On the Scalability of Social Order

—

Modeling the Problem of Double and Multi Contingency Inspired by Luhmann and Parsons

- APPENDIX -

Peter Dittrich[(1)], **Thomas Kron**[(2)], **Wolfgang Banzhaf**[(3)]

[(1)] Friedrich-Schiller-University of Jena, Institute of Computer Science, D-07743 Jena, Germany; and Jena Centre for Bioinformatics (JCB)
[(2)] University of Hagen, Department of Sociology, D-58084 Hagen, Germany
[(3)] University of Dortmund, Department of Computer Science, D-44221 Dortmund, Germany

Contents

Journal of Artificial Societies and Social Simulation (JASSS), 2003
http://jasss.soc.surrey.ac.uk/

9 APPENDIX

The Appendix contains further details about the model and about the simulation software that was used for the experiments described in the paper.

9.1 Interpretation of the Constant c_f

The constant c_f specifies an additive component to the activity value (Eq. (2)). What does that mean? And how should we set c_f?

Assume that we choose proportional selection ($\gamma = 1$). In that case the probability that activity i is selected is proportional to its corresponding activity value w^i_{AV}. Further assume that for $c_f = 0$ exactly one activity value is 1 and all other activity values are 0. Let us take an example for $N = 4$ possible activities:

$$c_f = 0: \; w^1_{AV} = 1, \; w^2_{AV} = 0, \; w^3_{AV} = 0, \; w^4_{AV} = 0. \tag{1}$$

We can see that although we use proportional selection, activity number 1 is always selected with probability one. Our selection method parameter γ has no influence in that situation.

If there should be always a non-deterministic (random) influence on the activity selection process, we have to choose a positive small value for c_f. Let us see what happens when c_f is set to 1:

$$c_f = 1: \; w^1_{AV} = 1 + \frac{1}{4}, \; w^2_{AV} = \frac{1}{4}, \; w^3_{AV} = \frac{1}{4}, \; w^4_{AV} = \frac{1}{4}. \tag{2}$$

In case of proportional selection the activity probabilities are calculated by normalizing the activity values:

$$c_f = 1: \; w^1_{AP} = \frac{5}{8}, \; w^2_{AP} = \frac{1}{8}, \; w^3_{AP} = \frac{1}{8}, \; w^4_{AP} = \frac{1}{8}. \tag{3}$$

In this case, activity 1 is selected with probability $5/8 = 62.5\%$, only, and there is a chance of $3/8 = 37.5\%$ that an "error" occurs.

In general the probability p_{err} that an "error" occurs is

$$p_{err} = 1 - \frac{1 + c_f/N}{1 + c_f} = \frac{c(N-1)}{(1+c)N}. \tag{4}$$

If we would like to set c_f such that the probability that an error occurs is p_{err} we just have to rearrange the previous equation:

$$c_f = \frac{N p_{err}}{1 - N(1 + p_{err})}. \tag{5}$$

The following table shows p_{err} for different settings of N and c_f (rounded to two significant digits):

	$N = 2$	$N = 4$	$N = 10$	$N = 100$	$N = 1000$
$c_f = 1$	0.25	0.375	0.45	0.50	0.50
$c_f = 0.1$	0.045	0.068	0.081	0.09	0.09
$c_f = 0.05$	0.024	0.036	0.043	0.047	0.047
$c_f = 0.01$	0.005	.0074	0.009	0.010	0.010
$c_f = 0.001$	0.0005	0.00074	0.0009	0.0010	0.0010

We can see, when we fix c_f and increase the number of possible activities N then also the probability p_{err} that an "error" occurs increases and converges to

$$\lim_{N\to\infty} p_{err} = 1 - \frac{1}{1+c_f}. \tag{6}$$

For the experiments presented here we have chosen $c_f = 0.01$. This means that in a situation where an agent is as sure as possible what to do and proportional selection is used, there is a chance of about 1% (0.5 % for $N = 2$) that a different activity is selected then the most likely one.

