Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2015) 23: 279 – 297

doi:10.3906/elk-1302-3

A new systematic and flexible method for developing hierarchical decision-making models

Ulaş BELDEK[1,*], **Mehmet Kemal LEBLEBİCİOĞLU**[2]

[1]Department of Mechatronics Engineering, Faculty of Engineering, Çankaya University, Ankara, Turkey
[2]Department of Electrical and Electronics Engineering, Faculty of Engineering, Middle East Technical University, Ankara, Turkey

Received: 01.02.2013 • **Accepted:** 09.04.2013 • **Published Online:** 12.01.2015 • **Printed:** 09.02.2015

Abstract: The common practice in multilevel decision-making (DM) systems is to achieve the final decision by going through a finite number of DM levels. In this study, a new multilevel DM model is proposed. This model is called the hierarchical DM (HDM) model and it is supposed to provide a flexible way of interaction and information flow between the consecutive levels that allows policy changes in DM procedures if necessary. In the model, in the early levels, there are primary agents that perform DM tasks. As the levels increase, the information associated with these agents is combined through suitable processes and agents with higher complexity are formed to carry out the DM tasks more elegantly. The HDM model is applied to the case study 'Fault degree classification in a 4-tank water circulation system'. For this case study, the processes that connect the lower levels to the higher levels are agent development processes where a special decision fusion technique is its integral part. This decision fusion technique combines the previous level's decisions and their performance indicator suitably to contribute to the improvement of new agents in higher levels. Additionally, the proposed agent development process provides flexibility both in the training and validation phases, and less computational effort is required in the training phase compared to a single-agent development simulation carried out for the same DM task under similar circumstances. Hence, the HDM model puts forward an enhanced performance compared to a single agent with a more sophisticated structure. Finally, model validation and efficiency in the presence of noise are also simulated. The adaptability of the agent development process due to the flexible structure of the model also accounts for improved performance, as seen in the results.

Key words: Decision making, decision fusion, agents, genetic algorithms

1. Introduction

The design of a decision-making (DM) system requires knowledge about the complexity and nature of the particular DM problem under investigation. A DM system may include a single expert or multiple experts, each of which may also be considered as DM units with special duties. Problems with high complexity are better dealt with by separating the problem into simpler circumstances and developing fundamental DM units for each circumstance. When multiple experts are involved in the process, the design of the overall DM system actually depends on providing a suitable organization between these experts and finding advantageous ways of combining their individual actions [1].

Most of the standard DM systems assess the relative importance of alternatives [2]. Similarly, advanced expert systems employ several agents for this purpose. In expert systems, each expert takes part in different

*Correspondence: u.beldek@cankaya.edu.tr

actions and each has a different degree of inspiration to fulfill their task. Therefore, when multiple experts are involved in a DM task, arranging the relations between them, or briefly, how the group DM [3,4] is performed, becomes the critical issue. Group DM systems can be created in many different ways: providing a consensus among experts possessing different amounts of influence in a group [5,6], DM systems based on voting or dominance of a group of experts [4], weighted averaging methods [7], and aggregation [8] are the typical actions and the processes that help in the formation of final decisions.

When multiple experts enroll in a DM process, one of the indispensable issues is to follow intelligent techniques to coordinate the experts' actions and combine their decisions. Data fusion and evaluation methods as well as artificial intelligence techniques [9] can be applied together with different DM frameworks in order to guarantee an efficient way of decision aggregation [10].

In most of the expert system applications, interaction between experts is configured directly in a single level. However, as the complexity of the application increases, it is difficult and inefficient to provide the organization of experts so effortlessly. Dealing with a huge amount of data coming from different observations [11]; the necessity of task planning, gathering, and evaluating information from different heterogeneous sources [12]; the essentiality for applicability of a learning paradigm in different frameworks in order to handle DM problems more accurately [13]; and the comfort in establishing organization between agents without difficulty [14] are the typical reasons to use hierarchical and multilevel frameworks. In multilevel DM frameworks, experts in any level have a direct or indirect impact over the actions and decision of the experts' at other levels. The control of lower-ranked experts by the supervision of a leader expert is the most common framework; however, in some cases, experts with similar authority can also influence each other [15]. Hierarchical system definitions [16] can be applied and multiagent cooperation architectures for data fusion [17] can be used for hierarchical organizations. In [18], a complex structured DM model was put forward, which is a hierarchically organized DM framework to represent complex structured knowledge. Developing agent structures that function as experts for DM problems is also an important area of investigation for researchers [14,19]. Machine learning tools and heuristic search methods like genetic algorithms (GAs) [20] are widely employed to develop agent structures. However, integrating them into complicated multilevel DM systems is generally a demanding process. As an alternative, classical problem-specific tools are more preferred in multilevel DM systems with respect to agent structures.

If agents are to be employed for DM problems with a multifaceted nature, the main difficulty becomes selecting and employing an appropriate DM model to yield a high performance. Moreover, too much computational power is necessary and too many variables should be taken into account to develop the agents. Most of the DM systems given in the previous paragraph, although organized in a multilevel or hierarchical manner, employ special techniques to overcome DM tasks. Moreover, we have not encountered in the literature a general agent development process framework for hierarchical systems that combines lower-level information (decisions) with the help of a suitable decision fusion method by utilizing machine-learning techniques as an integral part of the development process. Hence, the formulation and construction of a suitable DM system architecture to deal with DM problems with high complexity in a systematic and a flexible way that permits policy changes with the help of processes are important issues. How should the DM system architecture and the processes and procedures in this architecture be organized in order to get high performance and higher flexibility and in order to promote the adaptation capacity of the system? In this study, a practical solution for these research questions is sought. The main aim of this study is to establish a new multilevel DM system architecture as in [14] for handling DM problems with high complexity. In this architecture, the main aspiration is to obtain

and utilize agents with increasing complexity in a hierarchical manner for the DM task(s) and carry on this procedure until the performance of the agents is sufficient. We suppose that this architecture has considerable potential to support information flow from lower to higher levels effectively and offers high adaptation capability and flexibility, as the processes can be reconfigured and the agent structures in the proposed DM system can be adjusted easily to enable policy changes in problem solving.

The remainder of the paper is as follows: Section 2 is devoted to the explanation of the hierarchical DM (HDM) model and the agent development process. In Section 3, details about the case study and DM task are given. In Section 4, two applications are carried out: first, the proposed model with an agent development process is applied for the DM task. In the second application, a single agent is developed for the same purpose. Next, the HDM model's agents and the single agent are compared in terms of performance under different circumstances. In Section 5, the adaptation capacity of the HDM model is monitored in the presence of noise. Interestingly, an enhancement in the performance is also observed. In Section 6, the conclusions are given and future studies are mentioned.

