Journal of New Results in Science (JNRS)
ISSN: 1304-7981
http://dergipark.gov.tr/en/pub/jnrs
Research Article
Open Access

Volume: 9
Issue: 1
Year: 2020
Pages: 19-36

Received: 25.03.2020 ***Accepted***: 18.04.2020

A New Approach to the Criteria-Weighted Fuzzy Soft Max-Min Decision-Making Method and Its Application to a Performance-Based Value Assignment Problem

Serdar Enginoğlu*, **Samet Memiş**

Department of Mathematics, Faculty of Arts and Sciences, Çanakkale Onsekiz Mart University, Çanakkale, Turkey
**Corresponding author, serdarenginoglu@gmail.com*

ABSTRACT: Recently, the criteria-weighted fuzzy soft max-min decision-making (WFSMmDM) method provided in [Razak, S. A., Mohamad, D., A decision making method using fuzzy soft sets, Malaysian Journal of Fundamental and Applied Sciences, 2013, 9(2), 99-104] has been configured to operate in the fuzzy parameterized fuzzy soft matrices (*fpfs*-matrices) space by Enginoğlu and Memiş [A configuration of some soft decision-making algorithms via *fpfs*-matrices, Cumhuriyet Science Journal, 2018, 39(4), 871-881] faithfully to the original. Even though this configured method, which is denoted by RM13 and constructed by and-product/or-product (RM13a/RM13o), is useful in soft decision-making, it is of great importance to improve the method in terms of running time and complexity when processing a large number of data. In this study, to improve WFSMmDM, we propose two algorithms, denoted by EM20a and EM20o. Furthermore, we prove that EM20a is equivalent to RM13a. Thereafter, we compare the running time of these algorithms. The results show that EM20a and EM20o outperform RM13a and RM13o, respectively, in any number of data. We then apply EM20o to the problem of performance-based value assignment concerning seven filters used in image denoising. Besides, we compare the proposed two methods' performance ranking with that of eight state-of-art soft decision-making methods. Finally, we provide the conclusive remarks and some suggestions for further research.

Keywords **–** *Fuzzy sets, Soft sets, Soft decision-making, Soft matrices, fpfs-matrices*

1. Introduction

Molodtsov (1999) propounded the concept of soft sets to deal with uncertainties and up to now many researchers have conducted various applied and theoretical studies thereon (Maji et al., 2001, 2003; Çağman and Enginoğlu, 2010a; Çağman et al., 2010; Çağman et al., 2011a,b; Deli and Çağman, 2015; Enginoğlu et al., 2015; Zorlutuna and Atmaca, 2016; Riaz and Hashmi, 2017, 2018; Riaz et al., 2018; Şenel, 2018; Ullah et al., 2018; Sezgin et al., 2019). To able to avail of the ability of this concept in computer mathematics (or sciences), Çağman and Enginoğlu presented the soft matrices (Çağman and Enginoğlu, 2010b) and fuzzy soft matrices (Çağman and Enginoğlu, 2012). The authors also proposed the soft max-min decision-making (SMmDM) method and the fuzzy soft max-min decision-making (FSMmDM) method using and-product of soft matrices and fuzzy soft matrices, respectively (Çağman and Enginoğlu, 2010b; Çağman and Enginoğlu, 2012). Although these methods are useful in decision-making, they cannot easily model a problem with a parameter containing uncertainty. Afterwards, Razak and Mohamad (2011, 2013) have presented Criteria-Weighted SMmDM (WSMmDM) and Criteria-Weighted FSMmDM (WFSMmDM) methods for soft matrices and fuzzy soft matrices, respectively. Although the methods therein have taken the parameters' weights into account, they still suffer from two drawbacks, i.e. running time and complexity.

Latterly, Enginoğlu and Memiş (2018a) have configured eighteen soft decision-making methods to operate in the fuzzy parameterized fuzzy soft matrices (*fpfs*-matrices) space (Enginoğlu, 2012; Enginoğlu and Çağman, n.d.) faithfully to the original. Since the configurations have been made as faithfully to the originals, the drawbacks mentioned above have also been transferred. Further, the authors have highlighted the significance of studies on the simplification and different configurations of these methods therein. Therefore, several soft decision-making algorithms in (Enginoğlu and Memiş, 2018a) have been simplified and applied to a given decision-making problem (Enginoğlu and Memiş, 2018b,c; Enginoğlu et al., 2018a,b, 2019b,c,d).

In this paper, we focus on improving two new methods free of the disadvantages mentioned above. In Section 2, we present the concept of *fpfs*-matrices (Enginoğlu, 2012; Enginoğlu and Çağman, n.d.) and RM13 constructed by and-product/or-product (RM13a/RM13o) (Razak and Mohamad, 2011, 2013, Enginoğlu and Memiş, 2018a). In Section 3, we propound two new methods, namely EM20a and EM20o, and prove that EM20a is equivalent to RM13a. In Section 4, we compare the running time of these algorithms. In Section 5, we apply EM20o to a decision-making problem in which the noise removal/image denoising filters can be ordered performance-wise. We then compare the ranking orders produced by the proposed methods with the ranking orders produced by eight state-of-art soft decision-making methods. Finally, we discuss the need for further research.

2. Preliminaries

In this section, firstly, the definitions of *fpfs*-sets (Çağman et al., 2010; Enginoğlu, 2012) and *fpfs*-matrices (Enginoğlu, 2012; Enginoğlu and Çağman, n.d.) are presented. Throughout this paper, let E be a parameter set, $F(E)$ be the set of all fuzzy sets over E, and $\mu \in F(E)$. Here, a fuzzy set is denoted by $\left\{{}^{\mu(x)}x : x \in E\right\}$.

Definition 1. (Çağman et al., 2010; Enginoğlu, 2012) *Let U be a universal set, $\mu \in F(E)$, and α be a function from μ to $F(U)$. Then, the set $\{({}^{\mu(x)}x, \alpha({}^{\mu(x)}x)) : x \in E\}$ being the graphic of α is called a fuzzy parameterized fuzzy soft set (fpfs-set) parameterized via E over U (or briefly over U).*

Example 1. *Let $E = \{x_1, x_2, x_3, x_4\}$ and $U = \{u_1, u_2, u_3, u_4, u_5\}$. Then,*

$$\alpha = \left\{\left({}^{1}x_1, \left\{{}^{0.5}u_2, {}^{0.8}u_4\right\}\right), \left({}^{0.7}x_2, \left\{{}^{0.2}u_1, {}^{1}u_3, {}^{0.8}u_5\right\}\right), \left({}^{0.5}x_3, \left\{{}^{0.8}u_1, {}^{0.4}u_3, {}^{0.7}u_4\right\}\right), \left({}^{0}x_4, \left\{{}^{0.9}u_2, {}^{0.6}u_5\right\}\right)\right\}$$

is an fpfs-set over U.

In the present paper, the set of all *fpfs*-sets over U is denoted by $FPFS_E(U)$.

Definition 2. (Enginoğlu, 2012; Enginoğlu and Çağman, n.d.) *Let $\alpha \in FPFS_E(U)$. Then, $[a_{ij}]$ is called the matrix representation of α (or briefly fpfs-matrix of α) and is defined by*

$$[a_{ij}] := \begin{bmatrix} a_{01} & a_{02} & a_{03} & \cdots & a_{0n} & \cdots \\ a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots \end{bmatrix}$$

such that for $i \in \{0,1,2,\cdots\}$ and $j \in \{1,2,\cdots\}$,

$$a_{ij} := \begin{cases} \mu(x_j), & i = 0 \\ \alpha({}^{\mu(x_j)}x_j)(u_i), & i \neq 0 \end{cases}$$

Here, if $|U| = m - 1$ and $|E| = n$, then $[a_{ij}]$ has order $m \times n$.

Example 2. *The fpfs-matrix of α provided in Example 1 is as follows:*

$$[a_{ij}] = \begin{bmatrix} 1 & 0.7 & 0.5 & 0 \\ 0 & 0.2 & 0.8 & 0 \\ 0.5 & 0 & 0 & 0.9 \\ 0 & 1 & 0.4 & 0 \\ 0.8 & 0 & 0.7 & 0 \\ 0 & 0.8 & 0 & 0.6 \end{bmatrix}$$

From now on, the set of all *fpfs*-matrices parameterized via E over U is denoted by $FPFS_E[U]$.

Definition 3. (Enginoğlu and Çağman, n.d.) *Let* $[a_{ij}]_{m\times n_1} \in FPFS_{E_1}[U]$, $[b_{ik}]_{m\times n_2} \in FPFS_{E_2}[U]$, *and* $[c_{ip}]_{m\times n_1 n_2} \in FPFS_{E_1\times E_2}[U]$ *such that* $p = n_2(j-1)+k$. *For all* i *and* p,
if $c_{ip} \coloneqq min\{a_{ij}, b_{ik}\}$, *then* $[c_{ip}]$ *is called and-product of* $[a_{ij}]$ *and* $[b_{ik}]$ *and is denoted by* $[a_{ij}] \wedge [b_{ik}]$,
if $c_{ip} \coloneqq max\{a_{ij}, b_{ik}\}$, *then* $[c_{ip}]$ *is called or-product of* $[a_{ij}]$ *and* $[b_{ik}]$ *and is denoted by* $[a_{ij}] \vee [b_{ik}]$,
if $c_{ip} \coloneqq min\{a_{ij}, 1-b_{ik}\}$, *then* $[c_{ip}]$ *is called andnot-product of* $[a_{ij}]$ *and* $[b_{ik}]$ *and is denoted by* $[a_{ij}] \overline{\wedge} [b_{ik}]$,
if $c_{ip} \coloneqq max\{a_{ij}, 1-b_{ik}\}$, *then* $[c_{ip}]$ *is called ornot-product of* $[a_{ij}]$ *and* $[b_{ik}]$ *and is denoted by* $[a_{ij}] \underline{\vee} [b_{ik}]$.

Secondly, we present the algorithm RM13a (RM13o) (Enginoğlu and Memiş, 2018a).

