In: Arbib M.A. (ed.), *The Handbook of Brain Theory and Neural Networks*, 2nd Ed., MIT Press, Cambridge MA, pp. 1163-1167, 2003.

Temporal Pattern Processing

DeLiang Wang

Department of Computer and Information Science
and Center for Cognitive Science
The Ohio State University, Columbus, OH 43210-1277, U.S.A.
dwang@cis.ohio-state.edu

INTRODUCTION

Temporal pattern processing is important for various intelligent behaviors, including hearing, vision, speech, music and motor control. Because we live in an ever-changing environment, an intelligent system, whether it be a human or a robot, must encode patterns over time, recognize and generate temporal patterns. Time is embodied in a temporal pattern in two different ways:

• Temporal order. It refers to the ordering among the components of a sequence. For example, the sequence *N-E-T* is different from *T-E-N*. Temporal order may also refer to a syntactic structure, such as subject-verb-object, where each component may be any of a category of possible symbols.

• Time duration. Duration can play a critical role for temporal processing. In speech recognition, for example, we want rate invariance while distinguishing relative durations of the vowel /i:/ (as in b*ee*t) and /i/ (as in b*i*t).

Following Wang and Arbib (1990), a sequence is defined as *complex* if it contains repetitions of the same subsequence like *C-O-N-F-R-O-N-T*, otherwise as *simple*. For generation of complex sequences, the correct successor can be determined only by knowing components prior to the current one. We refer to the prior subsequence required to determine the current component as the *context* of the component, and the length of this context as the *degree* of the component. The *degree of a sequence* is defined as the maximum degree of its components. Thus, a simple sequence is a degree 1 sequence.

Temporal pattern processing is a challenging topic because the information is embedded in time (thus inherently dynamic), not simultaneously available. Nonetheless, this topic has been studied by a number of investigators; see Sun and Giles (2001) for a recent collection of articles on this topic.

Fundamentally different from static pattern processing, temporal processing requires that a neural network have a capacity of short-term memory (STM) in order to maintain a component for some time. This is because a temporal pattern extends over a time period. How to encode STM thus becomes one of the criteria for classifying neural networks for temporal processing. In this article, I provide an outline of temporal pattern processing, discussing the topics of recognition and generation separately. In the end, I point out several outstanding issues, including time warping and chunking, that require future investigation.

STM MODELS

Delay Lines

The simplest form of STM uses a fixed-length buffer of N units to maintain the N most recent input items. This can be implemented by either a shift register with a fixed delay between consecutive units or an array of systematic delay lines. The delay line STM transforms a temporal pattern into a spatial one where time forms another dimension. This idea forms the basis of many recognition models (e.g. Waibel et al., 1989).

Decay Traces

Here an item in STM decays in time, corresponding to the decay theory of forgetting in human STM. The decay usually takes on an exponential form. Theoretically, time information can be precisely recovered from the current value. But due to rapid decay and noise, only a limited number of the most recent items can be reliably discerned from STM. This form has been used by Jordan (1986) and Wang and Arbib (1990), among others. Fig. 1a shows a typical decay trace.

Exponential Kernels

Tank and Hopfield (1987) proposed a set of normalized exponential kernels to sample the history, described as

$$f_k(t) = (\frac{t}{k})^{\alpha} e^{\alpha(1-t/k)} \qquad \text{for } k = 1, ..., K \tag{1}$$

where α regulates the width of each kernel. Notice that $f_k(t) = 1$ if $t = k$. Fig. 1b shows a set of 4 kernels. There are K units to represent each symbol. Unlike delay line STM, each unit here samples a symbol within a certain period peaked at a specific time step ($t = k$).

Along similar lines, de Vries and Principe (1992) proposed the *gamma* model, which uses a set of gamma kernels (integrands of Γ-functions, hence the name)

$$g_k(t) = \frac{\mu^k}{(k-1)!} t^{k-1} e^{-\mu t} \qquad \text{for } k = 1, ..., K \tag{2}$$

where μ is a parameter between 0 and 1. K is called the order of the memory, and there are K units for storing a symbol S in STM. Fig. 1c shows a set of 4 kernels. Since g_k has a maximum value at $t = (k–1)/\mu$, μ determines the depth of the peak of each kernel in STM. Thus, unlike normalized exponential kernels, an N-step history may be sampled by less than N gamma kernels. Note that gamma kernels can be computed recursively.

