MAGNUM: TACKLING HIGH-DIMENSIONAL STRUCTURES WITH SELF-ORGANIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

A big challenge in dealing with real-world problems is scalability. In fact, this is partially the reason behind the success of deep learning over other learning paradigms. Here, we tackle the scalability of a novel learning paradigm proposed in 2021 based solely on self-organizing principles. This paradigm consists of only dynamical equations which self-organize with the input to create attractor-repeller points that are related to the patterns found in data. To achieve scalability for such a system, we propose the Magnum algorithm, which utilizes many self-organizing subsystems (SubSigma) each with subsets of the problem's variables. The main idea is that by merging SubSigmas, Magnum builds over time a variable correlation by consensus, capable of accurately predicting the structure of large groups of variables. Experiments show that Magnum surpasses or ties with other unsupervised algorithms in all of the high-dimensional chunking problems, each with distinct types of shapes and structural features. Moreover, SubSigma alone outperforms or ties with other unsupervised algorithms in six out of seven basic chunking problems. Thus, this work sheds light on how self-organization learning paradigms can be scaled up to deal with high-dimensional structures and compete with current learning paradigms.

1 INTRODUCTION

One of the biggest reasons behind the success of deep learning lies in its ability to tackle high-dimensional problems Mayer & Jacobsen (2020). Tasks that were very challenging in the past such as speech recognition Amodei et al. (2016), image classification Huang et al. (2017), and natural language processing Devlin et al. (2018); Peters et al. (2018) are now easily solved by algorithms based on the deep learning paradigm. However, deep neural networks themselves once were only a perceptron invented to mimic neurons and solve simple logic problems. An abundance of time and effort made them into the quality and accuracy we see today. However, they are not without the absence of problems such as adversarial samples Papernot et al. (2016); Xiao et al. (2018); Goodfellow et al. (2014); Su et al. (2019) which raise questions if other paradigms are not the only solution. To answer this question we must first further develop such paradigms to the point we can compare and this is one of the greatest motivations of this paper.

Researchers from neuroscience have long demonstrated humans' ability to detect patterns from their surrounding environment in the form of time-series data without the need of supervision. This is partly achieved by utilizing chunking as a learning strategy and is proven to carry an important role in a diverse range of cognitive functions Ramkumar et al. (2016). That being mentioned, the first self-organizing algorithm that could learn structures of time-series data in an unsupervised way and without any optimization function was recently proposed in 2021, which is named SyncMap Vargas & Asabuki (2021). It utilizes self-organizing principles to map temporal correlations into spatial correlations. In fact, the spatial correlations learned are so accurate that it was shown to surpass most deep learning algorithms in their first development. Interestingly, the system of dynamical equations that are at the heart of SyncMap is dynamical and continual in nature, such that changes in the problem can be easily captured and learned by the algorithm. In other words, adaptation becomes an inherent property emergent from the algorithm. Mathematically speaking, SyncMap is based on the principle of positive and negative feedback loops that can allow for the emergence of attractor-repeller pairs for each of the patterns present in the input. Having said that, we show here that SyncMap is incapable of dealing with high-dimensional problems. To achieve scalabil-

ity, we propose both a modified learning algorithm and a consensus-building framework for self-organizing systems called Magnum. Essentially, Magnum works by creating and merging multiple self-organizing subsystems. It creates low-dimension partitions of the problem and assigns them to these subsystems which process and give a response back. These subsystems are called SubSigma, which overcome the limitation of SyncMap by introducing a better dynamical equation and intrinsic clustering method.

To summarize, this paper makes the following contributions, (i) **State-of-the-art result**: The proposed algorithm, Magnum, outperforms other unsupervised methods in 7 out of 8 tests. Moreover, its subsystem, SubSigma, surpasses others in 6 of 7 basic chunking tests; (ii) **Learning based on Dynamical Equations**: This paper investigates a novel self-organizing approach to learning without loss functions that is surprisingly precise; (iii) **High-dimensional Chunking with Self-Organization Alone**: To achieve state-of-the-art results with self-organization alone, Magnum employs a variable correlation by consensus approach that significantly increase its performance in various types of high-dimensional problems.

2 Related Work

Chunking. In neuroscience, chunking is a mechanism present in the brain which was shown important as a cost-effective learning strategy that attains compact representation of sequences Ramkumar et al. (2016). Chunking seems to be a crucial stepping stone in which other high-order functions can be attained, such as motor-skill learning Yokoi & Diedrichsen (2019) and language acquisition Buiatti et al. (2009), Saffran & Wilson (2003). Schapiro et al. Schapiro et al. (2013) show instead of repetitive patterns, sequences of co-occurring structures are more likely to be chunked. In 2020, a biology-inspired sequence learning model Minimization of Regularised Information Loss (MRIL) Asabuki & Fukai (2020) was proposed. It is capable of solving various chunking tasks including chunking with probabilistic transitions. Recently, in 2021, the first bio-inspired algorithm based on self-organizing alone was proposed (called SyncMap Vargas & Asabuki (2021)). SyncMap only uses dynamical equations without any type of optimization. Albeit the novel learning paradigm, it outperformed MRIL and all other algorithms in most of the tasks. In fact, it was shown to be able to adapt to changes in the environment without problems.

Word Embedding. Word embedding algorithms transform paragraphs and words into vectors, which are commonly used in natural language processing (NLP) Khattak et al. (2019); Mikolov et al. (2013). Some of them are contextualized word embeddings Peters et al. (2018); Devlin et al. (2018) or are enriched with information specific to NLP Bojanowski et al. (2017). Therefore, co-occurrence matrix-based GloVe Pennington et al. (2014) and prediction-based word2vec Mikolov et al. (2013) can be used for general problems and therefore applied to uncover chunks as well. In this paper, we consider a variation of word2vec as one of the baselines.

Complex Network and Community. Community detection is often used to understand the structure of large and complex networks. It is used in different application from social networks Tang et al. (2015) to biochemical networks Ito et al. (2001). Based on its importance, Clauset et al. Clauset et al. (2004) attempted to find communities in a large complex networks by maximizing *modularity*, a metric that measures the quality of the structure Newman (2006). Although this paper focuses on sequence chunking, we cannot ignore their capability in community chunks detection. Hence, we transform the input sequence into a complex network. Doing so allows us to experiment community detection algorithm with sequential data. Here, Clauset-Newman-Moore Modularity Maximization algorithm is chosen as a baseline Clauset et al. (2004); Newman & Girvan (2004).