RM13a (RM13o) Algorithm Steps

Construct three *fpfs*-matrices $[a_{ij}]$, $[b_{ik}]$, and $[c_{it}]$ such that $\sum_j a_{0j} = \sum_k b_{0j} = \sum_t c_{0t} = 1$

Obtain $[A_{ij}]$, $[B_{ik}]$, and $[C_{it}]$ defined by $A_{ij} \coloneqq a_{0j}a_{ij}$, $B_{ik} \coloneqq b_{0k}b_{ik}$, and $C_{it} \coloneqq c_{0t}c_{it}$ such that $i \in \{1,2,\dots,m-1\}$ and $j,k,t \in \{1,2,\dots,n\}$

Find and-product (or-product) *fpfs*-matrix $[d_{ip}]$ of $[A_{ij}]$ and $[B_{ik}]$

Obtain $[x_{ik}]$ defined by

$$x_{ik} \coloneqq \begin{cases} \min\limits_{p\in I_k}\{d_{ip}\}, & I_k \neq \emptyset \\ 0, & I_k = \emptyset \end{cases}$$

such that $i \in \{1,2,\dots,m-1\}$, $k \in \{1,2,\dots,n\}$, and $I_k \coloneqq \{p \mid \exists i, d_{ip} \neq 0 \ \wedge \ (k-1)n < p \leq kn\}$

Find and-product (or-product) *fpfs*-matrix $[e_{ir}]$ of $[x_{ik}]$ and $[C_{it}]$

Obtain $[y_{it}]$ defined by

$$y_{it} \coloneqq \begin{cases} \min\limits_{r\in I_t}\{e_{ir}\}, & I_t \neq \emptyset \\ 0, & I_t = \emptyset \end{cases}$$

such that $i \in \{1,2,\dots,m-1\}$, $t \in \{1,2,\dots,n\}$, and $I_t \coloneqq \{r \mid \exists i, e_{ir} \neq 0 \ \wedge \ (t-1)n < r \leq tn\}$

Obtain $[s_{i1}]$ defined by $s_{i1} \coloneqq \max\limits_t\{y_{it}\}$ such that $i \in \{1,2,\dots,m-1\}$ and $t \in \{1,2,\dots,n\}$

Obtain the decision set $\{^{s_{i1}}u_i | u_i \in U\}$

3. Soft Decision-Making Methods: EM20a and EM20o

Under this title, we first propose an algorithm denoted by EM20a.

EM20a Algorithm Steps

Construct three *fpfs*-matrices $[a_{ij}]$, $[b_{ik}]$, and $[c_{it}]$

Obtain score matrix $[s_{i1}]$ defined by

$$s_{i1} \coloneqq \begin{cases} \min\left\{\min\left\{\max_{j\in I_a}\{a_{0j}a_{ij}\}, \min_{k\in I_b}\{b_{0k}b_{ik}\}\right\}, \min_{t\in I_c}\{c_{0t}c_{it}\}\right\}, & I_a, I_b, I_c \neq \emptyset \\ 0, & \text{otherwise} \end{cases}$$

such that $i \in \{1,2,\dots,m-1\}$, $I_a \coloneqq \{ j \mid \exists i, a_{0j}a_{ij} \neq 0\}$, $I_b \coloneqq \{ k \mid \exists i, b_{0k}b_{ik} \neq 0\}$, and $I_c \coloneqq \{ t \mid \exists i, c_{0t}c_{it} \neq 0\}$

Obtain the decision set $\{^{s_{i1}}u_i | u_i \in U\}$

It is clear that the values s_{i1} give a ranking order over u_i. Therefore, the decision-maker can opt for the proper ones of the alternatives.

Theorem 1. EM20a *is equivalent to* RM13a.
PROOF**.** Let us consider the score matrices $[\tilde{s}_{i1}]$ and $[s_{i1}]$ provided in RM13a and EM20a, respectively. We first prove that $\exists t, I_t \neq \emptyset \Leftrightarrow (I_a \neq \emptyset \wedge I_b \neq \emptyset \wedge I_c \neq \emptyset)$. Then,

$$\begin{aligned} \exists t, I_t \neq \emptyset &\Leftrightarrow \exists i, e_{ir} \neq 0 \\ &\Leftrightarrow \exists i, \min\{x_{ik}, C_{it}\} \neq 0 \\ &\Leftrightarrow \exists i, (x_{ik} \neq 0 \wedge C_{it} \neq 0) \\ &\Leftrightarrow \exists i, \left(\min_{p\in I_k}\{d_{ip}\} \neq 0 \wedge C_{it} \neq 0\right) \\ &\Leftrightarrow \exists i, \left(\min_{p\in I_k}\left\{\min\{A_{ij}, B_{ik}\}\right\} \neq 0 \wedge C_{it} \neq 0\right) \\ &\Leftrightarrow \exists i, \left(A_{ij} \neq 0 \wedge B_{ik} \neq 0 \wedge C_{it} \neq 0\right) \\ &\Leftrightarrow \exists i, \left(a_{0j}a_{ij} \neq 0 \wedge b_{0k}b_{ik} \neq 0 \wedge c_{0t}c_{it} \neq 0\right) \\ &\Leftrightarrow I_a \neq \emptyset \wedge I_b \neq \emptyset \wedge I_c \neq \emptyset \end{aligned}$$

Therefore, $\forall t, I_t = \emptyset \Leftrightarrow (I_a = \emptyset \vee I_b = \emptyset \vee I_c = \emptyset)$. Here, $i \in \{1,2,\dots,m-1\}$, $j,k,t \in \{1,2,\dots,n\}$, $p = n(j-1)+k$, and $r = n(k-1)+t$. Moreover, it can be seen that $I_t \neq \emptyset \Rightarrow I_k \neq \emptyset$.

Suppose that $I_t = \{r_1^t, r_2^t, \dots, r_{w(t)}^t\}$, $k \in \{x_1, x_2, \dots, x_u\}$, and $I_c = \{c_1, c_2, \dots, c_v\}$. Then,

$$
\begin{aligned}
\tilde{s}_{i1} = \max_t\{y_{it}\} &= \max_t \begin{cases} \min\limits_{r\in I_t}\{e_{ir}\}, & I_t \neq \emptyset \\ 0, & I_t = \emptyset \end{cases} \\
&= \max\left\{ \begin{cases} \min\limits_{r\in I_1}\{e_{ir}\}, & I_1 \neq \emptyset \\ 0, & I_1 = \emptyset \end{cases}, \begin{cases} \min\limits_{r\in I_2}\{e_{ir}\}, & I_2 \neq \emptyset \\ 0, & I_2 = \emptyset \end{cases}, \ldots, \begin{cases} \min\limits_{r\in I_n}\{e_{ir}\}, & I_n \neq \emptyset \\ 0, & I_n = \emptyset \end{cases} \right\} \\
&= \max\left\{ \begin{matrix} \begin{cases} \min\left\{e_{ir_1^1}, e_{ir_2^1}, \ldots, e_{ir_{w(1)}^1}\right\}, & I_1 \neq \emptyset \\ 0, & I_1 = \emptyset \end{cases}, \\ \begin{cases} \min\left\{e_{ir_1^2}, e_{ir_2^2}, \ldots, e_{ir_{w(2)}^2}\right\}, & I_2 \neq \emptyset \\ 0, & I_2 = \emptyset \end{cases}, \ldots, \\ \begin{cases} \min\left\{e_{ir_1^n}, e_{ir_2^n}, \ldots, e_{ir_{w(n)}^n}\right\}, & I_n \neq \emptyset \\ 0, & I_n = \emptyset \end{cases} \end{matrix} \right\} \\
&= \max\left\{ \begin{matrix} \begin{cases} \min\left\{\min\{x_{ix_1}, C_{ic_1}\}, \min\{x_{ix_1}, C_{ic_2}\}, \ldots, \min\{x_{ix_1}, C_{ic_v}\}\right\}, & I_1 \neq \emptyset \\ 0, & I_1 = \emptyset \end{cases}, \\ \begin{cases} \min\left\{\min\{x_{ix_2}, C_{ic_1}\}, \min\{x_{ix_2}, C_{ic_2}\}, \ldots, \min\{x_{ix_2}, C_{ic_v}\}\right\}, & I_2 \neq \emptyset \\ 0, & I_2 = \emptyset \end{cases}, \ldots, \\ \begin{cases} \min\left\{\min\{x_{ix_u}, C_{ic_1}\}, \min\{x_{ix_u}, C_{ic_2}\}, \ldots, \min\{x_{ix_u}, C_{ic_v}\}\right\}, & I_n \neq \emptyset \\ 0, & I_n = \emptyset \end{cases} \end{matrix} \right\} \\
&= \max\left\{ \begin{matrix} \begin{cases} \min\left\{x_{ix_1}, \min\{C_{ic_1}, C_{ic_2}, \ldots, C_{ic_v}\}\right\}, & I_1 \neq \emptyset \\ 0, & I_1 = \emptyset \end{cases}, \\ \begin{cases} \min\left\{x_{ix_2}, \min\{C_{ic_1}, C_{ic_2}, \ldots, C_{ic_v}\}\right\}, & I_2 \neq \emptyset \\ 0, & I_2 = \emptyset \end{cases}, \ldots, \\ \begin{cases} \min\left\{x_{ix_u}, \min\{C_{ic_1}, C_{ic_2}, \ldots, C_{ic_v}\}\right\}, & I_n \neq \emptyset \\ 0, & I_n = \emptyset \end{cases} \end{matrix} \right\} \\
&= \min\begin{cases} \left\{\max\limits_k\{x_{ik}\}, \min\limits_{t\in I_c}\{C_{it}\}\right\}, & \exists t, I_t \neq \emptyset \\ 0, & \forall t, I_t = \emptyset \end{cases} \\
&= \min\begin{cases} \left\{\max\limits_k\left\{\min\limits_{p\in I_k}\{d_{ip}\}\right\}, \min\limits_{t\in I_c}\{C_{it}\}\right\}, & I_a, I_b, I_c \neq \emptyset \\ 0, & otherwise \end{cases} \\
\begin{pmatrix}\text{from} \\ \text{Theorem 4.1} \\ \text{in [27]}\end{pmatrix} \quad &= \min\begin{cases} \left\{\min\left\{\max\limits_{j\in I_a}\{A_{ij}\}, \min\limits_{k\in I_b}\{B_{ik}\}\right\}, \min\limits_{t\in I_c}\{C_{it}\}\right\}, & I_a, I_b, I_c \neq \emptyset \\ 0, & otherwise \end{cases} \\
&= \min\begin{cases} \left\{\min\left\{\max\limits_{j\in I_a}\{a_{0j}a_{ij}\}, \min\limits_{k\in I_b}\{b_{0k}b_{ik}\}\right\}, \min\limits_{t\in I_c}\{c_{0t}c_{it}\}\right\}, & I_a, I_b, I_c \neq \emptyset \\ 0, & otherwise \end{cases} \\
&= s_{i1}
\end{aligned}
$$

Hence, the functions s_{i1} provided in RM13a and EM20a are equal in any case. QED

Secondly, we propose another algorithm denoted by EM20o.