STORE Model

The STM models examined so far are autonomous, for the trace of each item is fully independent of other items in STM. A basic property of human STM is that it has a limited capacity (7±2), so that whether and how long an item is held in STM depends on other inputs entering STM. In addition, the study of human retention of sequences shows a recency factor, where more recent items tend to be better retained, and a primacy factor, whereby the beginning items of a sequence are less prone to forgetting. These two factors together give rise to the bowing effect, which motivated the STORE model (Bradski et al., 1992) using a pair of units

$$x_i(t+1) = x_i(t) + [\beta I_i(t) + y_i(t) - x_i(t)\, x(t)]\, I(t) \tag{3a}$$
$$y_i(t+1) = y_i(t) + [x_i(t) - y_i(t)]\, [1 - I(t)] \tag{3b}$$

where $x(t) = \sum_j x_j(t)$, the sum of STM item values, and $I(t) = \sum_j I_j(t)$, the sum of external inputs. The behavior of STORE has two aspects. The first is the global inhibition term in (3a), which reduces the value of x_i in favor of new items. The second is the excitatory loop between x_i and y_i, which favors old items in STM. Combined together, they are able to produce the bowing shape for a sequence of items. Fig. 1d shows three bows generated with different values of β.

TEMPORAL PATTERN RECOGNITION

The shared goal of all STM models is to make input history available simultaneously when recognition takes place. With a STM model in place, recognition is not much different from the recognition of static patterns.

Template Matching Using Hebbian Learning

The architecture for this type of recognition is simply a two-layer network: the input layer that incorporates STM, and the sequence recognition layer where each unit encodes an individual sequence. The recognition scheme is essentially template matching, where templates are formed through following Hebbian learning

$$W_{ij}(t) = W_{ij}(t-1) + C\, s_i(t)[x_j(t) - W_{ij}(t-1)] \tag{4}$$

where W_{ij} is the connection weight from unit x_j in the input layer to sequence recognizer s_i in the recognition layer. Parameter C controls learning rate. Hebbian learning is applied after the presentation of the entire sequence is completed. The templates thus formed can be used to recognize specific input sequences. The recognition layer typically includes recurrent connections for selecting a winner by self-organization (e.g. winner-take-all) during training or recognition.

Kohonen (1990) proposed an architecture, called the *phonetic typewriter* for phoneme recognition. The phonetic typewriter extracts a vector of frequency components using Fast Fourier Transform. After this step, his algorithm of feature mapping is applied for recognition, where winner-take-all is applied in both training and recognition. The phonetic typewriter has been applied to recognize Finnish and Japanese phonemes (Kohonen, 1990). Wang and colleagues (Wang and Arbib, 1990; Wang and Yuwono, 1995) adopted a learning method similar to (4), and showed that a recognition algorithm plus either decay trace or shift register can recognize complex sequences.

Associative Memory Approach

The dynamics of the Hopfield associative memory model (see ARTICLE) can be characterized as evolving towards the memory state most similar to the current input pattern. If one views each memory state as a category, the Hopfield net performs pattern recognition: the recalled category is the recognized pattern. This process of dynamic evolution can also be viewed as an optimization process, which minimizes a cost function until equilibrium is reached.

With normalized exponential kernel STM, Tank and Hopfield (1987) described a recognition network based on associative memory dynamics. A layer of sequence recognizers receives inputs from the STM model. Each recognizer encodes a different template sequence by its unique weight vector acting upon the inputs in STM. In addition, recognizers form a competitive network. The recognition process uses the current input sequence (evidence) to bias a minimization process so that the most similar template wins the competition, thus activating its corresponding recognizer. Due to the exponential kernels, they demonstrated that recognition is fairly robust to *time warping*, distortions in duration. A similar architecture is later applied to speaker-independent spoken digit recognition.

Multilayer Perceptrons

A popular approach to temporal pattern learning is multilayer perceptrons (MLP). MLPs have been demonstrated to be effective for static pattern recognition. It is natural to combine MLP with an STM model to do temporal pattern recognition. For example, using delay line STM Waibel et al. (1989) reported an architecture called Time Delay Neural Networks (TDNN) for spoken phoneme recognition. Besides the input layer, TDNN uses 2 hidden layers and an output layer where each unit encodes one phoneme. The feedforward connections converge from the input layer to each successive layer so that each unit in a specific layer receives inputs within a limited time window from the previous layer. They demonstrated good recognition performance: for the three stop consonants /b/, /d/, and /g/, the accuracy of speaker dependent recognition reached 98.5%.

