A cloud-based and web-based group decision support system in multilingual environment with hesitant fuzzy linguistic preference relations

Zixu Liu[a], Junyi Han[b], Fanlin Meng[c], Huchang Liao[d,*]

[a] *Crop Science Centre, National Institute of Agricultural Botany, Cambridge CB3 0LE, U.K.*

[b] *Department of Computer Science, The University of Manchester, Manchester M15 6PB, U.K.*

[c] *Department of Mathematical Sciences, University of Essex, Essex CO4 3SQ, U.K.*

[d] *Business School, Sichuan University, Chengdu 610064, China*

Abstract

Due to the growing needs in decision-making under uncertainty, existing studies introduced consistency and consensus-driven algorithms for group decision-making (GDM) problems with hesitant fuzzy linguistic preference relations (HFLPRs). A decision support system (DSS) that can host these GDM algorithms to provide decision-support services or tools for practical use is urgently needed. However, the state-of-the-art architectures cannot organize these algorithms and related data to run within one framework. This is mainly due to the running environments for these GDM algorithms are different since these algorithms were not originally designed to be compatible. Given the multilingual consistency and consensus-based decision support algorithms, how to design and implement a cloud-based DSS in a multilingual environment is still an open question. To fill this gap, this paper provides a web-based and cloud-based DSS with a novel architecture that utilizes the advantages of microservices. The proposed system implements a multilingual support framework to dynamically upload, manage and run multi-lingual consistency and consensus-based decision support algorithms. An algorithm recommendation module is developed to help users choose suitable decision support algorithms. Tokenization is applied to deal with regulatory issues of knowledge protection, data privacy, and security while storing, analyzing, and transforming data into different algorithms for effective decision-making. An expert feedback study verified that our web and cloud-based DSS is a right artifact to fulfill the objective claimed in this paper.

***Keywords*:** Group decision making; decision support system; hesitant fuzzy linguistic preference relation; cloud computing; pod-based architecture.

* Corresponding author. E-mail addresses: zixuxilan@gamil.com (Z.X. Liu); junyi.han@hotmail.com (J.Y. Han); fanlin.meng@essex.ac.uk (F.L. Meng); liaohuchang@163.com (H.C. Liao).

1. Introduction

With the fast development of the global economy, group decision making (GDM) has been widely used in many areas to obtain an optimal solution based on decision makers' opinions on an alternative set. The most common format to represent decision makers' opinions is the preference relation which is obtained based on the pairwise comparisons of alternatives. The linguistic preference relation (LPR) [1] which use the linguistic terms to represent pairwise assessments of alternatives, was developed in accordance with the habit of decision makers. However, given the increasing complexity of decision-making problems, the LPR is inadequate to express decision makers' opinions in the situation that decision makers are hesitant about linguistic variables. To solve this problem, the hesitant fuzzy linguistic preference relation (HFLPR) [2] was introduced. Different from the LPR, the HFLPR shows decision makers' preferences based on consecutive linguistic terms (LTs) which are represented by a hesitant fuzzy linguistic term set (HFLTS) [3, 4].

The study of HFLPRs is popular due to the growing needs in decision-making under uncertainty [5]. To apply HFLPRs in GDM problems, Zhu and Xu [2] presented α-normalization and β-normalization principles to solve the problem that HFLTSs in an HFLPR have different lengths and defined the additive consistency of HFLPRs. Zhao et al. [6] proposed a multiplicative consistency concept of HFLPRs. Liu et al. [7] developed a programming model to make HFLPRs achieve the given consistency threshold. Inspired by the additive consistency of LPRs, Tang et al. [8] gave an additive consistency definition for HFLPRs and then presented a method for GDM with incomplete and inconsistent HFLPRs. Liao et al. [9] alpha-cut based consensus measure for GDM with HFLPRs. Ren et al. [10] introduced a hesitant fuzzy linguistic geometric consistency index (HFLGCI) based on the geometric consistency of a multiplicative preference relation. The HFLGCI can indicate whether a HFLPR is consistent or not. With this measurement method of HFLPRs, Ren et al. [10] introduced a hesitant fuzzy linguistic geometric consistency index (HFLGCI) to check whether an HFLPR is consistent or not, and introduced a consensus driven GDM algorithm by measuring the group consensus based on the similarity between each decision maker's HFLPR and the overall perfect HFLPR. In general, these literatures provided algorithms and experimental results of GDM with HFLPRs, but they mostly ignored the issue of applications for the public to use or evaluate decision models. Redoing these works or setting the same algorithm running environment for their algorithm codes demands lots of programming. Further, users or decision makers may not understand the details of each decision-making model as they may lack the related knowledge. In this sense, a decision support system (DSS) with a friendly UI that can handle those developed GDM algorithms with HFLPRs is urgently needed to provide decision support services or tools for decision makers.

With the development of internet technology, web-based DSSs have become a trend in DSS research since it

features the following advantages: 1) zero to install: it only needs a browser to access; 2) centralized data, secure and easy to backup; 3) quick and easy to update. Power [11] first defined the concept of web-based DSS and introduced a traditional way of designing and developing an architecture for the Web-based DSS. Rahimi et al. [12] built a web-based medical diagnosis system with a dynamically generated web-based user interface to solve multicriteria decision making problems. To improve the traditional web-based DSS and integrate multidisciplinary data source and decision making tools, Zhang and Goddard [13] presented an integrated method to design and implement a web-based DSS in a distributed environment. Based on consistency and consensus measures, Alonso et al. [14] designed a web-based consensus support system. This system can deal with GDM problems with incomplete information or different representation formats of experts' preference information.

However, both mentioned DSSs can only apply decision support algorithms for a single running environment. For example, Rahimi et al. [12] implemented a multicriteria decision making algorithm with a parallel running setting on the running environment of C language, and Alonso et al. [14] ran their GDM algorithm in a JAVA environment. Most of the existing Web-based DSSs were designed in a layered architecture. It has the following drawback. First, its overall agility, the ability to modify the application to a changing environment, is low [15]. It is still challenging and time-consuming for a single modification due to the tight coupling of components. Secondly, scalability is a big issue for this pattern. The complexity of a system of tightly coupled components grows exponentially with the increasing number of components. This could be complicated by the fact that different researchers may use different programming languages to test and program their consistency and consensus-based decision support algorithms. In addition, the running environments for these algorithms and related data are different and these algorithms were not originally designed to be compatible. In summary, existing systems or platforms lack the ability to organize those algorithms and related data to run within one framework but only support one or a few algorithms in one programming language environment. By identifying the above research gap, in this paper we propose the following research question: given the multilingual consistency and consensus-based decision support algorithms and related running environment, how to address the design, specification, and implementation of a Web-based DSS in a multilingual support environment?

With the developments of web and cloud-computing technology, the multi-language support environment can be designed and deployed to a cloud server under the microservice architecture [16]. The microservice architecture pattern is quickly gaining ground in the industry as a viable alternative to monolithic applications by arranging an application as a collection of loosely coupled services [17]. However, this pattern does not naturally lend itself to the high-performance application due to the distributed nature of microservices architecture. Microservice has all the

associated complexities of the distributed system that cause a higher chance of failure during communication between different services. Also, it is difficult to manage a large number of services which will restrict the number of the decision-making algorithms hosted in the system. The research gaps analysed above show that no current DSS exhibits a suitable architecture that could manage and run multilingual algorithms as services in a seamless way in such a cloud-based running environment. Therefore, we propose a Web-based and Cloud-based DSS with a novel architecture that utilizes the advantages of microservices to fill the above-mentioned gaps. Furthermore, the regulatory issues of knowledge protection, data privacy, and security are needed to consider when designing a digital platform [18, 19].

To solve above mentioned research problems, this paper presents a Web-based and Cloud-based DSS for consistency and consensus-based algorithms for GDM with HFLPRs, from design to implementation. Our work provides an online platform for users to test, compare and evaluate different research work, and for researchers to get feedback on their work. The main contributions of the proposed system are summarized as follows:

1) The presented cloud-based DSS provides a multi-language support framework to manage and run multilingual consistency and consensus-based decision support algorithms based on a customized pod-based architecture. This framework is designed to provides a running environment for all main programming languages. The designed data translation workflow module provides the data interface between the system that can transfer the user input to specific data types required by the multilingual consistency and consensus-based decision support algorithms.

2) By using a container management tool, a pod pool is designed in the proposed DSS that supports dynamically adding, updating, and removing of the algorithms. An algorithm recommendation module is developed in the proposed system to help users choose suitable decision support algorithms.

3) Tokenization is applied to deal with regulatory issues of knowledge protection, data privacy, and security while storing, analyzing, and transforming data into different algorithms for effective decision-making.

The rest of this paper is organized as follows: In Section 2, we describe the overall architecture design and two illustrative algorithms used in the system. The data translation workflow and the algorithm recommendation module are then discussed in Sections 3 and 4, respectively. The way in which the system is implemented is described in Section 5. Section 6 presents the expert feedback study for the proposed system. The paper is concluded in Section 7.

2. System Architecture and illustrative algorithms for decision making

To maximize the compatibility on the diversity of running times and programming language, this study proposes a cloud-based system architecture that utilizes tokenization and container management methods. The fundamental

unit of our architecture is Pod (point of delivery). Thus, this section first describes the overall architecture design and then presents two illustrative algorithms used in the system.

2.1. Pod design and System architecture for decision making

Containers are a form of operating system virtualization. A container can be used to run everything from small microservices or software processes to large applications. The container contains all necessary executable files, binary code, libraries, and configuration files. However, unlike server or computer virtualization methods, containers do not contain operating system images. Therefore, they are light and portable, with little overhead. In large-scale application deployment, multiple containers can be deployed as one or more container clusters. Such clusters can be managed by container orchestration programs such as Kubernetes, and the corresponding management operations including the creation, deployment, and scaling of containers are named container management [20]. The chief benefit of container management is simplified management for clusters of container hosts.

In terms of container management, a pod is the basic execution unit of an application. In our case, the application is a bundle of microservices that in together form an algorithm service. Figure 1 depicts the details of a pod.

A pod includes two containers, and both of them share storage resources and have a unique network IP. Container 1 is customized by users. They can use any programming language and runtime to generate this container image and upload it onto our customized container registry as shown in Figure 2.

<Insert Fig. 1 here>

<Insert Fig. 2 here>

To work with Container 2 in a pod, two simple rules are designed for the benefit of generalization. The first rule is the location of input and output files, while the second rule is the input and output data formats. In this project, we propose to use JSON as our data format. JSON is an open standard file format and data interchange format that has been widely used in electronic data exchange, especially for web applications with servers. The data saved in JSON format consists of attribute-value pairs and arrays that make the data human-readable. It was driven from JavaScript but many modern programming languages include code to generate and parse JSON-format data, which means the JSON is language-independent data format. That is also the reason why we use JSON in the DSS for data interchange. Based on the user feedback in the future, more data formats can be added to our rules. Following these rules, the Algorithm Trigger can initiate an execution of the algorithm in Container 1, and Output Loader can transfer the output of this algorithm to the Driver Manager, which acts as the interface of a pod to communicate with other web services.

By using a container management tool, Kubernetes (K8S), we are able to create a pod pool that dynamically adds new algorithms, updates algorithms, and removes algorithms. In addition, we create a web server to

communicate with the algorithms in the pod pool. The overall architecture is shown in Figure 3.

<Insert Fig. 3 here>

Both the pod pool and the web server are deployed on the cloud. In this paper, Amazon Web Service is used and 1 p3.xlarge instance [21] is chosen as the node to host the pod pool. As the webserver is not compute-intensive, a t2.micro [22] is used for cost saving purpose. The web server is based on the Linux, Apache, MySQL, PHP (LAMP) technique stack. On one hand, it hosts our web application for users using desktops and mobile devices. On the other hand, it takes REST API [23] to pass users requests to the algorithm selector. If the request relates to an existing algorithm, the Algorithm Master communicates with a Driver Manager and relays the request to a corresponding pod in the pod pool. If the request regards an algorithm CRUD, the Algorithm Master calls K8S API [24] to create a corresponding service and a deployment using the algorithm name selected by users. After a user finished using an algorithm, the Algorithm Master will pass the results back to our user and kill the container to fulfill a dynamic and low-cost web service.

2.2. Algorithms in the pod pool

To facilitate the understanding of the proposed support system, we introduce two illustrative algorithms saved in the pod pool. Both algorithms are used to manage consistency and consensus in GDM with HFLPRs. The first algorithm provides a consistency improvement process for an inconsistent HFLPR. The second algorithm measures the consensus for a group of HFLPRs and provides a consensus reaching procedure. Before introducing these algorithms, we start from some basic concepts.