EM20o Algorithm Steps

Construct three *fpfs*-matrices $[a_{ij}]$, $[b_{ik}]$, and $[c_{it}]$

Obtain score matrix $[s_{i1}]$ defined by

$$s_{i1} := \begin{cases} \max\left\{\max\left\{\max_{j\in I_a}\{a_{0j}a_{ij}\}, \min_{k\in I_b}\{b_{0k}b_{ik}\}\right\}, \min_{t\in I_c}\{c_{0t}c_{it}\}\right\}, & I_a, I_b, I_c \neq \emptyset \\ 0, & \text{otherwise} \end{cases}$$

such that $i \in \{1,2,\dots,m-1\}$, $I_a := \{\, j \mid \exists i, a_{0j}a_{ij} \neq 0\}$, $I_b := \{\, k \mid \exists i, b_{0k}b_{ik} \neq 0\}$, and $I_c := \{\, t \mid \exists i, c_{0t}c_{it} \neq 0\}$

Obtain the decision set $\{^{s_{i1}}u_i | u_i \in U\}$

It is clear that the values s_{i1} give a ranking order over u_i. Therefore, the decision-maker can opt for the proper ones of the alternatives.

4. Simulation Results

This section first compares the running time of RM13a and EM20a by using MATLAB R2019b. In the absence of a difficulty, we use a laptop with 2.6 GHz i5 Dual-Core CPU and 4 GB RAM to compare the methods. However, in this study, we utilize a workstation with I(R) Xeon(R) CPU E5-1620 v4 @ 3.5 GHz and 64 GB RAM because the computer is incapable of running RM13a if the number of parameters exceeds 5000.

We present the running time of RM13a and EM20a in Table 1 and Figure 1 for 10 objects and 10-100 parameters. In Table 1, although the difference in running time is low, EM20a is 60 times faster than RM13a and has about 98% advantage

Table 1. The running time of the methods for 10 objects and 10-100 parameters (in second)

Parameter Count	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
RM13a	0.0254	0.0098	0.0042	0.0075	0.0164	0.0144	0.0166	0.0201	0.0154	0.0182
EM20a	0.0097	0.0030	0.0007	0.0006	0.0024	0.0022	0.0004	0.0002	0.0004	0.0003
Difference	0.0157	0.0069	0.0035	0.0069	0.0140	0.0122	0.0162	0.0199	0.0149	0.0180
Advantage (%)	61.8764	69.8312	83.8179	91.6161	85.3601	84.7762	97.4951	98.7605	97.0879	98.5553

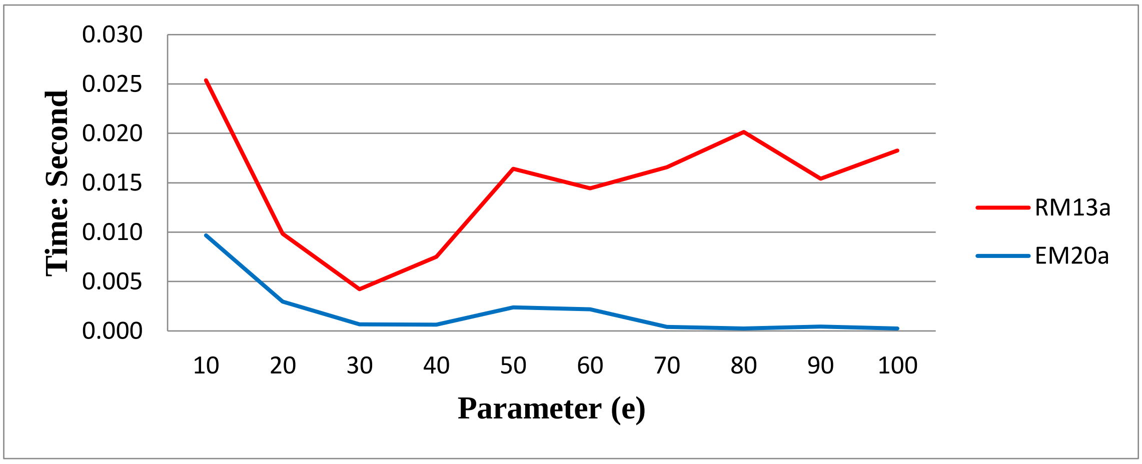

Figure 1. The figure for Table 1

We then give the running time data of RM13a and EM20a in Table 2 and Figure 2 for 10 objects and 1000-10000 parameters. Table 2 shows that EM20a has a 1046-second advantage over RM13a.

Table 2. The running time of the methods for 10 objects and 1000-10000 parameters (in second)

Parameter Count	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
RM13a	1.8102	12.4026	37.2990	80.1327	144.9890	242.7991	376.2401	546.7492	766.2646	1046.9352
EM20a	0.0108	0.0054	0.0028	0.0040	0.0065	0.0076	0.0055	0.0061	0.0077	0.0084
Difference	1.7994	12.3972	37.2962	80.1286	144.9824	242.7916	376.2345	546.7430	766.2569	1046.9268
Advantage (%)	99.4026	99.9566	99.9925	99.9950	99.9955	99.9969	99.9985	99.9989	99.9990	99.9992

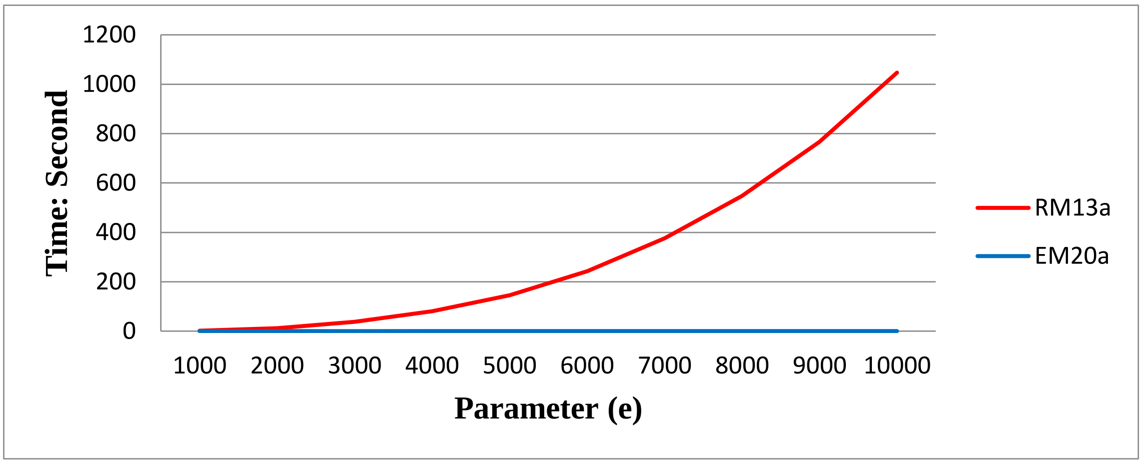

Figure 2. The figure for Table 2

In Table 3 and Figure 3, we give the running time for 10 parameters and 10-100 objects. Despite the little difference of running time between these methods, EM20a is about eight times faster than RM13a in 10 parameters and 100 objects.

Table 3. The running time of the methods for 10-100 objects and 10 parameters (in second)

Object Count	10	20	30	40	50	60	70	80	90	100
RM13a	0.0258	0.0095	0.0036	0.0042	0.0111	0.0091	0.0036	0.0040	0.0044	0.0048
EM20a	0.0111	0.0037	0.0006	0.0007	0.0035	0.0015	0.0004	0.0005	0.0005	0.0006
Difference	0.0147	0.0059	0.0030	0.0035	0.0076	0.0076	0.0031	0.0035	0.0038	0.0042
Advantage (%)	56.9685	61.5630	82.2879	82.4274	68.5547	83.2298	87.4329	88.0516	87.8472	87.8744

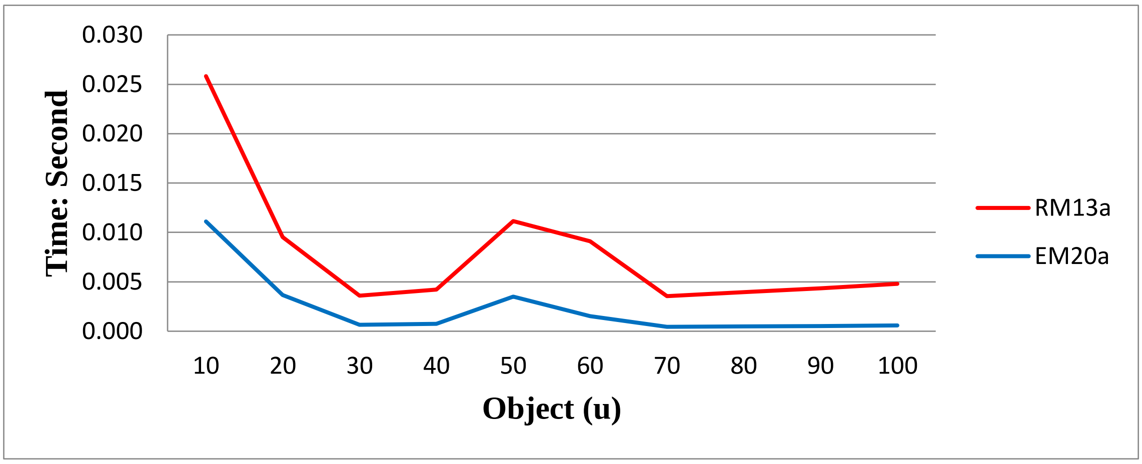

Figure 3. The figure for Table 3

We then offer the running time data in Table 4 and Figure 4 for 10 parameters and 1000-10000 objects. The results show that increasing the number of objects only does not affect the running time as severely as increasing the number of parameters does. Besides, EM20a works faster in a large number of parameters than in a large number of objects.