9.2 Influence of the Learning Rate

In Fig. 1 we can see how the average number of different activities in an interval of 50 steps and the average certainty O_{AV} depends on the learning rate r_{learn}. With increasing learning rate, the number of different activities decreases, as expected. We can see that the relative qualitative behavior of the model is independent of the choice of $r_{learn} > 0$. This is especially true for $r_{learn} > 0.2$.

9.3 Single Runs with Many Agents

Figure 13 and Fig. 14 show single runs of the multi-agent scenario. They correspond to the activity networks shown in Fig. 10 and Fig. 11, respectively.

9.4 Memory Models

This section describes additional memory models, which are implemented in our simulation software.

Memory Model 01 - Matrix Memory with Global Forgetting

Representation: The memory is represented by a $N \times N$ dimensional matrix $(m_{a,b})$ called **memory matrix**.

Initialization: The matrix is initialized with $m_{a,b} = 1/N$.

Memorize(a, b): First we reduce *every* entry in the memory matrix in order to model forgetting:

$$\forall i, j \in \{1, \ldots, N\} : m_{i,j} := \gamma_{mem} m_{i,j}. \tag{7}$$

Now we increase the entry in the memory matrix given by the index (a,b):

$$m_{a,b} := m_{a,b} + \sum_{i,j=1}^{N} (1 - \gamma_{mem}) m_{i,j}. \tag{8}$$

Lookup(a, b): Return the normalized entry of the memory matrix:

$$lookup(M, a, b) = \frac{1}{\sum_{b=1}^{N} m_{a,b}} m_{a,b}. \tag{9}$$

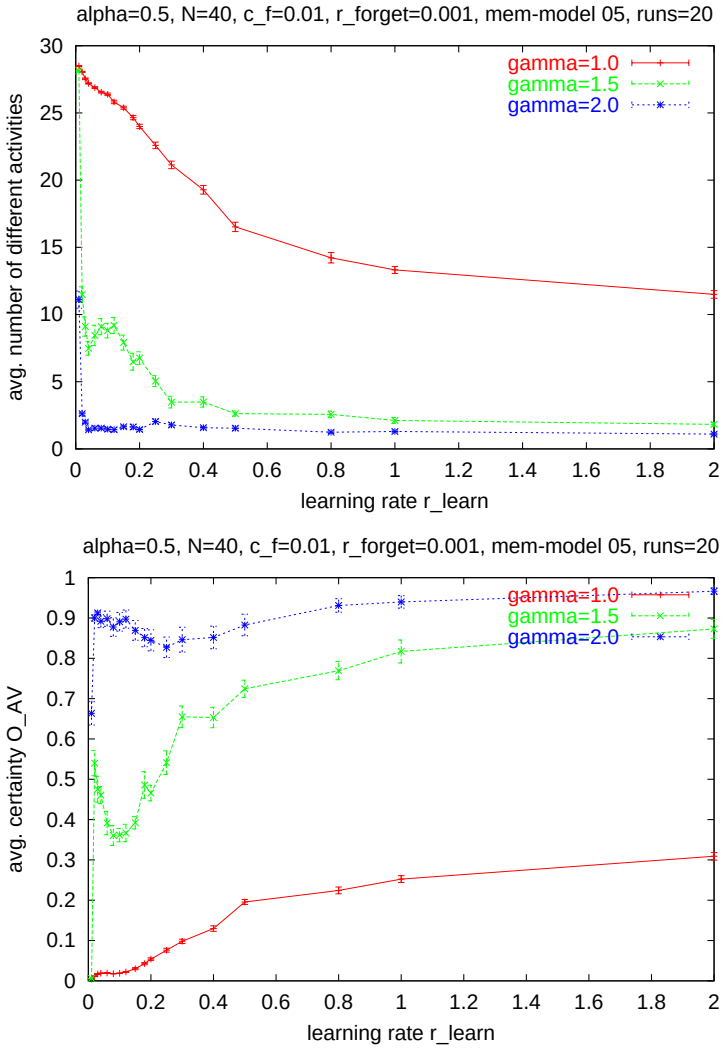

Figure 1: *Average number of different messages (activities) in an interval of 50 time steps for different learning rates $r_{learn}a$. Measurement started at time step 500, such that the transient phase at the beginning is not considered. Simulation time 1000 time steps for each run. Parameter setting: normal learning rate $r_{learn} = 0.2$, low forgetting rate $r_{forget} = 0.001$, number of activities $N = 40$, $\alpha = 0.5$.*