TEMPORAL PATTERN GENERATION

An early model of sequence generation is the *outstar avalanche* introduced by Grossberg in 1969, which is composed of *n* sequential outstars. Each outstar $\boldsymbol{M}_i$ stores a static pattern and is activated by a signal in the vertex v_i. These vertices are connected as: $v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_n$. So an initial signal at v_1 can generate sequentially the spatial patterns stored in $\boldsymbol{M}_1$, $\boldsymbol{M}_2$, ..., $\boldsymbol{M}_n$. See Grossberg (1982) for a more detailed description as well as some of the later extensions. In recent years, a number of more sophisticated solutions have been proposed for temporal pattern generation.

Associative Memory Approach

Since associative memory studies how to associate one pattern with another, its mechanism can be extended to generating a sequence. A sequence is treated as a set of pairs between consecutive components, and these pairs are stored into an associative memory. Hence, after the first component of the sequence is presented, the next component will be activated after some delay, which further activates the third one, etc. This basic idea, however, leads to ambiguity when generating a complex sequence, where one pattern may be followed by different ones. Several investigators have proposed to use high-order networks to deal with the problem. In a *k*th-order network, the input to each unit is the weighted sum of *k*-tuples, instead of individual units, and each *k*-tuple is a product of *k* units. In such a network, one component in a sequence is associated by a prior subsequence of length k. Thus, a sequence of degree k can be generated without ambiguity by a *k*th order associative memory (Guyon et al., 1988). A major problem with high order networks is the required number of connections, which grows exponentially with the order of the network.

Multilayer Perceptron Approach

Jordan (1986) described the first MLP architecture with recurrent connections for sequence generation. The input layer has two parts: plan units representing external input and the identity of the sequence and state units that receive one-to-one projections from the output layer, forming decay trace STM. After a sequence is stored into the network by backpropagation training, it can be generated by an external input representing the identity of the sequence. This input activates the first component of the sequence in the output layer. This component feeds back to the input layer and, together with the external input, activates the second component, and so on. A particular component of a sequence is generated by the part of the sequence prior to the component, earlier components having lesser roles due to exponential decay. Elman (1990) later modified Jordan's architecture by having the hidden layer connect to a part of the input layer, called the context layer. The context layer simply duplicates the activation of the hidden layer in the previous time step.

Elman used this architecture to learn a set of individual sequences satisfying a syntactic description, and found that the network exhibits a kind of syntax recognition. This result suggests a way of learning high-level structures, such as natural language grammar.

Anticipation Model and Multi-associative Networks

Generation of complex sequences has been a major issue. The approaches described so far rely on either fixed degree contexts, or a composite vector recording the history. The latter is prone to ambiguity. The former entails high system overhead, because, in order to avoid ambiguity, such a method must use a degree no smaller than the degree of the sequence, which is usually much greater than the degrees of most components. This analysis calls for a mechanism of self-organization, where each component in a sequence can learn the degree of its own context.

By extending basic ideas for complex sequence processing in Wang and Arbib (1990), Wang and Yuwono (1995) introduced such a mechanism of self-organization for generating complex patterns. This so-called *anticipation model* is based on the following two ideas. First, the system actively anticipates the next component in sequence learning, and a mismatch between the anticipated component and the actual component triggers context adjustment through competitive learning. Second, generation of a sequence component hinges on recognition of the component's context. Fig. 2a shows the architecture of the anticipation model, using a shift-register STM model. Each unit in the detector layer recognizes a specific context, and a winner-take-all mechanism is implemented within the detector layer. There is a modulator layer in correspondence to the detector layer, and each modulator receives a downward connection from its respective detector as well as upward connections from every input terminal. The modulators perform comparison between anticipation from a winning detector and the next input component. Wang and Yuwono (1995) showed that the anticipation model can learn to generate an arbitrary temporal sequence.

