Concept Reduction Methods

Miklós F. Hatwagner

Abstract <To be prepared>

1 The Motivating Problem

The title of Adrienn Buruzs's PhD thesis [1] is "Evaluation of Sustainable Regional Waste Management Systems with Fuzzy Cognitive Map". As the title suggests, she analyzed the internal driving forces, dynamic behavior and sustainability of Integrated Waste Management Systems (IWMSs), which are very complex systems including many aspects (environmental, economic, social, institutional, legal and technical) and stakeholders. Even at an early stage of her investigations became apparent that Fuzzy Cognitive Map (FCM) is an appropriate tool to describe the large number of interacting and coupled entities and it copes with the inherent uncertainties of the system.

At first, a new FCM model [3] was created, which contained six concepts. These concepts were identified on the basis of the literature. The strength of relationships among concepts were defined by the results of a survey filled out by 75 stakeholders. The simulation results provided by FCM were validated later in [5]. Time series data were collected based on the relevant literature and it served as the input of a Bacterial Evolutionary Algorithm to learn the connection weights among the already specified concepts and parameter λ of the threshold function. The goal of optimization was to find an FCM that generates as similar time series as possible. Unfortunately, a strong contradiction was explored between the models created by experts and machine learning.

In order to resolve the experienced problem the concepts of the original model were decomposed to further 4-7 sub-concepts according to the System-of-Systems approach, which led to a very detailed, completely new model of IWMS [4]. A

Miklós F. Hatwagner
Széchenyi István University, Győr, Hungary e-mail: miklos.hatwagner@sze.hu

workshop was organized with the help of 12 stakeholders who decided the sub-concepts and their interconnections. The result of their work is a FCM containing 33 concepts in total (Fig. 1). Unfortunately such an extremely complex model is often confusing for the experts (Fig. 2), and to work with them may be very laborious. Note that the number of connections is a quadratic function of the number of concepts.

That is why, in general, the following approach is suggested to follow in practice: start with an obviously oversized, fine-grained model. Experts are often uncertain about the importance of system components thus it worth include most or all of them in the preliminary model. Then start reducing the model automatically, in an algorithmic way until the balance of model size and required accuracy is found. The numerical accuracy of reduced models are always lower by their nature, but it does not cause a problem in practice until the decisions suggested by them are the same. On the other hand, simpler models are easier to understand, their visual representation is clearer, and it is often more important for experts and managers. In the following sections several possible ways of model reduction is presented.

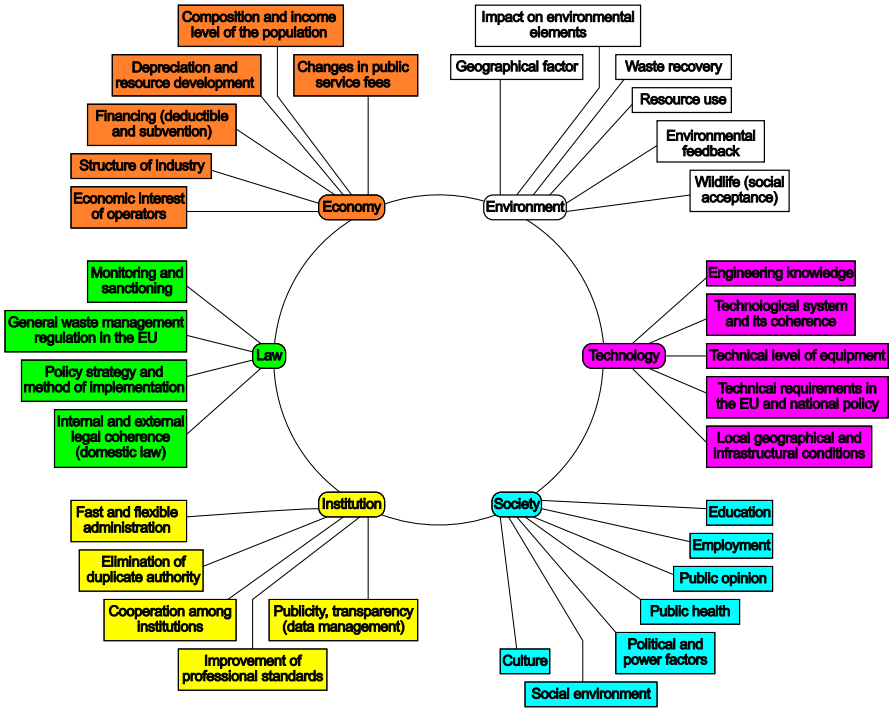

Fig. 1 The main concepts and their sub-concepts of regional IWMS.

