.

8. Basílio, M.P., Pereira, V., Costa, H.G., Santos, M., Ghosh, A.: A Systematic Review of the Applications of Multi-Criteria Decision Aid Methods (1977–2022), (2022). https://doi.org/10.3390/electronics11111720.

9. Laredj, A., Rouba, B., Duvallet, C.: Multi-criteria decision aid for group facilitator election: Application to a collaborative e-maintenance process. International Journal of Decision Support System Technology. 11, 93–102 (2019). https://doi.org/10.4018/IJDSST.2019010105.

10. Carneiro, J., Alves, P., Marreiros, G., Novais, P.: Group decision support systems for current times: Overcoming the challenges of dispersed group decision-making. Neurocomputing. 423, 735–746 (2021). https://doi.org/10.1016/j.neucom.2020.04.100.

11. Zahedi, F.M.: Group Decision Making. In: Encyclopedia of Operations Research and Management Science. pp. 668–677. Springer US, Boston, MA (2013). https://doi.org/10.1007/978-1-4419-1153-7_406.

12. Brahm, C., Kleiner, B.H.: Advantages and disadvantages of group decision- making approaches. Team Performance Management: An International Journal. 2, 30–35 (1996). https://doi.org/10.1108/13527599610105538.

13. Islam, R.: Group decision making through nominal group technique: an empirical study. J. for International Business and Entrepreneurship Development. 5, 134 (2010). https://doi.org/10.1504/JIBED.2010.036998.

14. Mukherjee, N., Hugé, J., Sutherland, W.J., Mcneill, J., Van Opstal, M., Dahdouh-Guebas, F., Koedam, N.: The Delphi technique in ecology and biological conservation: Applications and guidelines. Methods Ecol Evol. 6, 1097–1109 (2015). https://doi.org/10.1111/2041-210X.12387.

15. Nurmi, H.: Voting Procedures: A Summary Analysis. Br J Polit Sci. 13, 181–208 (1983). https://doi.org/10.1017/S0007123400003215.

16. Coulibaly, A., Zarate, P., Camilleri, G., Konate, J., Tangara, F.: A Voting Procedures Recommender System for Decision-Making. In: Lecture Notes in Business Information Processing. pp. 80–91. Springer Verlag (2019). https://doi.org/10.1007/978-3-030-21711-2_7.

17. Lee, D.: Game theory and neural basis of social decision making. Nat Neurosci. 11, 404–9 (2008). https://doi.org/10.1038/nn2065.

18. Bannon, L.J.: CSCW-A challenge to certain (G)DSS perspectives on the role of decisions, information, and technology in organizations? Bannon (1997).

19. Adla, A.: Aidè a la Facilitation pour une prise de Décision Collective : Proposition d'un Modèle et d'un Outil, https://tel.archives-ouvertes.fr/tel-00514908, (2010).

20. Kraemer, K.L., King, J.L.: Computer-based systems for cooperative work and group decision making. ACM Comput Surv. 20, 115–146 (1988). https://doi.org/10.1145/46157.46158.

21. Rees, J., Koehler, G.J.: Modeling search in group decision support systems. IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews. 34, 237–244 (2004). https://doi.org/10.1109/TSMCC.2004.829307.

22. Dennis A. R., George J. F., Nunamaker J. F., Valacich J. S., Vogel D. R.: ELECTRONIC MEETING SYSTEMS TO SUPPORT GROUP WORK. Commun ACM. 34, 40–61 (1991).

23. Chung, J.K.H.: Improving the implementation of VM studies in Hong Kong's construction industry-A GDSS Approach Intelligent Solutions for Dependable Built Environment View project Frontiers in Construction Management View project. In: Proceedings of SAVE Annual Conference. pp. 83–93. Ed. Vogl OJ, Society of American Value Engineers, Florida (2001).

24. Zaraté, P., Konate, J., Camilleri, G.: Collaborative Decision Making Tools : A Comparative Study Based on Functionalities. In: 13th International Conference Group Decision and Negotiation (GDN 2013). pp. 111–122 (2013).