Table 4. The running time of the methods for 1000-10000 objects and 10 parameters (in second)

Object Count	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
RM13a	0.1438	0.2608	0.5117	0.8513	1.2834	1.8454	2.4193	3.1301	3.9135	4.8779
EM20a	0.0169	0.0150	0.0200	0.0269	0.0372	0.0462	0.0550	0.0653	0.0752	0.0889
Difference	0.1268	0.2459	0.4917	0.8244	1.2462	1.7992	2.3643	3.0648	3.8382	4.7890
Advantage (%)	88.2163	94.2671	96.0854	96.8384	97.1001	97.4985	97.7267	97.9149	98.0780	98.1778

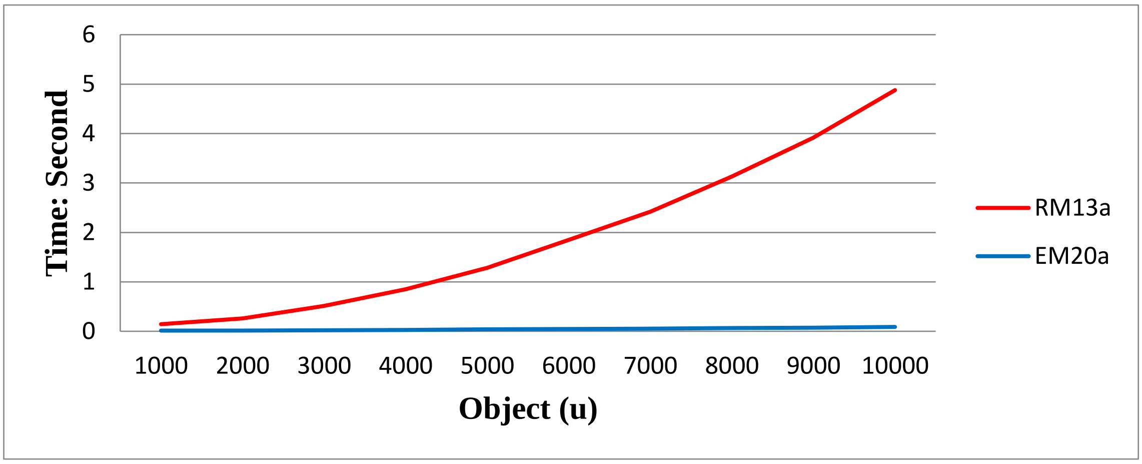

Figure 4. The figure for Table 4

In Table 5 and Figure 5, we present the running time for 10-100 parameters and 10-100 objects. Although the difference of running time between these methods is little, EM20a is up to 145 times faster than RM13a.

Table 5. The running time of the methods for 10-100 objects and 10-100 parameters (in second)

Count	10	20	30	40	50	60	70	80	90	100
RM13a	0.0254	0.0098	0.0090	0.0136	0.0431	0.0478	0.0701	0.0949	0.1196	0.1595
EM20a	0.0098	0.0026	0.0005	0.0005	0.0038	0.0018	0.0010	0.0008	0.0010	0.0011
Difference	0.0156	0.0072	0.0085	0.0131	0.0393	0.0460	0.0691	0.0941	0.1186	0.1584
Advantage (%)	61.4693	73.7303	94.8735	96.0255	91.2480	96.2688	98.5028	99.1123	99.1688	99.3137

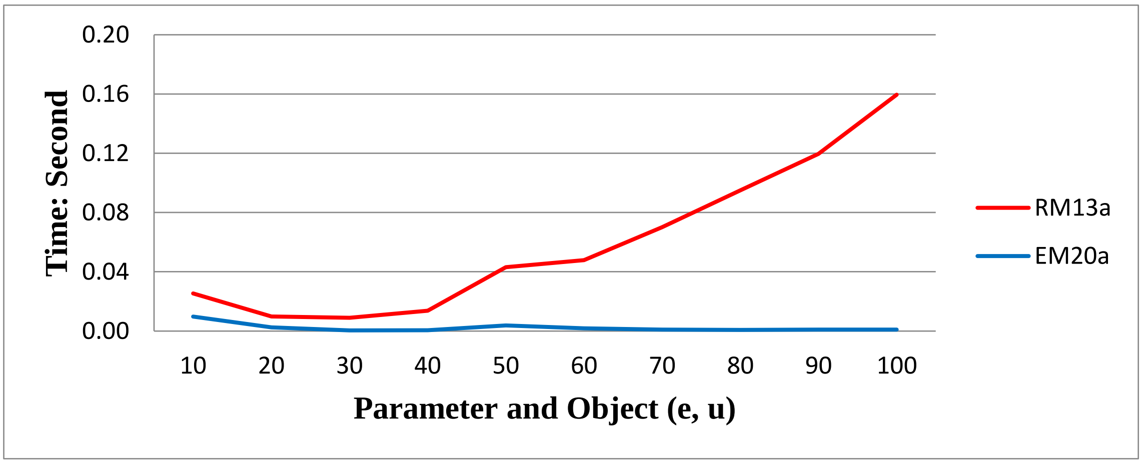

Figure 5. The figure for Table 5

The authors provide their running time in Table 6 and Figure 6 for 100-1000 parameters and 100-1000 objects. 0.0594-second and 1292-second running time data suggest that EM20a is more suitable than RM13a for any real-time software.

Table 6. The running time of the methods for 100-1000 objects and 100-1000 parameters (in second)

Count	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**
RM13a	0.2262	2.5518	11.8859	35.0138	81.3499	164.1251	297.2333	510.7264	831.2171	1292.8854
EM20a	0.0111	0.0056	0.0124	0.0094	0.0280	0.0221	0.0333	0.0467	0.0474	0.0594
Difference	0.2150	2.5462	11.8734	35.0043	81.3219	164.1031	297.2000	510.6797	831.1697	1292.8260
Advantage (%)	95.0751	99.7805	99.8954	99.9731	99.9656	99.9865	99.9888	99.9909	99.9943	99.9954

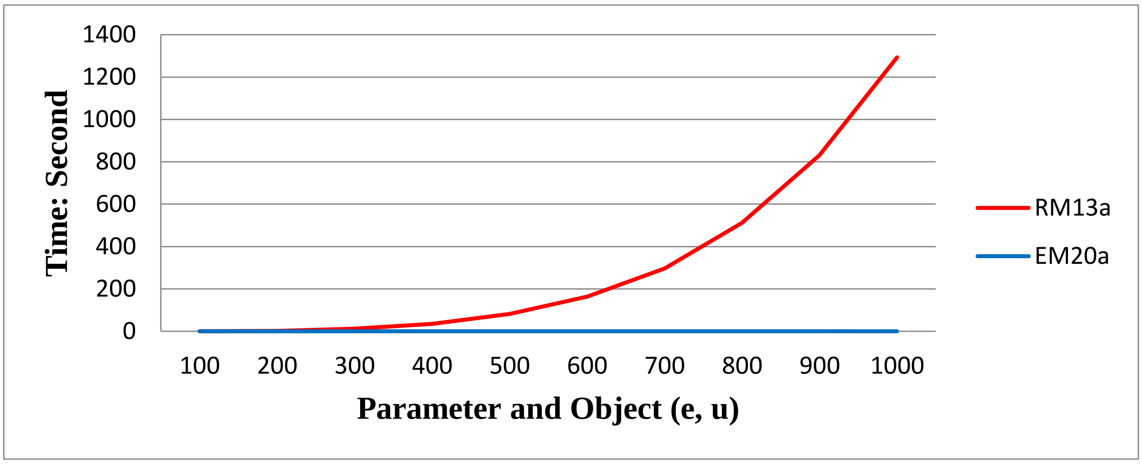

Figure 6. The figure for Table 6

All the results mentioned above indicate that EM20a outperforms RM13a in the presence of any number of data. Thus, EM20a can be more efficaciously in intelligent systems than RM13a. Similarly, EM20o performs better than RM13o in any number of data. However, there is a need for another comparison, e.g. of ranking performances, to better understand whether EM20o is fitter to be used in smart systems than RM13o is.

Secondly, we compare the running time data of RM13o and EM20o by using MATLAB R2019b and a workstation with I(R) Xeon(R) CPU E5-1620 v4 @ 3.5 GHz and 64 GB RAM because the computer is incapable of running RM13o if the parameters are more than 5000. In Table 7 and Figure 7, we present the running time data of RM13o and EM20o for 10 objects and 10-100 parameters. Even though the running time difference between these methods is little, EM20o performs about 35 times faster in 100 parameters and 10 objects than RM13o does. The contribution of the results and the importance of the results should be emphasised.

Table 7. The running time of the methods for 10 objects and 10-100 parameters (in second)

Parameter Count	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
RM13o	0.0215	0.0090	0.0056	0.0075	0.0095	0.0186	0.0167	0.0121	0.0155	0.0210
EM20o	0.0094	0.0027	0.0006	0.0006	0.0024	0.0021	0.0004	0.0003	0.0005	0.0006
Difference	0.0121	0.0063	0.0049	0.0069	0.0071	0.0165	0.0162	0.0118	0.0150	0.0205
Advantage (%)	56.4999	70.2257	88.4300	91.8359	75.0223	88.5756	97.5088	97.7622	96.9134	97.2908

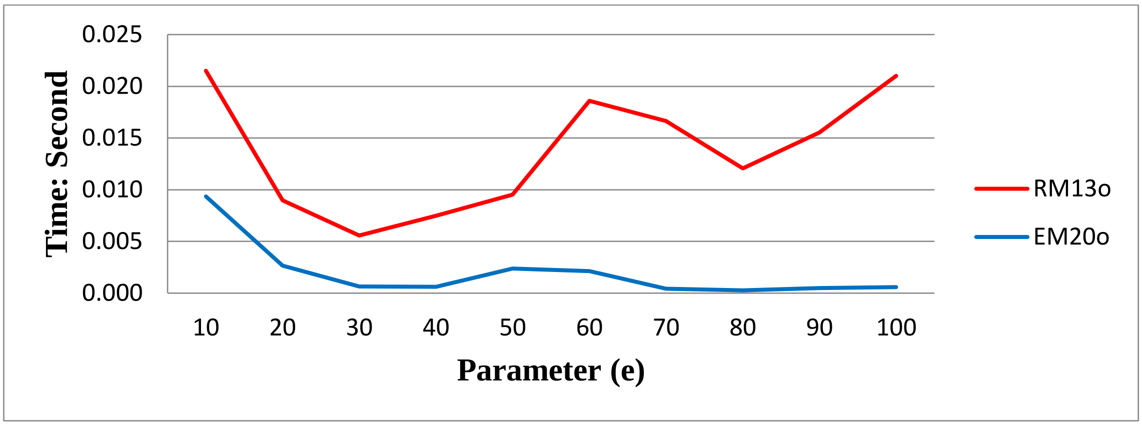

Figure 7. The figure for Table 7

In Table 8 and Figure 8, we offer the data on the running time of RM13o and EM20o for 10 objects and 1000-10000 parameters. It must be noted that the difference in running time between these methods has been remarkably increased. 1049-second running time shows that RM13o is unsuitable for any real-time software that processes a large number of data.