Linguistic term set (LTS) [25] conforms with the expression habit of decision makers. The basic form of an LTS is $S = \{s_\alpha | \alpha = 0,1, \dots 2\tau\}$, where τ is a positive number, s_α is a linguistic term (LT) and $2\tau + 1$ indicates the granularity of the LTS. For $s_\alpha \in S$ and $s_\beta \in S$, if $\alpha < \beta$, then $s_\alpha < s_\beta$; $s_\alpha \oplus s_\beta = s_{\alpha+\beta}$; $\lambda s_\alpha = s_{\lambda\alpha}$; $Neg(s_\alpha) = s_{2\tau-\alpha}$ [26]. For the convenience of expression, we use $I(s_\alpha)$ to denote the subscript of the linguistic variable s_α where $s_\alpha \in S$.

Let $L = (l_{ij})_{n\times n} \subset X \times X$ denote a preference relation on an alternative set $X = \{x_1, x_2, \dots, x_n\}$, where $l_{ij} \in S = \{s_\alpha | \alpha = 0,1, \dots 2\tau\}$ and l_{ij} indicates the preference degree of x_i over x_j. If $\forall i, j = 1,2, \dots, n$, $l_{ij} \oplus l_{ji} = s_{2\tau}$ and $l_{ii} = s_\tau$, L is called a linguistic preference relation (LPR) [1]. An HFLTS is an ordered finite subset of the consecutive LTs of the LTS $S = \{s_\alpha | \alpha = 0,1, \dots 2\tau\}$. [3]. Its mathematical notation is shown as $b = \{\langle x_i, b(x_i)\rangle | x_i \in X\}$ [27] and $b(x_i)$ is called a hesitant fuzzy linguistic element (HFLE). Zhu and Xu [2] gave the definition of the HFLPR. An HFLPR H on the alternative set $X = \{x_1, x_2, \dots, x_n\}$ for the predefined LTS $S = \{s_\alpha | \alpha = 0,1, \dots, 2\tau\}$ is a matrix $H = (h_{ij})_{n\times n}$, where $h_{ij} = \{h_{ij}^l | l = 1,2, \dots, m_{ij}\}$ (m_{ij} is the number of LTs in

h_{ij}) is an HFLE, denoting the qualitative hesitant preference degree of alternative x_i over x_j. The HFLPR H satisfies the following equations for all $i, j = 1,2, \dots, n$ with $i \neq j$: 1) $h_{ij}^l < h_{ij}^{l+1}$, $l = 1,2, \dots, m_{ij} - 1$; 2) $h_{ij}^l \oplus h_{ij}^{m_{ji}+1-l} = s_{2\tau}$, $l = 1,2, \dots, m_{ij}$; 3) $m_{ij} = m_{ji}$; 4) $h_{ii} = \{s_\tau\}$. Different HFLEs in an HFLPR may have different lengths. Zhu and Xu [2] introduced the normalized HFLPR using the optimized parameter ς ($0 \leq \varsigma \leq 1$). For the HFLE h_{ij} in an HFLPR, we can add the linguistic term $\varsigma h^+ + (1-\varsigma)h^-$ in h_{ij} $(i < j)$ and add the linguistic term $\varsigma h^- + (1-\varsigma)h^+$ in h_{ji} $(i < j)$ until all HFLEs have the same length, where h^+ and h^- are the maximum and minimum LTs in h_{ij}, respectively. The obtained normalized HFLPR is called an NHFLPR with ς.

As this study mainly aims to introduce a cloud-based group DSS for consistency and consensus-based algorithms with HFLPRs, below we briefly introduce the logic of the two algorithms. For more details of equations in these two algorithms, please see our previous work [10].

2.2.1. Algorithm I: Consistency-based procedure for an HFLPR

The first proposed algorithm used in the support system is named the Consistency procedure of an HFLPR. This algorithm can check the consistency of an inputted HFLPR and improve its inconsistency based on a critical value $\overline{HFLGCI}$. The hesitant fuzzy linguistic geometric consistency index (HFLGCI) was proposed in [10] based on the geometric consistency theory of multiplicative preference relation. If the HFLGCI of *H* is smaller than or equal to $\overline{HFLGCI}$, then *H* is acceptably consistent; otherwise, *H* is unacceptably consistent.

Algorithm I. (**A consistency-based procedure for an HFLPR)**

Input: An HFLPR $H = (h_{ij})_{n\times n}$, where $h_{ij} = \{h_{ij}^l | l = 1,2, \dots, m_{ij}\}$, a critical value $\overline{HFLGCI}$.

Output: The revised *H* with acceptable consistency and the number of loop rounds to reach the consistency.

Step 1. Let the round number *r*=1, and denote *H* as *H(r);*

Step 2. Calculate the priority weight vectors of *H(r);*

Step 3. Compute the *HFLGCI* of *H(r)* for each of the priority weight vectors; the priority vector that makes the *HFLGCI* minimum is the optimal solution;

Step 4. If $HFLGCI > \overline{HFLGCI}$, go to Step 5; otherwise go to Step 6;

Step 5. Obtain the perfectly consistent LPR $\bar{C}$ from *H(r),* where $\bar{C} = \left(\overline{c_{ij}}\right)_{n\times n}$. Based on $\bar{C}$, adjust each HFLE $h_{ij}(r)$ in *H(r)* to derive a new HFLPR by using Eq. (1) where $0 \leq \beta \leq 1$, denoted as *H(r+1);* let *r=r+1*, return to Step 2.

Step 6. Output *H(r)* and *r.*

$$h_{ij}(r+1) = \beta\left(h_{ij}(r)\right) + (1-\beta)\overline{c_{ij}} \quad (1)$$

In each loop of Algorithm I, the inputted HFLPR will be closer to its optimal solution which is a perfectly consistent LPR. The HFLGCI will also be smaller than that in previous loops. Once the HFLGCI converges to $\overline{HFLGCI}$, the algorithm stops. To analysis the time complexity of Algorithm I, the number of elementary operation of each step needs to be counted. The purpose of Step 2 is to calculate the priority vectors of the input HFLPR. Based on Proposition 3.1 in [10], the number of the priority vectors depends on the maximum m_{ij} in the HFLPR where m_{ij} is the number of LTs in h_{ij}. Since we set τ as 4, the granularity of the LTS is 9 that is the maximum value of m_{ij}. Each priority vector contains n elements. Thus, the number of elementary operation to calculate an element in the priority vectors is n^2 according to Eq. (3.5) in [10]. In total, the maximum number of elementary operations of step 2 is $9 * n * n^2$ and the time complexity is $O(n^3)$. From Eq. (3.5) in [10], the number of calculations of Step 3 to get the HFLGCI from a priority vector is $(n^2 - n)/2$. Therefore, the time complexity is $O((n^2 - n) * 9/2) = O(n^2)$ since there are at most 9 vectors. In Step 5, the perfectly consistent LPR $\bar{C}$ is calculated by using at most $9n^2$ times of elementary operation. That is because its maximum size is $9n^2$ and $\overline{c_{ij}}$ could be obtained by using one step of the calculation based on Eq. (3.5) in [10]. The adjustment procedure for each HFLE $h_{ij}(r)$ in *H(r)* to derive a new HFLPR is the main factor to determines the speed of the consistency procedure and the number of the loops in this algorithm. The adjustment function we use in Step 5 is $h_{ij}(r+1) = \beta\left(h_{ij}(r)\right) + (1-\beta)\overline{c_{ij}}$, where $0 \leq \beta \leq 1$. Normally, the value of β is chosen from a range of 0.4 to 0.6. The maximum distance of the elements between an HFLPR and its perfectly consistent LPR is 9. Therefore, the maximum value of loop round *r* in Algorithm 1 is $\log_{(\frac{1}{0.6})} 9$. In conclusion, the time complexity of Algorithm I is $O\left(\log_{(\frac{1}{0.6})} 9 * (n^3 + 2n^2)\right) = O(n^3)$.

From the previous analysis, we could see that the maximum units of memory space required to run Algorithm I is $3 * 9 * n^2 + 9 * n + 10$, where $3 * 9 * n^2$ units of space are used for *H(r)*, $\bar{C}$ and *H(r+1)*, $9 * n$ units for the priority weight vectors, and 10 units for *HFLGCI*s. The space complexity of Algorithm I is $O(n^2)$.

Since Algorithm I is a priority-deviation method, neither adding extra nor removing values from HFLTSs, it can consider all LPRs of an HFLPR equally and directly to avoid information loss.

2.2.2. Algorithm II: Consensus-based procedure for GDM with HFLPRs

The second algorithm measures the consensus of a group of HFLPRs and then provides a consensus reaching procedure. For k decision makers, let $H^p = (h_{ij}^p)_{n\times n} \subset X \times X$ $(p = 1,2 \dots, k)$ on the alternative set $X = \{x_1, x_2, \dots, x_n\}$ be their HFLPRs. The weight of each decision maker can be computed based on the similarity degree between H^p and its corresponding perfect consistent HFLPR $\widetilde{H^p}$. The weight calculation function was given in [10]. The similarity measure was defined in [27], and the perfect consistent HFLPR can be calculated by the method

studied in [2, 3]. Different from Zhang and Chen [28]'s work in which the similarity measures between HFLPRs were used, Algorithm II gives more trust to the decision maker who provides a more logical HFLPR. Suppose $\bar{H} = (\overline{h_{ij}})_{n\times n} \subset X \times X$ is the collective HFLPR aggregated by all $\widetilde{H^p} = (\widetilde{h_{ij}^p})_{n\times n}$ for $p = 1,2 \dots, k$. How to calculate the collective HFLPR was introduced in [10]. The consensus degree of each decision maker is measured by the similarity between its HFLPR H^p and $\widetilde{H^p}$. The minimum consensus degree among these decision makers is regarded as the consensus degree of the group, denoted as cd. γ is used as the consensus threshold.

Algorithm II. (Consensus-based procedure for GDM with HFLPRs)

Input: Decision makers' HFLPRs $H^p = (h_{ij}^p)_{n\times n} \subset X \times X$ $(p = 1,2 \dots, k)$, a critical value $\overline{HFLGCI}$ and a consensus threshold γ.

Output: A ranking on the alternative set X and the number of loop rounds to get the result.

Step 1. Calculate the acceptably consistent HFLPRs $\widehat{H^p} = (\widehat{h_{ij}^p})_{n\times n}$ for all HFLPRs $H^p = (h_{ij}^p)_{n\times n}$ for $p = 1,2 \dots, k$ by applying Algorithm I;

Step 2. Calculate the corresponding perfectly consistent HFLPRs $\widetilde{H^p}$ for all individual HFLPRs H^p for $p = 1,2 \dots, k$, and compute the weight of each decision maker based on $\widetilde{H^p}$ and H^p;

Step 3. Use $\widetilde{H^p} = (\widetilde{h_{ij}^p})_{n\times n}$ for $p = 1,2 \dots, k$ and the weight of each decision maker to obtain the collective perfectly consistent HFLPR $\bar{H} = (\overline{h_{ij}})_{n\times n}$. Let the round number r=1, $\widehat{H^p(r)} = \widehat{h_{ij}^p(r)}_{n\times n}$;

Step 4. For $p = 1,2 \dots, k$, compute the consensus degree of $\widehat{H^p(r)}$ and get the group consensus degree $cd(r)$ (minimum consensus degree among all DMs).

Step 5. If the group consensus degree $cd(r) \geq \gamma$, go to Step 8; otherwise, go to the next step;

Step 6. For the DM *dm* from p who has the minimum consensus degree, based on the distance measure defined in [29], find the maximum distance of $\widehat{H^{dm}(r)}$ on the alternatives from $\bar{H}$ and then revise it. The revise function for $\widehat{H^p(r)}$ on alternative *i* is shown in Eq. (2), where $0 \leq \beta \leq 1$, The modified HFLPR $(\widehat{H^{dm}(r)})_{new}$ will be closer to $\bar{H}$ and calculate its current consensus degree.

Step 7. Detonate the modified HFLPR and other HFLPRs as $H^p(\widehat{r+1})$. Let *r=r+1* and get the group consensus degree $cd(r)$. Return to Step 5.