25. Dhauj, F.: Strengthening Collaborative Mind Map Tool To Facilitate Requirement Elicitation Process. Journal of Mathematical Modelling and Applied Computing. (2024).

26. Mayer, I., De Jong, M.: Combining GDSS and Gaming for Decision Support. (2004).

27. Nunamaker, J.F., Briggs, R.O., Mittleman, D.D., Vogel, D.R., Pierre, B.A.: Lessons from a Dozen Years of Group Support Systems Research: A Discussion of Lab and Field Findings. Journal of Management Information Systems. 13, 163–207 (1996). https://doi.org/10.1080/07421222.1996.11518138.

28. Yao, J., Wang, J.: Group Support Systems as Tools for HR Decision Making. In: Strategic Information Systems. pp. 1087–1095. IGI Global (2010). https://doi.org/10.4018/978-1-60566-677-8.ch071.

29. Thorpe, T., Thorpe, S.J.: Online Facilitator Competencies for Group Facilitators. (2016).

30. Palha, R.P., Zarate, P., de Almeida, A.T., Nurmi, H.: Choosing a voting procedure for the GDSS GRUS. In: Lecture Notes in Business Information Processing. pp. 163–174. Springer Verlag (2017). https://doi.org/10.1007/978-3-319-63546-0_12.

31. Razmak, J., Aouni, B.: Decision support system and multi-criteria decision aid: A state of the art and perspectives. Journal of Multi-Criteria Decision Analysis. 22, 101–117 (2015). https://doi.org/10.1002/mcda.1530.

32. Salo, A., Hämäläinen, R.P., Lahtinen, T.J.: Multicriteria Methods for Group Decision Processes: An Overview. In: Handbook of Group Decision and Negotiation. pp. 863–891. Springer, Cham, Cham (2021). https://doi.org/10.1007/978-3-030-49629-6_16.

33. Matsatsinis, N.F., Samaras, A.P.: MCDA and preference disaggregation in group decision support systems. Eur J Oper Res. 130, 414–429 (2001). https://doi.org/10.1016/S0377-2217(00)00038-2.

34. Youcef Omari, Djamila Hamdadou, Mohammed Amine Mami: Assigning Decision Makers' Weights Using TOPSIS Method In Spatial Multicriteria Group Decision Support System. Journal of Positive School Psychology. 6, 8602–8609 (2022).

35. Omari, Y., Hamdadou, D., Mami, M.A.: Towards an intelligent agent-based multi-criteria group decision support system : A case study in land use management. International Journal of Computing and Digital Systems. 13, 303–325 (2023). https://doi.org/10.12785/ijcds/130125.

36. Amin, M.M., Sutrisman, A., Dwitayanti, Y.: Group decision support system model to determine supervisor lecturers for student creativity programs. Bulletin of Electrical Engineering and Informatics. 12, 2484–2494 (2023). https://doi.org/10.11591/eei.v12i4.4784.

37. Meyer, P., Roubens, M.: Choice, Ranking and Sorting in Fuzzy Multiple Criteria Decision Aid. In: Multiple Criteria Decision Analysis: State of the Art Surveys. pp. 471–503. Springer-Verlag, New York. https://doi.org/10.1007/0-387-23081-5_12.

38. Guarini, M.R., Battisti, F., Chiovitti, A.: Public initiatives of settlement transformation: A theoretical-methodological approach to selecting tools of multi-criteria decision analysis. Buildings. 8, (2017). https://doi.org/10.3390/buildings8010001.

39. Saaty, T.L.: The Analytic Hierarchy Process: Decision Making in Complex Environments. In: Quantitative Assessment in Arms Control. pp. 285–308. Springer US, Boston, MA (1984). https://doi.org/10.1007/978-1-4613-2805-6_12.

40. Papathanasiou, J., Ploskas, N.: Multiple Criteria Decision Aid: Methods, Examples and Python Implementations. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-91648-4.

