REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
04-10-2002	Final Report	21 September 2001 - 21-Mar-03

4. TITLE AND SUBTITLE

Title: Models of Distributed Computation

Subtitle: "Behavior Characterization of Intelligent Problem Solving for an Agent Hierarchy in a Competitive (Web) Environment"

5a. CONTRACT NUMBER
F61775-01-WE023

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Professor Andras Lorincz

5d. PROJECT NUMBER

5d. TASK NUMBER

5e. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Neumann János Számítógép-tudományi Társaság
Pazmany Peter setany 1/C
Budapest H-1117
Hungary

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

EOARD
PSC 802 BOX 14
FPO 09499-0014

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
SPC 01-4023

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report results from a contract tasking Neumann János Számítógép-tudományi Társaság as follows: The contractor will investigate approaches to alleviate performance limitations of large information management systems operating under the constraints of computational resources (computing power), communication bandwidth, and time. The main task will be to develop a global Function APProximator (FAPP) that will lead to optimised solutions for balancing the three constraints while assisting in the decision making process. The research will investigate methods for making 'rational choices' in a highly competitive environment, that can learn from experience, use previous examples to solve problems, and recognize novel examples where previous experience could be misleading. This methodology is comprised of three steps: (1) search fast, (2) efficiently remember what you have found, and (3) recognize if a new search is necessary (e.g., recognize novelty). The effort will consist of five phases:

1. JAVA based single object (unit) with flexible information collection and management capabilities
2. Text classification tools, including XML tools that have been developed during the present project
3. Develop a model system for distributed computation with communication and computational overhead. The model system will include probabilistic features of NP-complete problems and bandwidths that can be experienced over the internet.
4. Develop an optimization scheme based on recent re-phrasing of reinforcement learning
5. Demonstrate capabilities on real-life distributed computation problem: Distribute and optimize computations over the internet under constraints imposed by communication bandwidth and by computational costs of text classification.

15. SUBJECT TERMS

EOARD, Agent Based Systems, Joint Battlespace Infosphere, NP-complete problems, distributed computation, evolving networks

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18, NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Christopher Reuter, Ph. D.
UNCLAS	UNCLAS	UNCLAS	UL	247	19b. TELEPHONE NUMBER *(Include area code)* +44 +44 207 514 4474

Meta level analysis of Hebbian evolving networks

Zs. Palotai, G. Szirtes, A. Lorincz

April 11, 2002

Abstract

The goal of this paper is to explicitly address the question whether any true analogy exists between the central nervous system and the world wide web. By exploiting intuitive ideas of different fields ranging from neurobiology to sociology we investigate and present some properties of different evolving networks, whose dynamics is governed by the Hebbian learning rule. We intended to merge two different fields: the tools have been provided by the graph theory of evolving networks, while the properties of the Hebbian based learning rule is provided by computational neuroscience (CNS) models. We studied the emerging connection structure of purely feed-forward and feedback networks.

Contents

List of Figures

List of Tables

0.1 Introduction

The last few years have witnessed an increasing demand on describing systems with complex structure and topology. These structures emerge in many different fields, ranging from social sciences to genetics. Probably the most complex network is inside us: the most exciting properties of our brain have a lot to do with the special connection system among its units. Although our knowledge on these building blocks is increasing we are still far from a complete understanding of the structure [5]. At the same time, many intriguing questions emerge regarding the World Wide Web (WWW), which can qualify as a common container of human knowledge. This feature resulted in the intuitive idea to highlight the similarity between these descent phenomena. From one hand it has been suggested to use the mutual activity correlation in modeling organizational learning[4], while on the other hand similar structural characteristics of the web and the single completely described nervous system, the *C. elegans* has been reported[6]. In this Technical Report we investigate the question whether an evolving network governed by the Hebbian rule has the same or similar properties as found by studying the web or the social networks. Specifically, we studied the emergent small-world properties,like clusters and hubs which are presumably responsible for the robustness of many real networks against different perturbations. Their presence may also explain the network fragility against erasing central nodes, which have been also observed in many complex and important networks. The question is of central importance: in seeking the answer we hope to find in the near future some explanation on the fascinating robustness-plasticity dualism of the brain and the fast association memory system. Insofar as we do not intend to do the tedious job of building real models according to the computational neuroscience (CNS) standards, we have built different networks of units with minimal complexity. Inputs with different information content have been provided to the system. The system has two-fold response: providing the input it yields some output *and* modifies its connection. This self-organizing tuning was governed by different modifications of the Hebbian learning rule. One group of the networks consisted of purely feed-forward connections (that is units were organized in layers), while the other group have had feedback connections as well.

By watching the evolving networks we have tried to define and localize structures similar to those that are used in the terminology describing the internet. This Technical Report is primarily meant to summarize the simulation results at different parameter settings. The Report's construction is as follows: first we give a basic description of the most fundamental conceptions we used. To make the understanding easier we also provide a detailed list of the used notations. The next sections are for detailing the different models and presenting the governing equations. At last, all simulation results are presented with no analysis. A descriptive analysis would require further simulations. The results may hopefully give a qualitative picture of the Hebbian network's ('HebbWeb' in short) nature.

0.2 Definitions

0.2.1 Concepts

HebbWeb Set of neurons with a simple Hebbian learning rule gets some random external excitation. The connections among the neurons (or units or nodes) can be characterized by a weighted, directed graph. We address the question, whether the evolving network resembles the WEB's structure in its connection complexity. We are especially interested in emerging clusters, since they are supposed to contribute to the appearance of the famous small world property of the WEB[1].

Distance the path with minimum cost between two nodes. The cost of a path is the sum of link costs on the path. The link cost is $1/w$ where w is the weight between the nodes of the link.

Cluster Set of nodes which are strongly connected to each other and have fewer links to the other nodes.

Clustering coefficient is defined for each neuron in the following way: the ratio of the existing links and the possible number of links $(n * (n-1)$ between neurons linked by the actual neuron.

Hub and Authority values calculated iteratively [3] for each neuron. The weight matrix is treated as a transition matrix after a structure holding normalization. One iteration is the following:

$$\mathbf{aut} = \mathbf{P}^t * \mathbf{hub}$$

$$\mathbf{hub} = \mathbf{P} * \mathbf{aut}$$

where $\mathbf{P}$ is the normalized weight matrix, **aut** is authority vector and **hub** is the hub vector .

Feedback layer The neurons and the directed connections among them

Feedforward layer The connection layer between the input and the neurons. In pure feedback simulations the feedforward connection matrix is equals to the Identity matrix.

0.2.2 Notations

W: The weight matrix of the feedback layer

$\mathbf{W}^{ext}$**:** The weight matrix of the feedforward layer

a: The inner activity of the feedback layer

$\mathbf{a^f}$**:** The firing activity of the feedback layer

N**:** The number of neurons in the net.

r**:** The percentage of outer excitation relative to the number of neurons (N)

α**:** The learning rate of matrix $\mathbf{W}$

β**:** The learning rate of matrix $\mathbf{W}^{ext}$

γ**:** The decay factor of the inner activity, only used in the first model

*cycle_length***:** It defines the simulation length (number of update steps).

$K(.)$ This kernel function describes the temporal correlation between nodes which have been active at different times.

L**:** The length of the kernel function for negative time difference.

r_A**:** The ratio of the amplitude of weakening and strengthening in the kernel function.

θ**:** Threshold parameter in the firing rule - its meaning is case-dependent

$finite\_input$**:** It is set to 0, if all neurons may get excited, that is the sum of the external excitation is not bounded. Its value is 1, if this sum is bounded.

$\mathbf{x}(i)^{ext}$**:** The external input at time i.

0.3 Experiments

We studied networks with fix number of nodes, but changing connections strength. This change is induced by random inputs of the external world and principally governed by local, Hebbian like interactions among the nodes or neurons. Our numerical studies aim at revealing the possible parameter dependence of the cluster creation and authority-hub properties and are not meant to allow for direct comparison with sophisticated CNS models. Specifically, we have studied the evolution of Hebbian networks with and without feedback connections as a function of some fundamental parameters, such as tuning (learning) parameters, input dynamics and intrinsic network properties (inhibition, bias, and so on).

0.3.1 Network structure and dynamics

Model structure

Our models consist of two layers of simplified model neurons. In some cases, we used neurons, whose output is simply a linear function of their activity, instead of using the more sophisticated integrate-and-fire (IAF) type neurons. In other cases we applied neurons with IAF-like behavior, but made an effort to eliminate as much parameters as we could. In addition to changing the neurons' internal dynamics, different architectures have been investigated. In the first model type one layer ensures the connection with the external world: its neurons can be excited by external random inputs. The other layer's units are driven by the input neurons' excitation and the connections among the units are modified according to the activity of the sender and receiver units (see Figure 1). In addition to the pure feedforward model, the second model type also includes recurrent connections among the units of the internal layer (see Figure 2).

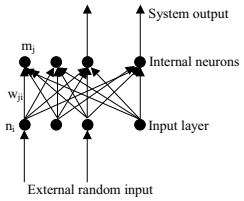


Figure 1: **Pure feedforward network**

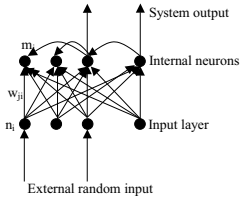


Figure 2: **Pure feedback network**
In all cases studied, the feedforward connection matrix is equal to the Identity matrix.

Input generation

We studied the system's behavior as a function of the inputs' spatio-temporal structure. In most cases the external input $\mathbf{x}^{ext}$ consists of zeros and $N * r$ number of ones at random (binary input), where r defines the induction ratio. For the special case of sine wave inputs, neurons received a half sine wave input in each step (analogue input).

The excitation of the (internal) neurons takes place in two different modes. In the first case ($finite\_input = 1$) the input to each neuron is weighted according to its connection strength. In other words, the sum of the input was kept unchanged after the transmission through the weighted connections. In the other case ($finite\_input = 0$) the excitation of one neuron is simply the scalar product of the input vector and the weight vector of its incoming connections (the i^{th} row of matrix W^{ext} describes the i^{th} neuron's connection strength).

Model's dynamics

The network has two responses to the external excitation. It not only generates output (which is a function of the activity measured in the internal layer), but also modifies the connections between and within the layers. We define two different output generation mode: in the first mode (it can be called analogue firing), the neurons broadcast a sign, which is a linear function of their input. In the second mode (spiking or binary firing), they produce binary signs, which are non-linear function of their input. The applied non-linearities will be detailed later (See subsection 0.3.2). The general update rule for the inner activity is as follows:

$$\mathbf{a}(t) = \mathbf{a}(t-1) + \mathbf{W}\mathbf{a}(t-1)^f + \mathbf{W}^{ext}\mathbf{x}(t)^{ext} \tag{1}$$

When analogue input was applied a decay factor γ was introduced. The modified update rule is described in Eq. 2.

$$\mathbf{a}(t) = (1-\gamma)\mathbf{a}(t-1) + \gamma \mathbf{W}\mathbf{a}(t-1)^f + \mathbf{W}^{ext}\mathbf{x}(t)^{ext} \tag{2}$$

where superscript stands for firing which can be one of the types listed above. Using this mode the resulting neuron model resembles the *leaky* IAF models. After firing, the neuron's inner activity is set to 0 (reset activity).

The other response of the system is an incremental change in its architecture. We applied different versions of the so called Hebbian learning rule for governing the tuning of each connection strength. The main inspiration behind this choice that this rule seems to be one of the most general and fundamental principles defining the neuronal plasticity. Furthermore, it is exclusively based on local interaction: there is no need to tune an external parameter or define some global optimization process [2].

The update rule for the feedforward matrix is the following:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \alpha \sum_{i=0}^{t} (K(\mathbf{a}^f(i), \mathbf{x}^{ext}(t), t-i) + K(\mathbf{a}^f(t), \mathbf{x}^{ext}(i), i-t))$$

where α is the learning parameter, $K(.)$ denotes the specific kernel function defined by the specific version of the Hebbian rule. Its arguments are the firing activity at a given time i, the actual input and time difference $t-i$ or $i-t$. This kernel describes the correlation between nodes which have been active at different time. If the information flow performed by the net has a specific direction, the kernel function cannot be symmetric over time. That is, 'pre' and 'post' active states can be assigned to the interacting nodes. Symmetric, linearly decreasing and asymmetric kernels have been studied.

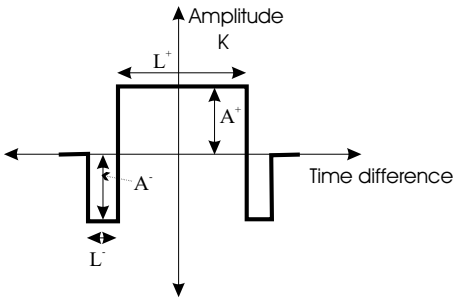


Figure 3: **The symmetric kernel function**
The kernel can be defined by the following parameters: the sum of the length of the weakening and the strengthening region ($L = L^+ + L^-$) and the ratio of the amplitudes ($r_A = \frac{A^+}{A^-}$).

The update rule for the feedback connection matrix is the following:

$$\mathbf{W}(t) = \mathbf{W}(t-1) + \alpha \sum_{i=0}^{t} (K((\mathbf{a}^f(i)), \mathbf{a}^f(t), t-i) + K((\mathbf{a}^f(t)), \mathbf{a}^f(i), i-t)) \tag{3}$$

where α is the learning parameter, $K(.)$ denotes the specific kernel function defined by the specific version of the Hebbian rule. Its arguments are the firing

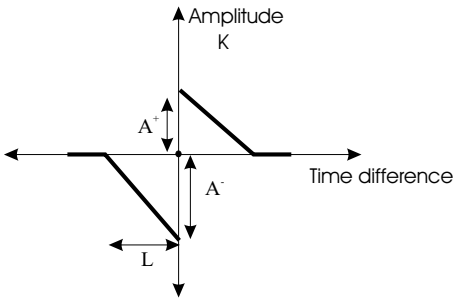


Figure 4: **Kernel function with linear decrease**
The kernel can be defined by the following parameters: the length of the weakening and the strengthening region (L) and the ratio of the amplitudes ($r_A = \frac{A^+}{A^-}$).

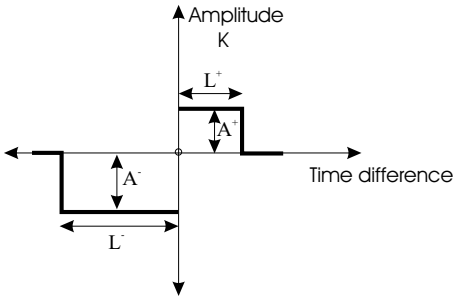


Figure 5: **The asymmetric kernel function**
This kernel can be defined by the following parameters: the ratio of the length of the weakening and the strengthening region ($r_L = \frac{L^+}{L^-}$) and the ratio of the amplitudes ($r_A = \frac{A^+}{A^-}$). In our simulations L^+ was fixed to 1 and L^- was changed.

activity at a given time i, the recent firing activity and the time difference $t - i$ or $i - t$.

0.3.2 Default values and model set up

Different firing models have been studied:

1. Simple threshold model: a neuron fires if its inner activity is larger than a fixed threshold (θ)

2. Probabilistic firing with threshold: a neuron fires if its inner activity is larger than a fixed threshold (θ). Below that the i^{th} neuron fires with probability $a(i)/\theta$

3. Probabilistic firing with fixed average number of firing neurons: the sum of all inner activity is normalized to $N * \theta$. The normalized activity of a neuron is the probability of its firing.

parameter	value
N	100
$cycle\_length$	5000
α	0.1
γ	0.1
r	0.1
θ	0.13
L	2
r_A	2

Table 1: **Default values for analogue firing**

Analogue firing

Default values (see Table 1)

During the simulations only the feedback layer's connections were changed and $\mathbf{W}^{ext}$ remained unchanged and equal to Identity, i.e. $\beta = 0$ and $\mathbf{W}^{ext} = \mathbf{I}$.

The following list summarizes the experiments with analogue output.

1. **kernel size and amplitude ratio** (20020124T105349 – 20020124T130526), asymmetric kernel, simple threshold firing model, $L = [1, 2, 4]$, $r_A = 1 : .5 : 4$, $\theta = [.01 : .04 : .21]$
2. **larger firing threshold** (20020124T134702 – 20020124T144448), asymmetric kernel, simple threshold firing model, $L = 1$, $r_A = [3, 3.2]$, $\theta = [.19 : .02 : .41]$
3. **Study of linearly decreasing kernel** (20020205T172012 – 20020206T113601), simple threshold firing model, $L = [2, 4, 8]$, $r_A = [2, 4, 8]$, $\theta = [.09, .13, .27]$
4. **moving sine valued input** (20020207T193755 – 20020207T232358), linearly decreasing kernel, probabilistic firing with threshold, $L = 2$, $r_A = 4$, $r = .1 : .2 : .9$, $\theta = [.5 : .5 : 4]$

Spiking type firing

Changing from analogue to binary output is meaningful for several reasons. First, it is much more plausible from the biologist aspect. Second, it is more feasible from the engineers' point of view. As the introduction of this non-linearity seems to complicate the picture we intended to further simplify the model by eliminating some parameters. During the simulations the γ decay factor (see equation 1 proved to be dispensable. Its normalization effect has been substituted by proper setting of the remaining parameters.

The idea behind using the feedforward layer is the preferential attachment. This layer should strengthen its weights to neurons which fire after receiving external excitation.

The α and β parameters is smaller than at the analogue firing case because the update is the sum of 0s and 1s and is independent from the actual values of inner activity. The system was sensitive to the setting of these parameters in that if they were too large then the weight distribution could not converge but started to behave in a chaotic manner. For default values see Table 2.

parameter	value
N	100
$cycle\_length$	5000
α	0.0001
β	0.001
r	0.1
θ	0.5
L	2
r_A	2

Table 2: **Default values for spike like firing**

Figure 6: **1 synchronous input**
$active\_length = 10$, $inactive\_length = 10$, $N = 20$, $r = 0.3$. The first input is without any noise. In the second one 80 percent of input is missing. On the third one 20 percent of neurons get external noise.

We studied the feedforward and feedback layers separately.

Some special non-random input with given spatio-temporal structures have been also studied with the symmetric kernel. These were:

1. Moving bar: $N * r$ neuron received unity input, after 10 steps the bar was shifted with one neuron and so on.

2. Moving bar with noise: random noise, i.e. 0-1 flip was added to the previous input with a given percentage.

3. Synchronous input: $N * r$ neurons got input for a given number of steps ($active\_length$) after that no input was given to any neurons for a given interval ($inactive\_length$). See Fig. 6. 1.

4. Synchronous input with noise: There is two types of noises: the input is not complete; neurons not belonging to the pattern also get external excitation. See Fig. 6.2-3.

5. 2 synchronous inputs (with noise): two inputs were mixed, they were synchronous with the time window, but had different period lengths. See Fig. 7.

6. 1 synchronous input and 1 out of synchronous work. See Fig. 8.

0.4 Measured values

The following type of graphs were generated:

1. Histogram of the minimum path length of the hebbian network.

2. Norm of weight matrix vs. time

Figure 7: **2 synchronous inputs**
active_length = 10(6), *inactive_length* = 14(9), $N = 20$. The first input has 2 separate synchronous inputs with $r = 0.3$. The second input has 2 synchronous inputs with 1 common neuron and $r = 0.4$.

Figure 8: **1 synchronous and 1 out of synchronous work input**
active_length = 10(15), *inactive_length* = 14(20), $N = 20$. The first input has 2 separate inputs with $r = 0.3$. The second input has 2 inputs with 1 common neuron and $r = 0.4$.

3. The resulting weight matrix
4. Histogram of the connection strengths. (Zero strength is not shown)
5. Hub-Authority values and ratios for neurons: upper graph: line is the authority value of neuron, dotted line is the hub value of neuron; middle graph: the authority/hub ratio if hub value $\neq 0$; lower graph: the hub/authority ratio if authority $\neq 0$.
6. Clustering coefficients of hebbian (-o-) and random (-x-)network
7. Global and local connectivity length of hebbian (-, -o-) and random (., .x.)network
8. Distance plot of hebbian network: -: the ordered directed distances between the neurons. x: the number of different neurons belonging to at least one of the countered connections, normalized by the maximum distance.

0.5 Results

The value of α, β and γ are set so that the system converges fast if possible - the values in default value tables are to show the order of these parameters.

0.5.1 Analogue firing

We have found that the net with this firing model always evolves to a random statistic net if the input is random. The typical random statistics of these nets can be seen in figure 9.

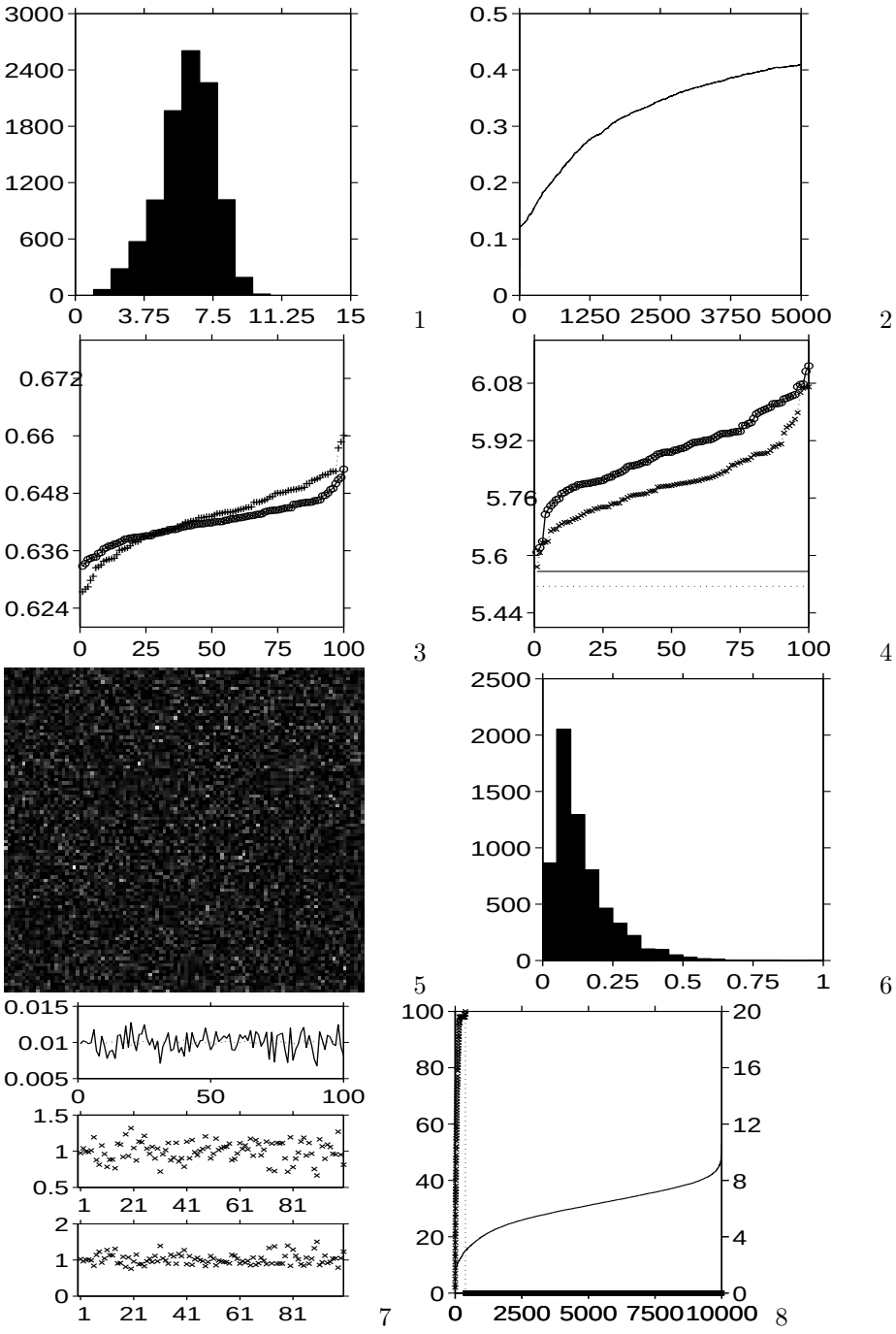


Figure 9: **Random statistics**
For figure details see section 0.4.The parameter values are summarized in Table 1, firing model is 1, time window is 1.

Large initial activity

In this case the initial activity had relatively large standard deviation (see Figure 10. The net had a short transient clustered but not *efficient* state, as it can be seen in figure 11. After this state the net had evolved to random statistics net (see Figures 12, 13). The default parameter values of Table 1 have been applied, the firing model was the simple threshold model, the kernel function was asymmetric.

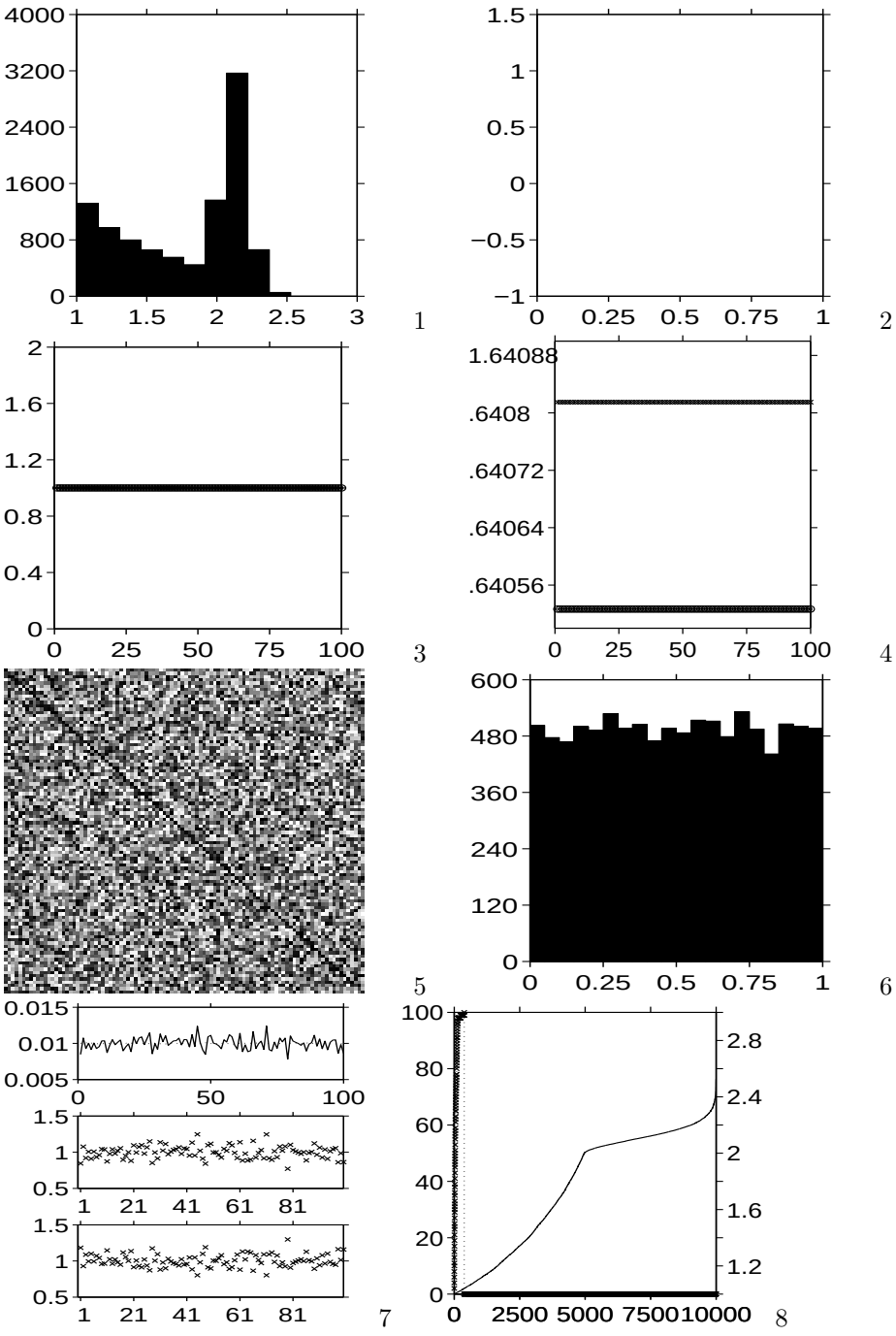


Figure 10: **Large initial activity: initial state statistics**
For figure details see section 0.4.

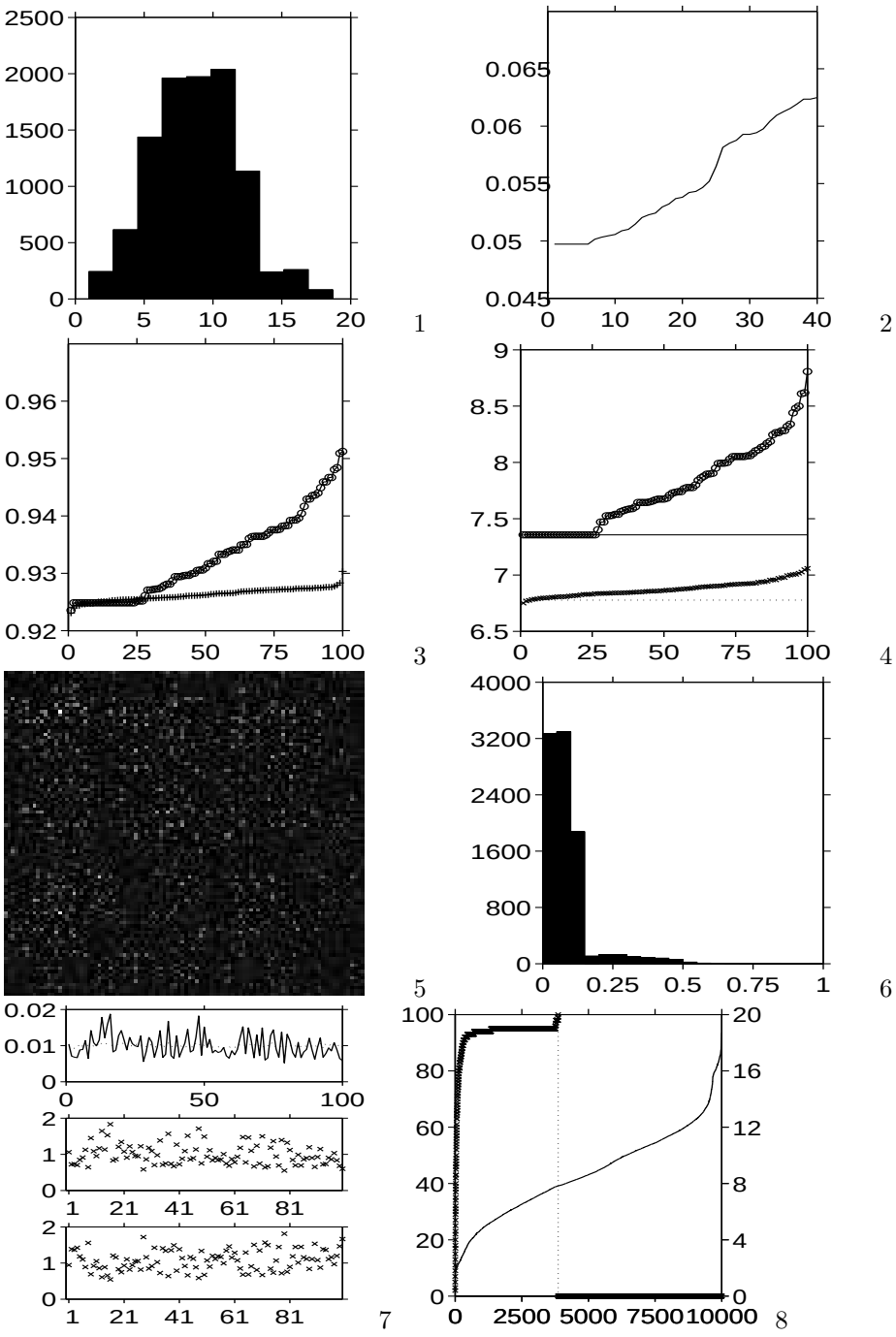


Figure 11: **Large initial activity: transient state statistics**
For figure details see section 0.4. The relatively big initial activity have caused the appearance of a transient, clustered, but not efficient state after 40 steps.

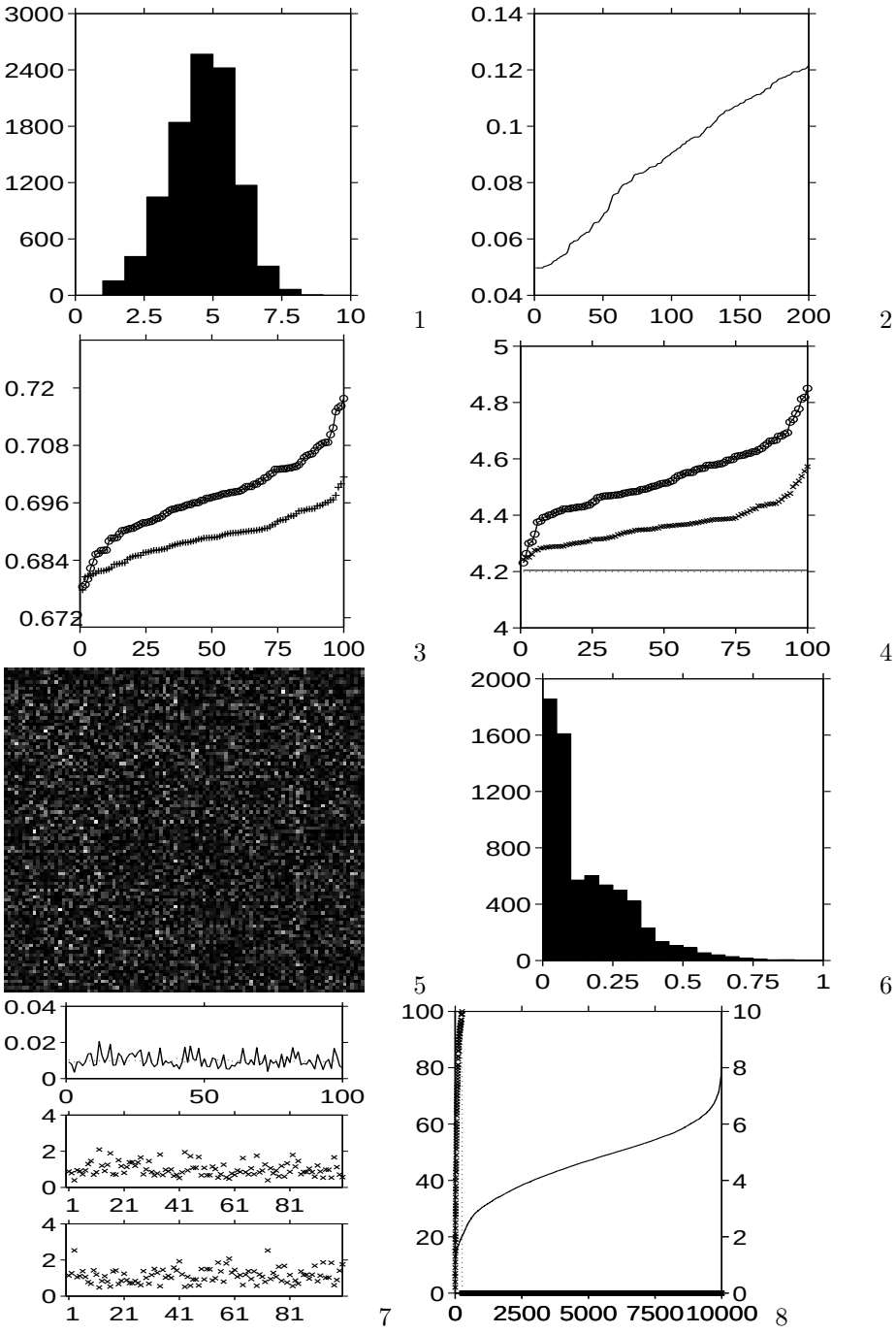


Figure 12: **Large initial activity: statistics of almost random state** For figure details see section 0.4. The transient clustered state has almost completely disappeared after 200 steps.

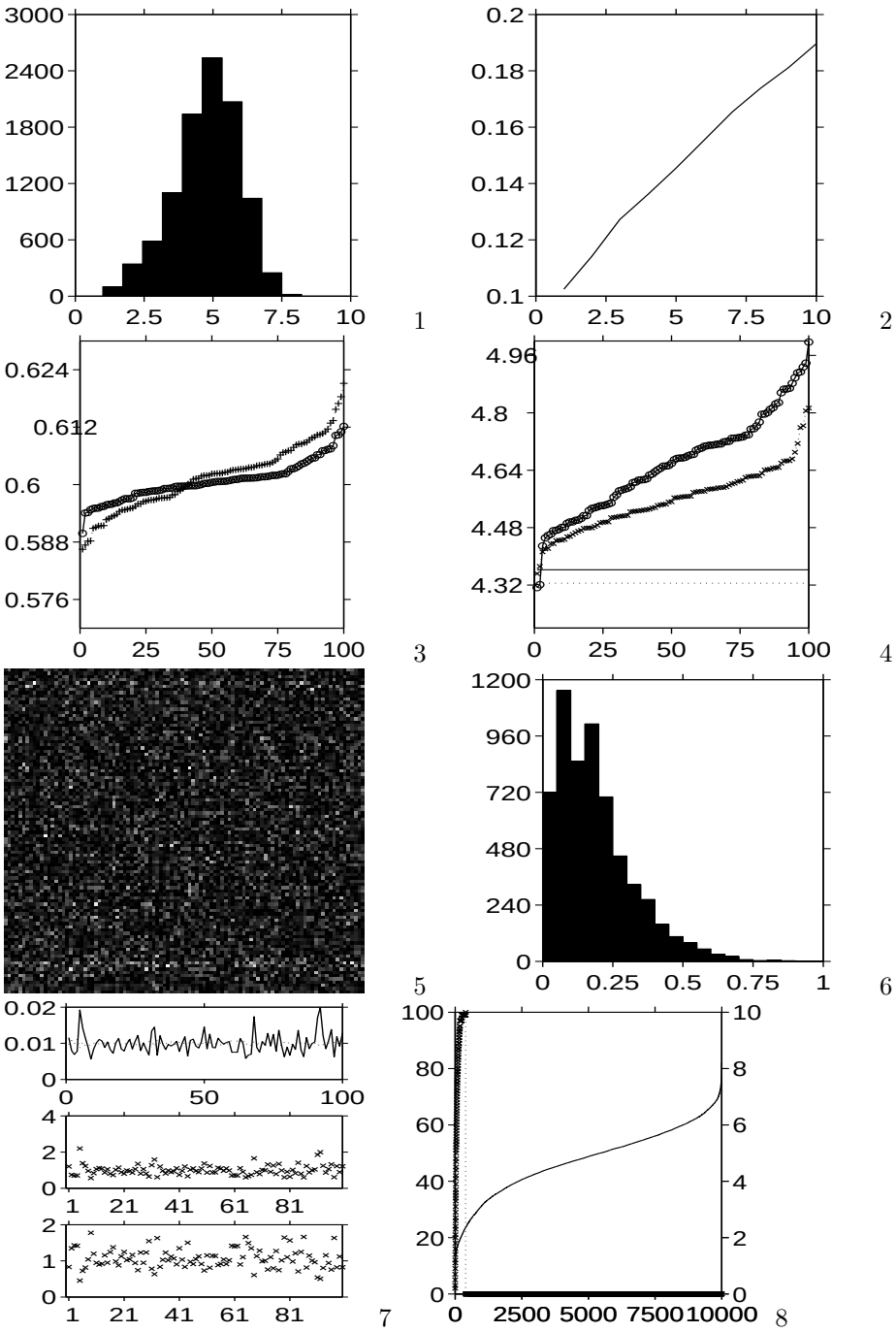


Figure 13: **Large initial activity: random statistics after 500 steps**
For figure details see section 0.4. After 500 steps the net had random statistics.

Kernel function length (L parameter)

We have found that the larger the kernel time window the faster the system evolves to random net (see Figures 14-16). Also when L is increased then the learning factor of the W matrix should be decreased to prevent oscillation because the update values increases as L increases. We used the default parameter values of Table 1, the firing model was the simple threshold model, the kernel function was linearly decreasing.

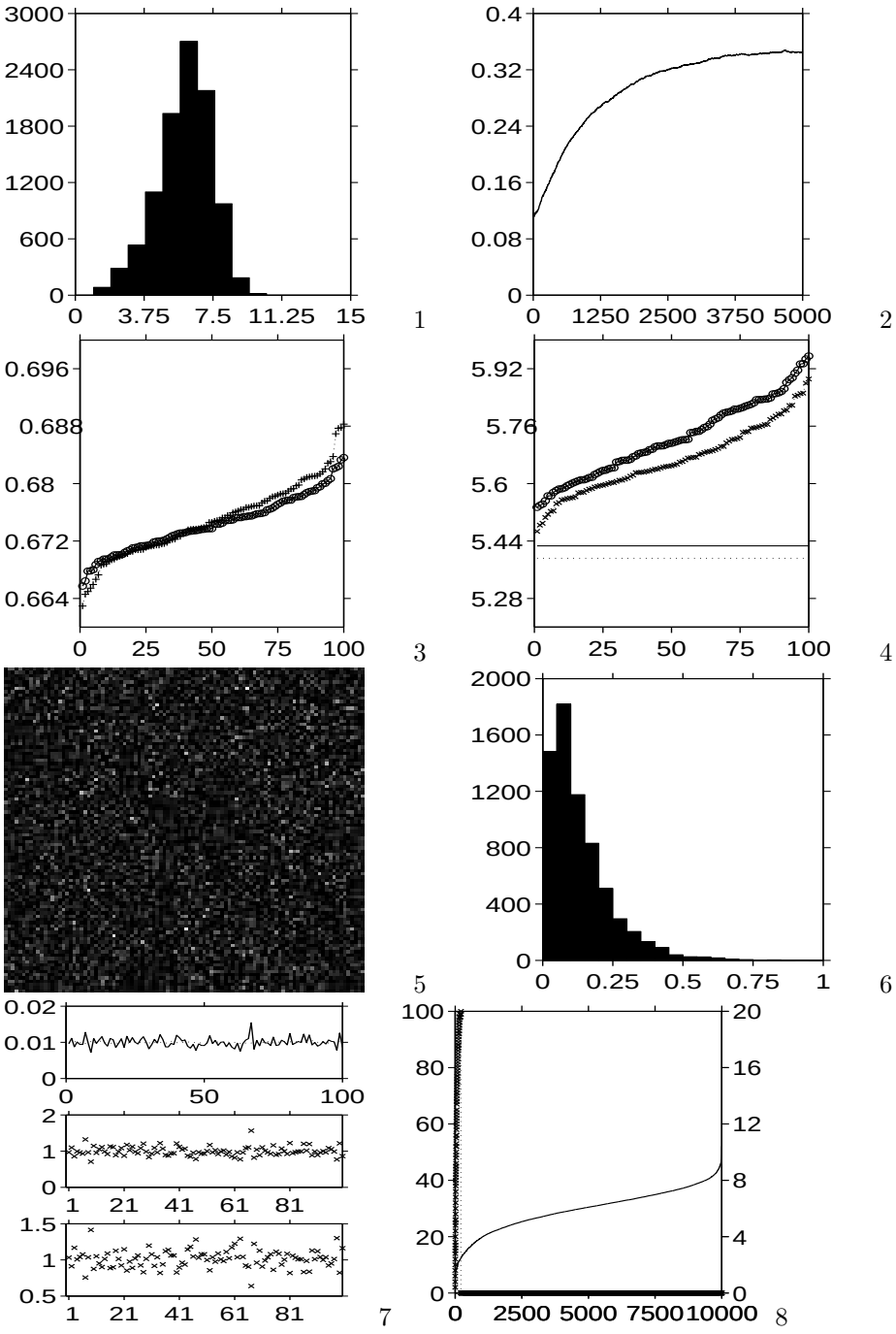


Figure 14: **Analogue:** $L = 2$For figure details see section 0.4.

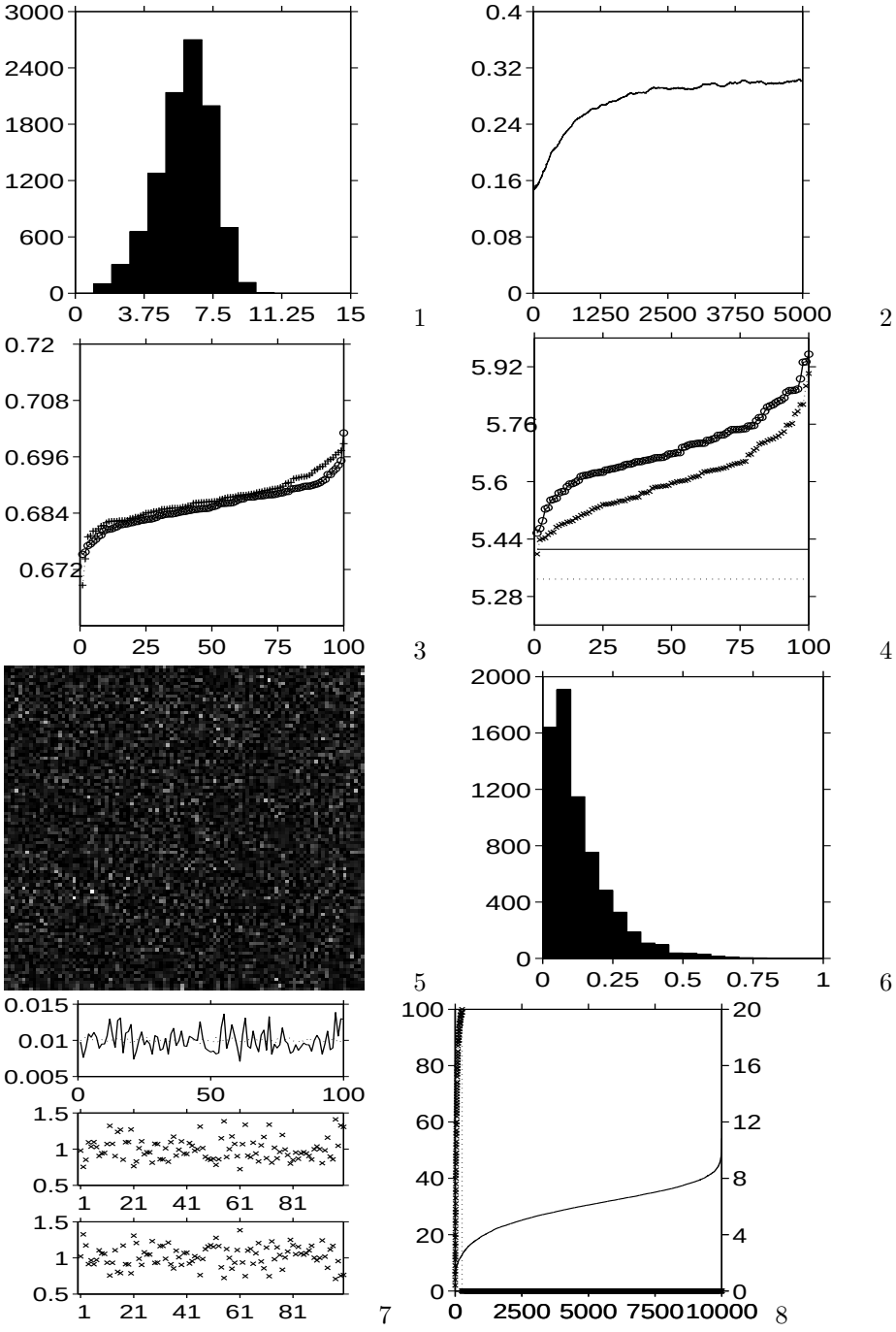


Figure 15: **Analogue:** $L = 4$For figure details see section 0.4.

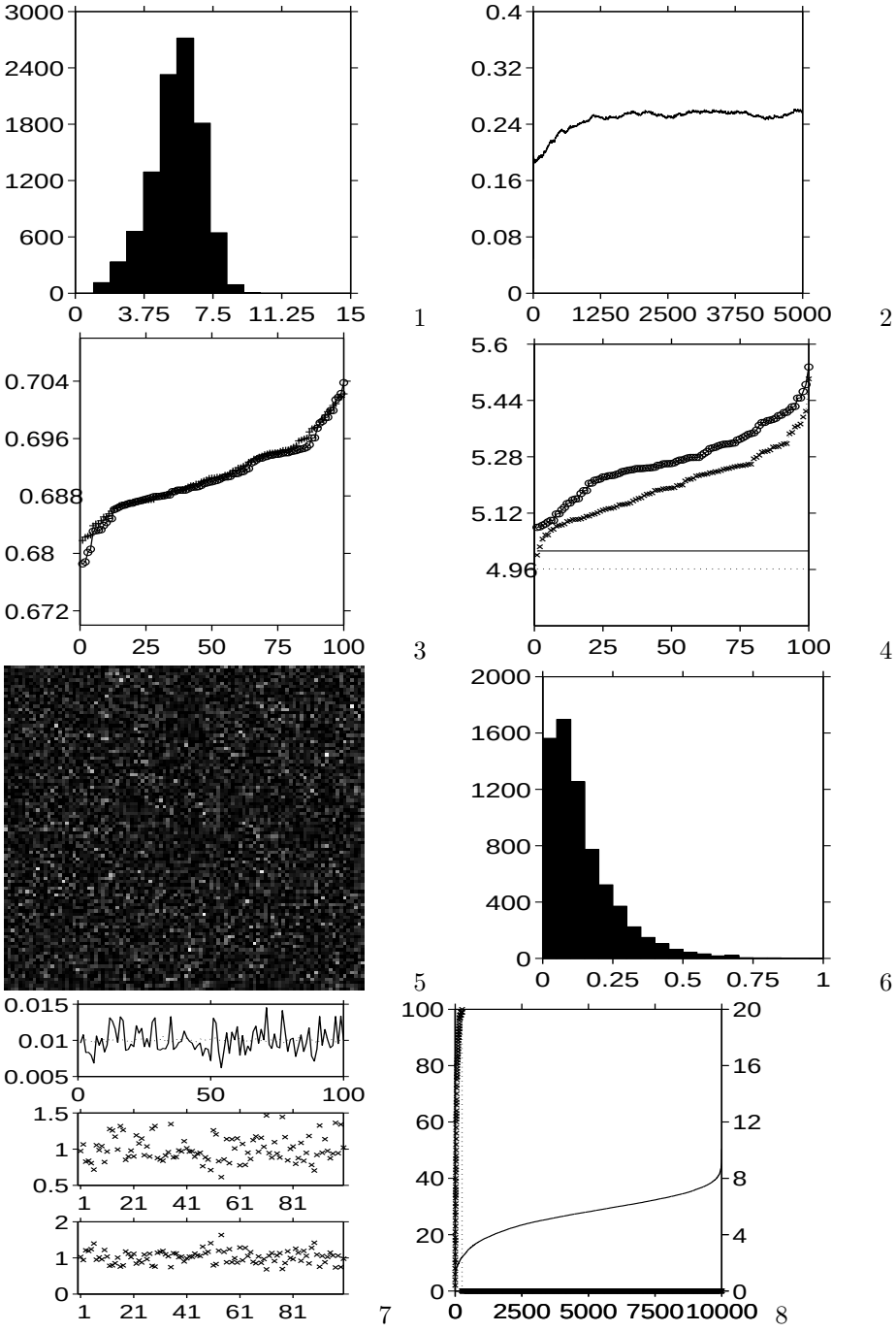


Figure 16: **Analogue:** $L = 8$For figure details see section 0.4.

Ratio of strengthening and weakening (r_A parameter)

Four different behaviors emerged depending on the value of ratio r_A:

- inhibition strength $<< 1 \rightarrow$ exponential growth of weights (see Figure 17
- inhibition strength $\approx 1 \rightarrow$ linear growth of weights(see Figure 18)
- inhibition strength $>> 1$ i.e. 2,4 $\rightarrow$ stable system (see Figure 19)
- inhibition strength $>> 10$ i.e. 50,100.. $\rightarrow$ stable system but very few non zero weights, changing step by step (see Figure 20)

We used the default parameter values of Table 1 except that the firing threshold was set to 0.09, the firing model was the simple threshold model, the kernel function was linearly decreasing.

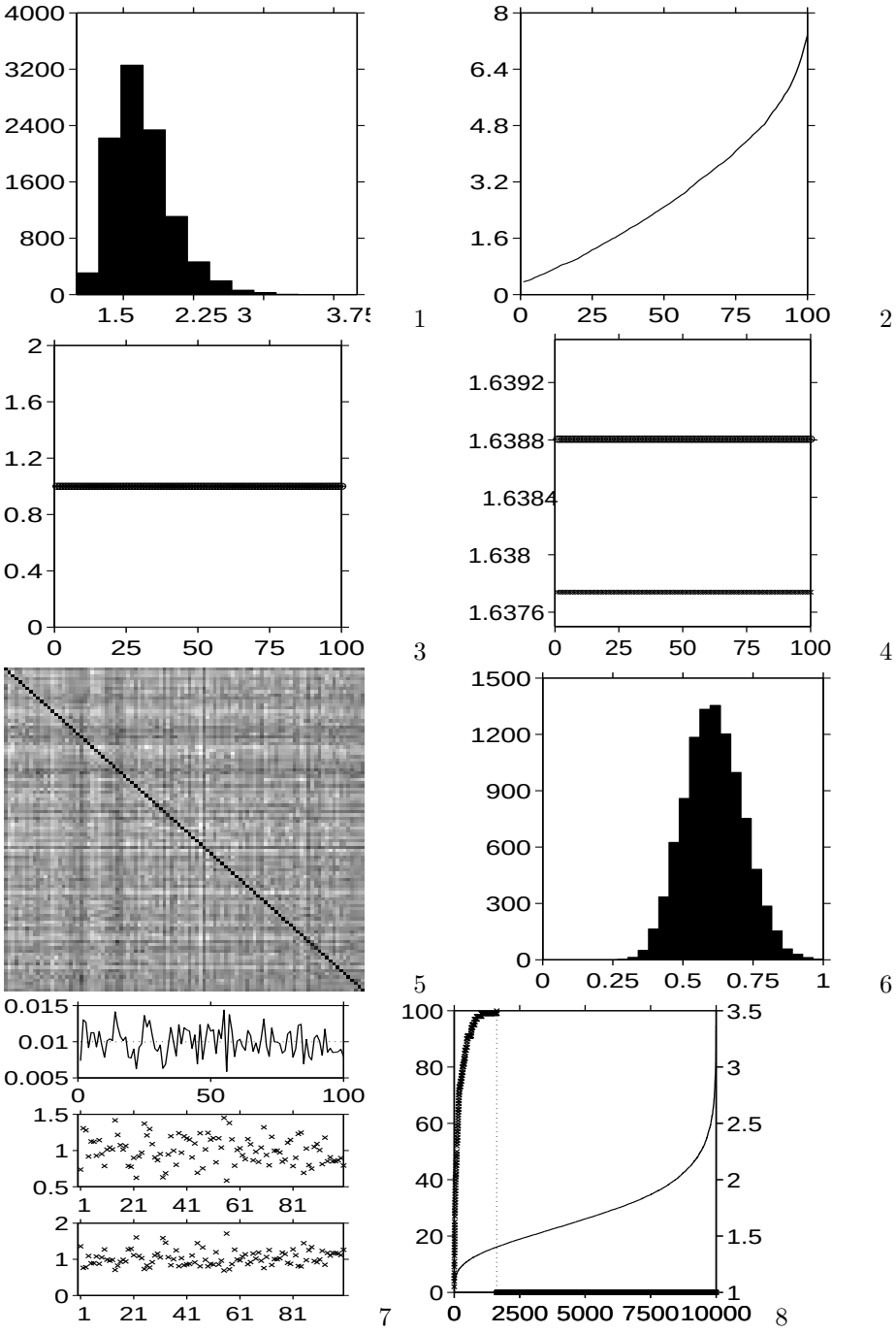


Figure 17: **Analogue:** $r_A = 01$
For figure details see section 0.4. The input ratio (r) was set to 0.3. Because of the exponential growth (2^{nd} plot) *cycle_length* was only 400.

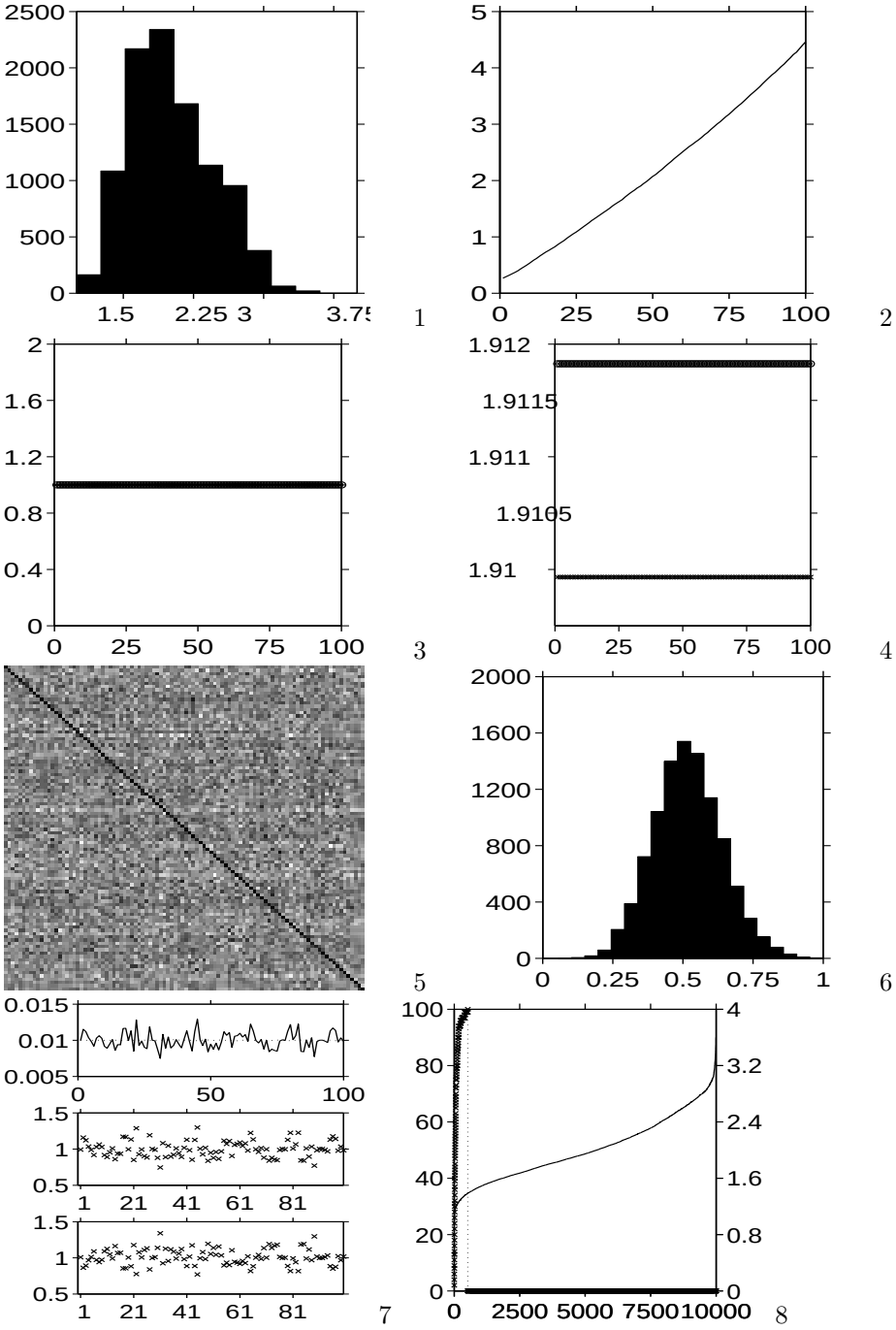


Figure 18: **Analogue:** $r_A = .5$For figure details see section 0.4.

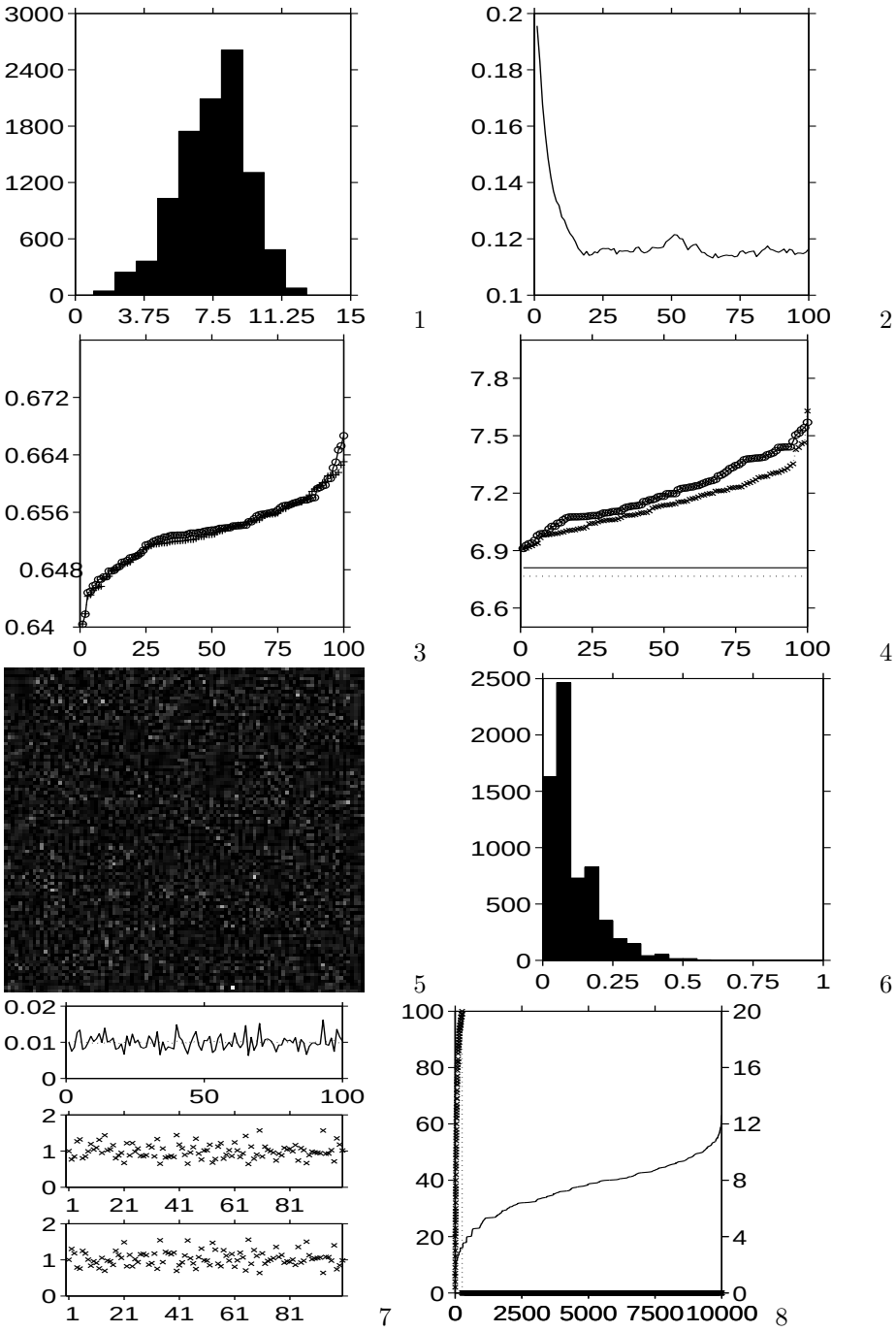


Figure 19: **Analogue:** $r_A = 2$For figure details see section 0.4.

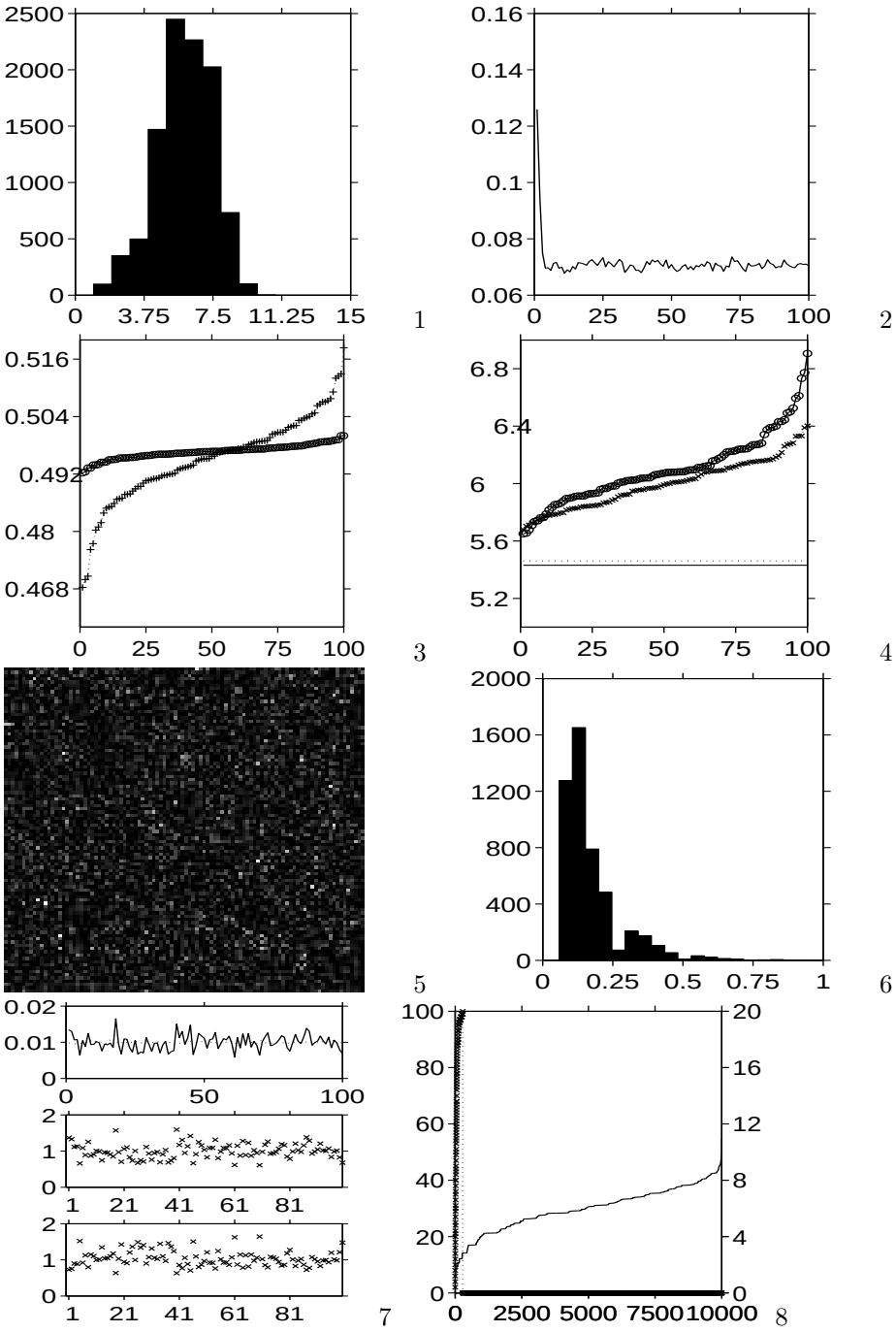


Figure 20: **Analogue:** $r_A = 50$For figure details see section 0.4.

Firing threshold with simple threshold firing model (θ parameter)

We have found 3 different convergence types depending on the threshold value. After eliminating γ the regions disappeared and the 2^{nd} behavior was observed everywhere in the whole parameter region.

1. $\theta < \beta * avg(x^{ext})$ In this case if a neuron got excited externally it can immediately fire. The norm of W is descending from higher value to a stable value (see Figure 21).

2. $\beta * avg(x^{ext} < \theta < 2 * \beta * avg(x^{ext}$ In this case a neuron needs more external excitation or more feedback activity to be able to fire. The norm of W ascending from a lower value to a stable value (see Figure 22).

3. $\theta > 2 * \beta * avg(x^{ext}$. Because of the exponential unlearning only a few will be able to fire. These neurons takes all the activity from other neurons (see Figure 23)

We used the default parameter values of Table 1, the firing model was the simple threshold model, the kernel function was linearly decreasing.

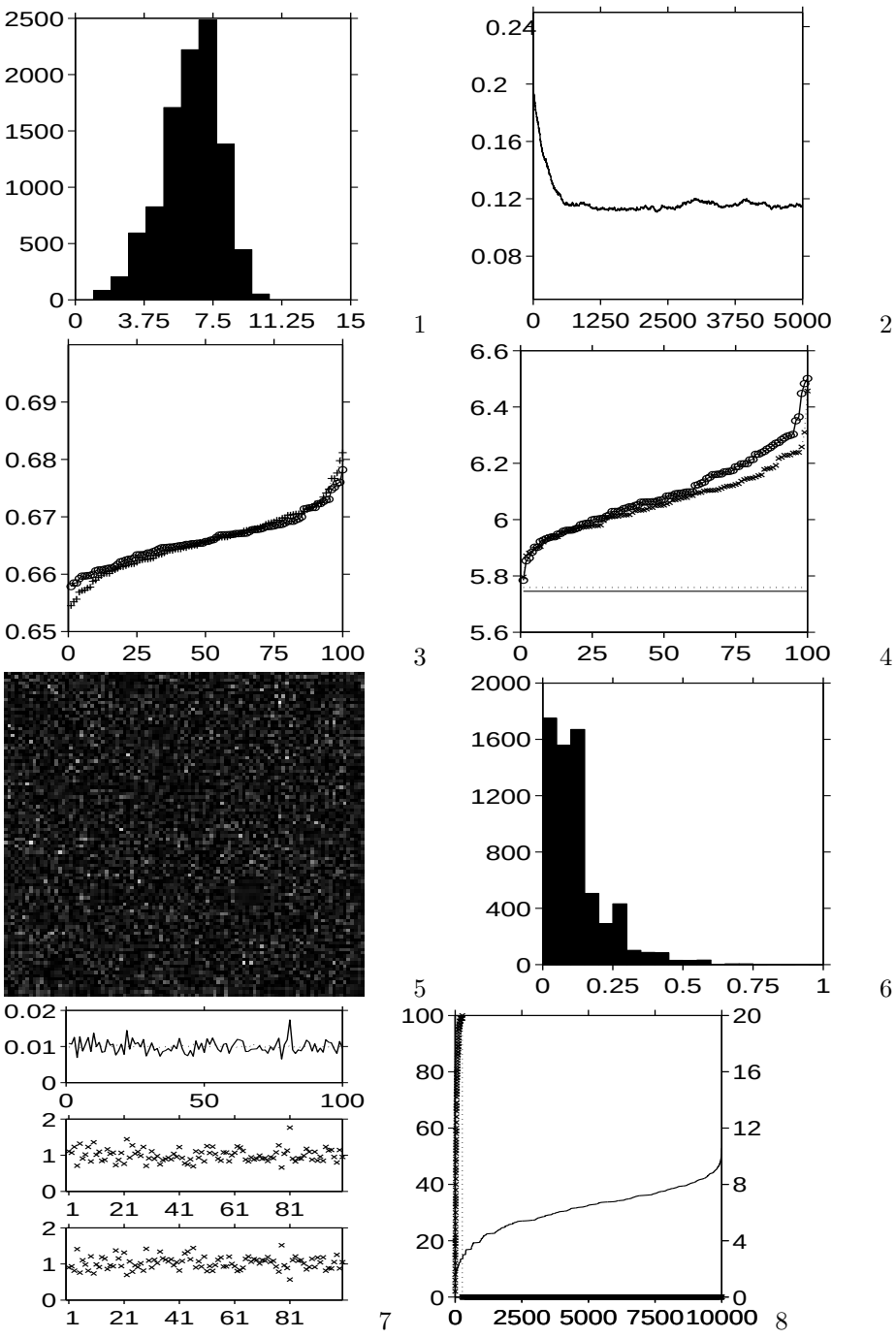


Figure 21: **Analogue:** $\theta = 0.09$For figure details see section 0.4.

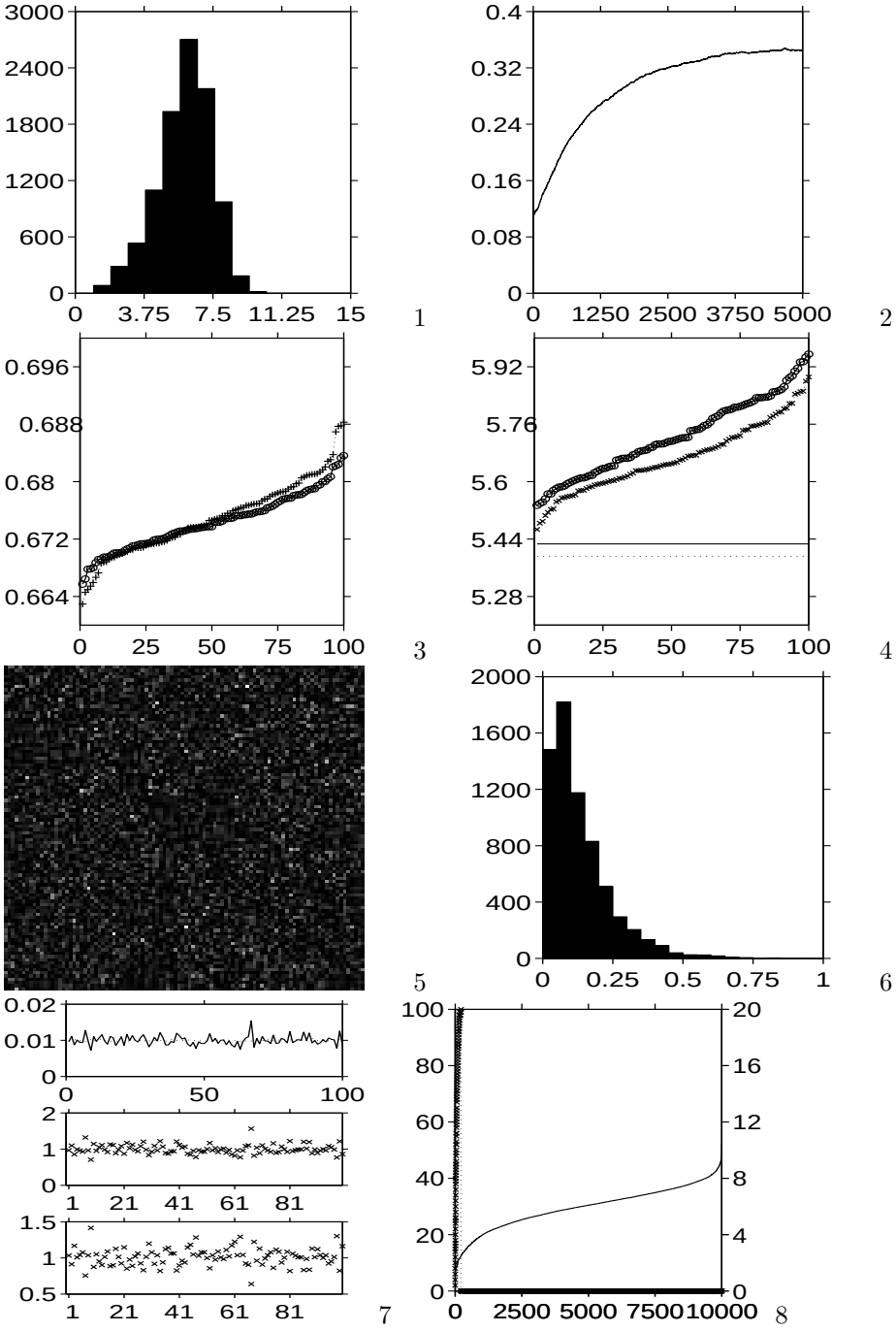


Figure 22: **Analogue:** $\theta = 0.13$For figure details see section 0.4.