2. HDM model and agent development process

The architecture of the HDM model can simply be described as in Figure 1. It is made up of k hierarchical levels. In each level, there are a number of agents that are arranged to fulfill several DM tasks. These processes help in combining information related to the lower-level agents in order to assist in the formation of higher-level agents. For example, in level 1, there are n_1 agents and they help in the composition of n_2 agents in level 2, and in level *k-1* there are n_{k-1} agents and they help in the formation of n_k agents in level k. The processes in Figure 1 are problem-specific and they might be used for many different purposes. Examples of these processes can be decision fusion, agent development, information fusion, communication, or a mixture of all of them. In this architecture, if the processes binding lower levels to higher levels are well-organized, the DM system has the potential to advance and form more successful agent structures. The HDM model can be applied to any DM problem when the input-target data set for the problem is available. Hence, some experimental or real-life input-target data are enough for its applicability.

Due to its flexible structure, the HDM model with different settings can easily be applied to a wide range of DM problems. In this study, we prefer to apply a simpler version of the model in Figure 1 to the case study. In the actual model, multiple numbers of agents are allowed in each level. For providing simplicity and comparison purposes, we prefer using a single agent in each level. The customized version of the HDM model employed for the case study is shown in Figure 2. Moreover, we also suggest a general agent development process framework that can be applied for all types of DM problems easily with the help of the HDM model. In this agent development process framework, we also embed an intelligent mechanism that is integrated with a decision fusion method such that the next level's agents are composed of an enhanced version of the components of the previous level's agents. The agent development process framework is described in Figures 3 and 4.

In Figure 3, the development process of a first-level agent is explained: the first-level agent is made up of only a dynamic component. The dynamic component refers to the portion of the agent being updated or developed through optimization and the output of optimization is the first-level agent ($\text{Agent}_{1,1}$). In Figure 4, the development process of a second-level agent is explained: 2 components contribute to the development of the second-level agent: the first component is the dynamic component as in the first level and it is typically a new arbitrary decision maker. This component is updated in the optimization stage. The second component remains unchanged in the optimization stage and hence it is called the static component; briefly it is the decisions of the first-level agent and the reliability values of these decisions computed due to a subjective performance

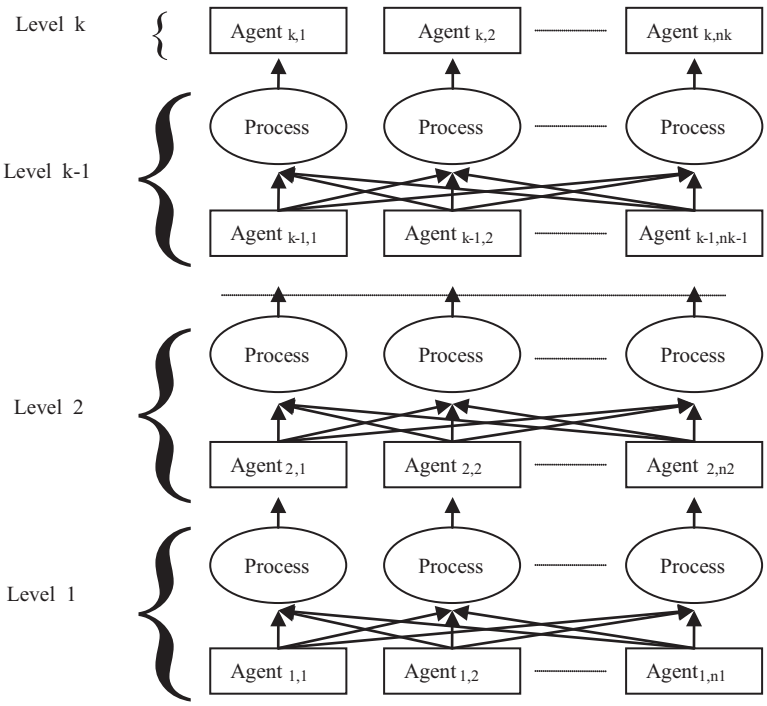

Figure 1. Structure of the proposed multilevel DM system.

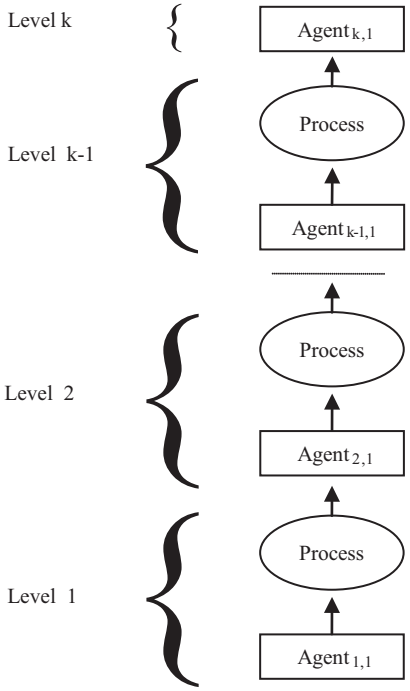

Figure 2. Structure of the applied HDM model to the dealt case study.

criterion. A suitable fusion technique is employed to fuse the decisions of the dynamic and the static components. Updating the dynamic component concurrent with the success of the fused decisions through an optimization of the cost function generates the second-level agent (Agent $_{2,1}$). The relationship between any 2 consecutive levels can be described in a similar fashion to the one observed between the first and second levels, as in Figure 4. The development process is carried on until the overall performance of the last agent goes above a desired value.

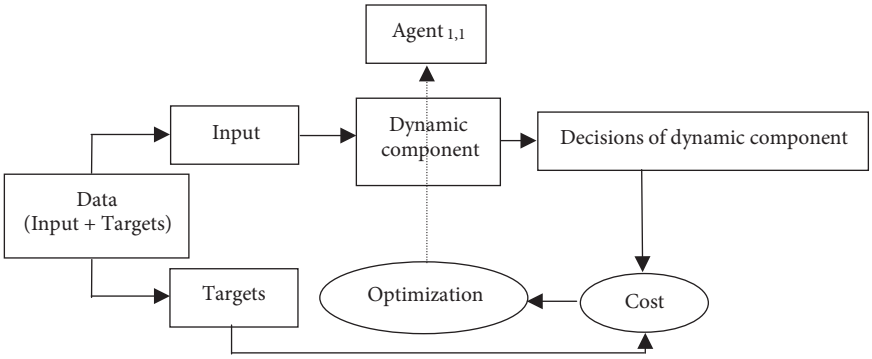

Figure 3. Development process of a first-level agent.