Table 8. The running time of the methods for 10 objects and 1000-10000 parameters (in second)

Parameter Count	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
RM13o	1.8135	12.3918	37.3141	79.0899	145.2155	242.4259	378.3586	547.9714	774.1607	1049.7234
EM20o	0.0116	0.0045	0.0028	0.0038	0.0065	0.0071	0.0056	0.0065	0.0091	0.0084
Difference	1.8019	12.3873	37.3113	79.0861	145.2090	242.4188	378.3530	547.9649	774.1516	1049.7150
Advantage (%)	99.3599	99.9635	99.9924	99.9952	99.9955	99.9971	99.9985	99.9988	99.9988	99.9992

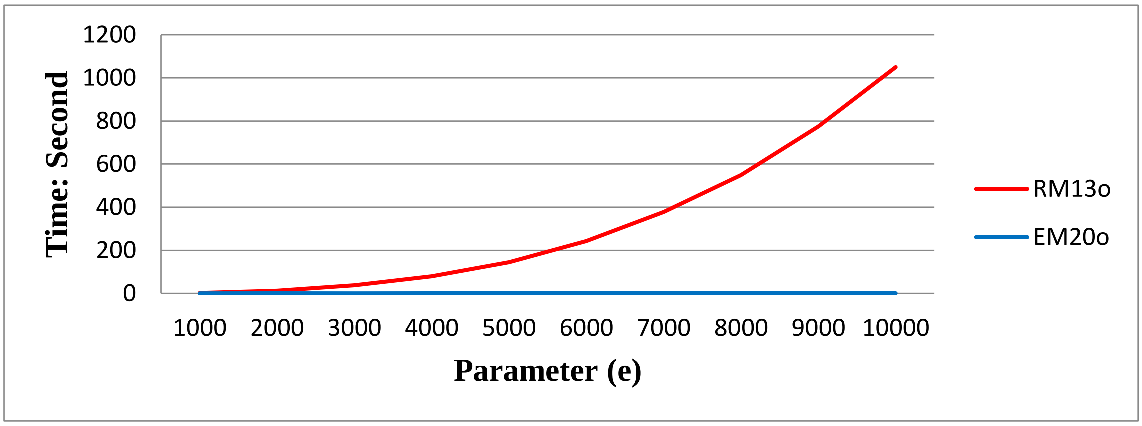

Figure 8. The figure for Table 8

Moreover, we give the running time data for 10 parameters and 10-100 objects in Table 9 and Figure 9. Despite the little difference of running time between these methods, EM20o performs nearly six times faster in the presence of 10 parameters and 100 objects than RM13o does.

Table 9. The running time of the methods for 10-100 objects and 10 parameters (in second)

Object Count	10	20	30	40	50	60	70	80	90	100
RM13o	0.0229	0.0122	0.0038	0.0042	0.0074	0.0063	0.0036	0.0040	0.0045	0.0067
EM20o	0.0100	0.0037	0.0007	0.0007	0.0027	0.0015	0.0005	0.0005	0.0006	0.0012
Difference	0.0129	0.0085	0.0031	0.0034	0.0047	0.0048	0.0031	0.0035	0.0039	0.0055
Advantage (%)	56.2014	69.8167	82.2833	82.3911	63.1816	76.1694	87.4118	87.0136	87.1234	82.3050

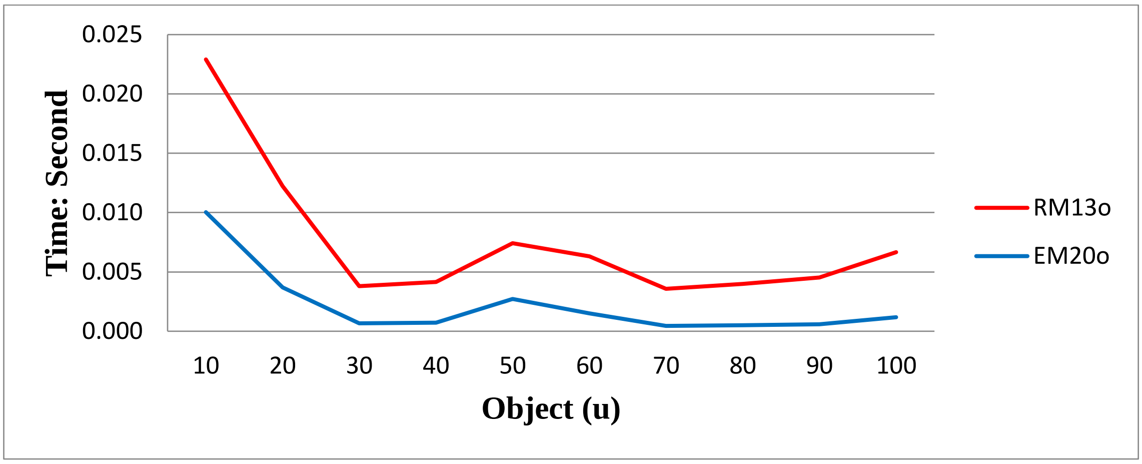

Figure 9. The figure for Table 9

In addition, we provide the running time data for 10 parameters and 1000-10000 objects in Table 10 and Figure 10. The results show that increasing the number of objects only does not affect the running time as dramatically as an increase in the number of parameters. Besides, EM20o works faster in a large number of parameters than of objects.

Table 10. The running time of the methods for 1000-10000 objects and 10 parameters (in second)

Object Count	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
RM13o	0.1081	0.2877	0.5093	0.8545	1.2902	1.8647	2.4231	3.1187	3.9172	4.8781
EM20o	0.0159	0.0154	0.0192	0.0270	0.0373	0.0466	0.0547	0.0648	0.0758	0.0882
Difference	0.0922	0.2723	0.4901	0.8276	1.2528	1.8180	2.3685	3.0539	3.8413	4.7899
Advantage (%)	85.2610	94.6338	96.2331	96.8426	97.1064	97.4999	97.7440	97.9218	98.0639	98.1925

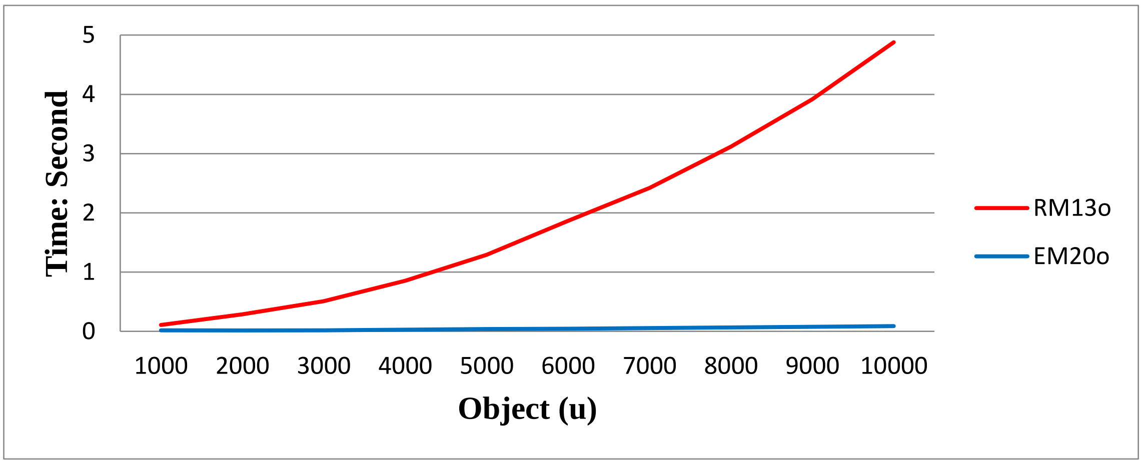

Figure 10. The figure for Table 10

Table 11 and Figure 11 offer the data on the running time for 10-100 parameters and 10-100 objects. Although the difference of running time between these methods is little, EM20o runs up to 138 times faster than RM13o does.

Table 11. The running time of the methods for 10-100 objects and 10-100 parameters (in second)

Count	10	20	30	40	50	60	70	80	90	100
RM13o	0.0227	0.0101	0.0073	0.0227	0.0360	0.0478	0.0672	0.0985	0.1332	0.1527
EM20o	0.0102	0.0028	0.0004	0.0009	0.0026	0.0019	0.0010	0.0012	0.0010	0.0011
Difference	0.0125	0.0073	0.0069	0.0217	0.0334	0.0460	0.0661	0.0972	0.1322	0.1515
Advantage (%)	55.1189	72.6515	93.8748	95.9867	92.6882	96.1294	98.4663	98.7575	99.2383	99.2587

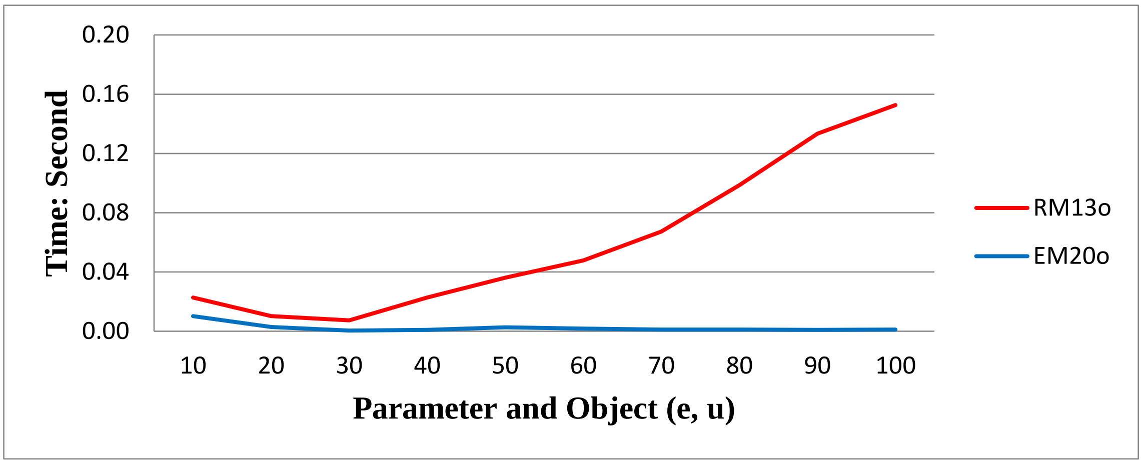

Figure 11. The figure for Table 11

In Table 12 and Figure 12, we give the running time data for 100-1000 parameters and 100-1000 objects. 0.0587-second and 1288-second running time suggests that EM20o is more suitable for any real-time software than RM13o.