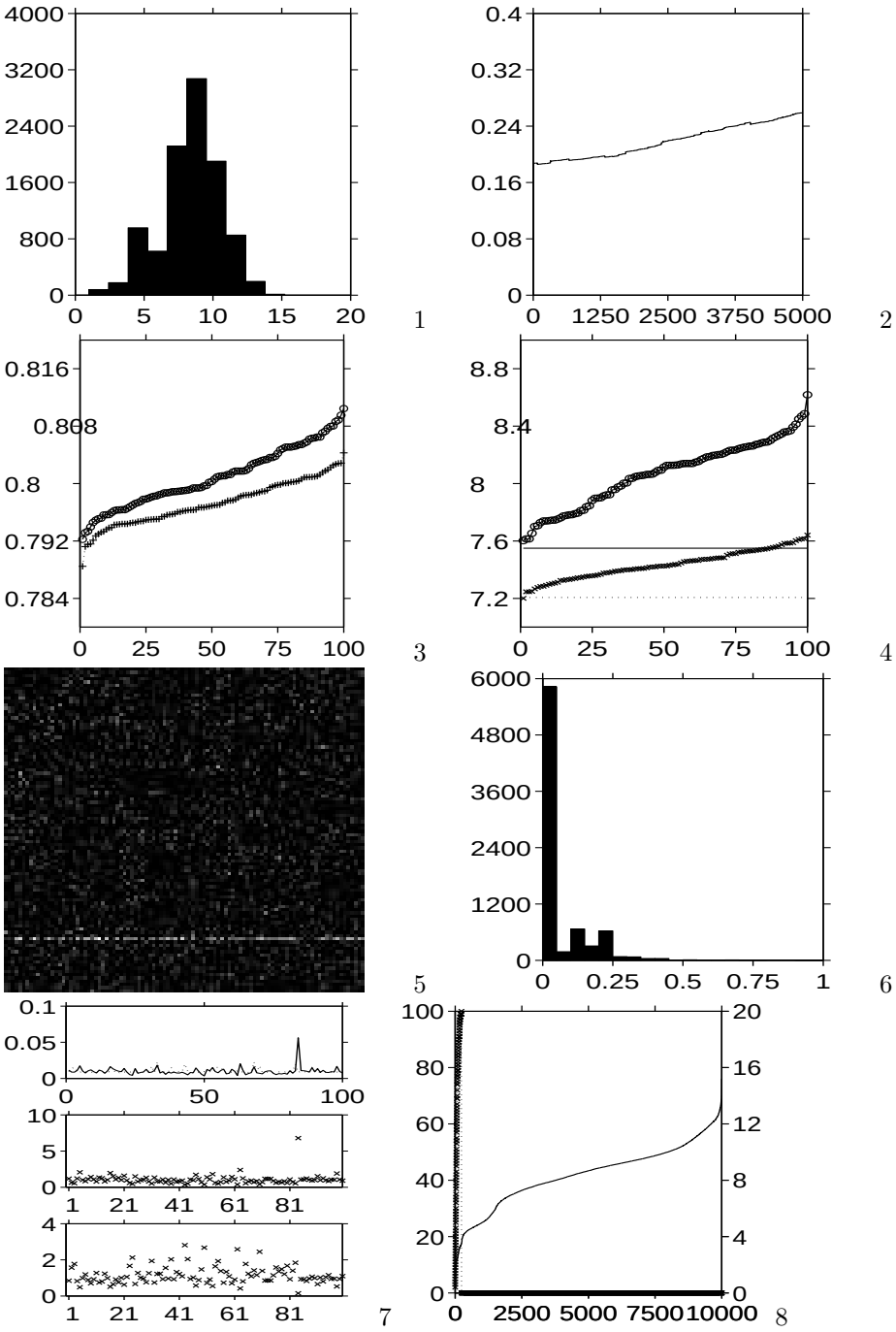


Figure 23: **Analogue:** $\theta = 0.27$For figure details see section 0.4.

firing models

The probabilistic firing model with threshold produced results like the simple threshold firing model with $\theta = 0.09$. The reason for it could be that in this case all neuron can fire with a nonzero probability (except those have exactly 0 inner activity), so relatively many neurons may fire (see Figures 24 - 26). We kept the simple threshold firing model because of its simplicity. We used the default parameter values of Table 1, the kernel function was linearly decreasing.

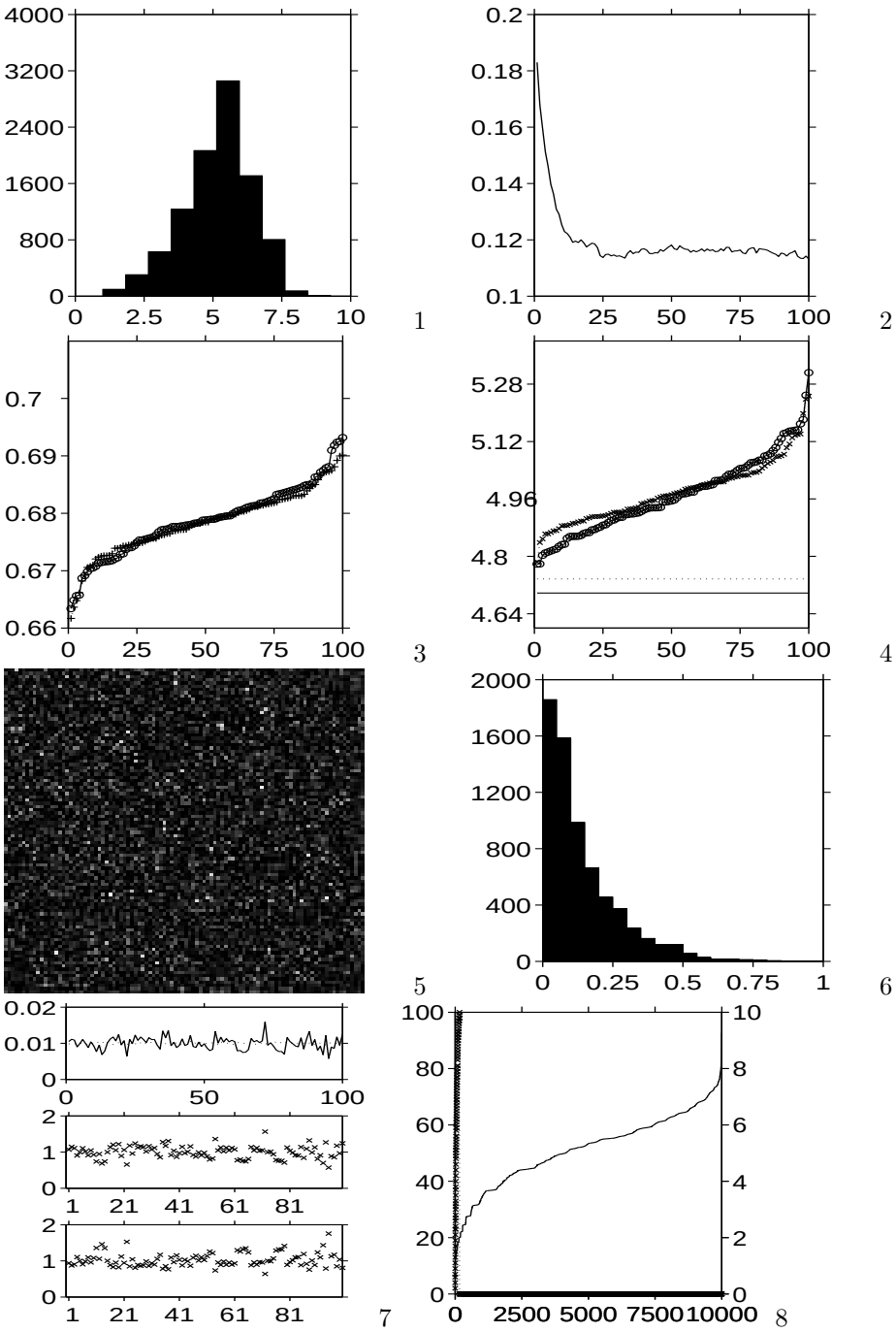


Figure 24: **Analogue: probabilistic firing model with threshold,** $\theta = 0.09$
For figure details see section 0.4.

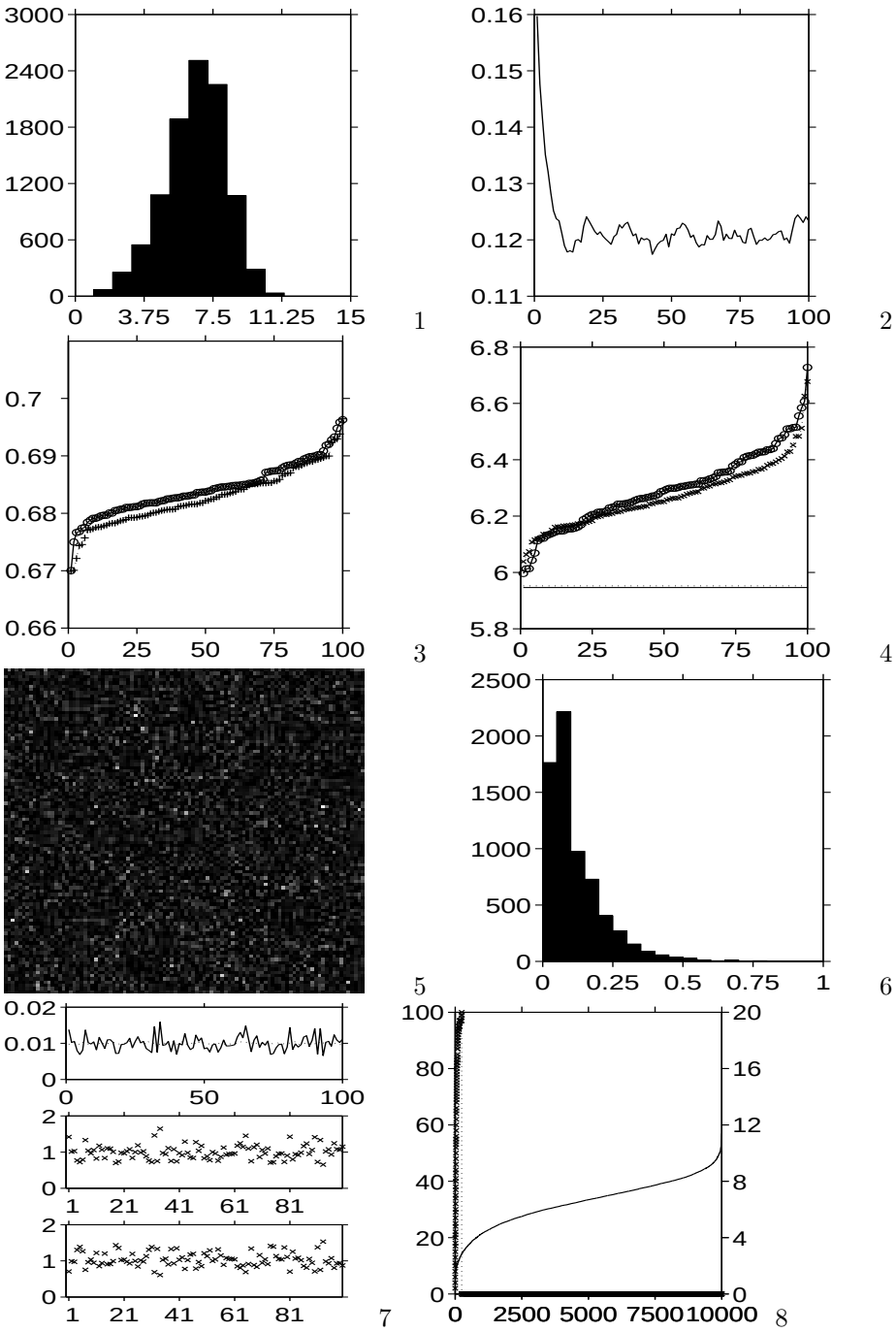


Figure 25: **Analogue: probabilistic firing model with threshold,** $\theta = 0.13$
For figure details see section 0.4.

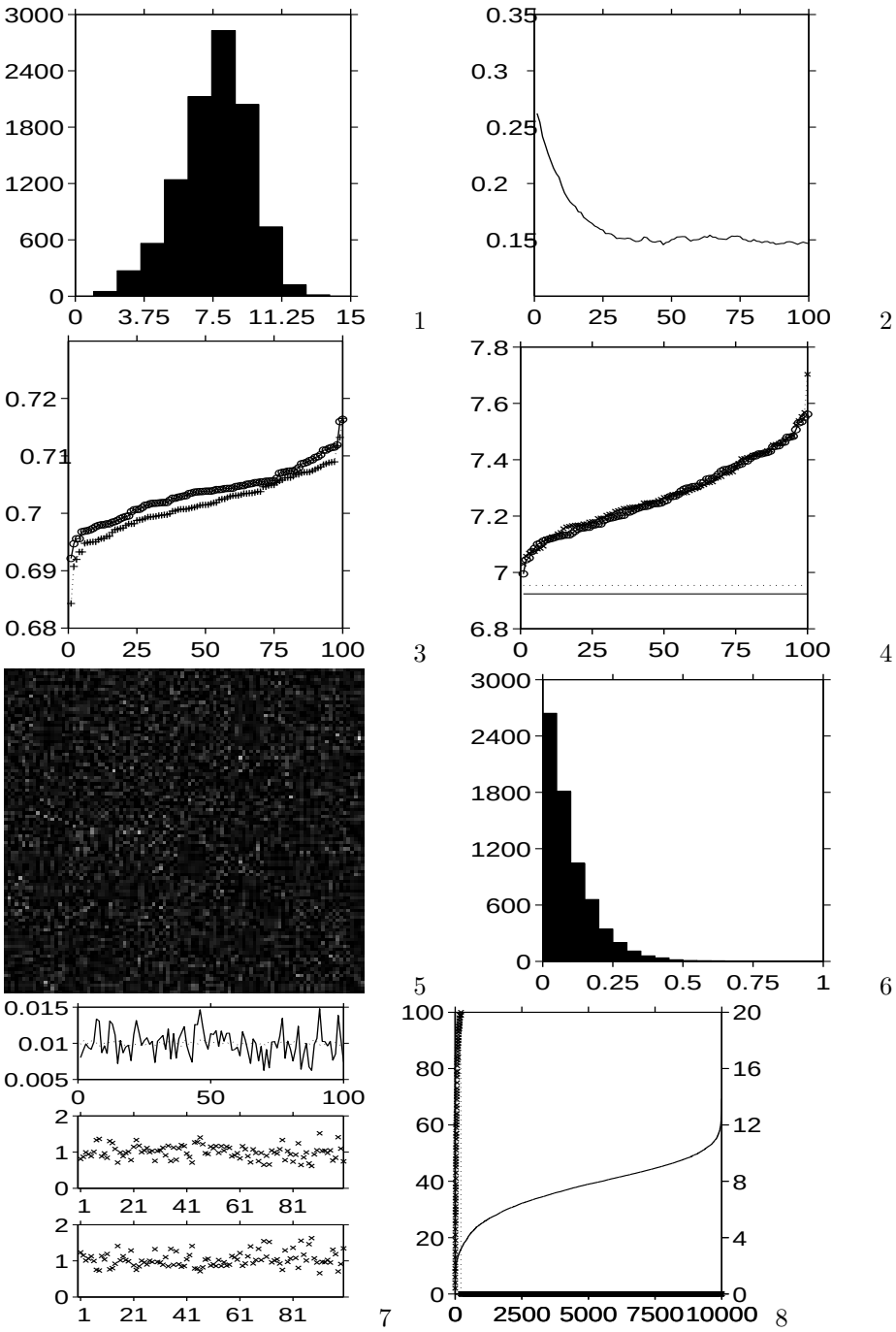


Figure 26: **Analogue: probabilistic firing model with threshold,** $\theta = 0.27$
For figure details see section 0.4.

Different kernels

There was no significant difference between the results with different kernels. See figures 27 - 37. We used the default parameter values of Table 1, the firing model is the simple threshold one.

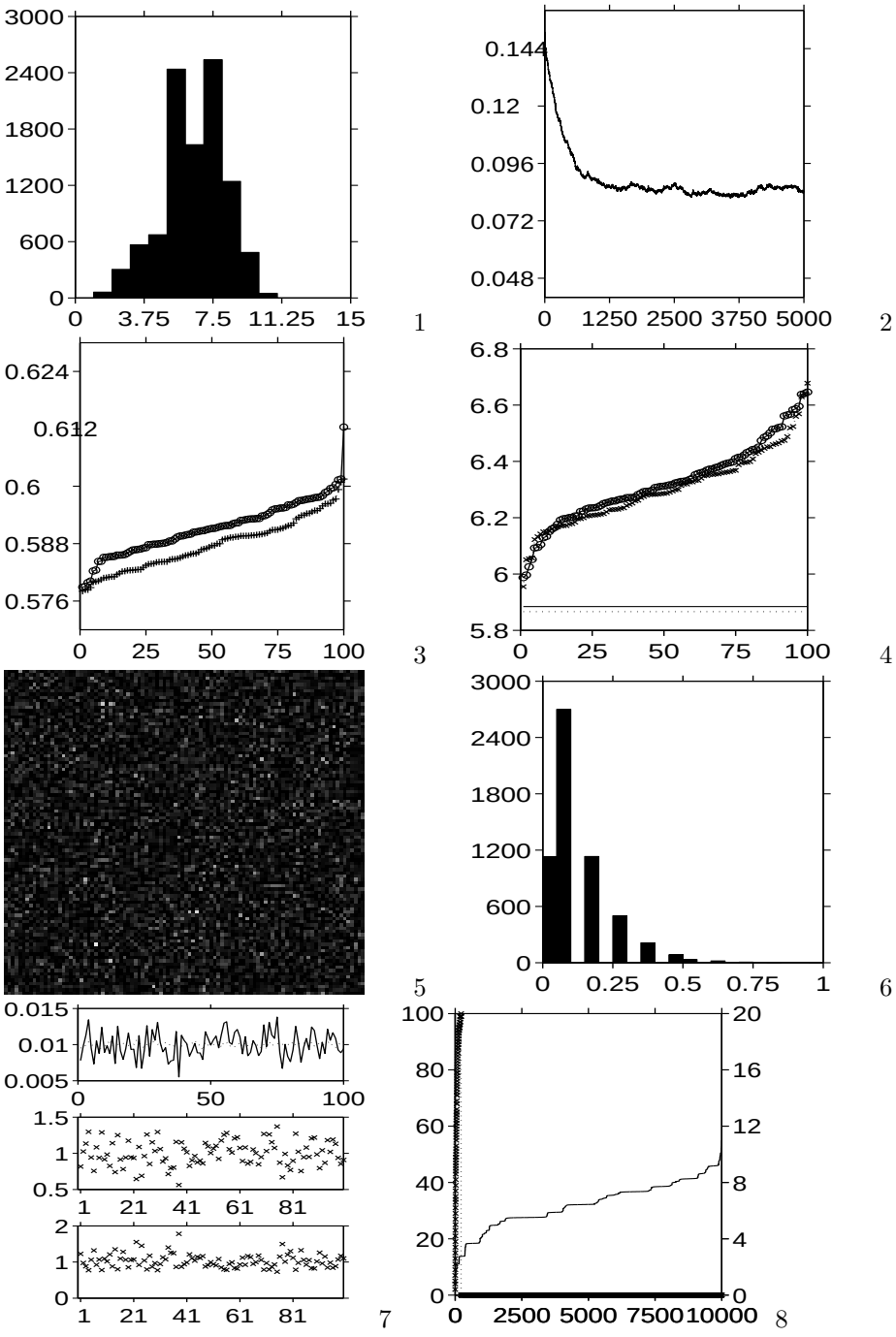


Figure 27: **Analogue: asymmetric time window,** $\theta = 0.09$
For figure details see section 0.4.

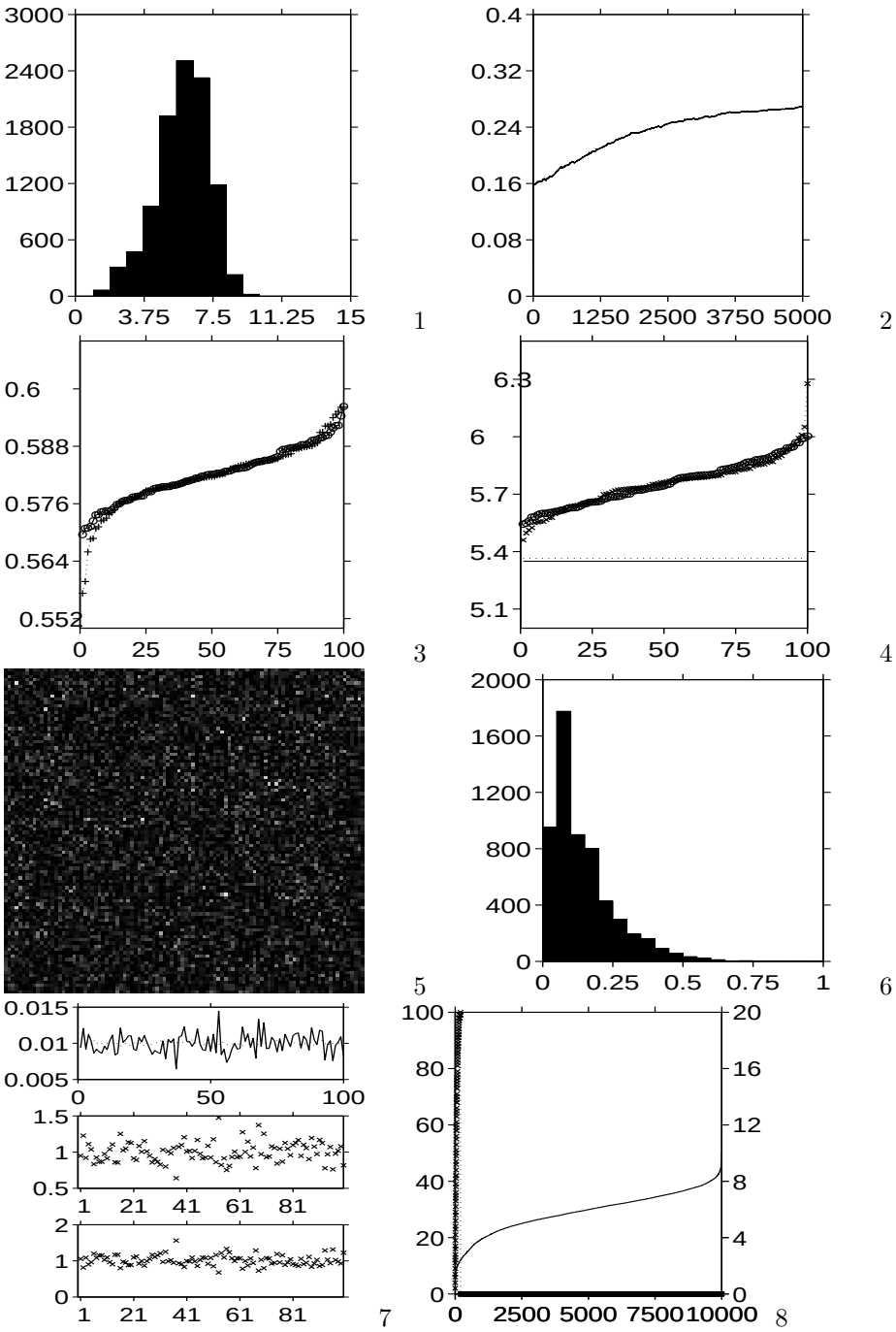


Figure 28: **Analogue: asymmetric time window,** $\theta = 0.13$
For figure details see section 0.4.

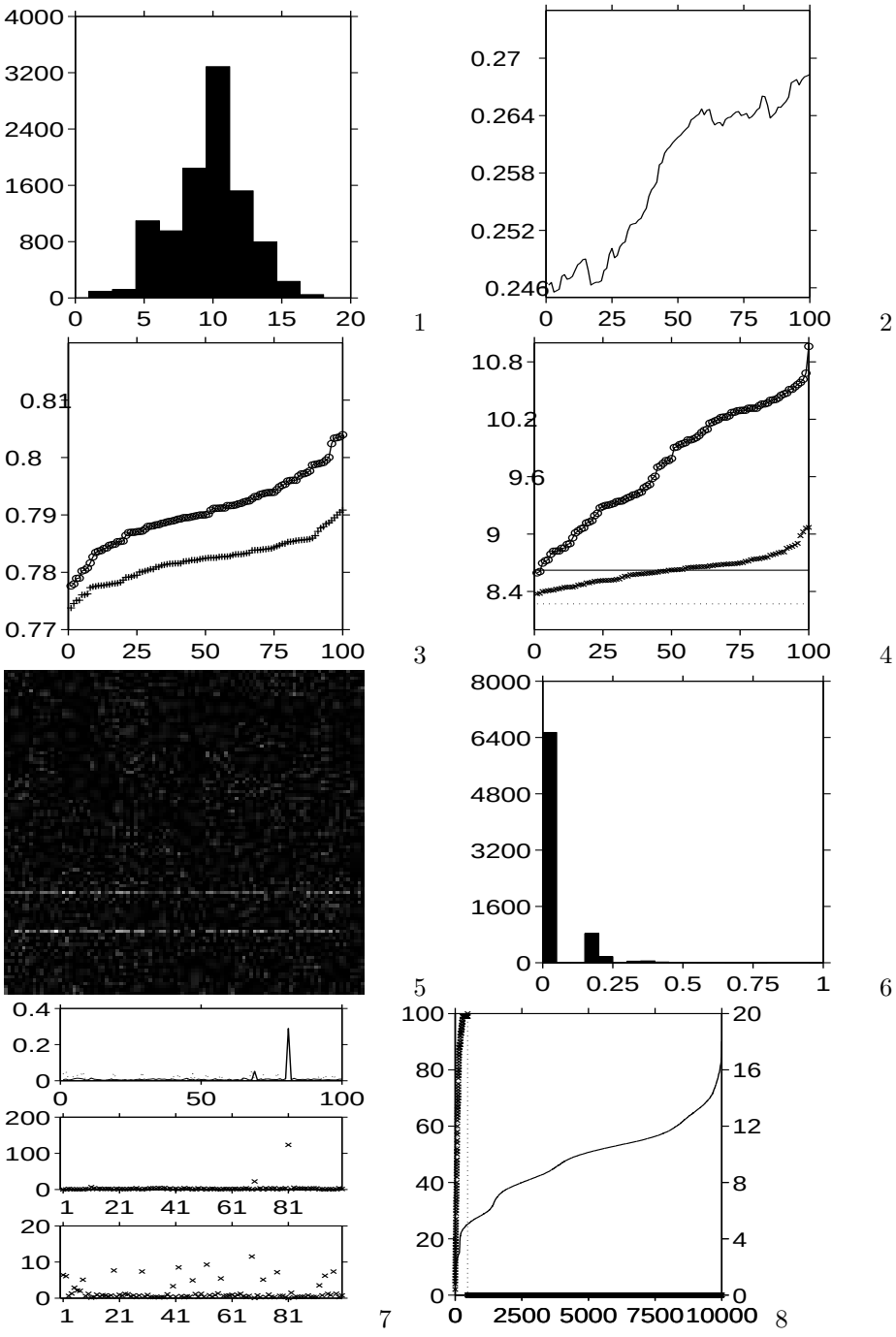


Figure 29: **Analogue: asymmetric time window,** $\theta = 0.27$
For figure details see section 0.4.

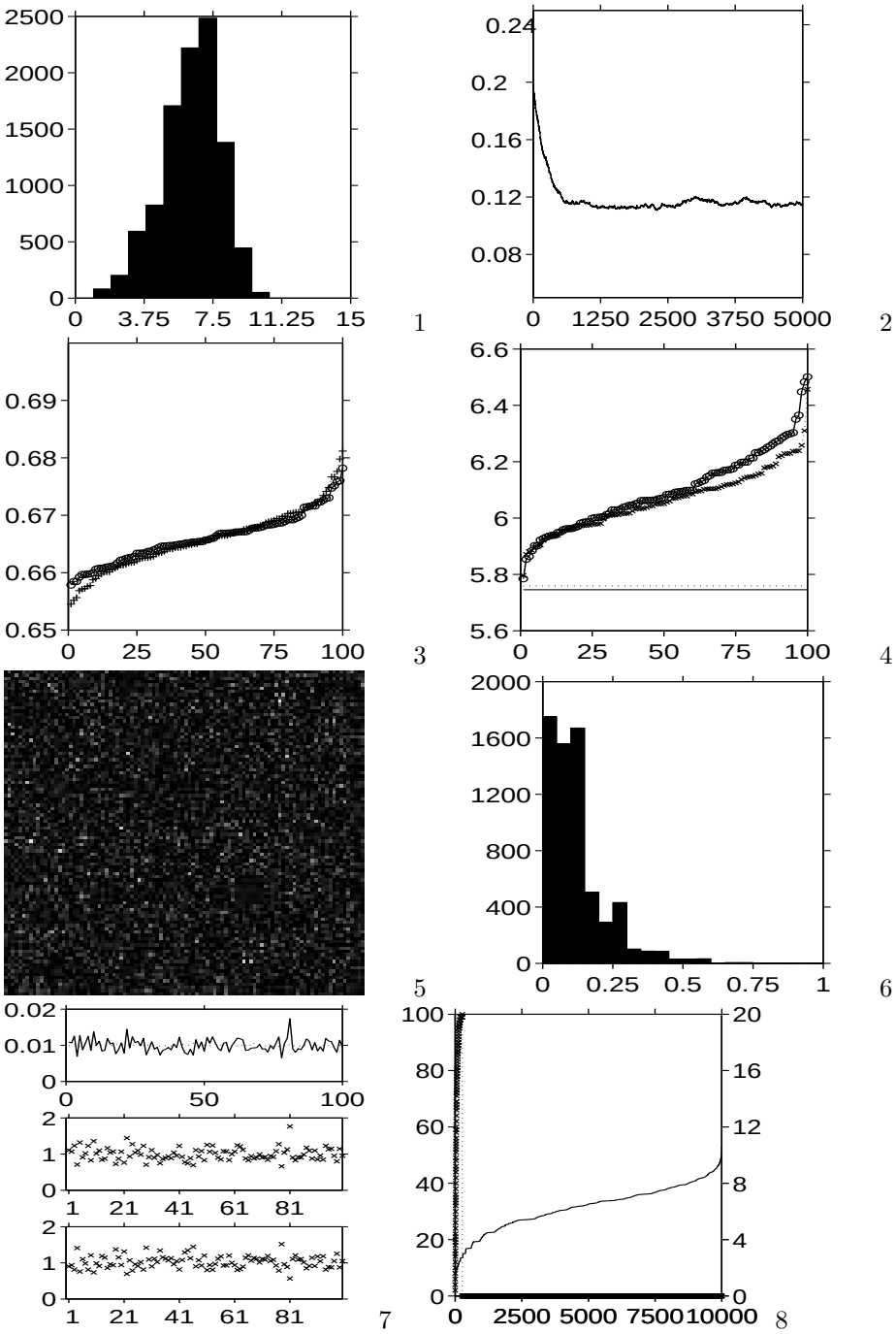


Figure 30: **Analogue: linearly decreasing time window,** $\theta = 0.09$
For figure details see section 0.4.

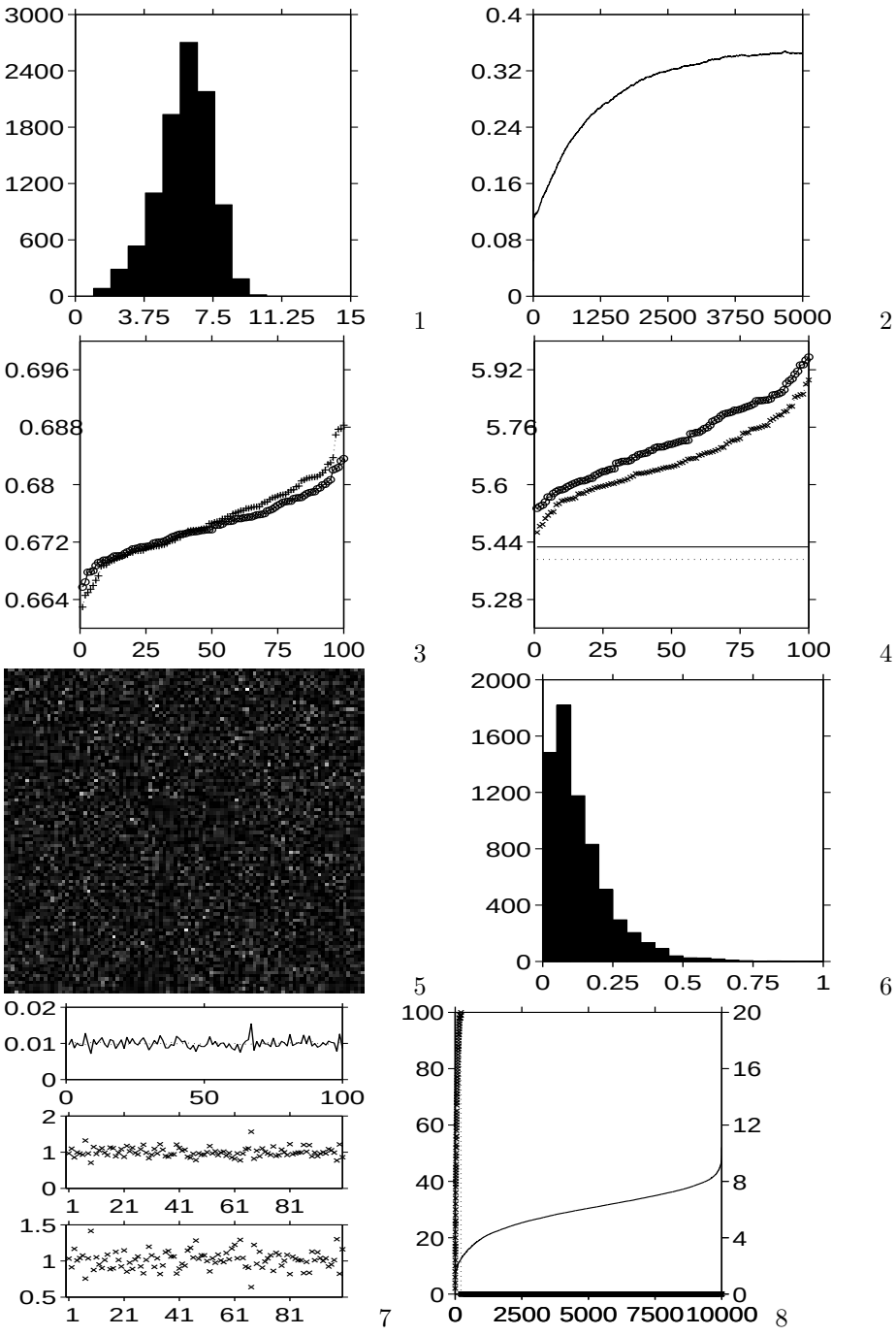


Figure 31: **Analogue: linearly decreasing time window,** $\theta = 0.13$
For figure details see section 0.4.

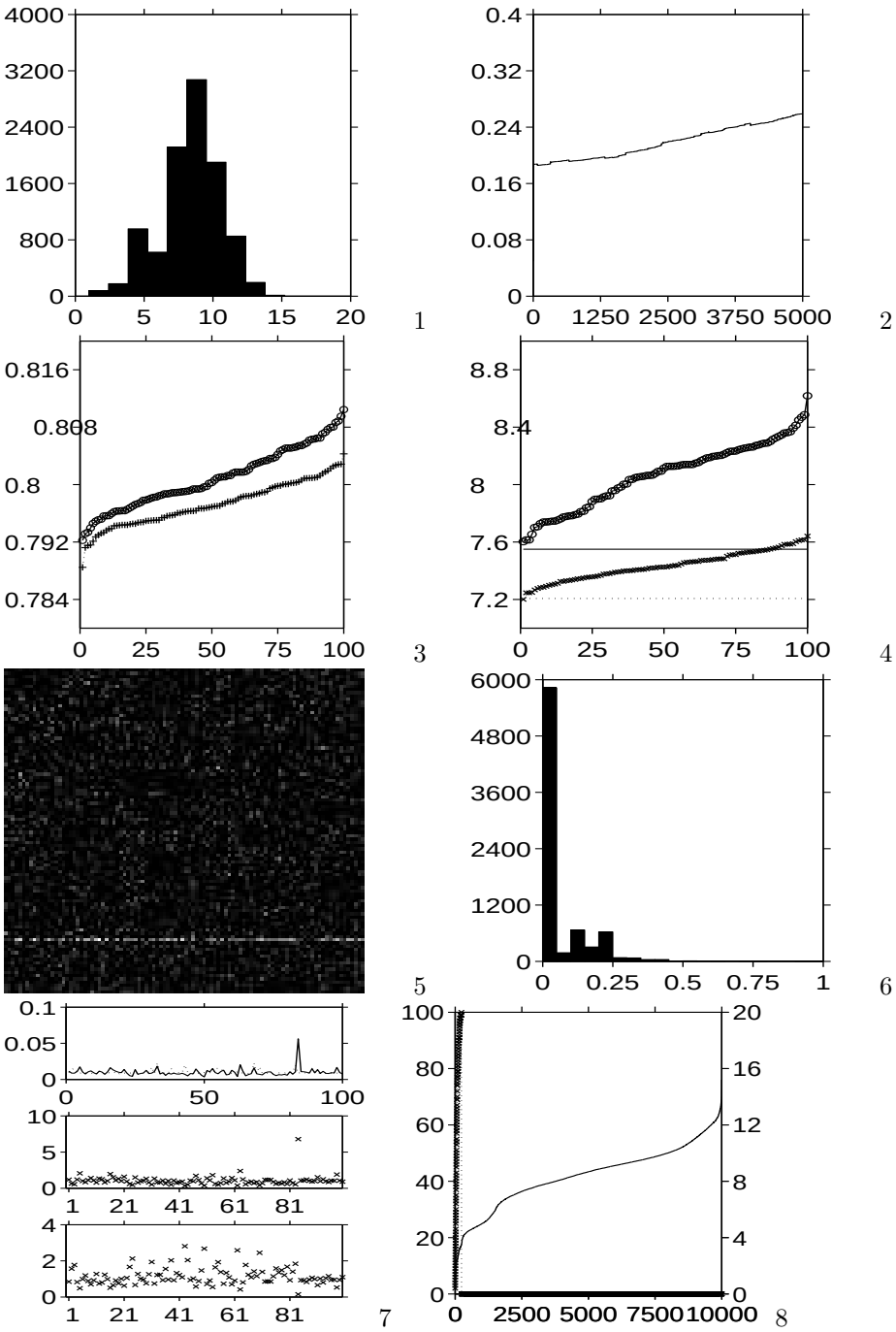
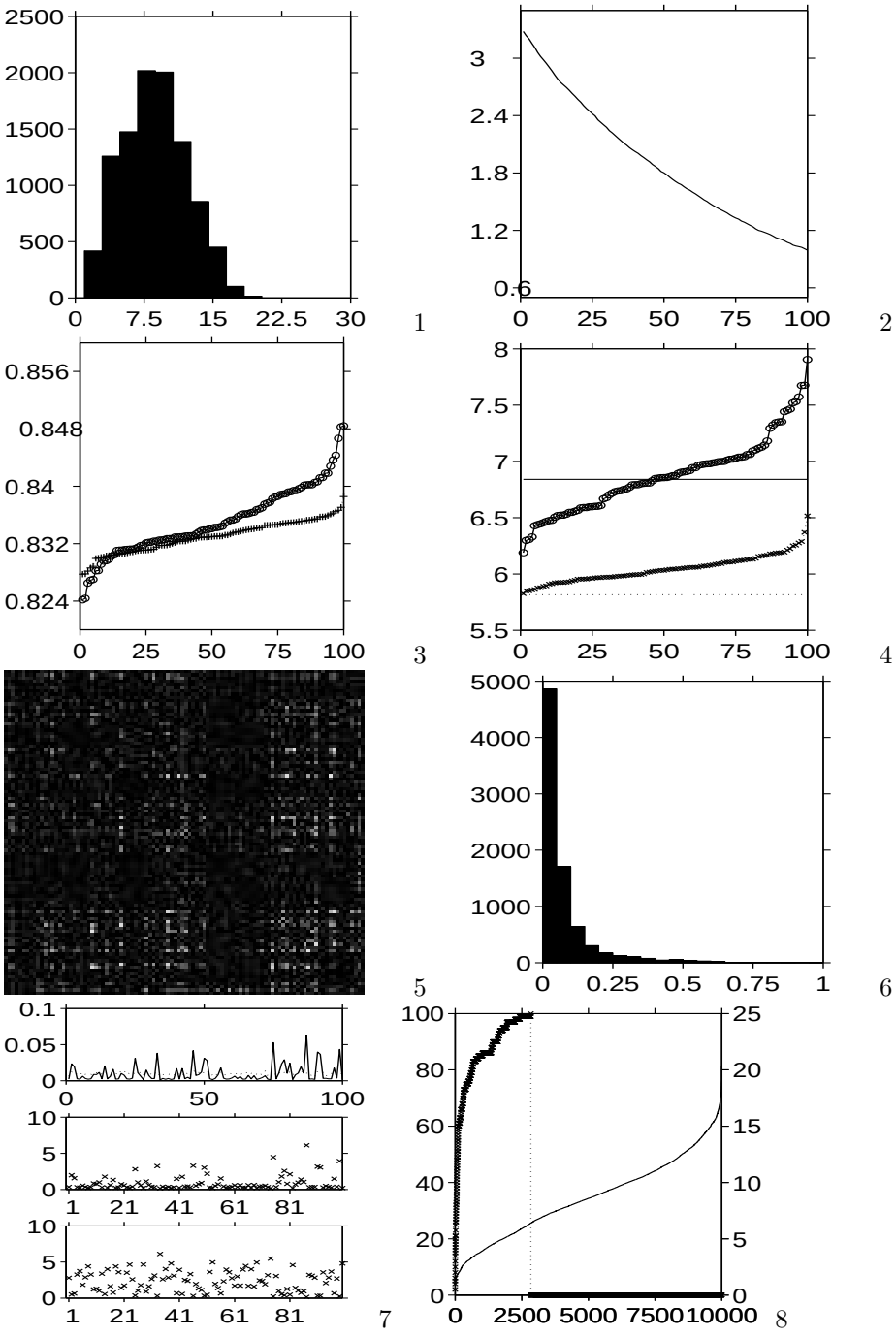


Figure 32: **Analogue: linearly decreasing time window,** $\theta = 0.27$
For figure details see section 0.4.



5

Figure 33: **Analogue: symmetric time window,** $\theta = 0.09$
For figure details see section 0.4.

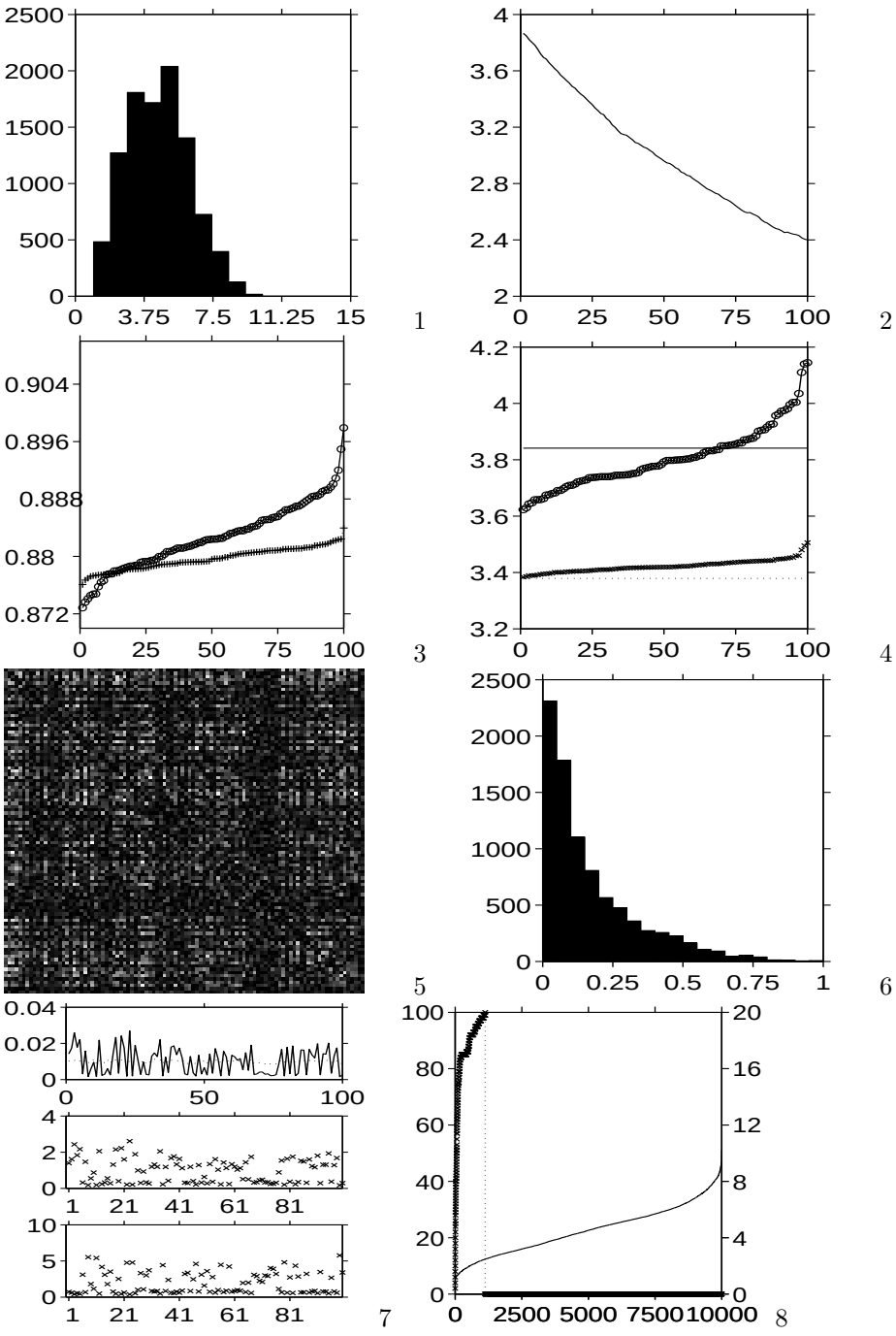


Figure 34: **Analogue: symmetric time window,** $\theta = 0.13$
For figure details see section 0.4.

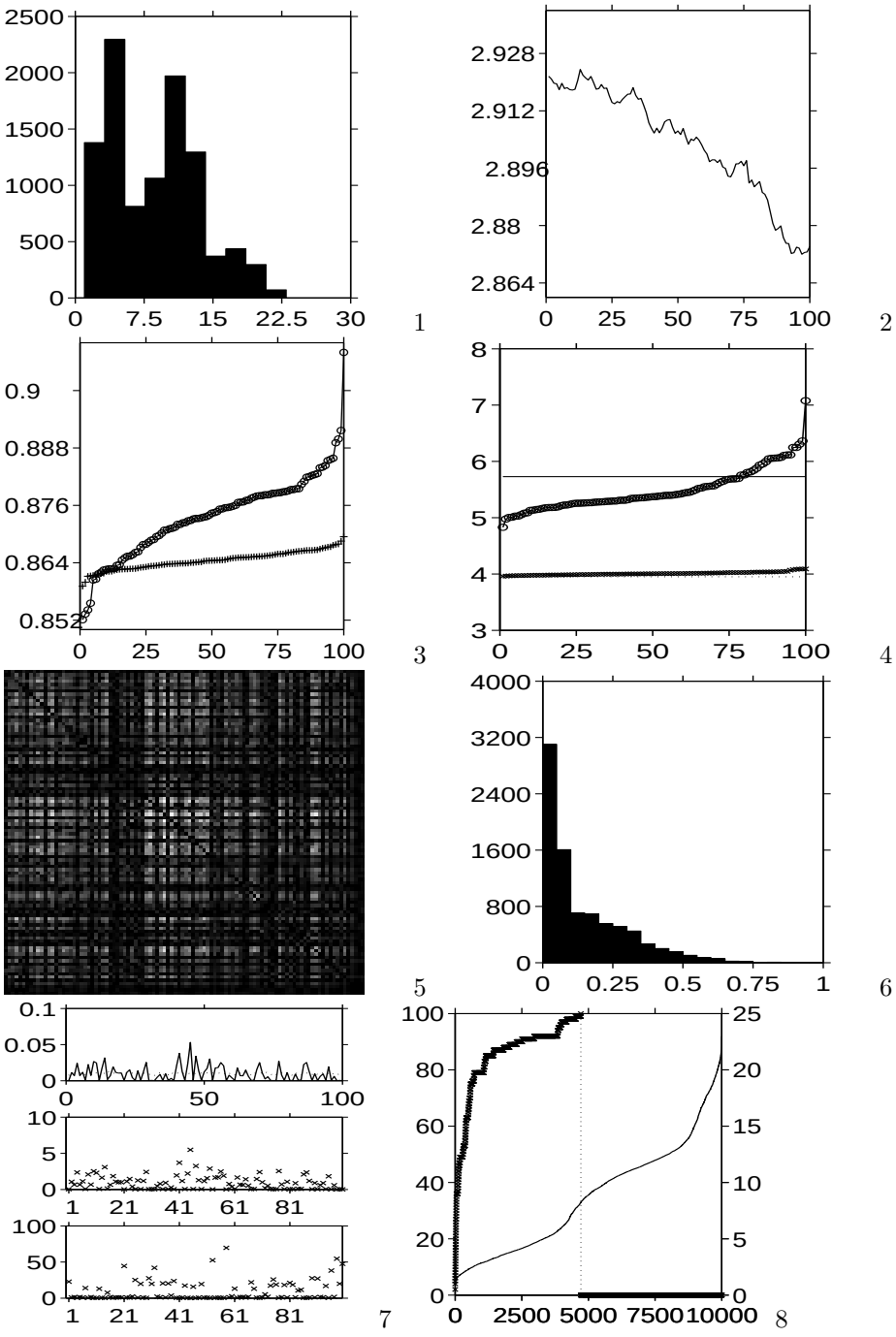


Figure 35: **Analogue: symmetric time window,** $\theta = 0.27$
For figure details see section 0.4.

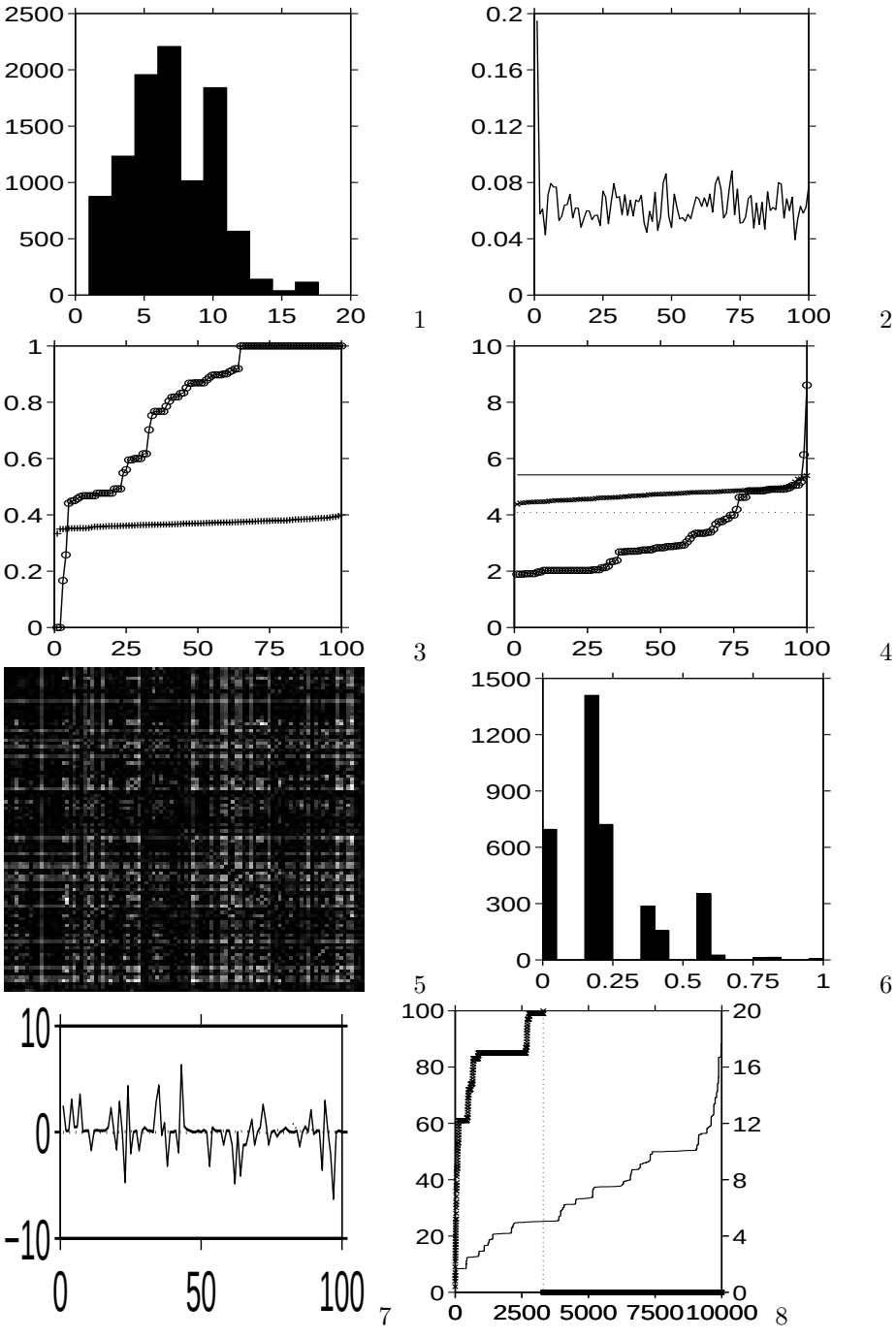


Figure 36: **Analogue: symmetric time window,** $\theta = 0.01$**,** $r = 0.8$
For figure details see section 0.4.In the spike like firing model the system tried to learn some synchronous behavior, here this property is not really characteristic.

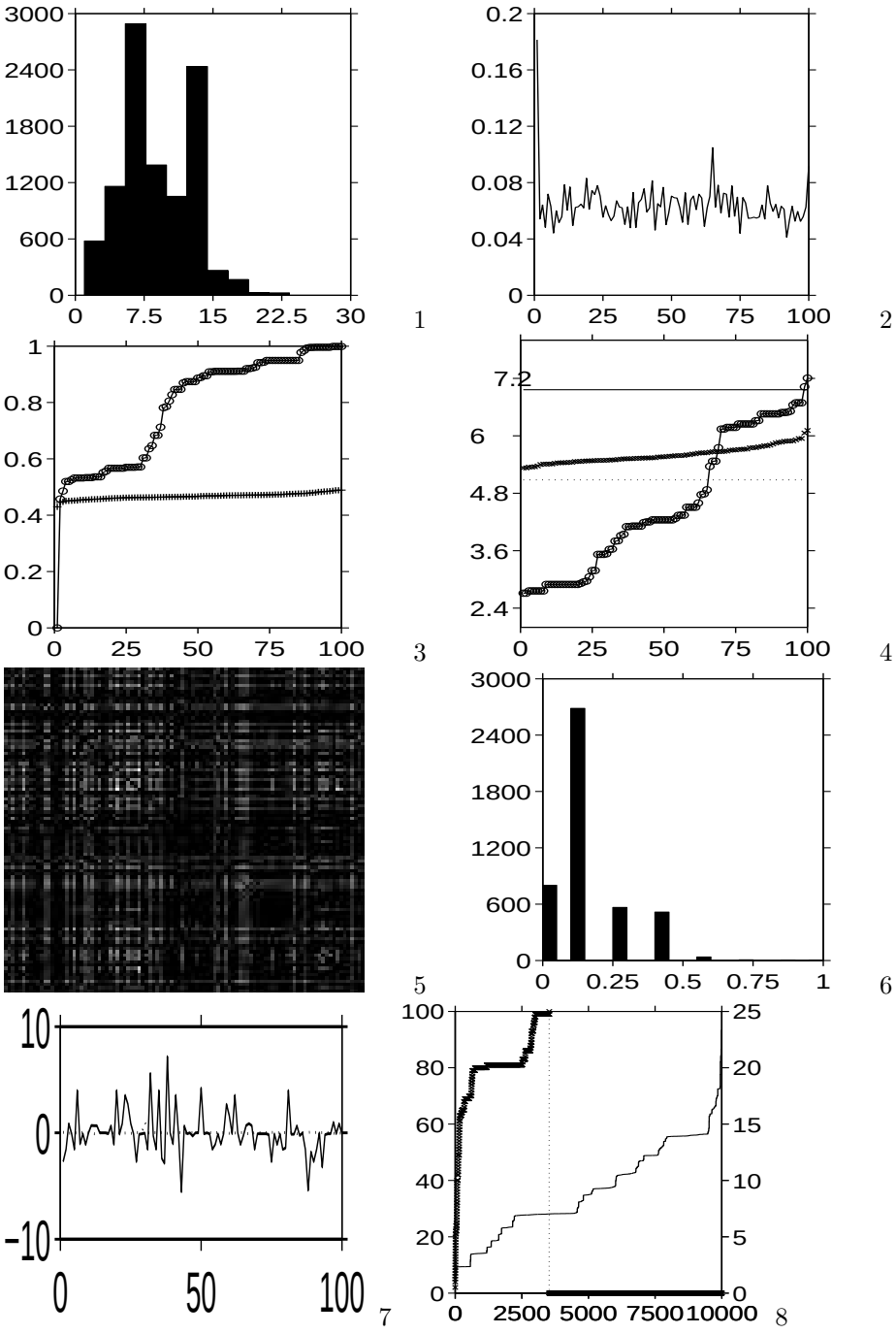


Figure 37: **Analogue: symmetric time window,** $\theta = 0.09$**,** $r = 0.8$
For figure details see section 0.4.In the spike like firing model the system tried to learn some synchronous behavior, here this property is not really characteristic.

moving sine valued input

The system learned a moving wave (see Figure 38). Default parameter values have been used, except for $\theta = 0.5$, the firing model is the probabilistic one with threshold, the kernel is linearly decreasing.

0.5.2 Spike like firing

With the symmetric kernel the weight matrix will be symmetric this means that the connections between the nodes are undirected.

Feedback layer

At first we investigated the symmetric time window and random input and probabilistic firing with fixed average number of firing neurons. We have found that the most important parameters are the ratio of external excitation (r) and the ratio of the average number of firing neurons (θ). We have found two different regions:

- **few neurons get external excitation or few neurons able to fire:** the system evolves to random network (see Figures 39 - 41).
- **more than half of neurons get external excitation and more than half of neurons able to fire:** the system learns some synchronous sequence with fast change due to random input (see Figure 42).

We used the default parameter values of Table 2 except for θ and r.

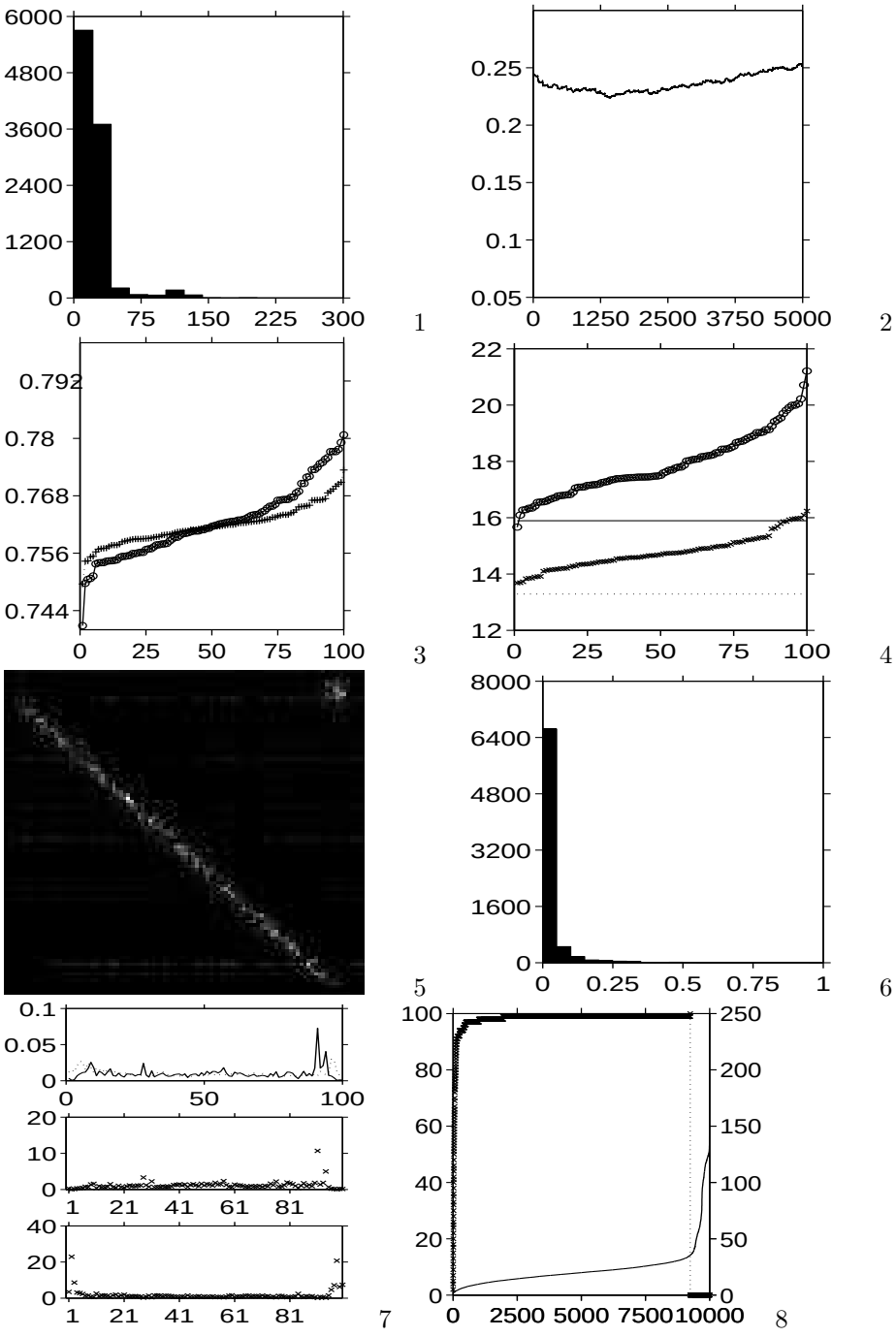


Figure 38: **Analogue: moving sine valued input**
For figure details see section 0.4.

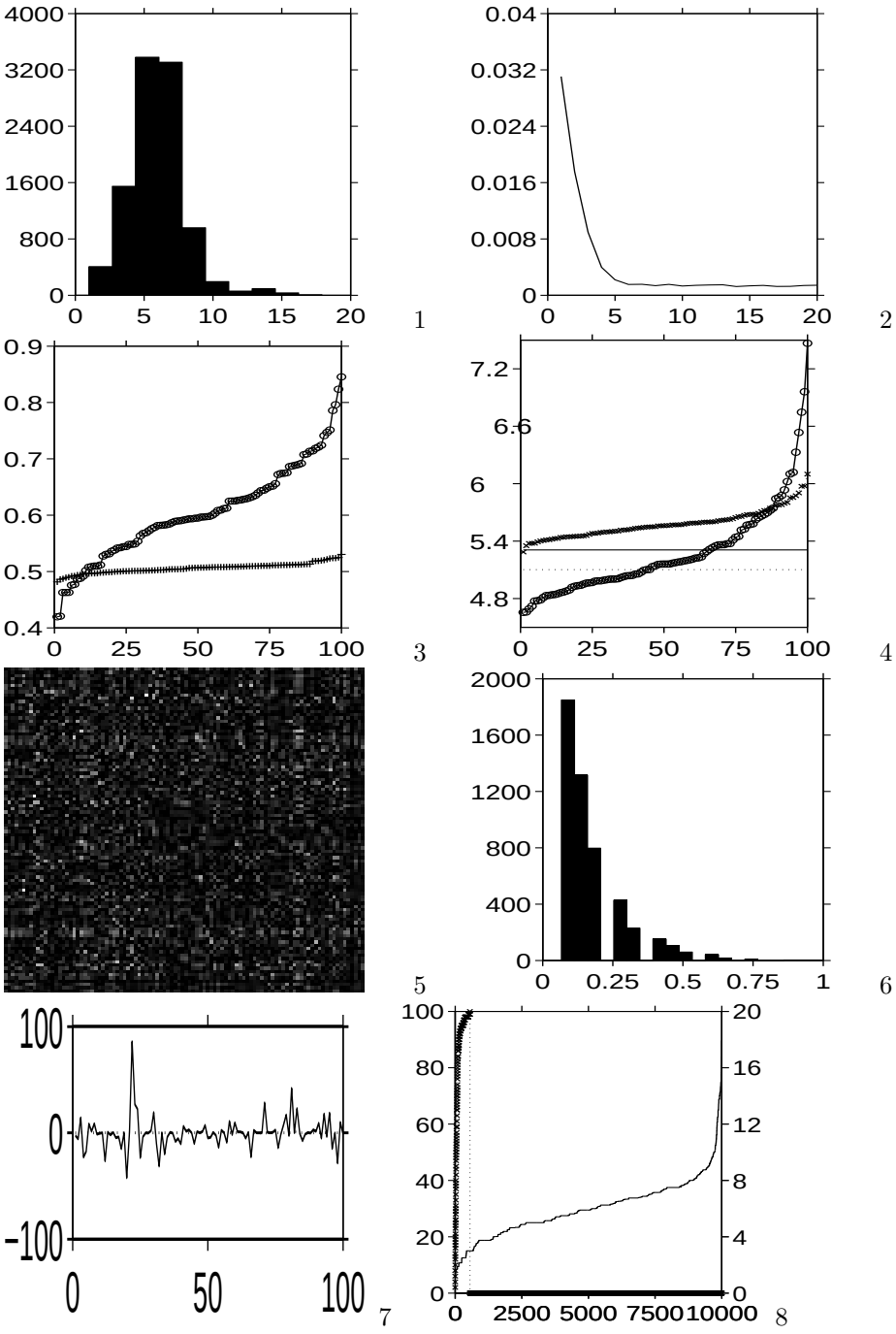


Figure 39: **Spike, feedback layer, symmetric kernel: random input, few neurons get external excitation and few neurons able to fire**
For figure details see section 0.4.$\theta = 0.3$, $r = 0.3$.

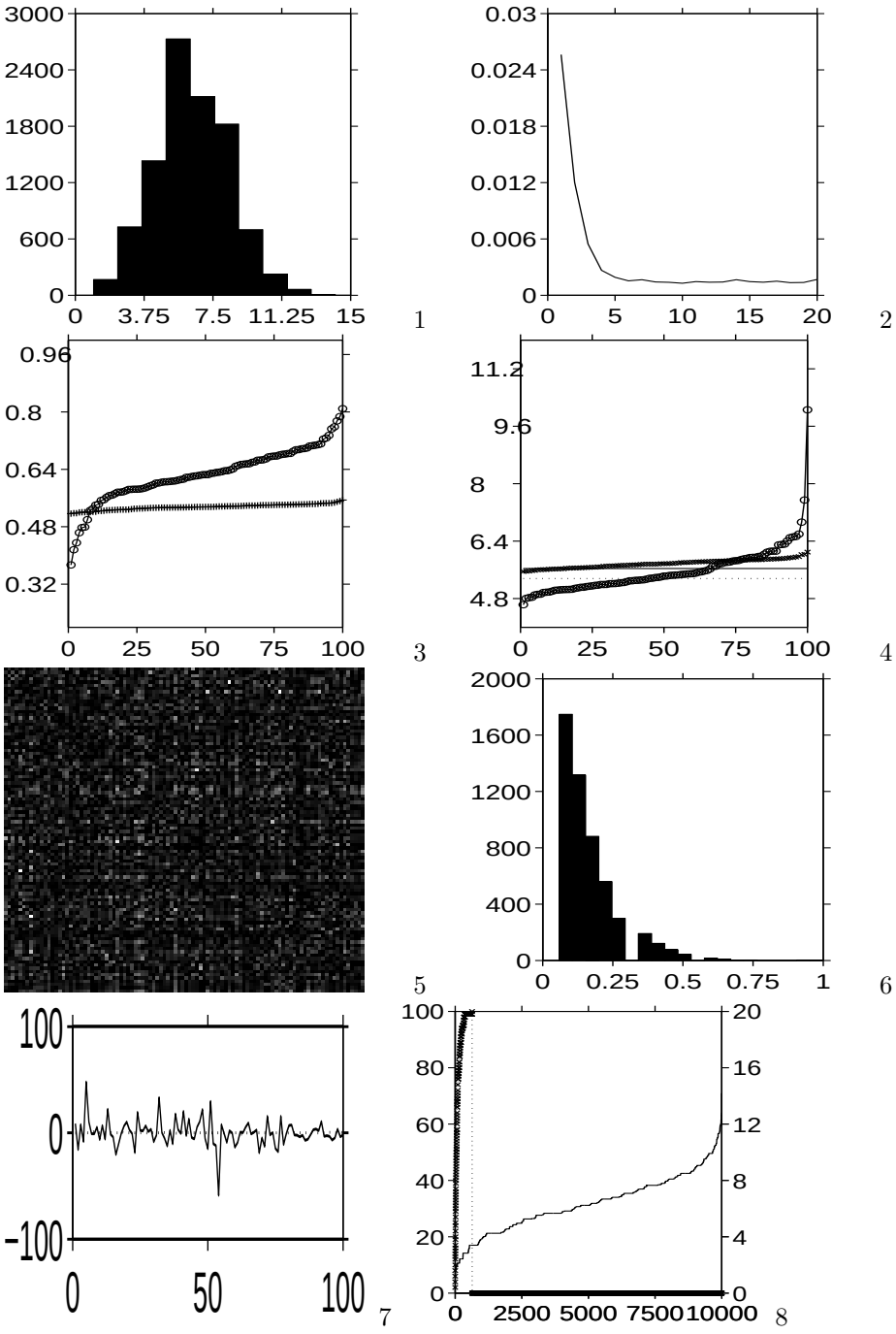


Figure 40: **Spike, feedback layer, symmetric kernel: random input, few neurons get external excitation but more neurons able to fire**
For figure details see section 0.4. $\theta = 0.7$, $r = 0.3$.

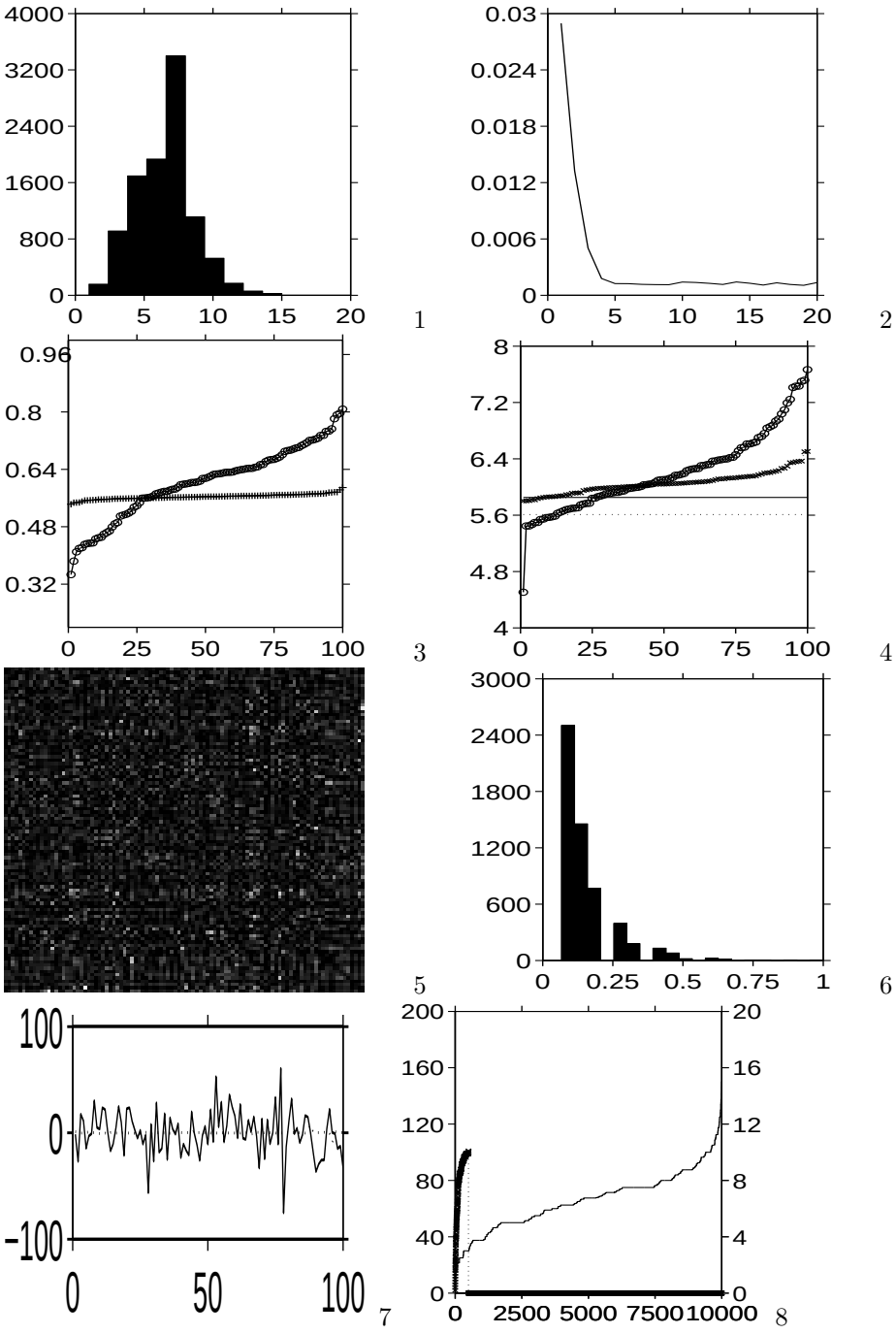


Figure 41: **Spike, feedback layer, symmetric kernel: random input, more neurons get external excitation but few neurons able to fire** For figure details see section 0.4.$\theta = 0.3$, $r = 0.7$.

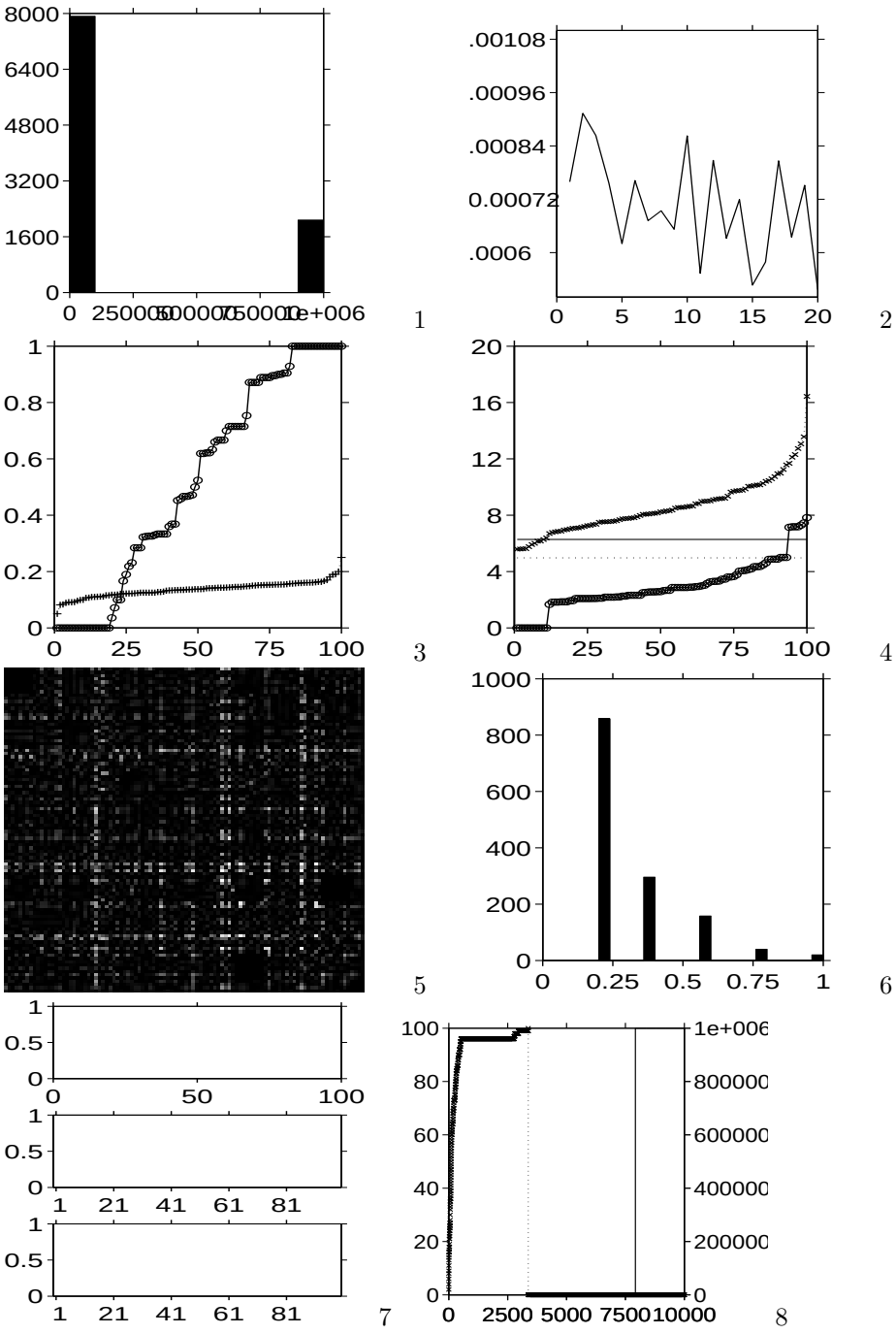


Figure 42: **Spike, feedback layer, symmetric kernel: random input, more neurons get external excitation and more neurons able to fire** For figure details see section 0.4.$\theta = 0.7$, $r = 0.7$.