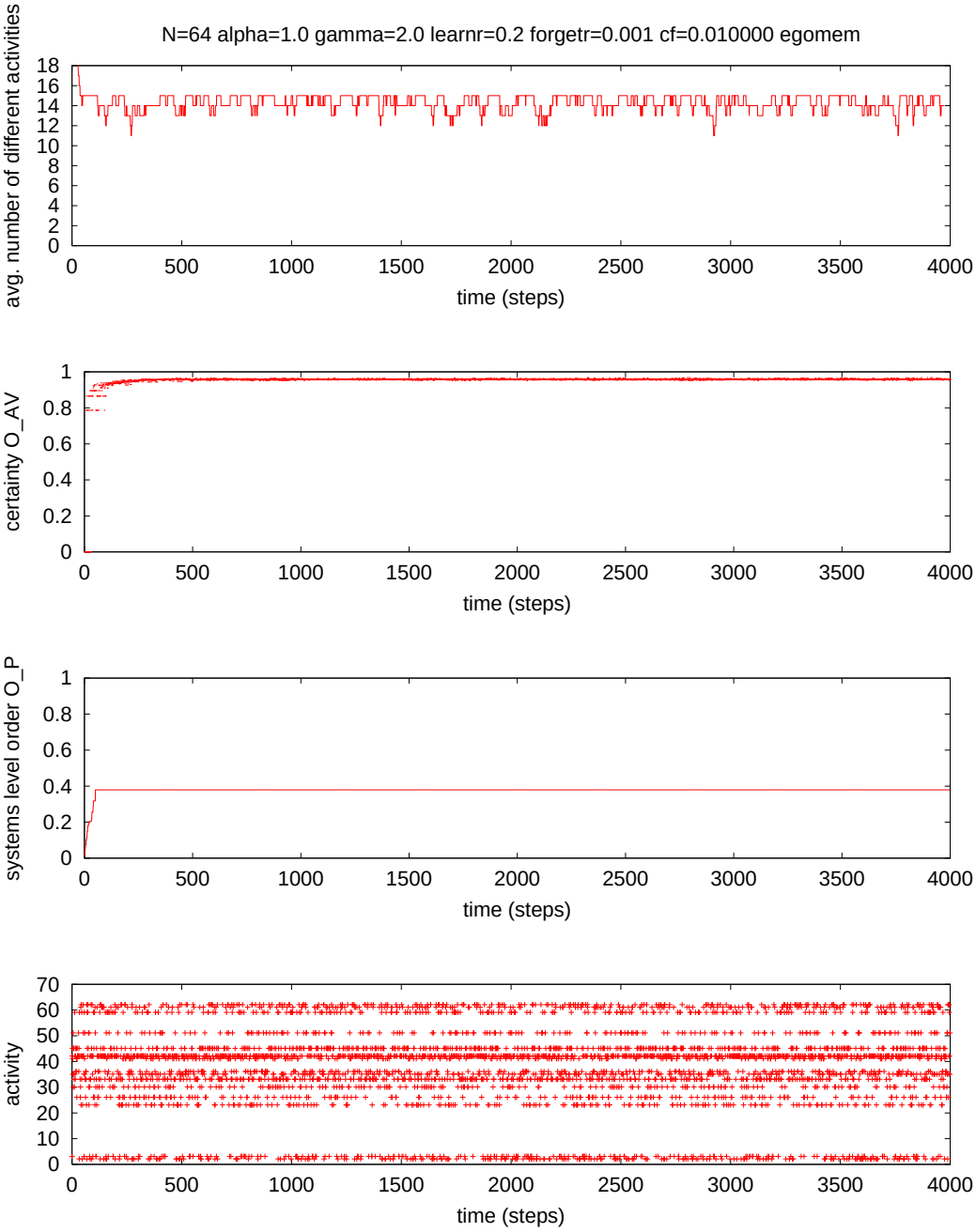

Figure 2: The run corresponding to Fig. 10 of the paper. The graph shown in Fig. 10 has been calculated at time step 4000. Parameters: $M = 20$ agents, $N = 64$ activities, $\alpha = 1.0$ (expectation-certainty only), $\gamma = 2.0$, $r_{learn} = 0.2$, $r_{forget} = 0.001$, $c_f = 0.010000$. $observers = 0$.