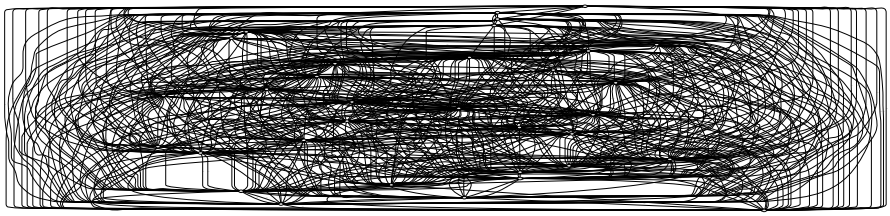

Fig. 2 The 33 concept model of regional IWMS and the relationships among its concepts. It is hard to illustrate complex FCM models appropriately and graphical model visualizations often confuse experts.

2 Early model reduction methods

Several methods had been suggested to solve the problem of oversized models before the complex model of IWMS saw the light of the day. These approaches are based on different perspectives.

In [2] an FCM is learned using historical data and its concepts are grouped into clusters in a unique way. The clustering is based on the DEMATEL [6] method. The concepts are arranged on a two dimensional plot. The vertical axis classifies concepts to cause and effect groups, the position of concepts along the horizontal axis expresses the importance of them. Based on this rearrangement of concepts two clustering methods are suggested. The first one uses K-Means clustering to create clusters according to the cause-effect behavior of concepts. The cluster centers replace the original concepts in the reduced model. The second method contains two consecutive steps, and takes also the importance of concepts into consideration. Regardless of the methods applied, experts have to define the number of clusters and they must be disjoint.

FCM was used to predict and discover knowledge about the HIV-1 drug resistance in [15]. The protease protein was modeled by FCM and causalities among sequence positions were estimated by Particle Swarm Optimization. Furthermore, Ant Colony Optimization was also applied in order to find the strongest sequence positions related to the resistance target. After that, some concepts and their connections were removed to decrease the complexity of the model until the quality of inference remained acceptable.

Another reduction method was introduced in [11]. Weak concepts and their connections were removed, then the inference capabilities of the simplified model were tested with time series data collected from real-world applications. The reduced model replaced the original if its estimation errors were acceptable.

3 Fuzzy Tolerance Relations-based reduction methods

The first results on a novel state reduction method family were presented in [8, 9]. The members of this family differ only in the applied metric, which is used to measure the "distance" of concepts from each other. The methods may be considered as generalizations of the state reduction of finite state machines and sequential systems with partially defined states. They are widely applied in digital design where the complexity of the problem makes it possible [13]. The common idea is to create clusters of identical or similar concepts, and use these clusters instead of their members in the reduced model. The methods are based on Fuzzy Tolerance Relations (FTR), which is an extension of compatibility relations among crisp concepts. The methods examine the connection weights among concepts, because they define the effect on other concepts.

3.1 Description of the main functions used for reduction

At the very beginning the new model contains exactly as many clusters as the number of concepts in the original model. These clusters are all disjoint single element sets containing one of the original concepts. In the following, the i^{th} concept is denoted by C_i, and similarly, the i^{th} cluster is denoted by K_i. The number of concepts is n. In the next steps further concepts are added to the clusters if they are "close enough" to each member of the cluster. The "distance" of concepts can be measured by various metrics. These metrics can be selected according to the specialities of the problem, and they differentiate the members of the algorithm family. Finally, some of these expanded clusters may be identical, containing exactly the same concepts. In order to keep clusters unique, only one of these clusters is retained.