Next we investigated the symmetric kernel with synchronous inputs. We have found that if the input is synchronous with L and r_A then the network is able to learn which neurons works together. The system has good associative property (Figures 45, 46) but the additive noise filtering is not so strong (Figures 47, 48). If the input is not synchronous then the system sometimes finds the synchronous work otherwise seems to be random (Figure 49) - it's like out-of-synch systems. With the 1^{st} firing model learns the synchronous input (Figure 44) but with the 2^{nd} one it learns something else (Figure 43). We used the default parameter values of Table 2.

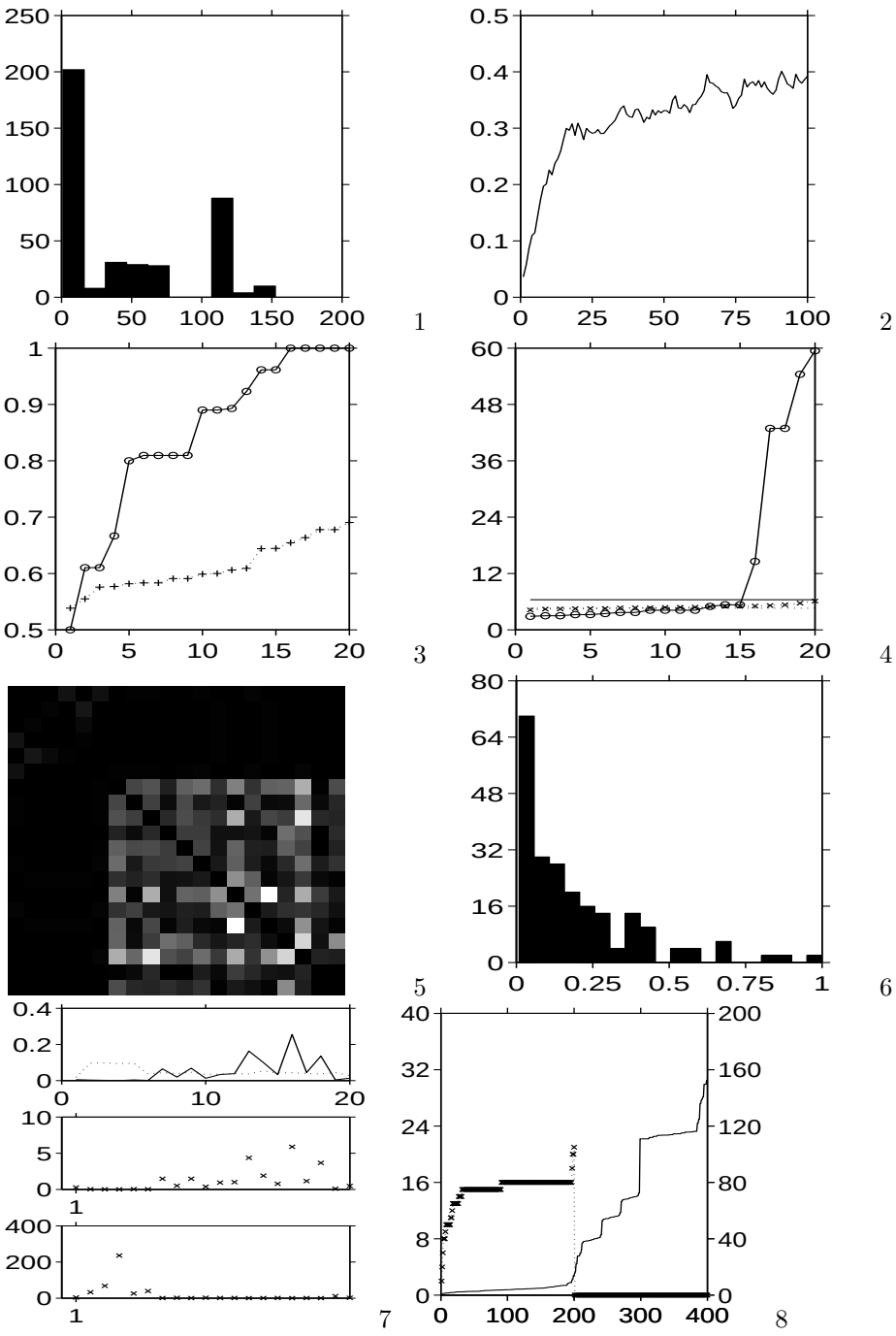


Figure 43: **Spike, feedback layer: synchronous input without noise, fixed average number of firing neurons**
For figure details see section 0.4.

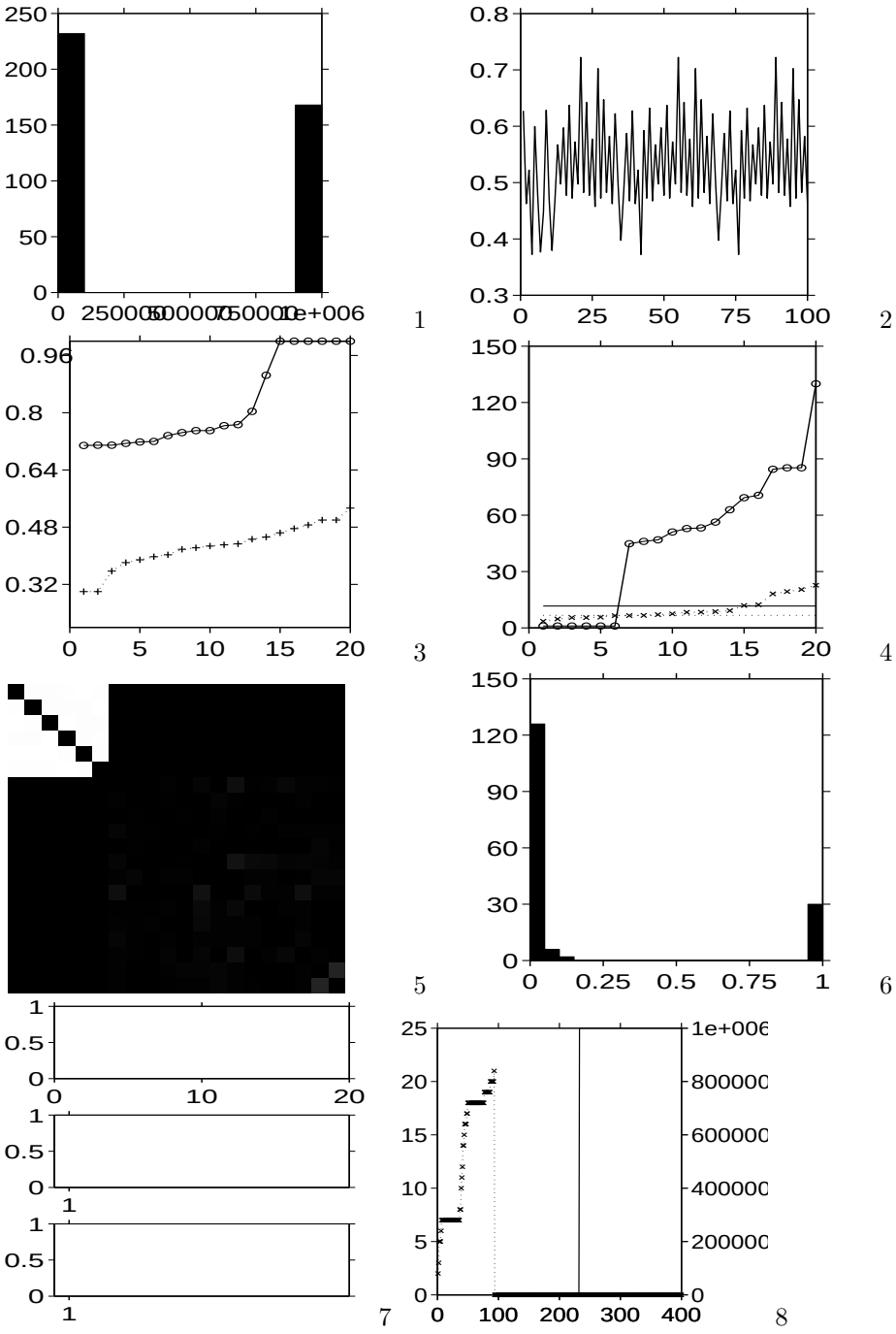


Figure 44: **Spike, feedback layer: synchronous input without noise, simple threshold firing model**
For figure details see section 0.4.

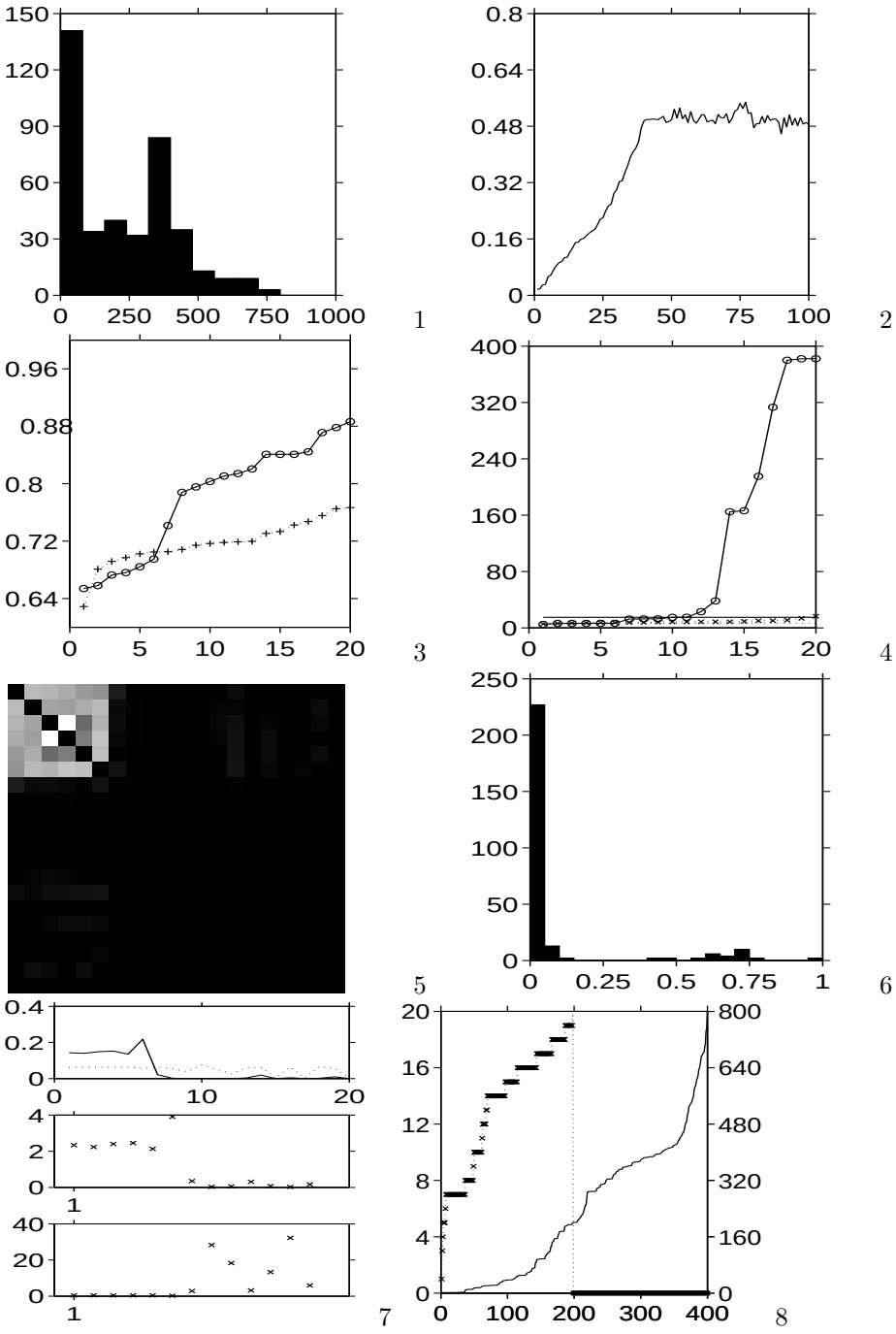


Figure 45: **Spike, feedback layer: synchronous input, 80 percent of input is missing** For figure details see section 0.4. Simple threshold firing model. The system learns the complete input.

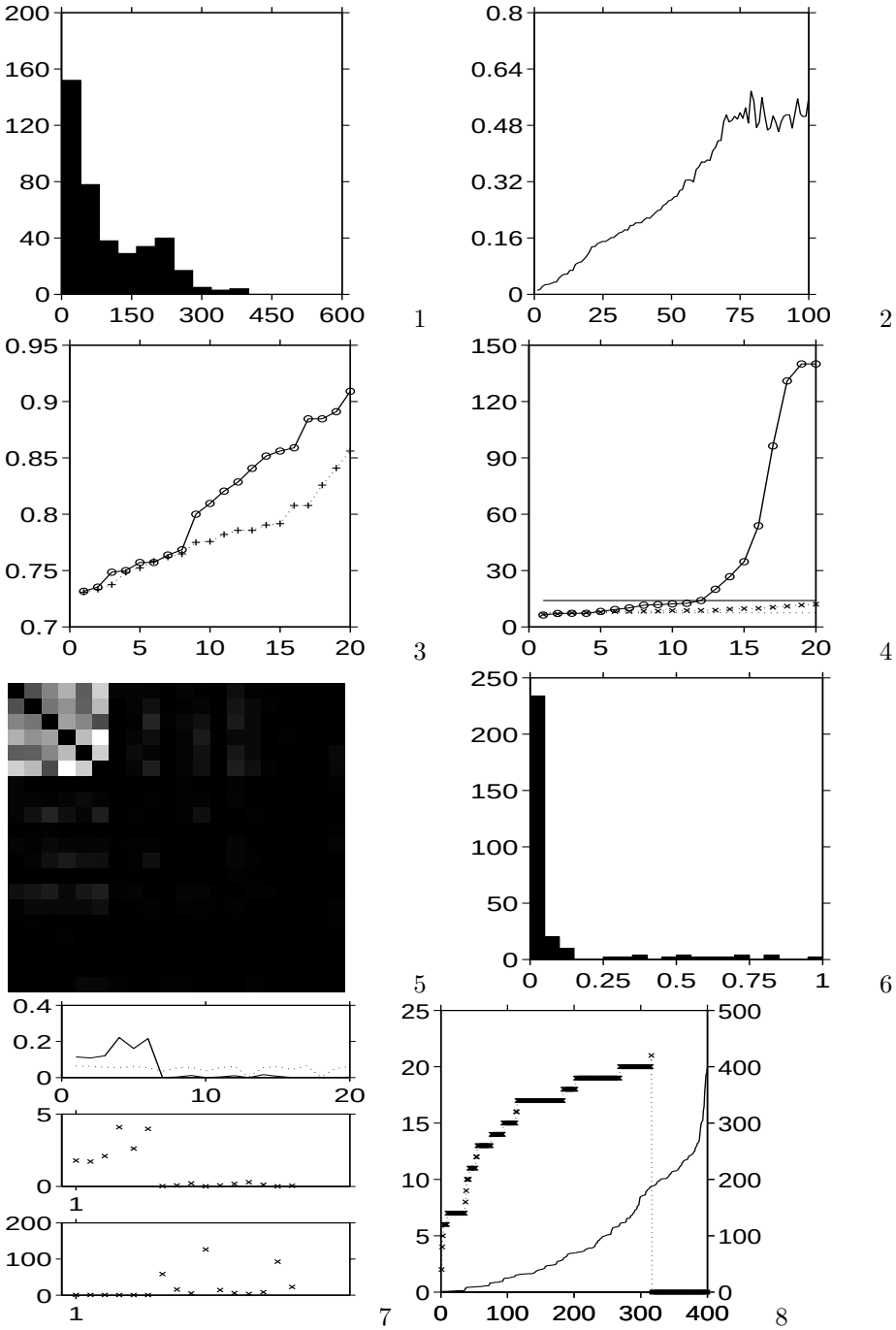


Figure 46: **Spike, feedback layer: synchronous input, 90 percent of input is missing**
For figure details see section 0.4. Simple threshold firing model. The system can't learn the complete input.

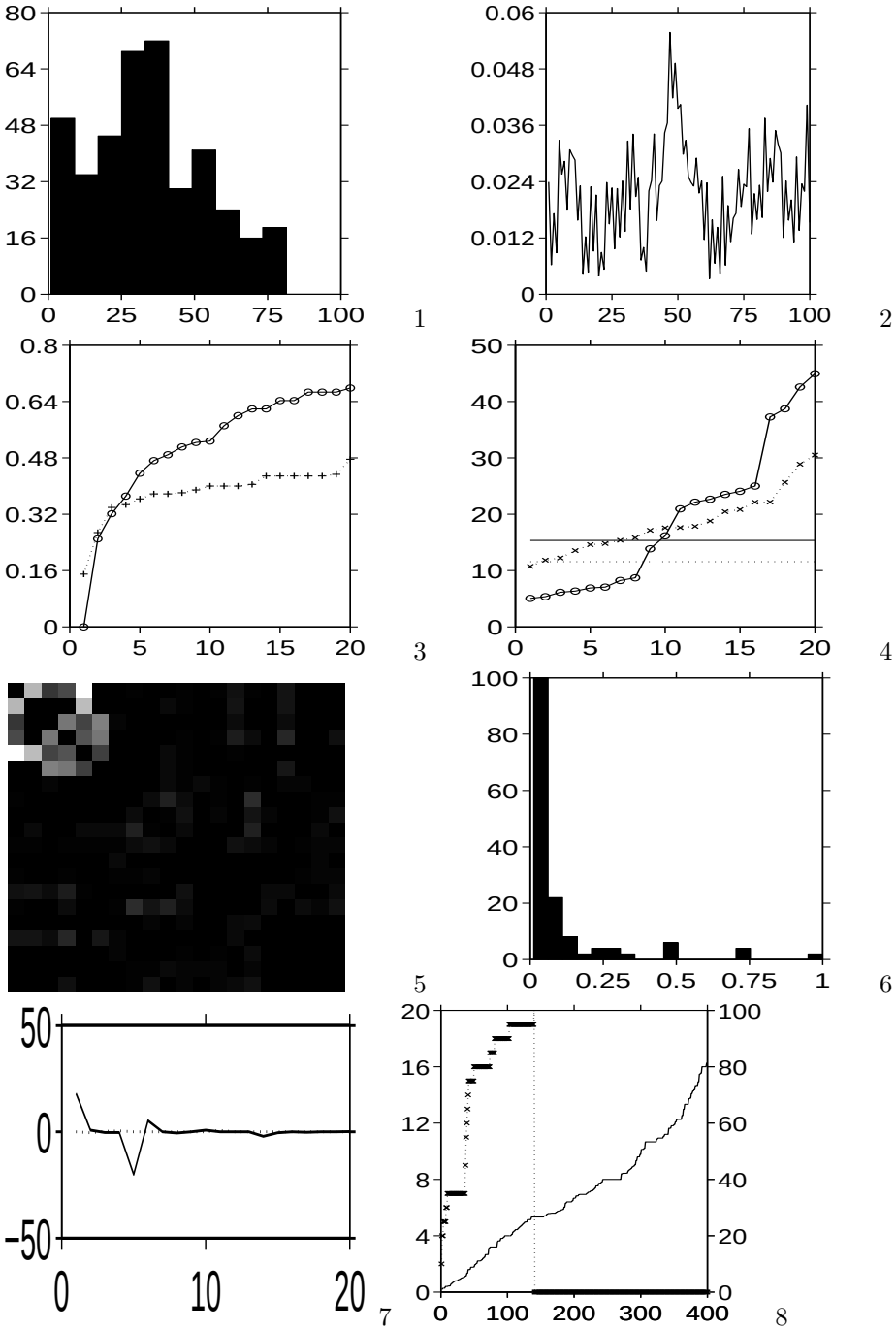


Figure 47: **Spike, feedback layer: synchronous input, 20 percent of neurons get external noise**
For figure details see section 0.4. Simple threshold firing model. The system learns the complete input without noise.

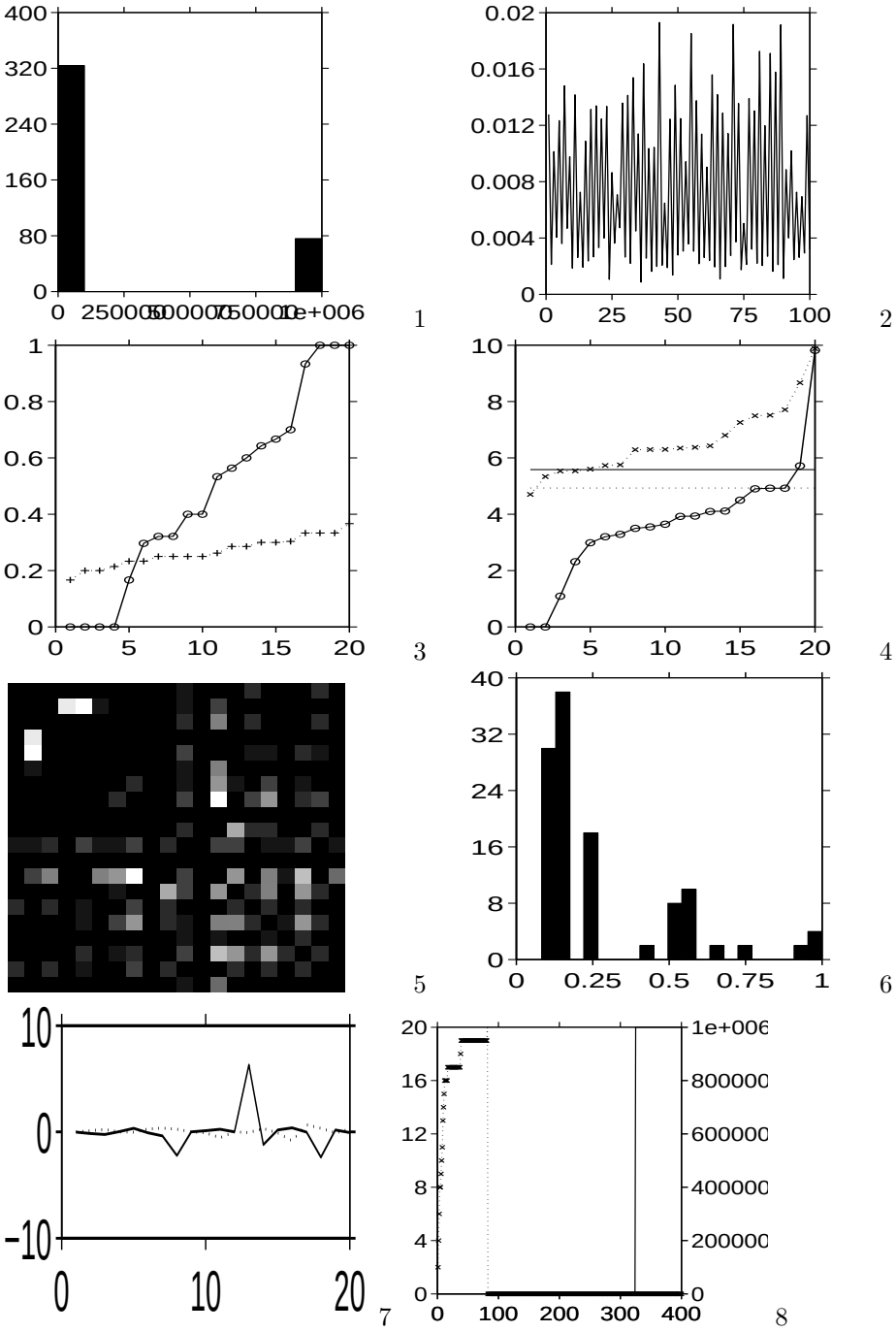


Figure 48: **Spike, feedback layer: synchronous input, 30 percent of neurons get external noise**
For figure details see section 0.4. Simple threshold firing model. The system can't learn the complete input without noise.

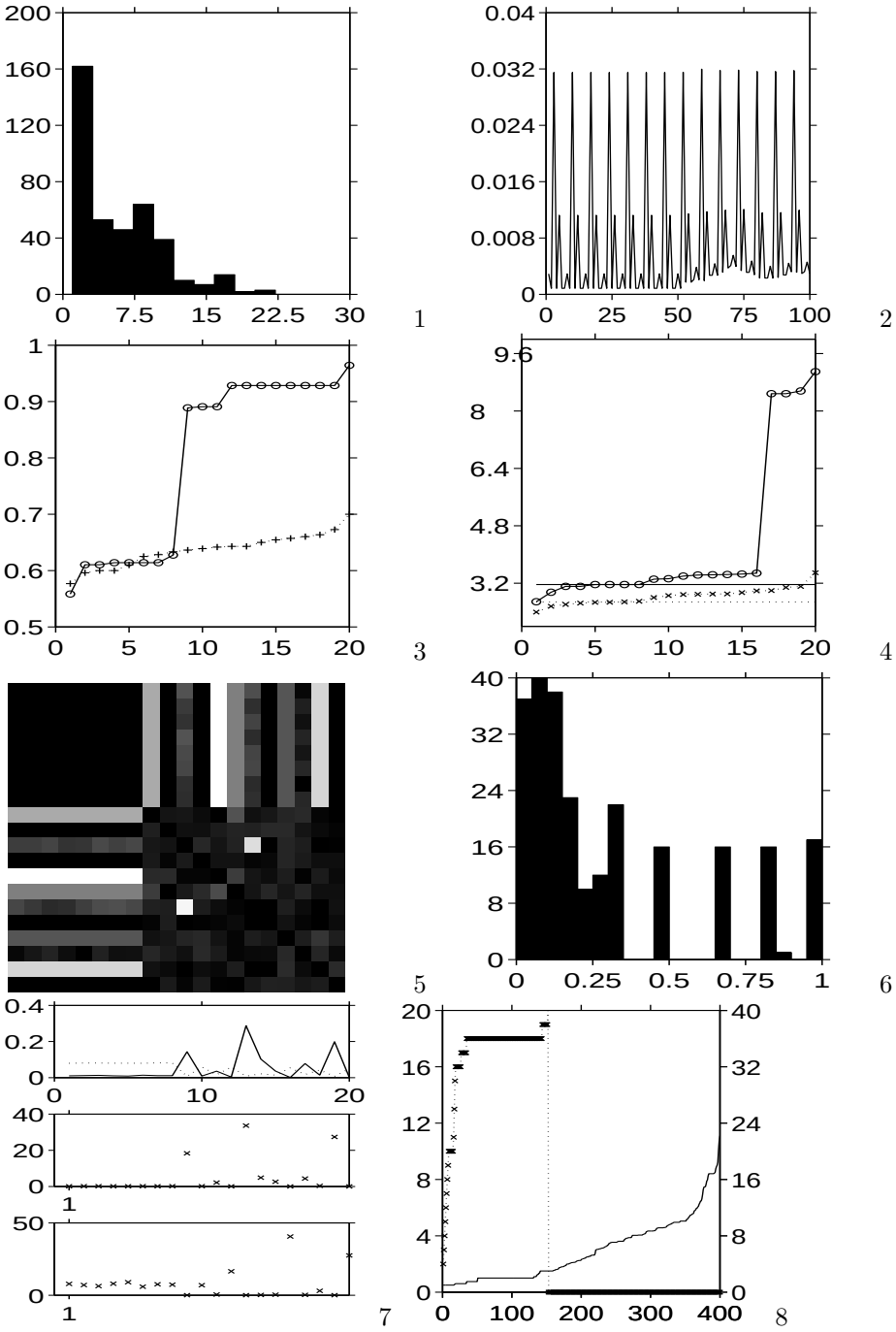


Figure 49: **Spike, feedback layer: out of synchronous work input without noise**
For figure details see section 0.4. Simple threshold firing model. The system can't learn the complete input.

Next we studied the symmetric kernel and simple threshold firing model with 2 synchronous inputs. We have found that if the inputs are synchronous with L and r_A then the network is able to learn which neurons works together - for the inputs independently (Figure 50). If the inputs share some neurons then the input for the common neurons goes out from synchronous work, so the net won't learn the work of that neuron (Figure 56). If one of the inputs is out of synchronous work then the net will learn just the synchronous one (Figures 58, 57). The noise filtering property is a little bit weaker than in the 1 synchronous input case (Figures 53 - 55). The input association strength remained the same (Figures 51, 52). We used the default parameter values of Table 2.

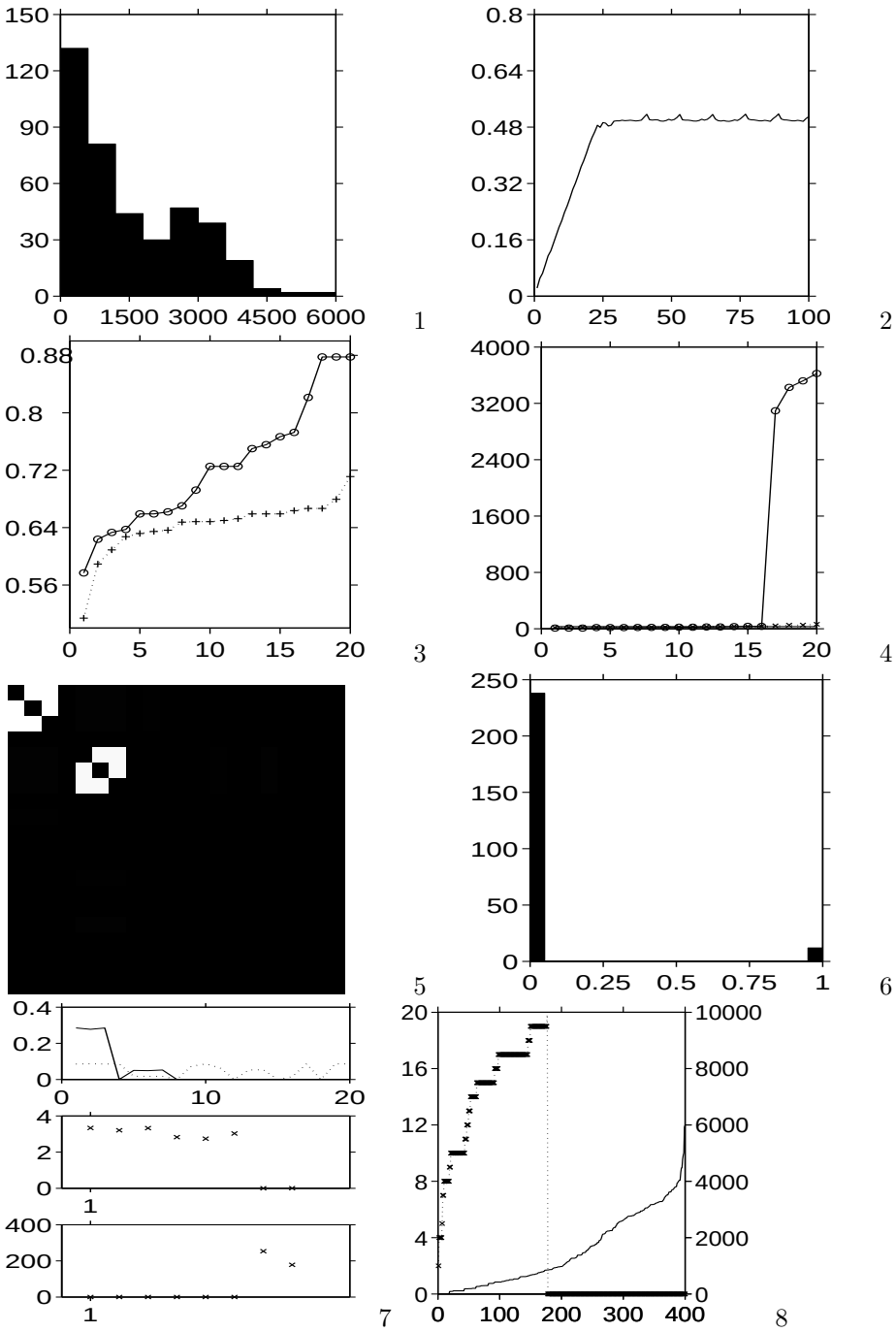


Figure 50: **Spike, feedback layer: 2 synchronous inputs without noise** For figure details see section 0.4. The system learns both complete input.

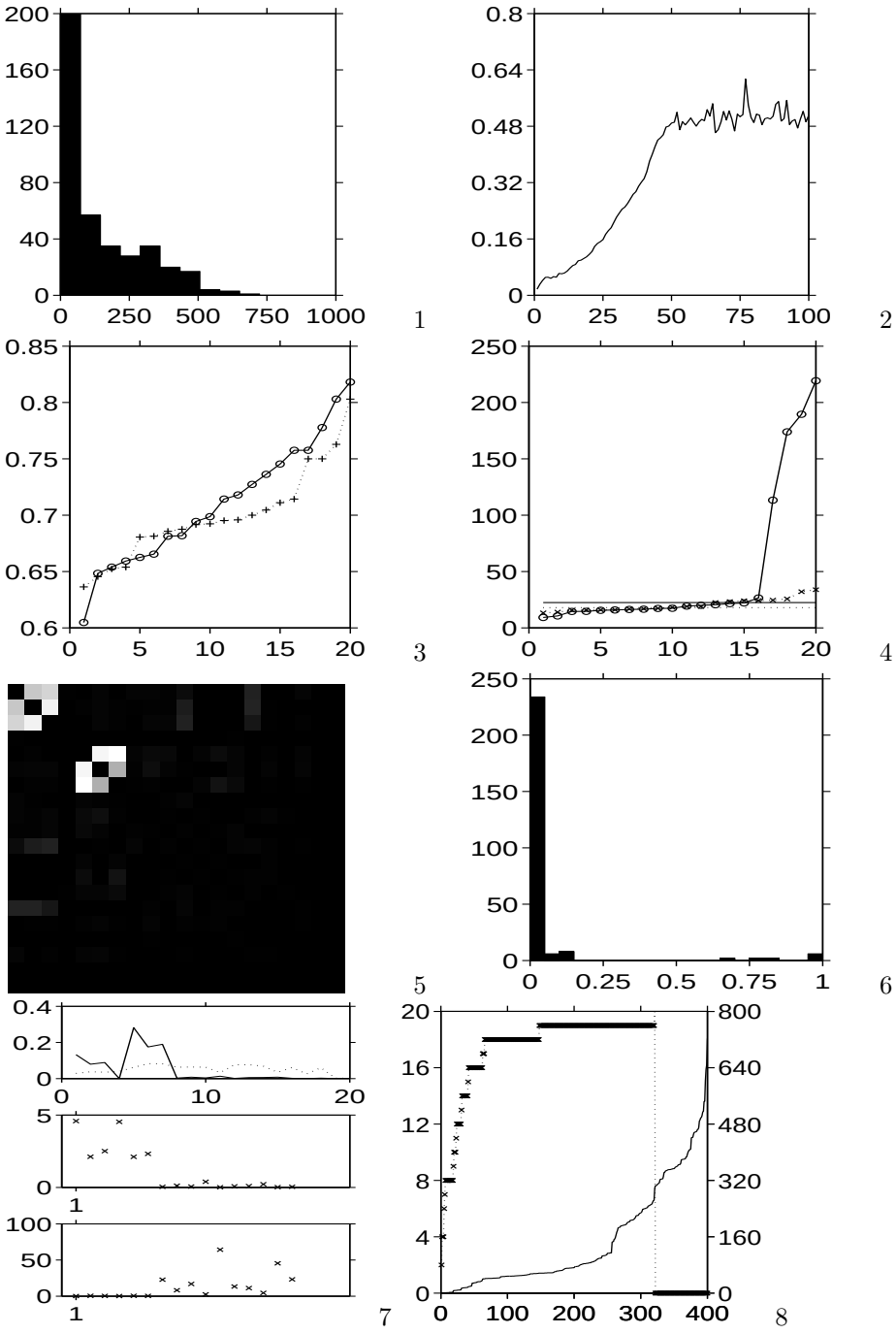


Figure 51: **Spike, feedback layer: 2 synchronous inputs, 80 percent of input is missing**
For figure details see section 0.4. The system learns both complete input.

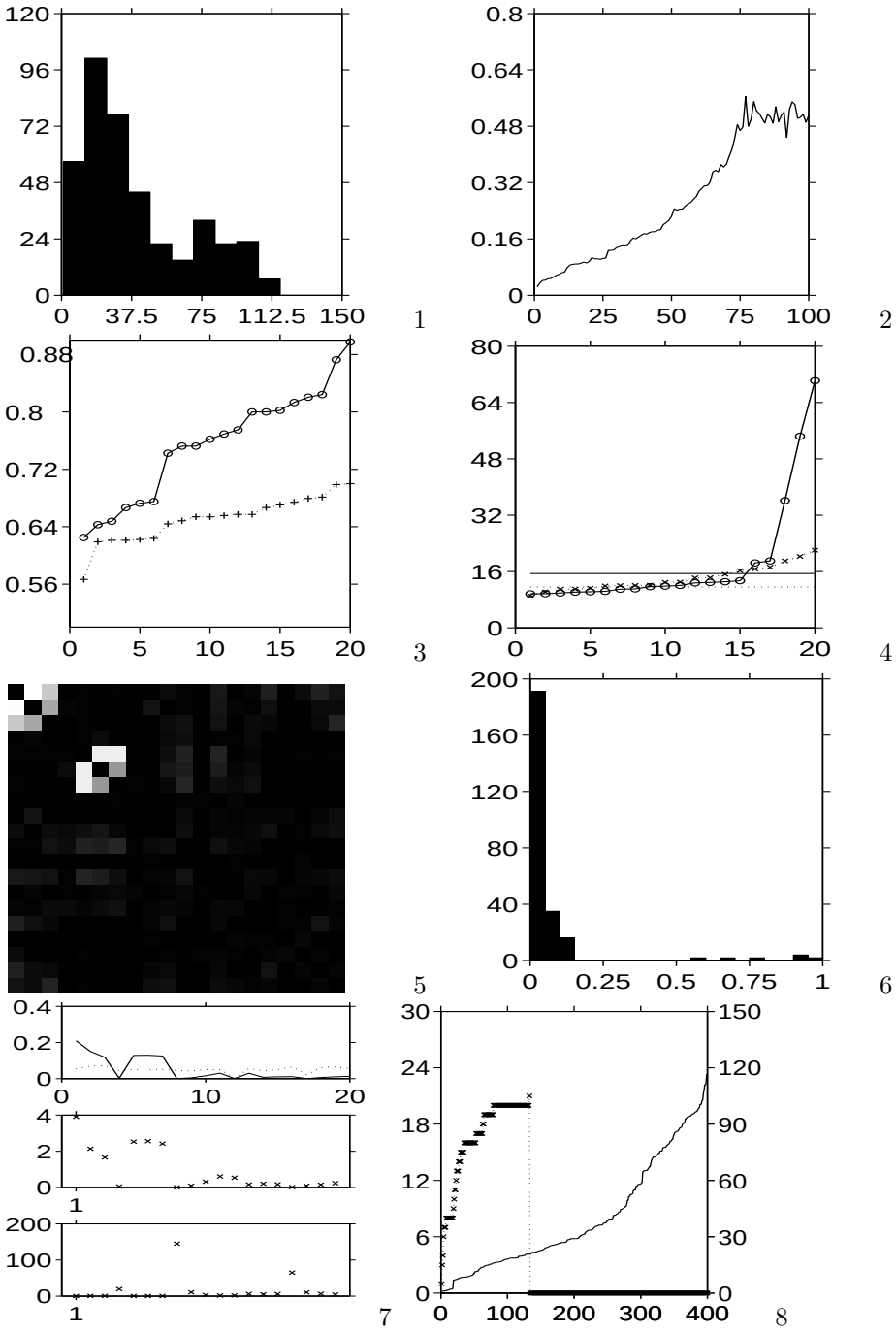


Figure 52: **Spike, feedback layer: 2 synchronous inputs, 90 percent of input is missing**
For figure details see section 0.4. The system can't learn neither complete input.

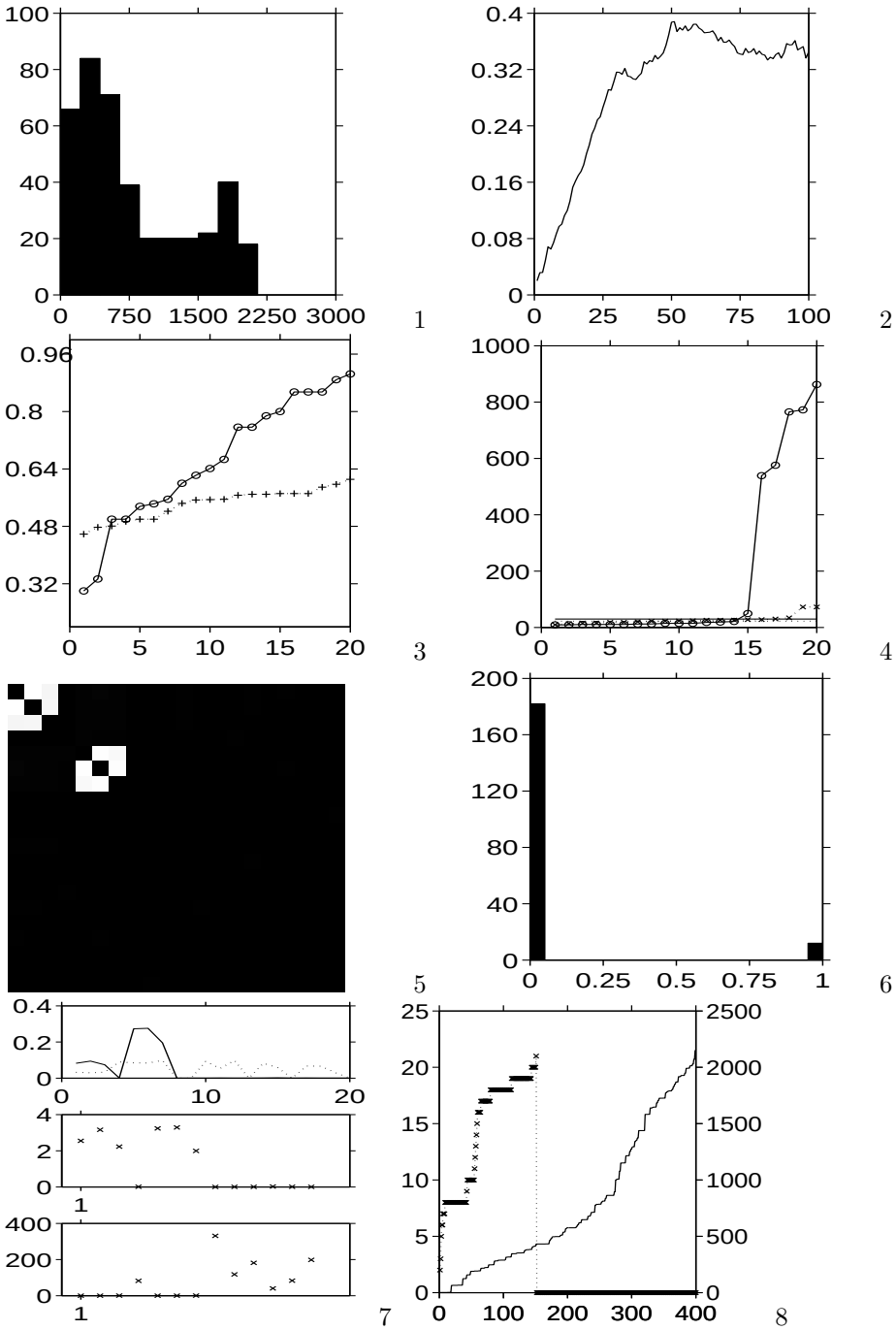


Figure 53: **Spike, feedback layer: 2 synchronous inputs, 10 percent of neurons get external noise**
For figure details see section 0.4. The system learns both complete input without noise.

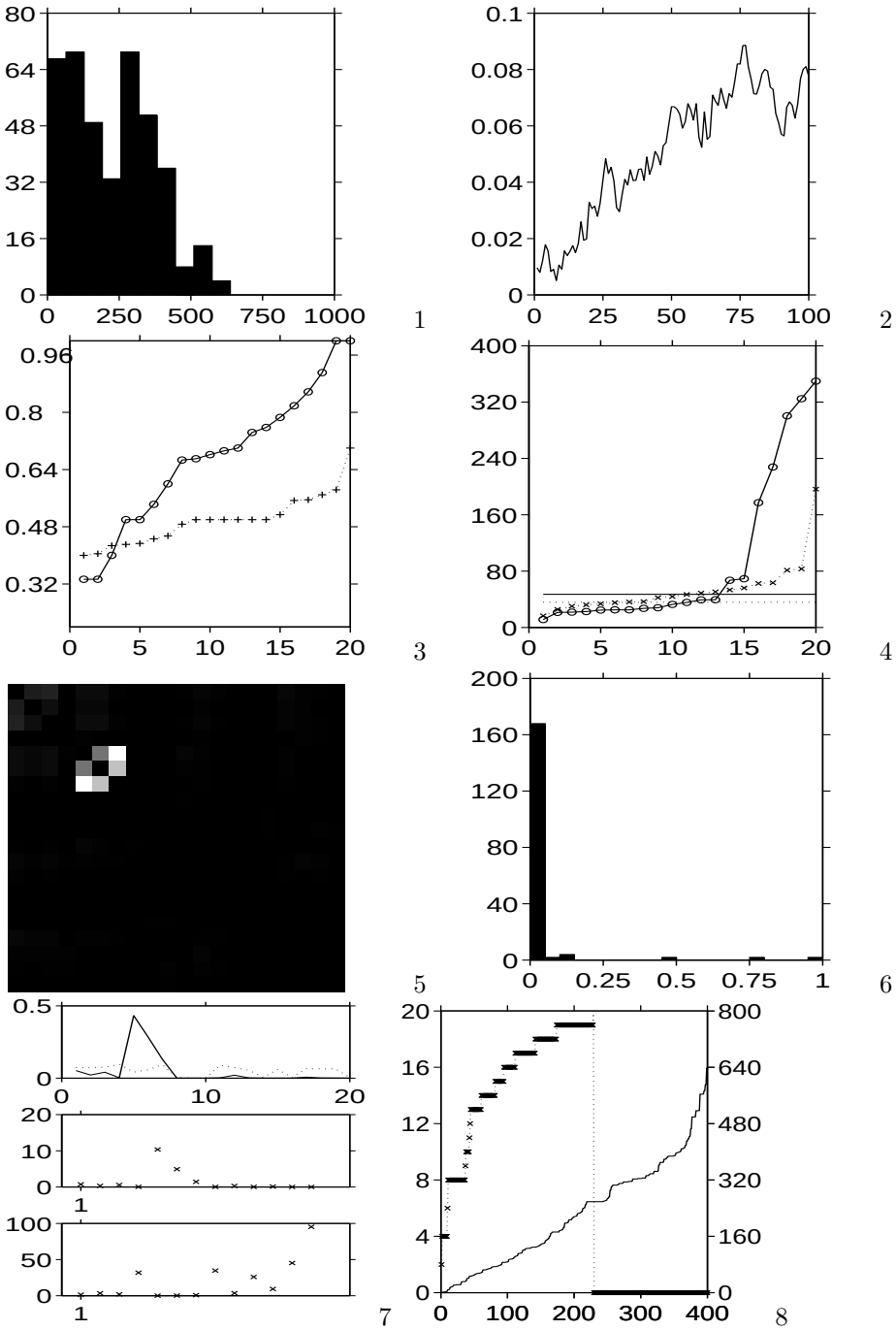


Figure 54: **Spike, feedback layer: 2 synchronous inputs, 20 percent of neurons get external noise**
For figure details see section 0.4. The system learns just one complete input without noise.

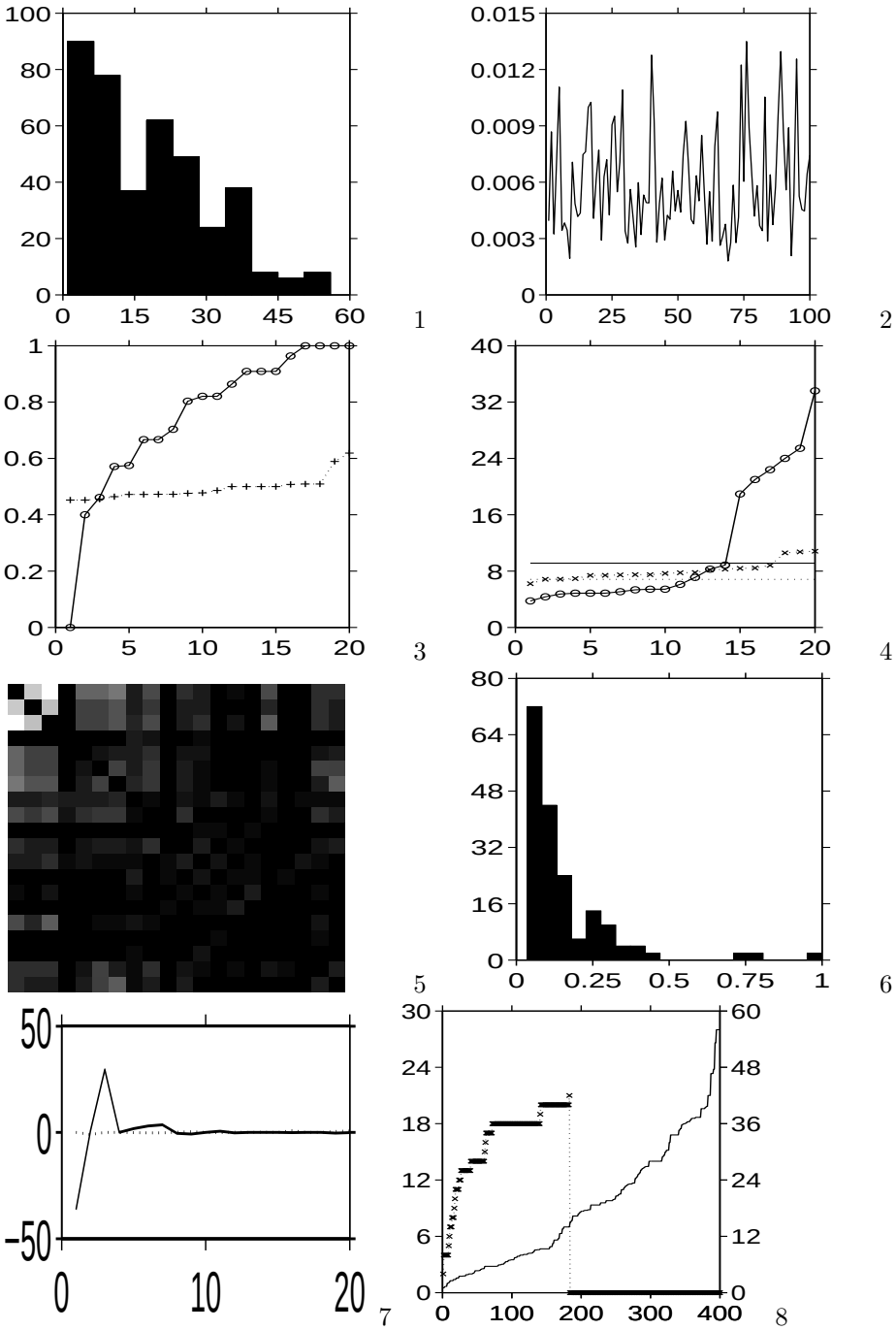


Figure 55: **Spike, feedback layer: 2 synchronous inputs, 30 percent of neurons get external noise**
For figure details see section 0.4. The system can't learn neither complete input without noise.

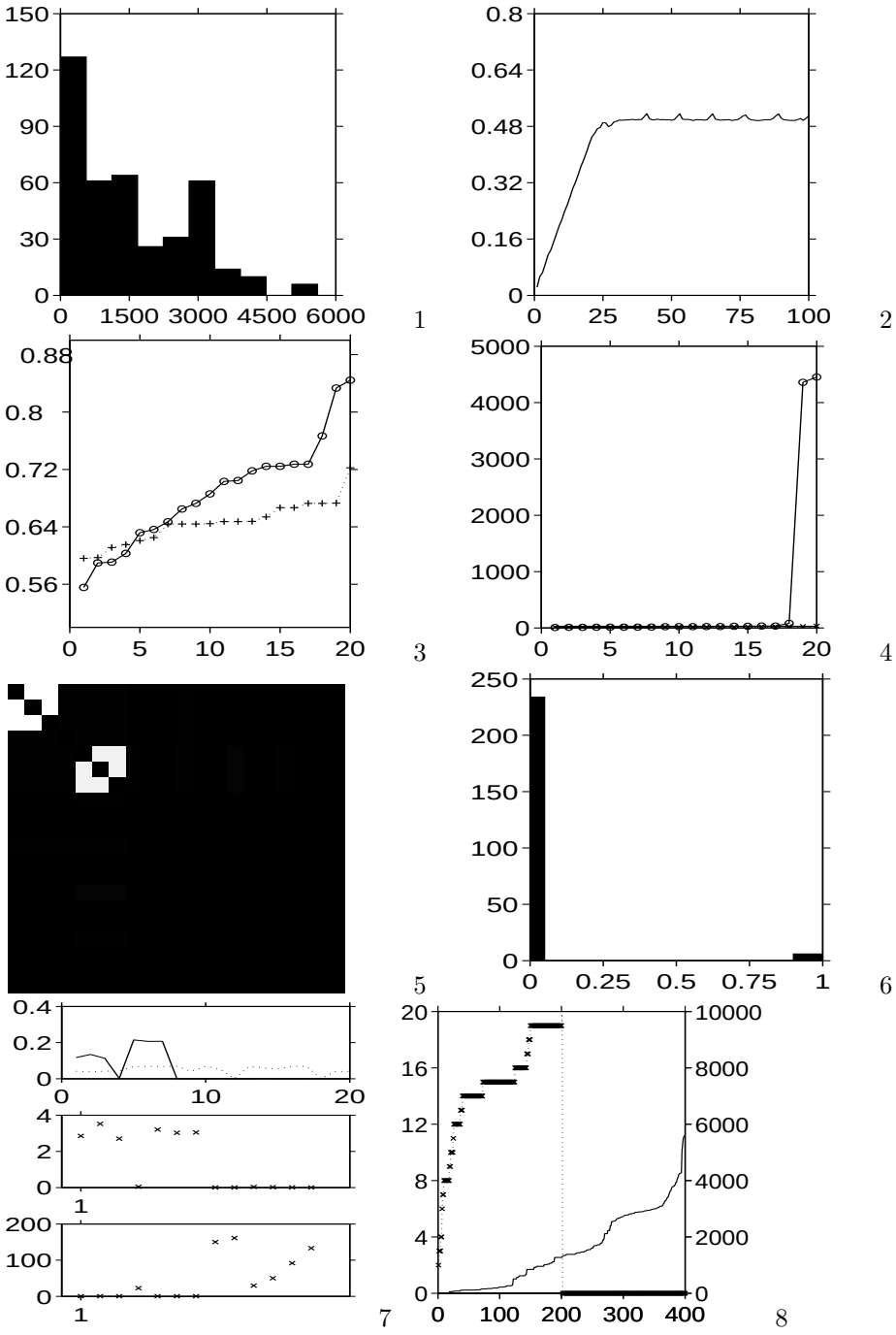


Figure 56: **Spike, feedback layer: 2 synchronous inputs without noise, with 1 common neuron**
For figure details see section 0.4. The system learns both complete input without the common neuron. Here one input is 4 neuron wide, but system learns only 3 neurons per input

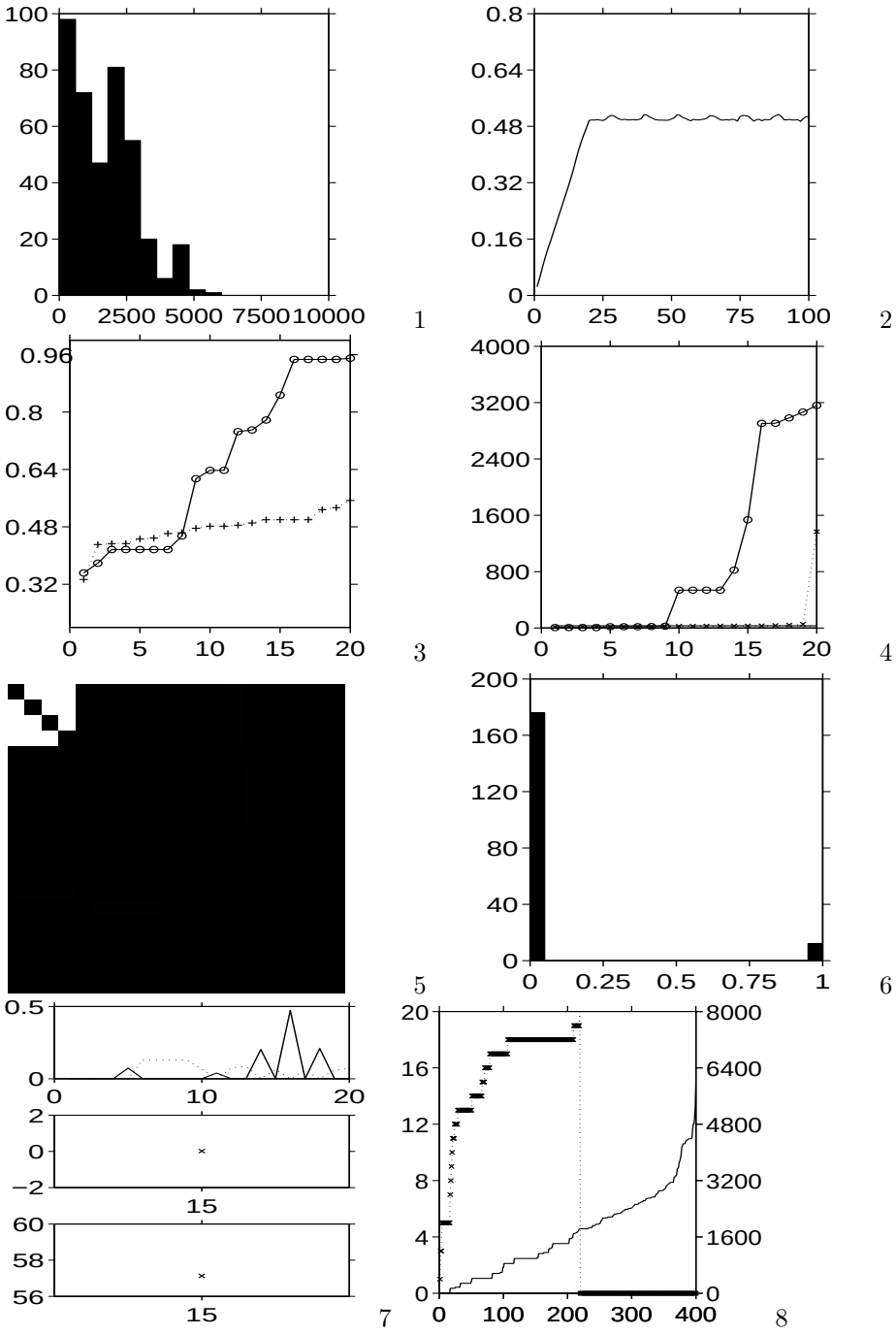


Figure 57: **Spike, feedback layer: 1 synchronous input and 1 out of synchronous work without noise, with 1 common neuron**
For figure details see section 0.4. The system learns the synchronous input without the common neuron. Here one input is 4 neuron wide, but system learns only 3.

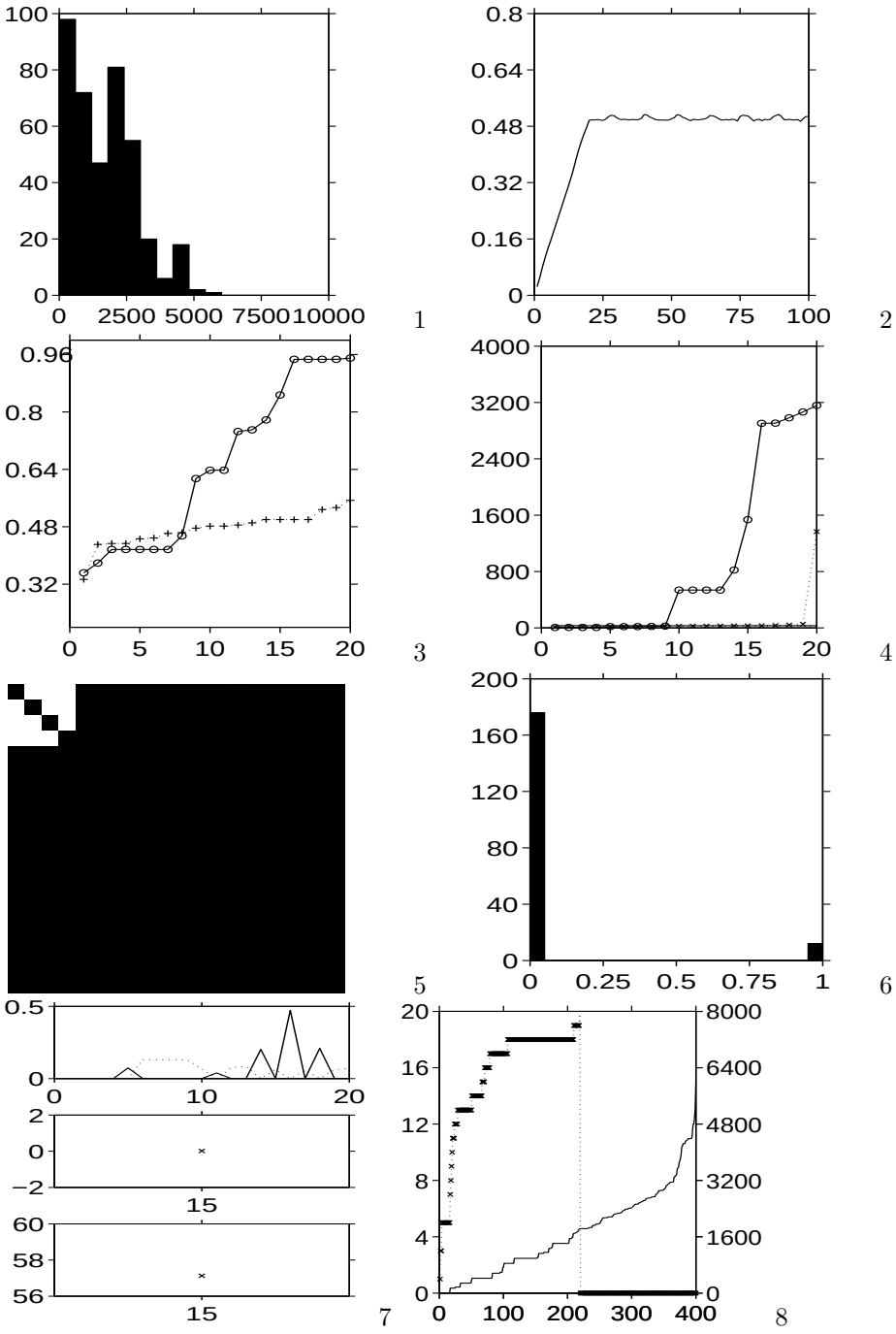


Figure 58: **Spike, feedback layer: 1 synchronous input and separate 1 out of synchronous work without noise**
For figure details see section 0.4. The system learns the synchronous input.

Next we investigated the symmetric kernel with simple threshold firing model for a special synchronous input: the moving bar input. The i^{th} neuron get external excitation in the $[j, j + barlength]$ time interval if $j \, mod \, N = i$. We have found the following different behaviors:

- On a relatively large range of time window sizes the system able to learn the time sequence of the bars. If $L \approx barlength$ the system is stable (Figures 60, 61). If $L \approx 2 * (barlength)$ the system is not stable but learns the time sequence (Figures 62, 63).
- The too short time window size, i.e. $L \leq (barlength)/2$ prevents the system to learn the sequence (Figure 59).
- The too large time window, i.e. $L \approx 2 * (N - bar)$ length), causes some strange effects (Figures 64, 61).
- If $L < N - bar_length$ but $L \gg barlength$ then the system learns the time sequence and the together excited neurons too (Figure 66).

We used the default parameter values of Table 2 except for $N = 20$ and $r = 0.3$ (bar length is $N * r = 6$).

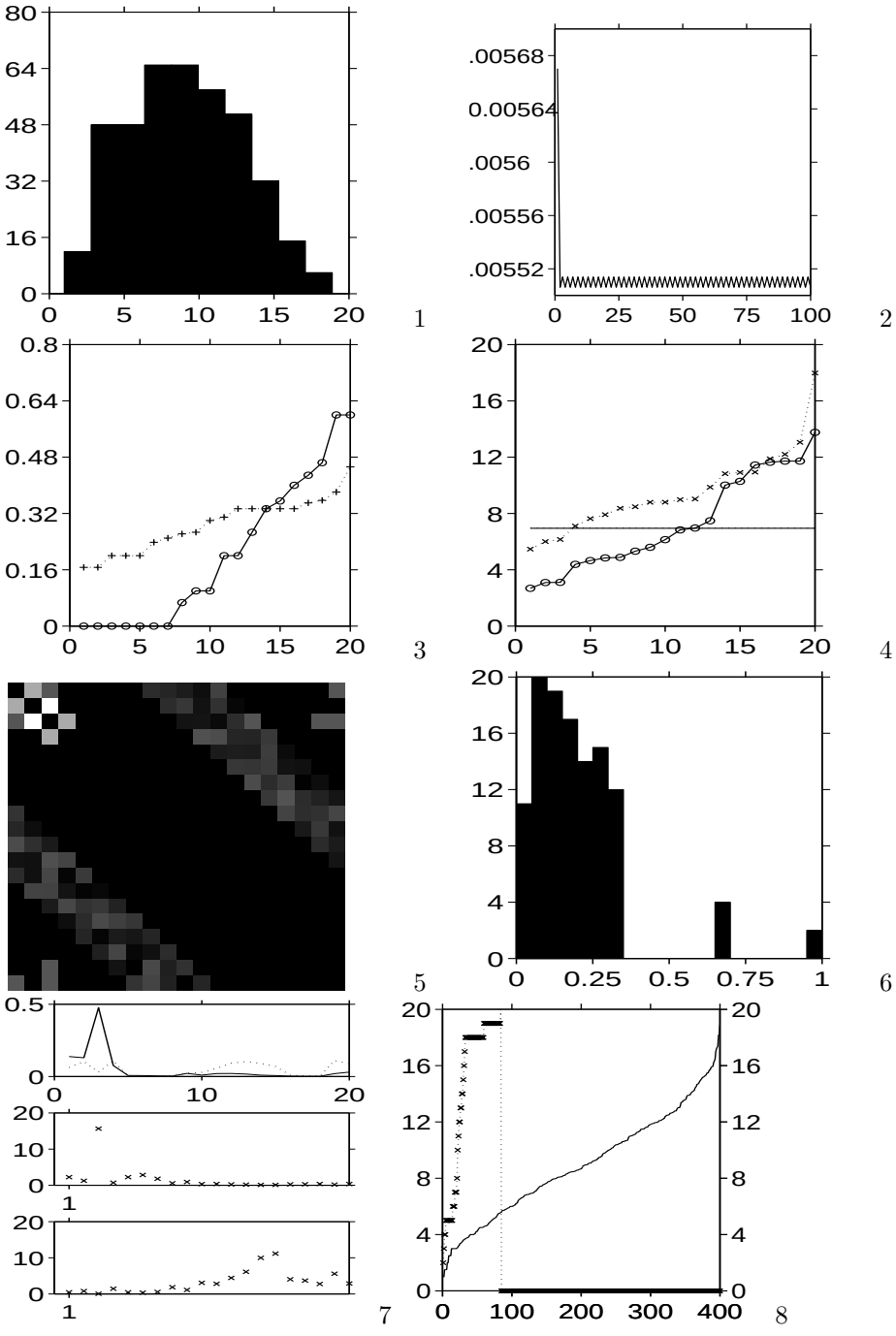


Figure 59: **Spike, feedback layer: moving bar input with $L = 2$**
For figure details see section 0.4. The system can't learn the time sequence because of too short time window.

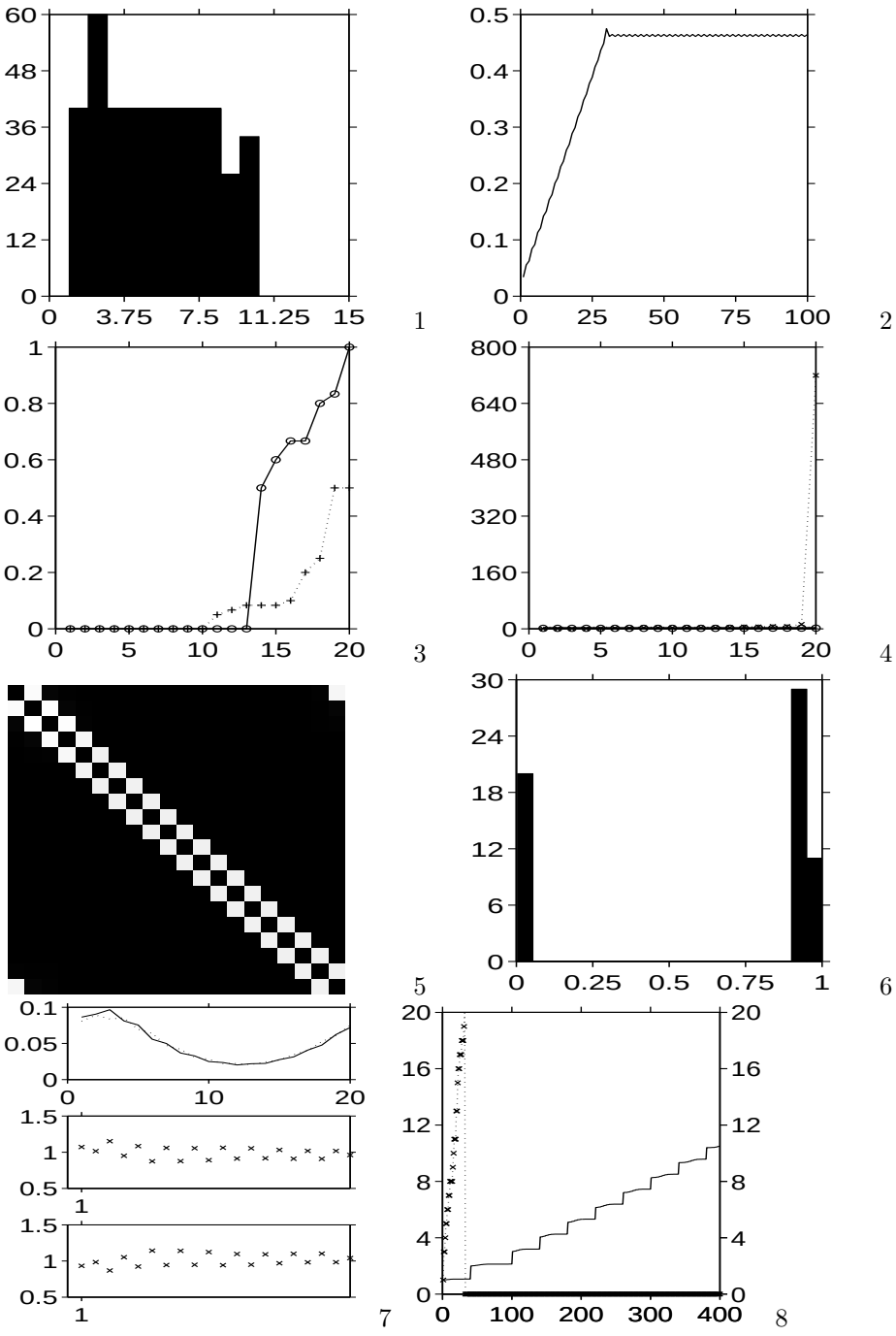


Figure 60: **Spike, feedback layer: moving bar input with $L = 4$**
For figure details see section 0.4. The system learns the time sequence.

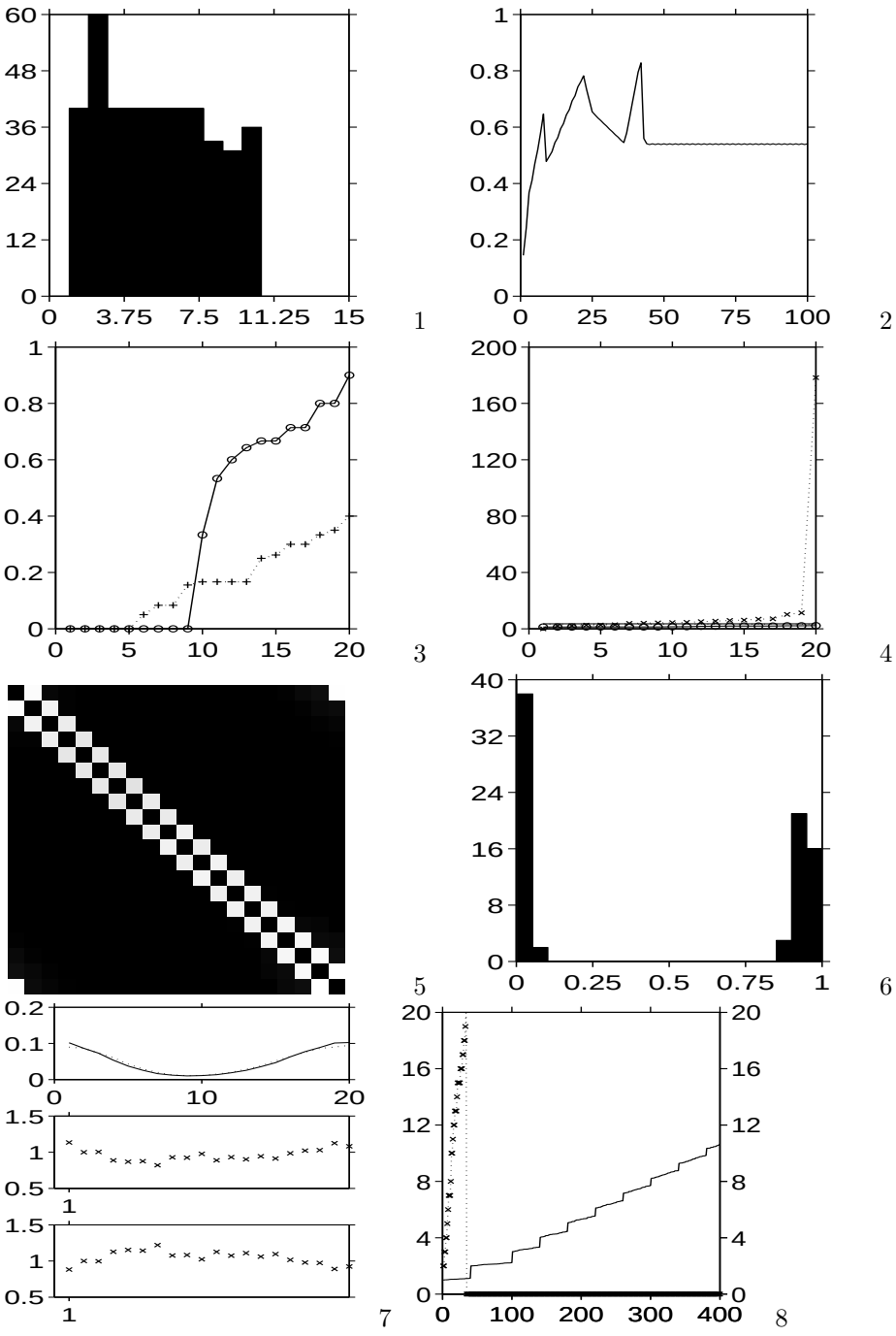


Figure 61: **Spike, feedback layer: moving bar input with $L = 6$**
For figure details see section 0.4. The system learns the time sequence.