Step 8. Use $\widehat{H^p(r)} = \widehat{h_{ij}^p(r)}_{n\times n}$ for $p = 1,2 \dots, k$ to obtain the collective HFLPR $\hat{H} = (\widehat{h_{ij}})_{n\times n}$, and then calculate the priority weight vector of $\hat{H}$. The ranking on X can be obtained by this priority weight vector.

$$(\widehat{h_{ij}^{p_r}(r)})_{new} = \beta\left(\widehat{h_{ij}^{p}(r)}\right) + (1-\beta)\overline{h_{ij}^p}, \quad \forall j = 1,2 \dots, n \tag{2}$$

Now we analyze the time complexity of Algorithm II. Step 1 here is the implementation of Algorithm I for each DM, the time complexity is $O(n^3k)$, where k is the number of DMs. Based on [28], the time complexity of calculating the corresponding perfect consistent HFLPR for all DMs can be derived as $O(n^3k)$. To get the weight of each DM, we need to calculate the similarity between each individual HFLPR and its perfect consistent HFLPR first. Since the granularity of the LTS is set as 9, the time complexity of similarity calculation can be deduced as $O(9n^2)$ from [29]. In total, the time complexity of Step 2 is $O(9kn^2 + n^3k) = O(n^3k)$. According to Eq. (3.10) in [10], the maximum number of the elementary operations for proposing the collective perfectly consistent HFLPR is $9 * k * n^2$. Hence the time complexity of Step 3 is $O(kn^2)$. In [10], we defined the consensus degree of the group as the minimum value of similarity between each individual HFLPR and its perfect consistent HFLPR. The time complexity of this calculation is already analyzed in Step 2, which is $O(9n^2k)$. In Step 6, the modification function for an HFLPR $\widehat{H^p(r)}$ is shown in Eq. (2). This function is similar to the adjustment function Eq. (1) used in Algorithm I. Like the loop analysis in Algorithm I, we can conclude that the maximum value of *r* in Algorithm II is $n * k * (\log_{(\frac{1}{0.6})} 9)$. Here, $n * k$ means the algorithm may alternately modify all HFLPRs on their all alternatives in the worst case. The revise function only has *n* operations and the calculation of the consensus degree needs at most $n^2 * 9$ operations. Therefore, the time complexity of the loop process in Algorithm II is $O\left(nk * (\log_{(\frac{1}{0.6})} 9) * (1 + n + 9n^2)\right) = O(n^3k)$. For Step 8, the time complexity of obtaining the collective HFLPR and its priority weight vector was already analyzed in Step 3 of Algorithm II and Step 2 of Algorithm 1 respectively, both are $O(9n^2k)$. In conclusion, the time complexity of Algorithm II is $O(3n^3k + 3n^2k + 1) = O(n^3k)$.

During the calculation of Algorithm II, most of the memory space is used by all DM's HFLPRs and their corresponding perfectly consistent HFLPRs that is $9 * n^2k * 2$ units of memory space. The rest of memory is required by collective perfectly consistent HFLPR ($9n^2$ units), weight vector of all DMs (k units), and the priority weight vectors ($9n$ units). From this, we could conclude that the space complexity of Algorithm II is $O(n^2k)$.

Figure. 4 shows a brief diagram of Algorithm II that also contains the process illustration of Algorithm I.

<Insert Fig. 4 here>

3. Data translation workflow

In this section, we describe how the HFLPR information from user input transfers to the data types that can be used in different consistency and consensus-based algorithms with multi-languages.

Figure 5 shows the HFLPR input form we designed in the web UI. Normally, users provide their hesitant fuzzy linguistic preference options using the LTS $S = \{s_\alpha | \alpha = 0,1, \dots 2\tau\}$. For users' convenience, we simplify the

representation of the element of an HFLPR to the above input form. Users only need to input the subscript of each LT and this will make the process of calculation easier in the back-end algorithms. Also, users only need to provide the maximum and minimum LTs rather than all LTs for each element of the HFLPR. The support system will automatically recover all HFLEs based on the maximum and minimum LTs while considering other constraints (to be discussed later) in the algorithm selector and then send them to the back-end algorithms in the pod pool.

<Insert Fig. 5 here>

We propose two input rules for the HFLPR input form:

R1) Input only the subscript number of an LT for each input box; only the maximum and minimum LTs for each element of an HFLPR are needed for the form. Here, $HFLE(i,j)_{max} \geq HFLE(i,j)_{min}$.

R2) All inputs must satisfy the following property of the HFLPR: $HFLE(i,j)_{max} \oplus HFLE(j,i)_{min} = s_{2\tau}$.

The input form validates the user's input according to these two rules. Our web UI is built in the Angular platform. Apart from providing some basic validation functions, Angular cloud also applies the customized validation functions or rules to the input form [30]. So, the input form in Fig. 5 can prevent a user from inputting non-numeric characters with our coded validation function. An error message will be shown under the corresponding input box if the user's input violates rule 2.

After the user completed the input form and clicks the submit button, all the information in the input form is collected and transferred to the following array: $HFLPR\ message = [n, L, \tau, \tau, X, X, \ldots, \tau, \tau]$. In this array, n is the number of alternatives, L is the maximum or minimum length of all elements in the HFLPR. Here we show an example. For the HFLPR H in Eq. (3), according to the input rules *R1* and *R2*, the user inputs *H'* in Eq. (4) into the input form. The webserver then sends the following array in JSON format to the back-end: [3,3,4,4,5,7,4,5,1,3,4,4,5,5,3,4,3,3,4,4].

$$H = \begin{Bmatrix} \{s_4\} & \{s_5, s_6, s_7\} & \{s_4, s_5\} \\ \{s_1, s_2, s_3\} & \{s_4\} & \{s_5\} \\ \{s_3, s_4\} & \{s_3\} & \{s_4\} \end{Bmatrix} \tag{3}$$

$$H' = \begin{Bmatrix} \{s_4\} & \{s_5, s_7\} & \{s_4, s_5\} \\ \{s_1, s_3\} & \{s_4\} & \{s_5, s_5\} \\ \{s_3, s_4\} & \{s_3, s_3\} & \{s_4\} \end{Bmatrix} \tag{4}$$

Along with other parameters such as the algorithm name selected by the user, the $HFLPR\ message$ is sent to the back end (the Algorithm Selector) of the system. Based on different algorithm selections, the Algorithm Selector will apply different methods to the $HFLPR\ message$.

As we mentioned before, the length of different HFLTSs in an HFLPR may vary. Some HFLPR algorithms only accept normalized HFLPR(s). If the user's selected algorithm requires a normalized HFLPR based on the α-normalization, the webserver will indicate the minimum length of all elements of an HFLPR in the $HFLPR\ message$.

The data translator module in the back end builds an empty 3-dimensional array with size $n * n * L$. Starting from the third position of the *HFLPR message,* it reads two numbers and recovers them to an HFLE based on the corresponding LTS. The LTs from HFLEs with a bigger number of LTs than the minimum length *L* are removed. All modified HFLEs are placed in a 3-dimensional array with their original locations.

For some algorithms which require a normalized HFLPR based on the β-normalization, the transfer process is similar to that based on the α-normalization. The only difference is that the HFLEs with a greater number of LTs than the maximum length *L* will add extra LTs. According to Zhu and Xu [2]'s normalized method, the added LT is determined by the optimized parameter ς. The maximum value of the LT in an HFLE is added to this HFLE if ς=1 (that means the decision maker is optimistic); the minimum value of the LT in an HFLE is added to this HFLE if ς=0 (that means the decision maker is pessimistic); the average value of the LT in an HFLE is added to this HFLE if ς=0.5 (that means the decision maker is neutral). If the HFLPR algorithms with β-normalization pre-define the value of ς, the data translator will add LTs according to this pre-defined value. If the HFLPR algorithms ask users to choose the value of ς, a selection box will be added to this algorithm's input form and users can select their preferred type as pessimistic, optimistic, or neutral. This parameter will be sent to the Algorithm Selector along with the $HFLPR\ message$ in JSON format. In the next section, we will demonstrate an extra parameter box added to the example Algorithm II's input form, which asks users to choose the value of the consensus threshold value. This shows our customized ability for the input form and web UI.

After the normalization process, the obtained 3-dimensional array which represents decision maker's HFLPR is sent to the selected algorithm pod for calculation. The array data type is widely defined in different programming languages, such as C, C++, Java, JavaScript, Python, MATLAB. It can store many elements at a time. Array elements share a common name and are stored in sequential memory locations. Any element of the array can be randomly accessed using indexes. These properties make the array the first choice to represent an HFLPR when programming simulation algorithms. Some algorithms may require other data types in input such as the *cell* in MATLAB. Since the *cell* may contain numeric arrays with different sizes to better represent an HFLPR, this kind of algorithm can do the normalization by itself. In this scenario, the Algorithm Selector will run a MATLAB function (also saved in a pod) to convert the *HFLPR message* to a *cell* which saves and sends the user's inputted HFLPR to the corresponding algorithm pod for calculation.

Existing work [2, 8] adopted the additive consistency of HFLPRs. But one of the main issues of additive consistency is that the sum of two linguistic terms might fall out of the predefined LTS. That's also why other scholars [6, 28] adopted the multiplicative consistency concept for HFLPRs. However, since what different consistency

concepts affect lies in the calculation process in different algorithms, and we treat each algorithm as a black box and just provide input in required formation, the different consistency concepts do not affect our data transfer workflow.

Although Saaty's 1-9 scale [31] has been applied in almost all HFLPR algorithms to indicate the preference degree of one alternative over another, the LTS of these algorithms may vary. For instance, Wu and Xu [32] used the LTS $S = \{s_\alpha | \alpha = -4, -3, \dots, 0, \dots, 4\}$; Zhang et al. [33] used the LTS $S = \{s_\alpha | \alpha = 0.1, 0.2, \dots, 0.9\}$. The LTS used in Algorithm I is $S = \{s_\alpha | \alpha = 0, 1, \dots, 8\}$. That is why the diagonal of the input form in Fig. 5 is set as 4. Our DSS provides two methods to address this issue regarding the evaluation scale. The first method is still using the input form shown in Fig. 5 for these algorithms. Since all these algorithms adopt Saaty's 1-9 scale, the user's input values can be transferred to the values used in the LTS of these algorithms based on the mapping relation between two different LTSs. This method is applied in the data transfer workflow module. The second method is changing the code of the input form and related validation functions to create a new input form. When a user changes the selection in the Algorithm Selection box in Fig. 5, the corresponding new input form will show below, which will be demonstrated in an example in the next section. Under this method, other fuzzy decision-making algorithm which did not use HFLPR to present decision makers' opinions in the situation could be adopted to our system. For example, Chen and Zou [34] develops a fuzzy group decision making method based on the generalized fuzzy soft set. The new input form which could enable the user to input decision makers' decision fuzzy soft set can be customized by our original input form, i.e., different from HFLPR which need to provide the maximum and minimum LTs for each element, the user only need to provides a single value for decision maker's evaluation (single value) of an alternative under a criterion; therefore the input form will be changed to simplified. Moreover, the number of decision makers, alternatives, and criteria could be specified by the user (similar to selecting the number of alternatives in Figure 5). This simplified version of the input form could also be used by the work of Singh and Benyoucef [35] who built a consensus-based group decision making methodology for consensus forming among the supply chain partners. Most of the mentioned work in this paragraph were focused on hesitant additive preference relations. How about the hesitant multiplicative preference relations (HMPRs) [36]? Lots of the research works solve the consistency and consensus-based GDM with HMPRs. For instance, Meng et al. [37] defined a distance measure to determine the weights of decision makers and proposed a consensus index to address GDM problems with HMPRs. To apply the works on HMPRs to the DSS, the input rule 2 needs to be modified to satisfy the properties of HMPR [36]: all inputs must satisfy the following property of the HMPRs: $HMPR(i,j)_{max} \otimes HFLE(j,i)_{min} = 1$. In conclusion, two mentioned methods regarding the evaluation scale issue are applied in the DSS and tested successfully.

This section introduced the data translation workflow module in the proposed support system. This workflow

builds bilateral communications between users and webserver, webserver, and algorithm selector, selector and the final HFLPR algorithms. All the discussion in this section proves that our data translation workflow is able to handle the data interface for different kinds of HFLPR algorithms in different programming languages, and deliver the HFLPR and related information from end-users to HFLPR algorithms for calculation.

4. Algorithm recommendation

Different algorithms employ different methods to do group decision making and may return different results, which makes it difficult for users to select a proper algorithm as they may not know each algorithm in great detail. With the algorithm pods saved in the pod pool increasing, the complexity of the algorithm selection for users is aggravated. Furthermore, users may have their own preferences for the methods or parameters used in the calculation according to their specific decision-making requests. To solve these problems, we propose an algorithm recommendation module in our DSS, which will be introduced in this section.

When researchers or users upload their algorithm (in the docker container image format) to the system, the related information of this algorithm needs to be specified in the meanwhile. Once the algorithm pod is created and saved in the pod pool, a new instance of the algorithm class which contains all the related information of this new algorithm is created and saved in the storage module. A storage module is developed to store, retrieve, and manage all the information of the saved algorithms and all these are needed by the algorithm recommendation module or other modules. Data is stored in a relational database (in MySQL). The algorithm class's data structure is illustrated in Figure 6. The algorithm class contains the following properties: *id*, *name*, *description*, *consistency*, *normalization*, *decisionmaker,* and *selectionProcess*. More properties can be added according to the user feedback in the future.