Unsupervised Learning for Sequences. Unsupervised learning for sequences is able to predict future input using extracted features Wu et al. (2018); Lei et al. (2019); Hyvarinen & Morioka (2016); Mikolov et al. (2013); Clark et al. (2019). Although these features are useful for the classification of sequences, they do not uncover the chunks present. Therefore, these methods are complementary rather than competing with methods that tackle the chunking problem.

3 Background

To lay the groundwork for the proposed method, we introduce the continual general chunking problem (CGCP) and the self-organizing system with the best results in CGCP (SyncMap Vasconcel-

los Vargas & Asabuki (2021)). Subsequently, the limitation of SyncMap is discussed, especially in relation to high-dimensional CGCP problems.

3.1 CONTINUAL GENERAL CHUNKING PROBLEMS

First defined by Vargas and Asabuki Vargas & Asabuki (2021), the Continual General Chunking Problem (CGCP) describes the problem of extracting co-occurring states from a time series of which the underlying generation process changes depending on the task as well as throughout the task. Its input sequence contains the state of variables (a variable that can be considered as an object or event occurring), transitioning by a first-order Markov chain. The transition matrix T is defined as: $T \in (0,1)^{|state| \times |state|}$, where $state$ is the set of all possible state variables occur in the sequence. $T_{state_i, state_j} = Pr[s_{t+1} = state_j | s_t = state_i]$ is the transition probability from $state_i$ to $state_j$. All the tasks in CGCP comprise only probabilistic chunks and fixed chunks. Fig. 1 shows the detail of these two chunks.

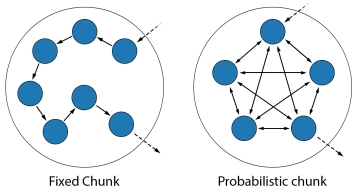

Figure 1: **Illustration of fixed and probabilistic chunks.** Big circles are considered chunks. Small circles in a chunk indicate state variables. Dashed arrows represent the transition from a chunk or the other way around, depending on the arrow's direction. Solid arrows are the transitioning direction within a chunk. (Left) Fixed chunks only have one-way transitions. (Right) Probabilistic chunks contain more than one possible transition and for each a given transition probability.

States in fixed chunk contain a static transition, which satisfies: $T_{state_i, state_j} = 1$, where state denoted as $state_i$ always transits to another predetermined state $state_j$. Oppositely, all states in probabilistic chunk transit to each other with the same probability. The transition matrix is defined as:

$$T_{state_i, state_j} = \begin{cases} \frac{1}{(|V(state_i)|-1)} & state_j \in V(state_i) \\ 0, & else \end{cases} \tag{1}$$

where $V(state_i)$ represents the states in the same probabilistic chunk with $state_i$, $|V(state_i)|$ be the cardinality of $V(state_i)$. In the original work where the CGCP was originally defined, experiments only consider problems that are of a small number of variables and chunks. This leads to the question of how large number of variables and chunks will affect SyncMap's performance. Hence, we expand the complexity of the original problem and take into account the high-dimensional versions of the CGCP.

3.2 SYNCMAP

In 2021, Vargas and Asabuki Vargas & Asabuki (2021) proposed a self-organizing chunking algorithm called SyncMap, which identifies chunks from sequence data by creating a dynamic map that preserves the temporal correlation between variables. The workflow of SyncMap can be divided into three parts: (i) input encoding, (ii) dynamics, and (iii) clustering phase.

Input encoding. Input encoding is a data pre-processing method that encodes a sequence of states into a series of decaying signals. Consider the input sequence $S = (s_0, s_1, s_2, \cdots, s_t, \cdots, s_\tau)$, where t is the time step, τ is the total time step and $s_t \in state$. With that, input encoding is modelled as an exponentially decaying vector X, where $X \in \mathbb{R}^{|state| \times \tau_{decay}}$ and $\tau_{decay} = decay \times \tau$:

$$X_{s_t, t} = \begin{cases} e^{-0.1*(t-ta)}, & ta < m_c * decay \\ 0, & otherwise \end{cases} \tag{2}$$

where ta is the most recent state transition to state to s_t, m_c is the state memory constant which equals to two, $m_c/decay$ is the most recent state transition to state s_{t+1}. In other words, since

each state transition happens every $decay$ steps, variables with a time of activation greater than $m_c \times decay$ steps are set to 0.

Dynamics. With the encoded input, multiple weights $w_{i,t}$ are randomly initialized $w_{i,t} \in \mathbb{R}^d$ in a dynamic map with d dimension. They are also linked to an input $x_{i,t}$, creating a tuple $(x_{i,t}, w_{i,t})$. Every time a new input x_t is presented, each of its constituents $x_{i,t}$ are divided into two sets: (i) Positive set PS_t and (ii) Negative set NS_t. Inputs with value greater than 0.1 are assigned to PS_t, where $PS_t = \{i | x_{i,t} > 0.1\}$. Otherwise, inputs are assigned to NS_t, where $NS_t = \{i | x_{i,t} \leq 0.1\}$. If the cardinality of both sets $|PS_t| > 1$ and $|NS_t| > 1$, we compute the centroids for both PS_t and NS_t, respectively $cp_t = \frac{\sum_{i \in PS_t} w_{i,t}}{|PS|}$ and $cn_t = \frac{\sum_{i \in NS_t} w_{i,t}}{|NS|}$. Otherwise, if the cardinality of both positive and negative sets is not greater than one, $PS_t \leq 1$ and $NS_t \leq 1$, weight update is skipped in that particular time step. Followed by the computation of cp_t and cn_t, the position of weight $w_{i,t}$ of its corresponding input is updated with:

$$\phi(i,t) = \begin{cases} 1, & i \in PS_t \\ 0, & i \in NS_t \end{cases} \tag{3}$$

$$w_{i,t+1} = w_{i,t} + \alpha[\phi(i,t)\Delta w_p + (1 - \phi(i,t))\Delta w_n] \tag{4}$$

where α is the learning rate, $\Delta w_p = \frac{(cp_t - w_{i,t})}{||w_{i,t} - cp_t||}$ and $\Delta w_n = \frac{(cn_t - w_{i,t})}{||w_{i,t} - cn_t||}$. At the end of each time step, all values of updated weights are normalized to a bounded space.

Clustering Phase. Once the dynamics are repeated for all inputs, clustering is applied over the dynamic map to identify the chunks. DBSCAN Schubert et al. (2017) is used here. The authors justified the use of DBSCAN based on its automatic discovery of the number of chunks.