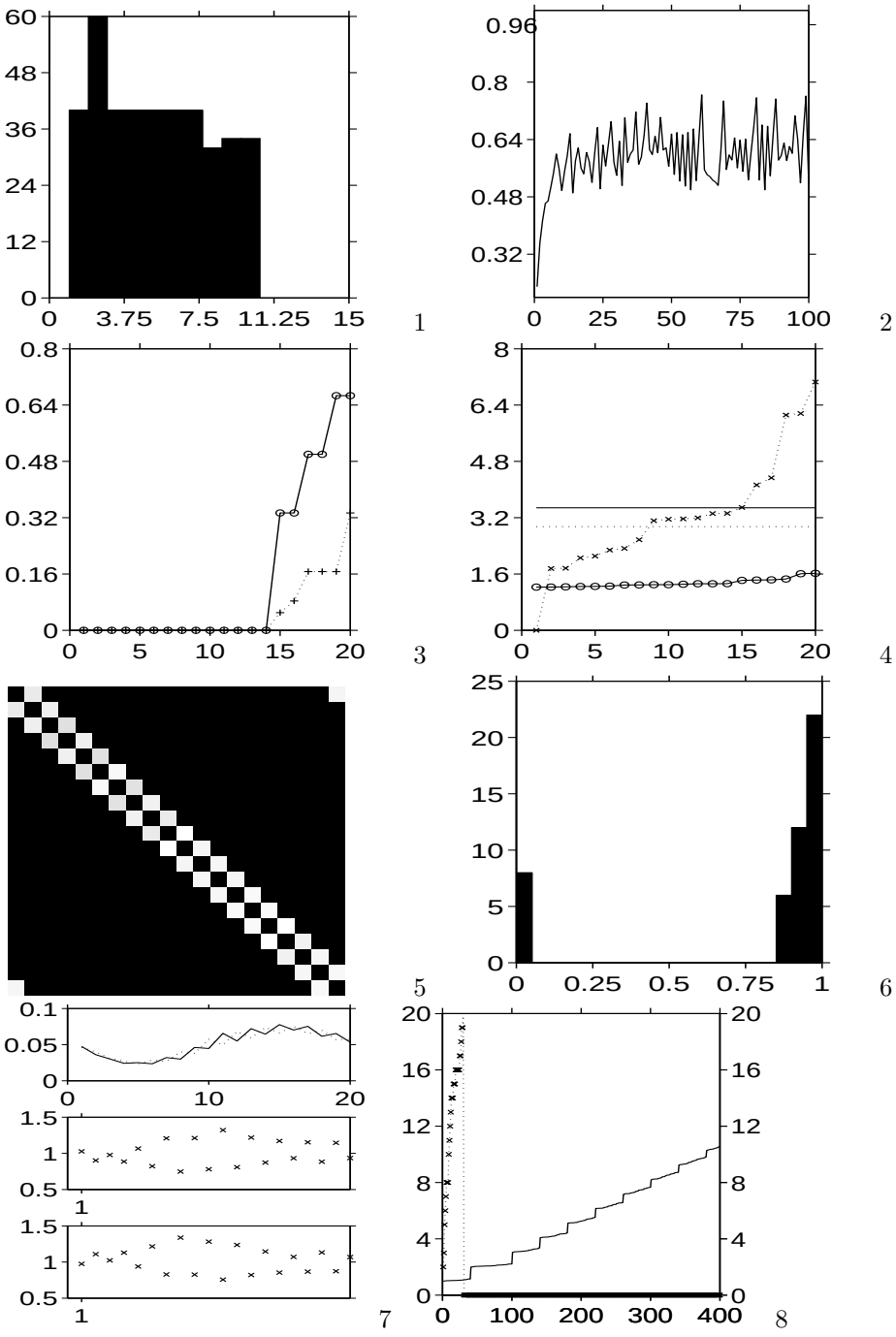


Figure 62: **Spike, feedback layer: moving bar input with** $L = 10$
For figure details see section 0.4. The system learns the time sequence.

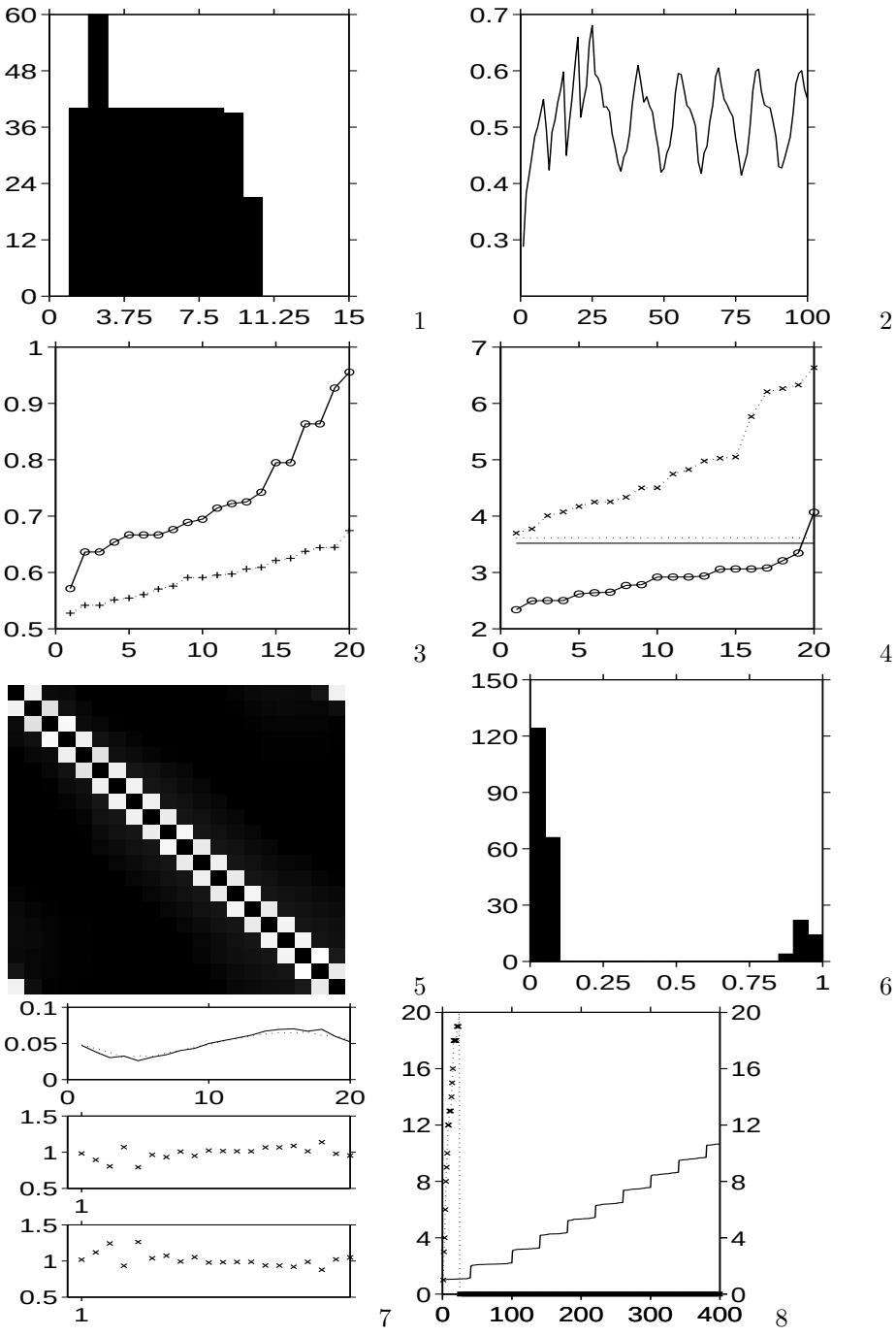


Figure 63: **Spike, feedback layer: moving bar input with** $L = 14$
For figure details see section 0.4. The system learns the time sequence.

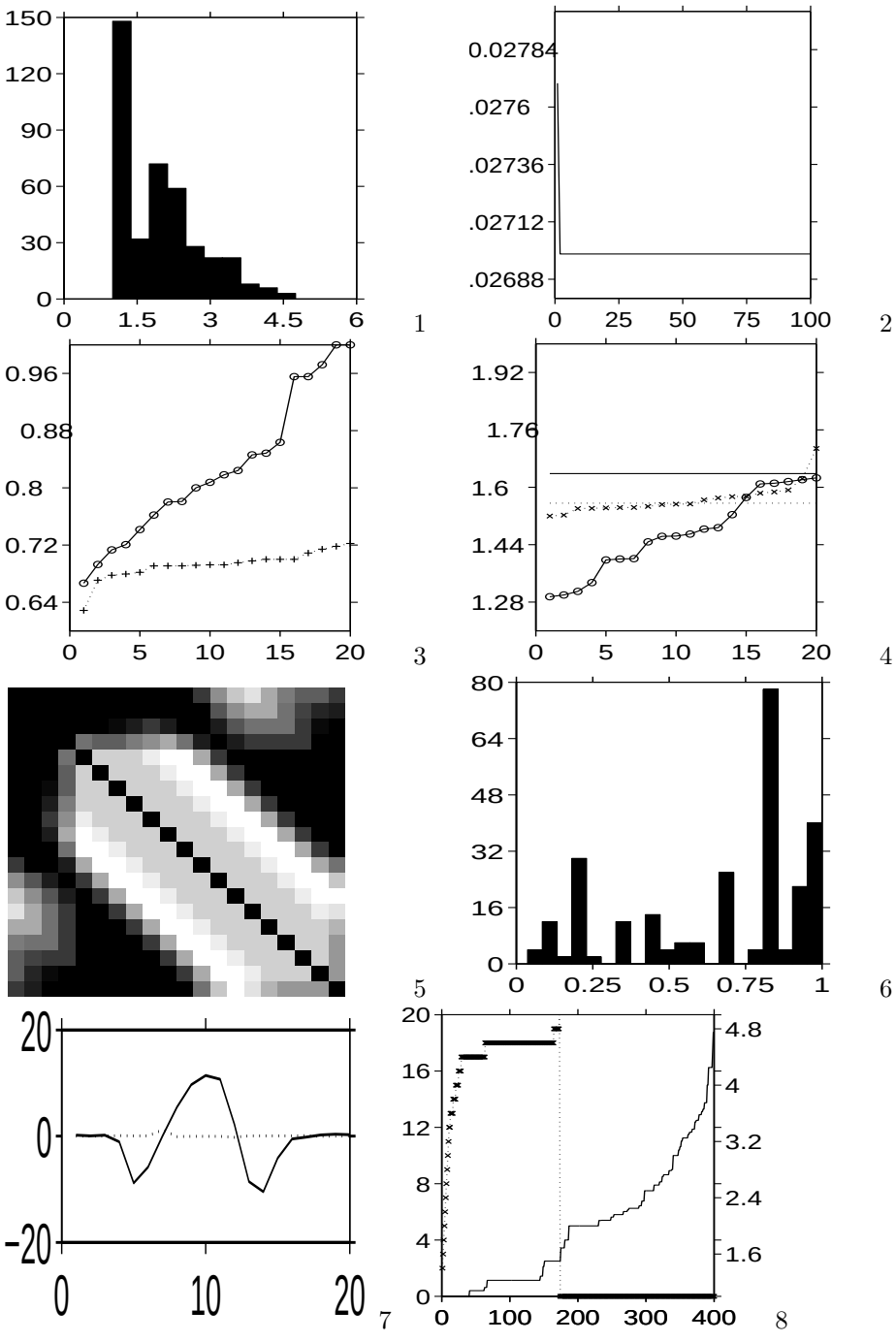


Figure 64: **Spike, feedback layer: moving bar input with $L = 18$**
For figure details see section 0.4. The system learns something, but not the sequence, because the time window is too large.

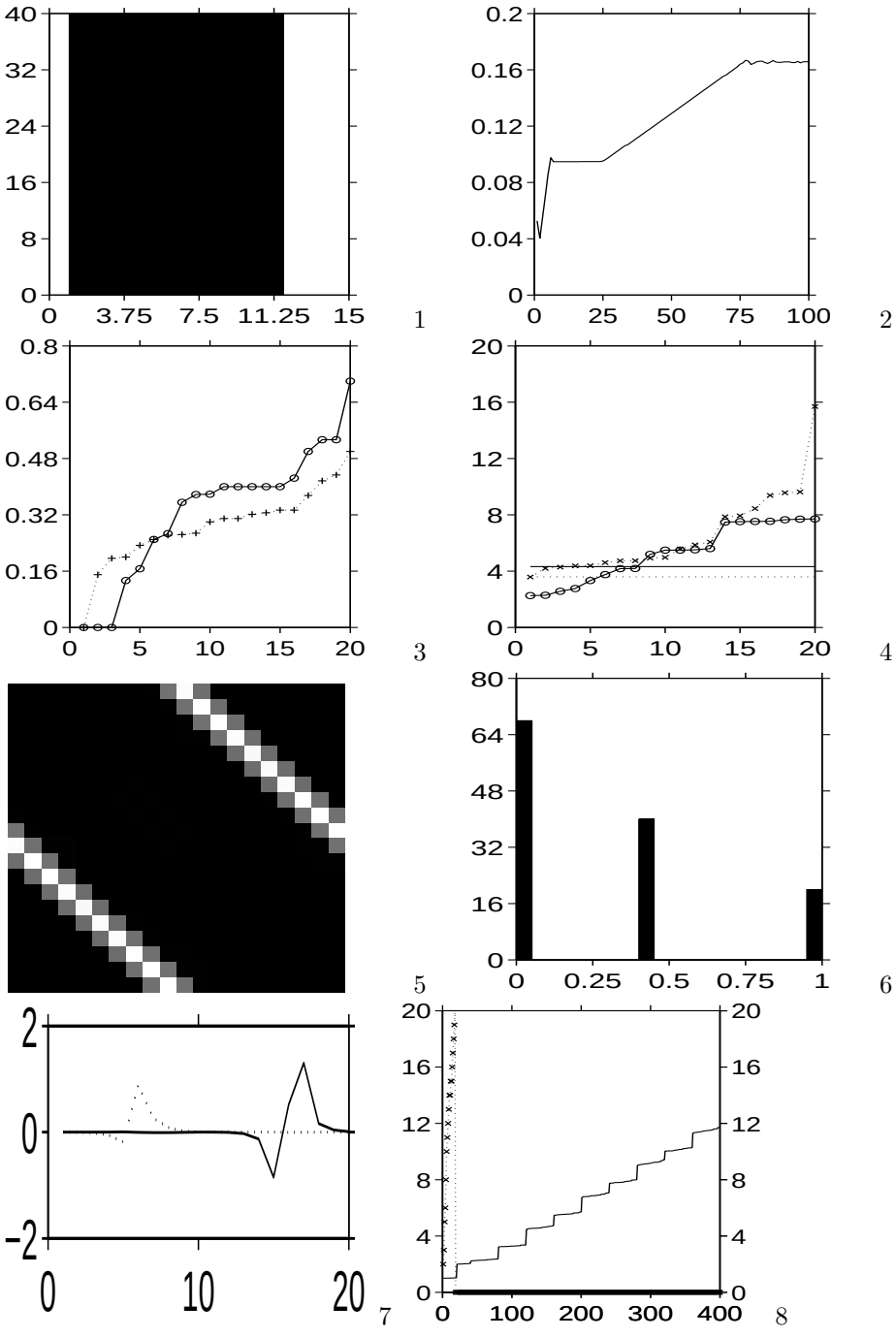


Figure 65: **Spike, feedback layer: moving bar input with $L = 26$**
For figure details see section 0.4. The system learns something, but not the sequence, because the time window is too large.

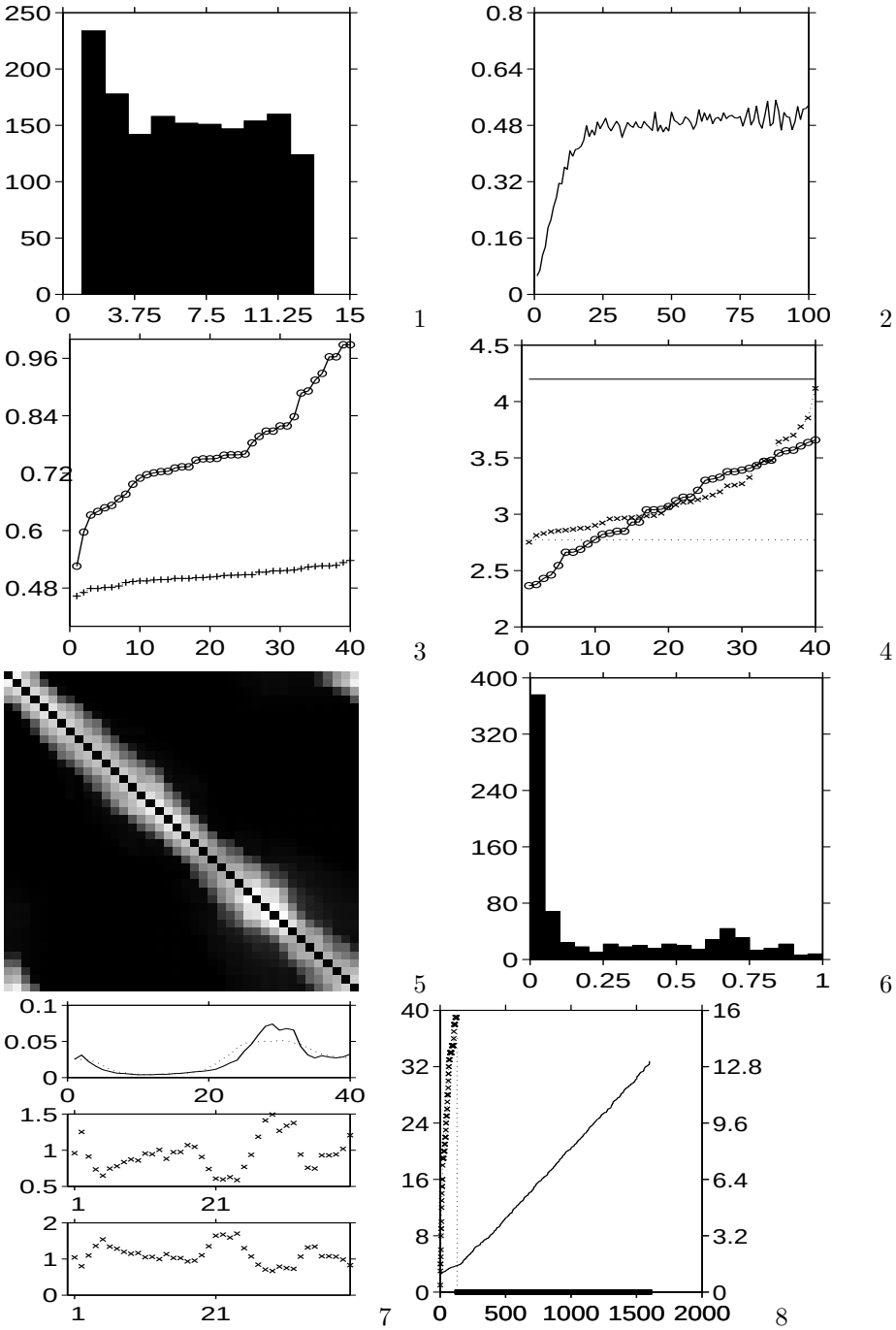


Figure 66: **Spike, feedback layer: moving bar input with $L = 26$ and 40 neurons**
For figure details see section 0.4. The system learns the input, because the 'inactive period' is large enough.

Feedforward layer

We investigated the purely feedforward network with random input and linearly decreasing kernel.

At first we used the probabilistic firing model with fixed average number of firing neurons and modified r and θ. We have found the following regions:

- if $r < .5$ the net will have random statistics (Figure 71).
- if $r > .5$ and $\theta < .2$ then the net will have approximately as many horizontal rows as many neurons can fire, i.e. $N * \theta$. (Figures 67 - 70). The strength of rows is increasing with the increasing of the size of time window (Figures 72, 73).

We used the default parameter values of Table 2 except for $L = 2$, r and θ.

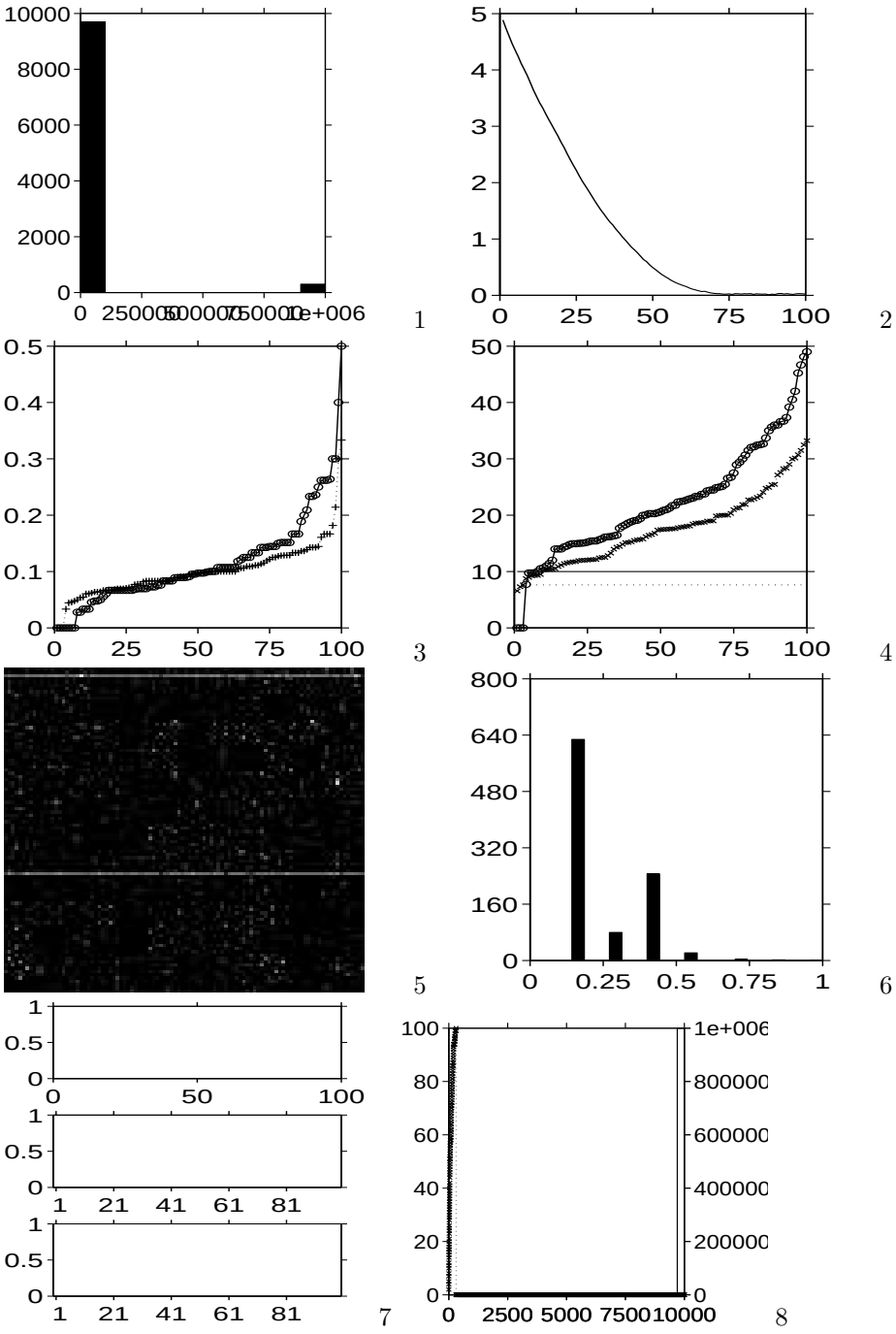


Figure 67: **Spike, feedforward layer: random input with $r = 0.9$ and $\theta = 0.02$**
For figure details see section 0.4.

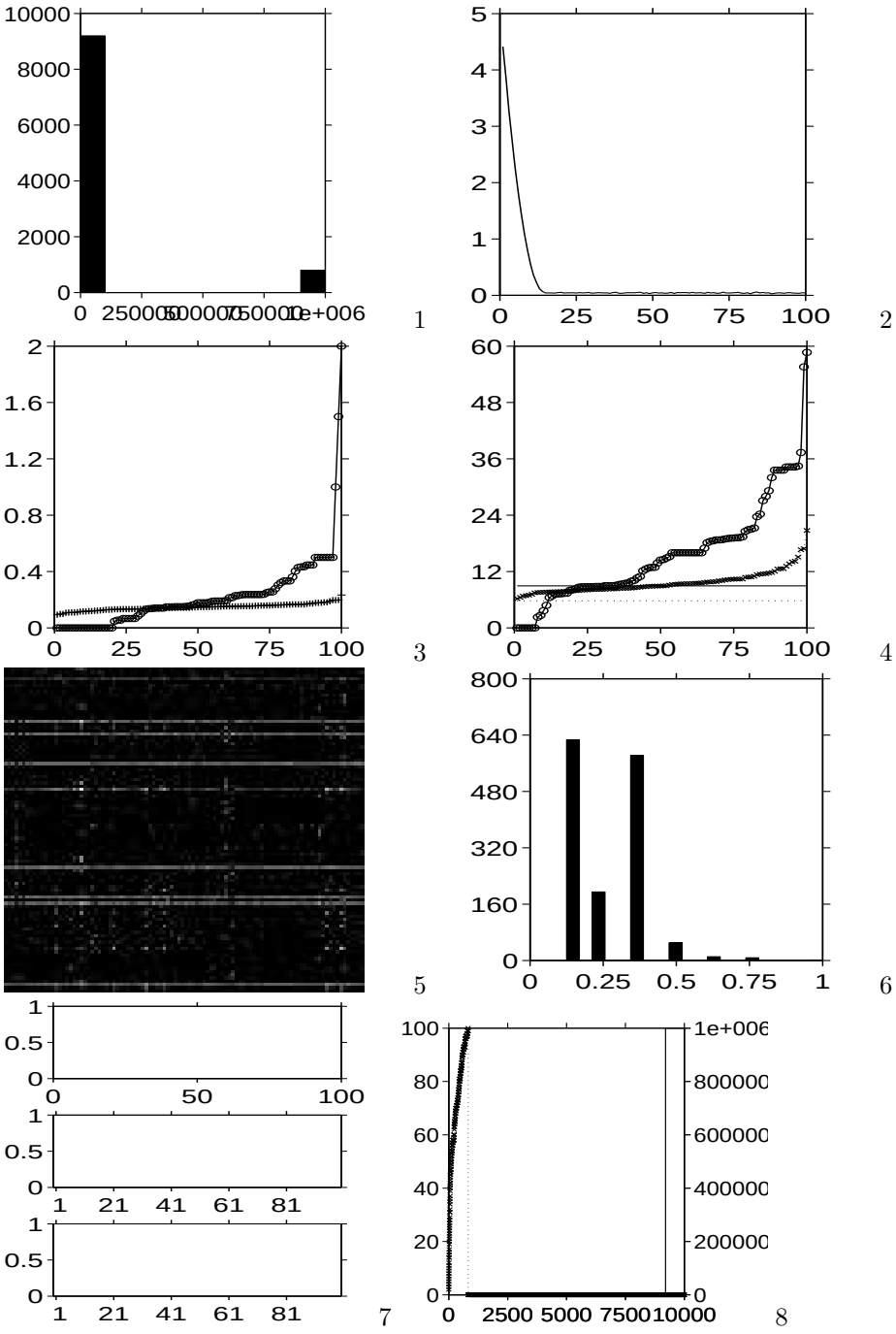


Figure 68: **Spike, feedforward layer: random input with $r = 0.9$ and $\theta = 0.1$**
For figure details see section 0.4.

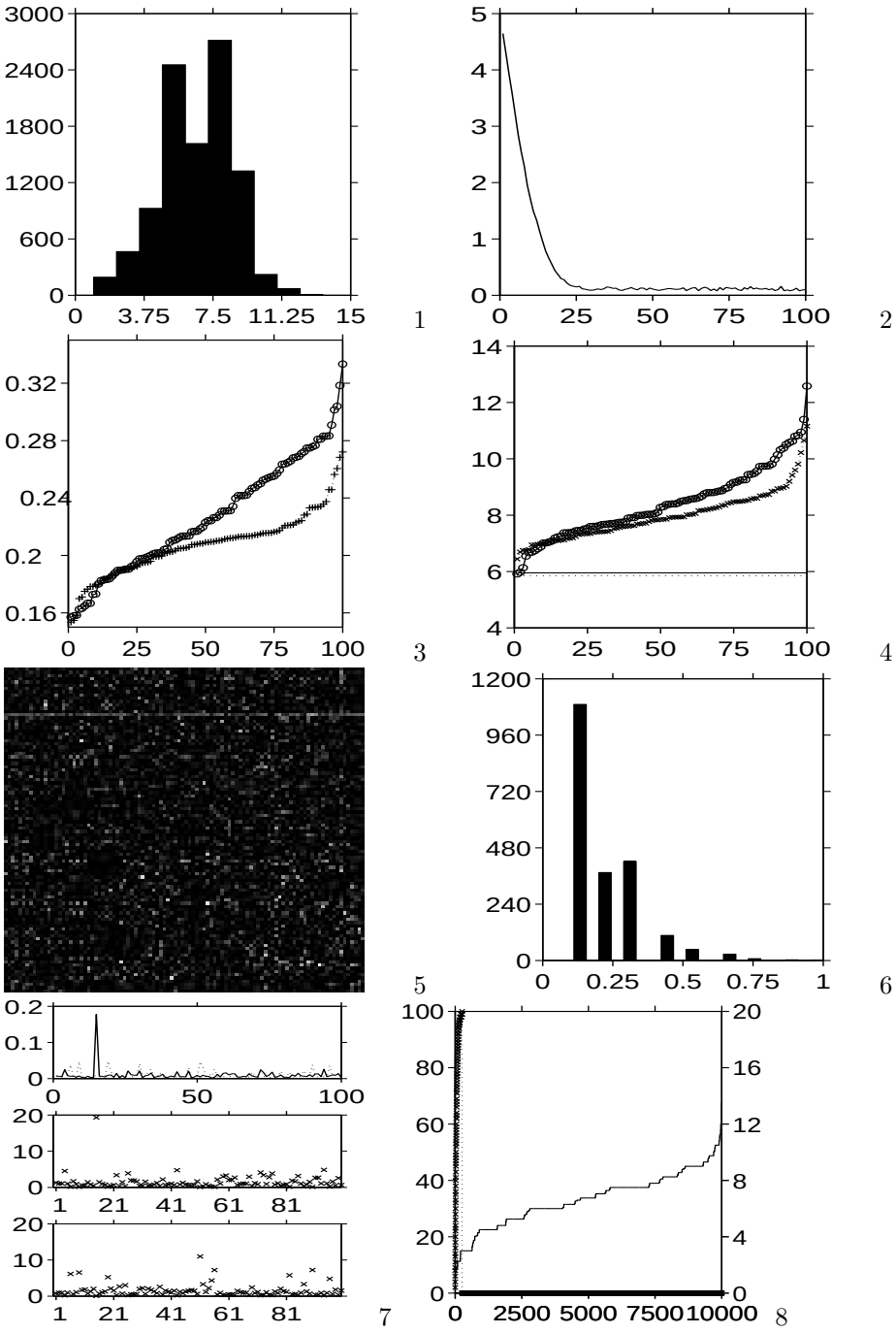


Figure 69: **Spike, feedforward layer: random input with $r = 0.7$ and $\theta = 0.02$**
For figure details see section 0.4.

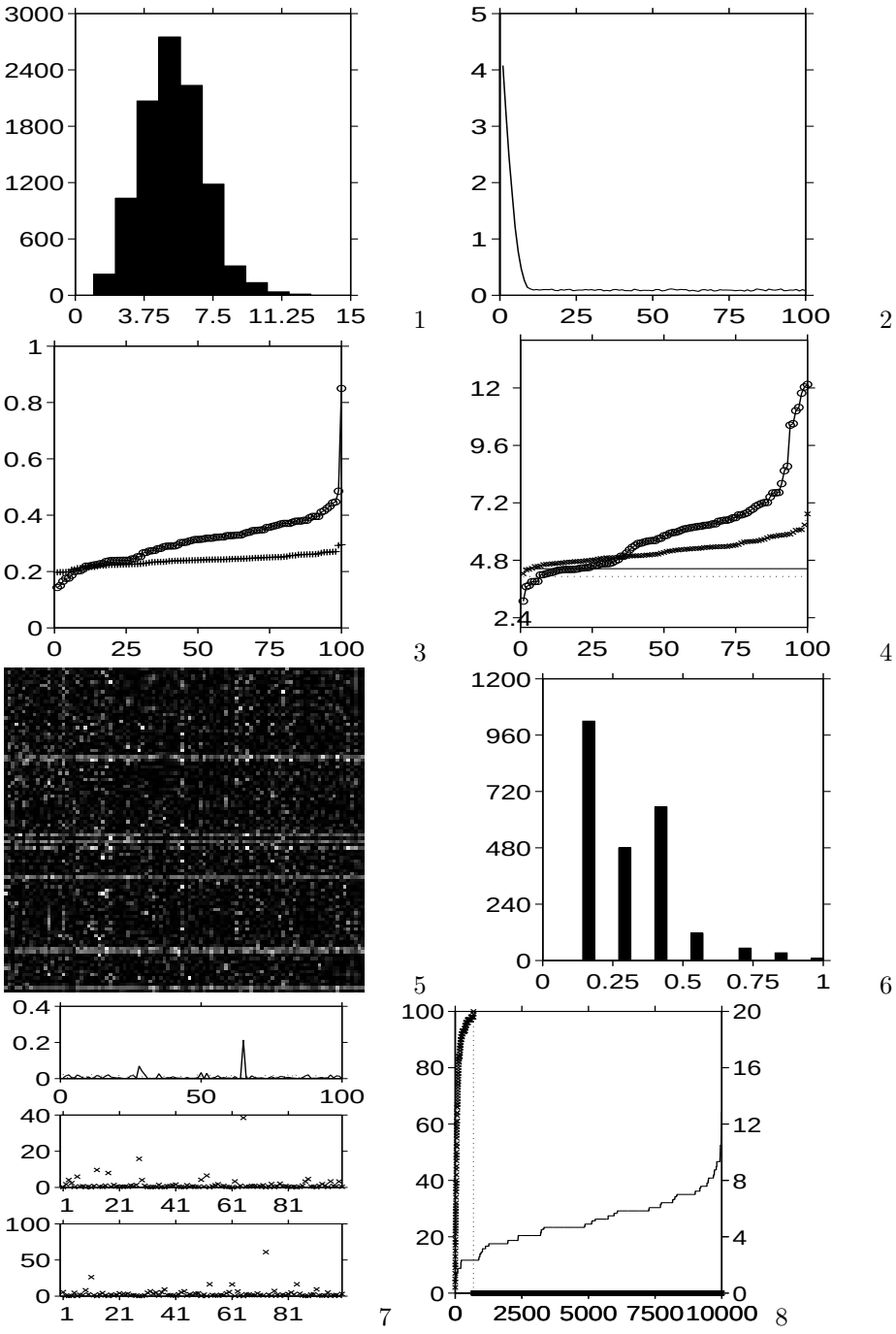


Figure 70: **Spike, feedforward layer: random input with $r = 0.7$ and $\theta = 0.1$**
For figure details see section 0.4.

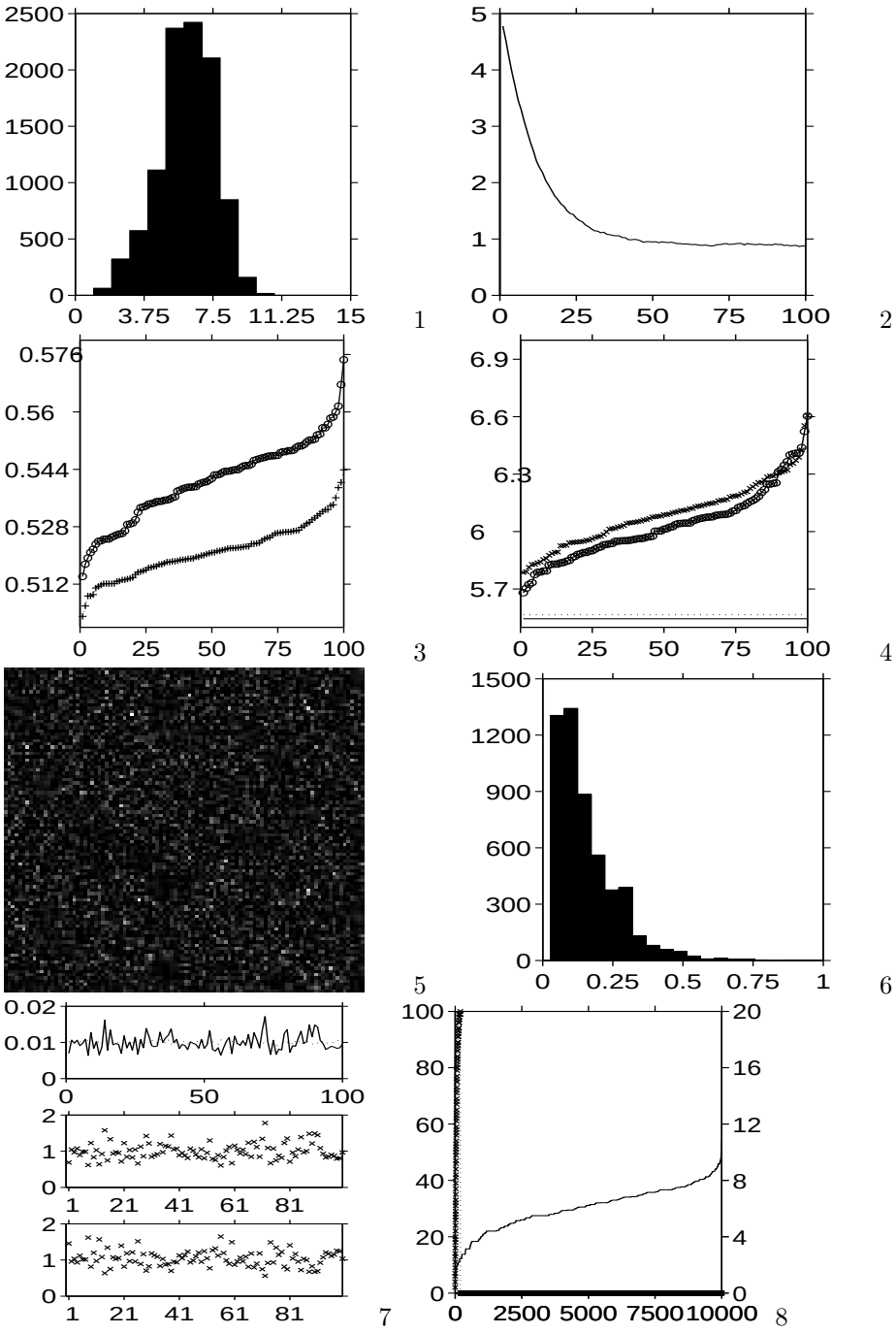


Figure 71: **Spike, feedforward layer: random input with $r = 0.3$ and $\theta = 0.02$**
For figure details see section 0.4. The input is too sparse to generate rows.

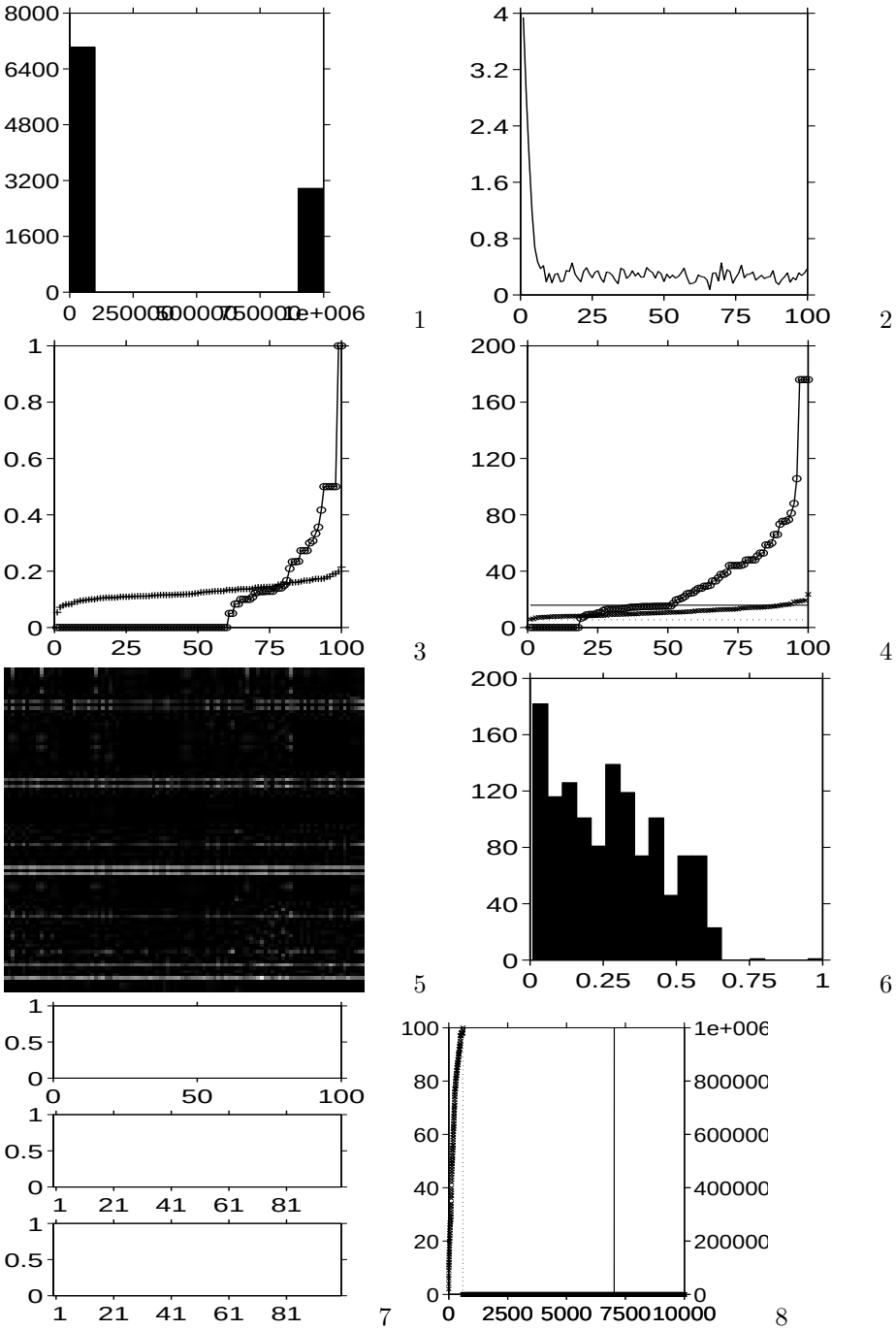


Figure 72: **Spike, feedforward layer: random input with $r = 0.7$ and $\theta = 0.02$ and $L = 10$**
For figure details see section 0.4. Now the kernel is large enough to generate rows.

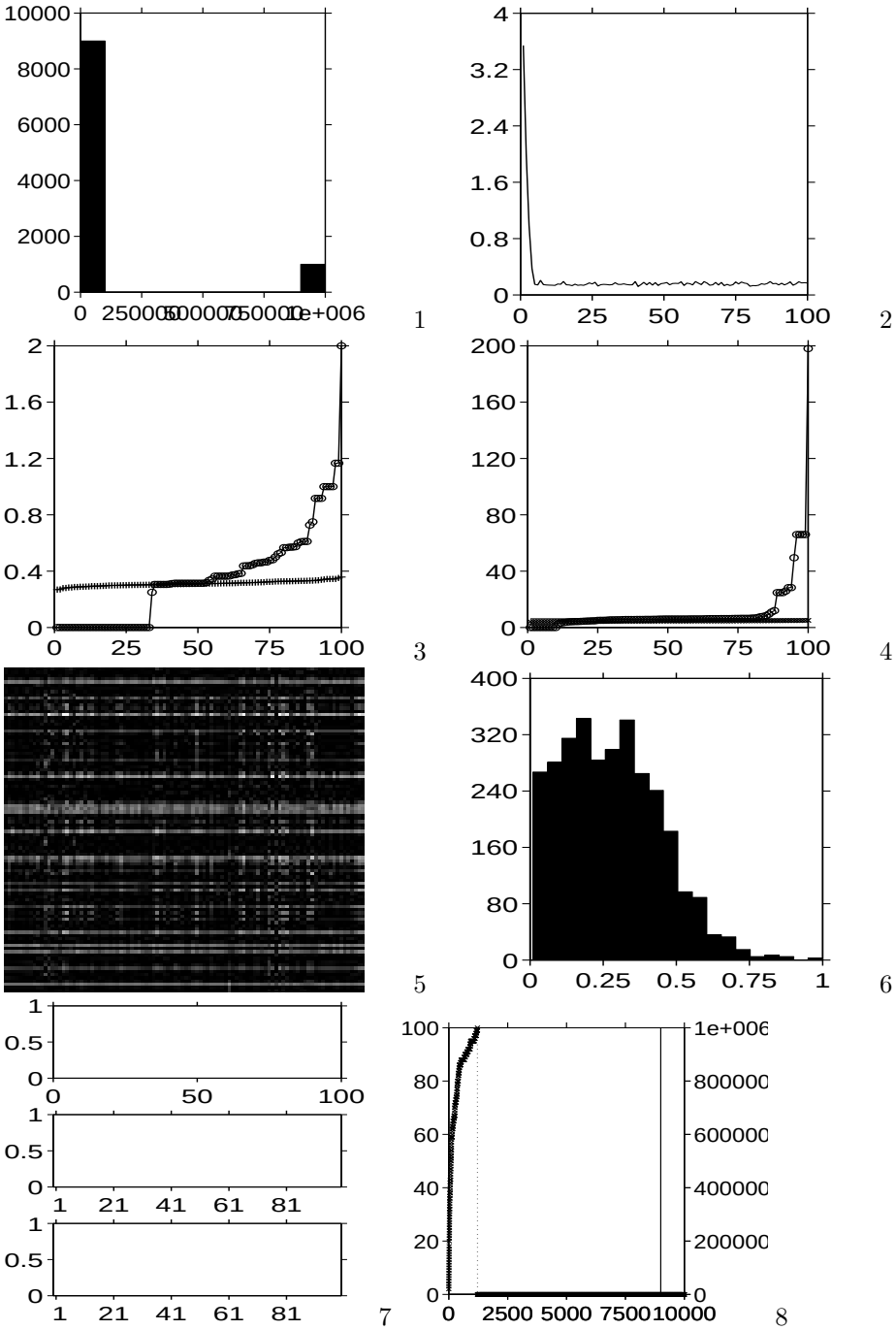


Figure 73: **Spike, feedforward layer: random input with $r = 0.7$ and $\theta = 0.1$ and $L = 10$**
For figure details see section 0.4. Now the kernel is large enough to generate rows.

Next we studied the simple threshold firing model. Depending on r and θ there are several possible cases - with $\theta = 0.5$ and $r = 0.9$ the system is cleared, all weights were 0. See Figures 74 - 82. The $finite\_input$ variable must be set 1 to produce this results, otherwise the net had random statistics. We used the default parameter values of Table 2 except for $L = 10$, r and θ.

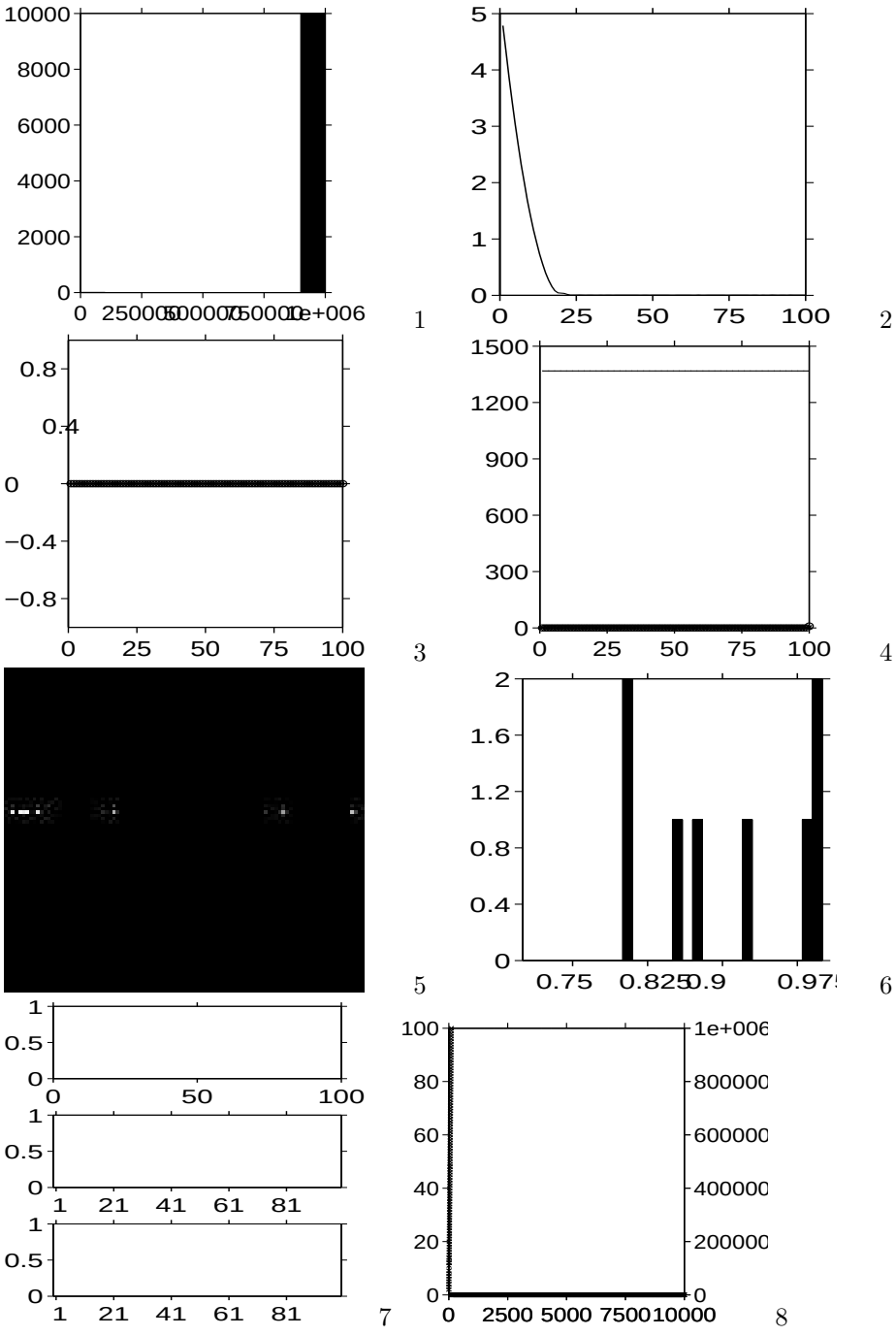


Figure 74: **Spike, feedforward layer: random input with $r = 0.9$ and $\theta = 1.1$.**
For figure details see section 0.4.

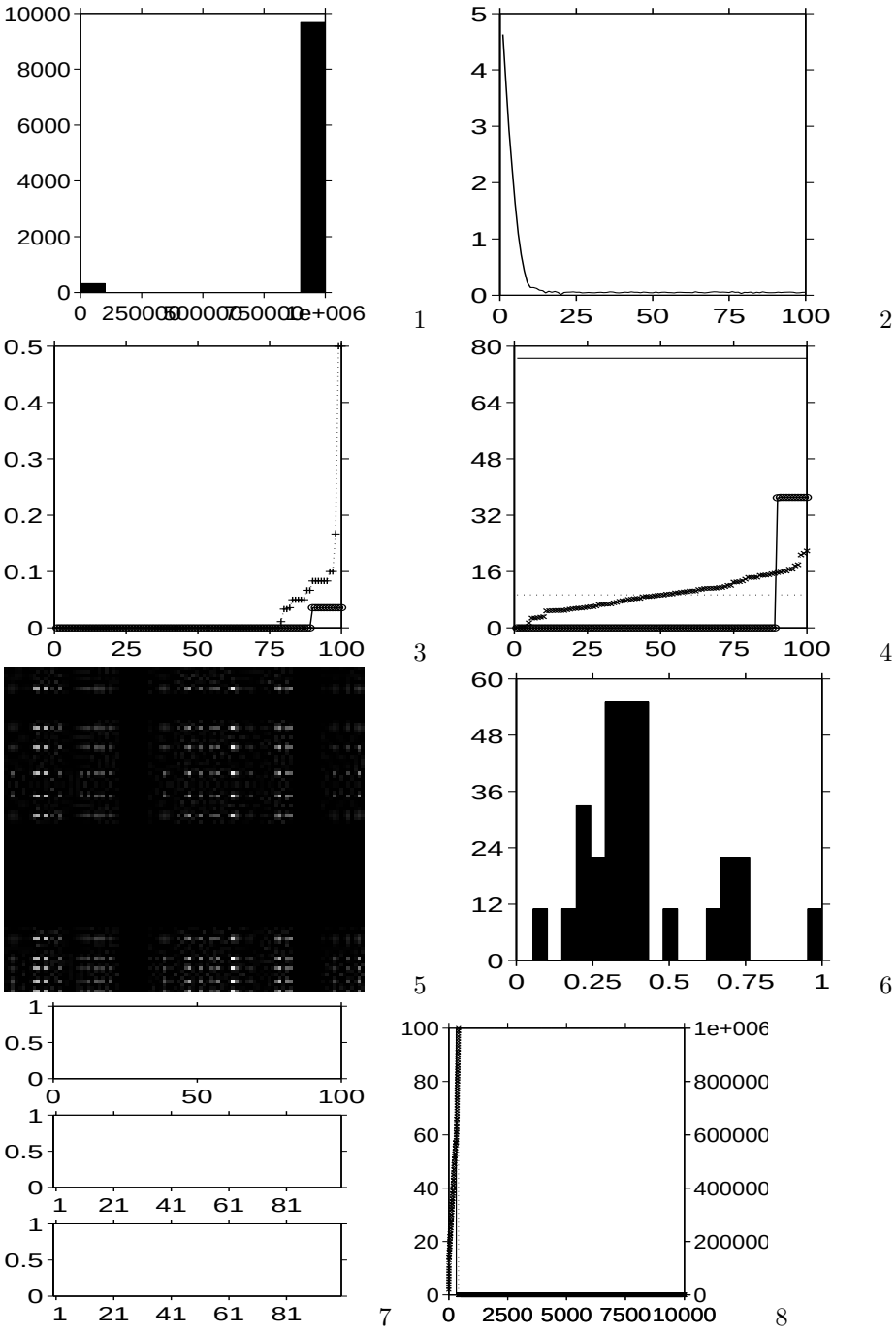


Figure 75: **Spike, feedforward layer: random input with $r = 0.7$ and $\theta = 1.1$.**
For figure details see section 0.4.

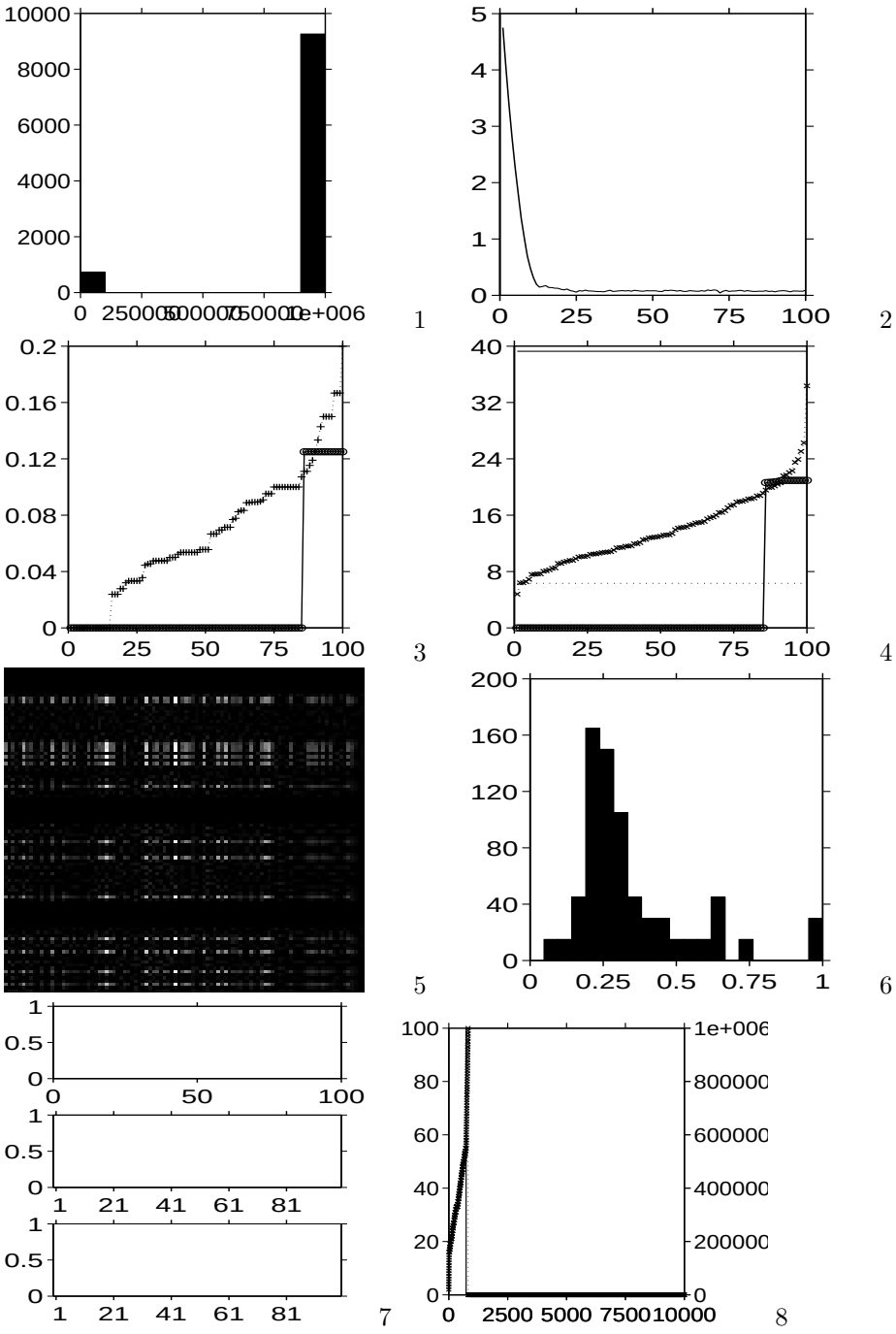


Figure 76: **Spike, feedforward layer: random input with $r = 0.5$ and $\theta = 1.1$.**
For figure details see section 0.4.

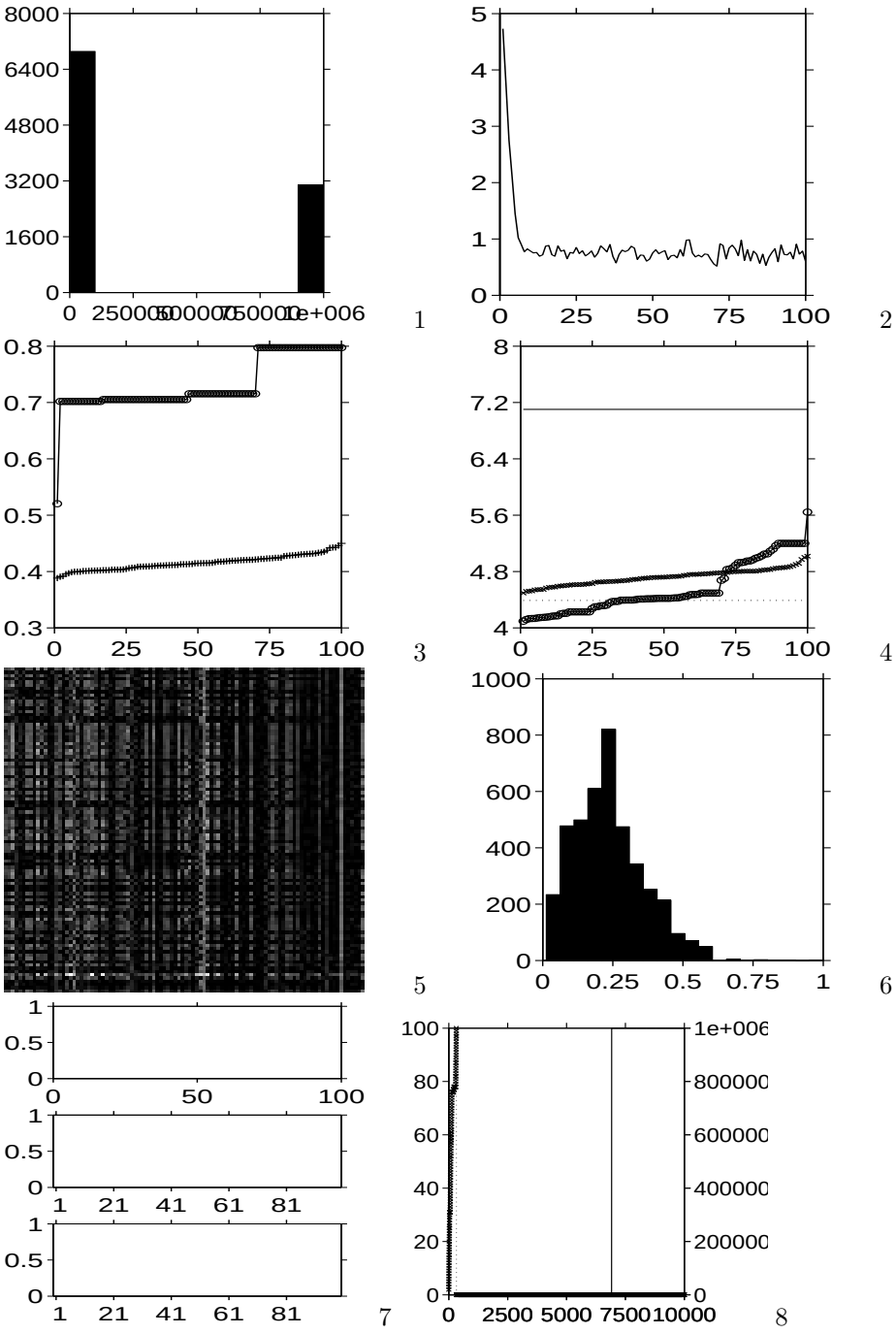


Figure 77: **Spike, feedforward layer: random input with $r = 0.3$ and $\theta = 1.1$.**
For figure details see section 0.4.

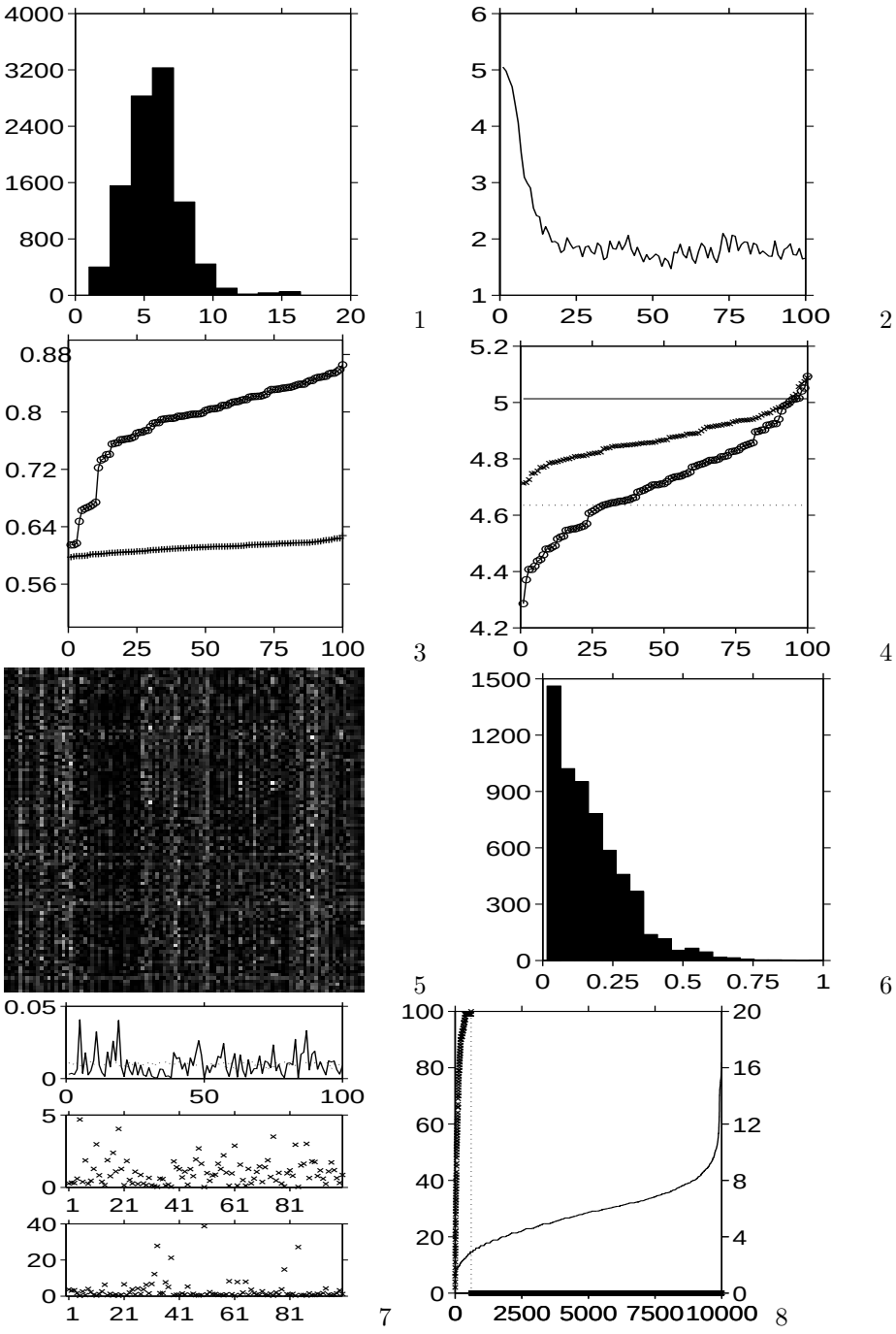


Figure 78: **Spike, feedforward layer: random input with $r = 0.1$ and $\theta = 1.1$.**
For figure details see section 0.4.

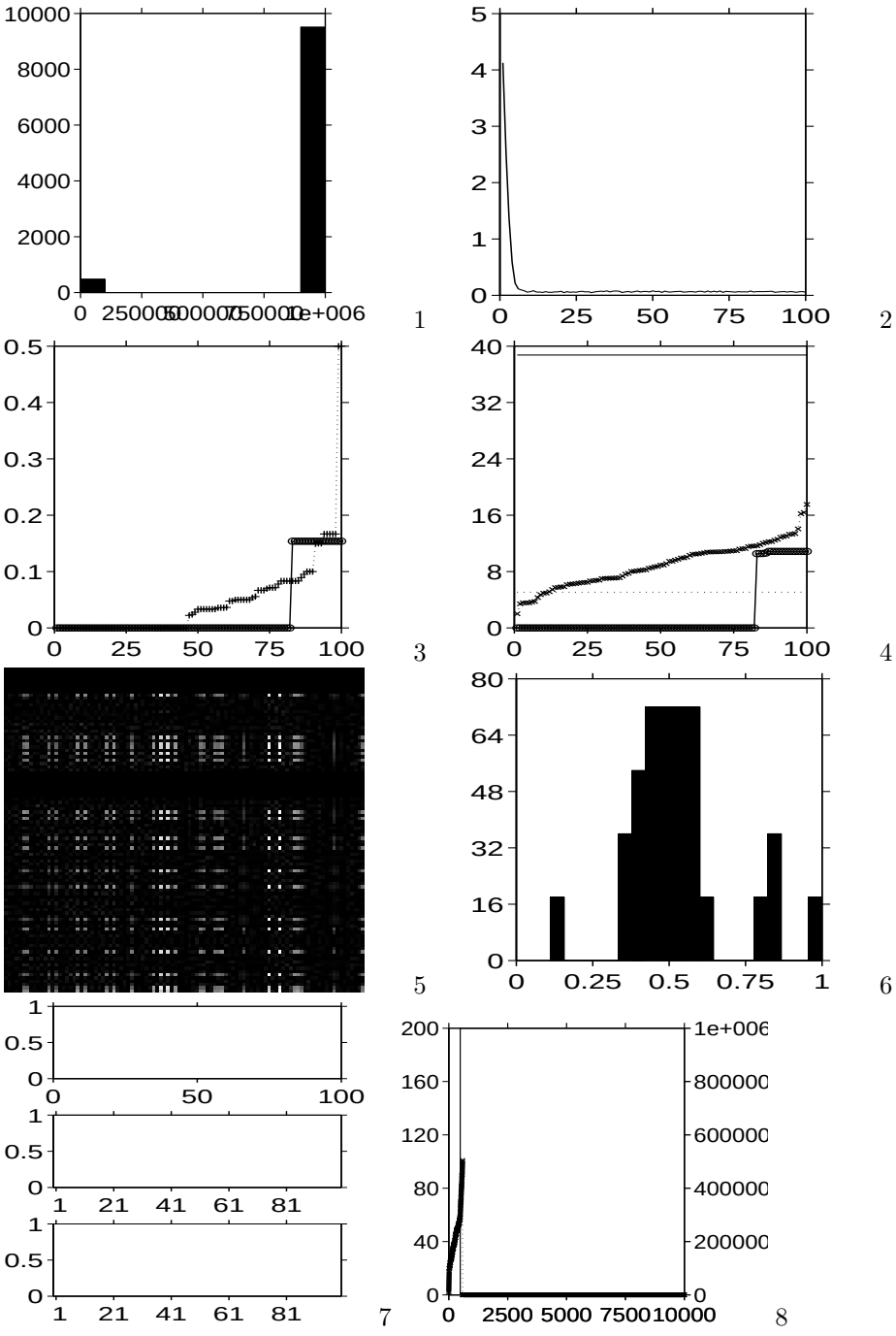


Figure 79: **Spike, feedforward layer: random input with $r = 0.7$ and $\theta = 0.5$.**
For figure details see section 0.4.

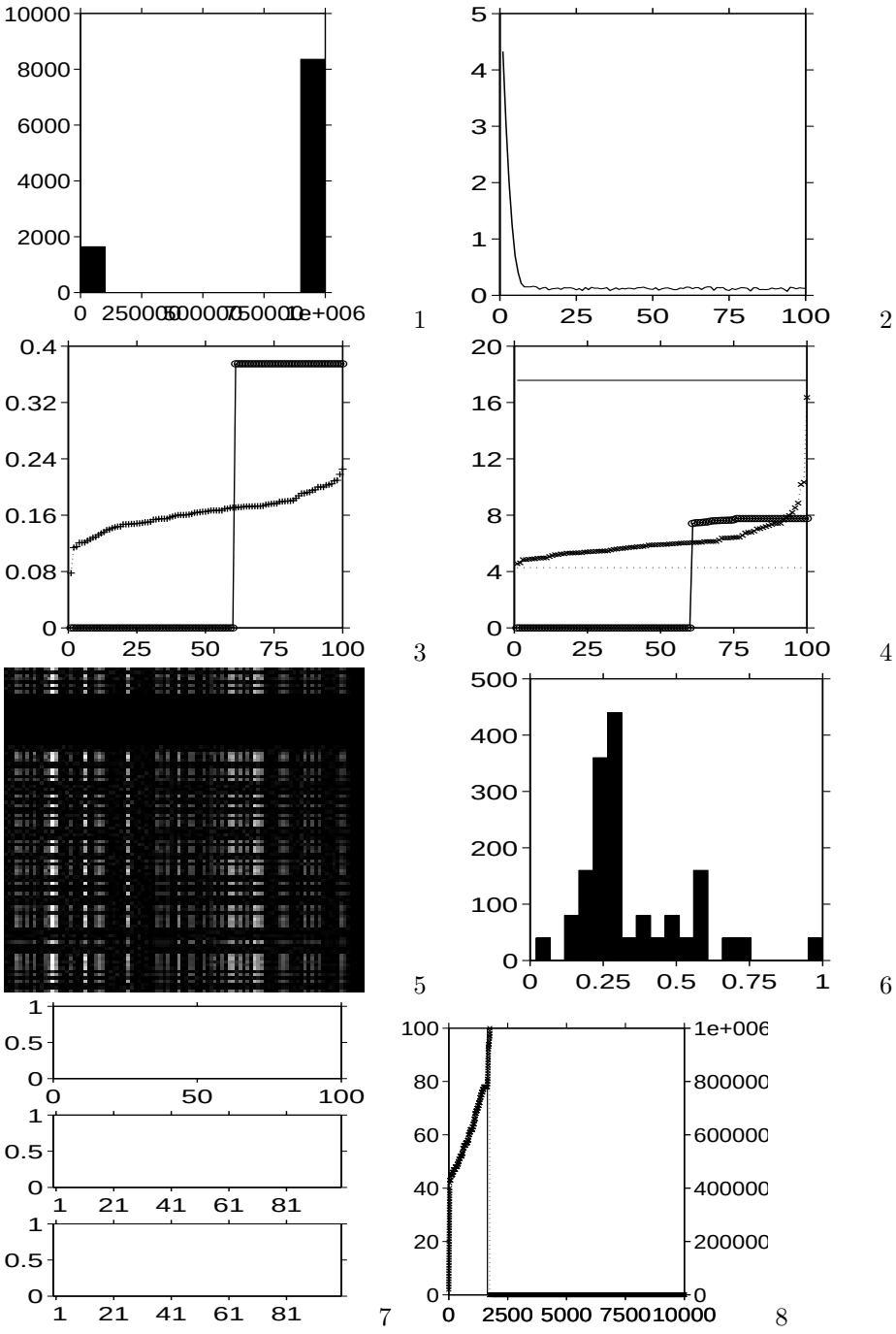


Figure 80: **Spike, feedforward layer: random input with $r = 0.5$ and $\theta = 0.5$.**
For figure details see section 0.4.

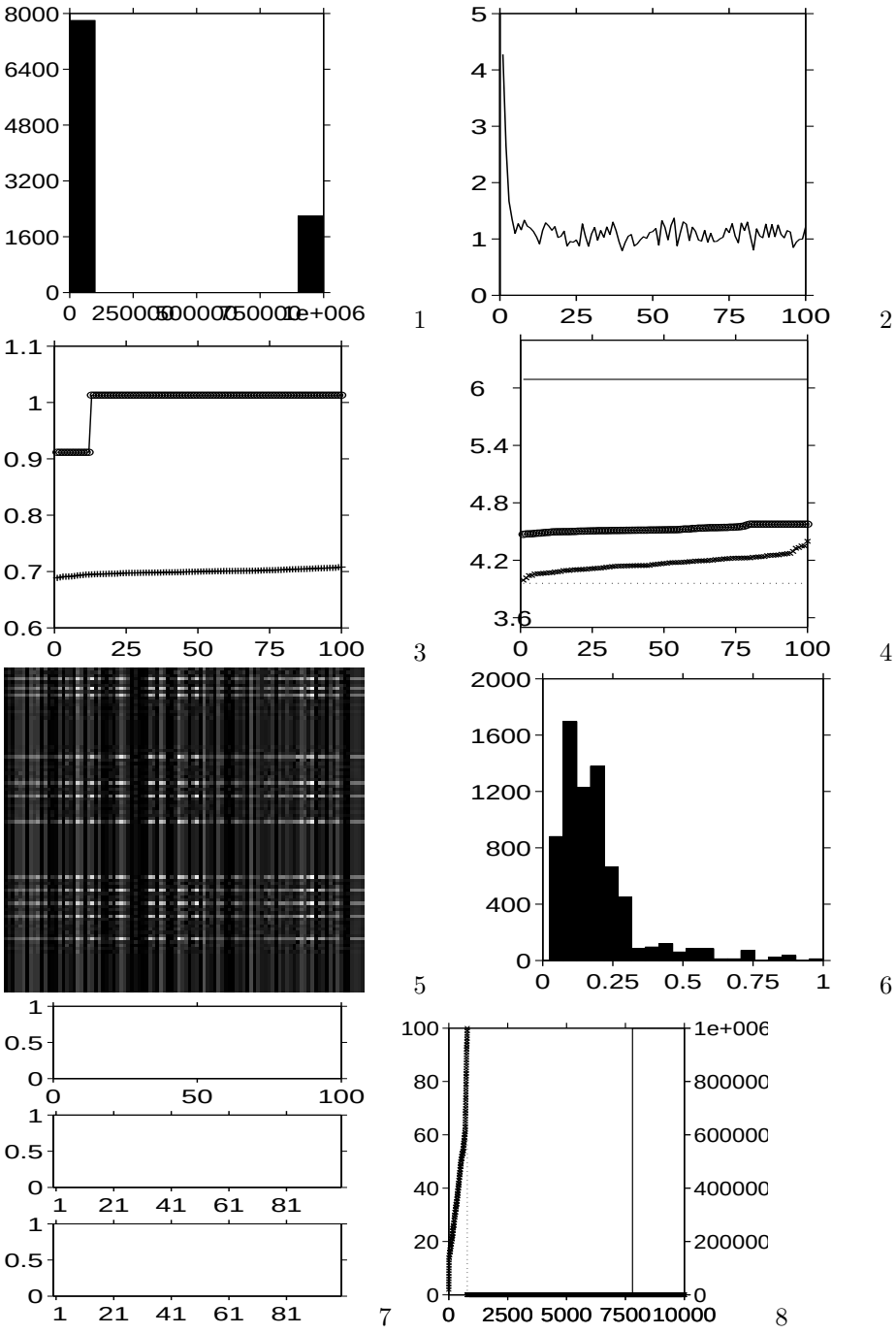


Figure 81: **Spike, feedforward layer: random input with $r = 0.3$ and $\theta = 0.5$.**
For figure details see section 0.4.