Table 12. The running time of the methods for 100-1000 objects and 100-1000 parameters (in second)

Count	100	200	300	400	500	600	700	800	900	1000
RM13o	0.2253	2.5523	11.8292	35.0828	81.2237	163.0558	298.5850	512.0823	832.7337	1288.3115
EM20o	0.0110	0.0055	0.0060	0.0096	0.0231	0.0223	0.0282	0.0367	0.0470	0.0587
Difference	0.2143	2.5469	11.8232	35.0732	81.2006	163.0335	298.5568	512.0456	832.6867	1288.2528
Advantage (%)	95.0969	99.7848	99.9495	99.9727	99.9716	99.9863	99.9906	99.9928	99.9944	99.9954

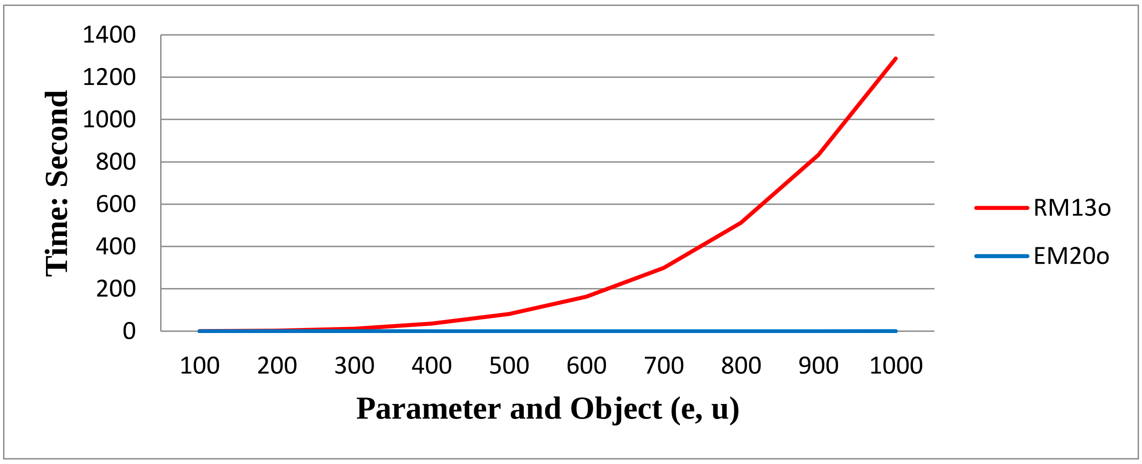

Figure 12. The figure for Table 12

The results show that EM20o outperforms RM13o in any number of data.

5. An Application of EM20o to a Performance-Based Value Assignment (PVA) Problem

In this section, we firstly apply EM20o to sort seven state-of-art filters used in image denoising in terms of noise removal performance. Even though it is more difficult to sort these filters in the event that the filters perform variously in different noise densities, EM20o ably copes with this difficulty. To illustrate, let us consider the mean-PSNR results (Table 13), the mean-SSIM results (Table 14), and the mean-VIF results (Table 15) for 20 traditional images provided in (Enginoğlu et al., 2019b).

Table 13. The mean-PSNR results for the 20 traditional images with different SPN ratios

Noise Density	10%	20%	30%	40%	50%	60%	70%	80%	90%
DBA	37.52	34.29	31.96	29.83	27.86	25.89	23.90	21.55	18.55
MDBUTMF	36.80	32.18	29.02	28.48	28.81	28.34	26.95	23.42	15.29
BPDF	36.98	33.54	31.03	28.88	26.82	24.60	21.98	17.74	10.51
NAFSMF	36.08	33.27	31.49	30.15	29.02	27.96	26.82	25.47	22.34
AWMF	36.34	35.00	33.83	32.69	31.47	30.14	28.68	26.99	24.70
DAMF	39.58	36.33	34.14	32.45	30.99	29.64	28.28	26.69	24.35
ARmF	**40.04**	**37.12**	**35.14**	**33.53**	**31.99**	**30.45**	**28.86**	**27.08**	**24.74**

Table 14. The mean-SSIM results for the 20 traditional images with different SPN ratios

Noise Density	10%	20%	30%	40%	50%	60%	70%	80%	90%
DBA	0.9796	0.9584	0.9315	0.8968	0.8520	0.7949	0.7213	0.6265	0.4966
MDBUTMF	0.9774	0.9197	0.8117	0.7973	0.8399	0.8410	0.8025	0.7023	0.3566
BPDF	0.9783	0.9536	0.9229	0.8838	0.8323	0.7634	0.6680	0.5096	0.2585
NAFSMF	0.9748	0.9504	0.9248	0.8973	0.8666	0.8320	0.7910	0.7357	0.6190
AWMF	0.9728	0.9622	0.9484	0.9315	0.9098	0.8816	0.8437	0.7904	0.7028
DAMF	0.9854	0.9699	0.9516	0.9303	0.9051	0.8748	0.8368	0.7846	0.6964
ARmF	**0.9868**	**0.9735**	**0.9581**	**0.9400**	**0.9173**	**0.8880**	**0.8491**	**0.7947**	**0.7056**

Table 15. The mean-VIF results for the 20 traditional images with different SPN ratios

Noise Density	10%	20%	30%	40%	50%	60%	70%	80%	90%
DBA	0.8548	0.7319	0.6179	0.5119	0.4095	0.3128	0.2229	0.1365	0.0635
MDBUTMF	0.8272	0.6713	0.5044	0.4420	0.4310	0.3978	0.3302	0.2212	0.0730
BPDF	0.8188	0.6858	0.5659	0.4564	0.3529	0.2541	0.1614	0.0783	0.0334
NAFSMF	0.7902	0.6751	0.5828	0.5030	0.4307	0.3604	0.2897	0.2129	0.1226
AWMF	0.7896	0.7366	0.6789	0.6181	0.5533	0.4833	0.4066	0.3129	0.1928
DAMF	0.8787	0.7816	0.6943	0.6162	0.5437	0.4731	0.3998	0.3096	0.1913
ARmF	**0.8832**	**0.7975**	**0.7210**	**0.6474**	**0.5741**	**0.4974**	**0.4158**	**0.3182**	**0.1955**

Assume that the success in high noise densities is more important than that in others. In this case, the values in Table 13, 14, and 15 can be represented in three *fpfs*-matrices as follows:

$$[a_{ij}] \coloneqq \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 0.8 & 0.9 \\ 0.9371 & 0.8564 & 0.7982 & 0.7450 & 0.6958 & 0.6466 & 0.5969 & 0.5382 & 0.4633 \\ 0.9191 & 0.8037 & 0.7248 & 0.7113 & 0.7195 & 0.7078 & 0.6731 & 0.5849 & 0.3819 \\ 0.9236 & 0.8377 & 0.7750 & 0.7213 & 0.6698 & 0.6144 & 0.5490 & 0.4431 & 0.2625 \\ 0.9011 & 0.8309 & 0.7865 & 0.7530 & 0.7248 & 0.6983 & 0.6698 & 0.6361 & 0.5579 \\ 0.9076 & 0.8741 & 0.8449 & 0.8164 & 0.7860 & 0.7527 & 0.7163 & 0.6741 & 0.6169 \\ 0.9885 & 0.9073 & 0.8526 & 0.8104 & 0.7740 & 0.7403 & 0.7063 & 0.6666 & 0.6081 \\ 1.0000 & 0.9271 & 0.8776 & 0.8374 & 0.7990 & 0.7605 & 0.7208 & 0.6763 & 0.6179 \end{bmatrix}$$

$$[b_{ik}] := \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 0.8 & 0.9 \\ 0.9796 & 0.9584 & 0.9315 & 0.8968 & 0.8520 & 0.7949 & 0.7213 & 0.6265 & 0.4966 \\ 0.9774 & 0.9197 & 0.8117 & 0.7973 & 0.8399 & 0.8410 & 0.8025 & 0.7023 & 0.3566 \\ 0.9783 & 0.9536 & 0.9229 & 0.8838 & 0.8323 & 0.7634 & 0.6680 & 0.5096 & 0.2585 \\ 0.9748 & 0.9504 & 0.9248 & 0.8973 & 0.8666 & 0.8320 & 0.7910 & 0.7357 & 0.6190 \\ 0.9728 & 0.9622 & 0.9484 & 0.9315 & 0.9098 & 0.8816 & 0.8437 & 0.7904 & 0.7028 \\ 0.9854 & 0.9699 & 0.9516 & 0.9303 & 0.9051 & 0.8748 & 0.8368 & 0.7846 & 0.6964 \\ 0.9868 & 0.9735 & 0.9581 & 0.9400 & 0.9173 & 0.8880 & 0.8491 & 0.7947 & 0.7056 \end{bmatrix}$$

and

$$[c_{it}] := \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 0.8 & 0.9 \\ 0.8548 & 0.7319 & 0.6179 & 0.5119 & 0.4095 & 0.3128 & 0.2229 & 0.1365 & 0.0635 \\ 0.8272 & 0.6713 & 0.5044 & 0.4420 & 0.4310 & 0.3978 & 0.3302 & 0.2212 & 0.0730 \\ 0.8188 & 0.6858 & 0.5659 & 0.4564 & 0.3529 & 0.2541 & 0.1614 & 0.0783 & 0.0334 \\ 0.7902 & 0.6751 & 0.5828 & 0.5030 & 0.4307 & 0.3604 & 0.2897 & 0.2129 & 0.1226 \\ 0.7896 & 0.7366 & 0.6789 & 0.6181 & 0.5533 & 0.4833 & 0.4066 & 0.3129 & 0.1928 \\ 0.8787 & 0.7816 & 0.6943 & 0.6162 & 0.5437 & 0.4731 & 0.3998 & 0.3096 & 0.1913 \\ 0.8832 & 0.7975 & 0.7210 & 0.6474 & 0.5741 & 0.4974 & 0.4158 & 0.3182 & 0.1955 \end{bmatrix}$$

Here, the entries of $[a_{ij}]$ except for its first row have been obtained by normalizing the values provided in Table 13 in consideration of the maximum value in the same table. If we apply EM20o to the *fpfs*-matrices $[a_{ij}]$, $[b_{ik}]$, and $[c_{it}]$, then the score matrix and the decision set are as follows:

$$[s_{i1}] = [0.4306 \quad 0.4712 \quad 0.3843 \quad 0.5889 \quad 0.5552 \quad 0.5473 \quad 0.5561]^T$$

and

$$\left\{ {}^{0.7742}\text{DBA}, {}^{0.8473}\text{MDBUTMF}, {}^{0.6911}\text{BPDF}, {}^{0.9151}\text{NAFSMF}, {}^{0.9984}\text{AWMF}, {}^{0.9841}\text{DAMF}, {}^{1}\text{ARmF} \right\}$$

The scores show that ARmF outperforms the others and the ranking order BPDF≺DBA≺MDBUTMF ≺NAFSMF≺DAMF≺AWMF≺ARmF is valid.