41. Regunathan, R., Murugaiyan, A., Lavanya, K.: A QoS-Aware Hybrid TOPSIS–Plurality Method for Multi-criteria Decision Model in Mobile Cloud Service Selection. In: Proceedings of the 2nd International Conference on Data Engineering and Communication Technology: ICDECT 2017. pp. 499–507. Springer Singapore (2019). https://doi.org/10.1007/978-981-13-1610-4_50.

42. Ossadnik, W., Schinke, S., Kaspar, R.H.: Group Aggregation Techniques for Analytic Hierarchy Process and Analytic Network Process: A Comparative Analysis. Group Decis Negot. 25, 421–457 (2016). https://doi.org/10.1007/s10726-015-9448-4.

43. Ishizaka, A., Nemery, P.: Multi-attribute utility theory. In: Multi-Criteria Decision Analysis. pp. 81–113. John Wiley & Sons Ltd, Chichester, UK (2013). https://doi.org/10.1002/9781118644898.ch4.

44. Alinezhad, A., Khalili, J.: New Methods and Applications in Multiple Attribute Decision Making (MADM). Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-15009-9.

45. Panjaitan, M.I.: Simple Additive Weighting (SAW) method in Determining Beneficiaries of Foundation Benefits. Jl. Kol. Yos Sudarso No.45 AB. 13, 19–25 (2019).

46. Brans, J.-P., De Smet, Y.: PROMETHEE Methods. In: Multiple Criteria Decision Analysis. International Series in Operations Research & Management Science. pp. 187–219. Springer, New York, NY (2016). https://doi.org/10.1007/978-1-4939-3094-4_6.

47. Fernandez, A., Zaraté, P., Gardey, J.C., Bosetti, G.: Supporting multi-criteria decision-making across websites: the Logikós approach. Cent Eur J Oper Res. 29, 201–225 (2021). https://doi.org/10.1007/s10100-020-00723-4.

48. Costa, C.A.B. e, De Corte, J.-M., Vansnick, J.-C., London School of Economics and Political Science Department of Operational Research: MACBETH. LSE, Dep. of Operational Research, London, UK (2003).

49. Nemdili, A.K., Hamdadou, D.: Modeling of an active multi-agent environment for the design of a multi-criteria group decision support system. Multiagent and Grid Systems. 17, 83–111 (2021). https://doi.org/10.3233/MGS-210344.

50. van Steen, M., Tanenbaum, A.S.: A brief introduction to distributed systems. Computing. 98, 967–1009 (2016). https://doi.org/10.1007/s00607-016-0508-7.

51. Min Huang, Bode, B.: A Performance Comparison of Tree and Ring Topologies in Distributed Systems. In: 19th IEEE International Parallel and Distributed Processing Symposium. pp. 258a–258a. IEEE. https://doi.org/10.1109/IPDPS.2005.57.

52. Fornerón Martínez, J.T., Agostini, F., La Red Martínez, D.L.: Resource and Process Management With a Decision Model Based on Fuzzy Logic. International Journal of Interactive Multimedia and Artificial Intelligence. 8, 134–149 (2023). https://doi.org/10.9781/ijimai.2023.02.009.

53. Coulouris, G., Dollimore, J., Tim: Distributed Systems: Concepts and Design. (2011).

54. Kshemkalyani, A.D., Singhal, M.: Distributed Computing. Cambridge University Press (2008). https://doi.org/10.1017/CBO9780511805318.

55. Shajil Kumar P. A., R. Srinivasa Rao Kunte: ABCD Analysis of Industries Using High Performance Computing. International Journal of Case Studies in Business, IT, and Education. 7, 2581–6942 (2023). https://doi.org/https://doi.org/10.5281/zenodo.8125105.

56. Saluja, S.K., Tiwari, T.: DIFFERENT COMPUTING TECHNOLOGIES AND VIRTUALIZATION. BSSS Journal of Computer. 14, 47–53 (2023). https://doi.org/10.51767/JC1407.