<Insert Fig. 6 here>

When designing the digital platform, the process of storing, analyzing, and transforming data into useful knowledge for effective decision-making while dealing with regulatory issues of knowledge protection, data privacy, and security is the research problem [18, 19] we identified in Section 1. Considering this problem, tokenization is applied in the DSS to facilitate the algorithm recommendation. Tokenization is the process of turning a meaningful piece of sensitive data, such as bank detail, into a random string or other formats of non-meaningful, non-sensitive data, referred to as a Token [38]. Token serves as a reference to the original data. It has no meaningful value if breached and cannot be used to guess those values. That is due to that tokenization does not have an underlying mathematical process to transform the sensitive information into the token, which is different from encryption. Instead of using the key or algorithm to derive the original data for a token, tokenization uses a database. This database is called a token value which stores the relationship between the sensitive value and the token.

In the proposed DSS, MySQL is applied in the data storage module. Figure 7 illustrates how we applied tokenization in the DSS. After the user submits its preference or requirement value to the token vault, the matched token value is obtained. The preference value is normally the methods or parameters used in the decision-making process. The token value such as the id is used to fetch the algorithm saved in the pod pool for use in the algorithm recommendation process. To the end-users, this operation is performed seamlessly by the web browser nearly instantaneously where the detail of the operation is hidden from them.

<Insert Fig. 7 here>

The *name* in the algorithm class (Figure 6) is the algorithm's name which shows in the selection box of "Algorithm Selection" of the input form. As we said in Section IV, some researchers adopted the additive consistency concept for HFLPRs, while others adopted the multiplicative consistency concept. Therefore, the saved algorithms in the DSS are classified by additive, multiplicative or other consistency types, and this information is saved in the *consistency* property of their class instances.

The *normalization* property in the algorithm class saves the information about the normalization methods used in an algorithm. If a class instance saves "α-*normalization*" in the *normalization* property, the value of *decisionMaker* in this instance will be "*NotConsidered*". For the algorithms which use the β-normalization method to normalize HFLPRs, the value of their *decisionmaker* varies according to the value of the optimized parameter ς in these algorithms: "*Optimistic*" if ς=1, "*Pessimistic*" if ς=0, "*Neutral*" if ς=0.5.

Before measuring the group consensus in GDM algorithms, the collective HFLPR should be calculated. There are serval different methods to aggregate HFLEs. Let $h_p(p = 1,2, \dots k)$ be a collection of HFLEs. Zhang and Chen [28] developed the hesitant fuzzy linguistic averaging (HFLA) operator and the hesitant fuzzy linguistic geometric (HFLG) operator for aggregating HFLEs:

$$HFLG(h_1, h_2, \dots h_p) = \oplus_{p=1}^{n} \left(\frac{1}{k} * h_p\right) = \cup_{S_{a_1} \in h_1, \dots, S_{a_k} \in h_k} \{S_{\sum_{i=1}^{k} a_i / k}\} \quad (5)$$

$$HFLG(h_1, h_2, \dots h_p) = \oplus_{p=1}^{n} \left((h_p)^{\frac{1}{n}}\right) = \cup_{S_{a_1} \in h_1, \dots, S_{a_k} \in h_k} \{S_{\prod_{i=1}^{k} {a_i}^{1/n}}\} \quad (6)$$

Based on the HFLA and HFLG operators, Lee and Chen [39] proposed the hesitant fuzzy linguistic weight average (HFLWA) operator, hesitant fuzzy linguistic weight geometric (HFLWG) operator, hesitant fuzzy linguistic ordered weight average (HFLOWA) operator, and hesitant fuzzy linguistic ordered weight geometric (HFLOWG) operator. Therefore, the possible value of "selectionProcess" in the algorithm class is one of the following: HFLA, HFLG, HFLWA, HFLWG, HFLOWA, HFLOWG, or Other.

Figure 8 is the user interface when using the algorithm recommendation functionality. There are serval new selection boxes shown above the input form. Besides the corresponding values in the algorithm class, these selection

boxes have an extra option: *No preference*, which gives users more choices. The optimized parameter selection box is shown and activated only when the β-normalization method is selected in the Normalization method selection box. Here we give an example for the algorithm recommendation functionality: if users select the following options in Figure 8: *Additive*, *β-normalization*, *Optimistic,* and *HFLWA*, the Algorithm II introduced in Section 3 is recommended and called by the DSS to do the calculation.

<Insert Fig. 8 here>

Sometimes there maybe two or more algorithms in the pod pool satisfying the user's requirement but returning different calculation results. In this scenario, recommending a better result is also an important function of our support system. Here, we set two criteria to filter the results from different algorithms. The first criterion is the consensus threshold value selected by the user in Fig. 8. Some algorithms' final consensus degrees may not be able to converge to a high enough value to meet users' requirement. Then, these algorithms will have a lower priority to be recommended. The second criterion is the efficiency of algorithms. This efficiency is not about the calculation efficiency, but the number of rounds needed to reach the consensus. For a specific decision-making problem, the algorithm which needs fewer rounds has better efficiency and higher priority. With these two criteria, the recommendation priority for two or more algorithms is obtained.

5. Decision support system use case

The term 'Use case' means the way in which a user uses a system. It is a collection of possible sequences of interactions related to a particular goal between the system under construction and its external actors [40]. Actors here are the users of the proposed support system. We illustrate three scenarios to present the use cases of the proposed DSS: the user inputs his HFLPR and uses Algorithm I to improve the consistency of the HFLPR (Section 5.1); the user inputs a group of HFLPRs and uses Algorithm II to get the ranking of all alternatives (Section 5.2); user case of algorithm upload and recommendation (Section 5.3). Figure 9 describes the use case diagram. These two use cases are available at http://34.92.80.18/ (username:uniman; password: friendintegrityfaith).

<Insert Fig. 9 here>

5.1. Use cases I: Consistency procedure of an HFLPR

The UI of this use case is already shown in Figure 5. In the Box of 'Algorithm Selection', a user can select the algorithm he wants to use. To use Algorithm I, the user needs to select his name '*Consistency procedure of an HFLPR'*. Then, based on his HFLPR, the user selects the number of alternatives in his HFLPR. The size of the input form will be changed according to the selected value in this box. Figure 10 shows the input form with 3 alternatives. After that, the user inputs the detailed information of his HFLPR. Here we use the HFLPR in Eq. (3) as an example. The user is

only required to input the key information of his HFLPR as we explained in Section 3. Now the UI is shown as Fig. 10.

<Insert Fig. 10 here>

If all the inputs are correct with no errors reported by form validation functions, the "submit" button will be activated. Once the form is submitted, the user's original HFLPR will be shown in the box of "User's Input HFLPR" and at this stage, the user can check his inputs are correct or not. The submitted data is sent to the data translator to normalize. Then, the algorithm holding serve calls the selected algorithm to do the calculation. The result that contains the consistency improved HFLPR is returned to webserver and then shown in the box of "Final Result" from the UI. This process is described in the use case diagram (Fig. 9). Fig. 11 describes the boxes of "User's Input HFLPR" and "Final Result" which are on the left side of the input form in the UI. Due to space limitations, we give the screenshots of the input and output separately.

<Insert Fig. 11 here>

5.2. Use cases II: Consensus-driven GDM with HFLPR

Currently, consensus-driven GDM algorithms have been widely used to solve decision-making problems in many aspects. For example, to assess the impact of hydropower stations on the environment in processing the flood discharge and energy dissipation, Ren et al. [41] used HFLPRs to describe the problem's uncertainty and fuzziness and applied a consensus-driven GDM algorithm. Given that the venture capital fund evaluation problem recently has become an important target to sustain social development in many countries. Algorithm II has been used to evaluate the performance of venture capital funds by [10]. However, all these studies only provided a theoretical way to solve problems. It is hard for policy makers or investors to utilize their algorithms to solve real problems. Through integrating with our proposed cloud-based system, we can provide a convenient way for users to solve their problems by applying advanced HFLPR algorithms.

When users select the algorithm of "*Consensus-driven group decision making with HFLPRs"* in the algorithm selection box, the UI is changed to Fig. 12. Serval information boxes are shown below the input form because the input of this algorithm is a group of HFLPRs. The user needs to input the HFLPR one by one. The process of inputting HFLPRs is the same as the previous use case. Now the "submit" button changes to "proceed". The key information of inputted HFLPRs is saved in the information box below. Users can check the information of the last inputted HFLPRs in the box of "User's Input HFLPRs". The "clear" button after the "proceed" button could clear all the information in the input form. There is another selection box above the HFLPR input form which is named "*consensus threshold value*". It lets the user select the consensus threshold value which will affect the stop condition and the

result of Algorithm II.

<Insert Fig. 12 here>

Assuming a user has an investment problem among three financial products. Four experts are invited to make the comparison between any of the two products. For an LTS S whose elements are s_0: extremely poor, s_1: very poor, s_2: poor, s_3: slight poor, s_4: fair, s_5: slight good, s_6: good, s_7: very good, and s_8: extremely good, four experts provide their pairwise comparisons on three alternatives as follows.

$$H^1 = \begin{bmatrix} \{s_4\} & \{s_5, s_6\} & \{s_4, s_5, s_6\} \\ \{s_2, s_3\} & \{s_4\} & \{s_2, s_3, s_4\} \\ \{s_2, s_3, s_4\} & \{s_4, s_5, s_6\} & \{s_4\} \end{bmatrix} \tag{7}$$

$$H^2 = \begin{bmatrix} \{s_4\} & \{s_4, s_5\} & \{s_5, s_6\} \\ \{s_3, s_4\} & \{s_4\} & \{s_3, s_4, s_5\} \\ \{s_2, s_3\} & \{s_3, s_4, s_5\} & \{s_4\} \end{bmatrix} \tag{8}$$

$$H^3 = \begin{bmatrix} \{s_4\} & \{s_4, s_5, s_6\} & \{s_4, s_5\} \\ \{s_2, s_3, s_4\} & \{s_4\} & \{s_3, s_4, s_5\} \\ \{s_3, s_4\} & \{s_3, s_4, s_5\} & \{s_4\} \end{bmatrix} \tag{9}$$

$$H^4 = \begin{bmatrix} \{s_4\} & \{s_5, s_6\} & \{s_5, s_6\} \\ \{s_2, s_3\} & \{s_4\} & \{s_3, s_4, s_5\} \\ \{s_2, s_3\} & \{s_3, s_4, s_5\} & \{s_4\} \end{bmatrix} \tag{10}$$

After inputting two HFLPRs, the button "submit" in the UI is activated. The "delete" button can erase the information of the last inputted HFLPRs and let the user re-input this HFLPR.

Once all the inputs are finished and the submit button is clicked, the information of all experts' HFLPRs is passed to the data translator module to normalize. Then, the algorithm holding server calls the selected algorithm to solve this GDM problem. The information about the final ranking of all three products and the number of rounds needed for this algorithm to reach consensus is returned to the webserver and shown in the UI, as shown in Fig. 13. The box of "User's Input HFLPR" shows the last inputted HFLPR which presents Eq. (10) and expert 4's preference. The final preference evaluating outcomes of three products are: $w^* = (0.4133, 0.2297,\ 0.3570)^T$, which indicates the ranking of the three products is $X_1 > X_3 > X_2$. This result shows that the first product is the best choice.

<Insert Fig. 13 here>

5.3. Use cases III: Algorithm upload and recommendation

The user case illustrated in Fig. 14 supports the algorithm upload and recommendation functionalities and describes the behavior of *the Algorithm provider*, *System hoster*, and *User* on the proposed DSS. In this use case, the Algorithm provide considers providing the API for their algorithm to the public while the System hoster is responsible for the success of integrating the new algorithm to the DSS. The user expects the DSS to recommend a suitable algorithm based on their preference or requirement. Two subcases are contained in this use case: *Uploading the new*

algorithm and *Algorithm recommendation*.

<Insert Fig. 14 here>

Uploading the new algorithm involves the actions of the System hoster and Algorithm provider. Algorithm providers who intend to have their algorithm hosted by the DSS have to provide the required algorithm information which is listed in Figure 6. This information is checked and saved to the Storge model database as a reference for the algorithm. The running environment of the provided algorithm code may vary depending on its programming language. Therefore, the System hoster needs to check and test the algorithm code to ensure its quality, then program the data interface that could be handled by the Data translation module. The modified code together with the runtime, system tools, system libraries and settings, and other things needed to run this code as an application, are containerized as a container image by using the docker. Once the System hoster uploads the image onto the customized container registry of DSS (See Fig. 2 for the uploading UI), a new algorithm pod is generated and saved to the pod pool by Kubernetes. With the help of this container management tool, the System hoster could update or remove this newly added algorithm. After the test, the user could run the new add algorithm from the DSS.