One of the main functions is called $buildCluster$ (Algorithm 1). Its goal is to create a cluster that initially contains only its $initialConcept$. Later this cluster will be expanded by merger of other concepts. The second parameter, ϵ specifies the maximum allowed distance between current cluster members and the concept under examination. The value of ϵ must be in the [0, 1] interval. This parameter plays an important role in model reduction, and have to be chosen properly by experts. Low values hardly reduce the size of the model, but high values may cause oversimplification. Its appropriate value is completely problem dependent.

Function $isNearA$ is called several times in $buildCluster$ to decide whether the current concept C_i can become a member of cluster K or not. The number of function calls depends on how many member concepts do the cluster already have. This function (Algorithm 2) implements one of the possible metrics, but can be repleced by any of the metrics presented here. The last one was suggested in [7].

Metric "A" calculates the absolute difference of two connection weights, $w_{i,k}$ and $w_{j,k}$ from concept C_i, a cluster member candidate and concept C_j, a concept of cluster K to a third concept C_k, where $i \neq j \neq k$, and $i, j, k = 1, ..., n$ and n is the number of concepts. If the half of both this distance and the distance of weights

Algorithm 1 The *buildCluster* function

```
1: function BUILDCLUSTER(initialConcept, ε)
2:     K ← {initialConcept}
3:     for i ← 0; i < n; i++ do
4:         if i ≠ initialConcept then
5:             member ← true
6:             while member and HASNEXTELEMENT(K) do
7:                 j ← NEXTELEMENT(K)
8:                 member ← ISNEARA(j, i, ε)
9:             end while
10:            if member then
11:                K ← K + {i}
12:            end if
13:        end if
14:    end for
15:    return c
16: end function
```

Algorithm 2 Function *isNearA* implementing *Metric "A"*

```
1: function ISNEARA(i, j, ε)
2:     near ← true
3:     for k ← 0; k < n and near = true; k++ do
4:         if k ≠ i and k ≠ j then
5:             if |w_{i,k} − w_{j,k}| / 2 ≥ ε or |w_{k,i} − w_{k,j}| / 2 ≥ ε then
6:                 near ← false
7:             end if
8:         end if
9:     end for
10:    return near
11: end function
```

in the opposite direction are below the design variable ϵ for all C_k, C_i is added to cluster K.

Metric "B" (Algorithm 3) is a slightly modified version of Metric "A". The latter works well in most cases, but sometimes a small proportion of weight differences exceed the allowed value, and prevent the merger of concepts. The second metric provides a simple solution to this problem with the usage of parameter p. The metric allows the merger even if the distances are greater than ϵ in a small, less than p proportion of the investigated cases.

Theoretically the value of p can be any in the $[0, 1]$ interval, but it should be rather low, however. High p values makes possible to practically merge any concepts regardless the value of ϵ, which propably leads to an unusable model.

Metric "B" allowed to lower the value of parameter ϵ with a simple trick, but the third approach (Metric "C", Algorithm 4) allows the omittance of the second parameter p by using the normalized, squared Euclidean distance. It is much easier to tune only one parameter instead of two in practice.

Finally, Metric "D" (Algorithm 5) works with the Manhattan-distance of concepts.

Algorithm 3 Function *isNearB* implementing *Metric "B"*

```
1: function isNearB(i, j, ϵ, p)
2:     near ← 0
3:     far ← 0
4:     for k ← 0; k < n; k++ do
5:         if k ≠ i and k ≠ j then
6:             if |w_{i,k} − w_{j,k}| / 2 < ϵ then
7:                 near ← near + 1
8:             else
9:                 far ← far + 1
10:            end if
11:            if |w_{k,i} − w_{k,j}| / 2 < ϵ then
12:                near ← near + 1
13:            else
14:                far ← far + 1
15:            end if
16:        end if
17:    end for
18:    if near = 0 or far/near ≥ p then
19:        return false
20:    else
21:        return true
22:    end if
23: end function
```

Algorithm 4 Function *isNearC* implementing *Metric "C"*

```
1: function isNearC(i, j, ϵ)
2:     sum ← 0
3:     for k ← 0; k < n; k++ do
4:         if k ≠ i and k ≠ j then
5:             sum ← sum + (w_{i,k} − w_{j,k})^2
6:             sum ← sum + (w_{k,i} − w_{k,j})^2
7:         end if
8:     end for
9:     if sum / ((n−2)*8) < ϵ then
10:        return true
11:    else
12:        return false
13:    end if
14: end function
```

The applied metrics, its parameters (ϵ, p) and also the details of implementation not specified here may affect the result of reduction. For example, if the concepts provided by $nextElement$ are in various orders, the content of clusters may be different, even is the size of the reduced model remains the same. The results should be revised by experts.