57. Bakhouya, M., Gaber, J.: Approaches for engineering adaptive systems in ubiquitous and pervasive environments. J Reliab Intell Environ. 1, 75–86 (2015). https://doi.org/10.1007/s40860-015-0010-6.

58. Alnawayseh, S.E.A., Khan, T.A., Farooq, U., Zulfiqar, S., Khan, S., Al-Kassem, A.H.: Research Challenges and Future Facet Of Cellular Computing. In: 2023 International Conference on Business Analytics for Technology and Security (ICBATS). pp. 1–8. IEEE (2023). https://doi.org/10.1109/ICBATS57792.2023.10111407.

59. Salama, R., Altrjman, C., Al-Turjman, F.: A Survey of the Architectures and Protocols for Wireless Sensor Networks and Wireless Multimedia Sensor Networks. NEU journal for artificial intelligence and internet of things. 2, (2023).

60. Guettala, M., Bourekkache, S., Kazar, O.: Ubiquitous learning a new challenge of ubiquitous computing: state of the art. In: 2021 International Conference on Information Systems and Advanced Technologies (ICISAT). pp. 1–5. IEEE (2021). https://doi.org/10.1109/ICISAT54145.2021.9678434.

61. Minar, N.: Distributed Systems Topologies: Part 2 Seven Evaluation Properties. In: Emerging Technology Conference. O'Reilly (2002).

62. Barrick, S.D.: Distributed data acquisition system for substation bus protection and monitoring. In: Wescon/98. Conference Proceedings (Cat. No.98CH36265). pp. 315–320. IEEE (1998). https://doi.org/10.1109/WESCON.1998.716474.

63. Azman, M., Panicker, J.G., Kashyap, R.: Wireless Daisy Chain and Tree Topology Networks for Smart Cities. In: 2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT). pp. 1–6. IEEE (2019). https://doi.org/10.1109/ICECCT.2019.8869252.

64. Ghate, P.V., Pati, H.K.: Collaborative distributed communication in heterogeneous environments: A comprehensive survey. Journal of Network and Computer Applications. 61, 1–20 (2016). https://doi.org/10.1016/j.jnca.2015.10.006.

65. Lusk, E., Gropp, W.: The MPI Message-Passing Interface Standard: Overview and Status. Presented at the (1995). https://doi.org/10.1016/S0927-5452(06)80017-1.

66. Fink, Z., Liu, S., Choi, J., Diener, M., Kale, L. V.: Performance Evaluation of Python Parallel Programming Models: Charm4Py and mpi4py. In: 2021 IEEE/ACM 6th International Workshop on Extreme Scale Programming Models and Middleware (ESPM2). pp. 38–44. IEEE (2021). https://doi.org/10.1109/ESPM254806.2021.00010.

67. So, K.C.W., Sirer, E.G.: Latency and bandwidth-minimizing failure detectors. In: Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007. pp. 89–99. ACM, New York, NY, USA (2007). https://doi.org/10.1145/1272996.1273008.

68. Biswas, A., Maurya, A.K., Tripathi, A.K., Aknine, S.: FRLLE: a failure rate and load-based leader election algorithm for a bidirectional ring in distributed systems. J Supercomput. 77, 751–779 (2021). https://doi.org/10.1007/s11227-020-03286-y.

69. Sperling, L., Kulkarni, S.S.: Privacy-Preserving Methods for Outlier-Resistant Average Consensus and Shallow Ranked Vote Leader Election. (2023). https://doi.org/https://doi.org/10.48550/arXiv.2301.11882.

70. Jiang, F., Cheng, Y., Dong, C., Yu, E.: A Novel Weight-based Leader Election Approach for Split Brain in Distributed System. In: IOP Conference Series: Materials Science and Engineering. p. 012005. Institute of Physics Publishing, Wuhan, China (2020). https://doi.org/10.1088/1757-899X/719/1/012005.