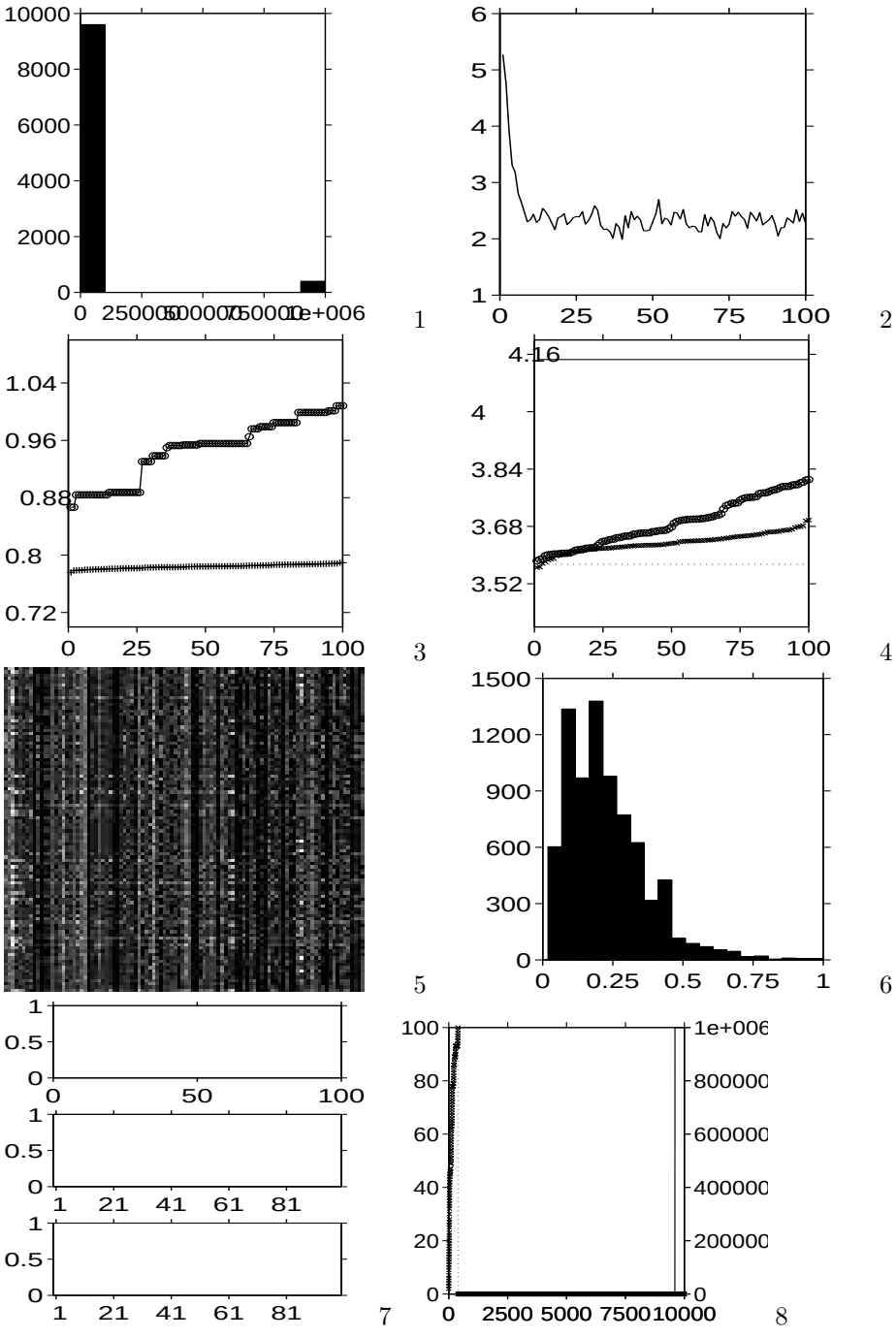


Figure 82: **Spike, feedforward layer: random input with $r = 0.1$ and $\theta = 0.5$.**
For figure details see section 0.4.

Summary

Conclusion on α, β

- If the order of values in weight matrix is in the order of α (β) then the system can not learn anything, become stabilized, independently from α (β)
- If the values in weight matrix are larger with some orders than α (β) and the system starts to oscillate then by decreasing α (β) the system can be stabilized

Bibliography

[1] R. Albert, H. Jeong, and A. L. Barabsi, *Diameter of the world wide web*, Nature **401** (1999), 130–131.

[2] R. Ferrer i Cancho and R.V. Sol, *Optimization in complex networks*, Santa Fe Institute Working Paper, Dec. 2001.

[3] J. Kleinberg, *Authoritative sources in a hyperlinked environment*, 9th ACM-SIAM Symposium on Discrete Algorithms, 1998, pp. 668–677.

[4] R. G. Kulkarni, R. R. Stough, and K. E. Haynes, *Towards modeling of communities of practice (cops) a hebbian learning approach to organizational learning*, Technological Forecasting and Social Change **64** (2000), 7183.

[5] P.T. Quinlan, *Structural change and development in real and artificial neural networks*, Neural Networks **11** (1998), 577–599.

[6] D. J. Watts and S. H. Strogatz, *Collective dynamics of small-world networks*, Nature **393** (1998), 440–442.

REPORT: MODEL SYSTEM FOR NP-COMPLETE PROBLEMS

REPORT EOARD-NIPG-ELU-03-JUNE-2002
CONTRACT: EOARD F61775-01-WE023

ADMINISTRATIVE CONTACT: ISTVÁN ALFÖLDI, NJSZT

PRINCIPAL INVESTIGATOR: DR. HABIL. ANDRÁS LŐRINCZ

Abstract. Recently, a novel algorithm, Stage, has been proposed for optimization problems. Stage exhibits impelling performance on a variety of tasks. Stage makes use of tunable function approximator (e.g., an artificial neural network, Fapp) to improve local search policies; and it can be seen as an efficient (and biased) model of reinforcement learning. However, Stage also has some disadvantages: it can be unstable, and it is a serial (i.e., non-parallel) technique. Another line of recent studies offer a solution here. These studies show that for NP-complete problems randomization of starting points is advantageous. Here, the combined technique of genetic algorithm (GA) using local searches and Stage. We have designed a three stage memory procedure – called Stagenis in the search problem: (i) local search forms the short-term memory, (ii) Stage is the medium-term memory, which aims to uncover the underlying global structure of the search problem, (iii) GA is the long-term memory, which stabilizes and selects found structures and/or schemas of those structures. In this algorithm Stage is embedded into the GA using a new genetic operator, called Stagenis. Demonstrations, which show the combined advantages of these algorithms, are provided on benchmarks of the Boolean satisfiability problem. Connection to the NFL theorem is made.

Another direction of research is concerned by developing Fapps for fast evaluation of a *family* of optimization problems. It is shown that such Fapps can be developed, and gains in speed of optimization can be achieved. In particular, large gains can be achieved for satisfiability problems having hard and soft constraints, where some of the constraints are mandatory, whereas others are *only* desirable constraints. The goal is to find *rational choices* fast. It was found, too, that the development of Fapps for families of problems with unknown families is not possible without specific further improvement of the algorithms. In particular, the winner-takes-all algorithm is not capable for partitioning the problem state into problem clusters with different Fapps belonging to each cluster. Our results pinpoint to the biased nature of the search applied by the global optimization scheme Stage. We shall argue that Stage or Stagenis can be augmented by a GA on the top. In turn, the suggested four stage algorithm would have a GA not to loose the best solutions, Fapps to find the partitioning of the problem state into clusters of similar problems and a GA for the optimization of the low-dimensional parameter space of the Fapps to overcome the biased nature of algorithm Stage. Computer demonstrations of this suggestion are outside of the scope of the present project – where, according to the basic assumptions, clusters of satisfiability problems can be identified.

Date: 3rd June 2002.

Author: András Lőrincz

Contents

List of Figures

9 **Different FAPPs tried on problems par8 and par8c with threshold = 12**
(a)-(f): Six different problems, par8–1 —- par8–3. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database. 23

10 **Different FAPPs tried on problem aim–200–1–6–yes1 with threshold = 7**
(a)-(d): Four different problems, aim–200–1–6–yes1–1 — aim–200–1–6–yes1–4. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database. 24

11 **Different FAPPs tried on problem aim–200–2–0–yes1 with threshold = 7**
(a)-(d): Four different problems, aim–200–2–0–yes1–1 — aim–200–2–0–yes1–4. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database. 25

12 **Different FAPPs tried on problem aim–200–3–4–yes1 with threshold = 7**
(a)-(d): Four different problems, aim–200–3–4–yes1–1 — aim–200–3–4–yes1–4. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database. 26

13 **Different FAPPs tried on problem aim–200–6–0–yes1 with threshold = 7**
(a)-(d): Four different problems, aim–200–6–0–yes1–1 — aim–200–6–0–yes1–4. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database. 27

14 **Different FAPPs tried on problems par8 and par8c with threshold = 7**
(a)-(f): Six different problems, par8–1 — par8–3. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for

'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database.

15 **Different FAPPs tried on problem aim–200–1–6–yes1 with threshold** $= 4$
(a)-(d): Four different problems, aim–200–1–6–yes1–1 — aim–200–1–6–yes1–4. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database.

16 **Different FAPPs tried on problem aim–200–2–0–yes1 with threshold** $= 4$
(a)-(d): Four different problems, aim–200–2–0–yes1–1 — aim–200–2–0–yes1–4. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database.

17 **Different FAPPs tried on problem aim–200–3–4–yes1 with threshold** $= 4$
(a)-(d): Four different problems, aim–200–3–4–yes1–1 — aim–200–3–4–yes1–4. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database.

18 **Different FAPPs tried on problem aim–200–6–0–yes1 with threshold** $= 4$
(a)-(d): Four different problems, aim–200–6–0–yes1–1 — aim–200–6–0–yes1–4. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database.

19 **Different FAPPs tried on problems par8 and par8c with threshold** $= 4$
(a)-(f): Six different problems, par8–1 — par8–3. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database.

20 **Different FAPPs tried on problem aim–200–1–6–yes1 with threshold** $= 2$
(a)-(d): Four different problems, aim–200–1–6–yes1–1 — aim–200–1–6–yes1–4. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third

1. Introduction

Several engineering applications involve NP-hard optimization problems, in which a heuristic is used to find the optimum or near-optimum of a cost or objective function. (It will be assumed throughout the paper that a function $Obj : X \to \mathbb{R}$ is given on the state space X, and the goal is to find the minimum of Obj.) There are many approximation techniques aiming to overcome time requirements of exhaustive searches. For a recent review of such approximations see, e.g., [Hochbaum, 1995]. Recently, a novel technique called Stage using function approximators (FAPPs), e.g., artificial neural networks, and a specific approximation of reinforcement learning (RL, [Sutton and Barto, 1998]) has been suggested [Boyan, 1998, Boyan and Moore, 2000]. Stage, indeed, showed attractive properties in our studies on real-world engineering problems, too [Palotai et al., 2001, Ziegler et al., 2001b].

Stage was able to solve hard problems, but it had its weaknesses. Stage can often recognize the structure of the state space and make use of this structure to guide the search to better regions of the search space quickly. However, it can sometimes be unstable and unreliable. Also, Stage is not a parallel technique.

This motivates the development of our new algorithm, which combines genetic algorithm (GA) and Stage. In this version of GA, called GA with Stagenis (GAw-Stagenis, or simply GAS) , the fitness of the individuals is determined as the best value found by a local search (LS) starting from the state the individual represents. New individuals are introduced by the standard crossover operator of GA and by a new genetic operator 'Stagenis' (*i.e.*, Stage-assisted genesis). Mutation is covered by this Stagenis operation. Stage makes use a FAPP to approximate and smoothen the objective function. GAS makes use of the same technique. In the Stagenis operation first a LS is executed on the FAPP and then an individual is generated at the minimum of this FAPP[1].

Several tests and measurements have been conducted to evaluate the performance of the new algorithm. We used Boolean satisfiability benchmarks from the DIMACS archive [DIM, 1992]. As local search policy, the algorithm WalkSat was used [Selman et al., 1996].

Another aspect of our work concerns experimental justification of recent findings on NP-hard problems [Gomes et al., 1998, Gomes et al., 2000]. We show that the approximated objective function improves performance of the randomization technique in this benchmark problem set. We shall consider this randomization approach, Stage, and Stagenis from the point of view of the No Free Lunch (NFL) theorem [Wolpert and Macready, 1997].

Report organization: In section 2 we first present the base algorithms, *i.e.* WalkSat, Stage and GA, laying special emphasis on their advantages and disadvantages that motivated our work. Section 3 describes how these algorithms are combined to form GAS. Rational choice considerations are provided in Section 4. Implementation details as well as the results of our empirical evaluation are covered in Section 5. In the discussion section (Section 6) we shall review the properties of our

[1] A mapping which renders an action to a state of the state-space in a general search problem, e.g., a mapping which renders an operation to a population in GA can be recast in the the form of 'policy', the basic concept of RL [Sutton and Barto, 1998]. Oftentimes, we shall use the word policy to refer to such mapping.

approach. Section 7 concludes the report. The Appendix contains the pseudo-code of some of the algorithms of the paper.

2. The base algorithms: WalkSat, Stage and GA

One of the most successful local search methods for solving satisfiability problems is WalkSat [Selman et al., 1996]. Therefore, we used it in all of our algorithms as LS policy.

Boolean satisfiability problems are often stated in Conjunctive Normal Form (CNF). A CNF contains elementary expressions made of singe Boolean variables that might be negated. Elementary operations are grouped in clauses by means of logical OR operators. The CNF formula contains such clauses connected by logical AND operators. The CNF is satisfiable, if there is a consistent assignment of truth values to all the Boolean variables of the CNF that makes the whole CNF formula 'true'. This also implies that all individual clauses of the CNF should be 'true', i.e., satisfied.

Given a CNF formula, WalkSat conducts a random walk in the space of possible truth assignments. The aim is to minimize

$$Obj(x) = \text{number of clauses unsatisfied by assignment } x.$$

WalkSat works in *rounds*. It starts a round by selecting an unsatisfied clause randomly, and tries to flip each variable in this clause, one at a time. It selects the best change greedily according to the overall improvement of Obj. If no improvement is possible, then it selects stochastically either the least worsening step, or a random step. If the worsening step exceeds a limit then another random unsatisfied clause is tried. Details on WalkSat are given in the Appendix.

WalkSat uses 4 parameters; the default values are chosen according to [Boyan, 1998]:

Cutoff: The probability of stopping after each round. This parameter determines the expected average length of the trajectories, which is $1/cutoff$. The default *cutoff* is the reciprocal value of the number of clauses, thus the expected average length of the trajectories equals the number of clauses.

Patience: The number of trials to find a next step suitable for starting a new round. If the state does not change for *patience* steps, then the algorithm terminates. The default value is 100.

Noise: If no improving step could be found, then with this probability a random step is chosen. The default value is 0.25.

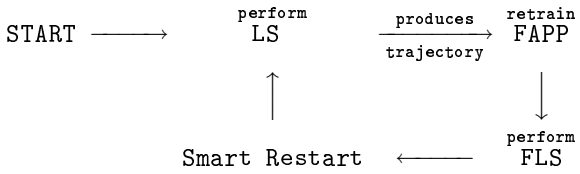
Delta: The maximum allowed worsening of Obj if no improving step is possible. If this parameter is set to 0, then WalkSat is not able to leave local minima, thus resorting to a true local search behavior. The default value is 10.

In our implementation, the original GA (*plainGA*) consists of a population P of N individuals, where each individual is an element of X. In each cycle, a new population is made from the old one using three genetic operations: recombination, selection, and mutation. First, some pairs of individuals are *selected* for recombination with the roulette method (*i.e.* the probability for the selection is proportional with Obj); from each pair two individuals of the new population are created using simple cross-over. The rest of the new population is filled up by keeping the best individuals of the old population. Mutation is performed either by altering some randomly chosen genes of some randomly chosen individuals of the new population

or by replacing some individuals with random elements of X. At the end of the cycle, the old population is discarded, and the recently created population becomes the old population.

The fitness of an individual x can either be determined by $Obj(x)$ or, alternatively, by the result of a LS starting from x. Using the latter approach, GA exploits *and* remembers the results of a particular LS policy π. Also, LS policy π improves the efficiency of GA by releasing the load imposed by *precision* [Ulder et al., 1994, Kammeyer and Belew, 1996, Freisleben and Merz, 1997, Dorne and Hao, 1998].

STAGE, [Boyan, 1998] like GA, works in cycles. Stage has an *inherited* LS policy π_S, builds (corrects) a FAPP from the result(s) of the LS, makes use of its own LS policy on this FAPP, called FAPP-LS or FLS. The result of the FLS is subject to LS policy π_S, and so on. The FAPP is constructed followed by the retraining of the FAPP using the trajectory data from the recent LS run, then another local search on the freshly retrained FAPP . The FLS will produce a promising "smart restart" (SR) point for π_S:



The FAPP is computed to approximate the 'value' of states given LS policy π_S: Any state is worth to the value that can be reached from that point using LS policy π_S. Such state-'value' pairs are created upon each LS and are used to approximate the 'value-function' by means of a FAPP. This FAPP depends on the LS policy and it is not subject to improvements. The FLS policy is subject to iteration[2].

If the same SR is found more than a pre-specified number of times, the algorithm resorts to random restarts (RR) to maintain exploration. The FAPP could be an artificial neural network, but for the sake of simplicity and efficiency, we used a simple quadratic approximator [Boyan, 1998]. STAGE can make use of features, *i.e.*, it can perform the approximation in a space of lower dimension (the feature space).

The advantages of STAGE are manifold:

- No *a priori* knowledge of the problem is necessary, *i.e.* any features, π_S policy, function approximator and FLS policy can be used.
- If *a priori* knowledge is available, it can be incorporated into the LS strategy (i.e., into the concept of neighbors) or into the form of features (condensed forms of regularities, or invariances). Features may lead to an *effective* reduction of the dimension of the search space. This property of STAGE can be advantageous to overcome complexity and the dimensionality of the search problem. Even very simple features – such as the mean and variance of state variables – can be used effectively.
- STAGE weights the features automatically, *i.e.*, it can select the relevant features and discard unimportant ones.

[2]Note that STAGE makes use of RL terminology. The corresponding policy iteration, however, may not always satisfy convergence theorems of RL. For this reason, the approximated 'value-function' will be called 'FAPP, which generates smart restarts'.

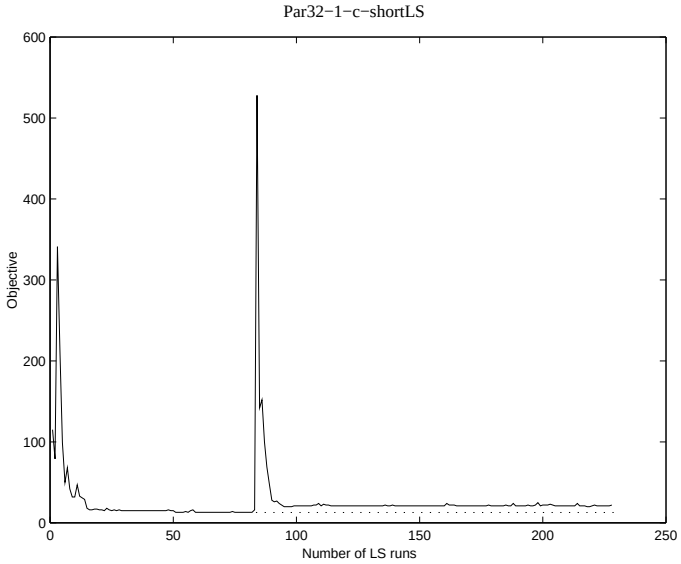


FIGURE 1. Instability of STAGE

- If there is a structure in the objective function, then STAGE may unravel this structure and may make use of it.
- If no structure can be found, STAGE gracefully degrades to a simple random restart strategy.

However, STAGE also has a couple of disadvantages:

- It is unstable, *i.e.* if the structure of the LS smoothened objective function is more complex than the class of functions that the FAPP can approximate, then its predictions may oscillate wildly and chaotically as it has been experienced by the authors themselves in some engineering applications.
- It can definitively drift away from already found good states. Because the memory of STAGE is limited to the FAPP, which is a restricted in storage, STAGE sometimes completely forgets the best state it found and drifts to worse regions of the state space. This phenomenon is caused by the fast but biased approach of STAGE: An SR, for example, may start a long trajectory having a sub-optimal local minimum. The long trajectory may give rise to significant modifications in the FAPP and subsequent SRs may be trapped in the attractor region of this local minimum. As an example, consider the run of STAGE in Figure 1. The horizontal axis corresponds to the sequence of LS runs, the vertical axis shows the found best objective values of the individual LS runs. (Discrete dots are interconnected to enhance visibility.) The first LS runs produced results around 100, then the FAPP learned the structure of the local optima and gave better and better SRs. After the 23^{rd} LS run, the FAPP stabilized at $Obj = 13$, and a semi-stationary operation went until the 83^{rd} run. At this point, however, the FAPP began a transient behavior, during which Obj temporarily worsened to 528. After the transient was gone at the 96^{th} run, the final stationary operation began. No further LS found the previous optimum anymore.

STAGE stagnated around objective values of 20...22 until the program was stopped after 10^6 evaluations.

- Even if structure is present, it may remain unobserved, if the FAPP is unable to represent it. Therefore, the choice of the FAPP is important.
- STAGE is inherently serial. The FAPP is retrained when the LS is completed (because retraining requires the best value found during the LS), the FLS starts when retraining is completed, and the LS starts when the completion of the FLS generates a restart point. Therefore, STAGE is not parallel in its current form.

On the other hand, genetic algorithms have the following advantages:

- They are robust to noise and transients. Since the population can store past results, and thus the new population depends not only on the current state, but also on the past, GA performs a kind of temporal integration during the search. Therefore, GA can be made robust against noise and infrequent transients. GA has the potential to stay in good regions of X.
- They are proven to converge in the limit and to make use of implicit parallelism. (See *e.g.* [Baum et al., 1995] and references therein.)
- They can be applied to any optimization problem, without any *a priori* knowledge.

However, they have the following disadvantages:

- The effectiveness depends heavily on the encoding of states. Most notably, the recombination operation works effectively only if the encoding enables the identification of (loosely interacting) modules in the genom code.
- The effectiveness depends heavily on parameter settings.
- GA often converges very slowly because of the high number of unnecessary computations (which may be partly attributed to also following not really promising directions). This is a negative side-effect of the integration in time domain, and thus the price for robustness.
- The population often gets degenerated, *i.e.* the population loses its diversity. In this case, some good but not necessarily optimal traits spread across the whole population, preventing further exploration.

Some of the disadvantages of GA can be compensated using LSs [Ulder et al., 1994, Kammeyer and Belew, 1996, Freisleben and Merz, 1997, Dorne and Hao, 1998], as it has been mentioned before. This change can boost the performance of GA. This is in accordance with recent results of Gomes *et al* [Gomes et al., 1998, Gomes, 2000, Gomes et al., 2000], stating that in combinatorial search problems, 'local search' strategies (WalkSat, for example) benefit from frequent random restarts and many short parallel runs, especially in the case of heavy-tailed runtime distributions that are typical of NP-complete problems. Therefore, GA, which starts by launching parallel random LS runs can be advantageous in NP complete problems.

3. GA WITH STAGENIS (GAS)

The advantages and disadvantages of STAGE and GA are largely complementary to each other, so that the unification of the two algorithms can be expected to be stable and parallel as GA, and smart as STAGE. On the other hand, since genetic algorithms are bound to give a chance to less promising individuals as well (which is the source of their robustness), they tend to perform unnecessary computations,

making GAS somewhat slower than STAGE. Of course, this can be compensated in the case of GAS if enough parallel processors are available.

The combination of the two algorithms was realized by embedding STAGE into the GA as a new genetic operation. That is, the genetic algorithm is kept as a framework, but a FAPP is trained with the trajectories of the LSs that are used to calculate the fitness of the new individuals. FLS is used to obtain new SR states. These new SR states are injected into the population. We call this operation Stagenis (STAGE-assisted genesis). The population can be regarded as a container for the starting points of LSs. This solves the problem that STAGE sometimes 'forgets' good solutions (see section 2), and thus stability can be improved. Moreover, the standard genetic operations also help optimizing the starting points: recombination generates new individuals from the old ones, and selection makes sure that only those individuals will be kept from which the most successful LSs can be started. Mutation was discarded in favor of Stagenis because SRs (or, in the worst case, RRs) suffice to guarantee variety.

GAS works as follows. First, a population P of N individuals is created. These individuals are partly created randomly, after which their fitness is calculated by conducting a LS starting from them. The LS trajectories are then used to train the FAPP, and the remaining individuals are created using Stagenis.

In later cycles, the new population is always constructed from the old one by using the operations: recombination, selection, and Stagenis. Recombination and Stagenis create new individuals, each resulting in a LS and the retraining of the FAPP. In our first implementation, we used Stagenis only for the creation of the last individuals of the new population. However, it turned out to be useful to make use of SRs as early as possible. Therefore, the Stagenis calls are distributed evenly in the code that generates the new population.

For the sake of efficiency, the FAPP is retrained as late as possible, *i.e.* when a new individual has to be created with Stagenis. At this time, all the collected information will be used by the FAPP to create the new Stagenis member. The trajectories are actually cached until that moment, and the FAPP is retrained with all recent trajectories at once.

More details on the implementation are given in our technical report [Ziegler et al., 2001a]. The pseudo-code of GAS can be found in the Appendix.

Parallel form STAGE: Algorithm GAS. One important aspect of GAS is its parallel form, which was one of its most important motivations. In the following paragraphs we describe one (out of many) possible scheme.

Our GA implementation is inherently parallel because the LS runs – that make almost all of the execution time – are completely independent and can thus be executed by different processors, yielding a practically linear speed-up. On the other hand, STAGE is inherently serial; therefore, its parallel formulation is not completely straight-forward. Of course, the bottleneck is the FAPP, because it is a central object that nevertheless has to be retrained by many processes, and is also used to generate new SRs using the FLS.

Fortunately, FLSs can be made parallel just as well as LSs because they require only read-only access to the FAPP, which can be granted to any number of processes simultaneously. We propose to have an extra copy of the FAPP which is always available for reading. This copy is updated in an atomic step after each retraining in order to guarantee that it contains the latest information.

Therefore, the only problem is the retraining of the FAPP, because this definitely requires exclusive write access. The main idea is to minimize the time of the exclusive write access. The problem is that uploading the trajectory data – which can easily be tens or hundreds of kilobytes – to the FAPP can take long time. Instead, we suggest that a copy of the FAPP should be downloaded, edited locally, and then uploaded. The FAPP is usually a concise representation; for instance in the case of an incremental FAPP (such as linear architecture quadratic regression [Boyan, 1998]), it is a constant size matrix. This way, the time interval in which no other write access is allowed, can by very short. Furthermore, the processes could pass the FAPP object to each other in a predefined order (assuming a kind of ring topology), so that the uploading of the FAPP and the next downloading is actually only one step of data transfer. Figure 2 shows the GAS algorithm in the proposed architecture for 3 processors.

In the following calculations, we will use t_R, t_S, t_{LS} according to Figure 2. t_{LS} means the total execution time of several LS and FLS runs that should be made without retraining the FAPP by that particular processor. t_S gives the time necessary to send (or receive) the FAPP object. t_R amounts to the time needed for retrain the FAPP. Let k denote the number of processors that can be used parallel; then the following inequality must hold for k:

$$2t_S + t_{LS} \geq (k-1) \cdot t_R + k \cdot t_S$$

that is

$$1 + \frac{t_{LS} + t_S}{t_R + t_S} \geq k \tag{1}$$

If the processes use shared memory to communicate (in which case the FAPP object does not have to be passed physically between the processes) or the communication time between the processes is negligible compared to the retraining time or the execution time of the LS runs, *i.e.* , $t_s \to 0$, then (1) transforms to:

$$1 + \frac{t_{LS}}{t_R} \geq k \tag{2}$$

That is, the number of usable processors is bounded by the ratio of the execution time of the batched LS/FLS runs and the retraining time. t_R is determined by the type and size of the FAPP, but t_{LS} can be chosen large enough to allow the use of an arbitrary number of processors and achieve almost linear speed-up. As a consequence, the trajectory of an LS run will not be used immediately to retrain the FAPP, but in the worst case only after t_{LS} time.

On the other hand, if the number of processors available is given and the optimal number of batched LS/FLS runs, *i.e.* , t_{LS}, should be determined, (2) can be rephrased:

$$t_{LS} \geq (k-1) \cdot t_R \tag{3}$$

To use the trajectories of the LS runs as soon as possible, t_{LS} should be chosen as small as possible but satisfying (3).

4. Considerations on rational choice

In many cases, it is possible that we can formulate constraints in the form of soft satisfiable instances. The easiest is to provide cost for every clause and to

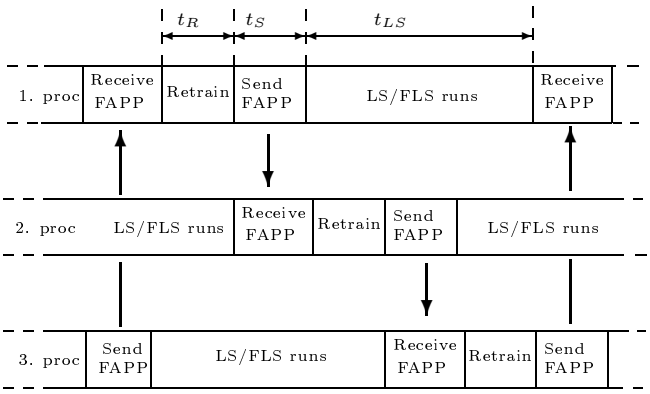


FIGURE 2. Parallel execution of the GAS algorithm in the proposed architecture for 3 processors

search for low cost solutions. Hard constraints will have to exhibit costs, which are *above threshold*, whereas soft constraints may assume real costs in terms of dollars, or time. We have studied such examples for STAGE and for FAPPs developed on different problem families. FAPPs of different problem families were cross checked against each other. Such optimization should be able to find *rational choices* much faster than optimal solutions.

The particular feature set of our problems were chosen as follows:

- proportion of clauses currently unsatisfied
- proportion of clauses satisfied by exactly 1 variable
- proportion of clauses satisfied by exactly 2 variable
- proportion of variables that would break a clause if flipped
- *proportion of the variable set identical to their initial 'naive' setting*

Experiments on 'rational choice' can be found in Subsection 5.2.

5. EMPIRICAL EVALUATION

5.1. **Experimenting with Stagenis.** A test-program was implemented, which was able to solve satisfiability problems with different heuristic algorithms. These algorithms include *i)* STAGE using WalkSat as local search; *ii)* GAS as described in Section 3; *iii)* a special version of genetic algorithm, called GAwRR, using random restarts as a special mutation operator; *iv)* and for the sake of comparison the pure WalkSat algorithm with multiple random restarts (see the Appendix for the pseudo-code of these algorithms).

These algorithms have lots of inner parameters that can influence their performance drastically. It was not intended to fine-tune them for the actual problem (which is, in fact, another search problem), so most of these parameters were set to a default value and they remained unchanged in our tests.

One exception is the *cutoff* parameter of the WalkSat algorithm, where we tried two versions. Since all of these algorithms use WalkSat as their inner local search heuristic, it is an input parameter for the whole program. The first version is called *longLS,* where *cutoff* is set to $1/number\_of\_clauses$, while in *shortLS* it is set to $\min(0.5, 100/number\_of\_clauses)$. Some of the tests were addressed to decide whether *shortLS* or *longLS* is better for our benchmark problems.

The remaining parameters were configured as follows:

- The *delta* parameter of WalkSat was set to zero, so that WalkSat became a simple LS strategy unable to leave local optima. The *patience* parameter was set to 100, the *noise* was set to 0.25.
- In the GAwRR and GAS algorithms, the *recombination rate* was set to 35% (*i.e.* 70% of the new population is created using recombination), the *stagenis rate* was set to 5%; the *saved best part* of the population was hence 25%. Another idea was to rely more on the FAPP, so — in a separate experiment — we increased the *stagenis rate* to 70% and completely omitted recombination. This version was called GAwStagenis-no-recomb. To illustrate the power of these algorithms we also ran plainGA (*stagenis rate*=0% and without LS).
- STAGE used a quadratic FAPP.

To be able to compare the speed and effectiveness of these algorithms, we introduced machine-independent step-parameters:

#Evals: This counter is the number of evaluated states counted separately both for LS and for FLS.

#LS: This counter is the number of LS runs, counted separately both for the LS and for FLS.

#Gens: This counter is the number of generations for GA type algorithms, or the number of cycles for STAGE, respectively.

The termination criteria of the program can be explicitly set by means of these counters. In most cases, $10^7 \ldots 10^8$ state evaluations (*#Evals*) were allowed.

To sum up: we ran the following algorithms: STAGE, GAS, GAwRR, plainGA, GAS-no-recomb and WalkSat. It would be impossible to compress the results of all the six algorithms in one table or figure, so in most cases the relevant results are shown to increase visibility.

In the case of the harder problems none of the algorithms was able to find an exact solution. In turn, it is senseless to define the running time of these algorithms; they were stopped after a predefined number of steps. However, we note that it takes them approximately a day to make 10^8 steps (*i.e.* evaluated states) on the largest problems.

For the tests, the satisfiability benchmarks from the DIMACS archive [DIM, 1992] were used. Only satisfiable instances were considered, because in this case the global optimum is known in advance. In the case of unsatisfiable problems it is hard to value the found best solution, since it should be compared to the real optimum, which is unknown.

The results of the first mass tests can be seen in Table 2, which is located at the end of the paper, due to its size. These tests consist of both easy (*ii16d2, ii32a1, ii32c3, ii8a4, ii8b3*) and harder (*f2000, par16*, *par32*) problems to detect the limits of each algorithm. Note that our goal is to develop a parallel algorithm, whereas test results reflect performance on serial machines. It can be seen that the simple WalkSat is very effective on smaller problems, although it can fail on both types. It can be stated that WalkSat is a good heuristic for solving satisfiability problems, so it is worth using it in the more complex algorithms as LS heuristic. From now on we focus on problems, which are harder for WalkSat, and investigate how the other algorithms perform.

TABLE 1. Summary of the results of STAGE, GAS and plainGA on problem instances `par16-1, par16-2`

	STAGE – shortLS		GAS – shortLS	
Problem	BSF	#Evals	BSF	#Evals
`par16-1`	26	$1 \cdot 10^7$	15	$6 \cdot 10^6$
`par16-2`	27	$1 \cdot 10^7$	48	$1 \cdot 10^7$

	STAGE – longLS		GAS – longLS	
Problem	BSF	#Evals	BSF	#Evals
`par16-1`	6	$3 \cdot 10^7$	11	$3 \cdot 10^7$
`par16-2`	5	$3 \cdot 10^7$	8	$3 \cdot 10^7$

	plain GA		
Problem	BSF	#Evals	#Gens
`par16-1`	298	$1 \cdot 10^6$	6097
`par16-2`	288	$1 \cdot 10^6$	6052

Legend: **BSF** = *Obj* of best state found, **#Evals** = number of evaluated states, **#gens** = number of generations

It was not unequivocal whether shortLS or longLS is better; it seems to depend on the current problem and on the current run. Finding the optimal length of the LS runs for each problem was beyond the scope of our studies.

In most cases GAS outperforms GAwRR after some generations, showing that GA can effectively use the information gained from the FAPP. It can be seen that STAGE and GAS produce the best results according to our expectations. A possible explanation why GAS is not *better* than STAGE could be, that these problems may have a recognizable structure for the FAPP and STAGE uses the FAPP more intensively than GAS during the same number of evaluated states. To investigate it further, in the next round of the test cases we thoroughly examined STAGE, GAS, and PlainGA on two relatively hard problems *(par16-1* and *par16-2).*

The best objectives found by each algorithm are summarized in Table 1. The results of plainGA were at least an order of magnitude worse than those of the other algorithms indicating that this problem family is not fortunate for GA in this state encoding and both LS and Stagenis can considerably improve it.

To better understand the behavior of STAGE and GAS, consider Figs. 3 and 4 where the averaged best-objective-so-far values and the best and worst case runs can be seen. The vertical lines indicate the first 5 generation changes to picture the length of a generation (which is approximately constant during the runs, although it seems to shorten because of the logarithmic-scale diagram). The numbers in parenthesis mean the number of averaged runs. At the beginning GAS starts better, especially in the shortLS case, but at the end STAGE achieves slightly better results in the averaged values. An advantage of GAS is that it performs more reliably, *i.e.* the deviation between best and worst runs is significantly smaller than in the case of STAGE, especially around $10^4 \ldots 10^5$ evaluations.

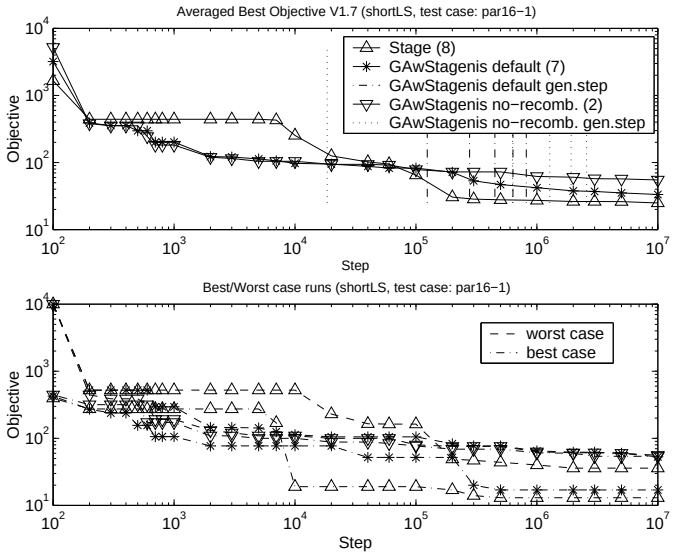


FIGURE 3. Comparison of average performances of STAGE, GAS and GAS-no-recomb on `par16-1`. The birth time of new generations are depicted by vertical lines (`gen.step`) on the upper panel. Lines becomes frequent at he right hand side of the figure, these frequent lines are not shown.

The surprisingly good performance of STAGE led us to try GAS-no-recomb in order to use the power of the FAPP more effectively. Unfortunately the results do not seem to confirm this idea. It can be seen that GAS-no-recomb is worse than GAS, hence it may not be possible to raise the performance of GAS by simply increasing the Stagenis rate.

The advantage of GAS compared to STAGE is that it is in parallel form, whereas STAGE is inherently sequential. That is, although STAGE and GAS perform similarly, the results of GAS could also be reached n times faster, if a sufficient number of processors is available.

5.2. **Experiments on 'rational choice'.** The experiments were conducted on different problem families. Examples are from CNF categories with prescribed number of clauses (x) and prescribed number of variables (y) downloaded from the web from site ftp://dimacs.rutgers.edu/pub/challenge/satisfiability/benchmarks/cnf/.

The files of the first type of family of problems on the web are specified by the notation aim$-x-y-z-yes1$ where z denotes the number of the clause within category, $yes1$ denotes that the clause is satisfiable. 4 examples were tested in each (x, y) category. Notation used in the figures: a: aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), fourth digit denotes the instance number of the database.

The other type of problem family is on parity checking. Here: p stands for 'par8', 8 means 8 bit problem, presence of 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database.

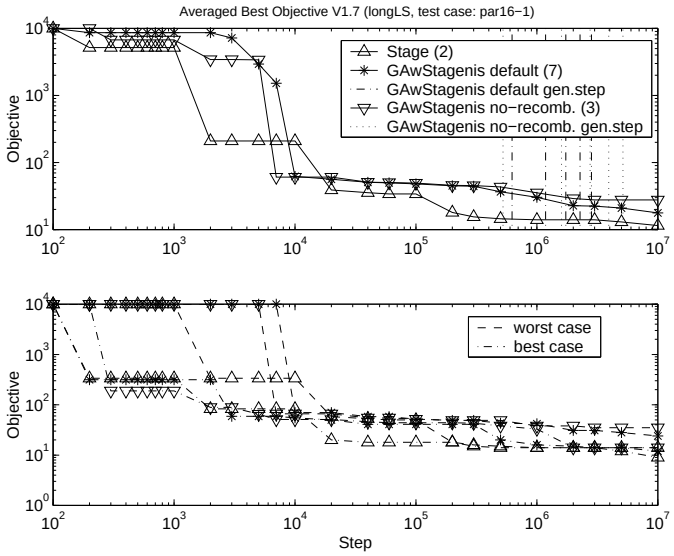


FIGURE 4. Comparison of best case and worst case performances of STAGE, GAS and GAS-no-recomb on `par16-1`. The birth time of new generations are depicted by vertical lines (`gen.step`) on the upper panel. Lines becomes frequent at he right hand side of the figure, these frequent lines are not shown.

5.3. **Experiments with threshold 12.**

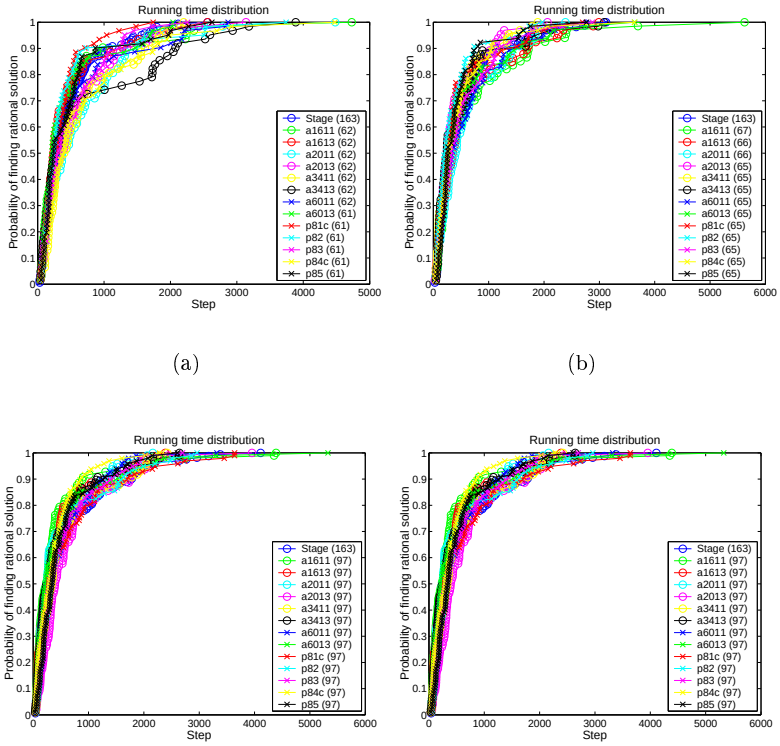


FIGURE 5. **Different FAPPs tried on problem aim–200–1–6–yes1 with threshold = 12**
(a)-(d): Four different problems, aim–200–1–6–yes1–1 — aim–200–1–6–yes1–4. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit = y, second digit = z, third digit = 1 (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database.

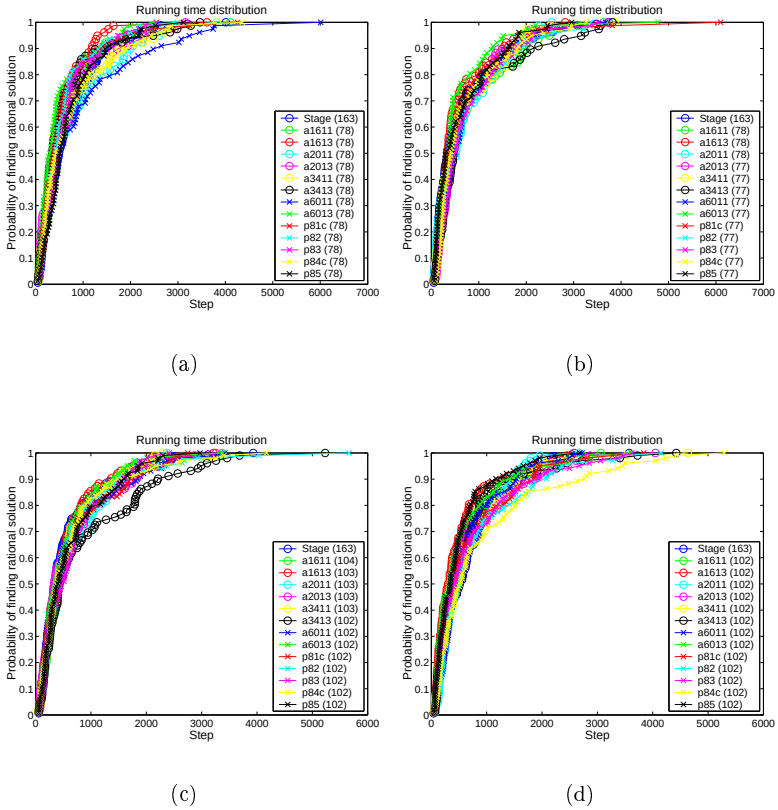


FIGURE 6. **Different FAPPs tried on problem aim–200–2–0–yes1 with threshold** $= 12$
(a)-(d): Four different problems, aim–200–2–0–yes1–1 — aim–200–2–0–yes1–4. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database.

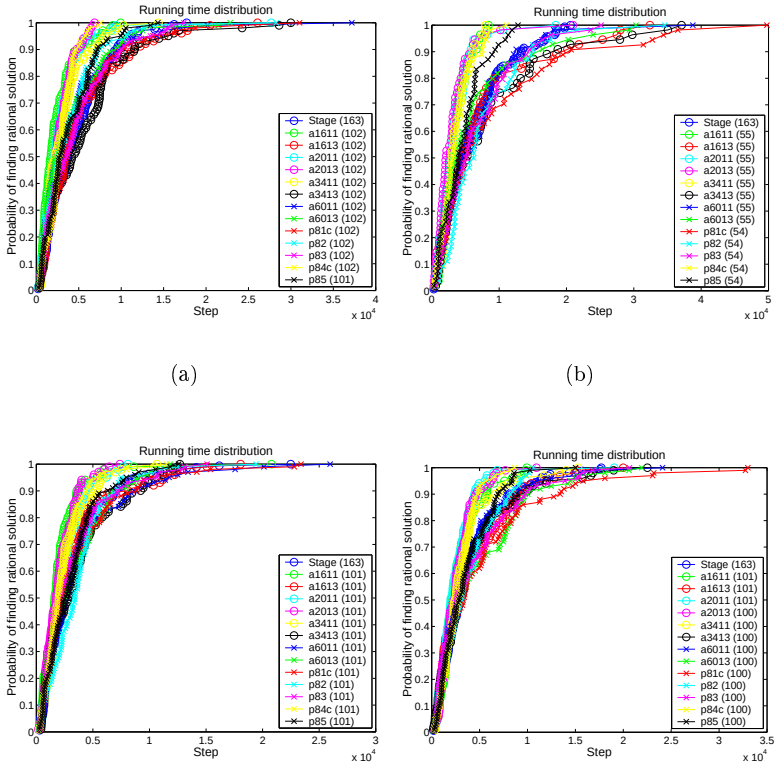
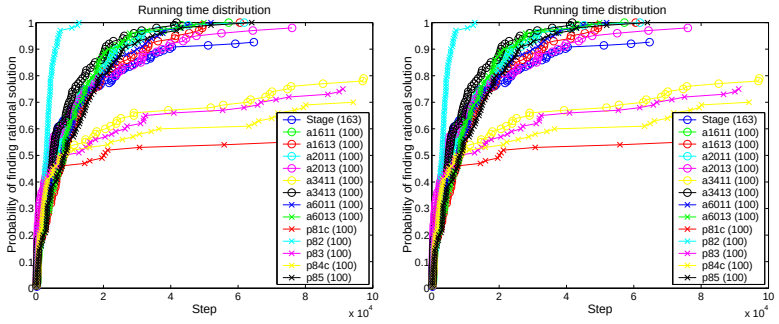
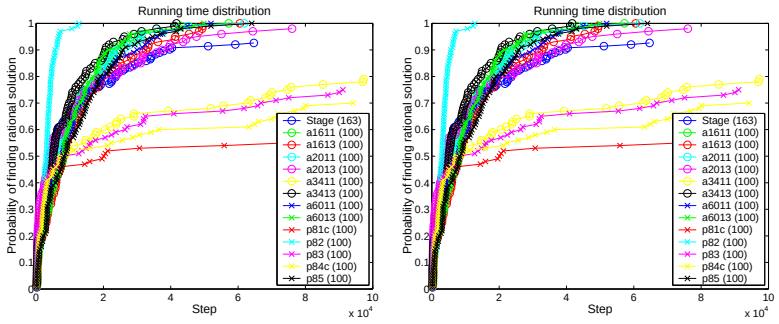


FIGURE 7. **Different FAPPs tried on problem aim–200–3–4–yes1" with threshold** = 12
(a)-(d): Four different problems, aim–200–3–4–yes1–1 — aim–200–3–4–yes1–4. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit = y, second digit = z, third digit = 1 (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database.



(a) (b)



(c) (d)

FIGURE 8. **Different FAPPs tried on problem aim–200–6–0–yes1 with threshold** $= 12$
(a)-(d): Four different problems, aim–200–6–0–yes1–1 — aim–200–6–0–yes1–4. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database.

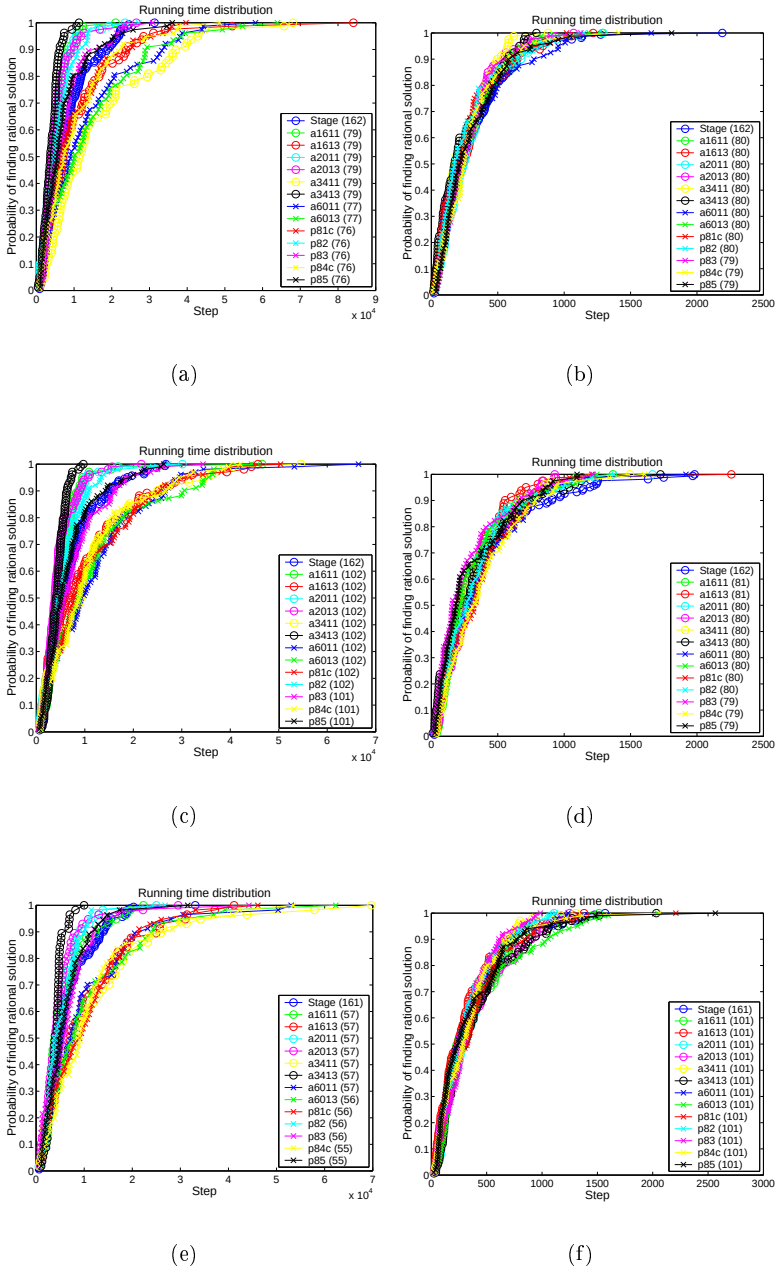


FIGURE 9. **Different FAPPs tried on problems par8 and par8c with threshold** $= 12$
(a)-(f): Six different problems, par8–1 —– par8–3. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database.

5.4. **Experiments with threshold 7.**

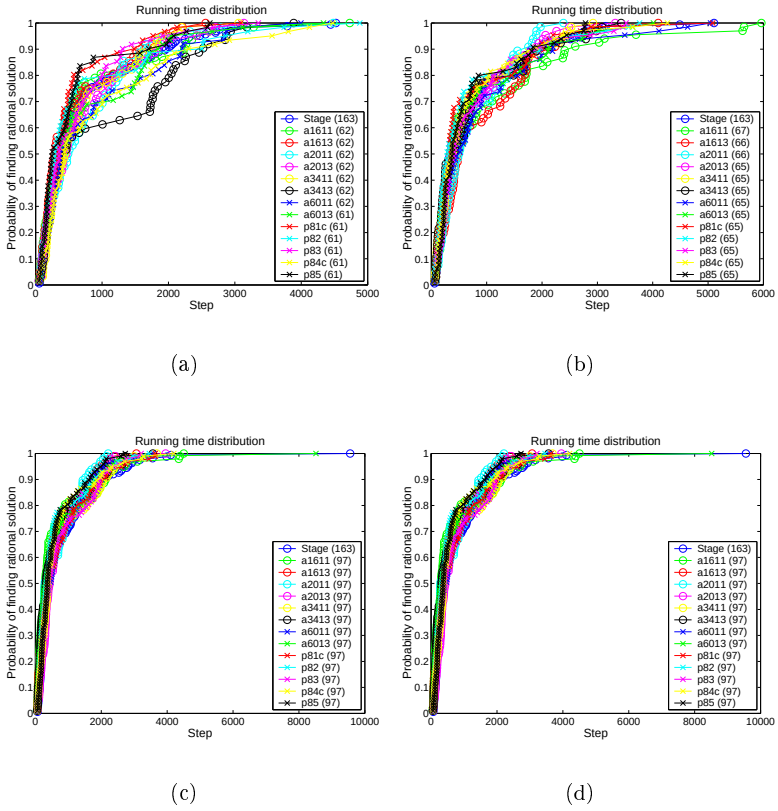


FIGURE 10. **Different FAPPs tried on problem aim–200–1–6–yes1 with threshold** $= 7$
(a)-(d): Four different problems, aim–200–1–6–yes1–1 — aim–200–1–6–yes1–4. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database.

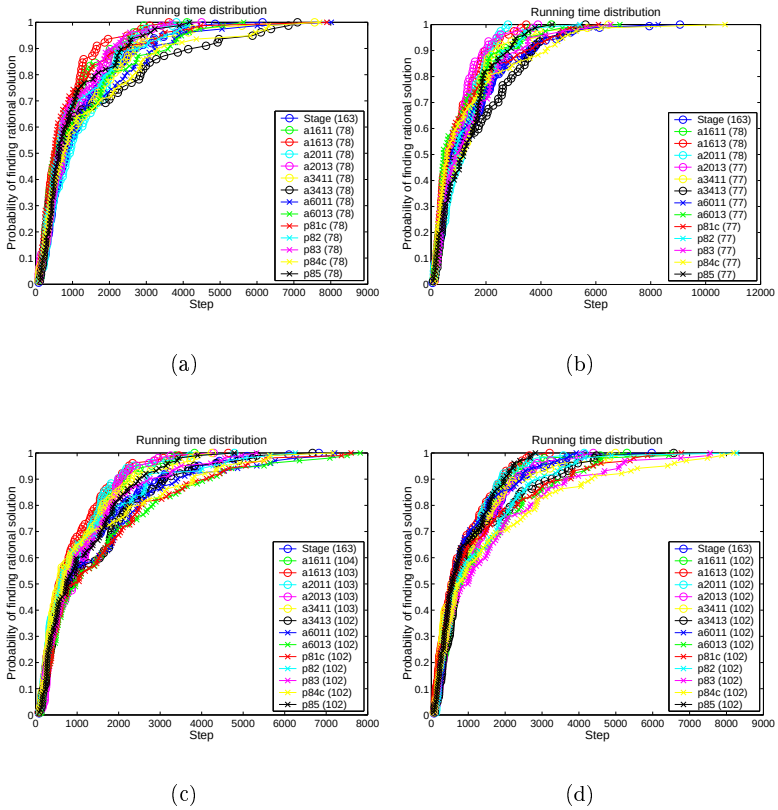


FIGURE 11. **Different FAPPs tried on problem aim–200–2–0–yes1 with threshold** $= 7$
(a)-(d): Four different problems, aim–200–2–0–yes1–1 — aim–200–2–0–yes1–4. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database.

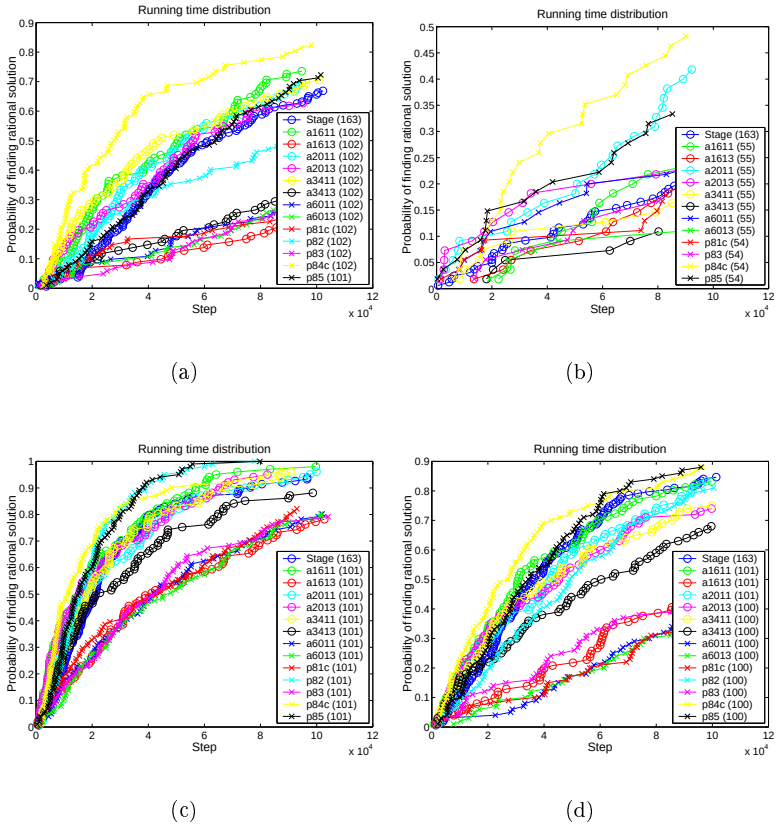
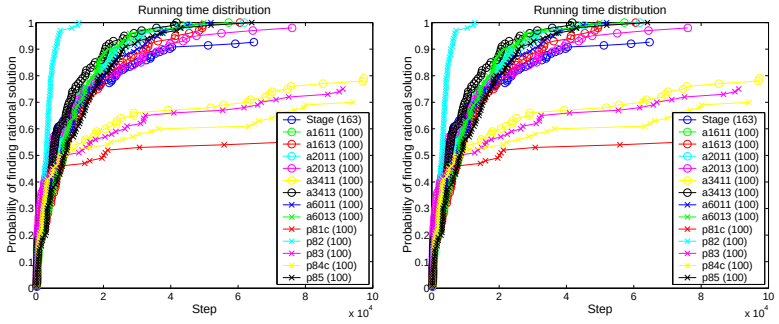
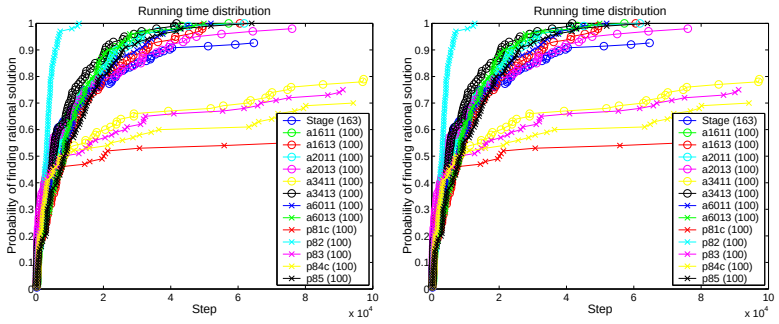


FIGURE 12. **Different FAPPs tried on problem aim–200–3–4–yes1 with threshold = 7**
(a)-(d): Four different problems, aim–200–3–4–yes1–1 — aim–200–3–4–yes1–4. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit = y, second digit = z, third digit = 1 (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database.



(a) (b)



(c) (d)

FIGURE 13. **Different FAPPs tried on problem aim–200–6–0–yes1 with threshold** $= 7$
(a)-(d): Four different problems, aim–200–6–0–yes1–1 — aim–200–6–0–yes1–4. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database.

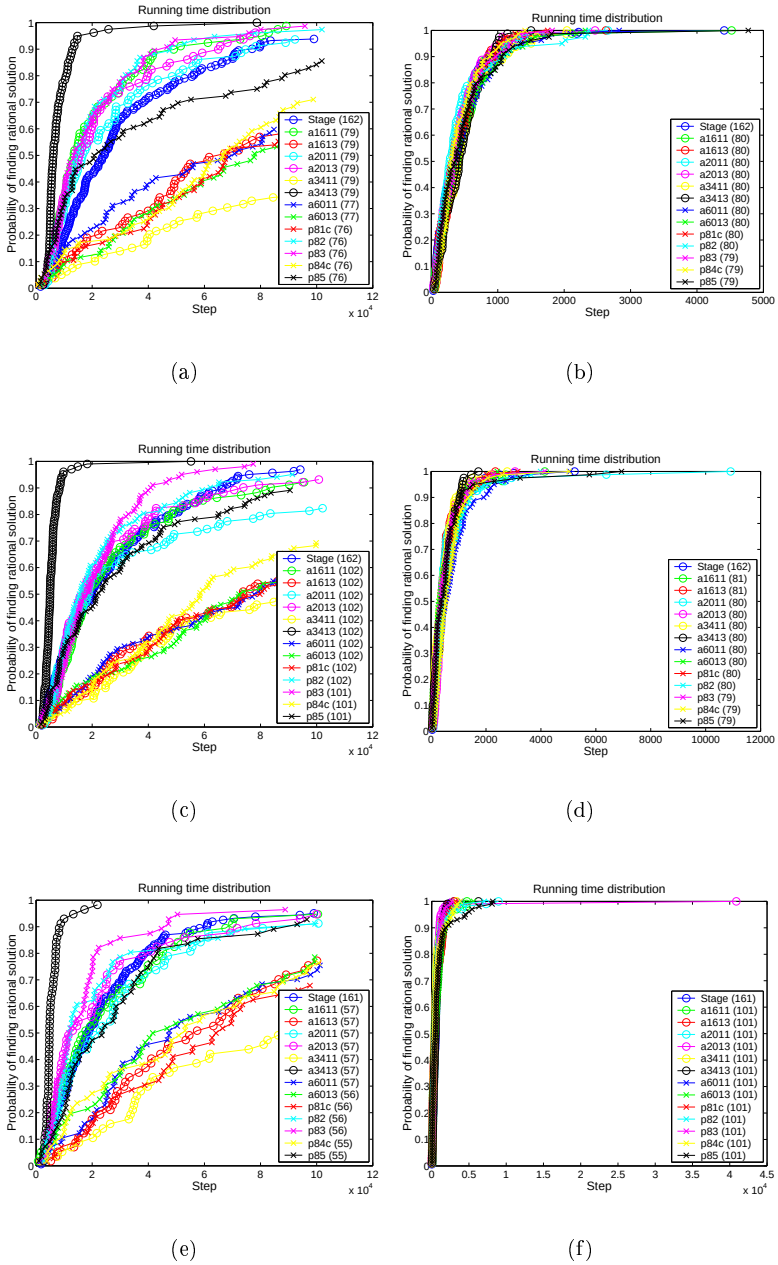


FIGURE 14. **Different FAPPs tried on problems par8 and par8c with threshold** $= 7$
(a)-(f): Six different problems, par8–1 — par8–3. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database.

5.5. **Experiments with threshold 4.**

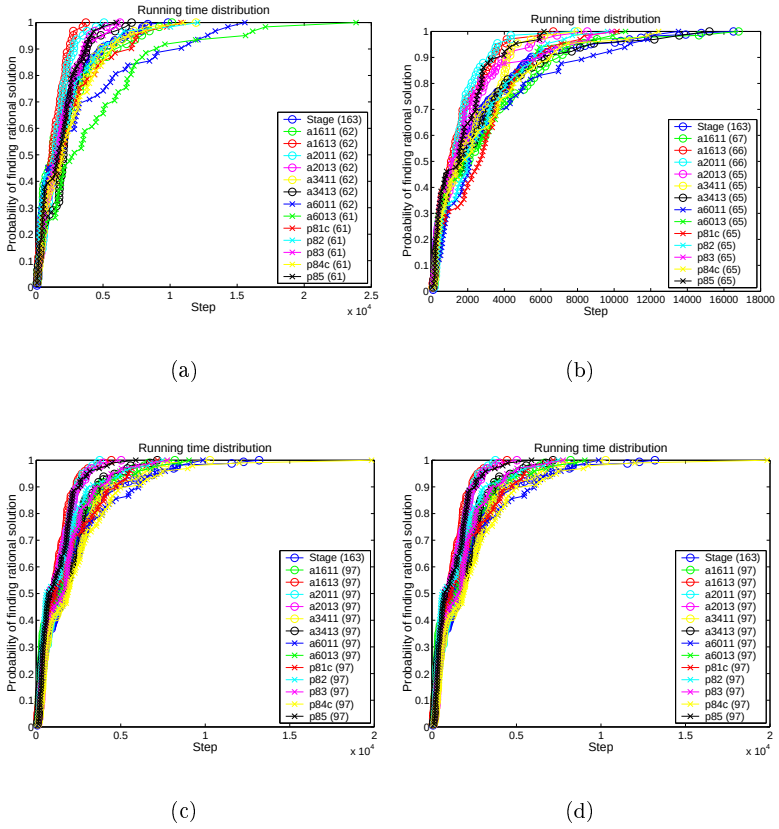


FIGURE 15. **Different FAPPs tried on problem aim–200–1–6–yes1 with threshold** = 4
(a)-(d): Four different problems, aim–200–1–6–yes1–1 — aim–200–1–6–yes1–4. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit = y, second digit = z, third digit = 1 (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database.

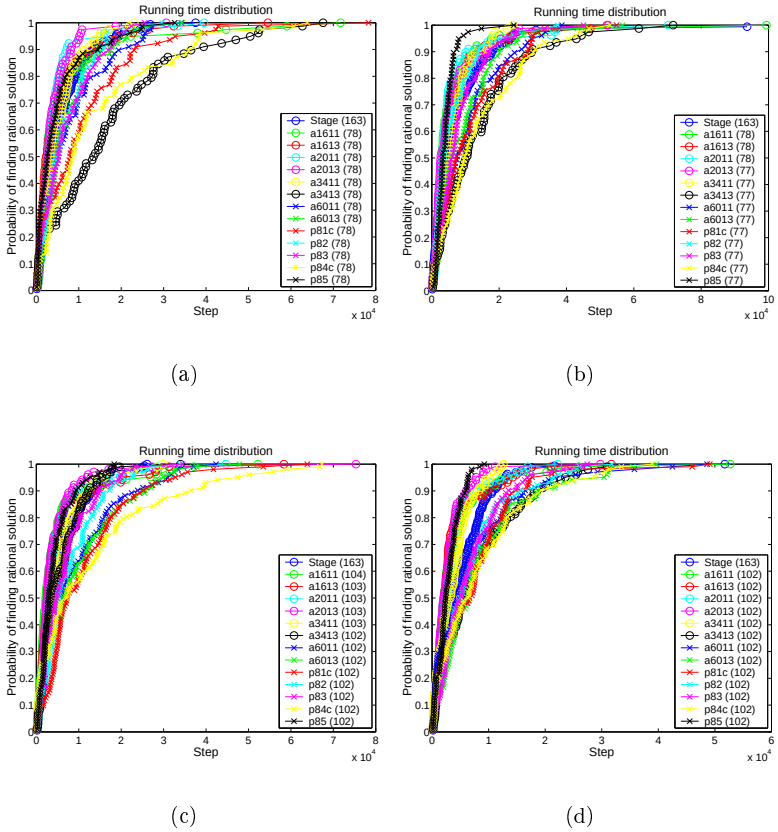


FIGURE 16. **Different FAPPs tried on problem aim–200–2–0–yes1 with threshold** $= 4$
(a)-(d): Four different problems, aim–200–2–0–yes1–1 — aim–200–2–0–yes1–4. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database.

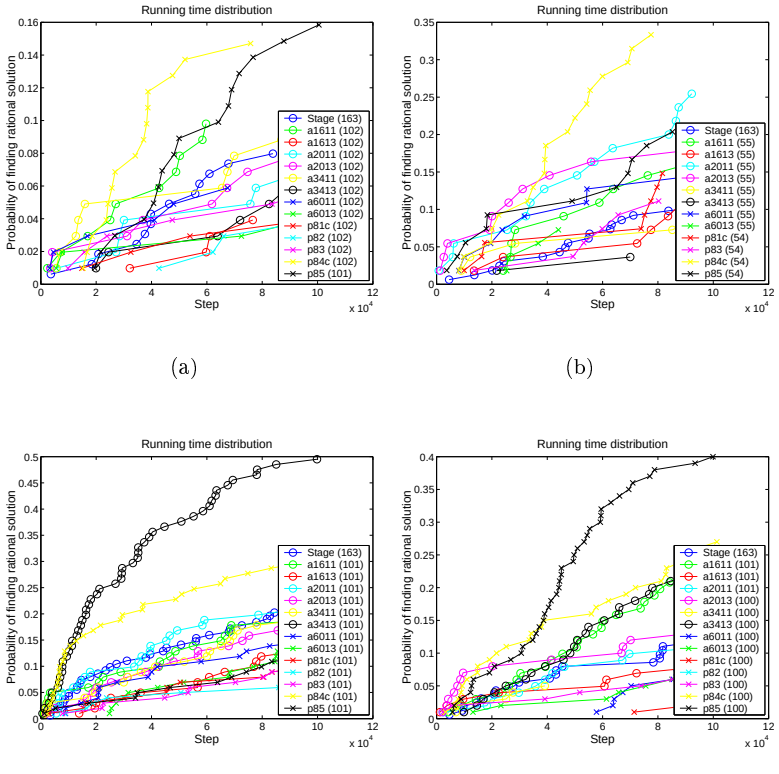


FIGURE 17. **Different FAPPs tried on problem aim–200–3–4–yes1 with threshold** $= 4$
(a)-(d): Four different problems, aim–200–3–4–yes1–1 — aim–200–3–4–yes1–4. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database.

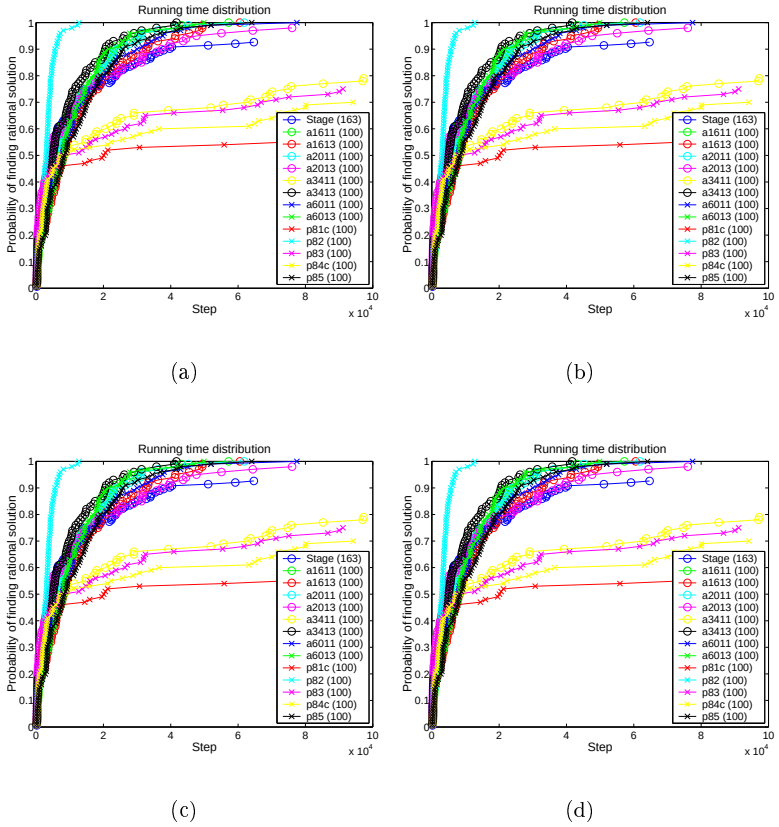


FIGURE 18. **Different FAPPs tried on problem aim–200–6–0–yes1 with threshold** $= 4$
(a)-(d): Four different problems, aim–200–6–0–yes1–1 — aim–200–6–0–yes1–4. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database.

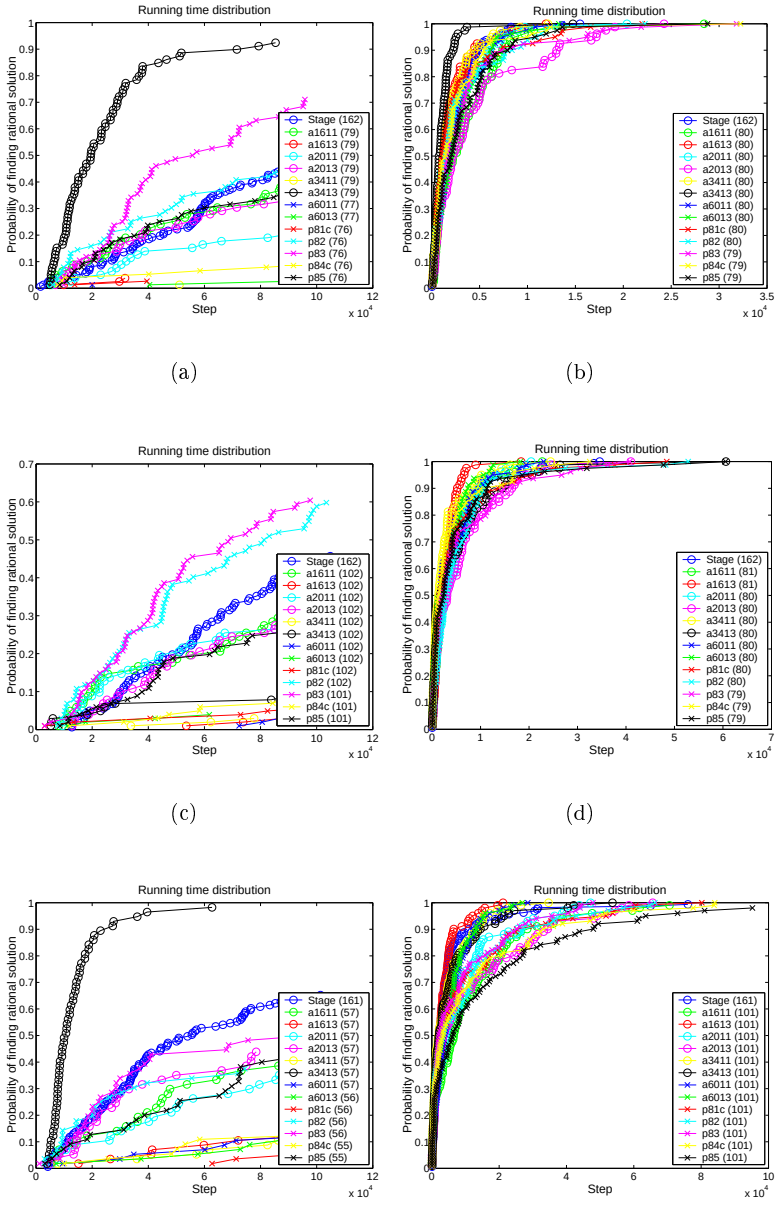
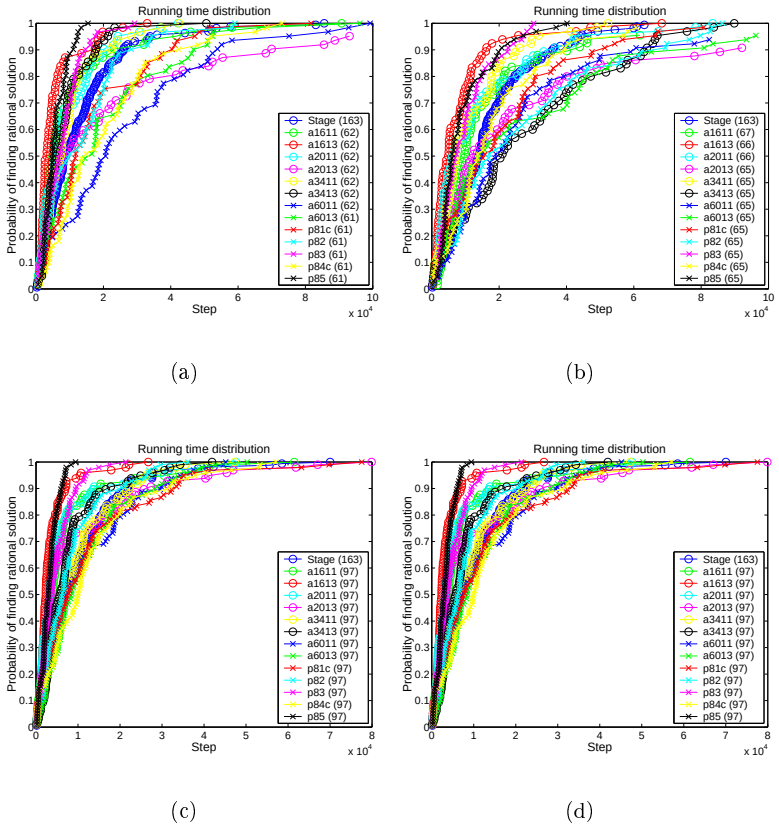


FIGURE 19. **Different FAPPs tried on problems par8 and par8c with threshold** $= 4$
(a)-(f): Six different problems, par8–1 — par8–3. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database.