3.3 Limitations of SyncMap

Deadlock. We observed that in certain situations, the current equations of SyncMap can enter in a type of deadlock (Fig. 4). For instance, correlated variables that belong to the same chunk cannot attract toward each other because other variables of a different chunk are blocking their path. Vice versa, the latter variables are also difficult to move when variables that belong to a different chunk are around them. We discuss the solution to this problem using SubSigma's dynamic in the later section.

Incorrect Clustering. Moreover, the clustering algorithm, DBSCAN, used in SyncMap can induce errors in clustering when dealing with varying densities or high-dimensional data. This issue can not be prevented during the dynamic step, as chunking is performed separately on top of the learned map (Fig. 4). Therefore, we seek a method that can perform chunking intrinsically without relying on a separate clustering algorithm, which is later shown in section 4.2.

4 SubSigma

To address the limitations of SycnMap, we proposed SubSigma, which introduces a novel dynamical equations (SubSigma dynamics) and chunking methods (meta-variables).

4.1 SubSigma Dynamics

Inspired by SyncMap, SubSigma follows the general workflow of SyncMap specified in section 3.2, unless detailed in this section. SubSigma introduces the concept analogous to physical bodies. It considers inertia $I_{i,t+1}$ as the potential moving trend which contain the current acceleration $(\Delta a_p, \Delta a_n)$ and previous inertia $I_{i,t}$ (i.e., memory of the previous acceleration in weight space). Specifically, the update of inertia is as follows:

$$I_{i,t+1} = \theta I_{i,t} + [\phi(i,t)\Delta a_p + (1 - \phi(i,t))\Delta a_n] \tag{5}$$

where accelerations, the change in weight over one step, are denoted as $\Delta a_p = \left[3\left(\frac{cp_t - w_i}{||w_{i,t} - cp_t||}\right) - \left(\frac{w_i - cn_t}{||w_{i,t} - cn_t||}\right)\right]$ and $\Delta a_n = \left[3\left(\frac{cn_t - w_i}{||w_{i,t} - cn_t||}\right) - \left(\frac{w_i - cp_t}{||w_{i,t} - cp_t||}\right)\right]$, θ is the inertia decaying coefficient

and $\phi(i,t)$ is the switch function (Eq. 3). Each time step, weight is updated as follows, with α be the learning rate:

$$w_{i,t+1} = w_{i,t} + \alpha I_{i,t+1} \tag{6}$$

Fig. 2 shows the comparison between the dynamic of SyncMap and SubSigma, which explain why SubSigma does not suffer from the deadlock issue.

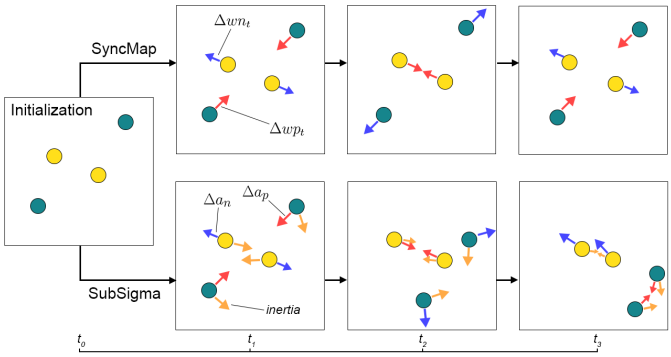

Figure 2: **Comparison of the dynamic between SubSigma and SyncMap.** t_0 shows the initialization of four weights within a weight space, separated into two chunks, i.e., 'yellow' and 'green' color chunks. In SyncMap dynamic, the correlated weights move directly toward a common center point cp but are repelled by the other chunks. This cause correlated weights to fail in forming chunk as the system get caught in an equilibrium. Hence, the issue of deadlock. To address such an issue, inertia is added into the system to create a strong-enough acceleration in any direction that makes the deadlock unlikely to occur. The presence of additional forces in Δa_p and Δa_n also added extra degrees of freedom to cancel out the equilibrium. Consequently, at the end of the time step, SyncMap accuracy is affected by the issue of deadlock; While SubSigma does not suffer from the same issue.

4.2 META-VARIABLES

Without relying on the clustering method, SubSigma can create meta-variables recursively to identify chunks. At the end of each sequence of inputs, SubSigma joins the two closest weights (w_i and w_j) into a newer weight $w_{ij} = \frac{w_i + w_j}{2}$, composing a meta-variable. The generated meta-variable behaves as a single variable and can join with other variables/meta-variables in the next iterations of the algorithm. This simple procedure in a subset of variables is equivalent to clustering in the long run when combined with other subsets.

Stopping Criteria. Because meta-variable will be produced recursively, we need to set stopping criteria to terminate further formation of meta-variables and eventually produces chunks as an output. Based on stopping criteria, the model decides whether to (i) feed the processed weights back to the dynamic step again or (ii) terminate its procedure, outputting the set of combined weights to Magnum. Such stopping criteria are defined as:

$$\begin{cases} terminate & \frac{minD(WS)}{meanD(WS)} > 0.3 \text{ or } |WS| < 4 \\ continue & otherwise \end{cases} \tag{7}$$

where WS is the set of weights in the SubSigma, $minD(WS)$ and $meanD(WS)$ are the minimum distance and mean distance of weights respectively, and $|WS|$ refers to the number of variables inside SubSigma. Essentially, stopping criteria stops the algorithm when either (i) all meta-variables are more or less equally spaced in the dynamic map ($\frac{minD(WS)}{meanD(WS)} > 0.3$) or (ii) the number of meta-variables has decreased to only three ($|WS| < 4$).