Algorithm recommendation describes the sequences of interactions related to recommending DM algorithm based on input information between the User and DSS. By using this functionality, the User is required to select his/her preference information on the UI shown in Figure 8. This information together with the input HFLPR(s) are sent to the backend of the webserver after the User's submission. The procedure of HFLPR(s) processing is the same as the use cases I and II. The preferred algorithm information is passed to algorithm recommendation modules, which checks all stored algorithm information from the Storge module database to match with the User's preferences or requirements. Once the decision is made, a request is sent to the Algorithm selector to initiate one execution of the recommended algorithm. After that, the recommended algorithm processes the input HFLPR(s) and returns the decision-making results to the web UI. Note, however, multiple algorithms may be recommended and called to run as described in Section 4. The corresponding results are compared in the backend of the webserver and only one result is chosen to show on the UI based on the selection criteria mentioned in Section 4.

Assume a user has a GDM problem as in Example 8 in [42] with the following algorithm preference: *0.8 (consensus threshold value), Additive*, *-normalization* and *HFLWA.* Among Algorithm II, GDM algorithm in [42] and [28], the DSS selects the first two algorithms to run since multiplicative is adopted in the third algorithm. Ultimately, Algorithm II is recommended, and its result is shown in the UI. This is based on the fact that Algorithm II only needs 1 iteration to reach a consensus under the input *consensus threshold value* while the GDM algorithm in [42] needs 3 iterations. The final output is illustrated in Figure 15.

<Insert Fig. 15 here>

6. Feedback from preference relation and group decision making expert

The Use Cases shown in Section 5 confirm that the proposed DSS is correctly implemented with respect to the conceptual model, while the feedback from experts confirms the proposed DSS provides a reasonable picture of a real-world system to people who are knowledgeable of the real-world system. Therefore, the aim of the expert feedback study is to capture the expert views about the effectiveness of the proposed system. The study verifies whether or not the idea of our cloud-based DSS is the right artefact to fulfill the following goals as claimed in this paper: 1) to support researchers to integrate their HFLPR algorithms into the system conveniently; 2) to help users/DMs solve their GDM problem by applying suitable algorithms; 3) and to provide an online platform for users or decision makers to test, compare and evaluate different research work and for researchers to get feedback on their work.

Due to the Covid-19 epidemic, the study was conducted in the form of an online demonstration. The views of study-participants on the proposed web-based DSS were captured using a questionnaire survey tool (presented in the Appendix). A total of 12 respondents participated in the survey. Three questions are used to capture the participants' background: 1) field of work; 2) level of expertise in preference relations/ group decision making; 3) holding role or position. The information of participants' backgrounds is presented in Table I. Since some participants hold more than one position within their fields, the sum of responses for the role or position held exceeds the total number of participants.

<Insert Table 3 here>

The participants in the field of academics were asked questions relating to what purposes that our proposed DSS would serve in their research work. This question in the survey focused on establishing the utility and the ability of the functionalities and purposes of the proposed DSS to produce desired or intended results in relation to our claimed goals 1) and 3) (the claimed goals of this study is listed in the start of this section). The number of responses to different purposes is indicated in parenthesis following each option:

(a) Uploading your HFLPR algorithm to the system and getting a customized and user-friendly UI for users to solve their GDM problems (3);

(b) Uploading your HFLPR algorithm to the system and enabling peer researchers to test/evaluate your work in a public platform in an efficient way (4);

(c) Using the system to test/evaluate others' work without downloading the algorithm code and setting the running environment (5);

(d) No help (1);

(e) other (1): the system is able to run the existing algorithms and compare the outcome with the researcher's newly designed algorithm.

The study then explored the programming language most frequently used in the research area of HFLPRs. The answer shows that MATLAB is the most commonly used programming language with the number of 6, followed by Python with the number of 4, the number of people who select JAVA and R are the same: 2, no one select C&C++. All these programming languages are supported by our multi-language support framework.

Further, the participants were required to rate the effectiveness of the methods for the researcher to test/evaluate other's HFLPR algorithms, and for the user to find a suitable HFLPR algorithm solving his specific GDM problems. The rating was based on a five-point Likert scale: ineffective (1), slightly effective (2), rarely effective (3), effective (4), and very effective (5). The result shows on average, the use of 'Redo other research work' was rated lowest at 1.66, then 'Download others' work and set the running environment for it or run their work in virtual machine' at 2.16 which was much lower than the use of 'IT-assisted solutions such as our proposed web-based decision support system' at 4.33.

The next question requires the participants to indicate the expected benefits of using a web-based platform such as our proposed decision support system from the user's perspective. The participants were able to select multiple choices. The result of expected benefits was as follows.

(a) provides easily accessed and user-friendly portal and UI (8);

(b) helps to solve GDM problems without downloading the algorithm code and setting the running environment (12);

(c) recommends a suitable algorithm for the user's specific GDM problem based on reasonable classification criteria for all saved algorithms (7).

The study then explored the concerns that might prevent participants from using the proposed system. This question also allows participants to select multiple options, and the result is listed as follows.

(a) System security and integrity (5);

(b) algorithm or data privacy (8);

(c) reliability of the recommended GDM algorithm and the running result (9);

(d) the programming language of your GDM algorithm is not supported (1);

(e) your GDM algorithm needs highly customized UI (4);

(f) auditability of the system (4).

Furthermore, an assessment of participants' likelihood to recommend the use of our proposed system to solve their GDM problems and upload their algorithms to the system was conducted. The participants were asked to indicate their likelihood on the scale from 1 (very unlikely) to 10 (very likely). The following is the result: the likelihood of 9 (2 responses), the likelihood of 8 (5 responses), the likelihood of 7 (3 response) and the likelihood of 5 (2 responses).

To conclude the study, the opinions of participants were sought out on 4 aspects about the functionality of our proposed DSS. The Likert scale from 1 to 5 is used in this question: strongly disagree (1), disagree (2), neither agree nor disagree (3), agree (4), and strongly agree (5). The findings showing the average level of agreement to the general statements are as follows.

(a) I find the Web-based decision support system is easy to access and use (4);

(b) I find the multi-language support system is needed to support the multilingual consistency and consensus-based decision support algorithms (4.16);

(c) I find the recommendation system/module which could recommend a suitable GDM based on users' requirements to solve their problem is useful (3.66);

(d) I find that the classification criteria of the consistency and consensus-based decision support algorithms by the proposed system (this paper) is reasonable (3.5);

In conclusion, the expert feedback study described in this section constitutes primarily a formative evaluation activity in design science research in the following two aspects: (1) it interprets the expectations of our proposed system in relation to the utility and efficacy; (2) it produces a foundation for confirming the appropriateness of decisions that led to our proposed system artefact design. On both counts, the evaluation described in this section reaches its overall objective.

7. Conclusion

With the development of internet technology, web and cloud based DSSs have become a new trend in DSS research since it features many advantages. This paper presented a web-based group DSS for consistency and consensus-based algorithms with HFLPRs, from the design to implementation. The presented cloud-based DSS provided a multi-language support framework to manage and run multilingual consistency and consensus-based decision support algorithms. Customized and user-friendly UI were provided for different algorithms to allow users to input their preferences with HFLPR format. With the multi-language support framework and data translation workflow, existing and future approaches and algorithms for GDM with HFLPRs can be integrated into the proposed support system directly. Furthermore, an algorithm recommendation module was designed in the proposed system to

help users choose suitable decision support algorithms. The user cases showed in the paper confirmed that the proposed DSS was correctly implemented with respect to the conceptual model. The expert feedback study also verified that the idea of our cloud-based DSS is the right artefact to fulfill the goals claimed in this paper.

This research work has serval limitations that provide the directions for future research. First, the DSS currently can only drive the GDM algorithms that utilize HFLPR or a similar format as the preference information representation form. Although the applicability of the data transfer workflow has been discussed in section 3, the input of the DM's preference information is still restricted to numbers within the format of Satty's 1-9 scale. In large-scale GDM problems, it is natural to allow experts to express preferences in various formats considering the heterogeneity of experts. For instance, Wu and Liao [43] propose a consensus reaching process for large-scale GDM with heterogeneous reference information. How to apply such kind of algorithms to our system and transfer various formats of preference (such as intuitionistic fuzzy sets, interval-valued intuitionistic fuzzy set, and probabilistic linguistic term set) to a single format to aggregate preference information are the problems we plan to investigate in our future work.

Secondly, the proposed GDM algorithms and most of the reviewed works assume the DM could provide the exact and numerical evaluation on alternatives, which is difficult in the practical implementation. The DM may only have serval sentences to describe his/her felling for an item, just like the online shop reviews. In this case, how to get their evaluation score from their review text is the problem. To solve this, for instance, Wu and Liao [44] used a sentiment analysis tool "Stanford CoreNLP" to estimate a reviewer's probabilistic evaluation for a product on 1-5 scale. Kou et al. [45] tested and implemented different machine learning models (e.g., support vector machine, K-nearest neighbors) to train a sentiment classifier and then predict reviewers' probabilistic evaluation based on input review comments. One could also deploy and host the advanced sentiment classification models such as Transformer, TextCNN, etc. Considering all such models require powerful computation resources, our proposed cloud-based DSS with the benefit of cloud computing could provide a practical solution. Therefore, using advanced sentiment classification models within our cloud-based DSS framework to learn DM's preference and estimating their probabilistic evaluation from the input review comments is another direction worth researching in our future work.

Finally, the algorithm extensibility of the DSS is relatively low since the uploading service of DSS is hardcoded that can only be implemented by the hoster. Further, the algorithm uploader may also want to design their web UI based on the formats of preference information utilized by their algorithm. Therefore, an automated algorithm uploading workflow that integrates with web design tools is another future work for our proposed DSS.

Acknowledgments

The work was supported by the National Natural Science Foundation of China (71771156, 71971145, 72171158). We wish to thank Long Jin for his support and contribution to implementing the DSS.

References

[1] Herrera F, Herrera-Viedma E, Verdegay JL. A sequential selection process in group decision making with a linguistic assessment approach. *Inf Sci*. **1995**;*85(4)*:223-239.

[2] Zhu B, Xu ZS. Consistency measures for hesitant fuzzy linguistic preference relations. *IEEE Trans Fuzzy Syst*. **2014**;*22(1)*:35-45.

[3] Rodriguez RM, Martinez L, Herrera F. Hesitant fuzzy linguistic term sets for decision making. *IEEE Trans Fuzzy Syst*. **2012**;*20(1)*:109-119.

[4] Liao HC, Xu ZS, Zeng XJ, Merigó JM. Qualitative decision making with correlation coefficients of hesitant fuzzy linguistic term sets. *Knowl-Based Syst*. **2015**;*76*:127-138.

[5] Liao HC, Xu ZS, Herrera-Viedma E, Herrera F. Hesitant Fuzzy Linguistic Term Set and Its Application in Decision Making: A State-of-the-Art Survey. *Int J Fuzzy Syst*. **2018**;*20(7)*:2084-2110.

[6] Zhao N, Xu ZS, Liu FJ. Group decision making with dual hesitant fuzzy preference relations. *Cognit Comput*. **2016**;*8(6)*:1119-1143.

[7] Liu NN, He Y, Xu ZS. A new approach to deal with consistency and consensus issues for hesitant fuzzy linguistic preference relations. *Appl Soft Comput*. **2019**;*76*:400-415.

[8] Tang J, Meng FY, Xu ZS, Yuan RP. Qualitative hesitant fuzzy group decision making: An additively consistent probability and consensus-based perspective. *Expert Syst*. **2020**;*37(3)*:e12510.

[9] Liao HC, Jiang LS, Fang R, Qin R. A consensus measure for group decision making with hesitant linguistic preference information based on double alpha-cut. *Appl Soft Comput*. **2021**; *98*: 106890. doi:10.1016/J.ASOC.2020.106890

[10] Ren PJ, Liu ZX, Zhang WG, Wu XL. Consistency and Consensus Driven for Hesitant Fuzzy Linguistic Decision Making with Pairwise Comparisons. *arXiv Prepr arXiv211104092*. Published online 2021.

[11] Power DJ. *Decision Support Systems: Concepts and Resources for Managers*. *Greenwood Publishing Group;* **2002**.

[12] Rahimi S, Gandy L, Mogharreban N. A web-based high-performance multicriteria decision support system for medical diagnosis. *Int J Intell Syst*. **2007**; *22(10)*: 1083-1099.

[13] Zhang SF, Goddard S. A software architecture and framework for Web-based distributed Decision Support Systems. *Decis Support Syst*. **2007**; *43(4)*: 1133-1150.