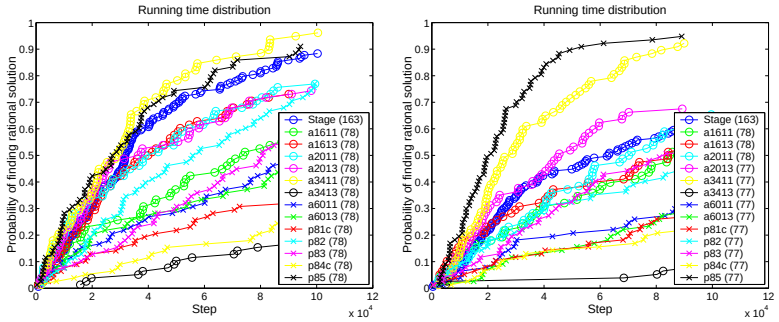
5.6. **Experiments with threshold 2.**



(a) (b)

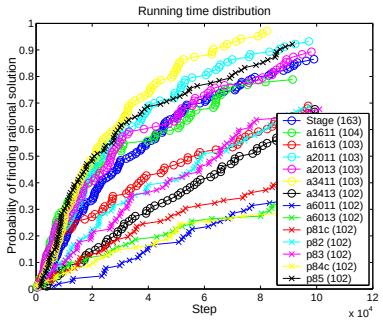
(c) (d)

FIGURE 20. **Different FAPPs tried on problem aim–200–1–6–yes1 with threshold** $= 2$
(a)-(d): Four different problems, aim–200–1–6–yes1–1 — aim–200–1–6–yes1–4. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database.

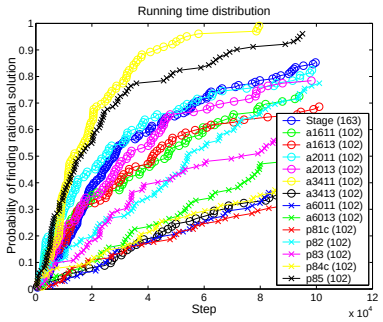


(a)

(b)



(c)



(d)

FIGURE 21. **Different FAPPs tried on problem aim–200–2–0–yes1" with threshold** $= 2$
(a)-(d): Four different problems, aim–200–2–0–yes1–1 – aim–200–2–0–yes1–4. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database.

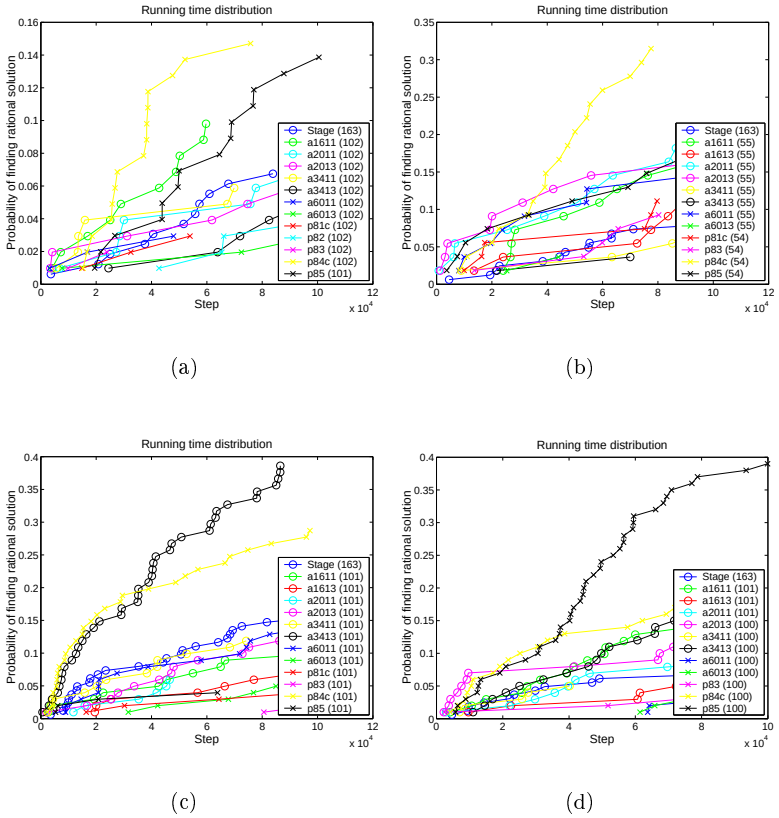
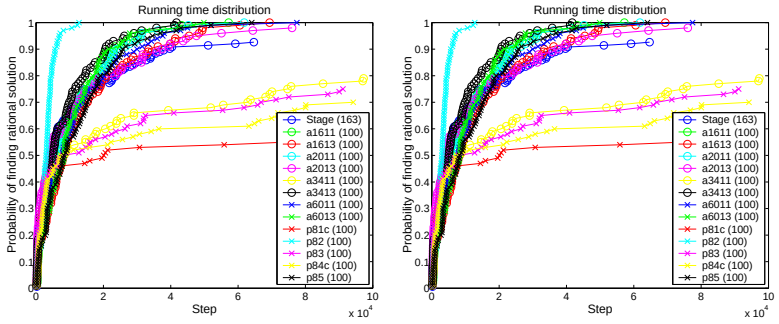
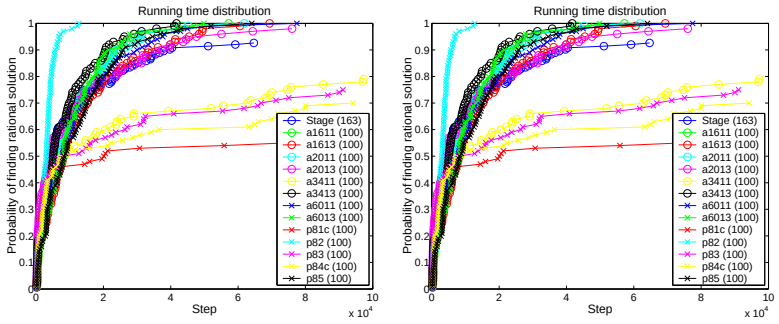


FIGURE 22. **Different FAPPs tried on problem aim–200–3–4–yes1 with threshold** $= 2$
(a)-(d): Four different problems, aim–200–3–4–yes1–1 — aim–200–3–4–yes1–4. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database.



(a) (b)



(c) (d)

FIGURE 23. **Different FAPPs tried on problem aim–200–6–0–yes1 with threshold** $= 2$
(a)-(d): Four different problems, aim–200–6–0–yes1–1 — aim–200–6–0–yes1–4. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database.

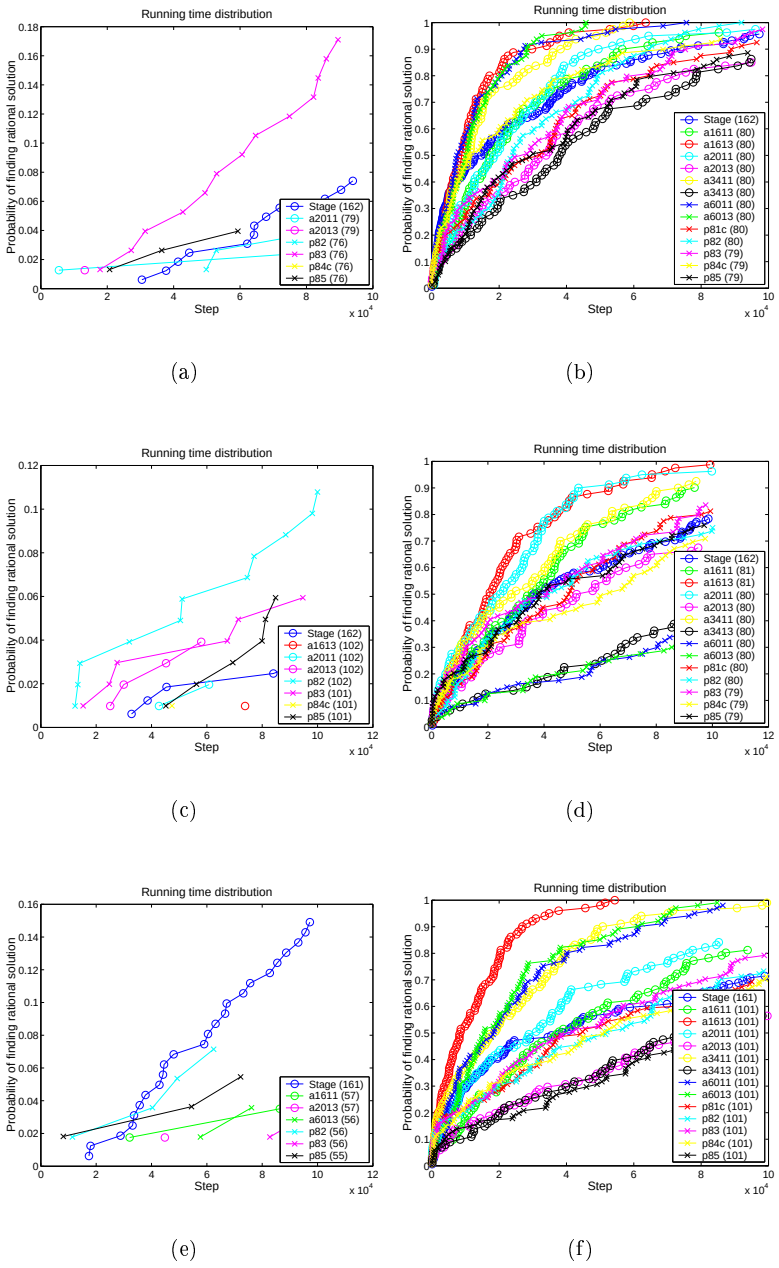


FIGURE 24. **Different FAPPs tried on problems par8 and par8c with threshold** $= 2$
(a)-(f): Six different problems, par8-1 — par8–3. Legend of the figure refers shows the codes the corresponding FAPPs. (1) circles 'a': aim $x = 200$, first digit $= y$, second digit $= z$, third digit $= 1$ (yes), last digit denotes the instance number of the database. (2) crosses 'p' stands for 'par8', 'c' or the lack of 'c' refers to the classes of the database, last digit denotes the instance number of the database.

6. Discussion

In this section we shall relate our work to evolutionary search algorithms (ESA) and to artificial neural networks. ESA can be traced back to 1953, when the Metropolis Monte Carlo algorithm was published N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller. This work considers the model (our a priori knowledge about the problem) and the search algorithm, which aims to find a global minima given our knowledge. This search algorithm is a special form of hill-climbing, called simulated annealing (SA), which makes use of *neighborhoods* and *selection*. Further algorithms were developed (interestingly, sometimes quite independently), which improved on the original ideas. In 1975 Holland published a book, in which he introduced genetic algorithms [Holland, 1975], Rechenberg developed evolutionary strategies [Rechenberg, 1973]. Ever since, a large body of research has been devoted to these ideas. For an recent review, see [Sharpe, 2000], and the web [cit,]. Interestingly, it seems to us that Monte Carlo sampling methods are not considered as part of this field.

Another widely used optimization technique makes use of a special form of function approximators (FAPPs), and as such it has its origin at the very beginning of mathematics. This particular area of FAPPs, artificial neural networks (ANNs) feature local learning rules and has started earlier than ESA [McCulloch and Pitts, 1943]. An excellent introductory book to this algorithm family has been published recently [Haykin, 1999].

Below, we review a families of algorithms, which intend to unify these two distinct routes. One specific route deals with the *selection* (this is an ESA term) of *weights* (this is an ANN term). This direction of algorithm development is called evolving artificial neural networks (EANNs). For a recent review on EANNs, see [Yao, 1999]).

The route we follow is seemingly different. In our approach the cost function is approximated and the selective algorithm is used to stabilize multiple, possibly globally optimal, solutions[3]. The FAPP of Stage [Boyan and Moore, 1998, Boyan and Moore, 2000] represents the low-frequency regularities of the cost function. High frequencies tend to be smeared by construction: to each search trajectory the best value of that trajectory is rendered before approximating the cost function.

One might ask how this approach works on NP-complete problems. It has been known (see e.g., [Gomes et al., 1998, Gomes et al., 2000]) that NP-complete searches often have heavy tails. In turn, it has been suggested by the Cornell group [Gomes et al., 1998, Gomes et al., 2000] that randomized search (random restarts) is appropriate for this family. We relate this observation to the 'No Free Lunch' (NFL) Theorem of optimization [Wolpert and Macready, 1997] and read the NFL Theorem backwards: If we have an upper limit for the duration of the search then it is better to *make use* of the NFL Theorem and not spend too much time with long searches.

Stage has an interesting property from this point of view. To see it, consider the experiments depicted in Fig. 25. The figures shows 30 independent computations started using Stage and 25 independent computations for GA with Stagenis (GAS). Slow improvements can be seen in the figure depicting Stage beyond iteration 4×10^4. For the case of GAS in each run we used a population of 100 individuals. In

[3]For other types of approximations to search problems, see [Hochbaum, 1995].

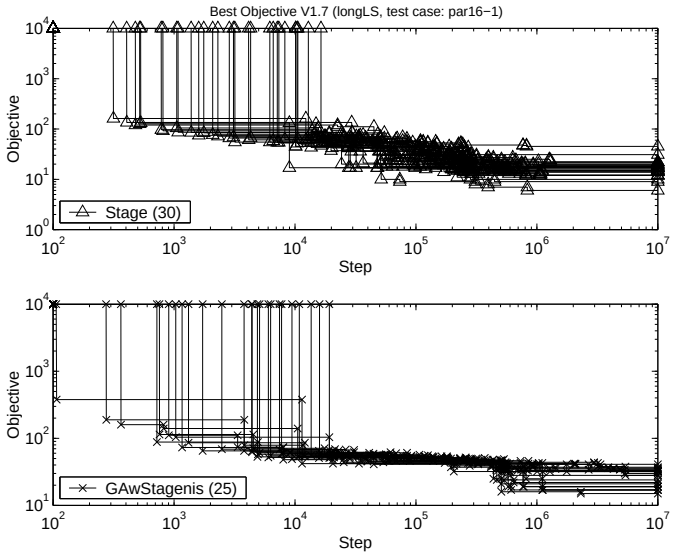


FIGURE 25. Random restarts and the effect of smart restarts on `par16-1`

turn, for GAS, Stagenis did not make any effect on the search until 4×10^5 evaluation steps. Up to that point there is little difference between the different WalkSats used for fitness search, not much improvement can be seen and the distribution becomes narrow. Up to this point the case corresponds to the strategy suggested by the Cornell group [Gomes et al., 1998]. One can see also that sweeping of the length of random restarts might help, indeed. The first sudden improvement occurs as soon as circa 200 evaluation steps in one of the runs. The last one occurs at around 2×10^4. Beyond that there is a continuous slow improvement. This improvement could be a function of parameter delta of WalkSat, and could be tuned. The point, however, is that around 1×10^5, when the initialization of the first population is just becomes to be finished, Stagenis starts to make an effect[4]. The effect is strong, there is a marked improvement in performance here. Slow improvement can be seen from generation to generation either by the effect of GA or the effect of STAGE or both (improvements are achieved at the cross signs).

Let us compare the performance of STAGE with that of GAS. The best value of GAS is at about 15, whereas STAGE was able to reach 5 in one of the runs. This is the case when computations are made on a serial computer. Also, this comparison is not perfect, because GAS improves barely between 2×10^4 and 5×10^5, and WalkSat with true local search is suboptimal for GAS: all WalkSats could be stopped at 2×10^4. Taking this 1.5 order of magnitude into account and considering 100 parallel computers the results of STAGE 3×10^3 need to be compared to the final results of GAS. This is a considerable improvement gained in this the parallel form of STAGE.

Let us consider STAGE from the point of view of the NFL theorem [Wolpert and Macready, 1997]. According to this theorem, if we know nothing about the cost function then no algorithm is better than the random search. STAGE is not exception to the NFL

[4]At this point GA can not make an effect yet, because the initial generation is being calculated.

theorem: if the conditions of the NFL theorem hold then there is non reason to choose STAGE or Stagenis either. The NFL theorem does not hold, for example, under the following related conditions

- if our universe is of low-complexity [Schmidhuber, 1994],
- if it is more likely to encounter simpler problems than complicated ones [Cover and Thomas, 1991], and
- if our problem is entailed by the Solomonoff-Levin a priori distribution [Li and Vitányi, 1997].

Under these conditions STAGE has the following properties. STAGE 'looks for' global structure given the neighboring relations. In turn, STAGE is based on the assumption that local minima may uncover the global structure. STAGE is an *off-line method*: individuals created by STAGE in Stagenis can be computed without interacting with the system, i.e., without measuring the fitness. That is, STAGE and other parameterized function approximators can be relatively inexpensive provided that optimization is not limited by computer time but by the time that we may interact with the system. One may say that STAGE is harmless if the NFL theorem holds. However, STAGE may cause harm if the NFL theorem does not hold, because STAGE can bias our search. On the other hand, if the NFL theorem does not hold and if the cost (time) of interaction is relatively large (long) compared to the cost (time) of parametric optimizations then STAGE – or other FAPP methods – may provide considerable improvements over random search. The danger of biased search can be avoided by keeping a constant proportion of random restarts.

In fact, one can read the NFL theorem backwards: If we have encoded all of our knowledge into the search problem and have no further information about it, if we have surplus off-line computer power, and if we keep our search unbiased then the NFL theorem warrants that our efforts to find regularities and to make use of those regularities *during search* will cause no harm. STAGE and GAS make use of such *parameterized* off-line evaluations. Off-line evaluations are closely related to off-policy methods, discussed extensively by Sutton and Barto [Sutton and Barto, 1998] within the framework of value estimation. An off-policy method makes use of off-line computation but does not influence the search procedure. Further theoretical work is needed here. The question is how to estimate the value of off-policies if the condition of the NFL Theorem (about the infinity of the problem space) may not hold. One would like to know how to create and how to select among off-policies and when to switch on- and off-policies.

We showed in Sec. 2 that the FAPP of Stage is sometimes unstable. This problem was avoided by saving the 'best so far' solutions to the search problem. In the case of GAS, this problem is taken into account by the population management. There is another advantage of the GA part of GAS. GA has the potential to break the problem into components given its implicit parallelism. In turn, GA has the potential to discover 'chunks' (subgoals), a favorable step of optimization [Solomonoff, 1986]. The algorithm could be improved by extending this implicit chunking functions via off-line search for chunks (see, e.g., [Schmidhuber, 1994]). For low-complexity problems special care is needed to avoid biased search, e.g., the degradation of GA population. Random restarts and/or mutations can be used for this purpose.

It is easy to see that GA could also be introduced at the level of FAPPs. It is noteworthy that if the FAPPs themselves underwent evolutionary computing then

the best FAPPs would be saved and the best regions, (which could be narrow) would be more exposed to evolutionary explorations. In turn, FAPPs can be seen as encoding technique for GAs. This way one arrives to EANNs.

6.1. All the FAPPs trained on other examples perform well on other problems. Pre-trained FAPPs can be used to find rational solutions quickly. Training of the FAPP is not necessary but could be accomplished. Care is needed, however.

One should note that in some cases – in fact, in a large number of cases – FAPPs of one problem family perform better on another problem family.

We have tried to learn the best FAPPs by using a winner-takes-all algorithm. This means, that we have started all available FAPPs on a single problem and after several thousand steps the best FAPP was selected and was trained on that problem. It was expected that FAPPs will partition the problem space and we shall gain the best FAPPs for each subproblem. This experiment did not succeed.

The two facts, i.e., that FAPPs are not the best approximators on their on problem and the failure of this WTA experiment pinpoints to the possibility that the biased search of STAGE plays a role here. Tuning of STAGE can be misled by improvement using the FAPP but then the local search may start in the neighborhood of a different and possibly less favorable local minimum. This seems to be the case here.

One may solve this problem by enforcing global optimization in the FAPP space. This global optimization should not be based on tuning of the FAPPs, which would be misleading. As it appears to us, a better method is to use a fitness value for each FAPP. The fitness could be equal to the negative value of the best cost tha FAPP did achieve on the full set of problem families. On the one hand, this may require tremendous computations. On the other hand, this is an off-line method and results that become available during the different optimization procedures can be used for generating novel FAPPs. Moreover, there should be large real-time gain for individual problems, using rigid FAPPs.

Again, the genetic algorithm could be of use here. This is more so, because the feature space of the FAPPs is small (it was five in our case) and GA can be very efficient under such circumstances. Moreover, up to our best knowledge, this is the first instance when gradient methods may destroy globally optimal solutions. The unpleasant consequence of the clever combination of local search methods using smoothing technique (Boyan and Moore, 2000) is not a major drawback, it can be easily fixed by adding GA, as it has been mentioned. This point requires further studies.

Here is the main message of our models. This message is supported by both parts of our computational studies; (i) the GA for keeping a larger population of solutions beyond the best so far instance, and (ii) the development of FAPP families.

7. CONCLUSIONS

Experiences with STAGE and GA led us to combine these algorithms and to create GA with Stagenis (GAS). We expected that GAS can fuse the robustness and parallelism of GA and the smartness of STAGE. In our tests, we compared GAS with STAGE, GAwRR and a couple of other heuristics on satisfiability benchmark problems.

We made the following experiences that are in accordance with the literature:

- The plainGA performed poorly, probably due to unsuitable state encoding. The importance of state encoding is a well-known drawback of genetic algorithms (see, e.g., *e.g.* [Falkenauer, 1996]) and we have neglected this problem for the sake of the demonstration.
- Using LS to compute the fitness of the individuals can boost the performance of GA (see also [Freisleben and Merz, 1997]).
- WalkSat is very efficient on smaller problem instances.
- STAGE shows impelling performance on all of our test problems.

Our new findings are:

- STAGE performs better on the given problem domain than GA (with the given state encoding).
- Stagenis can boost the performance of GA.
- The performance of GAS is comparable to that of STAGE. In most cases GAS is somewhat slower when executed on single processor, but on some problems it was even superior to STAGE.
- We proposed a parallel algorithm, the GAS. Therefore, GAS can be speeded up if enough parallel processors are available.
- STAGE is an off-line method and may boost search for 'nothing' if enough computational power is available and if either search time or interaction time with the real system is limited.

Time-limited optimization may be desirable in cases when the clauses of the SAT problem 'are not equal', e.g., the failure to satisfying some of them may be costly but this cost could be smaller than the cost not to offer any approximate solution. Such rational choice solutions were studied and it was found that reasonable speed-up can be gained.

Finally, we note that the combined algorithm may perform well on problem families, i.e., on a group of problems, provided that a small change of the problem (in some metric of the problem space) gives rise to a small change of the solution (in a matric of the solution space). In case if such metric is not known, or in case of unrelated problems no harm will be made by the combined algorithm, provided that biased search (e.g., a recurrence) is avoided. If this is not the case, then other methods can be used to augment the optimization and to overcome the biased search of STAGE.

TABLE 2. Results of the mass testing

		STAGE		GAS		GAwRR		WalkSat	
Problem	type of LS	BSF	#Evals	BSF	#Evals	BSF	#Evals	BSF	#Evals
f2000	long LS	19	512 223	28	585 039	28	585 039	32	350 373
	short LS	45	714 975	78	540 667	64	392 234	64	725 426
ii16d2	long LS	0	247 709	32	273 974	32	273 974	70	680 705
	short LS	0	242 625	11	565567	32	486 755	32	796 073
ii32a1	long LS	0	11 046	17	30 091	17	30 091	0	84 402
	short LS	0	43464	0	47 672	0	47 672	0	131 449
ii32c3	long LS	0	1 404 703	2	1 560 916	7	1 734 502	7	2 323 008
	short LS	0	96287	1	2 691 189	7	2 571 988	7	3 368 739
ii8a4	long LS	0	1 229	0	243 828	0	243 828	0	82959
	short LS	0	43981	0	223 092	0	103 595	0	6 742
ii8b3	long LS	1	525 976	1	5310	1	5310	1	850 164
	short LS	1	1 316 227	1	1 077 128	1	700 994	1	1 297 332
par16-2	long LS	5	1 694 021	15	1 608 343	15	1 214 563	15	1 928 988
	short LS	16	1 893 826	10	1 961 045	52	1 112 929	59	2 011 803
par32-1-c	long LS	20	849 085	13	140 793	13	140 793	13	1 000 547
	short LS	13	1 187 205	36	2 289 301	51	702 845	25	1 203 244
par32-3-c	long LS	12	1 165 261	13	521 962	13	521 962	13	1 253 889
	short LS	14	1 065 066	1	2 691 189	24	685 952	24	1 209 921
xx_ii32d1	long LS	0	68 169	0	1 413 775	9	3 471 353	9	3 786 561
	short LS	0	30916	0	514 149	9	3 836 964	9	5 126 792

BSF = *Obj* of best state found, **#Evals** = number of evaluated states

Appendix

Algorithm 1 The WalkSat algorithm in pseudo-code

```
PROCEDURE doOneStep(clause)
   SET Q={states reachable by flipping one
                 variable in the clause}
   SELECT (one of) the state(s) with the best
        Obj value in Q: s_t
   IF Obj(s_t) < Obj(previous_state)
   THEN ACCEPT the step
   ELSE
      IF Obj(s_t) < (Obj(previous_state) + DELTA)
      THEN ACCEPT
           with probability (1 - NOISE) this step
           with probability NOISE a random step
      ELSE remain in the current state
      FI
   FI
END

//main loop
REPEAT
   doOneStep(random unsatisfied clause)
   IF Obj(s_t) < Obj(best_state_so_far)
   THEN best_state_so_far = s_t
   FI
UNTIL (random value < CUTOFF)
   OR (the state remained unchanged
       for PATIENCE steps)
   OR (<#Gens|#LS|#Evals> exceeds
       <MaxGens|MaxLS|MaxEvals>)
RETURN best_state_so_far, and the trajectory
```

Algorithm 2 The STAGE algorithm in pseudo-code

```
REPEAT
   RUN  LS from x_0
        //The trajectory is (x_0, x_1, ..., x_N)
   SET y to the value of the best state
        in the trajectory
   RETRAIN the FAPP with the pairs (x_i, y)
   RUN FLS starting from x_N
   SET x_0^new to the end state of the FLS
   IF x_0^new = x_0
   THEN SET x_0 to a random starting state
   ELSE x_0 = x_0^new
   FI
UNTIL <#Gens|#LS|#Evals>
      exceeds <MaxGens|MaxLS|MaxEvals>
RETURN the best state found
```

Algorithm 3 The GAwRR algorithm in pseudo-code

```
PROCEDURE LS.optimize(x)
   RETURN (the best Obj found by a LS
          starting from x
END

INITIALIZE old_population with random states
FOR EACH x_i ∈ old_population
   Fitness(x_i) = LS.optimize(x_i)
REPEAT
   Step 1: SELECT recombination_rate · population_size
              pairs (x_i, x_j) from old_population
              with selection likelihood
              proportional to Fitness(·)
           FOR EACH pair (x_i, x_j)
              SET (x'_i, x'_j) = recombinate(x_i, x_j)
              INSERT (x'_i, x'_j) into new_population
              SET Fitness(x'_i) = LS.optimize(x'_i)
              SET Fitness(x'_j) = LS.optimize(x'_j)
   Step 2: INSERT mutation_rate · population_size
                   random members in new_population
           FOR EACH new member x_i
              SET Fitness(x_i) = LS.optimize(x_i)
   Step 3: FILL_UP the rest of new_populationwith
               the best members of old_population
               according to their fitness
           FOR EACH such member
               retain the old Fitness(·)
UNTIL <#Gens|#LS|#Evals> exceeds
      <MaxGens|MaxLS|MaxEvals>
RETURN the best state ever found during LS runs.
```

Algorithm 4 The GAS algorithm in pseudo-code

```
PROCEDURE LS.optimize(x)
   RETURN the best Obj found by a LS starting from x
END

PROCEDURE FLS.optimize(x)
   RETURN the best state found by a FLS starting from x
END

INITIALIZE old_population with population_size · (1 − stagenis_rate) random x_i members
FOR EACH x_i ∈ old_population Fitness(x_i) = LS.optimize(x_i) LOAD trajectory i into FAPP
FOR i = 1 TO population_size · stagenis_rate
   SET x_i = FLS.optimize(randomstate)
   INSERT x_i in old_population
   Fitness(x_i) = LS.optimize(x_i)
   LOAD trajectory i into FAPP
REPEAT
   FOR i = 1 TO population_size· (stagenis_rate + 2 · recombination_rate)
      IF i ≡ 0 mod (⌊(population_size·(stagenis_rate+2·recombination_rate))/(population_size·stagenis_rate)⌋)
      THEN
         x_i = FLS.optimize(randomstate)
         INSERT x_i in new_population
         Fitness(x_i) = LS.optimize(x_i)
         LOAD trajectory i into FAPP
      ELSE
         SELECT (x_i, x_j) of old_population with
            selection likelihood proportional to Fitness(·)
         SET (x'_i, x'_j) = recombinate(x_i, x_j)
         INSERT (x'_i, x'_j) in new_population
         Fitness(x'_i) = LS.optimize(x'_i)
         Fitness(x'_j) = LS.optimize(x'_j)
         LOAD both trajectories into FAPP
         INCREMENT i //recomb. generates 2 new members!
         FI
      FILL_UP the rest of new_population
          with the best members of old_population
          according to their fitness
      FOR EACH such member retain the old Fitness(·)
   UNTIL <#Gens|#LS|#Evals> exceeds <MaxGens|MaxLS|MaxEvals>
RETURN the best state ever found during LS runs.
```

References

[cit,] Researchindex, the NECI scientific literature digital library.

[DIM, 1992] (1992). The DIMACS satisfiability challenge benchmarks.

[Baum et al., 1995] Baum, E. B., Boneh, D., and Garrett, C. (1995). Where genetic algorithms excel. Technical report, NEC Research Institute.

[Boyan, 1998] Boyan, J. A. (1998). *Learning Evaluation Functions for Global Optimization*. PhD thesis, School of Computer Science Carnagie Mellon University, Pittsburgh PA 15213.

[Boyan and Moore, 1998] Boyan, J. A. and Moore, A. W. (1998). Learning evaluation function for global optimization and boolean satisfiability. In *In Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI)*, page 14.

[Boyan and Moore, 2000] Boyan, J. A. and Moore, A. W. (2000). Learning evaluation functions to improve optimization by local search. *Journal of Machine Learning Research*, 1:77–122.

[Cover and Thomas, 1991] Cover, T. and Thomas, J. (1991). *Elements of Information Theory*. John Wiley and Sons, New York, USA.

[Dorne and Hao, 1998] Dorne, R. and Hao, J. (1998). A new genetic local search algorithm for graph coloring. In Eiben, A., Bäck, T., Schoenauer, M., and Schwefel, H., editors, *Parallel Problem Solving from Nature – PPSN V*, Lecture Notes in Computer Science 1498, pages 745–754, Berlin. Springer-Verlag.

[Falkenauer, 1996] Falkenauer, E. (1996). A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics*, 2(1):5–30.

[Freisleben and Merz, 1997] Freisleben, B. and Merz, P. (1997). Genetic local search for the TSP: New results. In *Proceedings of The IEEE International Conference on Evolutionary Computation*, pages 159–164. IEEE Press.

[Gomes et al., 1998] Gomes, C., Selman, B., and Kautz, H. (1998). Boosting combinatorial search through randomization. In *Proceedings of AAAI98*, Madison, WI.

[Gomes, 2000] Gomes, C. P. (2000). Structure, duality, and randomization — common themes in AI and OR. In *Proceedings of AAAI00*, Austin, Texas. Invited talk.

[Gomes et al., 2000] Gomes, C. P., Selman, B., Crato, N., and Kautz, H. (2000). Heavy-tailed phenomena in satifiability amd constraint satisfaction problems. *Journal of Automated Reasoning*, (24):67–100.

[Haykin, 1999] Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Prentice Hall.

[Hochbaum, 1995] Hochbaum, D. (Editor, 1995). *Approximation Algorithms for NP - Hard Problems*. PWS Publishing Co., Boston.

[Holland, 1975] Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan, USA.

[Kammeyer and Belew, 1996] Kammeyer, T. and Belew, R. K. (1996). Stochastic context-free grammar induction with a genetic algorithm using local search. In Belew, R. K. and Vose, M.,

editors, *Foundations of Genetic Algorithms IV*, University of San Diego, CA, USA. Morgan Kaufmann.

[Li and Vitányi, 1997] Li, M. and Vitányi, P. (1997). *An introduction to Kolmogorov complexity and its applications*. Springer-Verlag, New York, USA, 2nd edition.

[McCulloch and Pitts, 1943] McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133.

[Palotai et al., 2001] Palotai, Z., Kandár, T., Mohr, Z., Visegrády, T., Ziegler, G., Arató, P., and Lőrincz, A. (2001). Value prediction in the allocation problem of high level synthesis with IPs. *Journal on Applied Artifical Intelligence*. Accepted paper.

[Rechenberg, 1973] Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*. Fromman-Holzboog, Germany.

[Schmidhuber, 1994] Schmidhuber, J. (1994). *Proc. 12th Int. Conf. on Machine Learning*, chapter Discovering solutions with low Kolmogorov complexity and high generalization capability, pages 488–496. Morgan Kaufmann Publishers, San Mateo, CA.

[Selman et al., 1996] Selman, B., Kautz, H. A., and Cohen, B. (1996). Local search strategies for satisfiability testing. *Second DIMACS Challenge on Cliques, Coloring and Satisfiability*, 26:521–531.

[Sharpe, 2000] Sharpe, O. (2000). *Towards a Rational Methodology for Using Evolutionary Search Algorithms*. Phd thesis, University of Sussex, UK.

[Solomonoff, 1986] Solomonoff, R. (1986). *Uncertainty in Artificial Intelligence*, chapter An application of algorithmic probability to problems in artificial intelligence, pages 473–491. Noth-Holland. Eds.: L.N. Kanal and J.F. Lemmer.

[Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*. MIT Pess.

[Ulder et al., 1994] Ulder, N., Aarts, E., Bandelt, H., van Laarhoven, P., and Pesch, E. (1994). Genetic local search algorithms for the travelling salesman problem. In *First International Conference on Parallel Problem Solving from Nature PPSN*, pages 109–116.

[Wolpert and Macready, 1997] Wolpert, D. and Macready, W. (1997). No Free Lunch Theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1:67–82.

[Yao, 1999] Yao, X. (1999). Evolving artificial neural networks. *Proceedings of teh IEEE*, 87:1423–1447.

[Ziegler et al., 2001a] Ziegler, G., Mann, Z. A., Orbán, A., Palotai, Z., and Lőrincz, A. (2001a). Stage for parallel computation. Technical report. Under preparation.

[Ziegler et al., 2001b] Ziegler, G., Palotai, Z., Cinkler, T., Arató, P., and Lőrincz, A. (2001b). Value prediction in engineering applications. In Monostori, L., Váncza, J., and Ali, M., editors, *Engineering of Intelligent Systems. Proceedings of AIE/IEA 2001, Budapest Hungary*, volume 2070 of *LNAI*, pages 25–34. Springer. ISBN 3-540-42219-6.

1117 Budapest, Pázmány Péter sétány 1/C.

E-mail address: alorincz@axelero.hu, titkarsag@njszt.hu

REPORT: MODELS OF DISTRIBUTED COMPUTATION

REPORT EOARD-NIPG-ELU-07-JUNE-2002
CONTRACT: EOARD F61775-01-WE023

ADMINISTRATIVE CONTACT: ISTVÁN ALFÖLDI, NJSZT

PRINCIPAL INVESTIGATOR: DR. HABIL. ANDRÁS LŐRINCZ

Abstract. The first task is to discover and characterize the substrate of distributed computation, which is the Internet in our case. The second task is to develop methods (models) for collaborative interactions.

According to recent discoveries, the Internet has a special structure, called scale-free small world. Up to some extent, the structure of Internet is described by different models. In a parallel project we have introduced a novel model, the HebbNet, which represent competitive-collaborative activities in a more sensible way than existing other models of the Internet. The particular properties of the Internet and HebbNets are described here.

Second, we have downloaded a portion of the Internet. The working and the gain of efficiency using competitive-collaborative model were studied on this downloaded part. A model for injecting novel information into the downloaded portion of the Internet was developed. Computer simulation demonstrate the idea: (i) competition gives rise to a division of work, the division of searched portions of the reinforced robotic Internet agents, (ii) the division of work gives rise to specialization and increased efficiency for the specialized robotic agents.

The conclusion section (Section 5) contains information about necessary future steps, which are not covered by the present project and are being at basic research phase at this very moment.

The Report is extended by Appendices:

(1) Description of concepts and methods, like reinforcement learning, text classification, SVMs are appended to the Report are provided in the Appendix

(2) A large body of experiments on HebbNets are provided in an accompanying technical report, titled: Meta level analysis of Hebbian evolving networks `http://people.inf.elte.hu/lorincz/Files/NIPG-ELU-21-04-2002.pdf`

Date: 6th June 2002.

Author: András Lőrincz

Contents

List of Figures

1. Introduction

Cooperation requires some knowledge about the 'arena' of cooperation. In our case, this arena is the Internet. Interestingly, recent discoveries demonstrated that a large variety of networks, such as social networks, scientific collaborative networks, hardware networks at the level of proxies, Internet connections, html links, etc., share the same structure. Moreover, the suspicion has been increasing that all (or at least a large number of) 'sustained' systems share the same common connectivity property, provided that the underlying concept of connectivity is discovered. Two concepts have to be explained here:

- the concept of a sustained system and
- the word connectivity.

A system is called sustained (i) if it is not closed from the environment and (ii) if it is receiving energy from the environment. A cell of the body, the earth of the solar system, the research group of the University are such examples.

Connectivity is a loosely defined word to describe interaction amongst collaborative-competitive units. It is a loose term, because in its crudest form is not concerned with the vehemence, but only with the existence of the interaction. The usage of this word is as follows: If two units are engaged in kind of interaction then *they are connected*. Interaction can be directed or could be mutual. For example, (mostly) directed interaction occurs between the wind and the sailboat. Interactions are never fully directed; the sailboat – although up to a minor extent, but – influences the local properties of the wind. An example for a more balanced interaction is the teacher student relation, whereas an almost fully balanced may occur for some ants. Given the type(s) of interaction, one ends up with a directed or with an undirected network of interacting units.

Report organization.

(1) First, novel concepts of the Internet are provided. This part of the report contains the description of our model of the Internet, which we call HebbNets.
(2) The second part contains the results of our competitive-collaborative model. Specialization of Internet agents, division of work and the increase of efficiency are demonstrated on a downloaded part of the Internet.
(3) Discussions can be found and conclusions are drawn in Sections 4 and 5, respectively.
(4) Details about intelligent crawlers are contained by the Appendix.
(5) A large body of experiments on HebbNets are provided in an accompanying technical report, titled: Meta level analysis of Hebbian evolving networks `http://people.inf.elte.hu/lorincz/Files/NIPG-ELU-21-04-2002.pdf`

2. Internet and other networks

2.1. **Short overview.** Interactive systems with network structure have recently become a fascinating area of research interest. Dynamic systems that govern the formation of these networks are of central importance because of the intriguing similarities among many biological, social, and information processing networks. The original model of Watts and Strogatz [D. J. Watts and

S. H. Strogatz, Nature **393**, 440, (1998)] explore random restructuring of links among a finite number of 'nodes'. Other works have dealt with growing structures and optimization of link structure of finite systems. In this paper we present and study the 'HebbNets', networks in which structural changes are governed by Hebbian learning rules. We find that Hebbian learning is also capable to develop all kinds of network structures, including small-world and scale-free networks. Our results may support the idea of Edelman [G. M. Edelman: Neural Darwinism, New York, Basic Book (1987)] that the development of central nervous system may have evolutionary components.

2.2. **Introduction.** The last few years have witnessed the evolution of novel and efficient ways of describing complex interactive systems (CISs). The novel description of a CIS is based on graphs with nodes and (directed) edges, representing constituents of the system and the interactions among them. Classification of CISs is based on the statistical properties of the network. Similar network structures may be found in many different fields. Both these systems and the corresponding dynamical models which define the formation of these networks may be of fundamental importance to understand the behaviors of CISs. The interest in CISs is boosted by the intriguing similarities of biological, social, and information processing networks [Watts and Strogatz, 1998, Kleinberg, 1998, Albert et al., 1999, Barabási and Albert, 1999, Barabási et al., 2000, Marchioria and Latorac, 2000, Latora and Marchiori, 2001, Bohland and Minai, 2001, i Cancho and Solé, 2001, Albert and Barabási, 2002]. The original model of the the World Wide Web (WWW) by Watts and Strogatz [Watts and Strogatz, 1998] explored random restructuring of the links among a finite number of 'nodes'. Barabási and his colleagues introduced the concept of *preferential attachment* to provide a more sophisticated model of WWW [Albert et al., 1999, Barabási and Albert, 1999]. The idea has been extended to other types of networks [Barabási et al., 2000]. Another approach deals with optimization of link structure of finite systems [i Cancho and Solé, 2001].

Probably the most complex network is inside us: the most exciting properties of our brain have a lot to do with the special connection system among its units. Although our knowledge on the building blocks is increasing, we are still far from a complete understanding of the structure and research of the central nervous system (CNS) is primarily targeting at these questions. At the same time, many similar questions on the architecture of the WWW arise. The recognition of the parallel nature resulted in the intuitive idea to highlight the similarity between these descent phenomena. From one hand, it has been suggested to use the mutual activity correlation (that is the original form of Hebbian learning) in modeling organizational learning [Kulkarni et al., 2000]. On the other hand, similar structural characteristics of the web and the single completely described nervous system of the nematode *C. elegans* has been reported [Watts and Strogatz, 1998]. In this paper we investigate the question whether an evolving network, governed by Hebbian rule, has the same or similar properties as found by studying the web or social networks. The question is of central importance: in seeking the answer we hope to find general underlying principles, which give rise to small-world like structures in cooperative-competitive systems.

It may be important to note here that concepts of Hebbian learning have undergone revolutionary changes in the last few years. The original suggestion of Hebb [Hebb, 1943] has been modified by recent findings [Markram et al., 1997, Magee and Johnston, 1997, Bell et al., 1997]. (For a review, see, e.g. the work of Abbott and Nelson [Abbott and Nelson, 2000]). The novel concept is called spike-time dependent synaptic plasticity (STDP). The underlying mathematical concepts could correspond to linear and non-linear versions of principal component analysis [Oja, 1982, Abbott and Nelson, 2000].

2.3. **Description of HebbNet.** We intended to construct a model network (HebbNet in short) in which the structural changes are governed by Hebbian rules and the interaction with the environment and all the interacting elements of the network are as simple as possible. We assume that the inputs to the network have no spatio-temporal structure, i.e., the input will be randomly generated. In turn, search for either spatial or temporal correlations is meaningless. In contrast to the web (WWW), the input does not depend on the actual network structure. Results on temporally symmetric STDP (Fig. 1) are reported here. Quantitatively similar results can be derived for temporally asymmetric STDP. Results are thoroughly detailed in our technical report (TR) [Palotai et al., 2002].

Our models consist of N number of simplified integrate-and-fire like 'neurons' or nodes. The dynamics of the internal activity is written as

$$\dot{a}_i = \sum_{j \in \mathcal{S}} w_{ij} a_j^s + x_i^{(ext)}, \tag{1}$$

for $i = 1, 2, \ldots, N$. Here $x^{(ext)} \in (0,1)^N$ denotes the randomly generated input from the environment, a_i is the internal activity of neuron i, w_{ij} is the connection strength from neuron j to neuron i. $\mathcal{S}$ is the set of nodes with nonzero connection to node i. a_j^s is equal to 1 if the j^{th} neuron fires (the neuron outputs a spike) and superscript s stands for 'spiking'. $w_{ij} * a_j^s$ is the excitation received by neuron i from neuron j when neuron j fires. After firing ($a_j^s = 1$) a_j is set to zero immediately. Equation 1 describes the simplest form of 'integrate–and–fire' network models.

Instead of applying a given threshold ϑ, we simply sorted out r_f percent of nodes with the highest activity. This choice, which requires global information about network properties is only a matter of convenience: quantity ratios are used throughout the paper. Using threshold or treating internal activity as firing probability have yielded qualitatively the same results [Palotai et al., 2002]. The latter methods make use of local (instead of global) properties. Synaptic strengths were modified as follows:

$$\dot{w}_{ij} = \sum_{(t_i, t_j)} K(t_j - t_i) a_i^{t_i, s} a_j^{t_j, s}, \tag{2}$$

where K is a symmetric kernel function which defines the influence of the temporal activity correlation on synaptic efficacy. The form of our chosen kernel is shown in Fig. 1. The kernel is a function of the time differences and is independent of time shifts. Given the large variety of neuronal connectivity and interaction types, we extend strict neurobiological modelling, where experimental observations are directly applied in choosing the kernel function

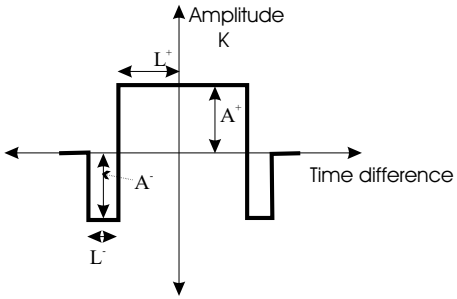


FIGURE 1. **The symmetric kernel function**
Dependence of temporal kernel on time difference $t_i - t_j$ between spike sent by neuron j to neuron i and spike emitted by neuron i. Kernel parameters used in the computer studies: (i) ratio of the length of weakening and of strengthening regions ($r_L = \frac{L^+}{L^-}$) and (ii) ratio of the amplitudes ($r_A = \frac{A^+}{A^-}$). During our simulations r_L was set to 1, whereas r_A was 0.5. For results on other parameters, see the technical report [Palotai et al., 2002].

(e.g. as findings in [Zhang et al., 1998] have been used in [Levy et al., 2001]). For the sake of generality, broader ranges of parameters are studied. However, it is worth mentioning that such symmetric kernels do exist in real neuronal networks [Abbott and Nelson, 2000].

In the first place, we have been interested in the emerging local and global connectivity structure of $\mathbf{W}$. It has been debated whether the global structure property (L, the average number of edges on the shortest path) and the clustering coefficient (C) proposed by Watts and Strogatz [Watts and Strogatz, 1998] are appropriate for describing weighted networks [Latora and Marchiori, 2001]. We have applied both the Watts and Strogatz measures and the *connectivity length* (D_h) measure suggested in [Marchioria and Latorac, 2000], and similar results have been obtained [Palotai et al., 2002]. Due to its simplicity and logic we have chosen the latter to present our results.

The physical distance between two nodes is considered as the inverse of their connection weight. One can also talk about distances of (indirect) paths between the same two nodes. The *shortest distance* on the graph d_{ij} is defined by the lower bound of all possible path length in the graph from node j to node i. Consider that path length — apart from a constant scaling factor — is inversely proportional to *delivery time*, or *rate of information transfer*. The method of connectivity length[Marchioria and Latorac, 2000] makes use of the efficiency of information transfer instead of direct physical distances between nodes: The local harmonic mean distance for node i is

$$D_h(i) = \frac{n^j}{\sum_{j:w_{ij}>0} \frac{1}{d_{ij}}}, \tag{3}$$

where n^j is the number of neurons around neuron i with $w_{ij} > 0$. The mean global distance in the network can be characterized by the following quantity:

$$D_h = \frac{N(N-1)}{\sum_{i,j} \frac{1}{d_{ij}}}. \tag{4}$$

Global distance provides a measure for the *size* (or the diameter) of the network. According to [Marchioria and Latorac, 2000, Bohland and Minai, 2001] local harmonic mean distance measure may correspond to $1/C$ (inverse of the clustering coefficient), whereas the global value corresponds to L.

2.4. **Results.** In our simulations the network parameters were systematically varied [Palotai et al., 2002]. The emerging structures were most sensitive to the following parameters: (i) the magnitude of the external excitation (defined by the average percentage of neurons receiving excitation from the environment, r_{ex}) and (ii) the ratio of firing neurons (r_f). According to the results, for $N = 100$, two essentially different parameter regions may be separated: Region 1: the values of both r_{ex} and r_f parameters are under 0.5. Region 2: the values of both r_{ex} and r_f are above 0.5. The following figures show some typical results.

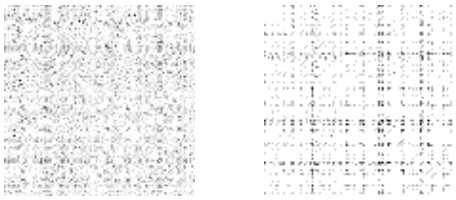


FIGURE 2. **Connection matrices (W) for** 100 **neurons** **A**: Connection matrix for Region 1. r_{ex}, the ratio of externally excited neurons is 30%. r_f, the percent of neurons allowed to fire is 30%. **B**: Connection matrix for Region 2. $r_{ex} = r_f = 70\%$. The matrixes are normalized and the darker colors correspond to higher values (black: 1, white: 0).

While Fig. 2(**A**) resembles a random connection matrix, Fig. 2(**B**) represents a sparse, yet almost fully connected structure. Only 4 out of 100 neurons became isolated, while the number of connections was 1796 out of the possible 10,000. After normalization (by setting the maximum component of matrix **W** to 1), the average weight was about 0.24. Similar results were obtained in a broader parameter range.

The following figure is to characterize the size of the network. We compared the resulting HebbNet structures with a random net, in which the same weights of the dynamic network have been randomly assigned to different node pairs. Fig. 3 highlights clearly the emerging small-world properties (i.e., high clustering coefficients and short path lengths). Although the global connectivity length was almost the same for all the HebbNets and their corresponding random nets, local distances are much smaller in Region 2. That is, connectivity structure is sparse but information flow is still efficient. The Watts-Strogatz

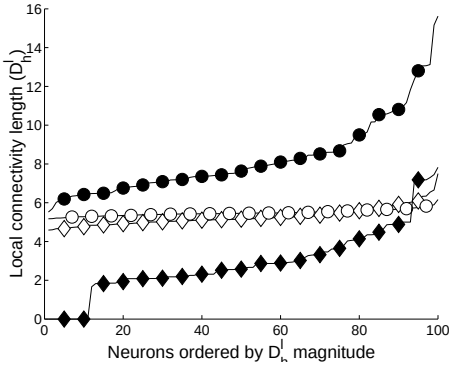


FIGURE 3. **Harmonic mean distances**
This figure displays the local harmonic mean distances in ascending order for both regions. For better visualization not all data points are marked and the point are connected with a solid line. Lines with diamonds belong to the network that was developed using STDP learning. Lines with circles belong to networks with the same but randomly redistributed weights. Line with empty (solid) markers represent a HebbNet of Region 1 (Region 2). In region 2 the emerging HebbNet has much smaller local harmonic distances as compared to the corresponding random net. Global harmonic mean distance for the original and for the randomized networks in Region 1 (Region 2) are $D_h = 5.25$ and $D_h^r = 5.01$ ($D_h = 6.28$ and $D_h^r = 4.98$), respectively.

(WS) rewiring model [Watts and Strogatz, 1998] is known to yield both random and small-world structures. For $N = 100$ neurons with 6 connections of 0.5 strength to each and with rewiring probability p around 0.9, the structure obtained by the WS model is in good agreement with our random net. Our small-world net, on the other hand, can be characterized with rewiring probability between 0.1 and 0.2.

Limited scale-free properties seem to emerge for such small networks in Region 2 as shown by Fig. 4. The robustness of the network to the magnitude of the excitation is illustrated by the last figure. By increasing the excitation level, the efficiency of the random net is drastically decreasing down to one third of its original level, whereas the efficiency of the small-world network does not change too much in the same region. However, for the network with parameters $r_A = 0.5$, $r_L = 0.67$ (Fig. 5(**A**)), there is a sharp cut-off around 50%, where local distances suddenly drop in both networks, due to the high ratio of excitations. Qualitatively similar behavior can be found for network parameters $r_A = 0.5$, $r_L = 1.5$ (Fig. 5(**B**)), but the cut-off is around 90%. Detailed results belonging to the parameter range given in Table 1 can be found in our technical report [Palotai et al., 2002].

In summary, we have demonstrated that small-world networks with scale-free domains may emerge under STDP Hebbian learning rule. The existence of such 'HebbNets' may support the speculative view of Kandel et al.

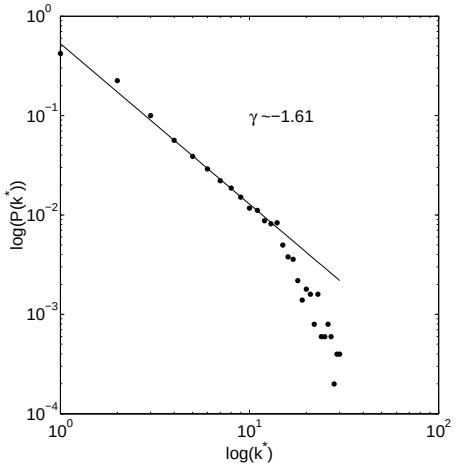


FIGURE 4. **Cumulative weighted degree distribution** Cumulative distribution of the sum of the weights of outgoing connections for the network ($r_{ex} = r_f = 70$, $N = 100$, the number of histogram bins (discretization) is $d = 30$). k^* denotes the discretized weighted connection number. The figure shows scale-free properties (power-law distribution $P(k^*) \approx k^{*-\gamma} e^{-k^*/\xi}$ with $\gamma \approx 1.61$ and $\xi \approx 20$) in a relatively broad region.

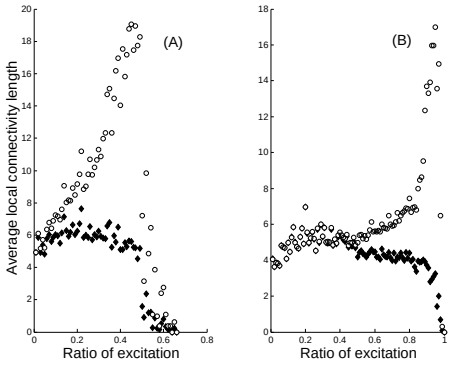


FIGURE 5. **Average local distance vs. excitation ratio** These two figures demonstrate the systems robustness to the magnitude of the excitation.
A $r_A = 0.5$, $r_L = 0.67$, **B** $r_A = 0.5$, $r_L = 1.5$. Diamonds: average local distances for the evolving network. Circles: average local distances for the corresponding random net.

[Kandel and O'Dell, 1992] that structural development and learning plasticity in CNS may have a common basis. Furthermore, our results support the original suggestion of Edelman that learning in the CNS may have evolutionary components [Edelman, 1987]. Present models on small-world networks may

Parameters					
r_L	1/2	2/3	1	3/2	2
r_A	1	1/2	1/4	1/8	1/16
r	0.01 - 0.99				
θ	0.1 - 0.9				

TABLE 1. **Parameter ranges for random inputs, symmetric kernel function and 100 neurons** [Palotai et al., 2002]
r_L: ratio between strengthening and weakening time intervals of the kernel. r_A: ratio of between amplitudes (see Fig. 1). r: percent of the neurons with external input. θ: percent of neurons allowed to fire (excitability).

allow for the following. At early stages of development, the evolutionary component may be guided by preferential attachment like mechanisms. At later stages, this component may be maintained by noise randomly generated within the CNS.

As far as other evolving networks are considered, the profound implication of our result is that local (Hebbian) learning rules may be sufficient to form and maintain an efficient network in terms of information flow. This feature differs from existing models, such as the model on preferential attachment (which is equivalent to an increasing number of non-interacting surfing agents with constant trapping rate) [Barabási and Albert, 1999], the global optimization scheme [i Cancho and Solé, 2001], and also from the original Watts and Strogatz model [Watts and Strogatz, 1998].

3. Competition, which Results in Collaboration

Rapid development of Internet technologies increases the use of this unique medium for collaboration. While interoperability is a main focus of these collaborative efforts, privacy protection, along with reputation management is in demand. Recently several works have emerged that focus on these problems. However, works on providing confidentiality and reputation are limited. For example, most of the works focus on a particular problem and problem domain, say reputation management for e-commerce applications. Considerably less work has been done to accommodate novel requirements, such as collaborative access requirements and developing protocols that allow anonymity while support accountability. Furthermore, most of the existing works were developed with special application in mind and are not generally applicable.

3.1. **Consequences of Novel Findings about the Internet.**

We assume - without limiting generality - that the goal of the Internet is to publish. Publication is seen here as a general way of accomplishing a task and making it available for others in a understandable (readable) form. In turn, the goal of collaboration on the Internet can be seen as editing to reach the goal in an efficient manner. In our approach we take into account the general features discovered over the last few years about publishing and collaboration. These features can be best described in terms of networks. Take an author

(author A) and his/her co-authors. Say, author B is a co-author of author A. This is a minimal network. The 'distance' between the two co-authors is 1. Alternatively, there is a link between authors A and B. B has co-authors, too. Author C is a co-author author B, whereas author C is not a co-author of author A. There are links between authors A and B and authors B and C. However, there is no link between author A and C. The distance between authors A and C is equal 2. A broad range of social networks shows general features when formulated this way [Watts and Strogatz, 1998, Barabási and Albert, 1999, i Cancho and Solé, 2001]. The general features are called 'small-world phenomenon' and 'scale-free networks'. For example, similar networks have been discovered

(1) amongst Hollywood actors (collaboration is playing in the same movie),
(2) authorship in scientific communications,
(3) social networks ('collaboration' means knowing each other), and
(4) in the so called Erdos-point in mathematics.

One should design security architecture for originally non-hierarchical networks. Further, one may allow the development of a hierarchy based on the efficiency of the collaborating partners. An example is the Erdos point, where there is a clear hierarchical structure with Paul Erdos, the famous mathematician on the top of that hierarchy. Our model is called 'Editorial Board' and may have the following ranks:

- reader
- author
- reviewer
- board member
- editor

Other structures are possible, including specialization in the form of action editors, to mention a simple extension.

Hierarchy can be imposed in a rigid fashion. One corresponding structure is

- customer/user
- worker
- quality assurance
- member of the advisory board
- CEO

Note, that the role of the reader or that of the user is special: they provide the reinforcing feedback of the external world by purchasing the information or the product.

At this point, we note the following. Starting from efficiency based self-organizing communities will favor competition as a derivative within and amongst communities. Competition will require security rules. Security rules can be imposed at two different levels: (i) at the level of the self-organizing community, and (ii) at the level of interaction between such communities. These concepts are novel, so we give an example. If a particular self-organizing community is made of readers and authors only, then the collaboration with another community with more strict quality assurance, e.g., board members, action editors, editor will be limited. Limitations are also restricted by the definition of readers. If the community is closed then special rules for interaction with other

communities need to be established. If anybody can be a reader, then self-organizing collaborative efforts between communities may emerge. In turn, some communities will be efficient whereas others will be less efficient. The efficiency is governed by - at least - two factors:

(1) The efficiency gained by collaboration
(2) The slowing down of project organization as determined by the time constraints of security rules

Note that we aim to develop concepts for collaborative schemes amongst humans and Internet robots. The time constraints could be most restrictive for large organizations.

A good way to think of such communities is to consider

(1) the participants of the community as agents,
(2) the community as an assembly of agents,
(3) the security rules of the community as the language of the community,
(4) the interaction amongst communities as communication between agents speaking possibly different 'languages'.

Alike to real languages, there will be a competition amongst these languages. This competition will favor languages, which are simple for level communication and which are abundant. The formulation of the S2OS as agents speaking languages is not the subject of this proposal. At this point, one may think to establishing security rules,

(1) which meet the common general structure of self-organizing human collaboration,
(2) which allow the derivation of 'pre-wired' hierarchies,
(3) which allow collaboration amongst communities,
(4) which can be secure if required by the founders of the self-organizing community.

The proposed model consists of independent, autonomous sub-systems, governed by a set of local goals. Dynamics of these independent subsystems defines a global, self-organizing model without the necessity of central authorization. The aim is to develop a system architecture that supports collaboration while guarantees information confidentiality, integrity, and availability. In addition, anonymity of the initiator and responder are guaranteed as well as accountability that is used to deter misuse or for billing services.

Our long-term goals are to provide privacy (hide identity of initiator and responder), provide mechanism for secure online collaboration (protocols what can be shared between the parties, policy that defines who is allowed to access what and what level), and provide accountability (misuse detection, or billing purposes. Our assumption is, that users may form ad-hoc communities based on their interest and expertise. The same user may belong to different communities at different roles or may assume different roles within a given community.

In the editorial board example, all users are allowed to submit papers to be reviewed, and all users are allowed to review papers. Both submitters and reviewers are classified according to the quality of work they perform. Initially all users and all papers are assigned an 'initial' impact factor. Based on the

submitted reviews, the impact factor of each publication may increase or decreases. In addition, reviewers' impact factor will be changed based on the response to their comments, i.e., level of update of the reviewed paper based on the evaluation.

Additional restrictions are involved on the model to guarantee security and anonymity while preventing frauds by providing accountability. For example, a user may not be able to change his/her impact factor by login in as new user, submit the same paper to get a better evaluation of the paper, or submit high quality reviews of his/her own work. Although, some of the above concepts have been independently considered by researcher, there does not exist a complete model that supports both security and anonymity in a single, dynamic environment, while preserving accountability.

3.2. **Self-Organizing System (SOS) for Online Collaboration.** SOS may be distributed and may have human as well as robotic participants. In order to demonstrate the generality of our approach a strictly robotic example is provided here. Access control issues, and some experiments about their efficiency are provided to support our argumentation.

An Internet robot may represent a hierarchy. The robot is on a host computer and will be called 'hostess'. The hostess can launch topic specific searches using crawlers, which download Internet documents one-by-one, using the links of the documents. Intelligent crawlers adapt according to the reinforcing signal received from the external world and transferred by the hostess. Communication is possible amongst

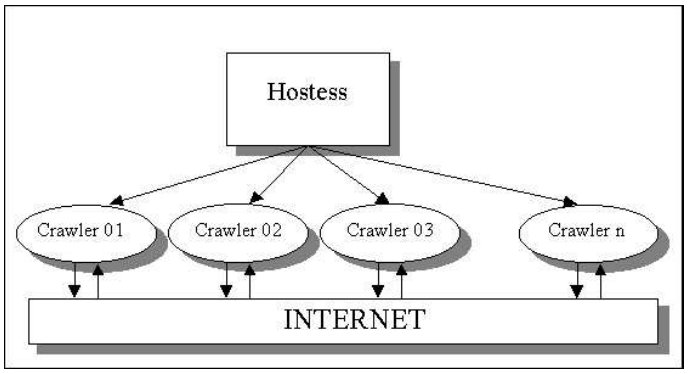


FIGURE 6. **Hostess-crawler system**
Internet is explored by crawlers (which download and examine documents), whereas communication with other entities is governed by the hostess, which can launch and modify crawlers based on reinforcements.

- crawlers and
- hostesses

Communication can be direct or indirect. For example, if the hostess launches crawlers on the same topic and provides positive reinforcement only to the one, which brings the information first, indirect communication happens. This indirect communication will manifest itself as a competition. The competition

will be directly addressed if crawler maintain and refresh a list of good links - we call those weblogs - which used to provide the most positive reinforcement from the crawler. One may see these weblogs as local sub-crawlers, which have starting points and search a smaller neighborhood. The algorithm of weblog selection is provided in Fig. 7

1) $s \leftarrow startNode$
2) FOR $i = 1$ *To memory*
   a) $s \leftarrow chooseRL(frontier)$
   b) $wl \leftarrow s, \quad w(s) \leftarrow \sum_{k=i}^{memory} r_k$
3) $sort(wl, w(s)), \quad head(wl, wlsize)$
4) REPEAT *(for every local search)*
   a) $w'(s) = w(s) * \gamma, \quad \forall\, s \in wl$
   b) $s \leftarrow choose(wl)$
   c) FOR $i = 1$ *To memory*
      i) $s \leftarrow chooseRL(frontier)$
      ii) $wl \leftarrow s, \quad w'(s) = w(s) + \sum_{k=i}^{memory} r_k$
      iii) $sort(wl, w(s)), \quad head(wl, wlsize)$
5) UNTIL end of experiment

FIGURE 7. **Algorithm for forming a weblog**
The algorithm has three important parameters: (i) *'memory'*: size of environment around a node of the network, examined during the evaluation of that node, (ii) *'wlsize'*: size of the weblog (number of links that can be remembered), and (iii) *'γ'*: discount factor of past rewards received around a node.

The origin of competition can be seen as follows. The small-world property can trap crawlers in a neighborhood where links are abundant and point to each other. Crawlers might not find any novel information. To overcome this limitation, each crawler may maintain a list of links, which are worth to visit. Assume two crawlers: one with a long list and another one with a single link. The second crawler will find the novel information on that single link with high probability. However, that information will not be novel for the first crawler anymore, therefore, the first crawler will not be able to "publish" it at the hostess. As the consequence, the first crawler will lower the value of this link and will visit this site less-and-less frequently.

The downloaded part of the Internet and its division between the two crawlers can be seen in Fug. 9

The competition gave rise to a decrease the age of found relevant document in our model experiments as it is shown in Figs. 10 and 11

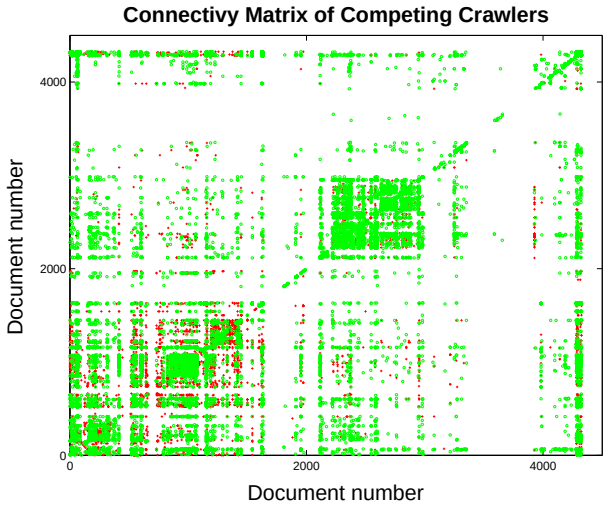


FIGURE 8. **Two competing crawlers using weblogs in a "small-world"**
The figure depicts the connectivity table (representing the links) between nodes. Typical scale-free small world connectivity can be seen here: some nodes have many links pointing onto them (rows), whereas others have a long list of links pointing to other nodes (columns). Green (red) dots depict the links used by the first (second) crawler. Efficient self-organizing division of the world, i.e., the task space, can be seen. Weblogs contained 10 links.

4. DISCUSSION

The competitive-collaborative method has the following advantages. Processing is local: each crawler modifies values of known links based on the reinforcement. There is a global improvement in performance.

Let us consider now interacting hostesses. Some of the hostesses (authors) can be selected to being a 'reviewer', a member of the board, or the editor. The selection could be based on their efficiency, or that the search is divided into topics and central hostesses have access to the context of the search. Another intriguing possibility is that in a general search problem, search topics are may also be subject to competition. Under this condition, small-worlds become larger (!), the 'diameter' belonging to topics is larger, however, searching may become more efficient. Central hostesses may emerge as a result of this competition. The emergence of central hostesses can be promoted by the type of interactions, the access control, allowed by the robotic community (see below). We conclude this paragraph by noting that the hostess-crawler system is one type of SOS, whereas collaborating hostesses can form another type of SOS.

Interaction among the system components fall into two categories:

(1) Interactions within a single SOS
(2) Interactions among different SOSs

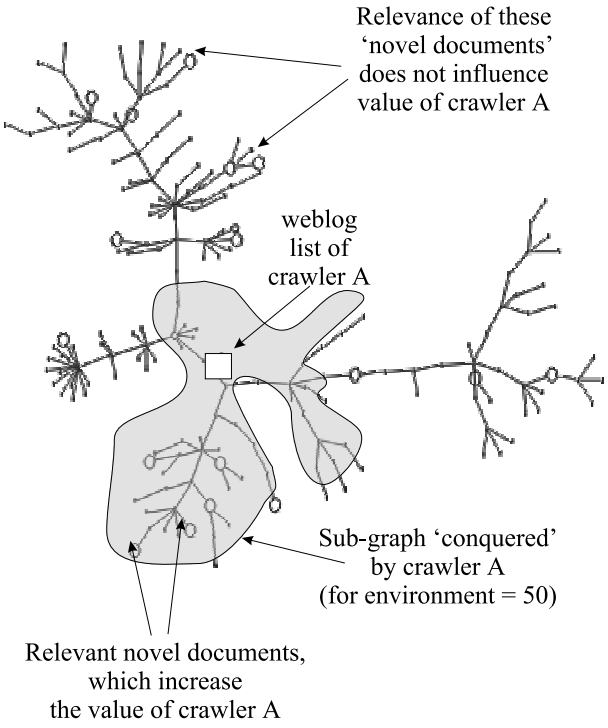


FIGURE 9. **Division of a substructre of the Internet between two-crawlers**
The gray region belongs to crawler A. Both crawlers maintain a list of weblogs, which are efficient restart links to continue search when novel information in the actual neighborhood has been collected. Weblogs are ordered by their values and change dynamically amongst crawlers during competition.

Each SOS is motivated to become stronger. The overall gain of a SOS, in this competitive environment, depends on the success of other S2OSs. Each member of a single SOS supports each other by sharing relevant information. On the other hand, no such information sharing is done between different SOSs. Temporal sequences and synchronizations support this information sharing process. If Crawler A discovers event p on site X then it might be useful to send that information to Crawler B "new information on site X". That information can serve as the "context" for crawler B and can help to accelerate search.

That is, one should be looking for synchronous or spatio-temporal patterns. The ideal tool – as it seems to us – is the two types of Hebbnet learning rules. In turn, one may expect that Hebbnets with symmetric learning rules will represent synchronous patterns, whereas Hebbnets with asymmetric learning rules will represent spatio-temporal patterns. The expected prototype form of such

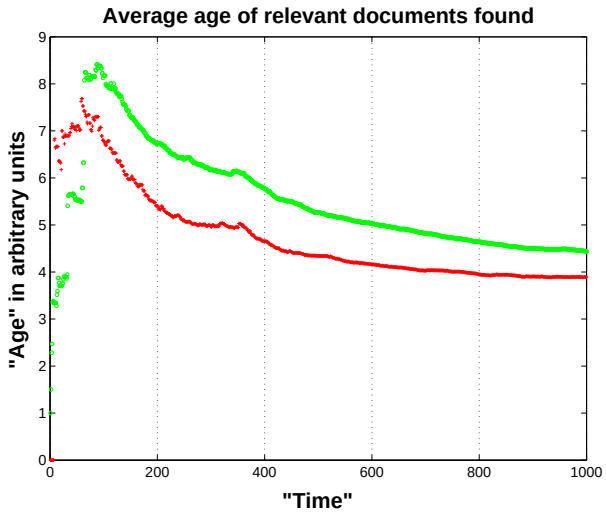


FIGURE 10. **Decrease of age of found novel documents** New documents appear, whereas old documents disappear in this experiment. Competition settles around "time" 600. The overlap amongst links searched by crawlers becomes minimized by this time. Both crawlers have high connectivity weblogs. The age of found novel documents is decreasing afterwards.

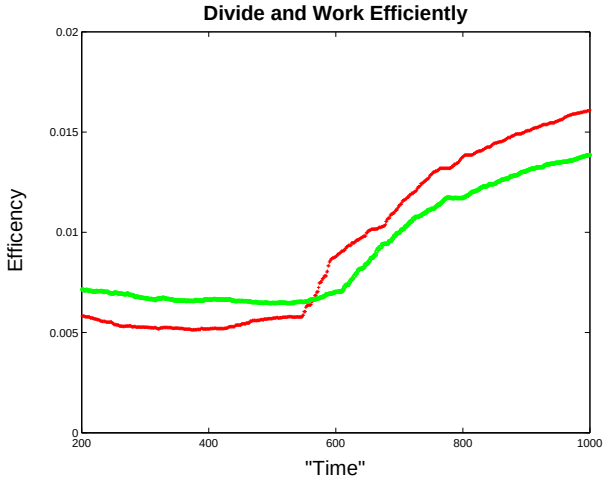


FIGURE 11. **Increase of access to novel documents.** New documents appear, whereas old documents disappear in this experiment. Competition settles around "time" 600. The overlap amongst links searched by crawlers becomes minimized by this time. Both crawlers have high connectivity weblogs. The efficiency of found novel documents is increasing afterwards.

spatio-temporal patterns is an avalanche-like pattern, alike in most `sustained` *systems.*

The point to remember is that communication amongst crawler or hostesses can be

- indirect based upon reinforcement
- direct
  - by sending documents to each other,
  - by sending elements of weblogs,
  - by sending information about the fact that a novel document has been found 'in my area',
- or combinations of any of these.

The communication amongst crawlers or hostesses is defined by the access control rules. These rules are of central importance. These rules determine the type of SOS that will emerge. To each type of communication, the access control model of the corresponding SOS must ensure that a component will not be able to disclose or modify resources if it does not have the proper authorization. These access rights determine the type of co-operation and the distribution of topics amongst the small sub-systems.

5. Conclusions

The competitive-collaborative model is efficient according to our preliminary studies. It allows one to impose competition simply via reinforcement. Reinforcement gives rise to the division of tasks and improved efficiency within subtasks.

The concept can be extended by communication methods other then the indirect communication of the reinforcing signal. Other types of message sending methods open questions about interactions amongst self-organizing systems (SOS), the access control amongst those regarding the type of information to be communicated.

The idea of establishing experience based connections using Hebbnets seem most promising as the tool of collaboration, provided that access control issues are solved. Learning rules of a Hebbnet can discover and, in turn, may allow clustering of synchronous patterns as well as spatio-temporal patterns, most probably in the form of avalanches.

Studies along these lines are beyond the scope of the present contract. Considerations on different possibilities are currently in *basic research phase*. In particular, studies about

- **access control rules**
- **learning and recognition of spatio-temporal patterns in early phases**
- **secure communication methods**

have been started and might reach a more mature state in about three months.

Appendix

6. Crawlers

6.1. **Introduction.** The number of documents on the world-wide web is way over 1 billion [Diligenti et al., 2000]. The number of new documents is over 1 million per day. The number of documents that change on a daily basis, e.g., documents about news, business, and entertainment, could be much larger. This ever increasing growth presents a considerable problem for finding, gathering, ordering the information on the web. The only search engine that may still warrant that the information it provides is not older than 1 month is AltaVista[1]. However, the number of indexed pages on Altavista is about 250 million documents. Google[2], on the other hand, is indexing about 1,300 million pages, but Google does not warrant any refresh rate of these documents.

The problem is complex: These search engines are not up-to-date and information gathering is not always efficient with these engines. Search engines may offer too many documents; sometimes on the order of hundreds or many thousands. Many web pages have no value, e.g., by making use of a large set of keywords, or being simply huge collections of documents originating from broad sources.

Specialized, possibly personalized crawlers are in need. This problem represents a real challenge for methods of artificial intelligence and has been tackled by several research groups [Cho et al., 1998, Dean and Henzinger, 1999, Chakrabarti et al., 1999a, McCallum et al., 1999, Kolluri et al., 2000, Lawrence, 2000, McCallum et al., 2000, Mukherjea, 2000, Murdock and Goel, 1999]. One of the first attempts in this direction was made by Chakrabarti et al. [Chakrabarti et al., 1999b] who put forth the idea of *focused crawling*. To understand the idea, let us consider crawling in general. Assume that 'you are at a node' of the web. This node has been analyzed and you have to decide what to do next. It is very possible that relevant information can be find in the immediate neighborhood of this node. In turn, you download all the documents next to you and start to analyze those documents. Doing so, you may found relevant documents or may not. When you are done you have the option to download all the documents that are two steps away from you and to analyze those documents. This approach is well known in the AI literature and is called breadth first technique. However, the world-wide web is *'small'*: The WWW had about 800 million nodes in 1999 and the number of minimal hops required to reach most documents from any particular document was 19 [Albert et al., 1999]. Such connectivity structure between units is called 'small world'. In turn, breadth first search incurs an enormous burden as a function of depth. At one point (at a given depth) breadth first search needs to be abandoned and a decision is to be made to which node to move next. To decide on that move, the values of the nodes need to be estimated from the point of view of the goal of the search. *Focused crawling* is based on this idea. Focused crawling makes an attempt to classify the content of the document. If the document falls into the search category then the document is downloaded and the links of the documents are followed.