5 MAGNUM

Imprecision is another issue especially when tackling high-dimensional problems, in addition to the issues of the DBSCAN and the equation dynamics mentioned in the end section about SyncMap. In

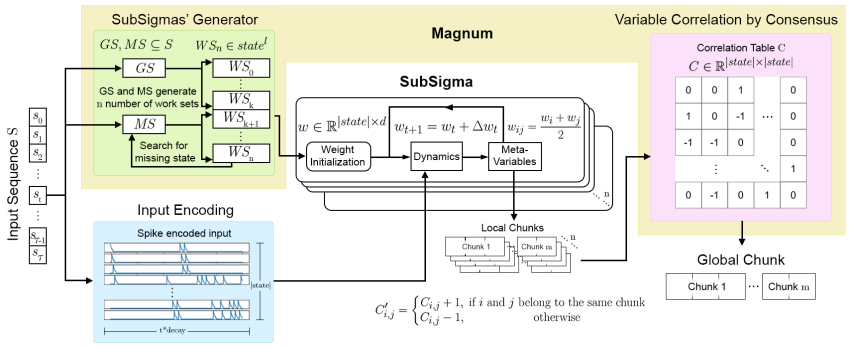

Figure 3: **General workflow of Magnum.** Magnum begins by sampling state variables from the input sequence and dividing them into k numbers of subsets WS. These subsets are assigned the same number of SubSigmas to yield multiple sets of local chunks, allowing high-dimensional state variables to be analyzed in multiple low-dimensional problems. After each SubSigma learns the local chunks, Magnum finds the consensus of SubSigmas' variable correlation to determine the prediction of global chunks.

high-dimensional problems, any imprecision on the lower level can cascade into inaccurate results. On that account, we propose Magnum, a framework that solves most of the above using a population of self-organizing subsystems (SubSigma). Specifically, Magnum creates several SubSigmas built from a subset of the entire number of variables. To join the local variable correlation found by SubSigmas, Magnum creates a variable correlation consensus. Note that SyncMap could be employed in place of SubSigma, however for the reasons mentioned before and evidenced by experiments, it would not confer an accurate system.

5.1 SubSigmas' Generator

Multiple subsequences of state variables are sampled from the input sequence and then the process is repeated multiple times, creating many SubSigmas with a different set of variables. This essentially helps to split high-dimensional state variables into a lower-dimensional one, which becomes easier to solve by SubSigma and reduce the variance of the final outcome. In detail, consider a raw input sequence S. We first randomly extract k subsequences from S, where the first state of the subsequence is defined as $GS_i, i = 1...k$. Starting from GS_i, a searching process is applied, so that each subsequence is set to contain l different state variables. We call these k subsequences the work sets $WS_i, i = 1,...,k$; note that WS_i can have arbitrary length. Some state variables, however, may not be taken into account any of WS_i. To handle these missing states, Magnum searches the first occurrence of that missing state from the raw sequence S, starting from $MS_i, i = k{+}1,...,n$ (similarly to GS_i above), and then generates additional work sets WS_{k+1} to WS_n, by using the same generation process as to create the first k of WS_i. To this end, a total of n work sets are generated, which will be assigned to n SubSigmas. Each assigned SubSigmas initialize weights and variable correlation only to the state variables from the corresponding WS_i.

5.2 Variable Correlation by Consensus

By taking consensus between multiple SubSigmas, the final variable correlation is found. In essence, it works by creating a matrix of $C \in \mathbb{R}^{|state|\times|state|}$ based on SubSigmas' responses. Specifically, for every pair of variables (i, j) within the same meta-variable, $C_{i,j}$ is incremented, otherwise, $C_{i,j}$ is decremented. Once matrix C has aggregated all the response from SubSigmas, Magnum chunks the variables in where $C_{i,j} > 0$.

6 Experiments

The experiments are categorized into two major categories: (i) we evaluate the capabilities of SubSigma in CGCPs [problems originates from Vargas & Asabuki (2021)] to understand the improve-

ment of dynamics compared to SyncMap and its performance compared to other baselines; (ii) We evaluate the performance of Magnum integrates with SubSigma in high-dimensional CGCPs. We discuss the setup and empirical results (Detail of statistical t-test are presented in the Appendix. D) of both categories in the following subsection.

6.1 BASELINES

We compared the proposed method with four baselines: (i) Skip-gram Word2vec Mikolov et al. (2013), a deep learning-based model that learns the associations of symbols and words embedded in a latent space; (ii) MRIL Asabuki & Fukai (2020), a self-supervised cognitive model that detects spatio-temporal correlation and repetitive patterns (chunks) solitary; and (iii) Modularity Maximization Clauset et al. (2004); Newman & Girvan (2004), a community detection algorithm that find communities in a large complex networks by maximizing the *modularity*. (iv) SyncMap Vargas & Asabuki (2021), a self-organizing system that learns to identify chunks from sequential data. The results are evaluated using the Normalized Mutual Information (NMI) score as a metric. Data preprocessing was applied to render inputs fair and suitable for the above baselines (see section 6.2)

6.2 CGCPS

Here, we test the capabilities of the proposed SubSigma with the experiments originating from the work that defined CGCPs and SyncMap. the CGCPs experiments utilize four types of problems to measure chunking quality: (i) Two pure chunking problems, fixed chunk, and long chunk problems, which contain only fixed and probabilistic chunks respectively. (ii) A mixed problem is composed of the cross arrangement of fixed chunks and probabilistic chunks. (iii) Two overlapped problems, in which there exist chunks that share some variables. (iv) Two real-world scenarios are two variations of the first-order Markov model of theme transitions for humpback whales' song types (Fig. 5). All the problems here contain less than 30 state variables, therefore can not be considered as high-dimensional CGCPs.

Experimental setup. All problems generate a sequence with 100,000 samples of state variables each trial. To ensure a fair comparison when analyzing the effectiveness of dynamical equations, both SyncMap and SubSigma used DBSCAN during the clustering phase. We compare SubSigma with the four baselines mentioned above. The detail of the hyperparameters setting is given below. Regarding SubSigma, we set the map dimension d to 3, the inertia decay rate θ to 0.999, and the learning rate α to $0.001*|state|$, with $|state|$ being the number of states. SyncMap has the same map dimension and learning rate as SubSigma. Further, we considered an MRIL model with five output neurons, training under a learning rate of $1e-3$. To evaluate a word embedding algorithm in the context of CGCP, we set up a skip-gram Word2vec model as one of the baselines. The Word2vec model takes the shape of a Variational Autoencoder, with a latent dimension of 3 and a Softmax output layer. The model is trained for 10 epochs with a batch size of 64, using a learning rate of $1e{-}3$ and a mean squared error as loss. A window of 100 steps is used to calculate the output probability of skip-gram, equivalent to 10 state variable transitions. Modularity maximization algorithm can only operate under the premise of a network, hence, we transform the input sequence into a weighted directed graph via an adjacency matrix, which can be constructed simply by recording the state-to-state transition of the input.

Results. CGCP results in Table 1 shows that SubSigma obtains overall optimal scores in four problems. It outperforms other baselines in 5 out of 7 problems. Moreover, SubSigma is the only algorithm that maintains overall high NMI scores as the complexity of the problems increases. It should be noted that SubSigma performs significantly better in Overlap1 and Overlap2 problems relative to other baselines, showing the generalization ability and the effectiveness when dealing with different settings of CGCP and real-world scenarios.