Wang and Yuwono (1996) later applied the anticipation model to learn multiple complex sequences sequentially, i.e. new training does not take place until existing sequences are acquired. Using a set of 97 highly correlated complex sequences, they demonstrated that learning a new sequence can interfere with already acquired ones. However, the number of intact sequences increases linearly with the size of the existing memory, while the amount of retraining needed to eliminate interference is independent of the size of the memory, as illustrated in Fig. 2b. Such characteristics are largely derived from the fact that each sequence is represented in a distributed manner, and different sequences and subsequences within a sequence may share context detectors. The interference properties of the anticipation model are consistent with human retroactive interference well documented in psychology, while contrasting MLP that shows catastrophic interference in sequential learning.

A limitation with the anticipation model is that it deals with symbol sequences rather than sequences of spatial patterns. Recently, L. Wang (1999) proposed to use multi-associative neural networks for learning and retrieving spatiotemporal patterns. STM is coded as systematic delay lines. The basic idea is that, when dealing with complex sequences, one pattern is allowed to be associated with a set of multiple subsequent patterns, and ambiguity can be eliminated by intersecting multiple sets associated by the previous pattern, the pattern prior to the previous pattern, etc. A complex sequence of degree k can thus be unambiguously generated with k systematic delay lines. Associations between spatial patterns are established through individual units in a competitive layer. A drawback of this model is that many network operations are algorithmically described, rather than arising from a network autonomously.

DISCUSSION

This brief tour of neural network processing of temporal patterns shows that effective models and techniques exist for both recognition and generation (see DOMINEY article for a more biologically oriented review). There are, however, many questions yet to be answered. In my view, the following two problems are particularly interesting and challenging for future research:

(1) Rate invariance and time warp. Humans show rate invariance to a certain extent in recognizing a temporal pattern. *Rate invariance* is different from what I call *interval invariance*, where the former is invariance to only global scaling of durations and the latter to all changes of durations. Interval invariance is exhibited in several models, but not rate invariance (see Wang et al., 1996, for a comprehensive discussion). One must be careful about time warping. We would like to have invariance over limited warping, but dramatic change in relative duration must be recognized differently.

(2) Chunking. A fundamental ability of human information processing is chunking, which, in the context of temporal processing, means that frequently encountered and meaningful subsequences organize into chunks that form basic units for further chunking at a higher level. Chunking and STM are closely related. A chunk often corresponds to a meaningful subsequence (such as a melody), but it may be just a convenient way of breaking a long sequence into shorter parts to cope with limited capacity of STM. One aspect of chunking is studied by Wang and Yuwono (1996), but the general problem of unsupervised chunking has been little addressed and will be increasingly important for future research into temporal pattern processing.

REFERENCES

Bradski, G., Carpenter, G.A., and Grossberg, S., 1992, Working memory networks for learning temporal order with application to three-dimensional visual object recognition, Neural Comp., 4: 270-286.

de Vries, B.D., and Principe, J.C., 1992, The gamma model - A new neural model for temporal processing, Neural Net., 5: 565-576.

Elman, J.L., 1990, Finding structure in time, Cognit. Sci., 14: 179-211.

*Grossberg, S., 1982, Associative and competitive principles of learning and development: The temporal unfolding and stability of STM and LTM patterns, in Competition and Cooperation in Neural Networks, (S. Amari and M.A. Arbib, Eds.), New York, Springer-Verlag, pp. 295-341.

Guyon, I., Personnaz, L., Nadal, J.P., and Dreyfus, G., 1988, Storage and retrieval of complex sequences in neural networks, Phys. Rev. A, 38: 6365-6372.

Jordan, M.I., 1986, Attractor dynamics and parallelism in a connectionist sequential machine, in Proceedings of the Eighth Annual Conference of the Cognitive Science Society, Hillsdale NJ, Erlbaum, pp. 531-546.

*Kohonen, T., 1990, The self-organizing map, Proceedings of IEEE, 78: 1464-1480.

*Sun, R., and Giles, C.L. (Eds.), 2001, Sequence Learning, Berlin, Springer.

Tank, D.W., and Hopfield, J.J., 1987, Neural computation by concentrating information in time, Proc. Natl. Acad. Sci. USA, 84: 1896-1900.

Waibel, A., Hanazawa, T., Hinton, G.E., Shikano, K., and Lang, K.J., 1989, Phoneme recognition using time-delay neural networks, IEEE Trans. Acoust. Speech Signal Process., 37: 328-339.