[1] http://www.altavista.com
[2] http://www.google.com

Diligenti et al. [Diligenti et al., 2000] have recognized the pitfall of focused crawling: searched information on the web is typically *hidden*: Sites of particular interest may have a lower number of directed links then sites of general interest. In turn, we might face the 'needle in the haystack' problem with the haystack being sites on general interest. The hidden property is thus the implicit consequence of our particular interest.

Let us consider sites dealing with support vector machines (SVMs). Sites about SVMs are not typical on the web. Not all sites dealing with SVM are linked. In turn, focused crawling could be rather inefficient and this direct search for SVM sites might fail. On the other hand, most of the SVM sites are within (i.e., linked to) academic environments, or within sites dealing with information technology. These topics are much more general and might have much more links and a much higher 'visibility'. In turn, searching for the environment of SVM sites, could be much more efficient. A hand-waving argument can be given as follows. Documents are linked to each other. Links are made by those for whom the document has value. These links form the one-step *context* of the document. The one-step context, in turn, may be characteristic to the document. The one-step environment of the document (i.e., documents that are one step away), documents that are two steps away, etc., form the 'context' of the document. When we search for a document, by definition, we shall encounter the environment of the document first. In turn, first we might search for the environment of the document. This is the idea behind 'context focused crawling' (CFC) [Diligenti et al., 2000]. This idea, which is trivial for graphs with high clustering probability (e.g., regular lattices), could be criticized for the case of 'small worlds', when documents – on average – are about as far as the environment of the document. However, the question is intriguing, because the visibility could be much less for searched documents than that of the environment of the searched documents.

CFC does not take into consideration the varieties on the web: Environments may differ. For example, for small universities or for small research institutes 'one-step context' may correspond to 'two-step context' for large departments of large universities. If the order of contexts might change then CFC will go close and will miss the documents. In turn, the decision whether to 'stay and download' at a given site or 'not to download but move' can be seriously jeopardized. Fast adapting value estimation method may provide an attractive solution to this search problem where information is hidden within not-yet-experienced environments. The environment of high value documents can provide reinforcing feedback in a straightforward fashion. Interestingly, reinforcement learning (RL) has not been found particularly efficient for searching the world-wide web [Rennie et al., 1999]. The efficiency of RL, however, depends strongly on feature extraction. It seems natural to explore the CFC idea as the initial feature extraction method for RL. Here we show combine CFC with RL to search on the web.

6.2. **Methods.**

6.2.1. *Preprocessing of texts.* There is a large variety of methods that try to classify texts [McCallum, 1996, Blum and Mitchell, 1998, Dumais et al., 1998, Kaski, 1998, Chakrabarti et al., 1998, Kolenda et al., 2000, Mitchell, 1999, McCallum, 1999,

Hofmann, 2000, Nigam et al., 2000, Kabán and Girolami, 2000, Vinokourov and Girolami, 2000a, Joachims, 2000, Dominich, 2000]. Most of these methods are based on special dimension reduction. First, the occurrence, or sometimes the frequency of selected words is measured. The subset of all possible words ('bag of words' (BoW)) is selected by means of probabilistic measures. Different methods are used for the selection of the 'most important' subset. The occurrences (0's and 1's) or the frequencies of the selected words of the subsets are used to characterize all documents. This low – typically 100 – dimensional vector is supposed to encompass important information about the type of the document. Different methods are used to derive 'closeness measures' between documents in the low dimensional spaces of occurrence vectors or frequency vectors. The method can be used both for classification, i.e., the computation of decision surfaces between documents of different 'labels' [Blum and Mitchell, 1998, Dumais et al., 1998, Joachims, 2000, Nigam et al., 2000] and clustering, a more careful way of deriving closeness (or similarity) measures when no labels are provided [McCallum, 1996, Kaski, 1998, Hofmann, 2000, Kabán and Girolami, 2000, Vinokourov and Girolami, 2000b].

We tried several BoW based classifiers on the 'Call for Papers' (CfP) problem[3]. CfP is considered a benchmark classification problem of documents: The ratio of correctly classified and misclassified documents can be automated easily by checking whether the document has the three word phrase *'Call for Papers'*, or not. Classifiers were developed for one-step, two-step environments, etc., for CfP documents. We found that these classifiers perform poorly for the CfP problem. In agreement with published results [Dumais et al., 1998], supervised SVM classification was superior to other methods. SVM was simple and somewhat better than Bayes classification. However, SVM requires a large number of support vectors for the CfP problem.

6.2.2. *SVM classification.* The SVM classifier operates similarly to perceptrons. SVM, however, has better generalizing capabilities, see, e.g., the comprehensive book of Vapnik [Vapnik, 1995] a tutorial material [Smola and Schölkopf, 1998], comparisons with other methods [Guyon et al., 1992, Cristianini and Shawe-Taylor, 1999], improved techniques [Keerthi et al., 1999] and references therein[4]. The trained SVM was used in 'soft mode'. That is, the output of the SVM was not a decision (yes, or no), but instead, the output could take continuous values between 0 and 1. A saturating sigmoid function[5] was used for this purpose. In turn, (i) the non-linearity of the decision surface was not sharp, (ii) for inputs close to the decision surface the classifier provides a linear output. The output of the sigmoid non-linearity can be viewed as the probability of a class. These probabilities for the different classes are distinct yardsticks working on possibly different features. The RL algorithm was used to estimate the *value* of these yardsticks.

[3]The CfP problem is defined by deleting the phrase 'call for paper' from the document, executing search on the internet and considering each document that contains the phrase 'call for paper' a 'hit'.

[4]Note that SVM has no adjustable parameters.

[5]$\text{output} = \frac{1}{1+\exp(-\lambda * \text{input})}$

6.2.3. *Value estimation.* There is a history of value estimation methods based on reinforcement learning: Some of the important steps — judged subjectively — are in the cited papers: [Korf, 1985, Minton, 1988, Sutton, 1988, Watkins, 1989, Schmidhuber, 1991, Mahadevan and Connell, 1992, Dayan and Hinton, 1993, Kaelbling, 1993, Rummery and Niranjan, 1996, Littman et al., 1995, Mataric, 1997, Dietterich, 2000]. A thorough review on the literature and the history of RL can be found in [Sutton and Barto, 1998]. In our approach, value estimation plays a central role. Value estimation works on states (s) and provides a real number, the *value*, that belongs to that state: $V(s) \in \mathrm{R}$. Value estimation is based on the *immediate rewards* (e.g., the number of hits) that could be gained at the given state by executing different actions (e.g., download or move). Value of a state (a node, for example) is the long-term cumulative reward that can be collected starting from that state and using a *policy*. Policy is a probability distribution over different actions for each state: policy determines the probability of choosing and action in a given state. Policy improvement and the finding of an optimal policy are central issues in RL. RL procedures can be simplified if all possible future states are available and can be evaluated. This is our case. In this case one does not have to represent the policy. Instead, one could evaluate all neighboring nodes of the actual state and move to (and/or download) the one with the largest estimated long term cumulated reward, the estimated value. Typically one includes random choices for a few percentages of the steps. These random choices are called *'explorations'*. The estimated value based greedy choice is called *'exploitation'*.

If the downloaded document contains the phrase 'call for papers' then the learning system incurs an immediate reward of 1. If a downloaded document does not contain this phrase then there is negative reward (i.e., a punishment) of -0.01. These numbers were rather arbitrary. The relative ratio between reward and punishment and the magnitude of the parameter of the sigmoid function do matter. These parameters influence learning capabilities. Our studies were constrained to a fixed set of parameters. One may expect improvements upon optimizing these parameters for a particular problem. In our case, search over the internet was time consuming and prohibited this optimization.

Value estimation makes use of the following upgrade

$$V^{+}(s_t) = V(s_t) + \alpha * (r_{t+1} + \gamma * V(s_{t+1}) - V(s_t)) \tag{5}$$

where α is the learning rate, $r_{t+1} \in \mathrm{R}$ is the immediate reward, $0 < \gamma < 1$ is the discount factor, and subscripts $t = 1, 2, \ldots$ indicate action number (i.e., time). This particular upgrade is called temporal differencing with zero eligibility, i.e., the TD(0) upgrade. TD methods were introduced by Sutton [Sutton, 1988]. An excellent introduction to value estimation, including the history of TD methods and description on the applications of parameterized function approximators can be found in [Sutton and Barto, 1998]. Concerning details of the RL technique, (a) we used eligibility traces. (b) Opposed to the description given above, we did not need explorative steps because the environments can be very different and that diminished the need for exploration. (c) We did not decrease the value of α by time to keep adaptivity. (d) We

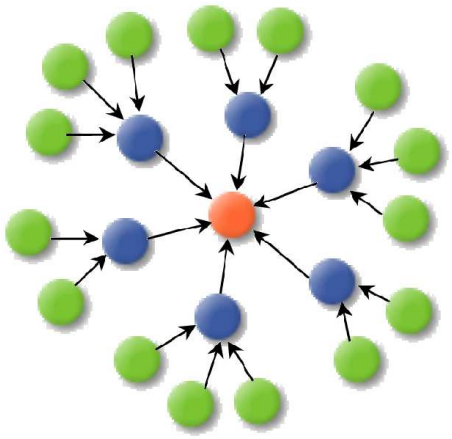


FIGURE 12. ***Context* of the document**
Document and its first and second 'neighbors'.

approximated the value function as follows

$$V(s) \approx \sum_{i=1}^{n} w_i \ \sigma(\text{SVM(i)}) \tag{6}$$

where the output of the i^{th} SVM (i.e., the i^{th} component of the output) is denoted by SVM(i), $\sigma(.)$ denotes the sigmoid function acting on the outputs of the SVM classifiers, w_i is the weight (or relevance) of the i^{th} classifier determined by upgrade Eq. 5. If the quality of the upgrade is measured by the mean square error of the estimations then the following approximate weight upgrade can be derived for the weights (see, e.g., [Sutton and Barto, 1998] for details):

$$\Delta w_i = \alpha * (r_{t+1} + \gamma * V(s_{t+1}) - V(s_t)) * \sigma(\text{SVM(i)}). \tag{7}$$

This upgrade — extended with eligibility traces [Sutton, 1988, Sutton and Barto, 1998] — was used in our RL engine.

6.3. **Features and learning to search.**

6.4. **Breadth first crawler.** A crawler is called *breadth first crawler*, if it first downloads the document of the launching site, continues by downloading the documents of all first neighbors of the launching site, then the documents of the neighboring sites of the first neighbor sites, i.e., the documents of the second neighbor sites, and so on.

6.4.1. *Context focused crawler.* A target document and its environment are illustrated in Fig. (12). . The goal is to locate the document by recognizing its environment first and then the document within. The CFC method [Diligenti et al., 2000] was modified slightly — in order to allow direct comparisons between the CFC method and the CFC method extended by RL value estimation — and the following procedure was applied. First, a set of irrelevant documents were collected. The k^{th} classifier was trained on (good) documents k-steps away from known target documents and on (bad) irrelevant documents. The classifier was trained to output a positive number ('yes') for

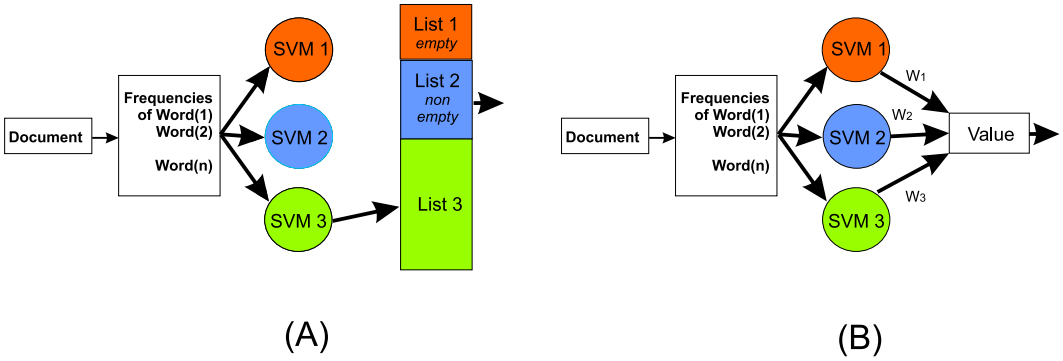


FIGURE 13. **SVM based document classifiers** (A) Classification of distance from document using SVM classifiers. The CFC method maintains a list of visited links ordered according to the SVM classification. One of the links belonging to the best non-empty classifier is visited next. (B) Value estimation based on SVM classifiers. Reinforcement learning is used to estimate the importance of the different classifiers during search.

good documents and to output a negative number ('no') for irrelevant documents. The outputs were scaled into the interval (0,1) by using the sigmoid function $\sigma(x) = (1 + exp(-\lambda x))^{-1}$. If the k^{th} classifiers output was close to 1 – according to its decision surface – there is a target document k-steps away from the actual site/document. If more than one classifier outputs 'yes' then only the best classifier is considered in CFC. Other outputs are neglected. The CFC idea with SVM classifiers is shown in Fig. (13)(a). CFC maintains a list of visited links ordered according to the SVM classification. One of the links belonging to the best non-empty classifier is visited next (this procedure is called backtracking).

The problem of the CFC method can be seen by considering that neighborhoods on the WWW may differ considerably. Even if the k^{th} classifier is the best possible such classifier for the whole web, it might provide poor results in some (possibly many) neighborhoods. For example, if there is a large number of connected documents all having the promise that there is a valuable document in their neighborhood – but there is, in fact, none – then the CFC crawler will download all invaluable documents before moving further. It is more efficient to learn which classifiers predict well and to move away from regions which have great but unfulfilled promises.

It has been suggested that classifiers could be retrained to keep adaptivity [Diligenti et al., 2000]. The retraining procedure, however, takes too long[6] and can be ambiguous if CFC is combined with backtracking. Moreover, retraining may require continuous supervisory monitoring and supervisory decisions. Instead of retraining, we suggest to determine the relevance of the classifiers during the search.

[6]Training may take on the order of a day or so on 700 MHz Pentium III according to our experiences.

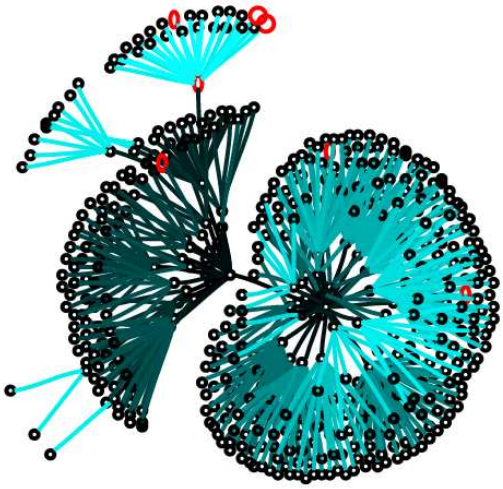


FIGURE 14. **Search pattern for breadth first crawler.** Search was launched from neutral site. A site is called neutral if there is very few target document in its environment. Diameter of open circles is proportional to the number of target documents downloaded. Edges are color coded. There are two extremes. Dark blue: Site was visited at the early stage during the search. Light blue: Recently visited site. (For further details, see text.)

6.4.2. *CFC and RL: Fast adaptation during search.* Reinforcement learning offers a solution here. If the prewired order of the classifiers is questionable then we could learn the correct ordering. There is nothing to loose here, provided that learning is fast. If prewiring is perfect then the fast learning procedure will not modify it. If the prewiring is imperfect then proper weights will be derived by the learning algorithm.

The outputs of the SVMs can be saved. These outputs can be used to estimate the value of a document at any instant. Value is estimated by estimating weights for each SVM and adding up the SVM outputs multiplied by these weights. In turn, one can compute value based ordering of the documents with minor computational effort and this reordering can be made at each step. This reordering of the documents replaces prewired ordering of the CFC method. The new architecture is shown in Fig. (13)(b).

6.5. **Results and discussion.** The CfP problem has been studied. Search pattern at the initial phase for the breadth first method is shown in Fig. (14).

Search patterns for the context focused crawler and the crawler using RL based value estimation are shown in Fig. (15) and Fig. (16). The launching site of these searches was a 'neutral site', a relatively large site containing few CfP documents (`http://www.inf.elte.hu`). We consider this type of launching important for web crawling: It simulates the case when mail lists are not available, traditional search engines are not satisfactory, and breadth

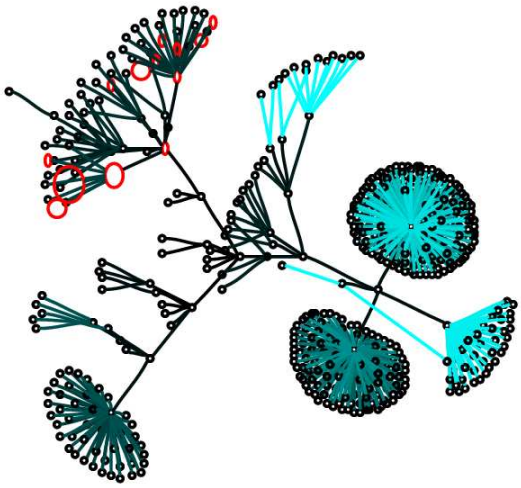


FIGURE 15. **Search pattern for context focused crawler.** Search was launched from neutral site. Diameter of open circles is proportional to the number of target documents downloaded. Edges are color coded. There are two extremes. Dark blue: Site was visited at the early stage during the search. Light blue: Recently visited site.

first search is inefficient. This particular site was chosen because breadth first search could find very few documents starting from this site.

'Scales' on Fig. (15) and Fig. (16) differ from each other and from that of Fig. (14). 'True surfed scale' would be reflected by normalizing to edge thickness. Radius of open circles is proportional to the number of downloaded target documents. The CFC is only somewhat better in the initial phase than the breadth first method. Longer search shows that CFC becomes considerably better than the breadth first method when search is launched from this neutral site.

Quantitative comparisons are shown in Fig. (17). According to the figure, upon downloading 20,000 documents, the number of hits were about 50, 200, and 1000 for the breadth first, the CFC and CFC based RL crawlers, respectively. These launches were conducted at about the same time. We shall demonstrate that the large difference between CFC and CFC based RL method is mostly due to the adaptive properties of the RL crawler.

There are two site types that have been investigated. The first site type is the neutral site that has been described before. The other site was a mail server on conferences. Also, for some examples there are runs separated by one month (March, 2001). A large number of summer conferences made announcements during this month.

First, let us examine the initial phase of the search. This initial phase of the search (the first 200 downloaded documents) is shown in Fig. (18). According to this figure downloading is very efficient from the mail server site in each occasion. The (non-adapting) CFC crawler utilizing averaged weights is superior

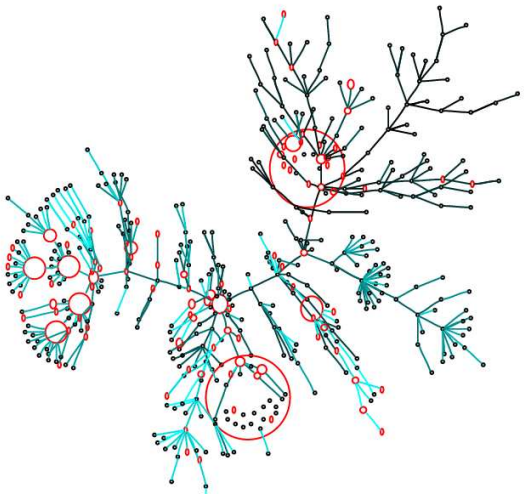


FIGURE 16. **Search pattern for CFC *and* reinforcement learning**
Search was launched from neutral site. Diameter of open circles is proportional to the number of target documents downloaded. Edges are color coded. There are two extremes. Dark blue: Site was visited at the early stage during the search. Light blue: Recently visited site.

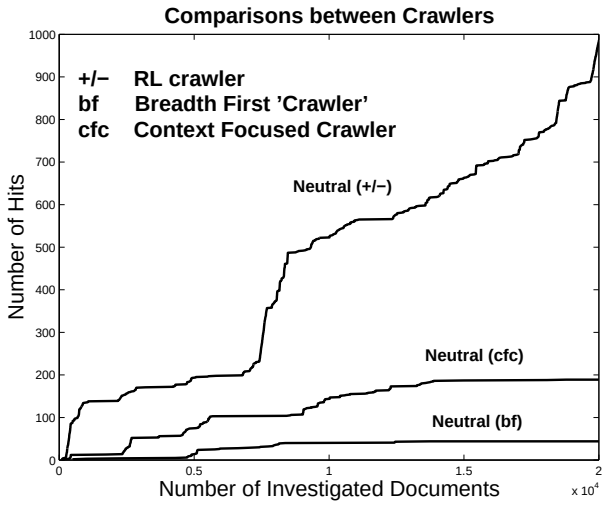


FIGURE 17. **Results of breadth first, CFC and CFC based RL methods.**

to all the other crawlers — almost all downloaded documents are hits. Close to this site there are many relevant documents and the 'breadth first crawler' is also efficient here. Nevertheless, the CFC crawler outperforms the breadth

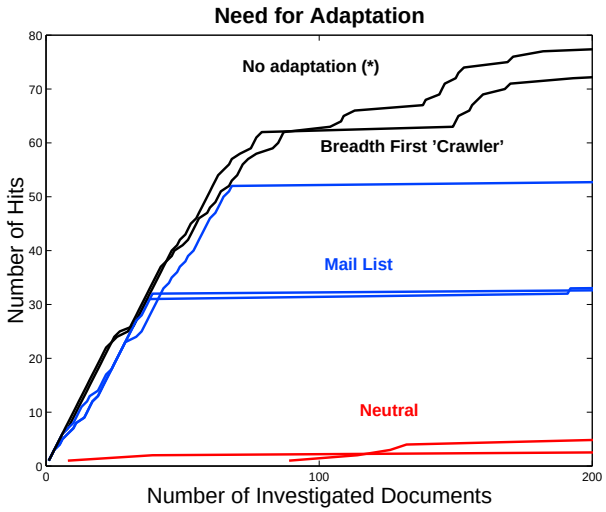


FIGURE 18. **Comparisons between 'neutral' and mail server sites in the initial phase.** Reward and punishment are given in the legend of the figure. Differences between similar types are due to differences in launching time. The largest time difference between similar types is one month. Neutral site (thin lines): `http://www.inf.elte.hu`. Mail list (thick lines): `http://www.newcastle.research.ec.org/cabernet/events/msg00043.html`. Search with 'no adaptation' (dotted line) was launched from mail list and used average weights from another search that was launched from the same place.

first crawler in this domain. Launching from neutral sites is inefficient at this early phase. Breadth first method finds no hit close to the neutral site (not shown in the figure).

Middle phase of the search is shown in Fig. (19). Performance in the middle phase is somewhat different. Sometimes, launches from the neutral site can find excellent regions. The CFC crawler is still competitive if launched from the mail server. Launches from the mail list spanning one month looked similar to each other; conference announcements barely modified the results.

Search results up to 20,000 documents are shown in Fig. (20). This graph contains results from a subset of the runs that we have executed. These runs were launched from different sites; the neutral site and the mail list, as well as a third type, the 'conference' site: `http:// www.informatik.uni-freiburg.de/index.en.html`. This latter is known to be involved in organizing conferences. Adapting RL crawlers collected a large number of documents from all site types and during the whole (one month) time region. The rate of collection was between 2%-5%. In contrast, although the collection rate is close to 100% for the CFC launched from the mail list site up to 200 downloads, the lack of adaptation prohibits this crawler to find new target documents in circa 17,000 downloads at later stages. Taken together:

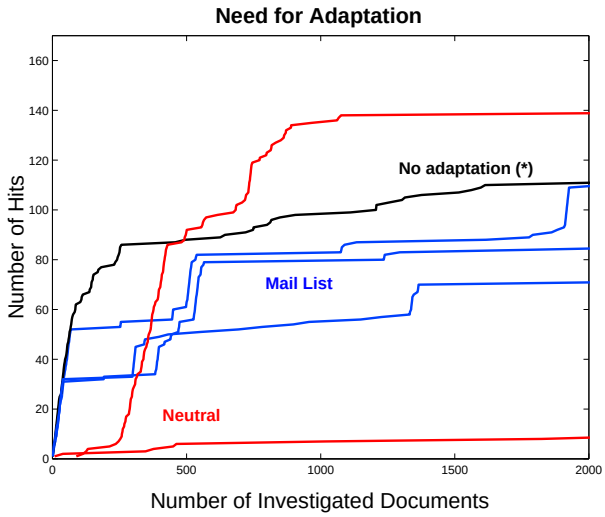


FIGURE 19. **Comparisons between 'neutral' and mail server sites up to 2000 documents.** Same conditions as in Fig. (18)

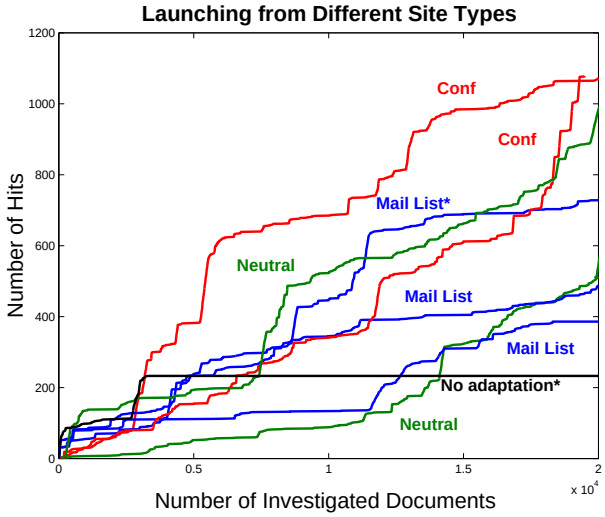


FIGURE 20. **Comparisons between different sites up to 20,000 documents.** Same conditions as in Figs. (18) and (19). Search with 'no adaptation' used average weights from another search that was launched from the same place (denoted by $*$)

(1) Identical launching conditions may give rise to very different results one month later.
(2) Starting from a neutral site can be as effective as starting from a mailing list for the adaptive RL crawler.

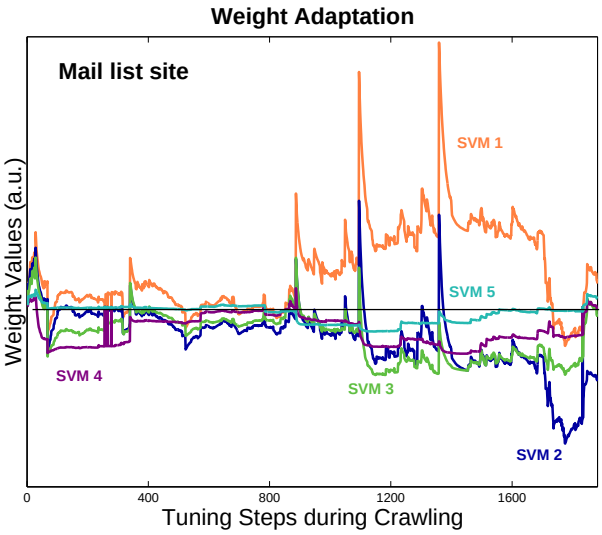


FIGURE 21. **Change of weights of SVMs upon downloads from mail site.**
Horizontal axis: Occasions when weights were trained.

(3) The lack of adaptation is a serious drawback even if the crawler is launched from a mailing list.

The importance of adaptation is also demonstrated by the RL weights assigned during search. These weights are shown in the following figures. Figure (21) depicts the weights belonging to the different SVMs launched from the mail list site. At the beginning of the search the weights are almost perfectly ordered; the largest weight is given to the SVM that predicts relevant document 'one step away' whereas the 4^{th} and the 5^{th} SVMs have the smallest weights. That is, RL 'pays attention' to the first SVM and pays less attention to the others. This order changes as time goes on. There are regions (at around tuning step number 1700 on the horizontal axis) where most attention is paid to the 5^{th} SVM and smaller attention is paid to the others. This means that the crawler will move away from the region. The order of importance changes again when a rich region is found; the importance of the first SVM recovers quickly and, in turn, crawling is dominated by the weight of the first SVM: The crawler 'stays' and downloads documents.

'Weight history' is different at the neutral site (Fig. (22)). Up to about 100 downloads very few relevant documents were found at this site. The value of weight of the 5^{th} SVM is slightly positive, whereas the values of the others are negative. The 1^{st} and the 2^{nd} SVMs are weighted the 'worst'; weights belonging to these classifiers are large negative numbers. At this site, the order of SVMs that were trained at around target documents is not appropriate. Situation changes quickly when a rich region is found. In such regions the 1^{st} SVM takes the lead. It is typical that the weight of the 5^{th} SVM is ranked second. That is, the adaptation concerns mostly whether the crawler should stay or if it should

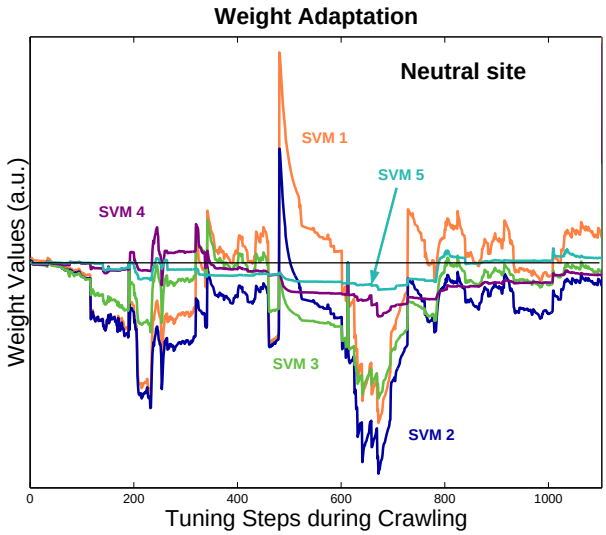


FIGURE 22. **Change of weights of SVMs in value estimation for 'neutral' site.**
Horizontal axis: Occasions when value estimation was erroneous and weights were trained.

move 'far away'. In turn, information contained by the 'context' is relevant and can be used to optimize the behavior of the crawler.

6.6. **Conclusions.** We have suggested a novel method for web search. The method makes use of combinations of two popular AI techniques, support vector machines (SVM) and reinforcement learning (RL). The method has a few adapting parameters that can be optimized during the search. This parameterization helps the crawler to adapt to different parts of the web. The outputs of the SVMs, together, formed a set of 'yardsticks' for the estimation of the distance from target documents. The value (the weight) of the different yardsticks may be very different at different neighborhoods. The point is that (i) RL is efficient with good features (the as k-step SVMs in this case), (ii) if there are just a few parameters for RL then these parameters can be trained quickly by rewarding for target documents. RL has many different formulations all of which could be applied here. Most promising are the approaches that can take into account (many) different criteria in the search objective [Fraser and Hauge, 1998, Gábor et al., 1998, Dubois et al., 2000]. Alas, RL methods are capable of extracting features [Thrun and Schwartz, 1995] that may complement the prewired SVM features.

References

[Abbott and Nelson, 2000] Abbott, L. and Nelson, S. (2000). Synaptic plasticity: taming the beast. *Nature Neuroscience*, 3:1178–1183.

[Albert and Barabási, 2002] Albert, R. and Barabási, A. (2002). Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47–91.

[Albert et al., 1999] Albert, R., Jeong, H., and A.-L. Barabási (1999). Diameter of the world-wide web. *Nature*, 401:130–131.

[Barabási and Albert, 1999] Barabási, A. L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286:509–512.

[Barabási et al., 2000] Barabási, A. L., Albert, R., and Jeong, H. (2000). Scale-free characteristics of random networks: The topology of the world wide web. *Physica A*, 281:69–77.

[Bell et al., 1997] Bell, C., Han, V., Sugawara, Y., and Grant, K. (1997). Synaptic plasticity in a cerebellum-like structure depends on temporal order. *Nature*, 387:278–281.

[Blum and Mitchell, 1998] Blum, A. and Mitchell, J. (1998). Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*. `http: //www.cs.cmu.edu /afs/cs.cmu.edu /project/theo-11 /www/wwkb /colt98_final.ps`.

[Bohland and Minai, 2001] Bohland, J. W. and Minai, A. A. (2001). Efficient associative memory using small-world architecture. *Neurocomputing*, 38-40:489–496.

[Chakrabarti et al., 1998] Chakrabarti, S., Dom, B., and Indyk, P. (1998). Enhanced hypertext categorization using hyperlinks. In Haas, L. M. and Tiwary, A., editors, *Proc. SIGMOD-98, ACM Int. Conf. on Management of Data*, pages 307–318, Seattle, US. ACM Press, New York, US. `http://www.almaden.ibm.com/cs/k53/irpapers/sigmod98.ps`.

[Chakrabarti et al., 1999a] Chakrabarti, S., Gibson, D., and McCurley, K. (1999a). Surfing the Web backwards. In *8th World Wide Web Conference*, Toronto, Canada. `http: //www8.org/w8-papers /5b-hypertext-media /surfing/`.

[Chakrabarti et al., 1999b] Chakrabarti, S., van der Berg, M., and Dom, B. (1999b). Focused crawling: a new approach to topic-specific Web resource discovery. In *8th International World Wide Web Conference (WWW8)*. `http: //www.cs.berkeley.edu / soumen/doc /www1999f/pdf /www1999f.pdf`.

[Cho et al., 1998] Cho, J., García-Molina, H., and Page, L. (1998). Efficient crawling through URL ordering. *Computer Networks and ISDN Systems*, 30(1–7):161–172. `http: //www-db.stanford.edu /pub /papers /efficient-crawling.ps`.

[Cristianini and Shawe-Taylor, 1999] Cristianini, N. and Shawe-Taylor, J. (1999). *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, UK.

[Dayan and Hinton, 1993] Dayan, P. and Hinton, G. E. (1993). Feudal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 5, pages 271–278, San Mateo, CA. Morgan Kaufmann.

[Dean and Henzinger, 1999] Dean, J. and Henzinger, M. (1999). Finding related pages in the world wide web. *WWW8 / Computer Networks*, 31(11-16):1467–1479. `http://www.research.compaq.com/SRC /personal/monika/papers /monika-www8-1.ps.gz`.

[Dietterich, 2000] Dietterich, T. G. (2000). Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303.

[Diligenti et al., 2000] Diligenti, M., Coetzee, F., Lawrence, S., Giles, C. L., and Gori, M. (2000). Focused crawling using context graphs. In *26th International Conference on Very Large Databases, VLDB 2000*, Cairo, Egypt. `http: //www.neci.nec.com /lawrence /papers /focus-vldb00 /focus-vldb00.ps.gz`.

[Dominich, 2000] Dominich, S. (2000). A unified mathematical definition of classical information retrieval. *Journal of the American Society for Information Science*, 51(7):614–624.

[Dubois et al., 2000] Dubois, D., Grabisch, M., Modave, F., and Prade, H. (2000). Relating decision under uncertainty and multicriteria decision making models. *International Journal of Intelligent Systems*, 15:967–979.

[Dumais et al., 1998] Dumais, S., Platt, J., Heckerman, D., and Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. In *7th International Conference on Information and Knowledge Management, ICIKM'98*. ACM. `http://robotics.stanford.edu/users/sahami/papers-dir/cikm98.pdf`.

[Edelman, 1987] Edelman, G. M. (1987). *Neural Darwinism: The theory of Neuronal Group Selection*. Basic Books, New York.

[Fraser and Hauge, 1998] Fraser, N. M. and Hauge, J. W. (1998). Multicriteria approval: application of approval voting concepts to mcdm problems. *Journal of Multi-Criteria Decision Analysis*, 7:263–273.

[Gábor et al., 1998] Gábor, Z., Kalmár, Z., and Szepesvári, C. (1998). Multi-criteria reinforcement learning. In *Proceedings of the Fifteenth International Conference on Machine Learning*.

[Guyon et al., 1992] Guyon, I., Vapnik, V. N., Boser, B. E., Bottou, L. Y., and Solla, S. A. (1992). Structural risk minimization for character recognition. In Moody, J. E., Hanson, S. J., and Lippmann, R. P., editors, *Advances in Neural Information Processing Systems 4*, pages 471–479, San Mateo, California. Morgan Kaufmann.

[Hebb, 1943] Hebb, D. (1943). *The Organization of Behavior: A Neuropsychological Theory*. Wiley, New York, USA.

[Hofmann, 2000] Hofmann, T. (2000). Learning the similarity of documents: An information-geometric approach to document retrieval and categorization. In *Neural Information Processing Systems*, volume 12, pages 914–920. MIT Press.

[i Cancho and Solé, 2001] i Cancho, R. F. and Solé, R. (2001). Optimization in complex networks. arXiv:arch-ive.cond-mat/0111222.

[Joachims, 2000] Joachims, T. (2000). Estimating the generalization performance of an SVM efficiently. In *Proc. 17th International Conf. on Machine Learning*, pages 431–438. Morgan Kaufmann, San Francisco, CA. `http:// www-ai.informatik.uni-dortmund.de /DOKUMENTE /joachims_99e.ps.gz`.

[Kabán and Girolami, 2000] Kabán, A. and Girolami, M. (2000). Clustering of text documents by skewness maximisation. In *2'nd International Workshop on Independent Component Analysis and Blind Source Separation, ICA'2000*, pages 435 – 440, Helsinki.

[Kaelbling, 1993] Kaelbling, L. (1993). Hierarchical learning in stochastic domains: Preliminary results. Proceedings of the Tenth International Conference on Machine Learning, San Mateo, CA. Morgan Kaufmann.

[Kandel and O'Dell, 1992] Kandel, E. R. and O'Dell, T. (1992). Are adult learning mechanisms also used for development? *Science*, 258:243–245.

[Kaski, 1998] Kaski, S. (1998). Dimensionality reduction by random mapping: Fast similarity computation for clustering. In *Proc. of Int. Joint Conf. on Neural Networks*, volume 1, pages 413–418. IEEE Service Center, Piscataway, NJ. `http://websom.hut.fi/websom/doc/ps/kaski98ijcnn.ps.gz`.

[Keerthi et al., 1999] Keerthi, S., Shevade, S., Bhattacharyya, C., and Murthy, K. (1999). Improvements to Platt's SMO Algorithm for SVM Classifier Design. Technical Report CD-99-14, Dept. of Mechanical and Production Engineering, National University of Singapore. `http: //guppy.mpe.nus.edu.sg /~mpessk/smo_mod.ps.gz`.

[Kleinberg, 1998] Kleinberg, J. (1998). Authoritative sources in a hyperlinked environment. In *9th ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677.

[Kolenda et al., 2000] Kolenda, T., Hansen, L., and Sigurdsson, S. (2000). Independent components in text. Ed.: M. Girolami, `http: //eivind.imm.dtu.dk /publications /1999 /kolenda.nips99.ps.gz`.

[Kolluri et al., 2000] Kolluri, R., Mittal, N., Ventakachalam, R., and Widjaja, N. (2000). Focussed crawling. `http: //www.cs.utexas.edu /users /ramki /datamining /fcrawl.ps.gz`.

[Korf, 1985] Korf, R. (1985). *Learning to solve problems by searching for macro-operators*. Pitman Publishers, Boston.

[Kulkarni et al., 2000] Kulkarni, R. G., Stough, R. R., and Haynes, K. E. (2000). Towards modeling of communities of practice (CoPs): A Hebbian learning approach to organizational learning. *Technological Forecasting and Social Change*, 64:71–83.

[Latora and Marchiori, 2001] Latora, V. and Marchiori, M. (2001). Efficient behavior of small-world networks. *Physical Review Letters*, 87(19).

[Lawrence, 2000] Lawrence, S. (2000). Context in web search. *IEEE Data Engineering Bulletin*, 23(3):25–32.

[Levy et al., 2001] Levy, N., Horn, D., Meilijson, I., and Ruppin, E. (2001). Distributed synchrony in a cell assembly of spiking neurons. *Neural Networks*, 14:815–824.

[Littman et al., 1995] Littman, M. L., Cassandra, A., and Kaelbling, L. P. (1995). Learning policies for partially observable environments: Scaling up. In Prieditis, A. and Russell, S., editors, *Proceedings of the Twelfth International Conference on Machine Learning*. Morgan Kaufmann.

[Magee and Johnston, 1997] Magee, J. and Johnston, D. (1997). A synaptically controlled, associative signal for hebbian plasticity in hippocampal neurons. *Science*, 275:209–213.

[Mahadevan and Connell, 1992] Mahadevan, S. and Connell, J. (1992). Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, 55:311–365.

[Marchioria and Latorac, 2000] Marchioria, M. and Latorac, V. (2000). Harmony in the small-world. *Physica A*, 285:539–546.

[Markram et al., 1997] Markram, H., Lubke, J., Frotscher, M., and Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science*, 215:213–215.

[Mataric, 1997] Mataric, M. (1997). Behavior-based control: Examples from navigation, learning, and group behavior. *J. of Experimental and Theoretical Artificial Intelligence*, 9:2–3.

[McCallum, 1996] McCallum, A. (1996). Bow: A toolkit for statistical language modelling, text retrieval, classification and clustering. `http://www.cs.cmu.edu/~mccallum/bow`.

[McCallum, 1999] McCallum, A. (1999). Multi-label text classification with a mixture model trained by em. `http: //www.cs.cmu.edu/~mccallum /papers /multilabel-nips99s.ps.gz`.

[McCallum et al., 1999] McCallum, A., Nigam, K., Rennie, J., and Seymore, K. (1999). Building domain-specific search engines with machine learning techniques. In *AAAI-99 Spring Symposium on Intelligent Agents in Cyberspace*. `http: //www.cs.cmu.edu/~mccallum /papers /cora-aaaiss98.ps`.

[McCallum et al., 2000] McCallum, A., Nigam, K., Rennie, J., and Seymore, K. (2000). Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163. `http: //www.cs.cmu.edu /afs /cs /user /kseymore /html /papers / cora-journal.ps.gz`.

[Minton, 1988] Minton, S. (1988). *Learning search control knowledge: An explanation based approach*. Kluwer Academic.

[Mitchell, 1999] Mitchell, T. (1999). The role of unlabeled data in supervised learning. In *Proceedings of the Sixth International Colloquium on Cognitive Science*. `http://www.ri.cmu.edu/pub_files/pub1/mitchell_tom_1999_2/mitchell_tom_1999_2.pdf`.

[Mukherjea, 2000] Mukherjea, S. (2000). WTMS: A system for collecting and analyzing topic-specific web information. In *9th World Wide Web Conference*. `http://www9.org/w9cdrom/293/293.html`.

[Murdock and Goel, 1999] Murdock, J. and Goel, A. (1999). Towards adaptive web agents. In *14th IEEE International Conference on Automated Software Engineering*. `http://www.cc.gatech.edu/morale /papers/ase99.ps`.

[Nigam et al., 2000] Nigam, K., Mccallum, A., Thrun, S., and Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134. `http://www.cs.cmu.edu/~knigam/papers/emcat-mlj99.ps.gz`.

[Oja, 1982] Oja, E. (1982). A simplified neuron model as a principal component analyzer. *J. Math. Biol.*, 15:267–273.

[Palotai et al., 2002] Palotai, Z., Szirtes, G., and Lőrincz, A. (2002). Meta level analysis of hebbian evolving networks. Technical Report NIPG-ELU-21-04-2002, ELTE. http://people.inf.elte.hu/lorincz/Files/NIPG-ELU-21-04-2002.pdf.

[Rennie et al., 1999] Rennie, J., Nigam, K., and McCallum, A. (1999). Using reinforcement learning to spider the web efficiently. In *16th International Conference on Machine Learning*, pages 335–343. Morgan Kaufmann, San Francisco, CA. `http: //www.cs.cmu.edu /~mccallum /papers /rlspider-icml99s.ps.gz`.

[Rummery and Niranjan, 1996] Rummery, G. A. and Niranjan, M. (1996). On-line q-learning using connectionist systems. Technical report, Cambridge University Engineering Department.

[Schmidhuber, 1991] Schmidhuber, J. (1991). Neural sequence chunkers. Technical Report FKI-148-91, Technische Universität München, München, Germany.

[Smola and Schölkopf, 1998] Smola, A. J. and Schölkopf, B. (1998). A tutorial on Support Vector Regression. Technical Report NC2-TR-1998-030, NeuroCOLT2.

[Sutton, 1988] Sutton, R. (1988). Learning to predict by the method of temporal differences. *Machine Learning*, 3:9–44.

[Sutton and Barto, 1998] Sutton, R. and Barto, A. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

[Thrun and Schwartz, 1995] Thrun, T. and Schwartz, A. (1995). Finding structure in reinforcement learning. volume 7 of *Advances in Neural Information Processing Systems*, San Mateo, CA. Morgan Kaufmann.

[Vapnik, 1995] Vapnik, V. (1995). *The nature of statistical learning theory*. Springer-Verlag, New York.

[Vinokourov and Girolami, 2000a] Vinokourov, A. and Girolami, M. (2000a). A probabilistic hierarchical clustering method for organizing collections of text documents. In *15th Int. Conf. on Pattern Recognition*, volume 2, pages 182–185. IEEE Computer Press. `http://cis.paisley.ac.uk/vino-ci0/vinokourov_ICPR00.ps`.

[Vinokourov and Girolami, 2000b] Vinokourov, A. and Girolami, M. (2000b). A probabilistic hierarchical clustering method for organizing collections of text documents. Technical Report ISSN 1461-6122, University of Paisley, Department of Computing and Information Systems, Paisley, PA1 2BE, UK. `http://cis.paisley.ac.uk/vino-ci0/hplsa_kais.ps`.

[Watkins, 1989] Watkins, C. (1989). *Learning from Delayed Rewards*. Phd thesis, King's College, Cambridge, UK.

[Watts and Strogatz, 1998] Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of small-world networks. *Nature*, 393:440–442.

[Zhang et al., 1998] Zhang, L. I., Tao, H., Holt, C., Harris, W. A., and Poo, M. (1998). A critical window for cooperation and competition among developing retiotectal synapses. *Nature*, 395:37–44.

1117 Budapest, Pázmány Péter sétány 1/C.

E-mail address: alorincz@axelero.hu, titkarsag@njszt.hu

REPORT: MODELS OF DISTRIBUTED COMPUTATION

REPORT EOARD-NIPG-ELU-18-SEPT-2002
CONTRACT: EOARD F61775-01-WE023

ADMINISTRATIVE CONTACT: ISTVÁN ALFÖLDI, NJSZT

PRINCIPAL INVESTIGATOR: DR. HABIL. ANDRÁS LŐRINCZ

Abstract. The first task is to discover and characterize the substrate of distributed computation, which is the Internet in our case. The second task is to develop methods (models) for collaborative interactions.

According to recent discoveries, the Internet has a special structure, called scale-free small world. Up to some extent, the structure of Internet is described by different models. In a parallel project we have introduced a novel model, the HebbNet, which represent competitive-collaborative activities in a more sensible way than existing other models of the Internet. The particular properties of the Internet and HebbNets are described here. Results are updated as compared to Report No. 3.

Second, we have downloaded a portion of the Internet. The working and the gain of efficiency using competitive-collaborative model were studied on this downloaded part. A model for injecting novel information into the downloaded portion of the Internet was developed. Computer simulation demonstrate the idea: (i) competition gives rise to a division of work, the division of searched portions of the reinforced robotic Internet agents, (ii) the division of work gives rise to specialization and increased efficiency for the specialized robotic agents.

Third, we have executed experiments on the Internet using our competition based collaborative system. The experiments are in full support to our previous efforts: Crawlers make compartments and work more efficiently by searching only these compartments. Collaboration occurs by the division of work. Two summarizing figures are shown at the beginning of this report.

The conclusion section (Section 5) contains information about necessary future steps, which are not covered by the present project and are being at basic research phase at this very moment.

The Report is extended by Appendices:

(1) Description of concepts and methods, like reinforcement learning, text classification, SVMs are appended to the Report are provided in the Appendix

(2) A large body of experiments on HebbNets are provided in an accompanying technical report, titled: Meta level analysis of Hebbian evolving networks `http://people.inf.elte.hu/lorincz/Files/NIPG-ELU-21-04-2002.pdf`

Date: 18th September 2002.

SUMMARY IN TWO FIGURES

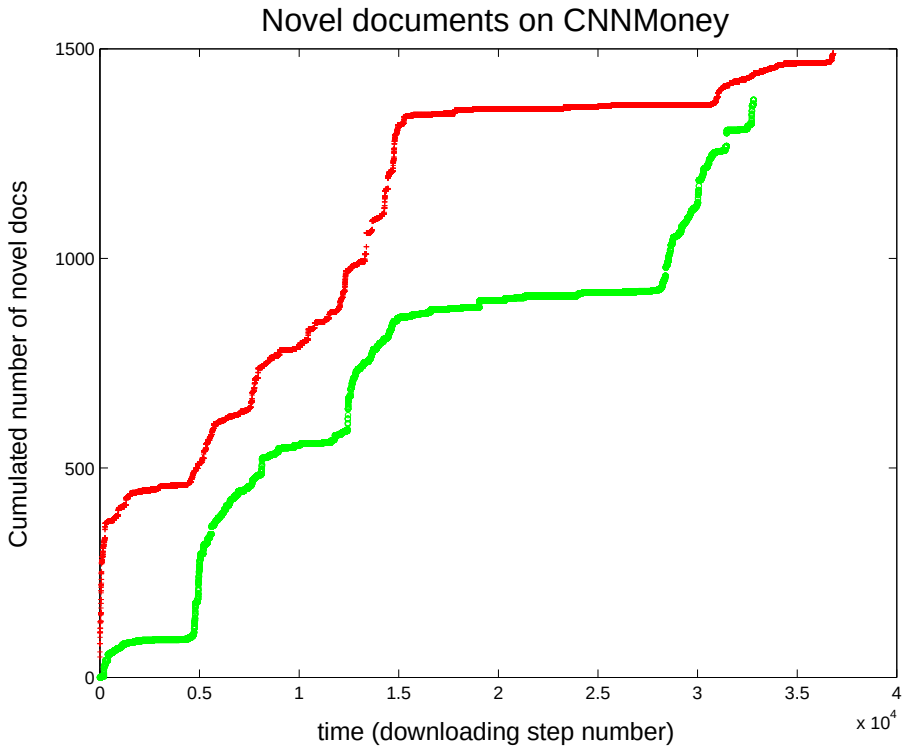


FIGURE 1. **Novel documents found by two competing crawlers**

Two competing crawlers are searching the **CNN Money**site for novel information (information not yet known to the crawlers). During this search only the crawler which delivers the information first, is reinforced. The red crawler starts better, the green crawler is following it in most of the time (there is no reward to this crawler). Suddenly, at around 5000th download the green crawler finds other routes to follow. This new routes turn out to be rewarding, the efficiency of the green crawler is about the same as that of the red crawler. Both crawlers are inefficient between downloads 15,000 and 30,000, this is the *weekend*. After the weekend, the novel routes found by the green crawler are more efficient. The number of novel documents is much larger at the start of the search when all documents are novel for the competing crawlers. This can be seen by considering the slope of the curves at the beginning of the search. For more details, see report.

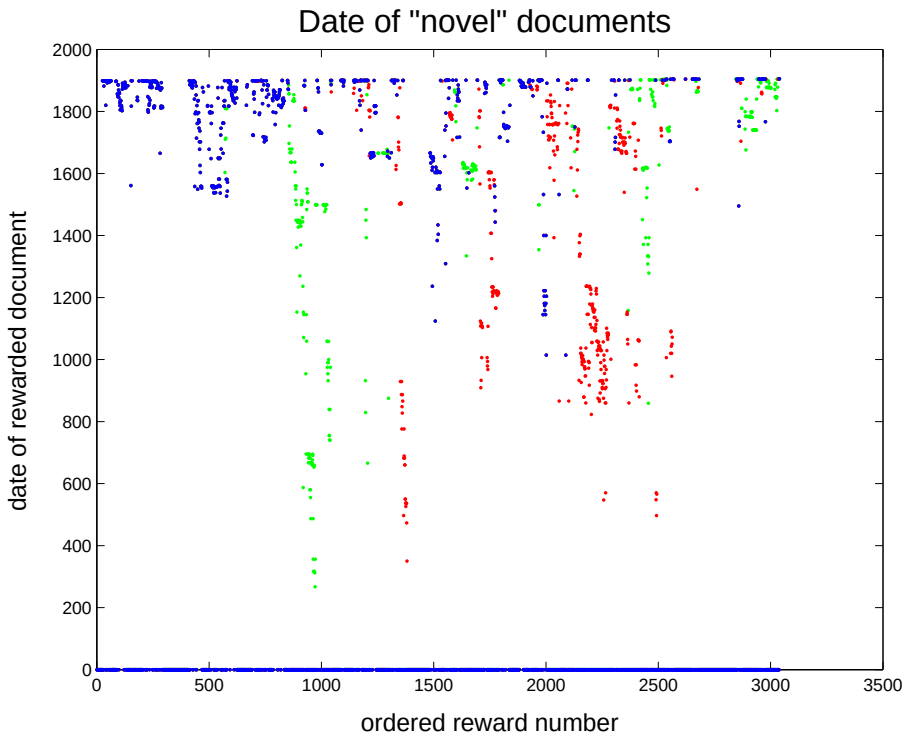


FIGURE 2. **Age of documents found by competing crawlers**

Two competing crawlers are searching the **CNN Money**site for novel information (information not yet known to the crawlers). In about half the cases we could establish the date of the document. Red (green) dots: document downloaded by the red (green) crawler. Blue dots: documents downloaded by both crawlers. After four days, there are still old documents find by the crawlers, but the relative rate of novel documents is increasing strongly after the start of the new week (at around reward number 2,700).

The number of novel documents is much larger at the start of the search when all documents are novel for the competing crawlers. Note that 365 units on the vertical scale correspond to a year. The crawlers collect the daily documents,too, as it can is shown by the slight slope of maxima (the convex hull of the maxima) of the points. For more details, see report.

Contents

List of Figures

1. Introduction

Cooperation requires some knowledge about the 'arena' of cooperation. In our case, this arena is the Internet. Interestingly, recent discoveries demonstrated that a large variety of networks, such as social networks, scientific collaborative networks, hardware networks at the level of proxies, Internet connections, html links, etc., share the same structure. Moreover, the suspicion has been increasing that all (or at least a large number of) 'sustained' systems share the same common connectivity property, provided that the underlying concept of connectivity is discovered. Two concepts have to be explained here:

- the concept of a sustained system and
- the word connectivity.

A system is called sustained (i) if it is not closed from the environment and (ii) if it is receiving energy from the environment. A cell of the body, the earth of the solar system, the research group of the University are such examples.

Connectivity is a loosely defined word to describe interaction amongst collaborative-competitive units. It is a loose term, because in its crudest form is not concerned with the vehemence, but only with the existence of the interaction. The usage of this word is as follows: If two units are engaged in kind of interaction then *they are connected*. Interaction can be directed or could be mutual. For example, (mostly) directed interaction occurs between the wind and the sailboat. Interactions are never fully directed; the sailboat – although up to a minor extent, but – influences the local properties of the wind. An example for a more balanced interaction is the teacher student relation, whereas an almost fully balanced may occur for some ants. Given the type(s) of interaction, one ends up with a directed or with an undirected network of interacting units.

Report organization.

(1) First, novel concepts of the Internet are provided. This part of the report contains the description of our model of the Internet, which we call HebbNets.
(2) The second part contains the results of our competitive-collaborative model. Specialization of Internet agents, division of work and the increase of efficiency are demonstrated on a downloaded part of the Internet.
(3) Discussions can be found and conclusions are drawn in Sections 4 and 5, respectively.
(4) Details about intelligent crawlers are contained by the Appendix.
(5) A large body of experiments on HebbNets are provided in an accompanying technical report, titled: Meta level analysis of Hebbian evolving networks `http://people.inf.elte.hu/lorincz/Files/NIPG-ELU-21-04-2002.pdf`

2. Internet and other networks

2.1. **Short overview.** Interactive systems with network structure have recently become a fascinating area of research interest. Dynamic systems that govern the formation of these networks are of central importance because of the intriguing similarities among many biological, social, and information processing networks. The original model of Watts and Strogatz [D. J. Watts and S. H. Strogatz, Nature **393**, 440, (1998)] explore random restructuring of links among a finite number of 'nodes'. Other works have dealt with growing structures and optimization of link structure of finite systems. In this paper we present and study the 'HebbNets', networks in which structural changes are governed by Hebbian learning rules. We find that Hebbian learning is also capable to develop all kinds of network structures, including small-world and scale-free networks. Our results may support the idea of Edelman [G. M. Edelman: Neural Darwinism, New York, Basic Book (1987)] that the development of central nervous system may have evolutionary components.

2.2. **Introduction.** The last few years have witnessed the evolution of novel and efficient ways of describing complex interactive systems (CISs). The novel

description of a CIS is based on graphs with nodes and (directed) edges, representing constituents of the system and the interactions among them. Classification of CISs is based on the statistical properties of the network. Similar network structures may be found in many different fields. Both these systems and the corresponding dynamical models which define the formation of these networks may be of fundamental importance to understand the behaviors of CISs. The interest in CISs is boosted by the intriguing similarities of biological, social, and information processing networks [Watts and Strogatz, 1998, Kleinberg, 1998, Albert et al., 1999, Barabási and Albert, 1999, Barabási et al., 2000, Marchioria and Latorac, 2000, Latora and Marchiori, 2001, Bohland and Minai, 2001, i Cancho and Solé, 2001, Albert and Barabási, 2002]. The original model of the the World Wide Web (WWW) by Watts and Strogatz [Watts and Strogatz, 1998] explored random restructuring of the links among a finite number of 'nodes'. Barabási and his colleagues introduced the concept of *preferential attachment* to provide a more sophisticated model of WWW [Albert et al., 1999, Barabási and Albert, 1999]. The idea has been extended to other types of networks [Barabási et al., 2000]. Another approach deals with optimization of link structure of finite systems [i Cancho and Solé, 2001].

Probably the most complex network is inside us: the most exciting properties of our brain have a lot to do with the special connection system among its units. Although our knowledge on the building blocks is increasing, we are still far from a complete understanding of the structure and research of the central nervous system (CNS) is primarily targeting at these questions. At the same time, many similar questions on the architecture of the WWW arise. The recognition of the parallel nature resulted in the intuitive idea to highlight the similarity between these descent phenomena. From one hand, it has been suggested to use the mutual activity correlation (that is the original form of Hebbian learning) in modeling organizational learning [Kulkarni et al., 2000]. On the other hand, similar structural characteristics of the web and the single completely described nervous system of the nematode *C. elegans* has been reported [Watts and Strogatz, 1998]. In this paper we investigate the question whether an evolving network, governed by Hebbian rule, has the same or similar properties as found by studying the web or social networks. The question is of central importance: in seeking the answer we hope to find general underlying principles, which give rise to small-world like structures in cooperative-competitive systems.

It may be important to note here that concepts of Hebbian learning have undergone revolutionary changes in the last few years. The original suggestion of Hebb [Hebb, 1943] has been modified by recent findings [Markram et al., 1997, Magee and Johnston, 1997, Bell et al., 1997]. (For a review, see, e.g. the work of Abbott and Nelson [Abbott and Nelson, 2000]). The novel concept is called spike-time dependent synaptic plasticity (STDP). The underlying mathematical concepts could correspond to linear and non-linear versions of principal component analysis [Oja, 1982, Abbott and Nelson, 2000].

2.3. **Description of HebbNet.** We intended to construct a model network (HebbNet in short) in which the structural changes are governed by Hebbian rules and the interaction with the environment and all the interacting elements

of the network are as simple as possible. We assume that the network is *sustained* by inputs with no spatio-temporal structure; the input is random noise. Our models consist of N number of simplified integrate-and-fire like 'neurons' or nodes. The dynamics of the internal activity is written as

$$\frac{\Delta a_i}{\Delta t} = \sum_j w_{ij} a_j^s + x_i^{(ext)}, \tag{1}$$

for $i = 1, 2, \ldots, N$. (N was 200 in our simulations.) Variable $x^{(ext)} \in (0,1)^N$ denotes the randomly generated input from the environment, a_i is the internal activity of neuron i, w_{ij} is ij^{th} element of matrix $\mathbf{W}$, i.e., the connection strength from neuron j to neuron i. If $\Delta t = 1$ then we have a discrete-time network and each parameter has a time index, or if Δt is infinitesimally small then Eq. 1 becomes a set of coupled differential equations. The neuron j outputs a spike (neuron j fires) when a_j exceeds a certain level, the threshold parameter θ. Spiking means that the output of the neuron a_j^s (superscript s stands for 'spiking') is set to 1. Otherwise, $a_j^s = 0$. Amount of excitation received by neuron i from neuron j is $w_{ij} a_j^s$ when neuron j fires. After firing, a_j is set to zero at the next time step. For continuous case a_j is set to zero after a very small time interval. Equation 1 describes the simplest form of 'integrate–and–fire' network models which is still plausible from a neurobiological point of view. No temporal integration occurs for the discrete case provided that the left hand side of Eq. 1 is replaced by a_i^+ where superscript + denotes time shifting. In this limiting case, and if the threshold is high enough, 'binary neurons' emerge. This model resembles the original model of McCullough and Pitts [McCullough and Pitts, 1943].

We examined the effect of local activity threshold and global activity constraint (selection of a given percent of nodes with the highest activity). The former one is more realistic biologically, while the latter one is more convenient: in this way the ratio of active units is always known and fixed. For these two cases, computer simulations showed negligible differences. Synaptic strengths were modified as follows:

$$\frac{\Delta w_{ij}}{\Delta t} = \sum_{(t_i, t_j)} K(t_j - t_i) a_i^{t_i, s} a_j^{t_j, s}, \tag{2}$$

where K is a kernel function which defines the influence of the temporal activity correlation on synaptic efficacy and $\Delta w_{ij}/\Delta t$ may be taken over discrete or over infinitesimally small time intervals. Possible examples are depicted in Fig. 3. The kernel is a function of the time differences. When the input is made of noise, as in our studies, only the ratio of the positive (strengthening) and the negative (weakening) parts of the kernel function should count. This is the result of the lack of temporal correlations in the input. Temporal grouping and reshaping of the kernel would not modify our results as long as the said ratio is kept constant. In turn, our results concern both types of kernels depicted in Fig. 3.

In the first place, we have been interested in the emerging local and global connectivity structure of $\mathbf{W}$. Instead of using global structural property (L, characteristic path length which is the average number of edges on the shortest path) and the clustering coefficient (C) proposed by Watts and Strogatz

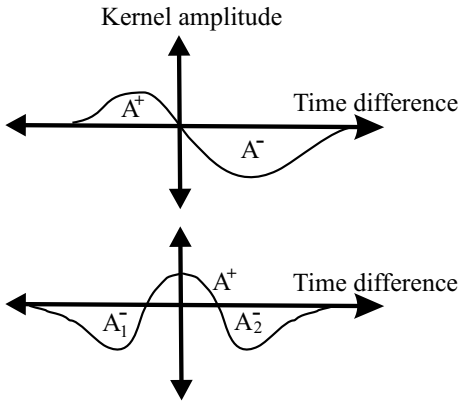


FIGURE 3. **Kernel functions**
Two temporal kernels as a function of time difference between spiking time of neuron i and j $(t_i - t_j)$. Relevant parameter of the shape for noise-sustained systems is the ratio (r_{A^+/A^-}) of the areas/sums of positive and negative parts/components of the kernel, A^+ and A^-, respectively $(r_{A^+/A^-} = A^+/A^-)$.

[Watts and Strogatz, 1998] we applied the so called connectivity length measure based on the concept of *network efficiency* [Latora and Marchiori, 2001]. This single measure is more appropriate for weighted networks [Marchioria and Latorac, 2000], equally well applicable for describing global and local properties and offers a unified theoretical background to characterize our system. According to the definition [Marchioria and Latorac, 2000, Bohland and Minai, 2001], local efficiency between nodes i and j in a weighted network with connectivity matrix $\mathbf{W}$ is $\epsilon_{ij} = 1/d_{ij}$, where $d_{ij} = \min_{n,\, k_1,\ldots k_n} \left(1/w_{ij}, 1/w_{ik_1} + \ldots + 1/w_{k_{n-1}k_n} + 1/w_{k_nj}\right)$ $(k_m \in (1, 2, \ldots N)$ for every $1 \leq m < N-1$ and $1 < n \leq N)$. For graphs with connection strengths of values 0 or 1, d_{ij} corresponds to the *shortest distance* between nodes i and j. The average of these values $(E[d_{ij}] = \frac{1}{N(N-1)} \sum_{i \neq j} \epsilon_{ij})$ characterizes the efficiency of the whole network. The local harmonic mean *distance* for node i is defined as

$$D_h(i) = \frac{n^{(i)}}{\sum_{j:w_{ij}>0} \epsilon_{ij}}, \tag{3}$$

where $n^{(i)}$ is the number of neurons around neuron i with $w_{ij} > 0$. In terms of efficiency, this inverse of this value describes how good the local communication is amongst the first neighbors of node i with node i removed. It is a measure of the fault tolerance of the system. The mean *global distance* in the network is defined by the following quantity:

$$D_h = \frac{N(N-1)}{\sum_{i,j} \epsilon_{ij}}. \tag{4}$$

Global distance provides a measure for the *size* (or the diameter) of the network, which influences the average time of information transfer. According to [Marchioria and Latorac, 2000, Bohland and Minai, 2001] local harmonic mean distance measure behaves like $1/C$ (inverse of the clustering coefficient),

whereas the global value corresponds to L. It can be shown that L is a good approximation of D_h (or $1/L$ for the global efficiency) under certain conditions [Bohland and Minai, 2001].

These connectivity length measures allowed us to study the emerging network structures as the function of the following parameters: (i) the magnitude of the external excitation (defined by the average percentage of neurons receiving excitation from the environment and (ii) the strengthening–weakening area ratio of the kernel, K. The binary neuron model was also investigated. Figures 4 and 5 summarize our findings in different parameter regions. The figure displays the appearance of scale free nets as a function of the excitation level and r_{A^+/A^-}. The length of the scale-free regions was determined by first plotting the distribution of the sum of the weights of outgoing connections (averaged over 10000 samples taken from 20 networks) for every parameter set studied. Results were depicted on loglog plot. Supposing a power-law distribution ($P(k^*) \approx k^{*-\gamma} e^{-k^*/\xi}$, where k^* denotes the discretized values of the connection strength), a linear fitting was made to approximate γ. The width of the scale-free region was estimated by the length of the region with power-law distribution relative to the full length covered on the log scale. Maximum error of the linear fit was set to 10^{-3} STD. That is, for 100 discretization points, the width of a region spreading an order of magnitude on the loglog plot is equal to 0.5.

Fig. 6 displays the emerging connections of a HebbNet for two different parameter sets. We compared the resulting HebbNet structures with a random net, in which the same weights of the dynamic network have been randomly assigned to different node pairs. The two inlets show the HebbNet connection matrices. While inlet (A) belonging to case (c) in Fig. 4 resembles a random connection matrix, inlet (B) belonging to case (d) in Fig. 4 represents a sparse structure. (Note that most elements are not zero, but very small.)

Fig. 6 highlights clearly the emerging small-world properties, i.e., small local connectivity values (high clustering coefficients) for case (d). Although the global connectivity length was almost the same for all HebbNets and their corresponding random nets, local distances are much smaller in case (d). That is, connectivity structure is sparse but information flow is still fault tolerant and efficient.

The robustness of the network to the external excitation is illustrated on the next figure. By increasing the excitation level, the average local connectivity length of the random net is drastically increasing, whereas the efficiency of the small-world network does not change too much in the same region. For the network with parameters $r_{A^+/A^-} = 0.1$ (Fig. 7(A)), there is a sharp cut-off around excittion level 0.55, where local distances suddenly drop, due to the high ratio of excitation. Qualitatively similar behavior can be seen for $r_{A^+/A^-} = 0.6$ (Fig. 7(B)), but the cut-off is around $r_{ex} = 0.9$.

For networks with significant interaction we have experienced a convergence of the exponent of the power-law distribution to -1. The width of the scale-free region was relatively broad (see, Fig. 8).

In summary, we have demonstrated that small-world architectures with scale-free domains may emerge in sustained networks under STDP Hebbian learning rule without any other specific constraints on the evolution of the net.

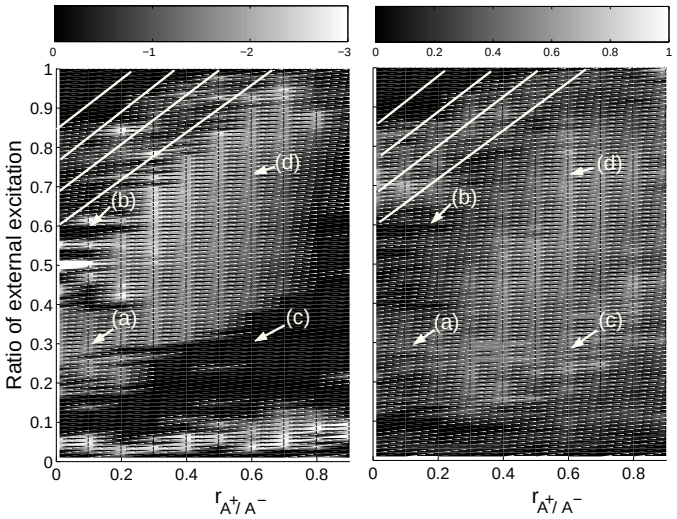


FIGURE 4. **Scale-free region with negligible interaction Left:** *exponent of the power law*, **right:** *relative percentage of the power-law domain* as a function of r_{A^+/A^-} and r_{ex} (the ratio of excited neurons). Contribution of other neurons to the neuronal inputs is negligibly small. Difference between binary and integrate-and-fire neurons disappears in this limiting case. Results are averaged over 20 runs, all sampled 50 times, $\theta = 0.5$. Stripes denote unstable region: components of matrix $\mathbf{W}$ may vanish. Log-log plots corresponding to points (a)–(d) are shown in Fig. 5. Power-law with negative (positive) exponent: cases (a) and (d) (case (c)). Positive exponents are thresholded to zero on the figure.

The role of noise in the central nervous system [Ferster, 1996, Miller and Troyer, 2002] is unclear. The existence of such 'HebbNets' may support the speculative view of Kandel et al. [Kandel and O'Dell, 1992] that structural development and learning plasticity in CNS may have a common basis. According to our results, evolution and plasticity of the networks may be maintained by noise randomly generated within the CNS. We conjecture that the sustained nature of noise and the competition imposed by small r_{A^+/A^-} values are the two relevant components of plasticity and learning. It might be equally important that exponents of HebbNets with significant interaction amongst neurons are similar in a broad range of parameters.

As far as other evolving networks are considered, the profound implication of our result is that local (Hebbian) learning rules may be sufficient to form and maintain an efficient network in terms of information flow. This feature differs from existing models, such as the model on preferential attachment [Barabási and Albert, 1999], the global optimization scheme [i Cancho and Solé, 2001], and also from the original Watts and Strogatz model [Watts and Strogatz, 1998].

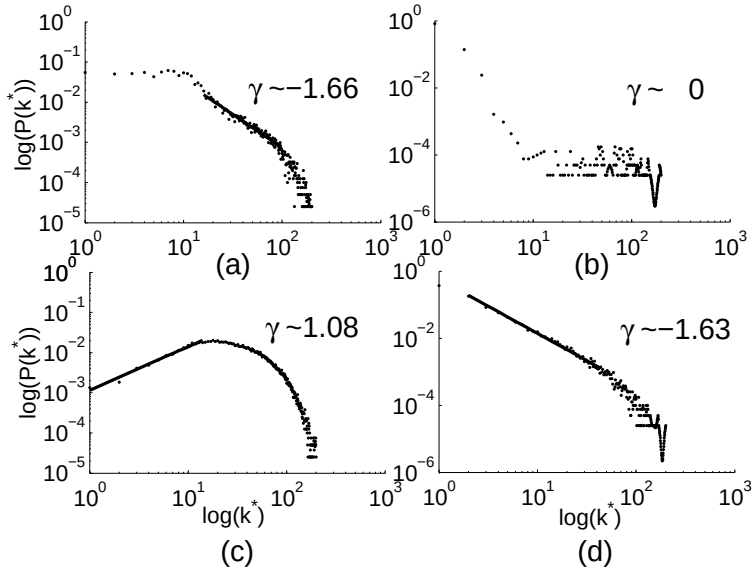


FIGURE 5. **Log-log plots for different parameters**
The four diagrams display typical distributions ($P(k^*)$) for parameters shown in Fig. 4 by (a), (b), (c) and (d). Cases (a) and (d) are arbitrary examples from the power law region.

3. Competition, which Results in Collaboration

Rapid development of Internet technologies increases the use of this unique medium for collaboration. While interoperability is a main focus of these collaborative efforts, privacy protection, along with reputation management is in demand. Recently several works have emerged that focus on these problems. However, works on providing confidentiality and reputation are limited. For example, most of the works focus on a particular problem and problem domain, say reputation management for e-commerce applications. Considerably less work has been done to accommodate novel requirements, such as collaborative access requirements and developing protocols that allow anonymity while support accountability. Furthermore, most of the existing works were developed with special application in mind and are not generally applicable.

3.1. Consequences of Novel Findings about the Internet.

We assume - without limiting generality - that the goal of the Internet is to publish. Publication is seen here as a general way of accomplishing a task and making it available for others in a understandable (readable) form. In turn, the goal of collaboration on the Internet can be seen as editing to reach the goal in an efficient manner. In our approach we take into account the general features discovered over the last few years about publishing and collaboration. These features can be best described in terms of networks. Take an author (author A) and his/her co-authors. Say, author B is a co-author of author A. This is a minimal network. The 'distance' between the two co-authors

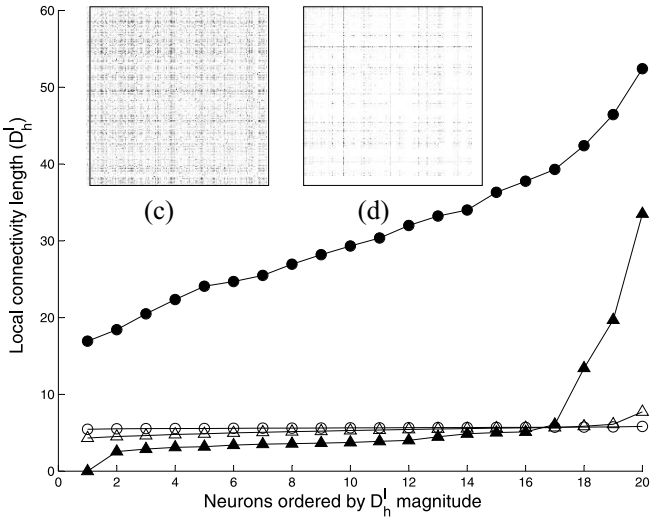


FIGURE 6. **Harmonic mean distances**
Local harmonic mean distances in ascending order are shown. For better visualization not all data points are marked and the points are connected with a solid line. Lines with upward triangle markers: STDP learning. Lines with circles: same but randomly redistributed weights. Line with empty (solid) markers: HebbNet of case (c) (case (d)). Global harmonic mean distances for the original and for the randomized networks in case (c) of Fig. 4 (case (d) of Fig. 4) are about the same $D_h \approx D_h^r \approx 5.5$ ($D_h \approx D_h^r \approx 10$). The two inlets show the resulting connection matrices.

is 1. Alternatively, there is a link between authors A and B. B has co-authors, too. Author C is a co-author author B, whereas author C is not a co-author of author A. There are links between authors A and B and authors B and C. However, there is no link between author A and C. The distance between authors A and C is equal 2. A broad range of social networks shows general features when formulated this way [Watts and Strogatz, 1998, Barabási and Albert, 1999, i Cancho and Solé, 2001]. The general features are called 'small-world phenomenon' and 'scale-free networks'. For example, similar networks have been discovered

(1) amongst Hollywood actors (collaboration is playing in the same movie),
(2) authorship in scientific communications,
(3) social networks ('collaboration' means knowing each other), and
(4) in the so called Erdos-point in mathematics.