6.3 HIGH-DIMENSIONAL CGCPS

In high-dimensional CGCP, the number of state variables and chunks is directly proportional to CGCP, with 10 times the amount used in the previous CGCP experiment. To evaluate and compare algorithms regarding their ability to tackle high-dimensional CGCPs, some synthetic problems are

Table 1: **Result on CGCP.** This table shows the NMI comparison over SubSigma, SyncMap, Modularity Maximization (MMAX), Word2vec, and MRIL in basic CGCP. The best mean and statistically similar results are in bold. (Data are mean±s.t.d.)

Baselines	Fixed chunks	Long chunks	Mixed chunks	Overlap1	Overlap2	Real world1	Real world2
MMax	**1.00±0.00**	0.63±0.00	0.77±0.00	0.68±0.00	0.46±0.00	0.40±0.02	0.48±0.00
Word2vec	**1.00±0.00**	0.63±0.20	0.80±0.09	0.27±0.33	0.11±0.11	0.28±0.06	**0.79±0.06**
MRIL	0.93±0.12	0.83±0.17	0.86±0.11	0.00±0.00	0.00±0.00	0.39±0.07	0.52±0.06
SyncMap	0.96±0.8	0.82±0.13	**0.96±0.07**	0.56±0.25	0.65±0.09	0.35±0.14	0.58±0.08
SubSigma	**1.00±0.00**	**0.98±0.06**	**0.98±0.04**	**0.98±0.07**	**0.78±0.04**	**0.42±0.04**	0.39±0.07

created to have a diverse set of features. Experiments are categorized into four categories: multiple chunks, big chunks, balanced chunks and imbalanced chunks; each of which contains two types of problems (mixed and fixed problems), resulting in a total of eight problems. Note that the chunk size, chunks composition, and the number of chunks are task-specific, refers to Fig. 6 for a detailed illustration of the environment's setting. The idea behind the designed problem categories is to encompass diverse features in the experiments. For instance, the multiple chunks category's goal is to evaluate how algorithms will respond to a situation with a large number of chunks, yet only contains a small number of state variables. Contrarily, the big chunks category considers a case of a small number of chunks with a larger chunk size. Besides, the balanced chunks category accounts for equal numbers of chunks and chunk sizes. Oppositely, imbalance chunks category considers an environment condition that closely resembles the real world, where the frequency of input events vary drastically.

Table 2: **Results on high-dimensional CGCP.** This table shows the NMI comparison over SubSigma, SyncMap, Modularity Maximization (MMAX), Word2vec, MRIL, Magnum with SyncMap and DBSCAN (Mag w/SM/D), Magnum with SubSigma and DBSCAN (Mag w/SS/D), and Magnum with SubSigma and Meta-variables (Mag w/SS/M) in high-dimensional CGCP. The best mean and statistically similar results are in bold. (Data are mean±s.t.d.)

	Multiple Chunks		Big Chunks		Balanced Chunks		Imbalanced Chunks	
Baselines	Mixed	Fixed	Mixed	Fixed	Mixed	Fixed	Mixed	Fixed
MMax	0.79±0.00	0.87±0.00	0.40±0.02	**0.67±0.00**	0.70±0.02	0.81±0.00	0.66±0.00	**0.79±0.00**
Word2vec	0.76±0.03	0.65±0.03	0.78±0.15	0.63±0.05	0.69±0.02	0.78±0.08	0.69±0.08	0.70±0.13
MRIL	0.56±0.01	0.45±0.01	0.01±0.00	0.16±0.05	0.24±0.06	0.28±0.04	0.11±0.12	0.26±0.03
SyncMap	0.39±0.05	0.59±0.02	**0.85±0.03**	0.22±0.04	0.70±0.02	0.38±0.12	0.59±0.02	0.37±0.02
SubSigma	0.83±0.02	0.57±0.04	0.27±0.06	0.20±0.02	0.35±0.04	0.47±0.03	0.31±0.05	0.36±0.02
Mag w/SM/D	0.93±0.02	0.72±0.08	0.15±0.29	0.53±0.02	0.66±0.09	0.70±0.04	0.20±0.21	0.62±0.01
Mag w/SS/D	**0.96±0.01**	0.92±0.02	0.58±0.20	0.60±0.02	0.69±0.03	0.80±0.03	0.69±0.13	**0.73±0.04**
Mag w/SS/M	0.93±0.01	**0.94±0.02**	**0.79±0.21**	**0.67±0.02**	**0.94±0.02**	**0.85±0.02**	**0.87±0.04**	**0.78±0.04**

Experimental setup. In all experiments, models are presented with $500,000$ samples generated from the problem. Note that the chunk size, chunk composition, and the number of chunks are task-specific, details and illustrations of the problem's setting are in the Appendix. Other than the four baselines mentioned previously, we also experiment with SubSigma and three different versions of Magnum. Regarding Magnum's parameter, we choose the length of the work set to be $l = 10$ and the initial number of the work sets to be $k = 100$. All the parameters of SubSigma, SyncMap, MRIL, Word2vec, and Modularity Maximization are identical to the previous experimental setup.

Results. The empirical results of high-dimensional CGCP are shown in Table 2. Despite excelling in CGCPs, SubSigma alone is insufficient to tackle high-dimensional CGCPs. By integrating SubSigmas/meta-variables, Magnum outperforms or performs equally to the best algorithm in all tests. it achieved the sole best performance in 5 out of 8 tasks. Other than that, Magnum with SubSigma/DBSCAN appears to perform better than Magnum with SyncMap/DBSCAN. The ablation study on different versions of Magnum suggests that Magnum with SubSigma as self-organizing subsystems performs better than that with SyncMap. In terms of the clustering phase, meta-variables are a better chunking approach compared to DBSCAN by solving the issue of incorrect clustering.

7 DISCUSSION

Regarding the performance of Magnum, high-dimensional problems are less of an issue for Magnum because of the use of multiple SubSigmas, each one working on an easier low-dimensional problem. A caveat of SubSigma in comparison to SyncMap is that although it is more reliable in relation to its dynamics, the stopping criteria used inside it might be deleterious. In other words, SubSigma might have responses that are too early ($|WS| < 4$) conferring a less precise response when compared with SyncMap. This, however, is remediated by how Magnum works, i.e., aggregating partial results with inconsistencies that are averaged out into a consistent final one. The above is true if there are enough correct results to average out incorrect ones. However, if the sampling of some regions is less than sufficient, it is possible that the final result will not be averaged out but contain the inconsistencies of the partial responses from SubSigmas.