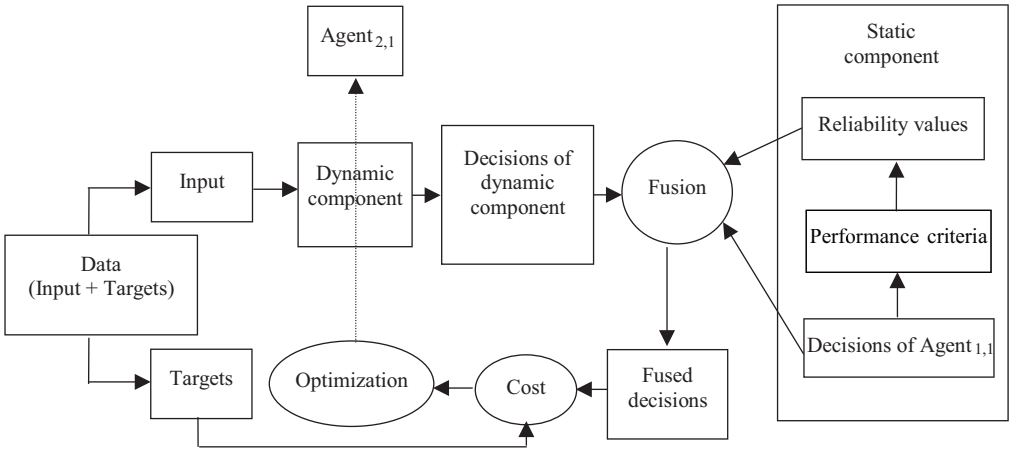

Figure 4. Development process of a second-level agent.

The framework is a good example of the adaptation capacity and flexibility provided by the model, as the modules of the framework can be adjusted easily due to requirements. These agent development frameworks are extremely important, especially as they allow for modification in the agent structure, even if the optimization stage is completely terminated. In most of the DM systems, once the agents are developed, maintaining some procedural changes is nearly impossible. However, by changing the performance criteria and decision fusion technique, these agents can be made more adaptable, especially for dynamic environments, even if there is no further optimization stage. A typical example of a dynamic environment is the presence of noise and test scenarios. Hence, our claim is that the HDM model with the proposed agent development process will increase the adaptability of the model and increase the performance for dynamic environments.

3. Case study

Details of the case study of 'Determining the fault degree in a specific tank in a 4-tank water circulation system' and the applied alternative DM model (local DM in multiple levels) to solve the same case study can be found in [14,19]. This system is made up of 4 water tanks and a reservoir, as shown in Figure 5 [14,19]. The reservoir feeds the tanks via 2 water pumps. The first pump feeds Tanks 1 and 4 and the second pump feeds Tanks 2 and 3. The tanks are placed such that water leaking from the holes of Tanks 3 and 4 flows to Tanks 1 and 2, respectively, and water leaking from the holes of Tanks 1 and 2 flows to the reservoir. These holes have some nominal values and their sizes are changed artificially; this is how faults are generated in the tanks to increase or decrease the amount of water flow from them. These fault conditions are called scenarios. In each scenario, a corresponding fault configuration is simulated for a predetermined duration using the mathematical model of the dynamic system. The states of the system are taken as the water levels in each tank. The states of the system due to different fault configurations and the actual states of the system when the system is error-free are compared with each other. After some postprocessing, the input data (called normalized error data in [14]) are obtained. Each input data element has 2 components. These are the input and the target. The input has 4 variables and these variables represent the normalized water height difference in each tank at a specified time instant. The target is the actual fault degree in a selected specific tank for the corresponding time instant. The HDM model's agents have the mission to execute a decision for each input. In other words, each agent predicts the normalized fault degrees in the selected tank at each instant. A portion of the input-target pairs (hence, a set of scenarios) is used in the training stage to develop the HDM model's agents through agent development processes and the rest is used in the validation stage to examine the efficiency of the agents. The details of construction of scenarios and composition of the input data used in the training and validation stages are mentioned in [14].

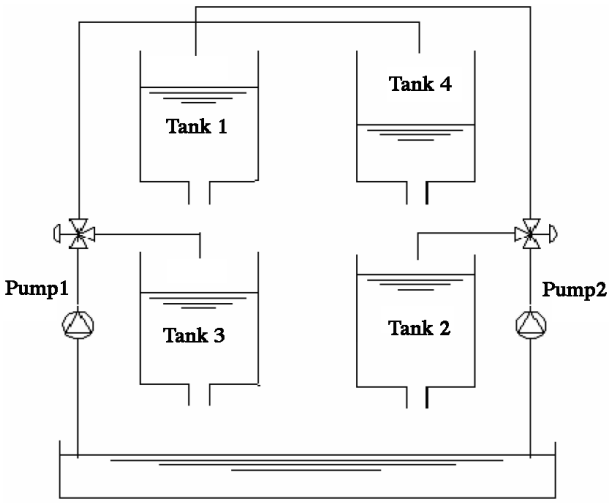

Figure 5. Water circulation system.

The chosen case study has a complex nature since a fault generated in a tank may affect the amount of flow in the other tanks as well. Furthermore, the processes where the input data are obtained are governed by nonlinear differential equations [14]. Therefore, accomplishing a sound DM is a rather difficult task; hence, this case study is very suitable to test the effectiveness of the HDM model and the agent development process. In order to successfully carry out the agent development processes, a suitable decision fusion method and an appropriate performance criterion should be selected.

4. Applications

In this section, 2 different applications are carried out. The first application is the development of the HDM model's agents and the second application is the development of a standard but very powerful single agent for the DM task [19]. In order to obtain the agents in the first application, the architecture in Figure 2 is used for 6 levels and the agent development processes in Figures 3 and 4 are utilized as the processes of the HDM model. In order to obtain the single-agent structure in the second application, the agent development process in Figure 3 is utilized. At the end, these applications are compared in terms of the 'computational effort spent in the training phase' and 'training performance'. The single agent in the second application seems better equipped compared to the HDM model's agents; it has a more sophisticated structure and extended computational effort is spent for its development. However, the simulation results prove the opposite: the HDM model's agents outperform the single agent. The HDM model's agents are later modified and used for obtaining better performance in the presence of noise in Section 5. For this purpose, without changing the dynamic components of the agents through the new optimization stage, only the performance criteria are modified. Finally, the agents adjusted due to the modification are applied in the presence of noise.

Before the applications, we want to identify how the agent development process modules shown in Figures 3 and 4 are carried out for the case study. These modules are explained in Sections 4.1, 4.2, 4.3, and 4.4, respectively.

4.1. Dynamic components and decisions of the dynamic components

In the agent development processes of both applications, the dynamic components of the agents are chosen as rule-bases. They contains rules in the form of '**IF .. THEN ..**' statements. The structure of an arbitrary rule is as below:

IF ($e_{1,t}$='att$_1$' AND $e_{2,t}$='att$_2$' AND $e_{3,t}$='att$_3$' AND $e_{4,t}$= 'att$_4$') THEN (p_t= 'att$_5$').