One should design security architecture for originally non-hierarchical networks. Further, one may allow the development of a hierarchy based on the efficiency of the collaborating partners. An example is the Erdos point, where there is a clear hierarchical structure with Paul Erdos, the famous mathematician on the

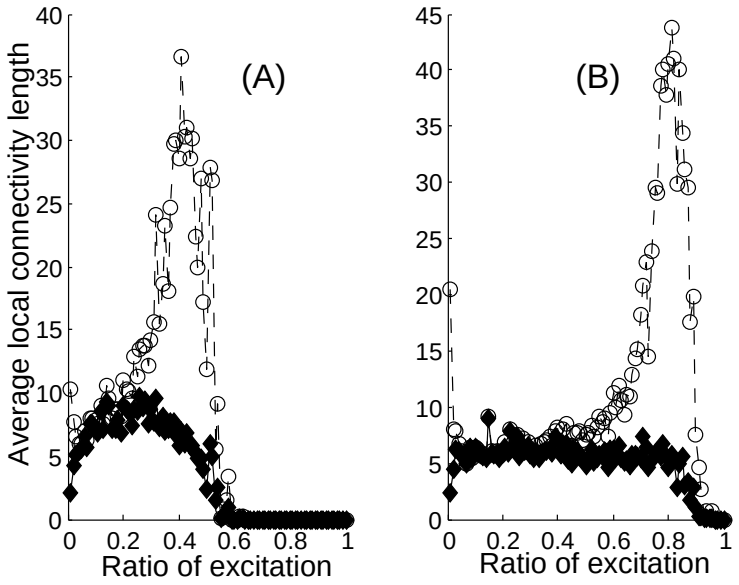


FIGURE 7. **Average local distance vs. excitation ratio**

A: $r_{A^+/A^-} = 0.1$, B: $r_{A^+/A^-} = 0.6$. Diamonds: average local distances for the evolving network. Circles: average local distances for the corresponding random net.

top of that hierarchy. Our model is called 'Editorial Board' and may have the following ranks:

- reader
- author
- reviewer
- board member
- editor

Other structures are possible, including specialization in the form of action editors, to mention a simple extension.

Hierarchy can be imposed in a rigid fashion. One corresponding structure is

- customer/user
- worker
- quality assurance
- member of the advisory board
- CEO

Note, that the role of the reader or that of the user is special: they provide the reinforcing feedback of the external world by purchasing the information or the product.

At this point, we note the following. Starting from efficiency based self-organizing communities will favor competition as a derivative within and amongst communities. Competition will require security rules. Security rules

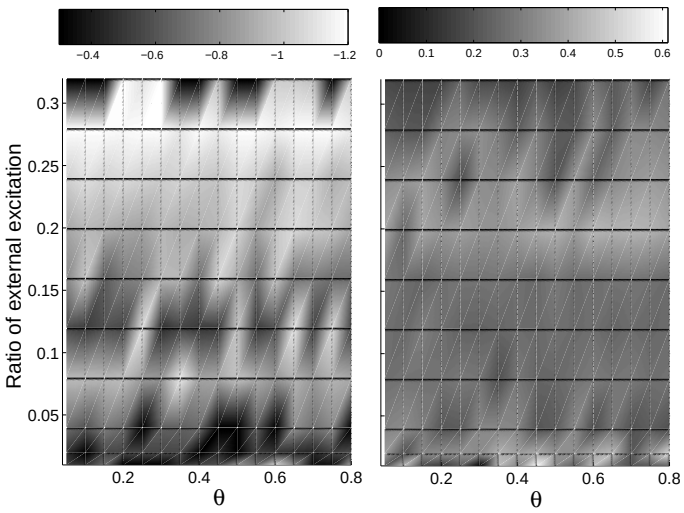


FIGURE 8. **Power-law with significant interaction** **Left:** *exponent of the power law*, **right:** *relative percentage of the power-law domain* as a function of r_{ex} and excitation threshold θ. $r_{A^+/A^-} = 0.1$ Results are averaged over 700 steps. Input from other neurons could exceed the external inputs by a factor of 10. The exponent of the power-law approximates -1 for broad regions of θ and r_{ex}. Outside this region the network may vanish or may start to oscillate.

can be imposed at two different levels: (i) at the level of the self-organizing community, and (ii) at the level of interaction between such communities. These concepts are novel, so we give an example. If a particular self-organizing community is made of readers and authors only, then the collaboration with another community with more strict quality assurance, e.g., board members, action editors, editor will be limited. Limitations are also restricted by the definition of readers. If the community is closed then special rules for interaction with other communities need to be established. If anybody can be a reader, then self-organizing collaborative efforts between communities may emerge. In turn, some communities will be efficient whereas others will be less efficient. The efficiency is governed by - at least - two factors:

(1) The efficiency gained by collaboration
(2) The slowing down of project organization as determined by the time constraints of security rules

Note that we aim to develop concepts for collaborative schemes amongst humans and Internet robots. The time constraints could be most restrictive for large organizations.

A good way to think of such communities is to consider

(1) the participants of the community as agents,
(2) the community as an assembly of agents,
(3) the security rules of the community as the language of the community,

(4) the interaction amongst communities as communication between agents speaking possibly different 'languages'.

Alike to real languages, there will be a competition amongst these languages. This competition will favor languages, which are simple for level communication and which are abundant. The formulation of the S2OS as agents speaking languages is not the subject of this proposal. At this point, one may think to establishing security rules,

(1) which meet the common general structure of self-organizing human collaboration,
(2) which allow the derivation of 'pre-wired' hierarchies,
(3) which allow collaboration amongst communities,
(4) which can be secure if required by the founders of the self-organizing community.

The proposed model consists of independent, autonomous sub-systems, governed by a set of local goals. Dynamics of these independent subsystems defines a global, self-organizing model without the necessity of central authorization. The aim is to develop a system architecture that supports collaboration while guarantees information confidentiality, integrity, and availability. In addition, anonymity of the initiator and responder are guaranteed as well as accountability that is used to deter misuse or for billing services.

Our long-term goals are to provide privacy (hide identity of initiator and responder), provide mechanism for secure online collaboration (protocols what can be shared between the parties, policy that defines who is allowed to access what and what level), and provide accountability (misuse detection, or billing purposes. Our assumption is, that users may form ad-hoc communities based on their interest and expertise. The same user may belong to different communities at different roles or may assume different roles within a given community.

In the editorial board example, all users are allowed to submit papers to be reviewed, and all users are allowed to review papers. Both submitters and reviewers are classified according to the quality of work they perform. Initially all users and all papers are assigned an 'initial' impact factor. Based on the submitted reviews, the impact factor of each publication may increase or decreases. In addition, reviewers' impact factor will be changed based on the response to their comments, i.e., level of update of the reviewed paper based on the evaluation.

Additional restrictions are involved on the model to guarantee security and anonymity while preventing frauds by providing accountability. For example, a user may not be able to change his/her impact factor by login in as new user, submit the same paper to get a better evaluation of the paper, or submit high quality reviews of his/her own work. Although, some of the above concepts have been independently considered by researcher, there does not exist a complete model that supports both security and anonymity in a single, dynamic environment, while preserving accountability.

3.2. **Self-Organizing System (SOS) for Online Collaboration.** SOS may be distributed and may have human as well as robotic participants. In order to

demonstrate the generality of our approach a strictly robotic example is provided here. Access control issues, and some experiments about their efficiency are provided to support our argumentation.

An Internet robot may represent a hierarchy. The robot is on a host computer and will be called 'hostess'. The hostess can launch topic specific searches using crawlers, which download Internet documents one-by-one, using the links of the documents. Intelligent crawlers adapt according to the reinforcing signal received from the external world and transferred by the hostess. Communication is possible amongst

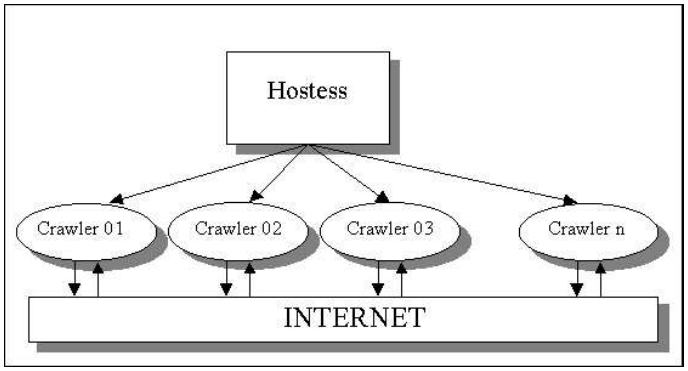


FIGURE 9. **Hostess-crawler system**
Internet is explored by crawlers (which download and examine documents), whereas communication with other entities is governed by the hostess, which can launch and modify crawlers based on reinforcements.

- crawlers and
- hostesses

Communication can be direct or indirect. For example, if the hostess launches crawlers on the same topic and provides positive reinforcement only to the one, which brings the information first, indirect communication happens. This indirect communication will manifest itself as a competition. The competition will be directly addressed if crawler maintain and refresh a list of good links - we call those weblogs - which used to provide the most positive reinforcement from the crawler. One may see these weblogs as local sub-crawlers, which have starting points and search a smaller neighborhood. The algorithm of weblog selection is provided in Fig. 10

The origin of competition can be seen as follows. The small-world property can trap crawlers in a neighborhood where links are abundant and point to each other. Crawlers might not find any novel information. To overcome this limitation, each crawler may maintain a list of links, which are worth to visit. Assume two crawlers: one with a long list and another one with a single link. The second crawler will find the novel information on that single link with high probability. However, that information will not be novel for the first crawler anymore, therefore, the first crawler will not be able to "publish" it at the hostess. As the consequence, the first crawler will lower the value of this link and will visit this site less-and-less frequently.

1) $s \leftarrow startNode$
2) FOR $i = 1$ *To memory*
   a) $s \leftarrow chooseRL(frontier)$
   b) $wl \leftarrow s, \quad w(s) \leftarrow \sum_{k=i}^{memory} r_k$
3) $sort(wl, w(s)), \quad head(wl, wlsize)$
4) REPEAT *(for every local search)*
   a) $w'(s) = w(s) * \gamma, \quad \forall\, s \in wl$
   b) $s \leftarrow choose(wl)$
   c) FOR $i = 1$ *To memory*
      i) $s \leftarrow chooseRL(frontier)$
      ii) $wl \leftarrow s, \quad w'(s) = w(s) + \sum_{k=i}^{memory} r_k$
      iii) $sort(wl, w(s)), \quad head(wl, wlsize)$
5) UNTIL end of experiment

FIGURE 10. **Algorithm for forming a weblog**
The algorithm has three important parameters: (i) *'memory'*: size of environment around a node of the network, examined during the evaluation of that node, (ii) *'wlsize'*: size of the weblog (number of links that can be remembered), and (iii) *'γ'*: discount factor of past rewards received around a node.

The downloaded part of the Internet and its division between the two crawlers can be seen in Fug. 12

The competition gave rise to a decrease the age of found relevant document in our model experiments as it is shown in Figs. 13 and 14

3.3. **Experiments on the Internet.** The internet experiments used two crawlers. Crawlers were launched from the CNN Money site (http://www.cnnmoney.com). The crawlers run for about five days, starting on September 12 and finishing on September 16, 2002. The weekend was on 14-15 of September.

At first, all documents were novel to the crawlers, the number of rewarded downloads was large. This can be seen in Fig. 15. The two crawlers are distinguished by colors. At the beginning, the green crawler mostly follows the red crawler and receives very few results. At around 5000th download the green crawler finds other routes to follow. This new routes turn out to be rewarding, the efficiency of the green crawler becomes about the same as that of the red crawler in this region. Both crawlers are inefficient between downloads 15,000 and 30,000, this is the *weekend*. After the weekend, the novel routes found by the green crawler are more efficient. The number of novel documents is much larger at the start of the search when all documents are novel for the competing crawlers. This can be seen by considering the slope of the curves at the beginning of the search.

In about half the cases we could establish the date of the document. Results are shown in Fig. 16. After four days, some of the documents found by the

FIGURE 11. **Two competing crawlers using weblogs in a "small-world"**
The figure depicts the connectivity table (representing the links) between nodes. Typical scale-free small world connectivity can be seen here: some nodes have many links pointing onto them (rows), whereas others have a long list of links pointing to other nodes (columns). Green (red) dots depict the links used by the first (second) crawler. Efficient self-organizing division of the world, i.e., the task space, can be seen. Weblogs contained 10 links.

crawlers are still 'old', but the relative rate of novel documents is increasing strongly after the start of the new week (at around reward number 2,700). The number of novel documents is much larger at the start of the search when all documents are novel for the competing crawlers. Note that 365 units on the vertical scale correspond to a year. The crawlers collect the daily documents,too, as it can is shown by the slight slope of maxima (the convex hull of the maxima) of the points.

More detailed insight can be gained by means of Fig. 17. This figure contains only the mst recent downloads, which are restricted to a few days and contain a large number of very recent documents. At this time the green crawler is more efficient: the novel routes found by the green crawler (forced by the competition) happened to be more rewarding during this last day of the studies.

The sharing of the links between the two crawlers is depicted in the following three figures. Figure 18 shows the full time region. Blue dots represent the links followed by both crawlers. It can be seen that during the full time domain, the explored regions are mostly covered by both crawlers.

This situation changed during competition, as it is demonstrated by Figs. 19 and 20. These figures show the links followed in the last 25% of the downloads and the last 10% of the downloads, respectively.

The last figure (Fig. 21)s serve to underline our statement that the improvement in crawler performance is due to the continuous improvement of weblogs. We show the best 10 weblogs used by both crawlers as a function of changes

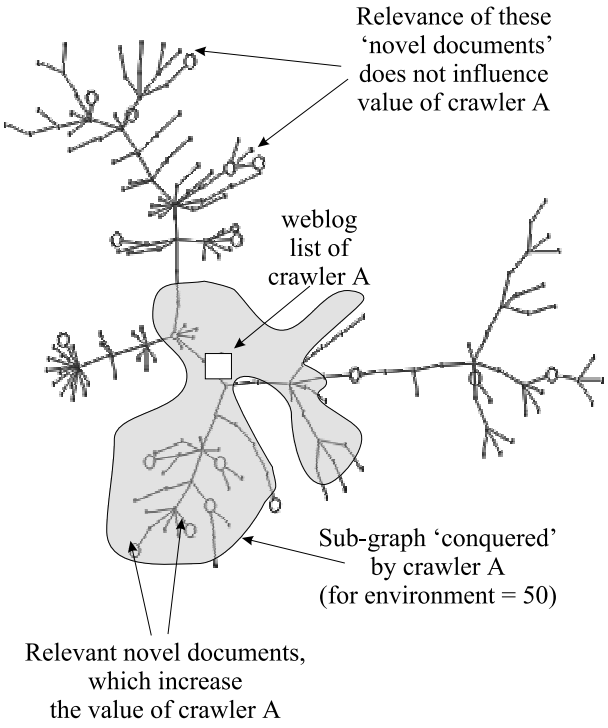


FIGURE 12. **Division of a substructure of the Internet between two-crawlers**
The gray region belongs to crawler A. Both crawlers maintain a list of weblogs, which are efficient restart links to continue search when novel information in the actual neighborhood has been collected. Weblogs are ordered by their values and change dynamically amongst crawlers during competition.

made in the weblogs. Recall that changes are *induced* by reinforcement. At the very beginning the overlap is clear, the crawlers use the same site. Newer and newer nodes are promoted to the top ten part of the weblog list. Overlap occurs when one of the crawlers become less efficient (see the blue region in the middle of the figure). In most of the time, the surfing region of the crawlers is launched from distinct html pages.

The concept of selection of weblogs is the key to improve performance, whereas time sharing and reinforcing only the first crawler provides the improvement in efficiency. This particular form of reinforcement, in turn, gives rise to collaboration amongst competitive agents.

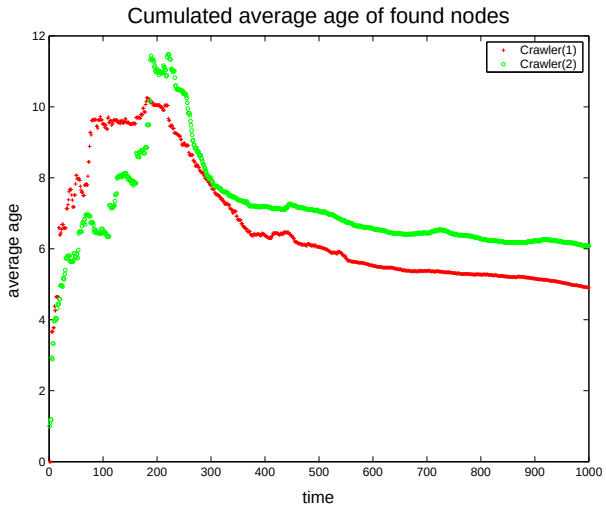


FIGURE 13. **Decrease of age of found novel documents** New documents appear, whereas old documents disappear in this experiment. Competition settles around "time" 600. The overlap amongst links searched by crawlers becomes minimized by this time. Both crawlers have high connectivity weblogs. The age of found novel documents is decreasing afterwards.

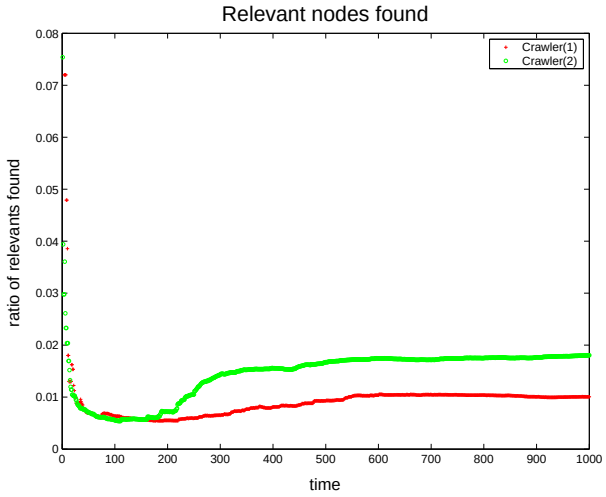


FIGURE 14. **Increase of access to novel documents.** New documents appear, whereas old documents disappear in this experiment. Competition settles around "time" 600. The overlap amongst links searched by crawlers becomes minimized by this time. Both crawlers have high connectivity weblogs. The efficiency of found novel documents is increasing afterwards.

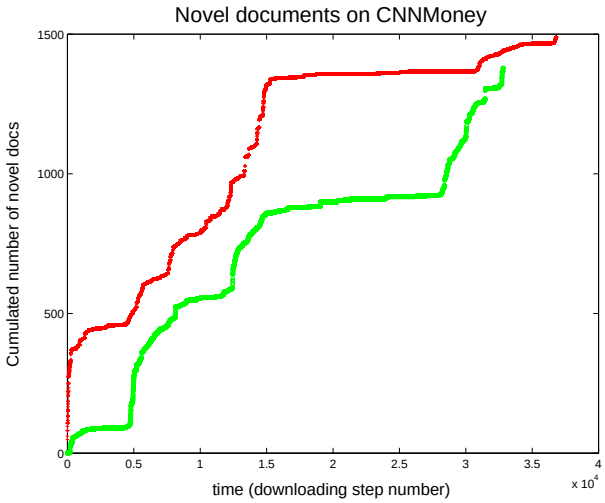


FIGURE 15. **Novel documents found by two competing crawlers**
Two competing crawlers are searching the **CNN Money**site for novel information (information not yet known to the crawlers). During this search only the crawler which delivers the information first, is reinforced.

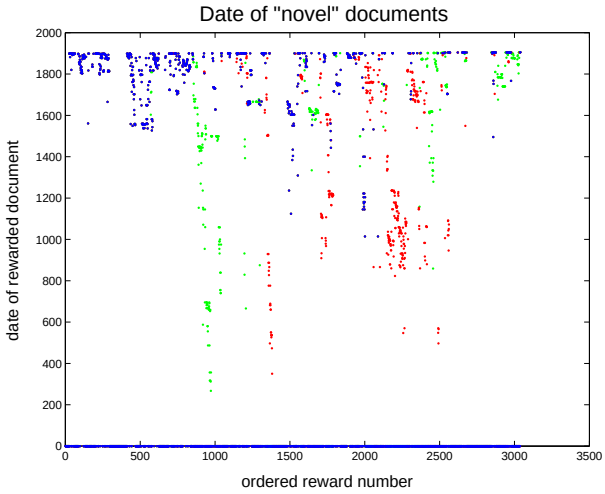


FIGURE 16. **Age of documents found by competing crawlers**
Age of the documents are shown as a function of reward order. Red (green) dots: document downloaded by the red (green) crawler. Blue dots: documents downloaded by both crawlers. Note that 365 units on the vertical scale correspond to a year.

4. SHORT DISCUSSION AND FUTURE DIRECTIONS

The competitive-collaborative method has the following advantages. Processing is local: each crawler modifies values of known links based on the reinforcement. There is a global improvement in performance.

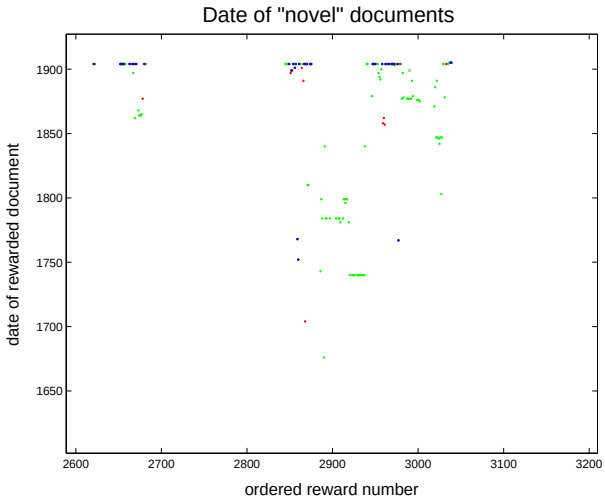


FIGURE 17. **Age of documents found by competing crawlers after four days**
Age of the documents are shown as a function of reward order. Red (green) dots: document downloaded by the red (green) crawler. Blue dots: documents downloaded by both crawlers. Note that 365 units on the vertical scale correspond to a year.

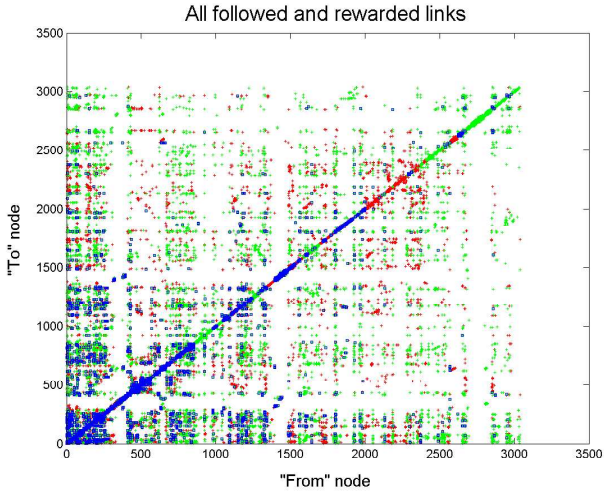


FIGURE 18. **Links used by crawlers**
Red, green and blue dots: links followed by red, green and both crawlers respectively All followed links are depicted.

Let us consider now interacting hostesses. Some of the hostesses (authors) can be selected to being a 'reviewer', a member of the board, or the editor. The selection could be based on their efficiency, or that the search is divided

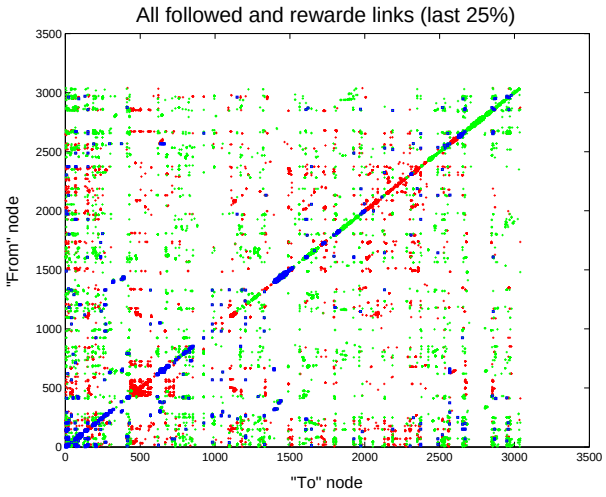


FIGURE 19. **Links used by crawlers in the last quarter of the experiments**
Red, green and blue dots: links followed by red, green and both crawlers respectively All followed links are depicted. Sharing of the links has been improved.

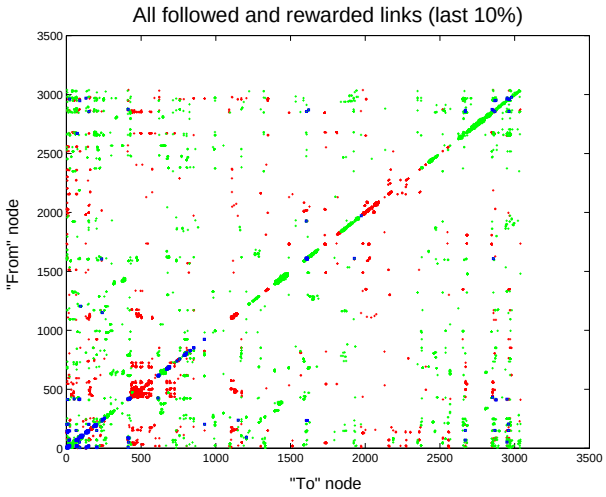


FIGURE 20. **Links used by crawlers in the last 10% of the experiments**
Red, green and blue dots: links followed by red, green and both crawlers respectively All followed links are depicted. Further improvement in link sharing can be seen.

into topics and central hostesses have access to the context of the search. Another intriguing possibility is that in a general search problem, search topics are may also be subject to competition. Under this condition, small-worlds

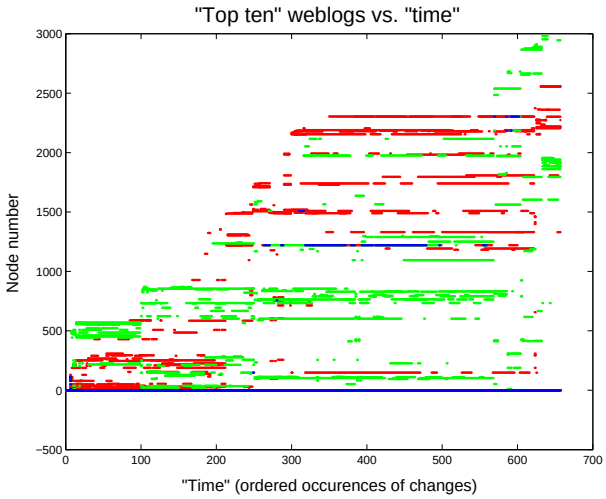


FIGURE 21. **Top ten links used by the crawlers**
Red, green and blue dots: links followed by red, green and both crawlers respectively All followed links are depicted. Further improvement in link sharing can be seen.

become larger (!), the 'diameter' belonging to topics is larger, however, searching may become more efficient. Central hostesses may emerge as a result of this competition. The emergence of central hostesses can be promoted by the type of interactions, the access control, allowed by the robotic community. We conclude this paragraph by noting that the hostess-crawler system is one type of SOS, whereas collaborating hostesses can form another type of SOS.

4.1. **Future directions.** Interaction amongst the components of the system fall into two categories:

(1) Interactions within a single SOS
(2) Interactions among different SOSs

Each SOS is motivated to become stronger – as a result of reinforcemnt. The overall gain of a SOS, in this competitive environment, depends on the success of other SOSs. Each member of a single SOS supports each other by sharing relevant information. On the other hand, no such information sharing is done between different SOSs. Temporal sequences and synchronizations support this information sharing process. If Crawler A discovers event p on site X then it might be useful to send that information to Crawler B "new information on site X". That information can serve as the "context" for crawler B and can help to accelerate search.

That is, one should be looking for synchronous or spatio-temporal patterns. The ideal tool – as it seems to us – is the two types of Hebbnet learning rules. In turn, one may expect that Hebbnets with symmetric learning rules will represent synchronous patterns, whereas Hebbnets with asymmetric learning rules will represent spatio-temporal patterns. The expected prototype form of such spatio-temporal patterns is an avalanche-like pattern, alike in most `sustained` *systems.*

The point to remember is that communication amongst crawler or hostesses can be

- indirect based upon reinforcement
- direct
  - by sending documents to each other,
  - by sending elements of weblogs,
  - by sending information about the fact that a novel document has been found 'in my area',
- or combinations of any of these.

The communication amongst crawlers or hostesses is defined by the access control rules. These rules are of central importance. These rules determine the type of SOS that will emerge. To each type of communication, the access control model of the corresponding SOS must ensure that a component will not be able to disclose or modify resources if it does not have the proper authorization. These access rights determine the type of co-operation and the distribution of topics amongst the small sub-systems.

5. Conclusions

The competitive-collaborative model is efficient according to our preliminary studies. It allows one to impose competition simply via reinforcement. Reinforcement gives rise to the division of tasks and improved efficiency within subtasks.

The concept can be extended by communication methods other then the indirect communication of the reinforcing signal. Other types of message sending methods open questions about interactions amongst self-organizing systems (SOS), the access control amongst those regarding the type of information to be communicated.

The idea of establishing experience based connections using Hebbnets seem most promising as the tool of collaboration, provided that access control issues are solved. Learning rules of a Hebbnet can discover and, in turn, may allow clustering of synchronous patterns as well as spatio-temporal patterns, most probably in the form of avalanches.

Studies along these lines are beyond the scope of the present contract. Considerations on different possibilities are currently in *basic research phase*. In particular, studies about

- **access control rules**
- **learning and recognition of spatio-temporal patterns in early phases**
- **secure communication methods**

have been started and might reach a more mature state in about three months.

Appendix

6. Crawlers

6.1. **Introduction.** The number of documents on the world-wide web is way over 1 billion [Diligenti et al., 2000]. The number of new documents is over 1 million per day. The number of documents that change on a daily basis, e.g.,

documents about news, business, and entertainment, could be much larger. This ever increasing growth presents a considerable problem for finding, gathering, ordering the information on the web. The only search engine that may still warrant that the information it provides is not older than 1 month is AltaVista[1]. However, the number of indexed pages on Altavista is about 250 million documents. Google[2], on the other hand, is indexing about 1,300 million pages, but Google does not warrant any refresh rate of these documents.

The problem is complex: These search engines are not up-to-date and information gathering is not always efficient with these engines. Search engines may offer too many documents; sometimes on the order of hundreds or many thousands. Many web pages have no value, e.g., by making use of a large set of keywords, or being simply huge collections of documents originating from broad sources.

Specialized, possibly personalized crawlers are in need. This problem represents a real challenge for methods of artificial intelligence and has been tackled by several research groups [Cho et al., 1998, Dean and Henzinger, 1999, Chakrabarti et al., 1999a, McCallum et al., 1999, Kolluri et al., 2000, Lawrence, 2000, McCallum et al., 2000, Mukherjea, 2000, Murdock and Goel, 1999]. One of the first attempts in this direction was made by Chakrabarti et al. [Chakrabarti et al., 1999b] who put forth the idea of *focused crawling*. To understand the idea, let us consider crawling in general. Assume that 'you are at a node' of the web. This node has been analyzed and you have to decide what to do next. It is very possible that relevant information can be find in the immediate neighborhood of this node. In turn, you download all the documents next to you and start to analyze those documents. Doing so, you may found relevant documents or may not. When you are done you have the option to download all the documents that are two steps away from you and to analyze those documents. This approach is well known in the AI literature and is called breadth first technique. However, the world-wide web is *'small'*: The WWW had about 800 million nodes in 1999 and the number of minimal hops required to reach most documents from any particular document was 19 [Albert et al., 1999]. Such connectivity structure between units is called 'small world'. In turn, breadth first search incurs an enormous burden as a function of depth. At one point (at a given depth) breadth first search needs to be abandoned and a decision is to be made to which node to move next. To decide on that move, the values of the nodes need to be estimated from the point of view of the goal of the search. *Focused crawling* is based on this idea. Focused crawling makes an attempt to classify the content of the document. If the document falls into the search category then the document is downloaded and the links of the documents are followed.

Diligenti et al. [Diligenti et al., 2000] have recognized the pitfall of focused crawling: searched information on the web is typically *hidden*: Sites of particular interest may have a lower number of directed links then sites of general interest. In turn, we might face the 'needle in the haystack' problem with the haystack being sites on general interest. The hidden property is thus the implicit consequence of our particular interest.

[1] http://www.altavista.com
[2] http://www.google.com

Let us consider sites dealing with support vector machines (SVMs). Sites about SVMs are not typical on the web. Not all sites dealing with SVM are linked. In turn, focused crawling could be rather inefficient and this direct search for SVM sites might fail. On the other hand, most of the SVM sites are within (i.e., linked to) academic environments, or within sites dealing with information technology. These topics are much more general and might have much more links and a much higher 'visibility'. In turn, searching for the environment of SVM sites, could be much more efficient. A hand-waving argument can be given as follows. Documents are linked to each other. Links are made by those for whom the document has value. These links form the one-step *context* of the document. The one-step context, in turn, may be characteristic to the document. The one-step environment of the document (i.e., documents that are one step away), documents that are two steps away, etc., form the 'context' of the document. When we search for a document, by definition, we shall encounter the environment of the document first. In turn, first we might search for the environment of the document. This is the idea behind 'context focused crawling' (CFC) [Diligenti et al., 2000]. This idea, which is trivial for graphs with high clustering probability (e.g., regular lattices), could be criticized for the case of 'small worlds', when documents – on average – are about as far as the environment of the document. However, the question is intriguing, because the visibility could be much less for searched documents than that of the environment of the searched documents.

CFC does not take into consideration the varieties on the web: Environments may differ. For example, for small universities or for small research institutes 'one-step context' may correspond to 'two-step context' for large departments of large universities. If the order of contexts might change then CFC will go close and will miss the documents. In turn, the decision whether to 'stay and download' at a given site or 'not to download but move' can be seriously jeopardized. Fast adapting value estimation method may provide an attractive solution to this search problem where information is hidden within not-yet-experienced environments. The environment of high value documents can provide reinforcing feedback in a straightforward fashion. Interestingly, reinforcement learning (RL) has not been found particularly efficient for searching the world-wide web [Rennie et al., 1999]. The efficiency of RL, however, depends strongly on feature extraction. It seems natural to explore the CFC idea as the initial feature extraction method for RL. Here we show combine CFC with RL to search on the web.

6.2. **Methods.**

6.2.1. *Preprocessing of texts.* There is a large variety of methods that try to classify texts [McCallum, 1996, Blum and Mitchell, 1998, Dumais et al., 1998, Kaski, 1998, Chakrabarti et al., 1998, Kolenda et al., 2000, Mitchell, 1999, McCallum, 1999, Hofmann, 2000, Nigam et al., 2000, Kabán and Girolami, 2000, Vinokourov and Girolami, 2000a, Joachims, 2000, Dominich, 2000]. Most of these methods are based on special dimension reduction. First, the occurrence, or sometimes the frequency of selected words is measured. The subset of all possible words ('bag of words' (BoW)) is selected by means of probabilistic measures. Different methods are used for the selection of the 'most important' subset. The occurrences (0's

and 1's) or the frequencies of the selected words of the subsets are used to characterize all documents. This low – typically 100 – dimensional vector is supposed to encompass important information about the type of the document. Different methods are used to derive 'closeness measures' between documents in the low dimensional spaces of occurrence vectors or frequency vectors. The method can be used both for classification, i.e., the computation of decision surfaces between documents of different 'labels' [Blum and Mitchell, 1998, Dumais et al., 1998, Joachims, 2000, Nigam et al., 2000] and clustering, a more careful way of deriving closeness (or similarity) measures when no labels are provided [McCallum, 1996, Kaski, 1998, Hofmann, 2000, Kabán and Girolami, 2000, Vinokourov and Girolami, 2000b].

We tried several BoW based classifiers on the 'Call for Papers' (CfP) problem[3]. CfP is considered a benchmark classification problem of documents: The ratio of correctly classified and misclassified documents can be automated easily by checking whether the document has the three word phrase *'Call for Papers'*, or not. Classifiers were developed for one-step, two-step environments, etc., for CfP documents. We found that these classifiers perform poorly for the CfP problem. In agreement with published results [Dumais et al., 1998], supervised SVM classification was superior to other methods. SVM was simple and somewhat better than Bayes classification. However, SVM requires a large number of support vectors for the CfP problem.

6.2.2. *SVM classification.* The SVM classifier operates similarly to perceptrons. SVM, however, has better generalizing capabilities, see, e.g., the comprehensive book of Vapnik [Vapnik, 1995] a tutorial material [Smola and Schölkopf, 1998], comparisons with other methods [Guyon et al., 1992, Cristianini and Shawe-Taylor, 1999], improved techniques [Keerthi et al., 1999] and references therein[4]. The trained SVM was used in 'soft mode'. That is, the output of the SVM was not a decision (yes, or no), but instead, the output could take continuous values between 0 and 1. A saturating sigmoid function[5] was used for this purpose. In turn, (i) the non-linearity of the decision surface was not sharp, (ii) for inputs close to the decision surface the classifier provides a linear output. The output of the sigmoid non-linearity can be viewed as the probability of a class. These probabilities for the different classes are distinct yardsticks working on possibly different features. The RL algorithm was used to estimate the *value* of these yardsticks.

6.2.3. *Value estimation.* There is a history of value estimation methods based on reinforcement learning: Some of the important steps — judged subjectively — are in the cited papers: [Korf, 1985, Minton, 1988, Sutton, 1988, Watkins, 1989, Schmidhuber, 1991, Mahadevan and Connell, 1992, Dayan and Hinton, 1993, Kaelbling, 1993, Rummery and Niranjan, 1996, Littman et al., 1995, Mataric, 1997, Dietterich, 2000]. A thorough review on the literature and the history of RL can be found in [Sutton and Barto, 1998]. In our approach, value estimation

[3]The CfP problem is defined by deleting the phrase 'call for paper' from the document, executing search on the internet and considering each document that contains the phrase 'call for paper' a 'hit'.

[4]Note that SVM has no adjustable parameters.

[5]$\text{output} = \frac{1}{1+\exp(-\lambda * \text{input})}$

plays a central role. Value estimation works on states (s) and provides a real number, the *value*, that belongs to that state: $V(s) \in \mathrm{R}$. Value estimation is based on the *immediate rewards* (e.g., the number of hits) that could be gained at the given state by executing different actions (e.g., download or move). Value of a state (a node, for example) is the long-term cumulative reward that can be collected starting from that state and using a *policy*. Policy is a probability distribution over different actions for each state: policy determines the probability of choosing and action in a given state. Policy improvement and the finding of an optimal policy are central issues in RL. RL procedures can be simplified if all possible future states are available and can be evaluated. This is our case. In this case one does not have to represent the policy. Instead, one could evaluate all neighboring nodes of the actual state and move to (and/or download) the one with the largest estimated long term cumulated reward, the estimated value. Typically one includes random choices for a few percentages of the steps. These random choices are called *'explorations'*. The estimated value based greedy choice is called *'exploitation'*.

If the downloaded document contains the phrase 'call for papers' then the learning system incurs an immediate reward of 1. If a downloaded document does not contain this phrase then there is negative reward (i.e., a punishment) of -0.01. These numbers were rather arbitrary. The relative ratio between reward and punishment and the magnitude of the parameter of the sigmoid function do matter. These parameters influence learning capabilities. Our studies were constrained to a fixed set of parameters. One may expect improvements upon optimizing these parameters for a particular problem. In our case, search over the internet was time consuming and prohibited this optimization.

Value estimation makes use of the following upgrade

$$V^{+}(s_t) = V(s_t) + \alpha * (r_{t+1} + \gamma * V(s_{t+1}) - V(s_t)) \tag{5}$$

where α is the learning rate, $r_{t+1} \in \mathrm{R}$ is the immediate reward, $0 < \gamma < 1$ is the discount factor, and subscripts $t = 1, 2, \ldots$ indicate action number (i.e., time). This particular upgrade is called temporal differencing with zero eligibility, i.e., the TD(0) upgrade. TD methods were introduced by Sutton [Sutton, 1988]. An excellent introduction to value estimation, including the history of TD methods and description on the applications of parameterized function approximators can be found in [Sutton and Barto, 1998]. Concerning details of the RL technique, (a) we used eligibility traces. (b) Opposed to the description given above, we did not need explorative steps because the environments can be very different and that diminished the need for exploration. (c) We did not decrease the value of α by time to keep adaptivity. (d) We approximated the value function as follows

$$V(s) \approx \sum_{i=1}^{n} w_i \; \sigma(\mathrm{SVM(i)}) \tag{6}$$

where the output of the i^{th} SVM (i.e., the i^{th} component of the output) is denoted by SVM(i), $\sigma(.)$ denotes the sigmoid function acting on the outputs of the SVM classifiers, w_i is the weight (or relevance) of the i^{th} classifier determined by upgrade Eq. 5. If the quality of the upgrade is measured by the mean square error of the estimations then the following approximate weight upgrade

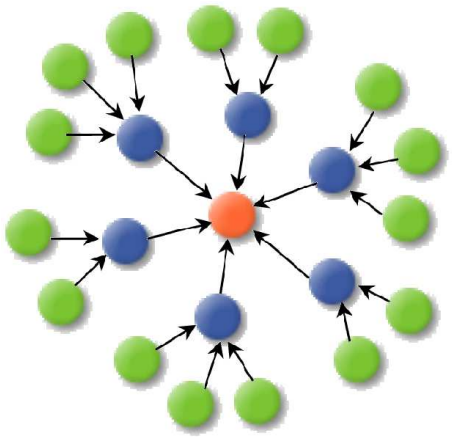


FIGURE 22. ***Context* of the document**
Document and its first and second 'neighbors'.

can be derived for the weights (see, e.g., [Sutton and Barto, 1998] for details):

$$\Delta w_i = \alpha * (r_{t+1} + \gamma * V(s_{t+1}) - V(s_t)) * \sigma(\text{SVM(i)}). \tag{7}$$

This upgrade — extended with eligibility traces [Sutton, 1988, Sutton and Barto, 1998] — was used in our RL engine.

6.3. **Features and learning to search.**

6.4. **Breadth first crawler.** A crawler is called *breadth first crawler*, if it first downloads the document of the launching site, continues by downloading the documents of all first neighbors of the launching site, then the documents of the neighboring sites of the first neighbor sites, i.e., the documents of the second neighbor sites, and so on.

6.4.1. *Context focused crawler.* A target document and its environment are illustrated in Fig. (22). . The goal is to locate the document by recognizing its environment first and then the document within. The CFC method [Diligenti et al., 2000] was modified slightly — in order to allow direct comparisons between the CFC method and the CFC method extended by RL value estimation — and the following procedure was applied. First, a set of irrelevant documents were collected. The k^{th} classifier was trained on (good) documents k-steps away from known target documents and on (bad) irrelevant documents. The classifier was trained to output a positive number ('yes') for good documents and to output a negative number ('no') for irrelevant documents. The outputs were scaled into the interval (0,1) by using the sigmoid function $\sigma(x) = (1 + exp(-\lambda x))^{-1}$. If the k^{th} classifiers output was close to 1 – according to its decision surface – there is a target document k-steps away from the actual site/document. If more than one classifier outputs 'yes' then only the best classifier is considered in CFC. Other outputs are neglected. The CFC idea with SVM classifiers is shown in Fig. (23)(a). CFC maintains a list of visited links ordered according to the SVM classification. One of the links

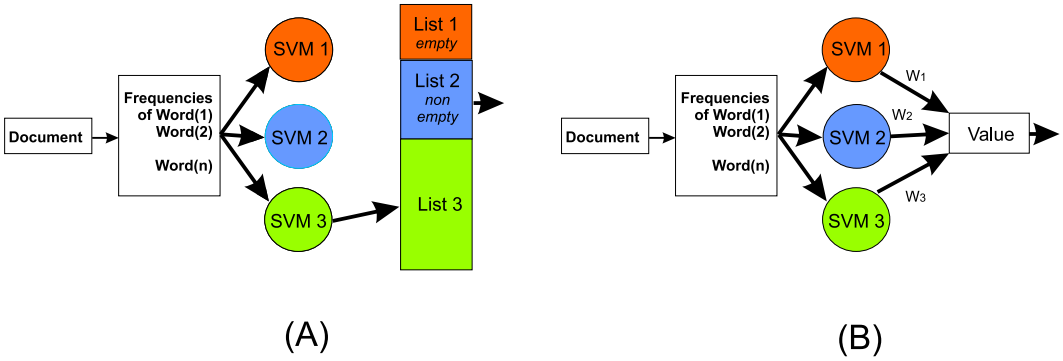


FIGURE 23. **SVM based document classifiers** (A) Classification of distance from document using SVM classifiers. The CFC method maintains a list of visited links ordered according to the SVM classification. One of the links belonging to the best non-empty classifier is visited next. (B) Value estimation based on SVM classifiers. Reinforcement learning is used to estimate the importance of the different classifiers during search.

belonging to the best non-empty classifier is visited next (this procedure is called backtracking).

The problem of the CFC method can be seen by considering that neighborhoods on the WWW may differ considerably. Even if the k^{th} classifier is the best possible such classifier for the whole web, it might provide poor results in some (possibly many) neighborhoods. For example, if there is a large number of connected documents all having the promise that there is a valuable document in their neighborhood – but there is, in fact, none – then the CFC crawler will download all invaluable documents before moving further. It is more efficient to learn which classifiers predict well and to move away from regions which have great but unfulfilled promises.

It has been suggested that classifiers could be retrained to keep adaptivity [Diligenti et al., 2000]. The retraining procedure, however, takes too long[6] and can be ambiguous if CFC is combined with backtracking. Moreover, retraining may require continuous supervisory monitoring and supervisory decisions. Instead of retraining, we suggest to determine the relevance of the classifiers during the search.

6.4.2. *CFC and RL: Fast adaptation during search.* Reinforcement learning offers a solution here. If the prewired order of the classifiers is questionable then we could learn the correct ordering. There is nothing to loose here, provided that learning is fast. If prewiring is perfect then the fast learning procedure will not modify it. If the prewiring is imperfect then proper weights will be derived by the learning algorithm.

[6]Training may take on the order of a day or so on 700 MHz Pentium III according to our experiences.

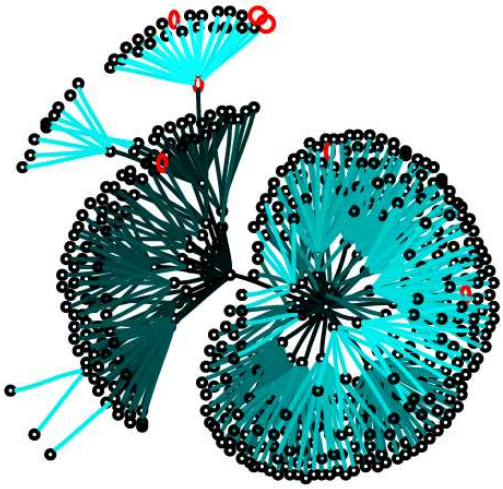


FIGURE 24. **Search pattern for breadth first crawler.** Search was launched from neutral site. A site is called neutral if there is very few target document in its environment. Diameter of open circles is proportional to the number of target documents downloaded. Edges are color coded. There are two extremes. Dark blue: Site was visited at the early stage during the search. Light blue: Recently visited site. (For further details, see text.)

The outputs of the SVMs can be saved. These outputs can be used to estimate the value of a document at any instant. Value is estimated by estimating weights for each SVM and adding up the SVM outputs multiplied by these weights. In turn, one can compute value based ordering of the documents with minor computational effort and this reordering can be made at each step. This reordering of the documents replaces prewired ordering of the CFC method. The new architecture is shown in Fig. (23)(b).

6.5. **Results and discussion.** The CfP problem has been studied. Search pattern at the initial phase for the breadth first method is shown in Fig. (24).

Search patterns for the context focused crawler and the crawler using RL based value estimation are shown in Fig. (25) and Fig. (26). The launching site of these searches was a 'neutral site', a relatively large site containing few CfP documents (`http://www.inf.elte.hu`). We consider this type of launching important for web crawling: It simulates the case when mail lists are not available, traditional search engines are not satisfactory, and breadth first search is inefficient. This particular site was chosen because breadth first search could find very few documents starting from this site.

'Scales' on Fig. (25) and Fig. (26) differ from each other and from that of Fig. (24). 'True surfed scale' would be reflected by normalizing to edge thickness. Radius of open circles is proportional to the number of downloaded target documents. The CFC is only somewhat better in the initial phase than

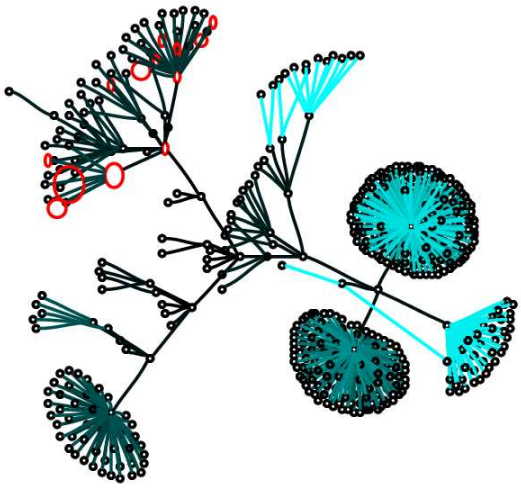


FIGURE 25. **Search pattern for context focused crawler.** Search was launched from neutral site. Diameter of open circles is proportional to the number of target documents downloaded. Edges are color coded. There are two extremes. Dark blue: Site was visited at the early stage during the search. Light blue: Recently visited site.

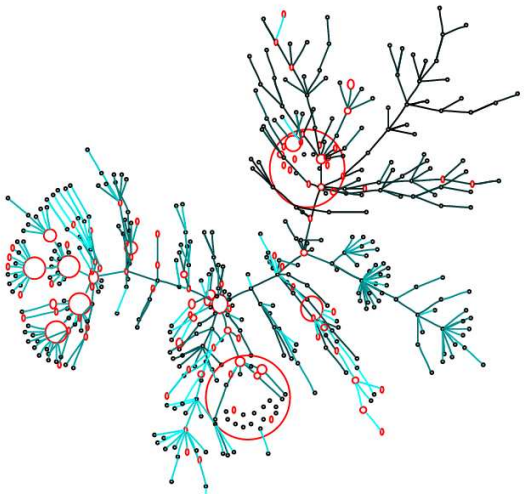


FIGURE 26. **Search pattern for CFC *and* reinforcement learning**
Search was launched from neutral site. Diameter of open circles is proportional to the number of target documents downloaded. Edges are color coded. There are two extremes. Dark blue: Site was visited at the early stage during the search. Light blue: Recently visited site.

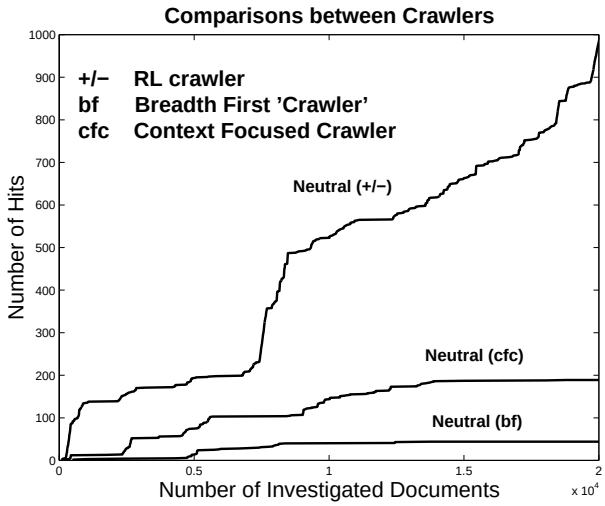


FIGURE 27. **Results of breadth first, CFC and CFC based RL methods.**

the breadth first method. Longer search shows that CFC becomes considerably better than the breadth first method when search is launched from this neutral site.

Quantitative comparisons are shown in Fig. (27). According to the figure, upon downloading 20,000 documents, the number of hits were about 50, 200, and 1000 for the breadth first, the CFC and CFC based RL crawlers, respectively. These launches were conducted at about the same time. We shall demonstrate that the large difference between CFC and CFC based RL method is mostly due to the adaptive properties of the RL crawler.

There are two site types that have been investigated. The first site type is the neutral site that has been described before. The other site was a mail server on conferences. Also, for some examples there are runs separated by one month (March, 2001). A large number of summer conferences made announcements during this month.

First, let us examine the initial phase of the search. This initial phase of the search (the first 200 downloaded documents) is shown in Fig. (28). According to this figure downloading is very efficient from the mail server site in each occasion. The (non-adapting) CFC crawler utilizing averaged weights is superior to all the other crawlers — almost all downloaded documents are hits. Close to this site there are many relevant documents and the 'breadth first crawler' is also efficient here. Nevertheless, the CFC crawler outperforms the breadth first crawler in this domain. Launching from neutral sites is inefficient at this early phase. Breadth first method finds no hit close to the neutral site (not shown in the figure).

Middle phase of the search is shown in Fig. (29). Performance in the middle phase is somewhat different. Sometimes, launches from the neutral site can find excellent regions. The CFC crawler is still competitive if launched from the mail server. Launches from the mail list spanning one month looked similar to each other; conference announcements barely modified the results.

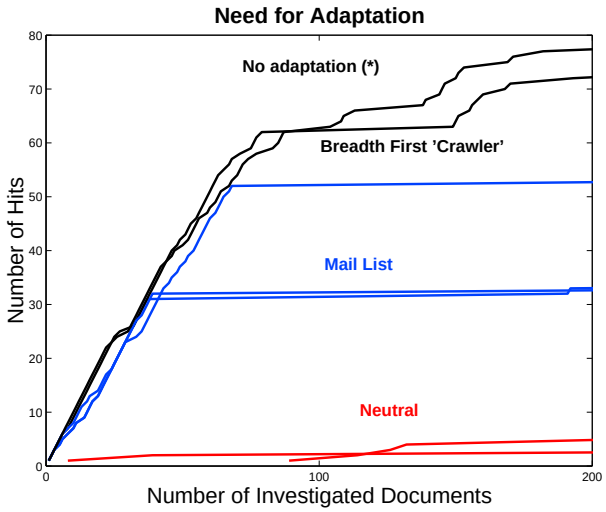


FIGURE 28. **Comparisons between 'neutral' and mail server sites in the initial phase.** Reward and punishment are given in the legend of the figure. Differences between similar types are due to differences in launching time. The largest time difference between similar types is one month. Neutral site (thin lines): `http://www.inf.elte.hu`. Mail list (thick lines): `http://www.newcastle.research.ec.org/cabernet/events/msg00043.html`. Search with 'no adaptation' (dotted line) was launched from mail list and used average weights from another search that was launched from the same place.

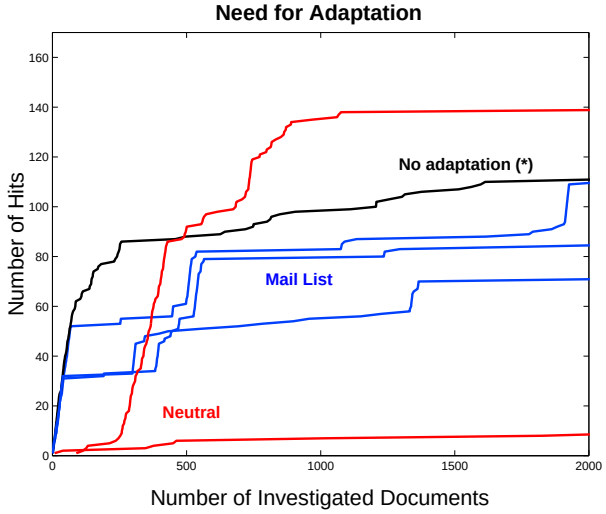


FIGURE 29. **Comparisons between 'neutral' and mail server sites up to 2000 documents.** Same conditions as in Fig. (28)

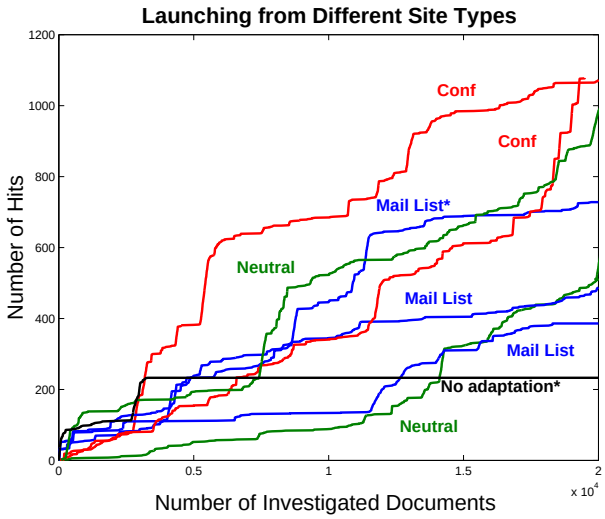


FIGURE 30. **Comparisons between different sites up to 20,000 documents.** Same conditions as in Figs. (28) and (29). Search with 'no adaptation' used average weights from another search that was launched from the same place (denoted by $*$)

Search results up to 20,000 documents are shown in Fig. (30). This graph contains results from a subset of the runs that we have executed. These runs were launched from different sites; the neutral site and the mail list, as well as a third type, the 'conference' site: `http:// www.informatik.uni-freiburg.de/index.en.html`. This latter is known to be involved in organizing conferences. Adapting RL crawlers collected a large number of documents from all site types and during the whole (one month) time region. The rate of collection was between 2%-5%. In contrast, although the collection rate is close to 100% for the CFC launched from the mail list site up to 200 downloads, the lack of adaptation prohibits this crawler to find new target documents in circa 17,000 downloads at later stages. Taken together:

(1) Identical launching conditions may give rise to very different results one month later.
(2) Starting from a neutral site can be as effective as starting from a mailing list for the adaptive RL crawler.
(3) The lack of adaptation is a serious drawback even if the crawler is launched from a mailing list.

The importance of adaptation is also demonstrated by the RL weights assigned during search. These weights are shown in the following figures. Figure (31) depicts the weights belonging to the different SVMs launched from the mail list site. At the beginning of the search the weights are almost perfectly ordered; the largest weight is given to the SVM that predicts relevant document 'one step away' whereas the 4^{th} and the 5^{th} SVMs have the smallest weights. That is, RL 'pays attention' to the first SVM and pays less attention to the others. This order changes as time goes on. There are regions (at around

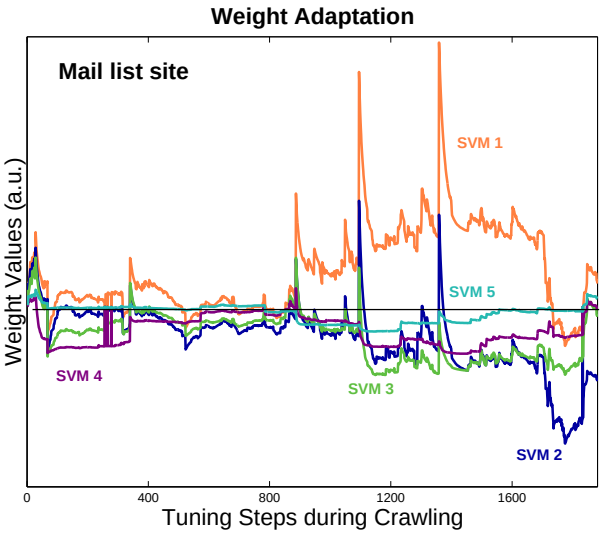


FIGURE 31. **Change of weights of SVMs upon downloads from mail site.**
Horizontal axis: Occasions when weights were trained.

tuning step number 1700 on the horizontal axis) where most attention is paid to the 5^{th} SVM and smaller attention is paid to the others. This means that the crawler will move away from the region. The order of importance changes again when a rich region is found; the importance of the first SVM recovers quickly and, in turn, crawling is dominated by the weight of the first SVM: The crawler 'stays' and downloads documents.

'Weight history' is different at the neutral site (Fig. (32)). Up to about 100 downloads very few relevant documents were found at this site. The value of weight of the 5^{th} SVM is slightly positive, whereas the values of the others are negative. The 1^{st} and the 2^{nd} SVMs are weighted the 'worst'; weights belonging to these classifiers are large negative numbers. At this site, the order of SVMs that were trained at around target documents is not appropriate. Situation changes quickly when a rich region is found. In such regions the 1^{st} SVM takes the lead. It is typical that the weight of the 5^{th} SVM is ranked second. That is, the adaptation concerns mostly whether the crawler should stay or if it should move 'far away'. In turn, information contained by the 'context' is relevant and can be used to optimize the behavior of the crawler.

6.6. **Conclusions.** We have suggested a novel method for web search. The method makes use of combinations of two popular AI techniques, support vector machines (SVM) and reinforcement learning (RL). The method has a few adapting parameters that can be optimized during the search. This parameterization helps the crawler to adapt to different parts of the web. The outputs of the SVMs, together, formed a set of 'yardsticks' for the estimation of the distance from target documents. The value (the weight) of the different yardsticks may be very different at different neighborhoods. The point is that (i) RL is efficient with good features (the as k-step SVMs in this case),

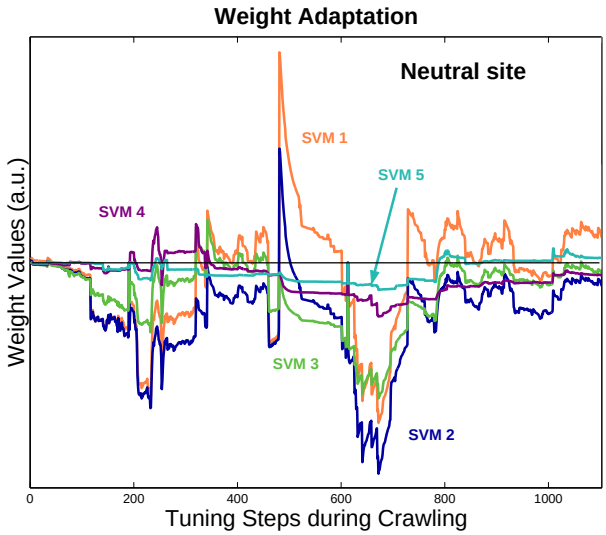


FIGURE 32. **Change of weights of SVMs in value estimation for 'neutral' site.**
Horizontal axis: Occasions when value estimation was erroneous and weights were trained.

(ii) if there are just a few parameters for RL then these parameters can be trained quickly by rewarding for target documents. RL has many different formulations all of which could be applied here. Most promising are the approaches that can take into account (many) different criteria in the search objective [Fraser and Hauge, 1998, Gábor et al., 1998, Dubois et al., 2000]. Alas, RL methods are capable of extracting features [Thrun and Schwartz, 1995] that may complement the prewired SVM features.

References

[Abbott and Nelson, 2000] Abbott, L. and Nelson, S. (2000). Synaptic plasticity: taming the beast. *Nature Neuroscience*, 3:1178–1183.

[Albert and Barabási, 2002] Albert, R. and Barabási, A. (2002). Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47–91.

[Albert et al., 1999] Albert, R., Jeong, H., and A.-L. Barabási (1999). Diameter of the world-wide web. *Nature*, 401:130–131.

[Barabási and Albert, 1999] Barabási, A. L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286:509–512.

[Barabási et al., 2000] Barabási, A. L., Albert, R., and Jeong, H. (2000). Scale-free characteristics of random networks: The topology of the world wide web. *Physica A*, 281:69–77.

[Bell et al., 1997] Bell, C., Han, V., Sugawara, Y., and Grant, K. (1997). Synaptic plasticity in a cerebellum-like structure depends on temporal order. *Nature*, 387:278–281.

[Blum and Mitchell, 1998] Blum, A. and Mitchell, J. (1998). Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*. `http: //www.cs.cmu.edu /afs/cs.cmu.edu /project/theo-11 /www/wwkb /colt98_final.ps`.

[Bohland and Minai, 2001] Bohland, J. W. and Minai, A. A. (2001). Efficient associative memory using small-world architecture. *Neurocomputing*, 38-40:489–496.

[Chakrabarti et al., 1998] Chakrabarti, S., Dom, B., and Indyk, P. (1998). Enhanced hypertext categorization using hyperlinks. In Haas, L. M. and Tiwary, A., editors, *Proc. SIGMOD-98, ACM Int. Conf. on Management of Data*, pages 307–318, Seattle, US. ACM Press, New York, US. `http://www.almaden.ibm.com/cs/k53/irpapers/sigmod98.ps`.

[Chakrabarti et al., 1999a] Chakrabarti, S., Gibson, D., and McCurley, K. (1999a). Surfing the Web backwards. In *8th World Wide Web Conference*, Toronto, Canada. `http: //www8.org/w8-papers /5b-hypertext-media /surfing/`.

[Chakrabarti et al., 1999b] Chakrabarti, S., van der Berg, M., and Dom, B. (1999b). Focused crawling: a new approach to topic-specific Web resource discovery. In *8th International World Wide Web Conference (WWW8)*. `http: //www.cs.berkeley.edu / soumen/doc /www1999f/pdf /www1999f.pdf`.

[Cho et al., 1998] Cho, J., García-Molina, H., and Page, L. (1998). Efficient crawling through URL ordering. *Computer Networks and ISDN Systems*, 30(1–7):161–172. `http: //www-db.stanford.edu /pub /papers /efficient-crawling.ps`.

[Cristianini and Shawe-Taylor, 1999] Cristianini, N. and Shawe-Taylor, J. (1999). *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, UK.

[Dayan and Hinton, 1993] Dayan, P. and Hinton, G. E. (1993). Feudal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 5, pages 271–278, San Mateo, CA. Morgan Kaufmann.

[Dean and Henzinger, 1999] Dean, J. and Henzinger, M. (1999). Finding related pages in the world wide web. *WWW8 / Computer Networks*, 31(11-16):1467–1479. `http://www.research.compaq.com/SRC /personal/monika/papers /monika-www8-1.ps.gz`.

[Dietterich, 2000] Dietterich, T. G. (2000). Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303.

[Diligenti et al., 2000] Diligenti, M., Coetzee, F., Lawrence, S., Giles, C. L., and Gori, M. (2000). Focused crawling using context graphs. In *26th International Conference on Very Large Databases, VLDB 2000*, Cairo, Egypt. `http: //www.neci.nec.com /lawrence /papers /focus-vldb00 /focus-vldb00.ps.gz`.

[Dominich, 2000] Dominich, S. (2000). A unified mathematical definition of classical information retrieval. *Journal of the American Society for Information Science*, 51(7):614–624.

[Dubois et al., 2000] Dubois, D., Grabisch, M., Modave, F., and Prade, H. (2000). Relating decision under uncertainty and multicriteria decision making models. *International Journal of Intelligent Systems*, 15:967–979.

[Dumais et al., 1998] Dumais, S., Platt, J., Heckerman, D., and Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. In *7th International Conference on Information and Knowledge Management, ICIKM'98*. ACM. `http://robotics.stanford.edu/users/sahami/papers-dir/cikm98.pdf`.

[Ferster, 1996] Ferster, D. (1996). Is neural noise is just a nuisance. *Science*, 273:1812.

[Fraser and Hauge, 1998] Fraser, N. M. and Hauge, J. W. (1998). Multicriteria approval: application of approval voting concepts to mcdm problems. *Journal of Multi-Criteria Decision Analysis*, 7:263–273.

[Gábor et al., 1998] Gábor, Z., Kalmár, Z., and Szepesvári, C. (1998). Multi-criteria reinforcement learning. In *Proceedings of the Fifteenth International Conference on Machine Learning*.

[Guyon et al., 1992] Guyon, I., Vapnik, V. N., Boser, B. E., Bottou, L. Y., and Solla, S. A. (1992). Structural risk minimization for character recognition. In Moody, J. E., Hanson, S. J., and Lippmann, R. P., editors, *Advances in Neural Information Processing Systems 4*, pages 471–479, San Mateo, California. Morgan Kaufmann.

[Hebb, 1943] Hebb, D. (1943). *The Organization of Behavior: A Neuropsychological Theory*. Wiley, New York, USA.

[Hofmann, 2000] Hofmann, T. (2000). Learning the similarity of documents: An information-geometric approach to document retrieval and categorization. In *Neural Information Processing Systems*, volume 12, pages 914–920. MIT Press.

[i Cancho and Solé, 2001] i Cancho, R. F. and Solé, R. (2001). Optimization in complex networks. arXiv:arch-ive.cond-mat/0111222.

[Joachims, 2000] Joachims, T. (2000). Estimating the generalization performance of an SVM efficiently. In *Proc. 17th International Conf. on Machine Learning*, pages 431–438. Morgan

Kaufmann, San Francisco, CA. http:// www-ai.informatik.uni-dortmund.de /DOKUMENTE /joachims_99e.ps.gz.

[Kabán and Girolami, 2000] Kabán, A. and Girolami, M. (2000). Clustering of text documents by skewness maximisation. In *2'nd International Workshop on Independent Component Analysis and Blind Source Separation, ICA'2000*, pages 435 – 440, Helsinki.

[Kaelbling, 1993] Kaelbling, L. (1993). Hierarchical learning in stochastic domains: Preliminary results. Proceedings of the Tenth International Conference on Machine Learning, San Mateo, CA. Morgan Kaufmann.

[Kandel and O'Dell, 1992] Kandel, E. R. and O'Dell, T. (1992). Are adult learning mechanisms also used for development? *Science*, 258:243–245.

[Kaski, 1998] Kaski, S. (1998). Dimensionality reduction by random mapping: Fast similarity computation for clustering. In *Proc. of Int. Joint Conf. on Neural Networks*, volume 1, pages 413–418. IEEE Service Center, Piscataway, NJ. http://websom.hut.fi/websom/doc/ps/kaski98ijcnn.ps.gz.

[Keerthi et al., 1999] Keerthi, S., Shevade, S., Bhattacharyya, C., and Murthy, K. (1999). Improvements to Platt's SMO Algorithm for SVM Classifier Design. Technical Report CD-99-14, Dept. of Mechanical and Production Engineering, National University of Singapore. http: //guppy.mpe.nus.edu.sg /~mpessk/smo_mod.ps.gz.

[Kleinberg, 1998] Kleinberg, J. (1998). Authoritative sources in a hyperlinked environment. In *9th ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677.

[Kolenda et al., 2000] Kolenda, T., Hansen, L., and Sigurdsson, S. (2000). Independent components in text. Ed.: M. Girolami, http: //eivind.imm.dtu.dk /publications /1999 /kolenda.nips99.ps.gz.

[Kolluri et al., 2000] Kolluri, R., Mittal, N., Ventakachalam, R., and Widjaja, N. (2000). Focussed crawling. http: //www.cs.utexas.edu /users /ramki /datamining /fcrawl.ps.gz.

[Korf, 1985] Korf, R. (1985). *Learning to solve problems by searching for macro-operators*. Pitman Publishers, Boston.

[Kulkarni et al., 2000] Kulkarni, R. G., Stough, R. R., and Haynes, K. E. (2000). Towards modeling of communities of practice (CoPs): A Hebbian learning approach to organizational learning. *Technological Forecasting and Social Change*, 64:71–83.

[Latora and Marchiori, 2001] Latora, V. and Marchiori, M. (2001). Efficient behavior of small-world networks. *Physical Review Letters*, 87(19).

[Lawrence, 2000] Lawrence, S. (2000). Context in web search. *IEEE Data Engineering Bulletin*, 23(3):25–32.

[Littman et al., 1995] Littman, M. L., Cassandra, A., and Kaelbling, L. P. (1995). Learning policies for partially observable environments: Scaling up. In Prieditis, A. and Russell, S., editors, *Proceedings of the Twelfth International Conference on Machine Learning*. Morgan Kaufmann.

[Magee and Johnston, 1997] Magee, J. and Johnston, D. (1997). A synaptically controlled, associative signal for hebbian plasticity in hippocampal neurons. *Science*, 275:209–213.

[Mahadevan and Connell, 1992] Mahadevan, S. and Connell, J. (1992). Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, 55:311–365.

[Marchioria and Latorac, 2000] Marchioria, M. and Latorac, V. (2000). Harmony in the small-world. *Physica A*, 285:539–546.

[Markram et al., 1997] Markram, H., Lubke, J., Frotscher, M., and Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science*, 215:213–215.

[Mataric, 1997] Mataric, M. (1997). Behavior-based control: Examples from navigation, learning, and group behavior. *J. of Experimental and Theoretical Artificial Intelligence*, 9:2–3.

[McCallum, 1996] McCallum, A. (1996). Bow: A toolkit for statistical language modelling, text retrieval, classification and clustering. `http://www.cs.cmu.edu/~mccallum/bow`.

[McCallum, 1999] McCallum, A. (1999). Multi-label text classification with a mixture model trained by em. `http: //www.cs.cmu.edu/~mccallum /papers /multilabel-nips99s.ps.gz`.

[McCallum et al., 1999] McCallum, A., Nigam, K., Rennie, J., and Seymore, K. (1999). Building domain-specific search engines with machine learning techniques. In *AAAI-99 Spring Symposium on Intelligent Agents in Cyberspace*. `http: //www.cs.cmu.edu/~mccallum /papers /cora-aaaiss98.ps`.

[McCallum et al., 2000] McCallum, A., Nigam, K., Rennie, J., and Seymore, K. (2000). Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163. `http: //www.cs.cmu.edu /afs /cs /user /kseymore /html /papers / cora-journal.ps.gz`.

[McCullough and Pitts, 1943] McCullough, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophysics*, 5:115–133.

[Miller and Troyer, 2002] Miller, K. D. and Troyer, T. W. (2002). Neural noise can explain expansive, power-law nonlinearities in neural response functions. *Journal of Neurophysiology*, 87:653–659.

[Minton, 1988] Minton, S. (1988). *Learning search control knowledge: An explanation based approach*. Kluwer Academic.

[Mitchell, 1999] Mitchell, T. (1999). The role of unlabeled data in supervised learning. In *Proceedings of the Sixth International Colloquium on Cognitive Science*. `http://www.ri.cmu.edu/pub_files/pub1/mitchell_tom_1999_2/mitchell_tom_1999_2.pdf`.

[Mukherjea, 2000] Mukherjea, S. (2000). WTMS: A system for collecting and analyzing topic-specific web information. In *9th World Wide Web Conference*. `http://www9.org/w9cdrom/293/293.html`.

[Murdock and Goel, 1999] Murdock, J. and Goel, A. (1999). Towards adaptive web agents. In *14th IEEE International Conference on Automated Software Engineering*. `http://www.cc.gatech.edu/morale /papers/ase99.ps`.

[Nigam et al., 2000] Nigam, K., Mccallum, A., Thrun, S., and Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134. `http://www.cs.cmu.edu/~knigam/papers/emcat-mlj99.ps.gz`.

[Oja, 1982] Oja, E. (1982). A simplified neuron model as a principal component analyzer. *J. Math. Biol.*, 15:267–273.

[Rennie et al., 1999] Rennie, J., Nigam, K., and McCallum, A. (1999). Using reinforcement learning to spider the web efficiently. In *16th International Conference on Machine Learning*, pages 335–343. Morgan Kaufmann, San Francisco, CA. `http: //www.cs.cmu.edu /~mccallum /papers /rlspider-icml99s.ps.gz`.

[Rummery and Niranjan, 1996] Rummery, G. A. and Niranjan, M. (1996). On-line q-learning using connectionist systems. Technical report, Cambridge University Engineering Department.

[Schmidhuber, 1991] Schmidhuber, J. (1991). Neural sequence chunkers. Technical Report FKI-148-91, Technische Universität München, München, Germany.

[Smola and Schölkopf, 1998] Smola, A. J. and Schölkopf, B. (1998). A tutorial on Support Vector Regression. Technical Report NC2-TR-1998-030, NeuroCOLT2.

[Sutton, 1988] Sutton, R. (1988). Learning to predict by the method of temporal differences. *Machine Learning*, 3:9–44.

[Sutton and Barto, 1998] Sutton, R. and Barto, A. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

[Thrun and Schwartz, 1995] Thrun, T. and Schwartz, A. (1995). Finding structure in reinforcement learning. volume 7 of *Advances in Neural Information Processing Systems*, San Mateo, CA. Morgan Kaufmann.

[Vapnik, 1995] Vapnik, V. (1995). *The nature of statistical learning theory*. Springer-Verlag, New York.

[Vinokourov and Girolami, 2000a] Vinokourov, A. and Girolami, M. (2000a). A probabilistic hierarchical clustering method for organizing collections of text documents. In *15th Int. Conf. on Pattern Recognition*, volume 2, pages 182–185. IEEE Computer Press. `http://cis.paisley.ac.uk/vino-ci0/vinokourov_ICPR00.ps`.

[Vinokourov and Girolami, 2000b] Vinokourov, A. and Girolami, M. (2000b). A probabilistic hierarchical clustering method for organizing collections of text documents. Technical Report ISSN 1461-6122, University of Paisley, Department of Computing and Information Systems, Paisley, PA1 2BE, UK. `http://cis.paisley.ac.uk/vino-ci0/hplsa_kais.ps`.

[Watkins, 1989] Watkins, C. (1989). *Learning from Delayed Rewards*. Phd thesis, King's College, Cambridge, UK.

[Watts and Strogatz, 1998] Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of small-world networks. *Nature*, 393:440–442.

1117 Budapest, Pázmány Péter sétány 1/C.

E-mail address: alorincz@axelero.hu, titkarsag@njszt.hu