The $buildCluster$ function creates one of the clusters of the reduced model. Another function, $buildAllClusters$ (Algorithm 6) calls $buildCluster$ subsequently with different $initialConcept$ values. According to the nature of the method, multiple

Algorithm 5 Function *isNearD* implementing *Metric "D"*

```
1: function isNearD(i, j, ε)
2:     sum ← 0
3:     for k ← 0; k < n; k++ do
4:         if k ≠ i and k ≠ j then
5:             sum ← sum + |w_{i,k} − w_{j,k}|
6:             sum ← sum + |w_{k,i} − w_{k,j}|
7:         end if
8:     end for
9:     if sum/((n−2)*4) < ε then
10:        return true
11:    else
12:        return false
13:    end if
14: end function
```

clusters may contain the same concepts. Only one the same clusters will be kept by the algorithm.

Algorithm 6 The *buildAllClusters* function

```
1: function buildAllClusters(ε)
2:     clusters ← {}
3:     for i ← 0; i < n; i++ do
4:         K ← buildCluster(i, ε)
5:         if !isElementOf(K, clusters) then
6:             clusters ← clusters + {K}
7:         end if
8:     end for
9:     return clusters
10: end function
```

Function *buildAllClusters* returns the clusters, the concepts of the reduced model, but the weights among clusters have to be defined also. Function *getWeight* (Algorithm 7) investigates the members of its parameter clusters, a and b, and calculates the average (arithmetic mean) weight of relationships between concepts included in cluster a to the concepts of cluster b. This function must be called to all possible a, b pairs to completely define the reduced model.

[16] presents step-by-step the reduction process of a straightforward, five-concept model in detail.

3.2 Reduction of the motivating problem

The original model of the motivating problem (IWMS) contains 33 concepts and 638 of the theoretically possible 1056 connections, making the usage of the model very

Algorithm 7 The *getWeight* function

```
1: function GETWEIGHT(a, b)
2:     count ← 0
3:     sum ← 0
4:     while HASNEXTELEMENT(a) do
5:         i = NEXTELEMENT(a)
6:         while HASNEXTELEMENT(b) do
7:             j = NEXTELEMENT(b)
8:             if i ≠ j then
9:                 count ← count + 1
10:                sum ← sum + w_{i,j}
11:            end if
12:        end while
13:    end while
14:    if count = 0 then
15:        return 0
16:    else
17:        return sum / count
18:    end if
19: end function
```

Table 1 The number of concepts in the reduced connection matrix, using different metrics [7]

Metric "A"		Metric "B"			Metric "C"		Metric "D"	
ϵ	No. of concepts	ϵ	p	No. of concepts	ϵ	No. of concepts	ϵ	No. of concepts
0.3	28	0.1	0.2	30	0.01	30	0.052	32
0.4	25	0.2	0.05	30	0.016	28	0.059	30
0.5	18	0.2	0.1	26	0.022	24	0.078	28
0.6	15	0.2	0.2	23	0.027	22	0.097	26
0.7	12	0.3	0.05	23	0.04	20	0.1	24
0.8	4	0.3	0.1	21	0.048	18	0.104	22
		0.3	0.2	15	0.054	15	0.149	20
		0.4	0.05	19	0.06	12	0.173	18
		0.4	0.1	10			0.188	15
							0.2	13

cumbersome for experts. Several experiments were conducted with different metrics and parameter (ϵ, p) values. This way the connection between reduction parameters and the evoked extent of reduction can be studied. Some interesting results are collected in Table 1.

Of course, the number of concepts after reduction does not characterize the usefulness of the model.