Each rule contains 5 variables. Among them, $e_{1,t}$, $e_{2,t}$, $e_{3,t}$, and $e_{4,t}$ are the premise variables. They represent the input data elements at the instant t. The remaining variable p_t is the consequent variable and it corresponds to the conclusion of the rule at the same instant. Eight attributes can be assigned to the premise variables (negative-big, negative-medium, negative-small, zero, positive-small, positive-medium, positive-big, and not important), whereas p_t can be assigned 7 attributes (not important is excluded). In Figure 6 [19], the membership functions of the attributes are shown.

When an input data element is fed into a rule-base, a 'min-max' type of fuzzy logic [21] rule interpolation [22] is followed by the defuzzification step executed by employing center of area defuzzification to yield an output. This output is the decision of the dynamic component for the corresponding input data element. For the particular case study we are dealing with, the input data element indicates the normalized water height differences in each tank and the output indicates the predicted fault amount (change in the hole size) in normalized units computed by the dynamic component for the tank where the fault degree classification is simulated. A typical decision is a real number between –1 and 1. A positive decision value indicates that the hole size is expected to increase, whereas a negative decision value indicates that the hole size is expected to decrease, compared to the nominal hole size value observed at the initial instant. In Table 1, some typical decisions and their corresponding explanations in terms of fault degrees are given.

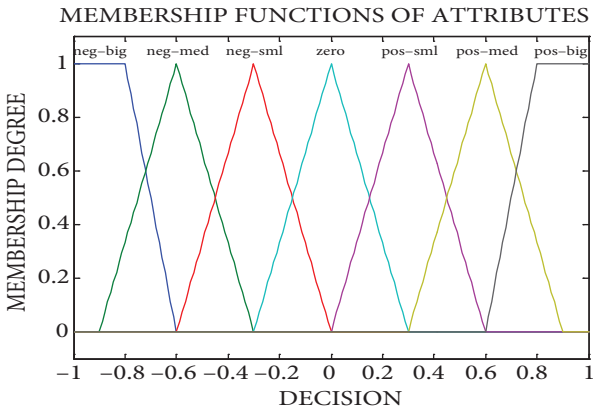

Figure 6. Membership functions of the attributes: negative-big (neg-big), negative-medium (neg-med), negative-small (neg-sml), zero (zero), positive-small (pos-sml), positive-medium (pos-med), and positive-big (pos-big).

Table 1. Explanation of decisions.

Decision	Explanation: fault degree (amount of variation of the hole size)
0	No change
–0.37	37% decrease in the hole size
0.78	78% increase in the hole size
1	The hole size is doubled (100% increase)
–1	The hole is totally closed (100% decrease)

4.2. Optimization and cost

The training of the agents is accomplished by updating the attributes assigned to the variables of each rule in the rule-base in a fashion to minimize the cost function. The cost function checks the consistency of all of the decisions altogether. In order to minimize the cost function, a GA is employed as the optimization algorithm [23]. A chromosome in the GA represents a potential solution of the optimization problem. For the case study, a chromosome represents a potential agent structure with both dynamic and static components. However, only the dynamic component is encoded on the chromosome, as there is no need to encode the static component. The optimization stage aims to update the attributes assigned to the variables of the dynamic component; however, the cost function is chosen based on the unification of the decisions of both the static and dynamic components. For this reason, the static component helps the development of the dynamic component, although it is unchanged in the optimization stage. The cost of a chromosome in level l is calculated by:

$$Cost_{ch,l} = \sum_{k=1}^{N_s} \sum_{i=1}^{N_d} |p_{l,k,i} - t_{k,i}|, \tag{1}$$

where N_s is the number of scenarios, N_d is the number of input data elements in each scenario, $p_{l,k,i}$ is the decision computed in level l for the input data element i in scenario k, $t_{k,i}$ is the target for the input data element i in scenario k, and $Cost_{ch,l}$ is the cost of the chromosome in level l. It should be noted that $p_{l,k,i}$ is the unified decision. The fitness of a chromosome in level l is then:

$$\text{Fitness}_{ch,l} = \frac{1}{Cost_{ch,l}}, \tag{2}$$

The best chromosome obtained at the end of the GA search in each level is affirmed as the agent of that level.

4.3. Static component: performance criteria and reliability values

In order to obtain the reliability values, the previous level's decisions ranging between 1 and –1 are divided into several subgroups. These subgroups are called local decision regions (LDRs). For each LDR a success rate (SR) is computed. The SR is a subjective measure of the performance of the lower-level decisions for the corresponding LDR. The SR values help in the construction of the performance criterion. The performance criterion is used to determine the reliability values. Computation of the SRs for each LDR can be summarized in the following steps:

Step 1. Define the LDRs (D_j).

Step 2. For each decision $p_{l,k,i}$, determine its LDR and find the number of decisions $n_{j,l}$ corresponding to each D_j.

Step 3. Assign a symbolic point m_j for each D_j.

Table 2 shows the LDRs and their parameters. The symbolic points of the LDRs and their sizes are selected to be coherent with the membership function distribution shown in Figure 6.

Table 2. LDRs and their parameters. Here, $p_{l,k,i}$ refers to the decision computed in level l for the input data element i in scenario k.

Decision	LDRs	Number of decisions inside each LDR	Symbolic point m_j
$-1 \leq p_{l,k,i} < -0.833$	D_1	$n_{1,l}$	–1
$-0.833 \leq p_{l,k,i} < -0.5$	D_2	$n_{2,l}$	–0.66
$-0.5 \leq p_{l,k,i} < -0.166$	D_3	$n_{3,l}$	–0.33
$-0.166 \leq p_{l,k,i} < 0.166$	D_4	$n_{4,l}$	0
$0.166 \leq p_{l,k,i} < 0.5$	D_5	$n_{5,l}$	0.33
$0.5 \leq p_{l,k,i} < 0.833$	D_6	$n_{6,l}$	0.66
$0.833 \leq p_{l,k,i} \leq 1$	D_7	$n_{7,l}$	1

Step 4. Assign a SR for each LDR.

$$s_{j,l} = 1 - \frac{\sum\limits_{p_{l,k,i} \in D_j} |p_{l,k,i} - t_{k,i}|}{n_{j,l}} \tag{3}$$

Here, $s_{j,l}$ is the SR value in level l calculated for D_j, $p_{l,k,i} \in D_j$ is any decision in the region D_j, $t_{k,i}$ is the target of the input data element i of scenario k, and $n_{j,l}$ is the number of decisions computed in D_j.