[14] Alonso S, Herrera-Viedma E, Chiclana F, Herrera F. A web based consensus support system for group decision making problems and incomplete preferences. *Inf Sci*. **2010**; *180(23)*: 4477-4495.

[15] Richards M. *Software Architecture Patterns*. *O'Reilly Media, Incorporated 1005 Gravenstein Highway North, Sebastopol, CA95472;* **2015**.

[16] Li Q, Tang QL, Chan IT, Wei HL, Pu YD, Jiang HZ, Li J, Zhou J. Smart manufacturing standardization: Architectures, reference models and standards framework. *Comput Ind*. **2018**; *101*: 91-106.

[17] Vresk T, Čavrak I. Architecture of an interoperable IoT platform based on microservices. *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE,* May, **2016**; Opatija, Croatia.

[18] Shree D, Kumar Singh R, Paul J, Hao A, Xu S. Digital platforms for business-to-business markets: A systematic review and future research agenda. *J Bus Res*. **2021**; *137*: 354-365.

[19] Han H, Trimi S. Towards a data science platform for improving SME collaboration through Industry 4.0 technologies. *Technol Forecast Soc Change*. **2022**; *174*: 121242. doi:10.1016/J.TECHFORE.2021.121242

[20] Bernstein D. Containers and Cloud: From LXC to Docker to Kubernetes. *IEEE Cloud Comput*. **2014**; *1(3)*: 81-84.

[21] p3.xlarge. https://aws.amazon.com/ec2/instance-types/p3/?nc1=h_ls. Accessed May 16, 2020.

[22] t2.micro. https://aws.amazon.com/ec2/instance-types/. Accessed May 16, 2020.

[23] REST API. https://en.wikipedia.org/wiki/Representational_state_transfer. Accessed August 26, 2020.

[24] Kubernetes Documentation. https://kubernetes.io/docs/concepts/. Accessed June 25, 2020.

[25] Zadeh LA. The concept of a linguistic variable and its application to approximate reasoning-I. *Inf Sci*. **1975**; *8(3)*: 199-249.

[26] Xu ZS. Deviation measures of linguistic preference relations in group decision making. *Omega*. **2005**; *33(3)*: 249-254.

[27] Liao HC, Xu ZS. Approaches to manage hesitant fuzzy linguistic information based on the cosine distance and similarity measures for HFLTSs and their application in qualitative decision making. *Expert Syst Appl*. **2015**;*42(12)*: 5328-5336.

[28] Zhang ZM, Chen S-M. A consistency and consensus-based method for group decision making with hesitant fuzzy linguistic preference relations. *Inf Sci*. **2019**; *501*: 317-336.

[29] Liao HC, Xu ZS, Zeng XJ. Distance and similarity measures for hesitant fuzzy linguistic term sets and their application in multi-criteria decision making. *Inf Sci*. **2014**; *271*:125-142.

[30] Angular Form. https://angular.io/api/forms/Form. Accessed July 7, 2020.

[31] Saaty TL. *The Analytic Hierarchy Process*. New York McGraw-Hill. **1980**.

[32] Wu ZB, Xu JP. Managing consistency and consensus in group decision making with hesitant fuzzy linguistic preference relations. *Omega*. **2016**; *65*: 28-40.

[33] Zhang ZM, Wang C, Tian XD. A decision support model for group decision making with hesitant fuzzy preference relations. *Knowl-Based Syst*. **2015**; *86*: 77-101.

[34] Chen WJ, Zou Y. Group decision making under generalized fuzzy soft sets and limited cognition of decision makers. *Eng Appl Artif Intell*. **2020**; *87*: 103344. doi:10.1016/J.ENGAPPAI.2019.103344

[35] Singh RK, Benyoucef L. A consensus based group decision making methodology for strategic selection problems of supply chain coordination. *Eng Appl Artif Intell*. **2013**; *26(1)*: 122-134.

[36] Zhu B, Xu ZS. Analytic hierarchy process-hesitant group decision making. *Eur J Oper Res*. **2014**; *239(3)*: 794-801.

[37] Meng FY, Tang J, An QX, Chen XH. A new procedure for hesitant multiplicative preference relations. *Int J Intell Syst*. **2019**; *34(5)*: 819-857.

[38] Scoping SIG, Taskforce T. Information supplement: Pci dss tokenization guidelines. *Stand PCI Data Secur Stand (PCI DSS)*. **2011**;*24*.

[39] Lee LW, Chen SM. Fuzzy decision making based on likelihood-based comparison relations of hesitant fuzzy linguistic term sets and hesitant fuzzy linguistic operators. *Inf Sci*. **2015**; *294*: 513-529.

[40] Ribu K. *Estimating Object-Oriented Software Projects with Use Cases*.; 2001. Accessed July 18, 2020.

[41] Ren PJ, Zhu B, Xu ZS. Assessment of the Impact of Hydropower Stations on the Environment with a Hesitant Fuzzy Linguistic Hyperplane-Consistency Programming Method. *IEEE Trans Fuzzy Syst*. **2018**;*26(5)*:2981-2992.

[42] Wu P, Zhou LG, Chen HY, Tao ZF. Additive consistency of hesitant fuzzy linguistic preference relation with a new expansion principle for hesitant fuzzy linguistic term sets. *IEEE Trans Fuzzy Syst*. **2019**; *27(4)*: 716-730.

[43] Wu Z, Liao HC. A consensus reaching process for large-scale group decision making with heterogeneous preference information. *Int J Intell Syst*. **2021**; 36: 4560–4591.

[44] Wu XL, Liao HC. Modeling personalized cognition of customers in online shopping. *Omega*.

2021;*104:*102471. doi:10.1016/J.OMEGA.2021.102471

[45] Kou G, Yang P, Peng Y, Xiao H, Xiao F, Chen Y, Alsaadi F. A cross-platform market structure analysis method using online product reviews. *Technol Econ Dev Econ*. **2021**;*27(5)*: 992-1018.

Appendix

The questionnaire used in expert feedback study is shown in Fig. A.1 and A.2.

<Insert Fig. A.1 here>

<Insert Fig. A.2 here>

Table in this paper.

TABLE I
PARTICIPATES' BACKGROUND RESULTS

Field		*Expertise*		*Role / position*	
Academic	8	Basic	25%	Academic	8
Professional/ industry	4	Intermediate	41.6%	IT developer / systems engineer / architect	6
		Expert	33.3%	Business / IT consultant	2
				Executive / manager	1
				Fuzzy system / decision support professional	7
				others	1

Figures in this paper.

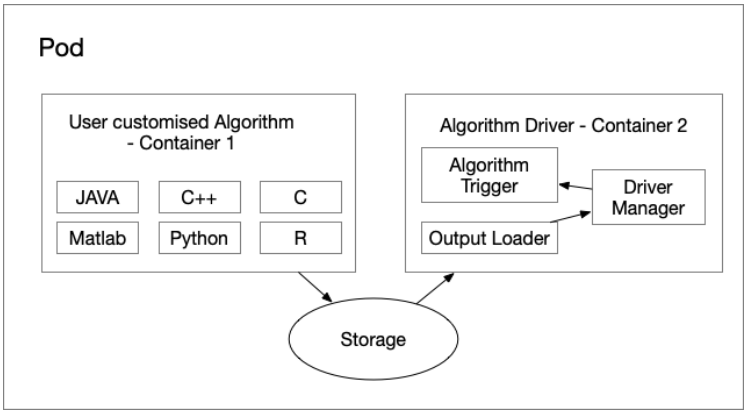

Fig. 1. Pod design

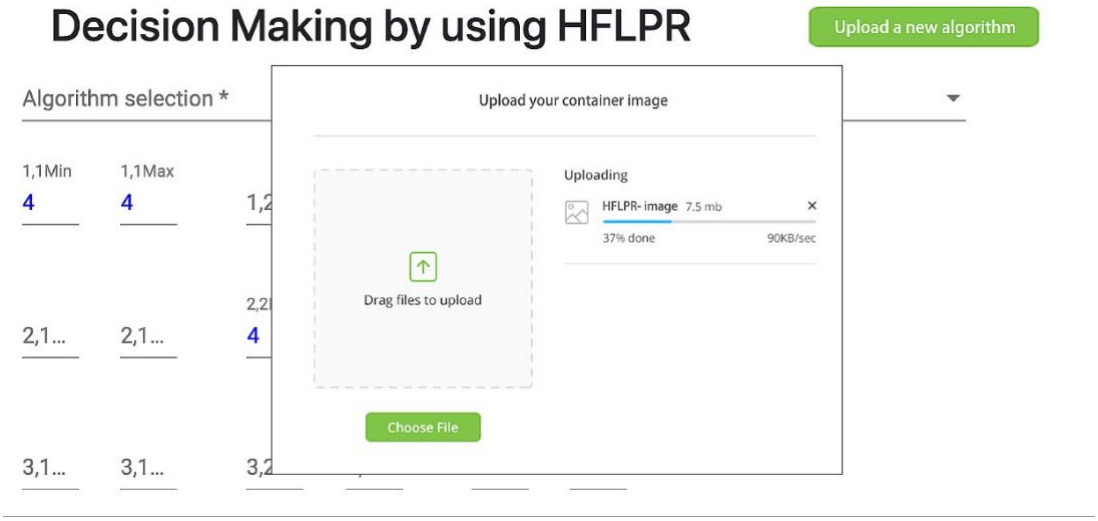

Fig. 2. The user interface of uploading algorithm images

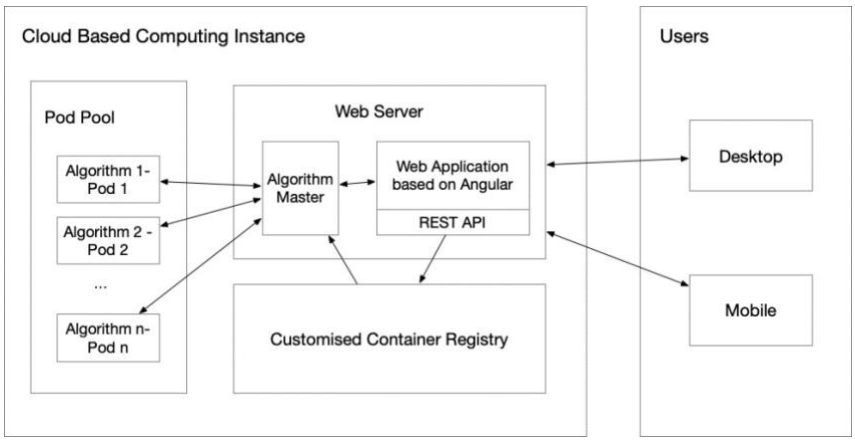

Fig. 3. System architecture

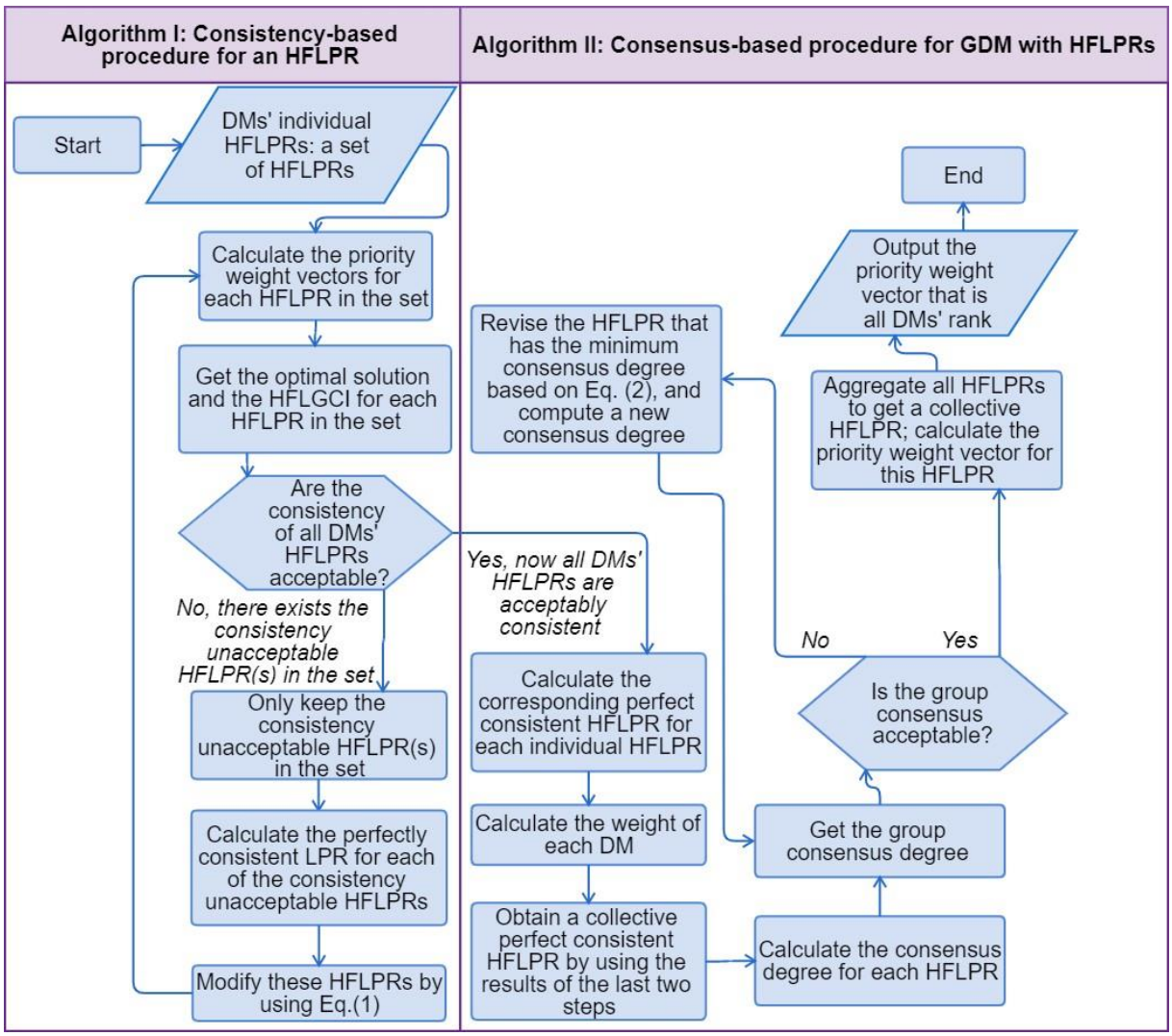

Fig. 4. Brief diagram of Algorithms I and II.