3.3 Theoretical background of FTR-based methods

The name of the reduction methods covered in this section refer to the fact that they all based on Fuzzy Tolerance Relations (FTR). If a binary relation $R(x, x)$ is *reflexive* and *symmetric*, but *not transitive* then it is called a compatibility relation in crisp contexts and *tolerance relation* in fuzzy contexts [12]. (All further definitions in this subsection are also quoted from [12].) A fuzzy binary relation "$R(x, x)$ is *reflexive* iff $R(x, x) = 1$ for all $x \in X$. A fuzzy relation is *symmetric* iff $R(x, y) = R(y, x)$ for all $x, y \in X$". The authors emphasize again that *a FTR is never transitive*. ("A fuzzy relation $R(x, x)$ is transitive (. . .) if $R(x, z) \geq \max_{y \in Y} \min[R(x, y), R(y, z)]$ is satisfied for each pair $\langle x, z \rangle \in X^2$.")

The four metrics defined by *isNear* functions are all real distance functions. As such, the following conditions are satisfied by function d, where $d : R(X, X) \rightarrow \mathbb{R}, \forall x, y, z \in X$:

1. $d(x, x) \geq 0$
2. $d(x, y) = 0$ iff $x = y$
3. $d(x, y) = d(y, x)$ (symmetry)
4. $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality)

The applied metrics are symmetric functions, and *isNear* functions always return true ($R(x, x) = 1$ or $\mu = 1$) if values of the parameters are the same (reflexivity). These metrics generate non-transitive mergers, therefore they create FTRs.

Function *buildAllClusters* initiates the process of cluster building with every single concepts as *initialConcept*. The called *buildCluster* function may extend these initially single element clusters with other concepts according to the connection weights among concepts and the value of parameter ϵ. According to this behavior, the following properties hold:

- all concepts of the original model will be included in at least one cluster,
- the same concept may be included in multiple clusters, thus clusters overlap (Fig. 3).

3.4 Two-Stage Learning Based Reduction

Original and reduced models obviously behave in a different way because the latter have less concepts, the concepts represent different objects of the real system, and the relationships among concepts must be different as well. The extent of differences rely primarily on the design parameter ϵ of model reduction. Certain differences in model behaviors do not mean a problem until the same decisions can be made with both models. Unfortunately behavioral differences are hard to measure, because the values of concepts representing different objects cannot be compared directly to each other. In order to overcome this difficulty the following actions were taken in [10].

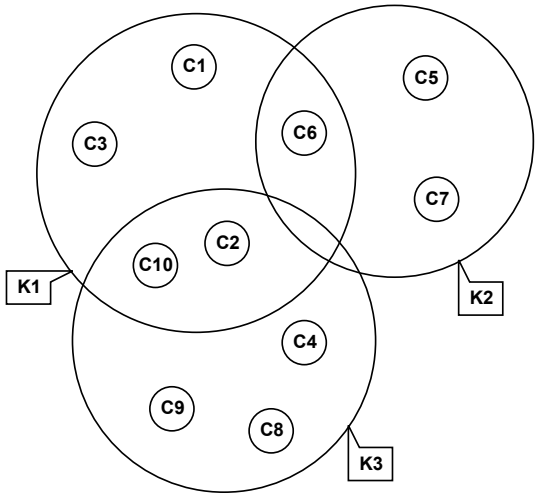

Fig. 3 Basic example of overlapping clusters [7]

3.4.1 Distinction of concept groups

Three groups of concepts were distingiuished:

1. Concepts affecting other concepts but not affected by other concepts are called *input concepts*. They serve as the inputs of the modeled system. The states of these concepts remain the same during simulations, thus they neither need transformation function nor its parameter λ.
2. Some concepts are both affected by other concepts and they also have an effect on more or less concepts. These are the *intermediate concepts*.
3. The third group of concepts is the *output concepts* (or decision concepts). Their states are defined by other concepts, but they do not influence the state of any other concepts.

In order to make the behavioral difference of original and reduced models (the error of simulation) measurable, a modified reduction method was suggested in [10], that allows the merger of intermediate concepts only. This way simulations can be started with the same initial input concept states and the results in output concepts can be directly compared. The relation between the value of the design parameter ϵ and simulation error of the reduced model can be defined statistically using several models, many initial state vectors and ϵ values.