Wang, D.L., and Arbib, M.A., 1990, Complex temporal sequence learning based on short-term memory, Proc. IEEE, 78: 1536-1543.

Wang, D.L., Liu, X.M., and Ahalt, S.C., 1996, On temporal generalization of simple recurrent networks, Neural Net., 9: 1099-1118.

Wang, D.L., and Yuwono, B., 1995, Anticipation-based temporal pattern generation, IEEE Trans. Syst. Man Cybern., 25: 615-628.

Wang, D.L., and Yuwono, B., 1996, Incremental learning of complex temporal patterns, IEEE Trans. Neural Net., 7: 1465-1481.

Wang, L., 1999, Multi-associative neural networks and their applications to learning and retrieving complex spatio-temporal sequences, IEEE Trans. Syst. Man Cybern. - Pt. B: Cybern., 29: 73-82.

FIGURE CAPTIONS

Figure 1. STM traces. **a**. Exponential decay. **b**. Normalized exponential kernels. $\alpha = 5.0$, and $k = 1, ..., 4$ for the curves from left to right, respectively. **c**. Gamma kernels. $\mu = 0.9$, and $k = 1, ..., 4$ for the curves from left to right, respectively. **d**. The STORE model. $\beta = 0.5$ for the empty square bow, 0.3 for the diamond bow, and 0.15 for the filled square bow. Seven items are kept in STM. (scanned from Fig. 2, 1st edition.)

Figure 2. Anticipation model. **a**. Architecture (adapted from Wang and Yuwono, 1995). Thin solid lines denote modifiable connections, thick and dashed lines fixed connections, and undirected lines bidirectional connections. **b**. The number of intact sequences and the number of retraining sweeps with respect to the number of training rounds for individual sequences (adapted from Wang and Yuwono, 1996). The 97 sequences denote all the session titles of the 1994 IEEE International Conference on Neural Networks.

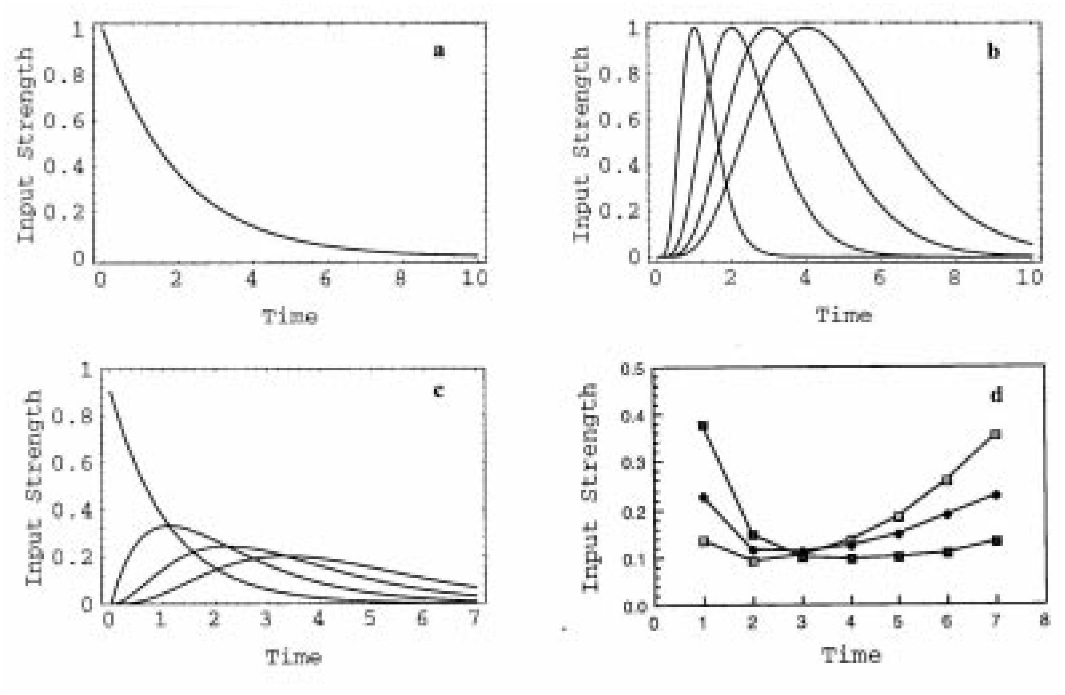
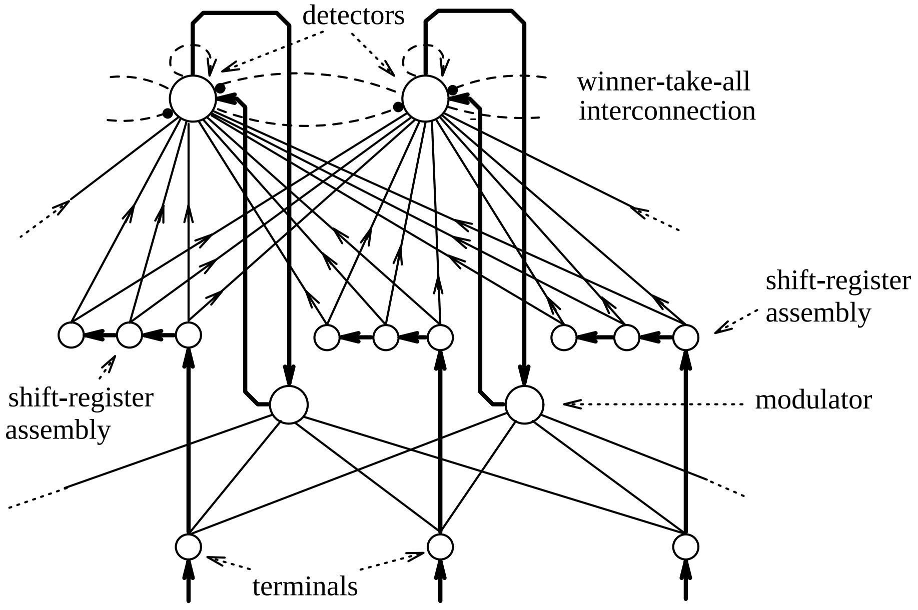