Modularity Maximization is a deterministic algorithm where the standard deviation should be zero if the transition matrix is precisely estimated. However, as shown in the results, this is the only time when the result has a non-zero standard deviation. Therefore, a possible reason for the worse results might be that the transition matrix did not have time to be precisely approximated from the random walk. It can be observed that the NMI score decreases when the size of chunks increases, showing that modularity maximization is good at detecting multiple small-size chunks.

Regarding SyncMap's performance, it performs poorly mostly on fixed problems. Fixed problems have variables that either can get stuck on deadlocks easily (given the relatively high repetition of patterns), have variables/chunks which are rarely visited, or both. However, it performs the best in mixed structure big chunks and second the best in balanced chunks mixed structure. This reveals that it is capable of identifying probabilistic chunks with high accuracy.

Regarding Word2vec, careful examination of the map learned by Word2vec reveals it is good at transferring the correlation from variables into a dispersed map. NMI scores are around 0.7 in most of the tests because it is hard to cluster chunks in such a dispersed map. Its consistency throughout reveals that although not accurate, it is one of the most reliable algorithms for high dimensions. Such a consistency might drop once accuracy is improved by any improvement in the method.

Regarding MRIL, a high variation of results was found in the experiment over time, suggesting that the algorithm suffers from instability. Increasing the number of outputs might further improve the results, but it is still not enough to find parameters. Moreover, the high-dimensional nature of the problems analyzed here makes little fluctuations or imprecision in the method even worse. One of the reasons lies in the fact that lateral inhibition can not be trained efficiently. Thus, for a large number of output neurons, it becomes impracticable.

8 CONCLUSION

In this paper, we proposed a framework that identifies chunks from high-dimensional sequences data. Magnum assigns subsets of large groups of the variable to numerous self-organizing subsystems (SubSigma) to create consensus upon aggregation. Such consensus allows for partial inaccurate responses to be merged into a consistent and accurate one. Specifically, SubSigma can be viewed as an enhancement of SyncMap by solving both the deadlock as well as removing the clustering algorithm. Overall, the empirical results showed that Magnum outperforms or ties with all other unsupervised algorithms in high-dimensional tasks. By solving complex high-dimensional problems, Magnum acts as a stepping-stone for self-organizing systems to further its application. However, our model required further validation based on real-world application. Currently, we are constrained by the limited amount of suitable datasets that satisfied our niche input requirement. Looking forward, we expect variations of Magnum should further increase the gap to other algorithms in high-dimensional continual general chunking problems. More experiments are required to validate real-world scenarios. We set our future directions to investigate more complex problems with dynamic, hierarchical, and causal relations. We also seek broader implications in the area of language processing to image/action recognition, as these areas involve input that can be serialized as high-dimensional sequence data and thus tackled with variations of the algorithm proposed here.

REFERENCES

Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pp. 173–182. PMLR, 2016.

Toshitake Asabuki and Tomoki Fukai. Somatodendritic consistency check for temporal feature segmentation. *Nature communications*, 11(1):1–13, 2020.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

Marco Buiatti, Marcela Peña, and Ghislaine Dehaene-Lambertz. Investigating the neural correlates of continuous speech computation with frequency-tagged neuroelectric responses. *Neuroimage*, 44(2):509–519, 2009.

David G Clark, Jesse A Livezey, and Kristofer E Bouchard. Unsupervised discovery of temporal structure in noisy data with dynamical components analysis. *arXiv preprint arXiv:1905.09944*, 2019.

Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Ellen C Garland, Luke Rendell, Luca Lamoni, M Michael Poole, and Michael J Noad. Song hybridization events during revolutionary song change provide insights into cultural transmission in humpback whales. *Proceedings of the National Academy of Sciences*, 114(30):7822–7829, 2017.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Aapo Hyvarinen and Hiroshi Morioka. Unsupervised feature extraction by time-contrastive learning and nonlinear ica. *Advances in Neural Information Processing Systems*, 29:3765–3773, 2016.

Takashi Ito, Tomoko Chiba, Ritsuko Ozawa, Mikio Yoshida, Masahira Hattori, and Yoshiyuki Sakaki. A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proceedings of the National Academy of Sciences*, 98(8):4569–4574, 2001. ISSN 0027-8424. doi: 10.1073/pnas.061034498. URL `https://www.pnas.org/content/98/8/4569`.

Faiza Khan Khattak, Serena Jeblee, Chloé Pou-Prom, Mohamed Abdalla, Christopher Meaney, and Frank Rudzicz. A survey of word embeddings for clinical text. *Journal of Biomedical Informatics: X*, 4:100057, 2019.

Qi Lei, Jinfeng Yi, Roman Vaculin, Lingfei Wu, and Inderjit S Dhillon. Similarity preserving representation learning for time series clustering. In *IJCAI*, volume 19, pp. 2845–2851, 2019.

Ruben Mayer and Hans-Arno Jacobsen. Scalable deep learning on distributed infrastructures: Challenges, techniques, and tools. *ACM Computing Surveys (CSUR)*, 53(1):1–37, 2020.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582, 2006.

Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS P)*, pp. 372–387. IEEE, 2016.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

Pavan Ramkumar, Daniel E. Acuna, Max Berniker, Scott T. Grafton, Robert S. Turner, and Konrad P. Kording. Chunking as the result of an efficiency computation trade-off. *Nature Communications*, 7(1):12176, Jul 2016. ISSN 2041-1723. doi: 10.1038/ncomms12176. URL https://doi.org/10.1038/ncomms12176.

Jenny R Saffran and Diana P Wilson. From syllables to syntax: multilevel statistical learning by 12-month-old infants. *Infancy*, 4(2):273–284, 2003.

Anna C Schapiro, Timothy T Rogers, Natalia I Cordova, Nicholas B Turk-Browne, and Matthew M Botvinick. Neural representations of events arise from temporal community structure. *Nature neuroscience*, 16(4):486–492, 2013.

Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, 42(3):1–21, 2017.

Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pp. 1067–1077, 2015.

Danilo Vasconcellos Vargas and Toshitake Asabuki. Continual general chunking problem and syncmap. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(11):10006–10014, May 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/17201.

Danilo Vasconcellos Vargas and Toshitake Asabuki. Continual general chunking problem and syncmap. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(11):10006–10014, May 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/17201.