3.4.2 Modified versions of the *getWeight* function

The *getWeight* function was also revised. The original version of it (*average*, Algorithm 7) defines the weight of connection between two clusters (a and b) as the arithmetic mean (average) weight of connections between concepts included in the

corresponding clusters. More formally, the weight calculation can be expressed by Eq. 1.

$$w_a(a,b) = \frac{\sum_{i\in a}\sum_{j\in b} a(i,j)\times w_{ij}}{\sum_{i\in a}\sum_{j\in b} a(i,j)} \tag{1}$$

where $a(i,j)$ is defined as

$$a(i,j) = \begin{cases} 1 \text{ if } w_{ij} \neq 0, \\ 0 \text{ otherwise.} \end{cases} \tag{2}$$

The second approach (*weighted average*) calculates the weighted average of connection weights among cluster members. Greater inter-concept weights play more important role in the definition of inter-cluster weights (see Eq. 3).

$$w_w(a,b) = \frac{\sum_{i\in a}\sum_{j\in b} a(i,j)\times |w_{ij}|\times w_{ij}}{\sum_{i\in a}\sum_{j\in b} a(i,j)\times |w_{ij}|} \tag{3}$$

The third method (*extreme*, Eq. 4) selects the connection between clusters with the maximum absolute value, and that will be used between the clusters. If there are conncections with the same absolute value but different sign, the positive one will be chosen.

$$w_e(a,b) = \begin{cases} min(w_{ij}) & \text{if } \left|min(w_{ij})\right| > \left|max(w_{ij})\right| \\ & \text{for } \forall i \in a \wedge \forall j \in b, \\ max(w_{ij}) & \text{otherwise.} \end{cases} \tag{4}$$

The last method (*sum*, Eq. 5) adds up the weight of connections among clusters. If that would not fit in the allowed $[-1,\ +1]$ interval, the method rounds the weight up or down to the nearest allowed value.

$$w_s(a,b) = \begin{cases} 1 & \text{if } s(a,b) > 1 \\ -1 & \text{if } s(a,b) < -1 \\ s(a,b) & \text{otherwise.} \end{cases} \tag{5}$$

where $s(a,b)$ is calculated as

$$s(a,b) = \sum_{i\in a}\sum_{j\in b} w_{ij} \tag{6}$$

3.4.3 λ optimization

Model reduction inevitably causes information loss and reduced models behave less reliable. In order to somewhat compensate this problem, concepts of the reduced model did not share a common λ as in the original model but used different steepness parameters. Their values were identified by an evolutionary optimization method, the Big Bang – Big Crunch (BB-BC) [17] algorithm. The relatively low computational cost and high convergence speed made it a proper choice, moreover, the designer

have to set only a few parameters to work with it. Input concepts have constant states during simulation, thus they do not need parameter λ at all. Every output concept had its own λ_{oj} parameter. It would have been the best if every intermediate concept has its own λ value, but it would have increased the computational cost of the optimization disproportionately. As a trade-off, every intermediate concept had a common λ_i parameter. According to these changes, the inference formula was modified to Eq. 7:

$$A_i^{(t+1)} = f_i\left(\sum_{j=1}^{M} w_{ji}A_j^{(t)} + A_i^{(t)}\right), i \neq j \tag{7}$$

where f_i is a sigmoidal threshold function, and its slope is specified by λ_i. These parameters were identified by the BB-BC algorithm that minimized the objective function given by Eq. 8:

$$\sum_{i=1}^{m} a(s_i) \times \sum_{j=1}^{n} |o_{ij} - r_{ij}| \tag{8}$$

Here, m denotes the number of investigated *scenarios* (initial state vectors of simulations), s_i is the i$^{\text{th}}$ scenario, and $a(s_i)$ is a function defined by Eq. 9. It expresses the fact that we are only interested in simulations leading to fixed point attractors (FPs), because only they can be taken into account in a decision making process. n is the number of output concepts, o_{ij} and r_{ij} are the values of the j$^{\text{th}}$ output concepts at the end of the simulation of scenario i, respectively.

$$a(s_i) = \begin{cases} 1 \text{ if } s_i \text{ resulted in fixed point attractor in} \\ \quad \text{original and reduced models,} \\ 0 \text{ otherwise.} \end{cases} \tag{9}$$

3.4.4 Test environment and the method of investigation

Simulation error caused by model reduction was measured by comparing the final values of output concepts of the original and reduced models. Many synthetic models were generated to describe the effect of reduction on a statistical basis, thereafter some models found in the literature were also studied.