71. Luo, Y., Chen, Y., Chen, Q., Liang, Q.: A new election algorithm for DPos consensus mechanism in blockchain. In: Proceedings - 7th International Conference on Digital Home, ICDH 2018. pp. 116–120. Institute of Electrical and Electronics Engineers Inc., Guilin, China (2019). https://doi.org/10.1109/ICDH.2018.00029.

72. Haddar, M.A.: SEALEA: Scalable and Energy Aware k-Leaders Election Algorithm in IoT Wireless Sensor Networks. Wirel Pers Commun. 125, 209–229 (2022). https://doi.org/10.1007/s11277-022-09547-8.

73. Cahng, H.C., Lo, C.C.: A consensus-based leader election algorithm for wireless ad hoc networks. In: Proceedings - 2012 International Symposium on Computer, Consumer and Control, IS3C 2012. pp. 232–235. IEEE, Taichung, Taiwan (2012). https://doi.org/10.1109/IS3C.2012.66.

74. Raychoudhury, V., Cao, J., Niyogi, R., Wu, W., Lai, Y.: Top K-leader election in mobile ad hoc networks. Pervasive Mob Comput. 13, 181–202 (2014). https://doi.org/10.1016/j.pmcj.2013.10.003.

75. Elsakaan, N., Amroun, K.: Distributed and Reliable Leader Election Framework for Wireless Sensor Network (DRLEF). In: Lecture Notes in Networks and Systems. pp. 123–141. Springer Science and Business Media Deutschland GmbH (2022). https://doi.org/10.1007/978-3-030-95918-0_13.

76. Julian, A., Marian Jose, J.: Multi-criteria Leader Selection in Ad Hoc Networks Using Fuzzy Analytical Hierarchy Process. In: Lecture Notes in Electrical Engineering. pp. 2875–2885. Springer, Singapore (2021). https://doi.org/10.1007/978-981-15-8221-9_269.

77. Kadjouh, N., Bounceur, A., Bezoui, M., Khanouche, M.E., Euler, R., Hammoudeh, M., Lagadec, L., Jabbar, S., Al-Turjman, F.: A Dominating Tree Based Leader Election Algorithm for Smart Cities IoT Infrastructure. Mobile Networks and Applications. 28, 718–731 (2023). https://doi.org/10.1007/s11036-020-01599-z.

78. Favier, A., Arantes, L., Lejeune, J., Sens, P.: Centrality-Based Eventual Leader Election in Dynamic Networks. In: 2021 IEEE 20th International Symposium on Network Computing and Applications (NCA). pp. 1–8. IEEE, Boston, MA, USA (2021). https://doi.org/10.1109/NCA53618.2021.9685390.

79. Biswas, T., Bhardwaj, R., Ray, A.K., Kuila, P.: A novel leader election algorithm based on resources for ring networks. International Journal of Communication Systems. 31, (2018). https://doi.org/10.1002/dac.3583.

80. Le Lann, G.: DISTRIBUTED SYSTEMS-TOWARDS A FORMAL APPROACH. IFIP Congress. 7, 155–160 (1977).

81. Chang, E., Roberts, R.: An improved algorithm for decentralized extrema-finding in circular configurations of processes. Commun ACM. 22, 281–283 (1979). https://doi.org/10.1145/359104.359108.

82. Hirschberg, D.S., Sinclair, J.B.: Decentralized extrema-finding in circular configurations of processors. Commun ACM. 23, 627–628 (1980). https://doi.org/10.1145/359024.359029.

83. Garcia-Molina: Elections in a Distributed Computing System. IEEE Transactions on Computers. C–31, 48–59 (1982). https://doi.org/10.1109/TC.1982.1675885.

84. Ereifej, J.S.: Impact of Group Decision Support System (GDSS) on Organizational Decision Making in Telecommunication Sector in Jordan. We'Ken- International Journal of Basic and Applied Sciences. 2, 15 (2017). https://doi.org/10.21904/weken/2017/v2/i2/120588.