We prefer constructing a mapping from the SR values to the reliability values. This mapping is called the performance criterion and its graphical representation is called the performance plot. This mapping can be summarized in the following items:

- The reliability value of any decision at the symbolic point is equal to the SR computed for the corresponding LDR. Hence, $p_{l,k,i} = m_j \Rightarrow r(p_{l,k,i}) = s_{j,l}$, where $r(p_{l,k,i})$ is the reliability value of decision $p_{l,k,i}$.
- Define safety region $[-c_j + m_j, c_j + m_j]$ around the symbolic point m_j for each LDR. Select c_j such that the safety region does not penetrate neighboring LDRs. The reliability value of any decision in the safety regions is equal to the corresponding computed SR value for the LDR. Hence, $-c_j + m_j \leq p_{l,k,i} \leq c_j + m_j \Rightarrow r(p_{l,k,i}) = s_{j,l}$.

- Outside of the safety regions, a line interpolation is used to connect the SRs of any 2 neighboring LDRs. Hence, if a decision is not in the safety region, its reliability value is between the SR obtained for its corresponding LDR and the SR obtained for the neighboring LDR.

The performance plot obtained for the first level's decisions is shown in Figure 7 [19]. Several other LDR determination methods, interpolation techniques, merging strategies, and SR computation procedures can also be applied for the construction of performance plots. However, it is observed throughout the simulations that the way it is obtained here is an efficient one.

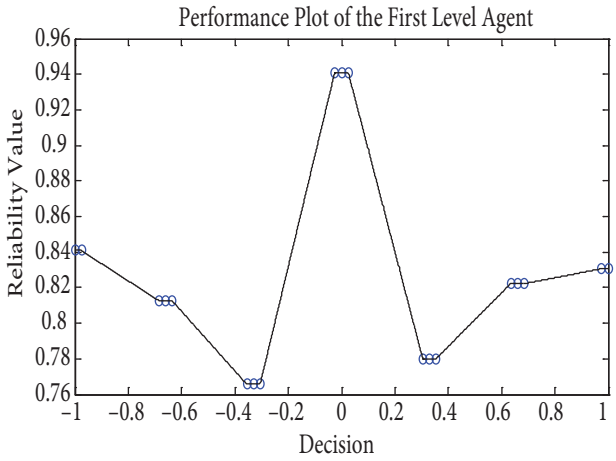

Figure 7. The performance plot for the decisions at the first level.

4.4. Fusion

The fusion of the decisions coming from different components (static and dynamic) is accomplished via a convex averaging equation. The convexity is provided with the help of the reliability values of the previous level's decisions. The fusion equation is:

$$p_{l,k,i} = (p_{l-1,k,i} \times r(p_{l-1,k,i})) + (p_{l,d} \times (1 - r(p_{l-1,k,i}))), \tag{4}$$

where $p_{l,k,i}$ is the decision for the scenario k and input data element i in the level l, $p_{l-1,k,i}$ is the decision for the scenario k and input data element i for the level $l-1$, $r(p_{l-1,k,i})$ is the reliability value of $p_{l-1,k,i}$ and $p_{l,d}$ is the decision of the dynamic component in level l. From Eq. (4), it is clear that, at the LDRs where the decisions have poor performance (i.e. the reliability value is comparably insignificant), the dynamic component of the agent has more autonomy and it can adjust $p_{l,k,i}$ more efficiently due to its weight coming from the multiplicative factor $1 - r(p_{l-1,k,i})$.

One of the examples of the structure of the fusion equation for multiagent applications can be taken as:

$$p_{l,k,i} = (\frac{1}{num} \sum_{s=1}^{num} p_{s,l-1,k,i} \times r(p_{s,l-1,k,i})) + (p_{l,d} \times (1 - \frac{1}{num} \sum_{s=1}^{num} r(p_{s,l-1,k,i}))), \tag{5}$$

where *num* represents the number of agents in level *l-1* that enroll in the fusion equation, $p_{l,k,i}$ is the decision for the scenario k and input data element i in the level l, $p_{s,l-1,k,i}$ is the decision of agent s for the scenario k and input data element i in level $l-1$, $r(p_{s,l-1,k,i})$ is the reliability value of $p_{s,l-1,k,i}$, and $p_{l,d}$ is the decision of the dynamic component in level l.

4.5. Application 1: HDM model

The agent in the first level is simply a developed rule-base by the agent development process in Figure 3. The rule-base encloses 30 rules. The number of rules is determined based on our observations over the simulation results and knowledge about the distribution of the input data used in training and validation stages. The number of rules is sufficiently high with regards to the performance and sufficiently small with regards to the computational complexity. The parameters and results of the GA simulations in the first level are as follows:

- Number of input data elements in the training stage: 34,170 (170 scenarios and 201 input data elements in each scenario).
- Number of chromosomes: 40.
- Number of generations: 2000.
- The number of genes in a chromosome: 150.
- Crossover style: One point crossover within each consecutive 25 genes.
- Crossover ratio: 90%.
- Reproduction: 10% (with elitist method).
- Gene mutation probability and style: Only half of the new chromosomes obtained as the result of the crossover operation are mutated. The mutation rate is chosen as 1% and mutation is first applied after the tenth generation. However, if the fitness of the best chromosome obtained remains the same for 8 consecutive generations, the mutation rate is increased to 5% for the next generation, and the search continues with this updated mutation rate for a single generation. The mutation rate subsequently returns to its nominal value (1%).

In the higher levels, the GA parameters are taken as the same as the first level parameters. However, the number of rules in the rule-bases is reduced to 20 rules; hence, a chromosome contains 100 variables this time. Taking a lower number of rules will decrease the computational effort in the optimization stage; moreover, it is consistent with the idea of obtaining gradually growing agent structures in each level. The HDM model is applied up to 6 levels. The cost values of the developed agents in the training scenarios are shown in Table 3 [19].

Table 3. Cost values of the agents for the training scenarios.

Level	1	2	3	4	5	6
Cost	3974	3787	3559	3372	3309	3257

It is clear that the HDM model is successful in the training stage since the costs are decreasing in consecutive levels. In Figure 8 [19], the performance plots of the 1st, 2nd, 3rd, 4th, 5th, and 6th level agents are shown together.

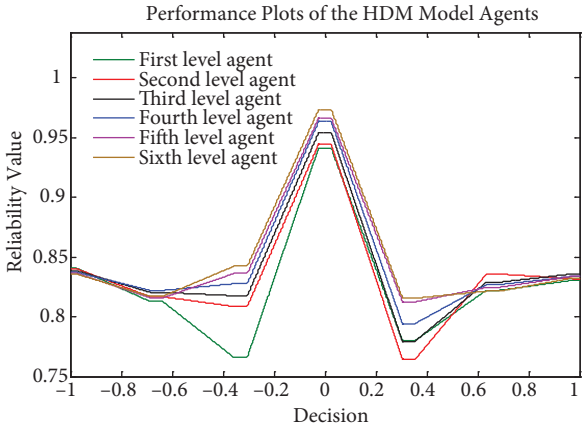

Figure 8. Performance plots of the HDM model's agents.

The agents are also tested in 2 different sets of validation scenarios. The costs of the agents are calculated similarly using Eq. (1) for the validation scenarios and the results are tabulated in Tables 4 and 5 [19].