The original synthetic models were generated randomly, taken the following properties in consideration: in order to meet the definitions in Section 3.4.1, input concepts were forced to have at least one outgoing connection but no input connections, and similarly, output concepts had to have at least one input connection but no outgoing connections. Intermediate concepts must had at least one incoming and outgoing connections, and they were not allowed to form islands of concepts. With other words, all intermediate concepts were on at least one path leading from an input to an output concept. This property was checked by Kruskal's well-known

Table 2 Model densities and corresponding ϵ values with linguistic terms.

No. of concepts	Density (%)	ϵ values		
		Low	Medium	High
20	10	0.06	0.08	0.10
20	20	0.10	0.11	0.13
20	30	0.13	0.15	0.16
20	40	0.16	0.17	0.18
20	50	0.18	0.20	0.21
20	60	0.21	0.22	0.24
20	80	0.24	0.26	0.28
30	10	0.06	0.07	0.08
30	20	0.11	0.12	0.14
30	30	0.14	0.16	0.18
30	40	0.18	0.19	0.20
30	50	0.20	0.21	0.22
30	60	0.22	0.24	0.26
30	80	0.26	0.28	0.30

minimum spanning tree algorithm [14]. Following Kosko's original idea, self-loops were not allowed in the generated models.

In order to imitate the diverse properties of real-life models, synthetic models with 20 and 30 concepts were generated. In the former case, there were 5 input and 3 output concepts, in the latter case 8 input and 4 output concepts were created. The density of the connection matrix may have an effect on the typical model behavior, thus models with 10, 20, 30, . . . , 60 and 80% densities were generated. The meaning of *density* was defined by Eq. 10, however, because in our specific case some items of the connection matrix have to be always zero (all diagonal items, columns of input and rows of output concepts).

$$d = 100\frac{n}{((c-i)(c-o)-(c-i-o))} \tag{10}$$

Here n stands for the number of non-zero matrix elements, c is the total number of concepts of which i are input and o are output concepts.

Simulations were performed with 25 model instances, thus finally 350 original models were involved in investigations. The common λ values of sigmoid threshold functions were also defined randomly for the test set. The simulations were started with 125 unique scenarios in case of all models.

Next, the reduction of models followed. Three ϵ values were defined for each model size–density pairs according to Table 2. The values were referenced by their linguistic terms (*low*, *medium* and *high*) instead of their real value, because the same parameter value causes different effect on models with different properties, and our goal was to cause comparable reduction rates.

The reduced models were created in four versions according to the various metrics applied in *getWeight* versions described in Section 3.4.2.

Due to the 2 model sizes, 7 densities, 4 weight calculation methods, 3 different reduction rates (ϵ) and 25 instances with the same properties altogether 4200 reduced model instances were obtained.

In the first stage, reduced models inherited the λ parameters of their corresponding parents. In the seconds step, however, λ-optimized models were created and not only the error caused by reduction with various ϵ parameters but the error after optimization were also determined.

If a simulation led to a FP at both the original and the corresponding reduced model, the error caused by reduction in that specific case was calculated by Eq. 11:

$$J_1 = \frac{\sum_{j=1}^{n} |o_j - r_j|}{n} \tag{11}$$

where n is the number of output concepts, o_j and r_j are the final activation values of the j$^{\text{th}}$ output concepts in the original and reduced models, respectively. This is a scenario-dependent error measure. In contrast, J_2 characterizes the model instance independently from its individual instances.

$$J_2 = \frac{\sum_{i=1}^{m} J_1(i)}{m} \tag{12}$$

Here m denotes the number of scenarios leading to FPs. The minimum, lower quartile, median, upper quartile maximum, arithmetic mean and standard deviation of J_2 errors were also calculated for reduced models with and without λ optimization. These error measures plausibly describe the information loss caused by reduction.