85. Zardari, N.H., Ahmed, K., Shirazi, S.M., Yusop, Z. Bin: Weighting Methods and their Effects on Multi-Criteria Decision Making Model Outcomes in Water Resources Management. Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-12586-2.

86. Odu, G.O.: Weighting methods for multi-criteria decision making technique. Journal of Applied Sciences and Environmental Management. 23, 1449 (2019). https://doi.org/10.4314/jasem.v23i8.7.

87. Keshavarz-Ghorabaee, M., Amiri, M., Zavadskas, E.K., Turskis, Z., Antucheviciene, J.: Determination of Objective Weights Using a New Method Based on the Removal Effects of Criteria (MEREC). Symmetry (Basel). 13, 525 (2021). https://doi.org/10.3390/sym13040525.

88. Ecer, F., Aycin, E.: Novel Comprehensive MEREC Weighting-Based Score Aggregation Model for Measuring Innovation Performance: The Case of G7 Countries. Informatica. 34, 53–83 (2023). https://doi.org/10.15388/22-INFOR494.

89. Choumas, K., Korakis, T.: On Using Raft Over Networks: Improving Leader Election. IEEE Transactions on Network and Service Management. 19, 1129–1141 (2022). https://doi.org/10.1109/TNSM.2022.3147958.

90. Molęda, M., Małysiak-Mrozek, B., Ding, W., Sunderam, V., Mrozek, D.: From Corrective to Predictive Maintenance—A Review of Maintenance Approaches for the Power Industry. Sensors. 23, 5970 (2023). https://doi.org/10.3390/s23135970.

91. Hedjazi, D., Zidani, A., Hedjazi, D., Zidani, A.: Development of an industrial e-maintenance system integrating groupware techniques. (2011).

92. Raza, S., Faheem, M., Guenes, M.: Industrial wireless sensor and actuator networks in industry 4.0: Exploring requirements, protocols, and challenges—A MAC survey. International Journal of Communication Systems. 32, (2019). https://doi.org/10.1002/dac.4074.

93. Hedjazi, D., Zidani, A.: Group awareness evaluation in e-maintenance Groupware system. In: The 1st International conference on artificial intelligence and information technology. , Ouargla, Algeria (2014).

94. Mukherjee, N., Zabala, A., Huge, J., Nyumba, T.O., Adem Esmail, B., Sutherland, W.J.: Comparison of techniques for eliciting views and judgements in decision-making. Methods Ecol Evol. 9, 54–63 (2018). https://doi.org/10.1111/2041-210X.12940.

95. Brown, B.B.: DELPHI PROCESS: A Methodology Used for the Elicitation of Opinions of Experts LI DELPHI PROCESS: A Methodology Used for the Elicitation of Opinions of Experts. (1968).

96. Hejblum, G., Lambotte, O., Galicier, L., Coppo, P., Marzac, C., Aumont, C., Fardet, L.: A web-based Delphi study for eliciting helpful criteria in the positive diagnosis of hemophagocytic syndrome in adult patients. PLoS One. 9, (2014). https://doi.org/10.1371/journal.pone.0094024.

97. Jetty, S., Rahalkar, S.: Securing Network Infrastructure: Discover practical network security with Nmap and Nessus 7. Packt Publishing Ltd (2019).

98. Mathew, K., Tabassum, M., Lu, M.V., Siok, A.: A Study Of Open Ports As Security Vulnerabilities In Common User Computers. (2014).

99. Mell, P., Spring, J., Dugal, D., Ananthakrishna, S., Casotto, F., Fridley, T., Ganas, C., Kundu, A., Nordwall, P., Pushpanathan, V., Sommerfeld, D., Tesauro, M., Turner, C.: Measuring the Common Vulnerability Scoring System base score equation. (2022). https://doi.org/10.6028/NIST.IR.8409.