Assume that the success in low noise densities is more important than that in others. In this case, the values given in Table 13, 14, and 15 can be represented with three *fpfs*-matrices as follows:

$$[d_{ij}] := \begin{bmatrix} 0.9 & 0.8 & 0.7 & 0.6 & 0.5 & 0.4 & 0.3 & 0.2 & 0.1 \\ 0.9371 & 0.8564 & 0.7982 & 0.7450 & 0.6958 & 0.6466 & 0.5969 & 0.5382 & 0.4633 \\ 0.9191 & 0.8037 & 0.7248 & 0.7113 & 0.7195 & 0.7078 & 0.6731 & 0.5849 & 0.3819 \\ 0.9236 & 0.8377 & 0.7750 & 0.7213 & 0.6698 & 0.6144 & 0.5490 & 0.4431 & 0.2625 \\ 0.9011 & 0.8309 & 0.7865 & 0.7530 & 0.7248 & 0.6983 & 0.6698 & 0.6361 & 0.5579 \\ 0.9076 & 0.8741 & 0.8449 & 0.8164 & 0.7860 & 0.7527 & 0.7163 & 0.6741 & 0.6169 \\ 0.9885 & 0.9073 & 0.8526 & 0.8104 & 0.7740 & 0.7403 & 0.7063 & 0.6666 & 0.6081 \\ 1.0000 & 0.9271 & 0.8776 & 0.8374 & 0.7990 & 0.7605 & 0.7208 & 0.6763 & 0.6179 \end{bmatrix}$$

$$[e_{ik}] := \begin{bmatrix} 0.9 & 0.8 & 0.7 & 0.6 & 0.5 & 0.4 & 0.3 & 0.2 & 0.1 \\ 0.9796 & 0.9584 & 0.9315 & 0.8968 & 0.8520 & 0.7949 & 0.7213 & 0.6265 & 0.4966 \\ 0.9774 & 0.9197 & 0.8117 & 0.7973 & 0.8399 & 0.8410 & 0.8025 & 0.7023 & 0.3566 \\ 0.9783 & 0.9536 & 0.9229 & 0.8838 & 0.8323 & 0.7634 & 0.6680 & 0.5096 & 0.2585 \\ 0.9748 & 0.9504 & 0.9248 & 0.8973 & 0.8666 & 0.8320 & 0.7910 & 0.7357 & 0.6190 \\ 0.9728 & 0.9622 & 0.9484 & 0.9315 & 0.9098 & 0.8816 & 0.8437 & 0.7904 & 0.7028 \\ 0.9854 & 0.9699 & 0.9516 & 0.9303 & 0.9051 & 0.8748 & 0.8368 & 0.7846 & 0.6964 \\ 0.9868 & 0.9735 & 0.9581 & 0.9400 & 0.9173 & 0.8880 & 0.8491 & 0.7947 & 0.7056 \end{bmatrix}$$

and

$$[f_{it}] \coloneqq \begin{bmatrix} 0.9 & 0.8 & 0.7 & 0.6 & 0.5 & 0.4 & 0.3 & 0.2 & 0.1 \\ 0.8548 & 0.7319 & 0.6179 & 0.5119 & 0.4095 & 0.3128 & 0.2229 & 0.1365 & 0.0635 \\ 0.8272 & 0.6713 & 0.5044 & 0.4420 & 0.4310 & 0.3978 & 0.3302 & 0.2212 & 0.0730 \\ 0.8188 & 0.6858 & 0.5659 & 0.4564 & 0.3529 & 0.2541 & 0.1614 & 0.0783 & 0.0334 \\ 0.7902 & 0.6751 & 0.5828 & 0.5030 & 0.4307 & 0.3604 & 0.2897 & 0.2129 & 0.1226 \\ 0.7896 & 0.7366 & 0.6789 & 0.6181 & 0.5533 & 0.4833 & 0.4066 & 0.3129 & 0.1928 \\ 0.8787 & 0.7816 & 0.6943 & 0.6162 & 0.5437 & 0.4731 & 0.3998 & 0.3096 & 0.1913 \\ 0.8832 & 0.7975 & 0.7210 & 0.6474 & 0.5741 & 0.4974 & 0.4158 & 0.3182 & 0.1955 \end{bmatrix}$$

Here, the entries of $[d_{ij}]$ except for its first row have been obtained by normalizing the values provided in Table 13 in view of the maximum value provided therein. If we apply EM20o to the *fpfs*-matrices $[d_{ij}]$, $[e_{ik}]$, and $[f_{it}]$, then the score matrix and the decision set are as follows:

$$[s_{i1}] = [0.8334 \quad 0.8272 \quad 0.8312 \quad 0.8110 \quad 0.8168 \quad 0.8897 \quad 0.9000]^T$$

and

$$\{{}^{0.9371}\text{DBA}, {}^{0.9191}\text{MDBUTMF}, {}^{0.9236}\text{BPDF}, {}^{0.9011}\text{NAFSMF}, {}^{0.9076}\text{AWMF}, {}^{0.9885}\text{DAMF}, {}^{1}\text{ARmF}\}$$

The scores show that ARmF outperforms the others and the ranking order NAFSMF≺AWMF ≺MDBUTMF≺BPDF≺DBA≺DAMF≺ARmF is valid.

Secondly, we apply RM13a, EM20a, RM13o, and EM20o to *fpfs*-matrices mentioned above and compare the ranking orders of the algorithms in Table 16.

Table 16. The ranking orders of the filters for RM13a, EM20a, RM13o, and EM20o

Algorithms	**Matrices**	**Ranking Orders**
RM13a	$[a_{ij}]$,$[b_{ik}]$,$[c_{it}]$	BPDF≺DBA≺MDBUTMF≺AWMF≺NAFSMF≺DAMF≺ARmF
EM20a	$[a_{ij}]$,$[b_{ik}]$,$[c_{it}]$	BPDF≺DBA≺MDBUTMF≺AWMF≺NAFSMF≺DAMF≺ARmF
RM13o	$[a_{ij}]$,$[b_{ik}]$,$[c_{it}]$	BPDF≺DBA≺MDBUTMF≺NAFSMF≺DAMF≺AWMF≺ARmF
EM20o	$[a_{ij}]$,$[b_{ik}]$,$[c_{it}]$	BPDF≺DBA≺MDBUTMF≺NAFSMF≺DAMF≺AWMF≺ARmF
RM13a	$[d_{ij}]$, $[e_{ik}]$,$[f_{it}]$	BPDF≺DBA≺MDBUTMF≺NAFSMF≺DAMF≺AWMF≺ARmF
EM20a	$[d_{ij}]$, $[e_{ik}]$,$[f_{it}]$	BPDF≺DBA≺MDBUTMF≺NAFSMF≺DAMF≺AWMF≺ARmF
RM13o	$[d_{ij}]$, $[e_{ik}]$,$[f_{it}]$	NAFSMF≺AWMF≺MDBUTMF≺BPDF≺DBA≺DAMF≺ARmF
EM20o	$[d_{ij}]$, $[e_{ik}]$,$[f_{it}]$	NAFSMF≺AWMF≺MDBUTMF≺BPDF≺DBA≺DAMF≺ARmF

Finally, we compare EM20o with four state-of-art soft decision-making methods sDB12 (Enginoğlu and Memiş, 2018c), EMC19o (Enginoğlu et al., 2019b), EMO18o (Enginoğlu et al., 2018b), and sMBR01 (Enginoğlu and Memiş, 2018b) by using the aforesaid *fpfs*-matrices $[a_{ij}]$,$[b_{ik}]$, and $[c_{it}]$. The results in Table 17 show that EM20o produce suitable ranking order to the state-of-art methods and experts' views.

Table 17. The ranking orders of the filters for the state-of-art methods

Algorithms	**Matrices**	**Ranking Orders**
EM20o	$[a_{ij}]$,$[b_{ik}]$,$[c_{it}]$	BPDF≺DBA≺MDBUTMF≺NAFSMF≺DAMF≺AWMF≺ARmF
sDB12	$[a_{ij}]$,$[b_{ik}]$,$[c_{it}]$	BPDF≺DBA≺MDBUTMF≺NAFSMF≺DAMF≺AWMF≺ARmF
EMC19o	$[a_{ij}]$,$[b_{ik}]$	BPDF≺DBA≺MDBUTMF≺NAFSMF≺DAMF≺AWMF≺ARmF
EMO18o	$[a_{ij}]$,$[b_{ik}]$	BPDF≺DBA≺MDBUTMF≺NAFSMF≺DAMF≺AWMF≺ARmF
sMBR01	$[a_{ij}]$	BPDF≺DBA≺MDBUTMF≺NAFSMF≺DAMF≺AWMF≺ARmF

It must be noted that EM20o and sDB12 algorithms use three *fpfs*-matrices, EMC19o and EMO18o algorithms use two *fpfs*-matrices, and sMBR01 algorithm uses one *fpfs*-matrix for the decision-making process.

4. Conclusion

WSMmDM and WFSMmDM have been proposed by Razak and Mohamad (2011, 2013). Recently, since such methods cannot model decision-making problems in the event that the parameters have uncertainties, these two methods have been configured (Enginoğlu and Memiş, 2018a). However, the configured method has a drawback such as its incapability of processing a large number of parameters on such a standard computer with 2.6 GHz i5 Dual-Core CPU and 4GB RAM.