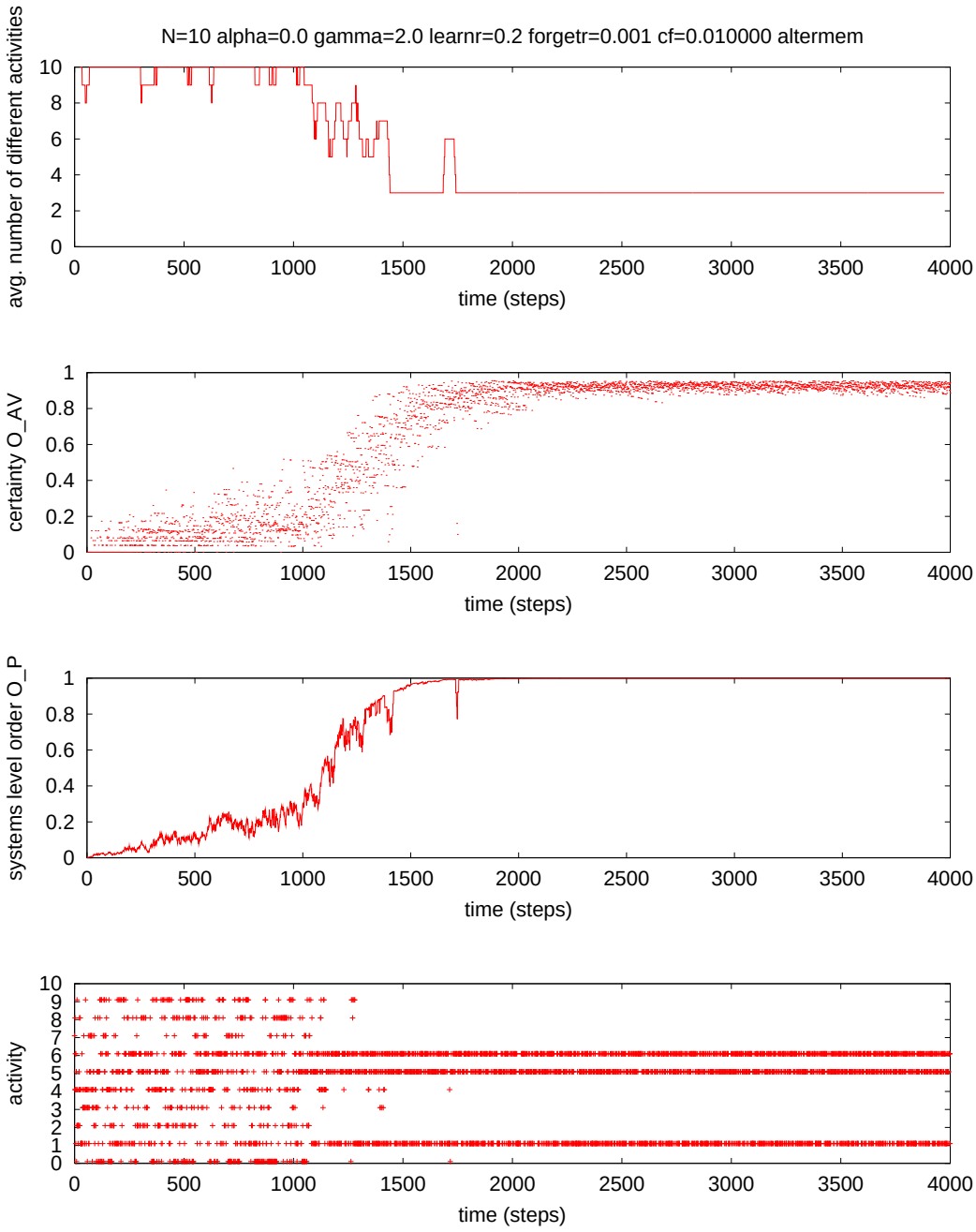

Figure 3: The run corresponding to Fig. 11 of the paper.The graph shown in Fig. 11has been calculated at time step 4000. Parameters: $M = 10$ agents, $N = 10$ activities, $\alpha = 0.0$ (expectation-expectation only), $\gamma = 2.0$ (quadratic selection), learning rate $r_{learn} = 0.2$, forgetting rate $r_{forget} = 0.001$, $c_f = 0.010000$. Alter-memory used for EE calculation. $observers = 0$.

Memory Model 02 - Matrix Memory with Local Forgetting

The same as Memory Model 01, but now events of the form (a', b') are forgotten only, if an event (a', b) is memorized.

Representation: The memory is represented by a $N \times N$ dimensional matrix $(m_{a,b})$ called **memory matrix**.

Initialization: The matrix is initialized with $m_{a,b} = 1/N$.

Memorize(a, b): First we reduce the entries of the memory matrix in row a in order to model forgetting:

$$\forall j \in \{1, \ldots, N\} : m_{a,j} := \gamma_{mem} m_{a,j}. \tag{10}$$

Now we increase the entry in the memory matrix given by the index (a,b):

$$m_{a,b} := m_{a,b} + \sum_{j=1}^{N} (1 - \gamma_{mem}) m_{a,j}. \tag{11}$$

Lookup(a, b): Return the normalized entry of the memory matrix:

$$lookup(M, a, b) = \frac{1}{\sum_{b=1}^{N} m_{a,b}} m_{a,b}. \tag{12}$$

Memory Model 03 - Non-Degenerating Memory