Fig. 5. User's input form for an HFLPR.

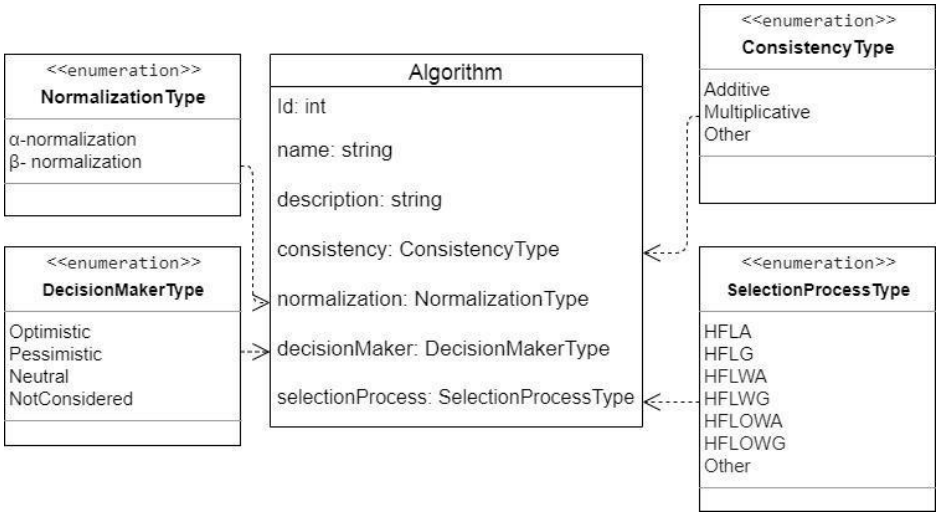

Fig. 6. The algorithm class in the storage module database.

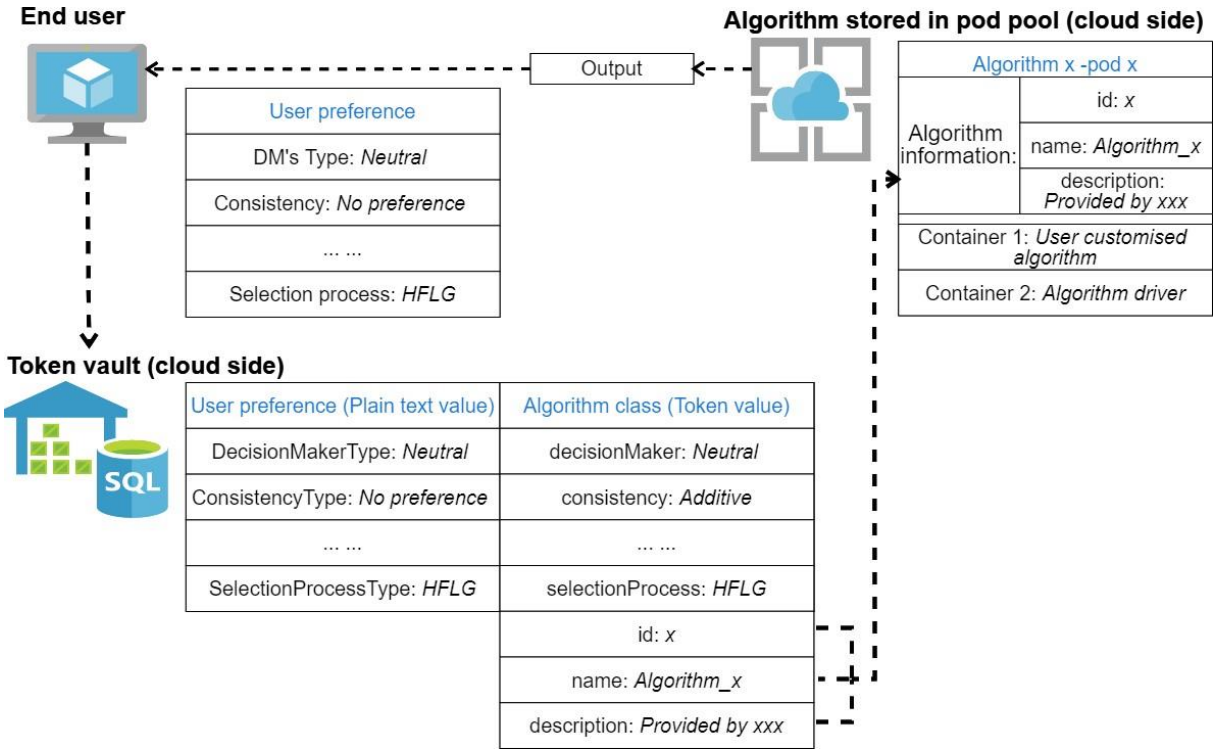

Fig. 7. Tokenization process illustration.

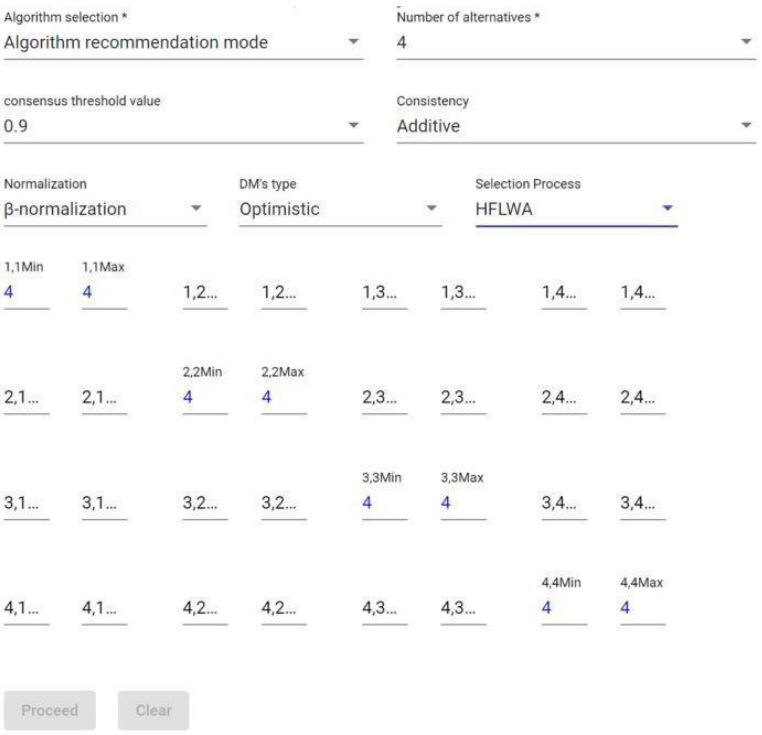

Fig. 8. The algorithm recommendation module UI.

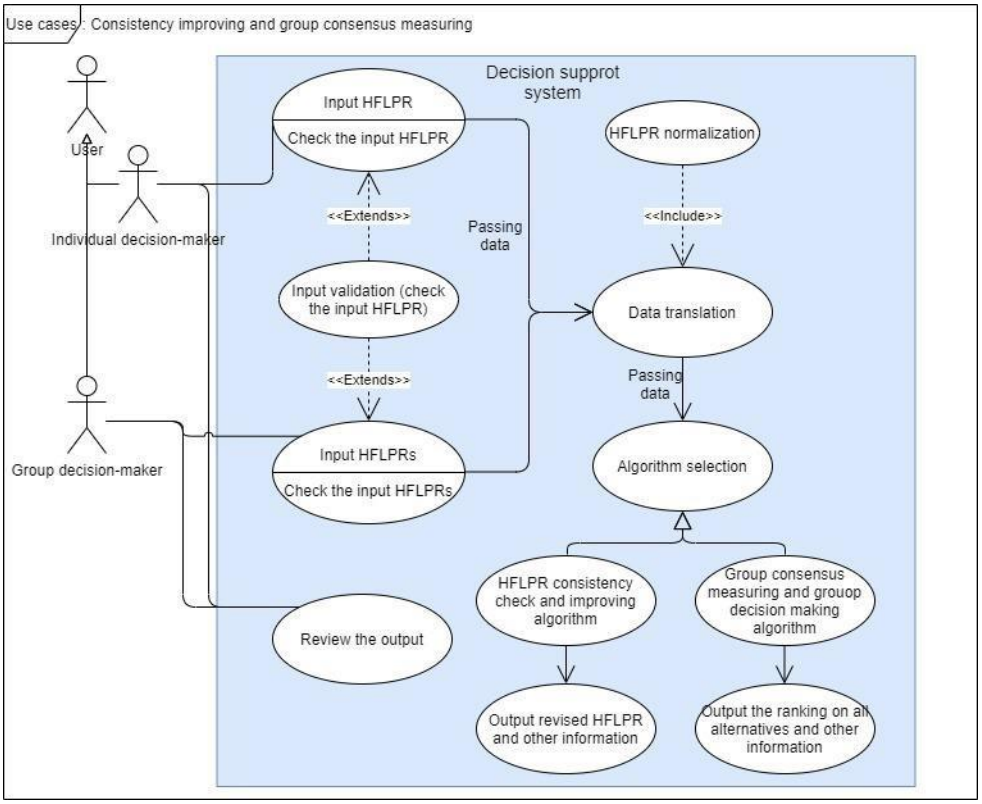

Fig. 9. Use case diagram of Use cases I and II.

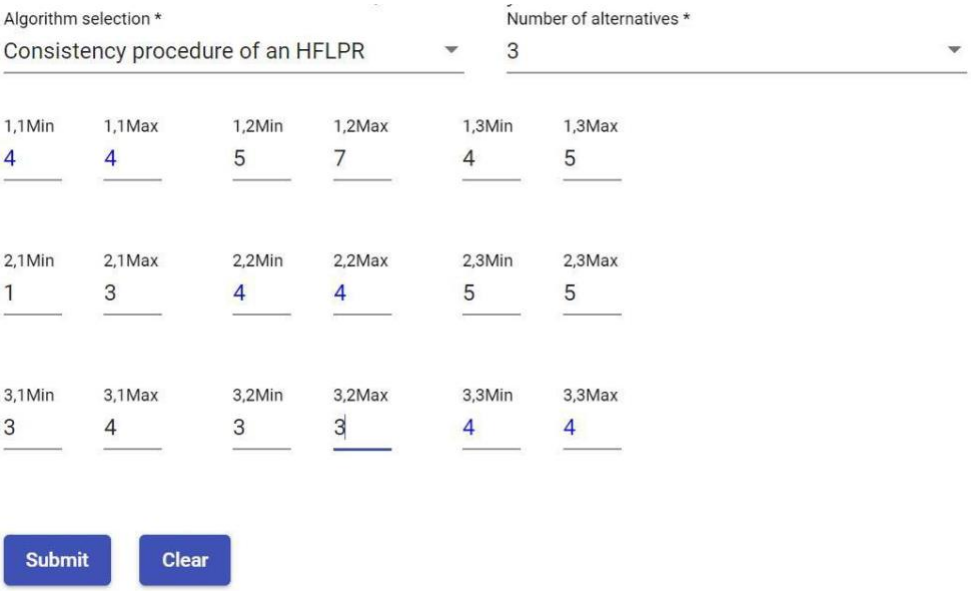

Fig. 10. The input information in the use case I.