Jia Wu, Weiru Zeng, and Fei Yan. Hierarchical temporal memory method for time-series-based anomaly detection. *Neurocomputing*, 273:535–546, 2018.

Chaowei Xiao, Bo Li, Jun yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 3905–3911. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi: 10.24963/ijcai.2018/543. URL https://doi.org/10.24963/ijcai.2018/543.

Atsushi Yokoi and Jörn Diedrichsen. Neural organization of hierarchical motor sequence representations in the human neocortex. *Neuron*, 103(6):1178–1190.e7, 2019. ISSN 0896-6273. doi: https://doi.org/10.1016/j.neuron.2019.06.017. URL https://www.sciencedirect.com/science/article/pii/S0896627319305677.

A LIMITATION OF SYNCMAP

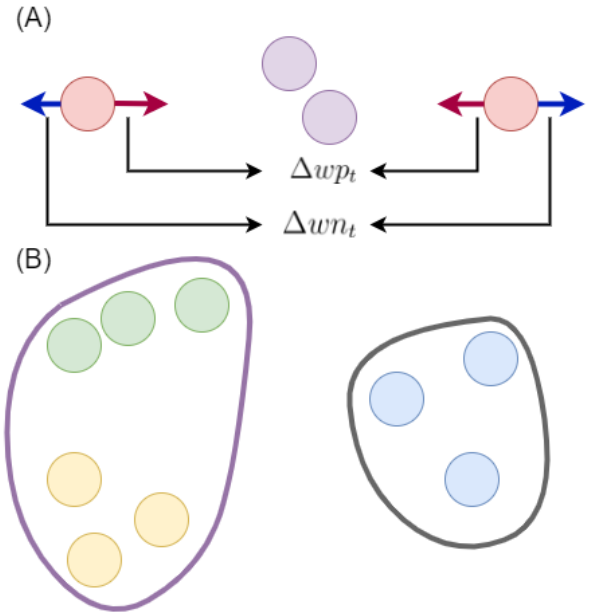

Figure 4: **SyncMap's Limitations.** (A) Deadlock: Purple points at the center will tend to stay together. However, this would block the two orange points that will be attracted towards the center (Δwp_t) while being repelled by the purple points (Δwn_t). (B) Clustering Miss: Beyond the dynamic, three different chunks tend to stay in each corner. However, DBSCAN chunks the points into different groups, which are in yellow and green.

B REAL WORLD SCENARIO IN CGCPS

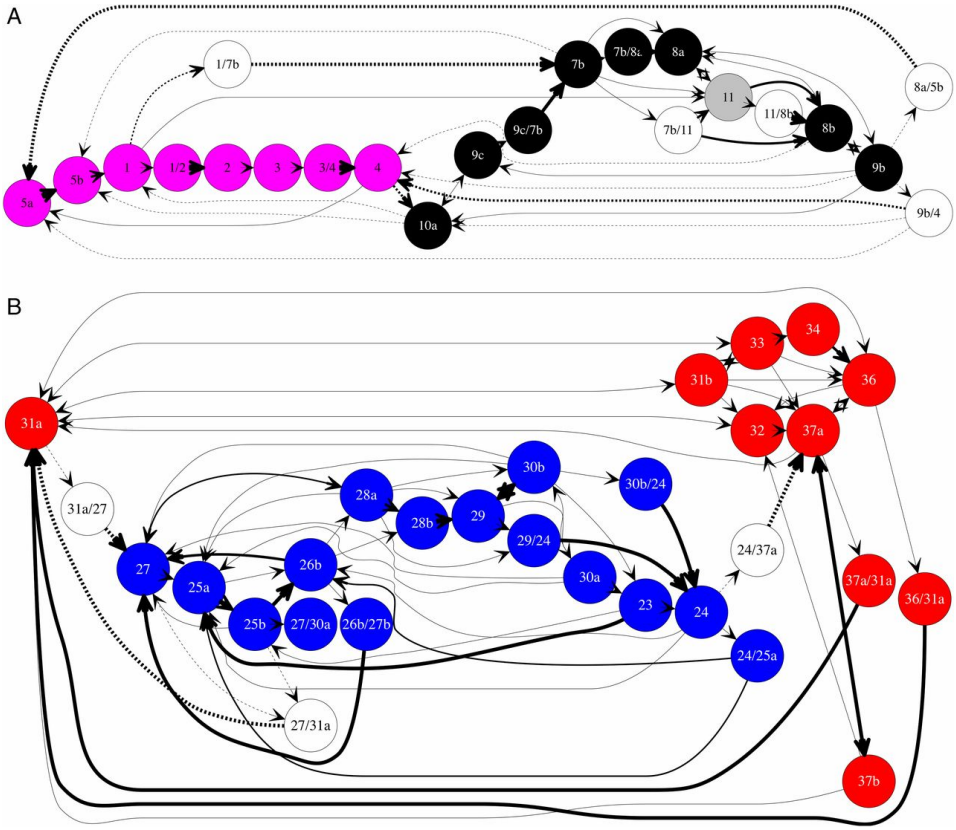

Figure 5: **Real World Scenario.** First-order Markov Chain of humpback whales' song types (Garland et al., 2017). A and B are real-world 1 and 2 respectively.

C HIGH DIMENSIONAL CGCPS

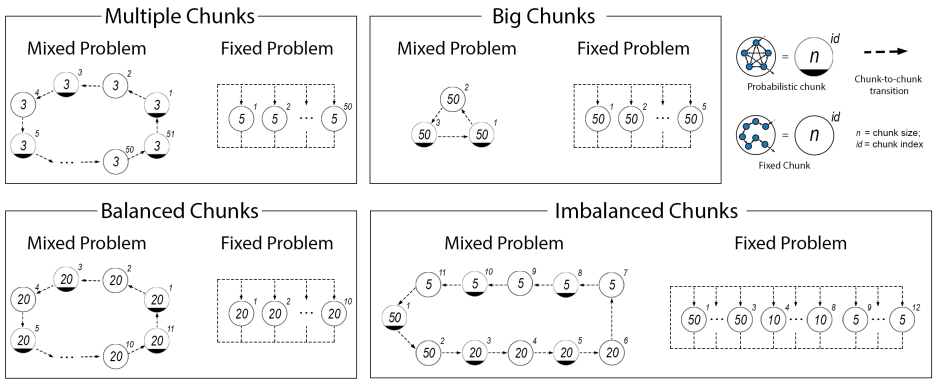

Figure 6: **Illustration of problem categories and settings in high-dimensional CGCP.** For better visualization, we symbolized probabilistic and fixed chunks according to the legend (Top right), along with its chunk size and its index (also indicate the total number of chunks). We categorize our experiments into four major categories (multiple chunks, big chunks, balanced chunks, and imbalanced chunks); each of which contains two types of problems (mixed and fixed problem), resulting in a total of eight environments. Chunk size, chunks composition, number of chunks, and transition of state variables for each environment are illustrated in detail.