Table 4. Cost values of the agents for the first set of validation scenarios.

Level	1	2	3	4	5	6
Cost	4135	4049	3814	3648	3609	3532

Table 5. Cost values of the agents for the second set of validation scenarios.

Level	1	2	3	4	5	6
Cost	3770	3589	3359	3200	3143	3091

4.6. Application 2: single agent

In the second application, only a single agent is developed. The agent is a rule-base with 70 rules. Due to having an excessive number of rules, the agent seems better equipped compared to agents in the first application. Once more, the GA is used in the training stage. However, different from the first application, the optimization stage is carried out for 6000 generations. The GA parameters and the result are as follows [19]:

- Number of genes for each chromosome: 350.
- Number of chromosomes: 40.
- The crossover and mutation rates: chosen as in the first application.
- Cost of the best chromosome obtained at the training scenarios: 3800.

The single agent has a more sophisticated structure and is trained for longer generations; however, it is outperformed by the HDM model's agents (see Table 3). Even in the second level, the cost of the developed agent with the use of the HDM model is better than the cost of the single agent. A 1.6 GHz laptop with 496 MB of RAM is used to simulate both of the applications in a MATLAB software environment. The computation time required to evaluate the fitness of each chromosome in a single generation in the optimization stage of the second application is on average, 5 to 6 times longer than the computation time required in the evaluation of the fitness of each chromosome in a single generation in the first application. Hence, the excessive number

of rules turns out to be a burden for the performance and computation time. Instead of developing a very complicated single-agent structure at one shot by optimizing too many variables for a specific task, it is more efficient to implement a level-by-level development process: simpler agent structures are developed first in the primary levels and then these primary agents are improved by insertion and optimization of new and simple components.

5. Effect of noise

In this section, the influence of noise over the performance of the HDM model is investigated. Noise with 2 different distributions (uniform and Gaussian noise) and with 3 different signal-to-noise ratio (SNR) settings (high, medium, and low) is applied to the training scenarios and the first set of validation scenarios. Details about the noise can be found in [14,19]. The cost values of the HDM model's agents are recalculated using Eq. (1) in the presence of noise and the results are tabulated in Tables 6–9 [19].

Table 6. Cost values of the HDM model's agents for the training scenarios when noise with uniform distribution is applied.

Cost	SNR (high)	SNR (medium)	SNR (low)
First-level agent	4254	5392	6738
Second-level agent	4315	5640	7115
Third-level agent	4295	5772	7289
Fourth-level agent	4437	6079	7623
Fifth-level agent	4629	6310	7810
Sixth-level agent	4728	6395	7850

Table 7. Cost values of the HDM model's agents for the first set of validation scenarios when noise with uniform distribution is applied.

Cost	SNR (high)	SNR (medium)	SNR (low)
First-level agent	4416	5552	6883
Second-level agent	4586	5902	7339
Third-level agent	4557	6040	7510
Fourth-level agent	4717	6364	7857
Fifth-level agent	4920	6605	8048
Sixth-level agent	5026	6700	8093

Table 8. Cost values of the HDM model's agents for the training scenarios when noise with Gaussian distribution is applied.

Cost	SNR (high)	SNR (medium)	SNR (low)
First-level agent	4238	5224	6565
Second-level agent	4307	5453	6916
Third-level agent	4287	5576	7099
Fourth-level agent	4436	5869	7439
Fifth-level agent	4634	6113	7651
Sixth-level agent	4744	6231	7731

The HDM model with its current settings seems unsuccessful in the presence of noise because the cost values are increasing in successive levels. If the agent development processes in Figures 3 and 4 are employed in

the HDM model, the input data should be fed to the model once at each level. As the input data are perturbed by noise, the decisions in each successive level will also be influenced. Due to the fusion style, the accumulation of error in the levels may be the reason why the cost increases. There are several ways to overcome this problem. An idea that can be applied for this purpose is to modify the performance criterion, as it is the main tool for information transfer between levels. On the other hand, the results are still promising. The cost values in Tables 4 and 5 decrease in successive levels in the validation scenarios when the system is noise-free, which indicates that the HDM model has identified the dynamics related to the case study correctly.

Table 9. Cost values of the HDM model's agents for the first set of validation scenarios when noise with Gaussian distribution is applied.

Cost	SNR (high)	SNR (medium)	SNR (low)
First-level agent	4397	5386	6712
Second-level agent	4566	5699	6712
Third-level agent	4541	5821	6712
Fourth-level agent	4707	6131	6712
Fifth-level agent	4919	6375	7875
Sixth-level agent	5031	6496	7954

5.1. Removing the effects of noise with the adaptation capacity of the HDM model

Several approaches can be attempted to remove the undesirable effects of noise and improve the performance. Adjusting the modules of the agent development processes or parameters of the HDM model, applying noise removal tools like filtering, or the use of different agent structures are some suitable approaches. The Kalman filter [24] can be one of the choices for the removal of noise. However, the Kalman filter or extended Kalman filter for nonlinear cases [25] requires an a priori assumption about the noise that it has Gaussian distribution. The input data for the DM problem are obtained as the result of some intermediate processing stages [14]. For this reason its structure is not suitable for Kalman filtering. Hence, the key issue to overcome this problem is to take advantage of the adaptation capacity of the HDM model due the flexibility provided by the agent development processes. It is clear that the performance plots are drawn only accounting for the input data elements used in the training stage. As they are constructed once, they are fixed and for this reason it is unfair to update them anew by reconsidering noisy data components. However, we can modify the performance plots without any further reconstruction process based on our knowledge about the data. As we check the distribution of input data in the training and validation stages, we realize that the target value for a dense group of them is 0 and a fine decision for any of them should remain in the LDR D_4. Moreover, the SR evaluated for D_4 is higher than the SRs evaluated for all of the other LDRs ($s_{4,l} \geq s_{i,l},\ i = 1, ..., 7$) in all of the levels, which indicates that the abundance of the decisions at D_4 is very precise. Thus, adding noise will probably have a more negative impact for the input data elements whose decisions lie on D_4 compared to other input data elements. Based on this observation, without any further analysis of the input data elements, we examine the consequence of assigning a better SR $s_{4,l}$ for D_4 in each level in the presence of noise. For this reason, $s_{4,l}$ is taken to be 1 for each level and performance plots are reconstructed under this new setting. Thus, if $p_{l-1,k,i}$ is in the safety region of D_4 (that requires $r(p_{l-1,k,i}) = 1$ due to new performance plots) then no matter what the next level's agent is, we will observe $p_{l,k,i} = p_{l-1,k,i}$ due to Eq. (4). Hence, in the presence of noise, the decisions residing on the safety region of D_4 are unchanged in higher levels once they are determined in the first level. Reevaluating the cost values tabulated in Tables 6–9 by taking $s_{4,l} = 1$, we obtain the new cost values for the agents, which are shown in Tables 10–13, respectively.