100. Aksu, M.U., Dilek, M.H., Tatli, E.I., Bicakci, K., Dirik, H.I., Demirezen, M.U., Aykir, T.: A quantitative CVSS-based cyber security risk assessment methodology for IT systems. In: 2017 International Carnahan Conference on Security Technology (ICCST). pp. 1–8. IEEE (2017). https://doi.org/10.1109/CCST.2017.8167819.

101. Kim, J., Yi, J., Park, H.-H.: A case study on oscillating behavior of end-to-end network latency. In: The International Conference on Information Network 2012. pp. 512–516. IEEE (2012). https://doi.org/10.1109/ICOIN.2012.6164430.

102. Valdecy, P.: PyDecision, https://pypi.org/project/pyDecision/, last accessed 2023/03/02.

103. Serrai, W., Abdelli, A., Mokdad, L., Hammal, Y.: Towards an efficient and a more accurate web service selection using MCDM methods. J Comput Sci. 22, 253–267 (2017). https://doi.org/10.1016/j.jocs.2017.05.024.

104. Singh, A., Gupta, A., Mehra, A.: Best criteria selection based PROMETHEE II method. OPSEARCH. 58, 160–180 (2021). https://doi.org/10.1007/s12597-020-00464-7.

105. Wang, M., Lin, S.-J., Lo, Y.-C.: The comparison between MAUT and PROMETHEE. In: 2010 IEEE International Conference on Industrial Engineering and Engineering Management. pp. 753–757. IEEE (2010). https://doi.org/10.1109/IEEM.2010.5675608.

106. Hedjazi, D.: Constructing collective competence: a new CSCW-based approach. International Journal of Information and Communication Technology. 12, 393 (2018). https://doi.org/10.1504/IJICT.2018.090418.

107. Kovacevic, A., Heckmann, O., Liebau, N.C., Steinmetz, R.: Location awareness-improving distributed multimedia communication. Proceedings of the IEEE. 96, 131–142 (2008). https://doi.org/10.1109/JPROC.2007.909913.

108. Biswas, A., Singh, M., Baranwal, G., Tripathi, A.K., Aknine, S.: Multi-attribute-based self-stabilizing algorithm for leader election in distributed systems. J Supercomput. 81, 556 (2025). https://doi.org/10.1007/s11227-025-07043-x.

5. Appendix

5.2. Delphi Questionnaire

The actual questionnaire that was used was written in French and not English.

Last Name: First Name:

Email: Round:

When using a Group Decision Support System during the construction of a Plan of Repair Action, one expert must be elected as the group facilitator. In this study, we seek to determine the most relevant and important criteria for electing the group facilitator who can also be the coordinator of the process. For each proposed criterion, indicate whether it is useful or not in selecting the most suitable expert for the role of GDSS facilitator.

1- Experience as a decision-maker: Number of times the expert has participated in a group decision-making process (General)

Useful ☐ Not Useful ☐

2- Level of education (Bachelor, DEA, Engineering, Master, Magister, PhD).

Useful ☐ Not Useful ☐

3- Number of languages spoken.

Useful ☐ Not Useful ☐

4- Level of languages spoken (A1, A2, B1, B2, C1, C2).

Useful ☐ Not Useful ☐

5- Number of breakdowns treated: Number of times the expert participated in a successful collaborative maintenance process which resulted in fixing a failed machine.

Useful ☐ Not Useful ☐

6- Distance between the expert and the breakdown site (KM).

Useful ☐ Not Useful ☐

7- Coordination experience: Number of times the expert participated in the maintenance of an outage as a group coordinator.

Useful ☐ Not Useful ☐

8- Maximum number of participants coordinated.

Useful ☐ Not Useful ☐

9- Facilitation and coordination certificates.

Useful ☐ Not Useful ☐

10- Certificates or diplomas in computer science.

Useful ☐ Not Useful ☐

11- Expert response time (minutes).

Useful ☐ Not Useful ☐

12- Number of open ports on the expert's machine (security).

Useful ☐ Not Useful ☐

13- Number of vulnerabilities on the expert's machine (security)

Useful ☐ Not Useful ☐

14- Status of vulnerabilities found (Fixed or not fixed).