D STATISTICAL TESTS

We used a two-sample t-test with a p-value larger than 0.05 to verify if the best result is statistically significantly different from other results. We calculate the two-tailed p value. All the tests are using 30 samples for each model.

Table 3: Statistical Results of CGCPs.

Problems	Description	p value	Problems	Description	p value
Fixed chunks	SubSigma vs mmax	NaN	Fixed chunks	SubSigma vs MRIL	2.6e-03
	SubSigma vs word2vec	NaN		SubSigma vs SyncMap	8.0e-03
Long chunks	SubSigma vs mmax	1.6e-38	Long chunks	SubSigma vs MRIL	2.7e-05
	SubSigma vs word2vec	6.6e-13		SubSigma vs SyncMap	8.6e-08
Mixed chunks	SubSigma vs mmax	5.1e-36	Mixed chunks	SubSigma vs MRIL	5.8e-07
	SubSigma vs word2vec	2.9e-14		SubSigma vs SyncMap	1.7e-1
Overlap1	SubSigma vs mmax	2.6e-31	Overlap1	SubSigma vs MRIL	5.2e-60
	SubSigma vs word2vec	1.2e-16		SubSigma vs SyncMap	2.2e-12
Overlap2	SubSigma vs mmax	3.7e-46	Overlap2	SubSigma vs MRIL	2.7e-68
	SubSigma vs word2vec	4.6e-38		SubSigma vs SyncMap	1.2e-09
Real world1	SubSigma vs mmax	1.7e-02	Real world1	SubSigma vs MRIL	4.6e-2
	SubSigma vs word2vec	3.0e-15		SubSigma vs SyncMap	1.0e-2
Real world2	word2vec vs mmax	1.2e-35	Real world2	word2vec vs SyncMap	1.3e-16
	word2vec vs MRIL	1.0e-24		word2vec vs SubSigma	1.3e-31

Table 4: Statistical Results of High-dimensional CGCPs

Problems	Description	P value	Problems	Description	P value
Multiple Chunks (Mixed)	Mag w/SS/D vs mmax	7.4e-65	Multiple Chunks (Mixed)	Mag w/SS/D vs SubSigma	1.9e-38
	Mag w/SS/D vs word2vec	1.8e-40		Mag w/SS/D vs Mag w/SM/D	7.6e-10
	Mag w/SS/D vs MRIL	1.2e-77		Mag w/SS/D vs Mag w/SS/M	8.9e-17
	Mag w/SS/D vs SyncMap	2.1e-54			
Multiple Chunks (Fixed)	Mag w/SS/M vs mmax	8.9e-27	Multiple Chunks (Fixed)	Mag w/SS/M vs SubSigma	5.6e-47
	Mag w/SS/M vs word2vec	2.7e-46		Mag w/SS/M vs Mag w/SM/D	4.2e-21
	Mag w/SS/M vs MRIL	3.2e-71		Mag w/SS/M vs Mag w/SS/D	2.7e-04
	Mag w/SS/M vs SyncMap	6.3e-57			
Big Chunks (Mixed)	SyncMap vs mmax	3.8e-57	Big Chunks (Mixed)	SyncMap vs Mag w/SM/D	4.7e-19
	SyncMap vs word2vec	1.5e-02		SyncMap vs Mag w/SS/D	8.7e-10
	SyncMap vs MRIL	2.2e-77		SyncMap vs Mag w/SM/M	1.2e-1
	SyncMap vs SubSigma	4.6e-48			
Big Chunks (fixed)	mmax vs word2vec	5.0e-0.5	Big Chunks (fixed)	mmax vs Mag w/SM/D	6.7e-43
	mmax vs MRIL	3.9e-52		mmax vs Mag w/SS/D	8.9e-27
	mmax vs SyncMap	1.4e-54		mmax vs Mag w/SS/M	1.0e-00
	mmax vs SubSigma	5.7e-73			
Balanced Chunks (Mixed)	Mag w/SS/M vs mmax	1.3e-47	Balanced Chunks (Mixed)	Mag w/SS/M vs SubSigma	1.6e-58
	Mag w/SS/M vs word2vec	1.3e-48		Mag w/SS/M vs Mag w/SM/D	9.6e-24
	Mag w/SS/M vs MRIL	3.7e-54		Mag w/SS/M vs Mag w/SS/D	1.1e-42
	Mag w/SS/M vs SyncMap	1.3e-47			
Balanced Chunks (Fixed)	Mag w/SS/M vs mmax	9.5e-16	Balanced Chunks (Fixed)	Mag w/SS/M vs SubSigma	6.0e-53
	Mag w/SS/M vs word2vec	1.9e-05		Mag w/SS/M vs Mag w/SM/D	7.4e-26
	Mag w/SS/M vs MRIL	5.8e-29		Mag w/SS/M vs Mag w/SS/D	2.9e-10
	Mag w/SS/M vs SyncMap	6.9e-29			
Imbalanced Chunks (Mixed)	Mag w/SS/M vs mmax	5.1e-36	Imbalanced Chunks (Mixed)	Mag w/SS/M vs SubSigma	2.4e-48
	Mag w/SS/M vs word2vec	7.4e-16		Mag w/SS/M vs Mag w/SM/D	2.0e-24
	Mag w/SS/M vs MRIL	3.2e-39		Mag w/SS/M vs Mag w/SS/D	1.1e-09
	Mag w/SS/M vs SyncMap	3.3e-40			
Imbalanced Chunks (Fixed)	Mag w/SS/M vs mmax	3.5e-04	Imbalanced Chunks (Fixed)	Mag w/SS/M vs SubSigma	7.4e-65
	Mag w/SS/M vs word2vec	8.1e-66		Mag w/SS/M vs Mag w/SM/D	2.6e-11
	Mag w/SS/M vs MRIL	3.7e-70		Mag w/SS/M vs Mag w/SS/D	1.7e-01
	Mag w/SS/M vs SyncMap	9.7e-71			