Acknowledgements If you want to include acknowledgments of assistance and the like at the end of an individual chapter please use the `acknowledgement` environment – it will automatically be rendered in line with the preferred layout.

References

1. Buruzs Adrienn. *Fenntartható regionális hulladékgazdálkodási rendszerek értékelése fuzzy kognitív térképpel*. PhD thesis, Doctoral School of Multidisciplinary Engineering Sciences (MMTDI), Széchenyi István University, Győr, Hungary, 2015.
2. Somayeh Alizadeh, Mehdi Ghazanfari, and Mohammad Fathian. Using data mining for learning and clustering fcm. *International journal of computational intelligence*, 4(2):118–125, 2008.
3. A Buruzs, RC Pozna, and LT Kóczy. Developing fuzzy cognitive maps for modeling regional waste management systems. In *3rd International Conference on Soft Computing Technology in Civil, Structural and Environmental Engineering, CSC 2013*. Civil-Comp Press, 2013.
4. Adrienn Buruzs, Miklós F Hatwágner, László T Kóczy, et al. Modeling integrated sustainable waste management systems by fuzzy cognitive maps and the system of systems concept. *Czasopismo Techniczne*, 2013(Automatyka Zeszyt 3-AC (11) 2013):93–110, 2013.
5. Adrienn Buruzs, Miklós F Hatwágner, RC Pozna, and László T Kóczy. Advanced learning of fuzzy cognitive maps of waste management by bacterial algorithm. In *2013 Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS)*, pages 890–895. IEEE, 2013.

6. A. Gabus and E. Fontela. Perceptions of the world problematique: Communication procedure, communicating with those bearing collective responsibility (dematel report no.1). Technical report, Battelle Geneva Research Centre, Geneva, Switzerland, 1973.
7. M. F. Hatwagner and L. T. Koczy. Parameterization and concept optimization of fcm models. In *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8, Istanbul, aug 2015. IEEE.
8. Miklós F Hatwágner, Adrienn Buruzs, Péter Földesi, and László T Kóczy. Strategic decision support in waste management systems by state reduction in fcm models. In *International Conference on Neural Information Processing*, pages 447–457. Springer, 2014.
9. Miklós Ferenc Hatwágner, Adrienn Buruzs, Péter Földesi, and László Tamás Kóczy. A new state reduction approach for fuzzy cognitive map with case studies for waste management systems. In *Computational Intelligence in Information Systems*, pages 119–127. Springer, 2015.
10. Miklós Ferenc Hatwágner, Engin Yesil, M Furkan Dodurka, Elpiniki Papageorgiou, Leon Urbas, and László T Kóczy. Two-stage learning based fuzzy cognitive maps reduction approach. *IEEE Transactions on Fuzzy Systems*, 26(5):2938–2952, 2018.
11. W. Homenda, A. Jastrzebska, and W. Pedrycz. *Computer Information Systems and Industrial Management: 13th IFIP TC8 International Conference, CISIM 2014, Ho Chi Minh City, Vietnam, November 5-7, 2014. Proceedings*, chapter Time Series Modeling with Fuzzy Cognitive Maps: Simplification Strategies, pages 409–420. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.
12. G. J. Klir and B. Yuan. *FUZZY SETS AND FUZZY LOGIC – Theory and Applications*. Prentice Hall, USA, 1995.
13. Z. Kohavi and N. K. Jha. *Switching and Finite Automata Theory*. Cambridge University Press, third edition, 2009.
14. Joseph B Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.
15. Gonzalo Nápoles, Isel Grau, Rafael Bello, and Ricardo Grau. Two-steps learning of fuzzy cognitive maps for prediction and knowledge discovery on the hiv-1 drug resistance. *Expert Systems with Applications*, 41(3):821 – 830, 2014. Methods and Applications of Artificial and Computational Intelligence.
16. Elpiniki I Papageorgiou, Miklós F Hatwágner, Adrienn Buruzs, and László T Kóczy. A concept reduction approach for fuzzy cognitive map models in decision making and management. *Neurocomputing*, 232:16–33, 2017.
17. Engin Yesil and Leon Urbas. Big bang-big crunch learning method for fuzzy cognitive maps. *World Academy of Science, Engineering & Technology*, 71:816–825, 2010.