Useful ☐ Not Useful ☐

15- Sum of severities of vulnerabilities (CVSS) detected on the expert's machine (security).

Useful ☐ Not Useful ☐

16- Wi-Fi security protocol on an expert's router (WEP, WPA, WPA2 & WPA3).

Useful ☐ Not Useful ☐

17- Operating system used (Windows, Linux, or MacOS).

Useful ☐ Not Useful ☐

18- Operating system update date.

Useful ☐ Not Useful ☐

19- Operating system security update date.

Useful ☐ Not Useful ☐

20- Network type used (Public or private).

Useful ☐ Not Useful ☐

21- Connection type (Mobile/ADSL/Satellite/Fiber): This indicates network stability.

Useful ☐ Not Useful ☐

22- Connection interface (Ethernet or Wi-Fi).

Useful ☐ Not Useful ☐

23- Average network latency (ms): Causes delays during audio and video calls and sending messages.

Useful ☐ Not Useful ☐

24- Number of server hops to reach the GDSS server.

Useful ☐ Not Useful ☐

25- Download Speed (Mbps): Affects the quality of the audio and video received, as well as the time required to receive files.

Useful ☐ Not Useful ☐

26- Upload Speed (Mbps): Affects the quality of the video and audio transmitted, and the time required to send a file.

Useful ☐ Not Useful ☐

- Are there any other election criteria you would like to suggest? If yes, please mention them bellow:

……
……

……
……

……
……

……
……

5.3. Delphi Results

Table 28 Delphi Results

	Round 1				Round 2				
	Useful		Not Useful		Useful		Not Useful		
Proposed Criterion	Nbr	Percentage	Nbr	Per	Nbr	Percentage	Nbr	Per	Results
Experience as a DM	19	95%	1	5%	18	90%	2	10%	Retained
Treated Breakdowns	15	75%	5	25%	16	80%	4	20%	Retained
Distance to site	10	50%	10	50%	15	75%	5	25%	Retained
Coordination Experience	18	90%	2	10%	19	95%	1	5%	Retained
Response Time	13	65%	7	35%	15	75%	5	25%	Retained
Open Ports	11	55%	9	45%	16	80%	4	20%	Retained
Vulnerabilities	12	60%	8	40%	17	85%	3	15%	Retained
Severity Sum	14	70%	6	30%	17	85%	3	15%	Retained
Connection Type	9	45%	11	55%	15	75%	5	25%	Retained
Avg Net Latency	10	50%	10	50%	18	90%	2	10%	Retained
Download Speed	17	85%	3	15%	17	85%	3	15%	Retained
Upload Speed	15	75%	5	25%	16	80%	4	20%	Retained
Connection Interface	14	70%	6	30%	13	65%	7	35%	Rejected
Education Level	10	50%	10	50%	11	55%	9	45%	Rejected

Number of Spoken Languages	12	60%	8	40%	13	65%	7	35%	Rejected
Level of Spoken Languages	10	50%	10	50%	12	60%	8	40%	Rejected
Max Number of Coordinated Participants	10	50%	10	50%	11	55%	9	45%	Rejected
Public or Private Network	5	25%	15	75%	4	20%	16	80%	Rejected
Number of Router Hops to GDSS Server	2	10%	18	90%	1	5%	19	95%	Rejected
State of Found Vulnerabilities	3	15%	17	85%	4	20%	16	80%	Rejected
Computer Science Certificates	3	15%	17	85%	2	10%	18	90%	Rejected
Coordination and Facilitation Certificates	2	10%	18	90%	1	5%	19	95%	Rejected
Wifi Security Protocol	1	5%	19	95%	0	0%	20	100%	Rejected
Operating System	6	30%	14	70%	8	40%	12	60%	Rejected
System Update Date	5	25%	15	75%	6	30%	14	70%	Rejected

Security Update Date	4	20%	16	80%	5	25%	15	75%	Rejected