Table 10. Cost values of the HDM model's agents for the training scenarios when noise with uniform distribution is applied and the SR for LDR D_4 is taken as 1 in each level.

Cost	SNR (high)	SNR (medium)	SNR (low)
First-level agent	4254	5392	6738
Second-level agent	4057	5471	7012
Third-level agent	3866	5511	7139
Fourth-level agent	3737	5699	7411
Fifth-level agent	3706	5878	7582
Sixth-level agent	3654	5938	7599

Table 11. Cost values of the HDM model's agents for the first set of validation scenarios when noise with uniform distribution is applied and the SR for D_4 is taken as 1 in each level.

Cost	SNR (high)	SNR (medium)	SNR (low)
First-level agent	4416	5552	6883
Second-level agent	4313	5718	7229
Third-level agent	4119	5768	7354
Fourth-level agent	4011	5977	7641
Fifth-level agent	3998	6171	7817
Sixth-level agent	3961	6245	7845

Table 12. Cost values of the HDM model's agents for the training scenarios when noise with Gaussian distribution is applied and the SR for LDR D_4 is taken as 1 in each level.

Cost	SNR (high)	SNR (medium)	SNR (low)
First-level agent	4238	5224	6565
Second-level agent	4043	5267	6793
Third-level agent	3852	5286	6918
Fourth-level agent	3729	5435	7179
Fifth-level agent	3705	5608	7363
Sixth-level agent	3662	5682	7415

Table 13. Cost values of the HDM model's agents for the first set of validation scenarios when noise with Gaussian distribution is applied and the SR for LDR D_4 is taken as 1 in each level.

Cost	SNR (high)	SNR (medium)	SNR (low)
First-level agent	4397	5386	6712
Second-level agent	4291	5503	7012
Third-level agent	4117	5524	7137
Fourth-level agent	3997	5693	7407
Fifth-level agent	3993	5871	7592
Sixth-level agent	3958	5957	7651

As seen from Tables 10–13, if the SNR is high, new performance criteria help in decreasing the cost values in successive levels. As we decrease the SNR, the cost values slightly increase in successive levels. However, the rate of increase is not as significant as those observed in Tables 6–9. Hence, the modification of the performance criteria is a supportive tool for providing robustness in the presence of noise. In Figure 9, the modified version of the performance plots of each agent of the HDM model when $s_{4,l} = 1$ is shown.

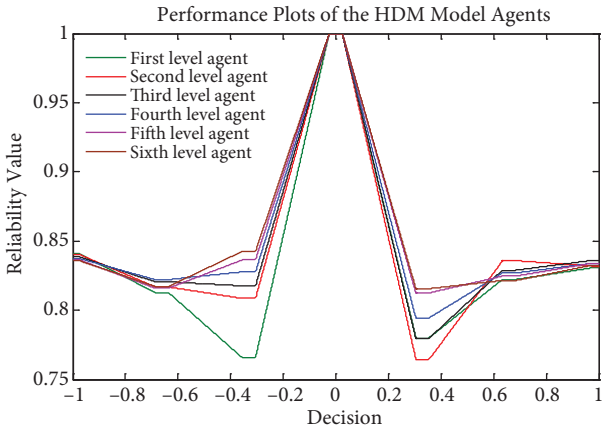

Figure 9. Performance plots of the modified HDM model's agents when the SR for the LDR D_4 is taken as 1.

It is also possible to increase the performance for low or medium SNR levels by further modifications in the performance plots. This time we prefer redefining the safety region for the LDR D_4 while keeping $s_{4,l} = 1$ and assume that the safety region for D_4 is augmented such that it includes the whole LDR D_4. In this case, if we reevaluate the cost values for the agents, we will obtain the results shown in Tables 14–17. The modified versions of the performance plots of each agent of the HDM model with these further modifications are shown in Figure 10.

Table 14. Cost values of the HDM model's agents for the training scenarios when noise with uniform distribution is applied and the SR for LDR D_4 is taken as 1 and the safety region for D_4 includes the whole LDR in the performance plots in each level.

Cost	SNR (medium)	SNR (low)
First-level agent	5392	6738
Second-level agent	5276	6836
Third-level agent	5231	6898
Fourth-level agent	5147	7042
Fifth-level agent	5128	7130
Sixth-level agent	5099	7134

Table 15. Cost values of the HDM model's agents for the first set of validation scenarios when noise with uniform distribution is applied and the SR for LDR D_4 is taken as 1 and the safety region for D_4 includes the whole LDR in the performance plots in each level.

Cost	SNR (medium)	SNR (low)
First-level agent	5552	6883
Second-level agent	5502	7042
Third-level agent	5432	7105
Fourth-level agent	5391	7261
Fifth-level agent	5389	7359
Sixth-level agent	5377	7375

From Tables 14–17, it is clear that the undesirable effect of noise is extremely removed and a continuous tendency of enhancement in the cost values, except for a few level transitions, is observed when the SNR is

medium. Moreover, even though the undesirable effects of noise are not totally removed for the cases with low SNR levels, the increase in the cost values is decelerated.

Table 16. Cost values of the HDM models agents for the training scenarios when noise with Gaussian distribution is applied and the SR for LDR D_4 is taken as 1 and the safety region for D_4 includes the whole LDR in the performance plots in each level.

Cost	SNR (medium)	SNR (low)
First-level agent	5224	6565
Second-level agent	5116	6627
Third-level agent	5054	6676
Fourth-level agent	5013	6791
Fifth-level agent	5008	6875
Sixth-level agent	4992	6895

Table 17. Cost values of the HDM models agents for the first set of validation scenarios when noise with Gaussian distribution is applied and the SR for LDR D_4 is taken as 1 and the safety region for D_4 includes the whole LDR in the performance plots in each level.

Cost	SNR (medium)	SNR (low)
First-level agent	5386	6712
Second-level agent	5339	6832
Third-level agent	5281	6884
Fourth-level agent	5259	7011
Fifth-level agent	5266	7100
Sixth-level agent	5265	7133

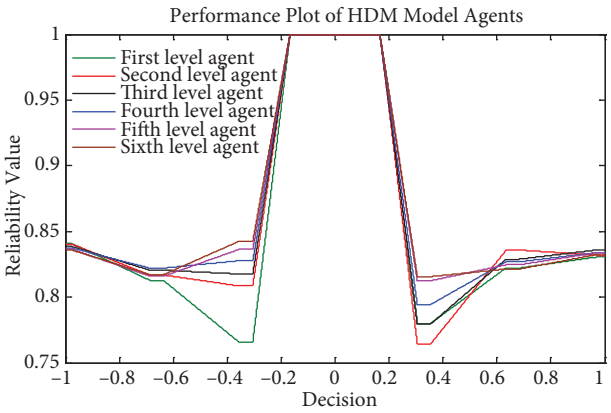

Figure 10. Performance plots of the modified HDM model's agents when the SR for the LDR D_4 is taken as 1 and the safety region of the decision region D_4 includes the whole decision region.