Representation: In the non-degenerating memory past events are stored in a table. Agents using that memory are able to "remember" the past n_{mem} events.

Initialization: There are two initialization methods: (1) The memory table is filled with random events (parameter `initRandomly` =1). (2) The memory table is empty at the beginning (parameter `initRandomly` =0).

Memorize(a,b): The operation $memorize(M, a, b)$ just stores the pair (a, b) in the table.

Lookup(a, b): For calculating the result $lookup(M, a, b)$ we do the following steps:

- Calculate the memory matrix $(m_{a,b})$:

$$m_{a,b} = \frac{c_M}{N} + \sum_{i=t-n_{mem}}^{t} \begin{cases} 1 & \text{if} \quad A[i] = a \text{ and } B[i] = b, \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

  where t is the current time step. A and B represent the columns of the table where the events are stored by $memorize$. $(A[i], B[i])$ is the entry which has been stored in the table of the memory at time step i.

- Return the normalized entry of the memory matrix:

$$lookup(M, a, b) = \frac{1}{\sum_{b=1}^{N} m_{a,b}} m_{a,b}. \tag{14}$$

Memory Model 04 - Linearly Degenerating Memory

Like Memory Model 03, but past events are less important.

In the linearly degenerating memory past events are stored in a table. Agents using that memory are able to "remember" the past n_{mem} events. The operation $memorize(M, a, b)$ just stores the pair (a, b) in the table.