In this paper, we proposed the method EM20a, which is faster than RM13a, and the method EM20o, which is faster than RM13o. Of course, simplifications of these methods can be investigated in view of other products.

Additionally, we compared the aforesaid methods in terms of their running time data. Besides, the results in Section of Simulation Results and the results in Table 18 and 19 too evidence that EM20a and EM20o perform better than RM13a and RM13o, respectively.

Table 18. The mean advantages and max advantages of EM20a over RM13a and max differences between EM20a and RM13a

Location	Objects	Parameters	Mean Advantage %	Max Advantage %	Max Difference
Table 1	10	10-100	86.9177	98.7605	0.0199
Table 2	10	1000-10000	99.9335	99.9992	1046.9268
Table 3	10-100	10	78.6237	88.0516	0.0147
Table 4	1000-10000	10	96.1903	98.1778	4.7890
Table 5	10-100	10-100	90.9713	99.3137	0.1584
Table 6	100-1000	100-1000	99.4646	99.9954	1292.8260

Table 19. The mean advantages and max advantages of EM20o over RM13o and max differences between EM20o and RM13o

Location	Objects	Parameters	Mean Advantage %	Max Advantage %	Max Difference
Table 7	10	10-100	86.0065	97.7622	0.0205
Table 8	10	1000-10000	99.9299	99.9992	1049.7150
Table 9	10-100	10	77.3897	87.4118	0.0129
Table 10	1000-10000	10	95.9499	98.1925	4.7899
Table 11	10-100	10-100	90.2170	99.2587	0.1515
Table 12	100-1000	100-1000	99.4735	99.9954	1288.2528

Finally, it is suggested that the methods constructed by means of min-max, max-max, and min-min decision functions should also be studied. Thus, applying such soft decision-making methods to more area can be possible. For more details, see (Enginoğlu and Aydın, 2019; Enginoğlu et al., 2019a,b,c,d; Memiş and Enginoğlu, 2019; Memiş et al., 2019; Enginoğlu and Öngel, 2020).

Acknowledgement

The authors thank Dr. Uğur Erkan for technical support. This work was supported by the Office of Scientific Research Projects at Çanakkale Onsekiz Mart University, Grant number: FBA-2018-1367.

References

Çağman, N., Enginoğlu, S. 2010a. Soft Set Theory and Uni-Int Decision Making. European Journal of Operational Research, 207(2): 848-855.

Çağman, N., Enginoğlu, S. 2010b. Soft Matrix Theory and Its Decision Making. Computer and Mathematics with Applications, 59(10): 3308-3314.

Çağman, N., Enginoğlu, S. 2012. Fuzzy Soft Matrix Theory and Its Application in Decision Making. Iranian Journal of Fuzzy Systems, 9(1): 109-119.

Çağman, N., Çıtak, F., Enginoğlu, S. 2010. Fuzzy Parameterized Fuzzy Soft Set Theory and Its Applications. Turkish Journal of Fuzzy Systems, 1(1): 21-35.

Çağman, N., Çıtak, F., Enginoğlu, S. 2011a. FP-soft Set Theory and Its Applications. Annals of Fuzzy Mathematics and Informatics, 2(2): 219-226.

Çağman, N., Enginoğlu, S., Çıtak, F. 2011b. Fuzzy Soft Set Theory and Its Applications. Iranian Journal of Fuzzy Systems, 8(3): 137-147.

Deli, İ., Çağman, N. 2015. Relations on FP-soft Sets Applied to Decision Making Problems. Journal of New Theory, 3: 98-107.

Enginoğlu, S., Soft matrices, 2012. PhD Dissertation, Tokat Gaziosmanpaşa University, Graduate School of Natural and Applied Sciences, p. 89, Tokat, Turkey (In Turkish).

Enginoğlu, S., Çağman, N. (n.d.) Fuzzy Parameterized Fuzzy Soft Matrices and Their Application in Decision-Making. TWMS Journal of Applied and Engineering Mathematics, In Press.

Enginoğlu, S., Memiş, S. 2018a. A Configuration of Some Soft Decision-Making Algorithms via fpfs-matrices. Cumhuriyet Science Journal, 39(4): 871-881.

Enginoğlu, S., Memiş, S. 2018b. Comment on "Fuzzy soft sets" [The Journal of Fuzzy Mathematics, 9(3), 2001, 589–602], International Journal of Latest Engineering Research and Applications, 3(9): 1-9.

Enginoğlu, S., Memiş, S. 2018c. A Review on An Application of Fuzzy Soft Set in Multicriteria Decision Making Problem [P.K. Das, R. Borgohain, International Journal of Computer Applications 38 (12) (2012) 33-37]. Eds: Akgül, M., Yılmaz, İ., İpek, A. Proceeding of The International Conference on Mathematical Studies and Applications 2018, pp. 173-178, October 4-6, Karaman, Turkey.

Enginoğlu, S., Çağman, N., Karataş, S., Aydın, T. 2015. On Soft Topology. El-Cezerî Journal of Science and Engineering, 2(3): 23-38.

Enginoğlu, S., Memiş, S., Arslan, B. 2018a. Comment (2) on Soft Set Theory and uni-int Decision Making [European Journal of Operational Research, (2010) 207, 848-855]. Journal of New Theory, 25: 85-102.

Enginoğlu, S., Memiş, S., Öngel, T. 2018b. Comment on Soft Set Theory and uni-int Decision Making [European Journal of Operational Research, (2010) 207, 848-855]. Journal of New Results in Science, 7(3): 28-43.

Enginoğlu, S., Ay, M., Çağman, N. Tolun, V. 2019a. Classification of the Monolithic Columns Produced in Troad and Mysia Region Ancient Granite Quarries in Northwestern Anatolia via Soft Decision-Making. Bilge International Journal of Science and Technology Research, 3(Special Issue): 21-34.

Enginoğlu, S., Aydın, T. 2019. A Configuration of Five of the Soft Decision-Making Methods via Fuzzy Parameterized Fuzzy Soft Matrices and Their Application to a Performance-Based Value Assignment Problem. Eds: Kılıç, M., Özkan, K., Karaboyacı, M., Taşdelen, K., Kandemir, H., Beram, A. Proceedings of International Conferences on Science and Technology Natural Science and Technology, pp. 56-67, August 26-30, Prizren, Kosovo.

Enginoğlu, S., Erkan, U., Memiş, S. 2019b. Pixel Similarity-Based Adaptive Riesz Mean Filter for Salt-and-Pepper Noise Removal. Multimedia Tools and Applications, 78(24): 35401-35418.

Enginoğlu, S., Memiş, S., Çağman, N. 2019c. A Generalisation of Fuzzy Soft Max-Min Decision-Making Method and Its Application to A Performance-Based Value Assignment in Image Denoising. El-Cezerî Journal of Science and Engineering, 6(3): 466-481.

Enginoğlu, S., Memiş, S. Karaaslan, F. 2019d. A New Approach to Group Decision-Making Method Based on TOPSIS under Fuzzy Soft Environment. Journal of New Results in Science, 8(2): 42-52.

Enginoğlu, S., Öngel, T. 2020. Configurations of Several Soft Decision-Making Methods to Operate in Fuzzy Parameterized Fuzzy Soft Matrices Space. Eskişehir Technical University Journal of Science and Technology A-Applied Sciences and Engineering, 21(1): 58-71.

Maji, P. K., Biswas, R., Roy, A. R. 2001. Fuzzy Soft Sets. The Journal of Fuzzy Mathematics, 9(3): 589-602.

Maji, P. K., Biswas, R., Roy, A. R. 2003. Soft Set Theory. Computers and Mathematics with Applications, 45(4-5): 555-562.

Memiş, S. Enginoğlu, S. 2019. An Application of Fuzzy Parameterized Fuzzy Soft Matrices in Data Classification. Eds: Kılıç, M., Özkan, K., Karaboyacı, M., Taşdelen, K., Kandemir, H., Beram, A. Proceedings of International Conferences on Science and Technology Natural Science and Technology, pp. 68-77, August 26-30, Prizren, Kosovo.

Memiş, S., Enginoğlu, S. Erkan, U. 2019. A Data Classification Method in Machine Learning Based on Normalised Hamming Pseudo-Similarity of Fuzzy Parameterized Fuzzy Soft Matrices. Bilge International Journal of Science and Technology Research, 3(Special Issue): 1-8.

Molodtsov, D. 1999. Soft Set Theory-First Results. Computers and Mathematics with Applications, 37(4-5): 19-31.

Razak, S. A., Mohamad, D. 2011. A Soft Set Based Group Decision Making Method with Criteria Weight. International Scholarly and Scientific Research & Innovation, 5(10): 1641-1646.

Razak, S. A., Mohamad, D., A. 2013. Decision Making Method using Fuzzy Soft Sets. Malaysian Journal of Fundamental and Applied Sciences, 9(2): 99-104.

Riaz, M., Hashmi, R. 2017. Fuzzy Parameterized Fuzzy Soft Topology with Applications. Annals of Fuzzy Mathematics and Informatics, 2017, 13(5): 593-613.

Riaz, M., Hashmi, R. 2018. Fuzzy Parameterized Fuzzy Soft Compact Spaces with Decision-Making, Punjab University Journal of Mathematics, 50(2): 131-145.

Riaz, M., Hashmi, R., Farooq, A. 2018. Fuzzy Parameterized Fuzzy Soft Metric Spaces. Journal of Mathematical Analysis, 9(2): 25-36.

Sezgin, A., Çağman, N., Çıtak, F. 2019. α-Inclusions Applied to Group Theory via Soft Set and Logic. Communications Faculty of Sciences University of Ankara Series A1 Mathematics and Statistics, 68(1): 334-352.

Şenel, G. 2018. Analyzing the Locus of Soft Spheres: Illustrative Cases and Drawings. European Journal of Pure and Applied Mathematics, 11(4): 946-957.

Ullah, A., Karaaslan, F., Ahmad, I. 2018. Soft Uni-Abel-Grassmann's Groups. European Journal of Pure and Applied Mathematics, 11(2): 517-536.

Zorlutuna, İ., Atmaca, S. 2016. Fuzzy Parametrized Fuzzy Soft Topology. New Trends in Mathematical Sciences, 4(1): 142-152.