Figure 1

a


detectors
winner-take-all interconnection
shift-register assembly
shift-register assembly
modulator
terminals

b

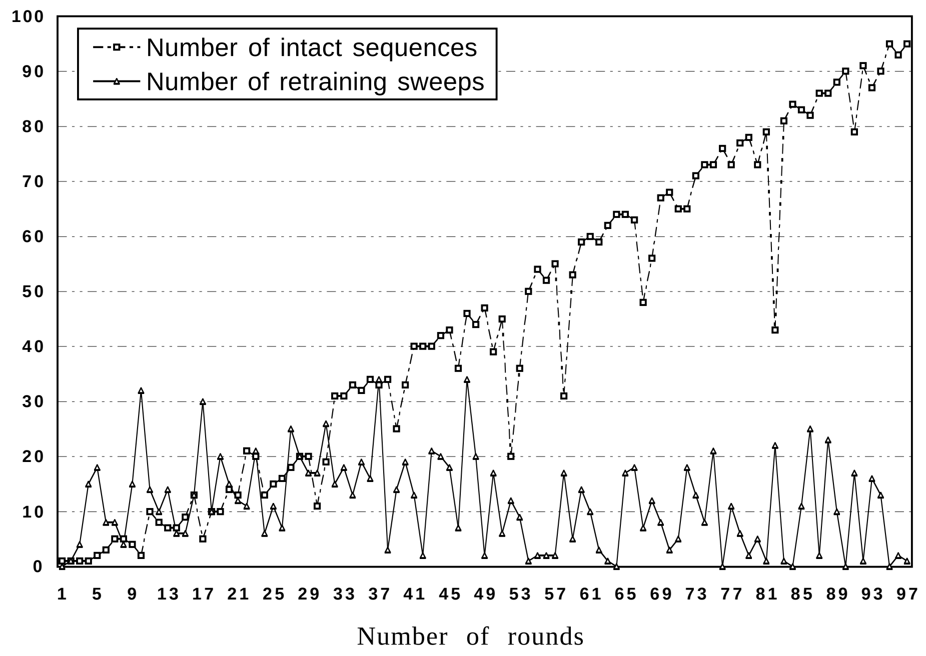

Number of intact sequences
Number of retraining sweeps
Number of rounds
100
90
80
70
60
50
40
30
20
10
0
1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97

Figure 2