The operation *lookup* is more complicated. For calculating the result $lookup(M, a, b)$ we do the following steps:

- Calculate the memory matrix $(m_{a,b})$:

$$m_{a,b} = \frac{c_M}{N} + \sum_{i=t-n_{mem}}^{t} \frac{n_{mem} - i + t}{n_{mem}} \begin{cases} 1 & \text{if} \quad A[i] = a \text{ and } B[i] = b, \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

  where t is the current time step. A and B represent the columns of the table where the events are stored by *memorize*. $(A[i], B[i])$ is the entry which has been stored in the table of the memory at time step i.

- Return the normalized entry of the memory matrix:

$$lookup(M, a, b) = \frac{1}{\sum_{b=1}^{N} m_{a,b}} m_{a,b}. \tag{16}$$

Memory Model 05 - Simple Neuronal Matrix Memory

The simple neuronal matrix memory is used for the experiments the paper and is described in Sec. 2.2.1 in detail.

9.5 Certainty Measures

Given a vector $(p_1, p_2, \ldots, p_N)$ the following functions for calculating the certainty are implemented:

Shannon Entropy

(Shannon and Weaver 1949)

$$f_{certainty}(p_1, p_2, \ldots, p_N) = 1 + \sum_{i=1}^{N} p_i \log_N p_i. \tag{17}$$

(This measure is used for the experiments described in this paper.)

Modified Standard Deviation

$$f_{certainty}(p_1, p_2, \ldots, p_N) = \sqrt{\frac{n}{n-1} \sum_{i=1}^{N} (\frac{1}{N} - p_i)^2}. \tag{18}$$

Maximum

$f_{certainty}(p_1, p_2, \ldots, p_N)$ returns the largest p_i.

Variance

$f_{certainty}$ is equal to the variance.

$$f_{certainty}(p_1, p_2, \ldots, p_N) = Var(p_1, \ldots, p_N). \tag{19}$$

9.6 Using the Simulation Software

The simulator is written in C++ and compiles with `gcc` (in our case version 2.95.2). There is no graphical user interface yet. (A Java version with GUI is currently under development and will be available from our website.) Parameters are specified in a parameter file or as command line arguments. The result is written to various data files named `<runName>.<suffix>` where `<runName>` is name of the simulation experiment, which can be set by the user (default: `run`).

Usage

Call the simulation program using:

```
luhmann3 -pf <parameter-file-name>
```

You can also set parameters using command line arguments. Command line arguments given *after* the argument `-pf <parameter-file-name>` overwrite settings in the parameter file `<parameter-file-name>`. Running the simulation creates a bunch of data files named `<runName>.<suffix>`. The `runName` can be set as a parameter.

Example

Here is an example of a simulation experiment with $M = 20$ agents, which are allowed to use $N = 10$ activities. The system is simulated for 1000 single interactions (steps).

```
luhmann3 -experiment multiWorld -steps 1000 -M 20 -N 10
```

Output Files

The following log-files are the result of a simulation experiment:

file name	description
`run.adoc`	Automatic documentation file. Contains the seed of the random number generator, parameter settings, how the program has been called, and important messages such as the termination criterion.
`run.bm`	Average behavior matrix at the end of the simulation.
`run.bm05`	Binary average behavior matrix obtained with cutoff $\tau = 5\% = 0.05$.
`run.bm10`	Binary average behavior matrix obtained with cutoff $\tau = 10\% = 0.10$.
`run.bm15`	Binary average behavior matrix obtained with cutoff $\tau = 15\% = 0.15$.
`run.bmg`	The average behavior matrix as a list of nodes and weighted edges, which can be used for visualization of the communication system.
`run.fi`	Activity values. Every step is represented by one row, which contains the decision values used by the agent acting at that time step.
`run.msg`	Activities which are performed by the agents.
`run.op`	Systems level order measure O_AP.
`run.status`	Very detailed status log of the whole memory matrix of each agent. Switched off by default. Use `-writeStatus 1` to switch on.
`run.log`	Main log file for certainty and selected activity (see below).
`run.msgstat`	Message statistics, message diversity (see below).
`run.pf`	A list of parameters, which can be used as a parameter file.

runName.log The most important log-file. Here every step is represented by one row.