6. Conclusions

The HDM model has satisfactory results for the training and the 2 sets of validation scenarios. Cost values improve at all of the hierarchical levels and the HDM model also outperforms the single-level agent. However, in the presence of noise, the unmodified HDM model's agents are unable to improve the cost values as the hierarchical levels increase. The flexibility provided by the agent development processes in this model is

employed to deal with this problem. Without new optimization stages, only the static component in the agent development process is modified to make the information transfer between hierarchical levels more robust to noise. The modification of the performance criterion depends on experimentation and a priori information about the input data elements. It is observed that too many input data elements are concentrated in a specific region. Based on this information, the performance criterion is adapted and enhanced cost values are obtained for high and medium SNR levels without the application of special techniques (i.e. filtering). For a low SNR, the results are still not at desired levels, but they are promising, as the increase in cost values are not as significant as they are observed when the original unmodified performance criterion is employed.

In the future, we will focus on improving the performance of the HDM model, especially in the presence of noise, by new strategies. For this purpose, new agent structures that can contribute to further improvement can be used. The decision may be a combination of different preferences due to different agents, which might boost the robustness. However, there is always an extra computational effort necessary in order to develop new agents for overcoming the noise problem.

DM problems where more than one criterion has to be considered together are called multicriteria DM problems. Multicriteria DM methods focus on achieving a balance point that satisfies each criterion at a desired or at least sufficient level by relaxing, or reducing the importance, or assigning a consensus degree of satisfaction [26]. Experts are extensively employed for multicriteria DM problems in order to quantize different alternatives and to combine the alternatives via weights to assess the relative importance of criteria [27]. The HDM model is also very suitable for multicriteria DM problems.

In this study, the agent structure utilizes a single input data element to perform its decision for an instant. However, the agent structure can be modified in such a way that it can use a sequence of data to perform the decision at a particular instant. These agents can be more effective in noisy environments. However, as agents turn out to be more complex, the computation time required to develop such agents is expected to be larger.

References

[1] Espinilla M, Liu J, Martinez L. An extended hierarchical linguistic model for decision-making problems. Comput Intell 2011; 27: 489–512.

[2] Kilincci O, Onal SA. Fuzzy AHP approach for supplier selection in a washing machine company. Expert Syst Appl 2011; 38: 9656–9664.

[3] Marimin M, Umano M, Hatono I, Tamura H. Hierarchical semi-numeric method for pairwise fuzzy group decision making. IEEE T Syst Man Cy B 2002; 32: 691–700.

[4] Pasi G, Yager RR. Modeling the concept of majority opinion in group decision making. Inform Sciences 2006; 176: 390–414.

[5] Ben-Arieh D, Easton T. Multi-criteria group consensus under linear cost opinion elasticity. Decis Support Syst 2007; 43: 713–721.

[6] Xu Z, Cai X. Group consensus algorithms based on preference relations. Inform Sciences 2011; 180: 150–162.

[7] Kacprzyk J, Zadrozny S. Towards a general and unified characterization of individual and collective choice functions under fuzzy and nonfuzzy preferences and majority via the ordered weighted average operators. Int J Intell Syst 2009; 24: 4–26.

[8] Wang YM, Parkan C. Two new approaches for assessing the weights of fuzzy opinions in group decision analysis. Inform Sciences 2006; 176: 3538–3555.

[9] Domshlak C, Hullermeier E, Kaci S, Prade H. Preferences in AI: an overview. Artif Intell 2011; 175: 1037–1052.

[10] Ralescu AL, Ralescu DA, Yamakata Y. Inference by aggregation of evidence with applications to fuzzy probabilities. Inform Sciences 2007; 177: 378–387.

[11] Pavlin G, Oude P, Maris M, Nunnink J, Hood T. A multi-agent systems approach to distributed Bayesian information fusion. Inform Fusion 2010; 11: 267–282.

[12] Sycara K, Glinton R, Yu B, Giampapa J, Owens S, Lewis M, Grindle LTC. An integrated approach to high-level information fusion. Inform Fusion 2009; 10: 25–50.

[13] Tan TZ, Ng GS, Quek C. A Novel biologically and psychologically inspired fuzzy decision support system: hierarchical complementary learning. IEEE-ACM T Comput Bi 2008; 5: 67–79.

[14] Beldek U, Leblebicioğlu K. Local decision making and decision fusion in hierarchical levels. TOP 2009; 17: 44–69.

[15] Lu J, Shi C, Zhang G. On bilevel multi-follower decision making: general framework and solutions. Inform Sciences 2006; 176: 1607–1627.

[16] Zeng XJ, Keane JA. Approximation capabilities of hierarchical fuzzy systems. IEEE T Fuzzy Syst 2005; 13: 659–672.

[17] Zhu Q, Aldridge SL, Resha TN. Hierarchical collective agent network (HCAN) for efficient fusion and management of multiple networked sensors. Inform Fusion 2007; 8: 266–280.

[18] Mendis BSU, Gedeon TD. Complex structured decision making model: a hierarchical frame work for complex structured data. Inform Sciences 2012; 194: 85–106.

[19] Beldek U. Design and improvement of multi-level decision-making models, PhD, Middle East Technical University, Ankara, Turkey, 2009.

[20] Asadzadeh L, Zamanifar K. An agent-based parallel approach for the job shop scheduling problem with genetic algorithms. Math Comput Model 2010; 52: 1957–1965.

[21] Zadeh LA. Is there a need for fuzzy logic? Inform Sciences 2008; 178: 2751–2779.

[22] Chen SM, Chang YC. Multi-variable fuzzy forecasting based on fuzzy clustering and fuzzy rule interpolation techniques. Inform Sciences 2010; 180: 4772–4783.

[23] Simon D, Rarick R, Ergezer M, Du D. Analytical and numerical comparisons of biogeography-based optimization and genetic algorithms. Inform Sciences 2011; 181: 1224–1248.

[24] Grewal MS, Andrews AP. Kalman Filtering. New York, NY, USA: John Wiley & Sons, 2001.

[25] Mei W, Shan G, Wang C. Practical development of the second-order extended Kalman filter for very long range radar tracking. Signal Process 2011; 91: 1240–1248.

[26] Dursun M, Karsak EE. A fuzzy MCDM approach for personnel selection. Expert Syst Appl 2010; 37: 4324–4330.

[27] Tsiporkova E, Boeva V. Multi-step ranking of alternatives in a multi-criteria and multi-expert decision making environment. Inform Sciences 2006; 176: 2673–2697.