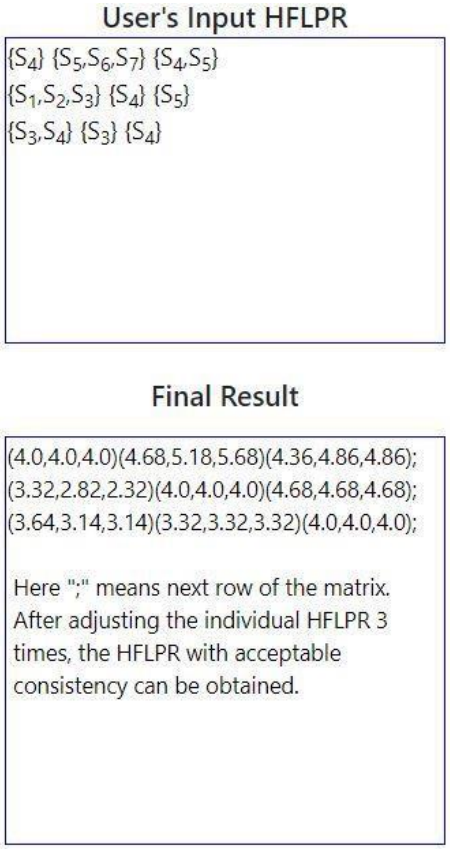

Figure 11. The output information in the use case I.

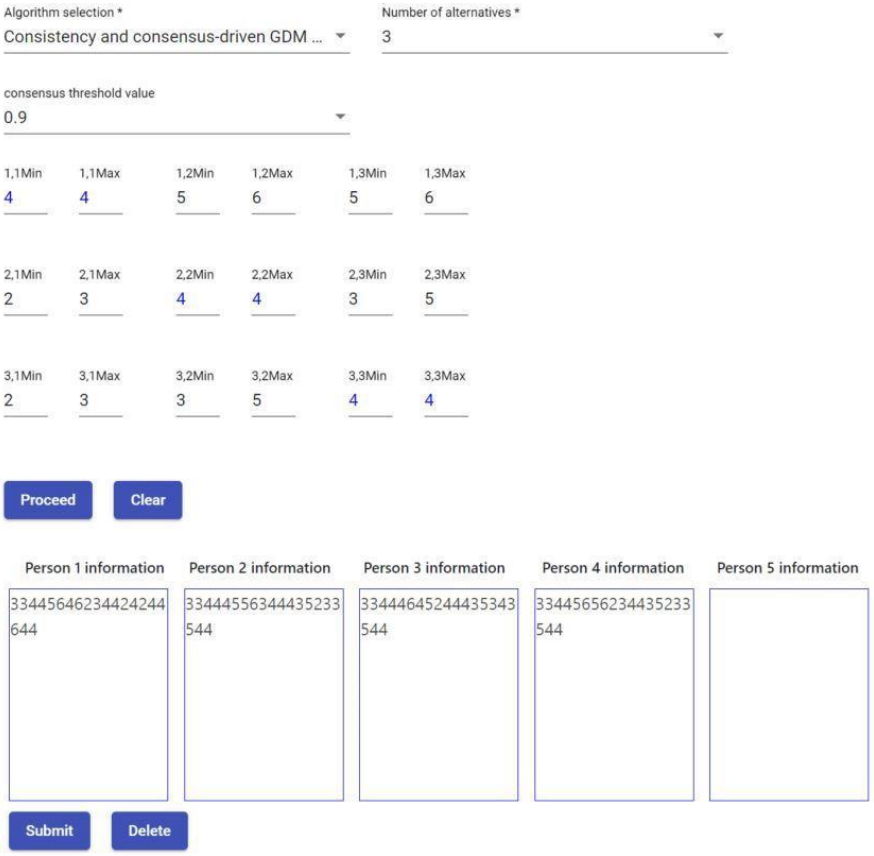

Fig. 12. The UI of Algorithm II.

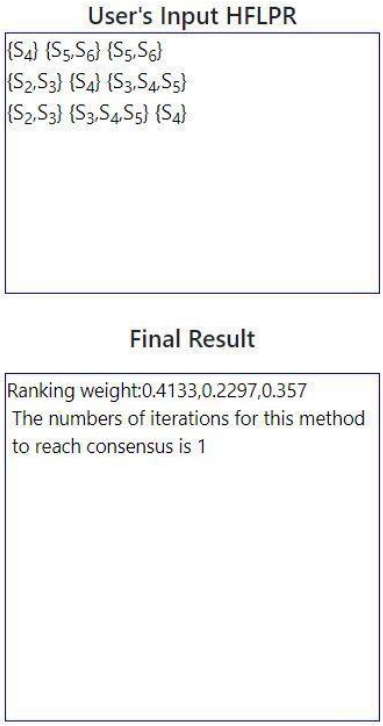

Fig. 13. The output information in use case II.

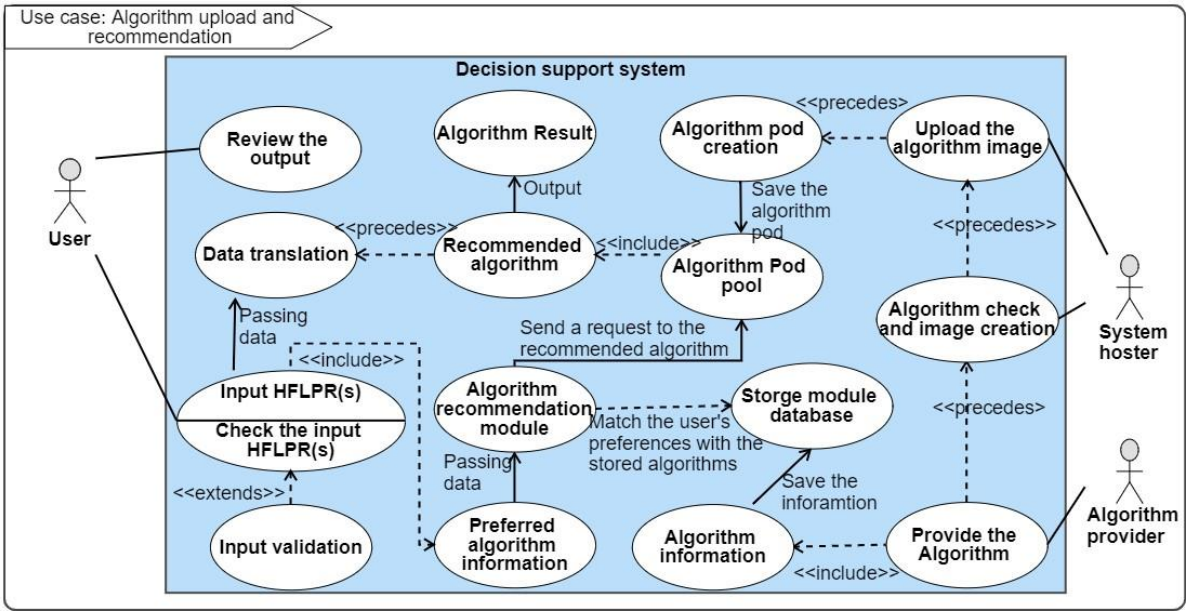

Fig. 14. Use case diagram of Algorithm upload and recommendation.

Fig. 15. Algorithm recommendation result for Example 8 in Ref. [42].

Research Study on the Web-based Group Decision Making Support System for Consistency and Consensus-based Algorithms with HFLPR

Participant information

Thank you for your time. All the data you provide will remain fully anonymous and subject to General Data Protection Regulation (GDPR) and the United Kingdom (UK) data protection laws. In case of any queries please contact Dr Zixu Liu at zixu.liu@manchester.ac.uk

The aim of this anonymous questionnaire is to establish your views about our proposed web-based group decision support system (WGDSS) in relation to your work. The findings will contribute to further improvements to the WGDSS so as to enable the system to provide a seamless user experience of its functionalities.

Please indicate your consent for using the responses provided in the questionnaire for research purposes by ticking this box: ☐ Date: _____

Background *(please tick all that apply)*

Field:
- Academic ☐
- Professional / Industrial ☐

Level of expertise in Preference Relation/ Group Decision Making (GDM):
- Basic ☐
- Intermediate ☐
- Advanced ☐

Role / Position:
- ☐ Academic
- ☐ IT Developer / Systems Engineer / Architect
- ☐ Business / IT Consultant
- ☐ Executive / Manager
- ☐ Fuzzy System/ Decision Support System Professional
- Other: .....

1. What do you think the WGDSS platform can do in your research/work? *(please skip this question if you are not academic researcher; please tick all that apply)*

a. Upload your hesitant fuzzy linguistic preference relation (HFLPR) algorithms to the system and get a customized and user-friendly User Interfaces for user to solve their GDM problems ☐

b. Upload your HFLPR algorithms to the system and enable peer researchers to test/evaluate your work/algorithms without providing them the code for privacy reasons ☐

c. Use the system to test/evaluate others' HFLPR algorithms without downloading the algorithm code and setting the running environment ☐

d. No help for your research/work ☐

Other, please specify: .....

2. Please tick the programming language(s) most frequently used in your research/work for hesitant fuzzy linguistic preference relation algorithms.

a. C & C++ ☐
b. JAVA ☐
c. Python ☐
d. MATLAB ☐
e. R ☐

Other, please specify: .....

Fig. A.1. The questionnaire page 1.

3. Based on your experience, please rate the effectiveness of the following ways for 1) testing/evaluating other researchers' HFLPR algorithms and comparing the results with yours, 2) finding a suitable HFLPR algorithm to solve your specific GDM problems, on the scale from 1 to 5:

ineffective (1), *slightly effective* (2), *rarely effective* (3), *effective* (4), *very effective* (5).

		1 2 3 4 5	
a.	Re-do other researchers' work when you cannot get the original code from them	ineffective ☐☐☐☐☐	very effective
b.	Download other researchers' work and set the running environment for their code or run their code in virtual machine such as docker	ineffective ☐☐☐☐☐	very effective
c.	IT-assisted solutions such as the **WGDSS** (test other researchers' HFLPR algorithms on the web-based platform without re-do their work and set the running environment; use the algorithm recommendation module to get a suitable HFLPR algorithms to solve their decision making problems)	ineffective ☐☐☐☐☐	very effective

4. What benefits do you expect from using a system such as the WGDSS in decision maker's (user's) perspective? *(please tick all that apply)*

a. Easy access (web-based), user-friendly User Interfaces ☐
b. Solving group decision problems without downloading the algorithm code and setting the running environment ☐
c. Using the algorithm recommendation module to get a suitable GDM algorithm (WGDSS classifies GDM algorithms with multi-criteria and saves their information in the database) ☐

Other, please specify: ……

5. What concerns might prevent you from using the WGDSS or uploading your algorithm to the WGDSS? *(please tick all that apply)*

a. System security and integrity ☐
b. Algorithm or data privacy ☐
c. Reliability of the recommended GDM algorithm and the running result ☐
d. The programming language of your GDM algorithm is not supported ☐
e. Your GDM algorithm need highly customized UI ☐
f. Auditability of the system ☐

Other, please specify: ……

6. Considering the WGDSS description, how likely would you use it in the following purposes: 1) solving your decision making problem; 2) uploading your work/algorithms to the system — on the scale from 1 (*very unlikely*) to 10 (*very likely*)?

1 2 3 4 5 6 7 8 9 10

very unlikely ☐☐☐☐☐☐☐☐☐☐ very likely

7. Is there any functionality that is desired but not currently captured by the WGDSS?

……

8. Please provide the following information and tick (✓) where appropriate:

strongly disagree (1), *disagree* (2), *neither agree nor disagree* (3), *agree* (4), *strongly agree* (5).

		1 2 3 4 5	
a.	I find the Web-based decision support system is easily to access and use.	disagree ☐☐☐☐☐	agree
b.	I find the multi-language support system is needed to support the multilingual consistency and consensus-based decision support algorithms.	disagree ☐☐☐☐☐	agree
c.	I find the recommendation system/module which could recommend a suitable GDM based on users' requirements to solve their problem is useful.	disagree ☐☐☐☐☐	agree
d.	I find that the classification criteria of the consistency and consensus-based decision support algorithms by the **WGDSS** is reasonable (classification criteria: *consistency type, normalization method, decision Maker type* and *selection Process* …)	disagree ☐☐☐☐☐	agree

Thank you very much for your cooperation! ☺

Fig. A.2. The questionnaire page 2.