column	symbol	description
2	$MAX(w^i_{AV})$	the largest activity value of Agent A)
3	$f_{certainty}(w_{AV})$	certainty of the activity values of Agent A
4	$f_{certainty}(w_{AP})$	certainty of the activity probabilities of Agent A
5		activity selected by Agent A (starting with 0)
6		activity selected by Agent A (starting with 1)
8	$\max(w^i_{AV})$	the largest activity value of Agent B)
9	$f_{certainty}(w_{AV})$	certainty of the activity values of Agent B
10	$f_{certainty}(w_{AP})$	certainty of the activity probabilities of Agent B
11		activity selected by Agent B (starting with 0)
12		activity selected by Agent B (starting with 1)

runName.msgstat Message statistics. Here the number of different messages which appear during a time interval is stored. At the end of the file the average is written in the format: `# average 27.2137`
So you can get the average number of different messages, e.g., by `grep average runName.msgstat`.
There are two parameters for the message statistics:

```
# intervalSize 50
# startAverage 500
```

They can be set, as usual, in the parameter file or by command line attributes.

Parameter File

The **parameter file** may look like this:

```
experiment       multiWorld
steps    1000
M        20
N        10
runName  run
seed     0
vicinity         0
useAlterMemoryForEE      1
rngNo    0
observers        0
trace    0
depth1   0
depth2   0
alpha    1.000000
cf       0.020000
cM       2.000000
selectionMethod  0
writeLog         1
writeStatus      0
writeOP  1
certainty        entropy
memory   05
forgetrate       0.010000
learnrate        0.100000
intervalSize     50
startAverage     500
```

The parameters have the following meaning:

par. name in program	symbol in paper	meaningful values	description
`experiment`		`dyadicWorld` `multiWorld`	Select the experiment type. In the "dyadic world" only two agents are present acting alternately. In the "multi world" two or more agents can be present interacting randomly (see above).
`steps`		50 - 10000	Number of simulation steps. One step consists of one activity selection step.
`runName`			The run name. All output is stored in data files named `<runName>.<suffix>`.
`N`	N	2 - 100	Number of activities (messages, symbols).
`M`	M	2 - 100	Number of agents present in the "multi world".
`observers`	n	0 - 5	Number of observers.
`alpha`	α	0.0 - 1.0	Fraction of expectation certainty. $\alpha = 0.0$: only expectation-expectation (EE) is used for activity selection. *alpha* = 1.0: only expectation-certainty is used for activity selection.
`certainty`		`entropy` `modifiedStddev` `selectMaximum` `variance`	Method for measuring the certainty.
`selectionMethod`		0,1,2,3,4	Selection method. 0: Select maximum. 1: Select proportional (equal to $\gamma = 1$). 2: select squared proportional (equal to $\gamma = 2$). 3: equal to *gamma* = 4. 4: selection method using `gamma` as an exponent.
`gamma`	γ	0.0 - 40.0	Exponent for selection method 4.
`memory`		01, 02, 03, 04	The memory model.
`memSize`		10 - 200	The memory size. Only valid for memory model 03 and 04.
`memGamma`	γ_{mem}	0.98	Recall rate. Only valid for memory model 01 and 02. Low value is equivalent to high forgetting rate.
`cM`	c_M	2.0	Special constant. See Eq. (15).
`cf`	c_f	0.02	Special constant. See Eq. (2).
`vicinity`		0	The size of the vicinity in a ring topology. `vicinity`=0: No topology. `vicinity`=1: Only direct neighbors on the ring can interact. (Not used here).
`seed`		0, any number	The seed for the pseudo random number generator. 0: create seed automatically using `time(0)`.
`rngNo`		0, 1	The type of the random number generator. 0: `drand48`, a linear congruence generator. 1: rand2 from "*Numerical Recipes in C*", a nonlinear matrix generator.

References

Shannon, C. E. and W. Weaver (1949). *The Mathematical Theory of Communication* (fourth printing, 1969 ed.). Urbana: University of Illinois Press.