Timeliness Criticality

(It’s fine until it isn’t)

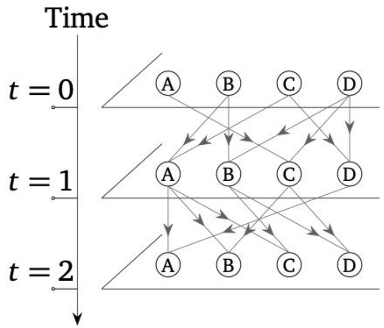
Based on the paper “Timeliness criticality in complex systems” by José Moran et al.
(published in *Nature Physics* 20, pages 1352–1358 (June 19, 2024)

What is Timeliness Criticality?

Efficiency

Resilience

- Socio-technical networks like transport or supply chains depend on timing
- Trade-off: efficiency (tight schedules) vs resilience (large buffers)



Question: Is there a critical threshold?

→ Moran et al. (2024) predict a **second-order phase transition at a critical Buffer size B_c***

Model and Key Terms

Noise ε ~ Exp(1) adds random perturbations

Delay of a node

$$\tau_i(t) = \left[\boxed{\max_j [A_{ij}(t-1)\tau_j(t-1)]} - B\right]^+ + \varepsilon_i(t)$$

Delay propagates through K random neighbors (Mean Field): inherited delay is the max delay of a neighbour (Markovian)

Buffer B absorbs delays up to a threshold: fixed and uniform across nodes

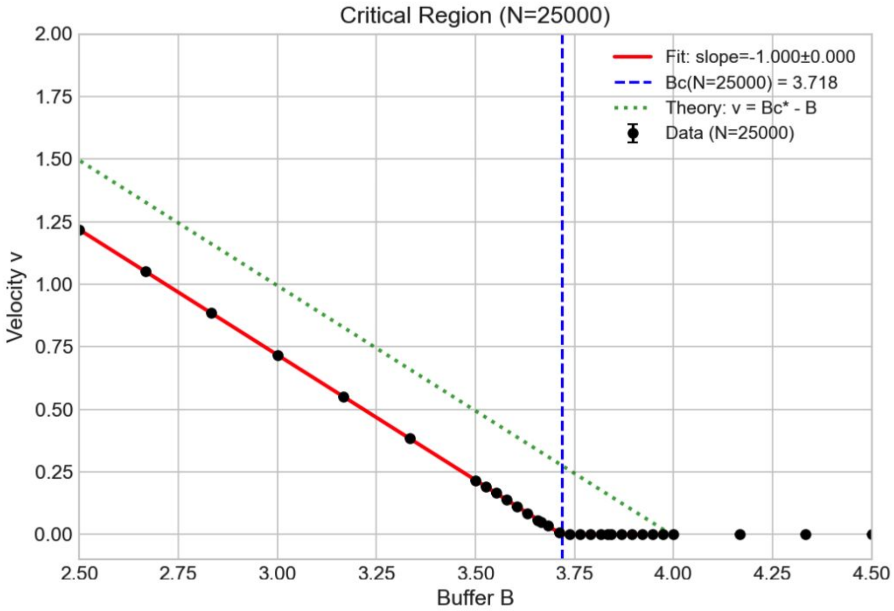
Key terms:

- Delay of node: τ
- Buffer B and Critical Buffer B_c*
- Avalanche: average node delay > Buffer size
- Velocity: $v = \mathbb{E}[\tau(t) - \tau(t-1)]$

The aim: computational validation of analytical theory

Phase transition in delay accumulation

Fitted slope = -1.0000 ± 0.0002
Theory predicts slope = -1.0
R^2 = 1.0000



- The system velocity v=E[τ(t)−τ(t−1)] acts as the order parameter for the phase transition:

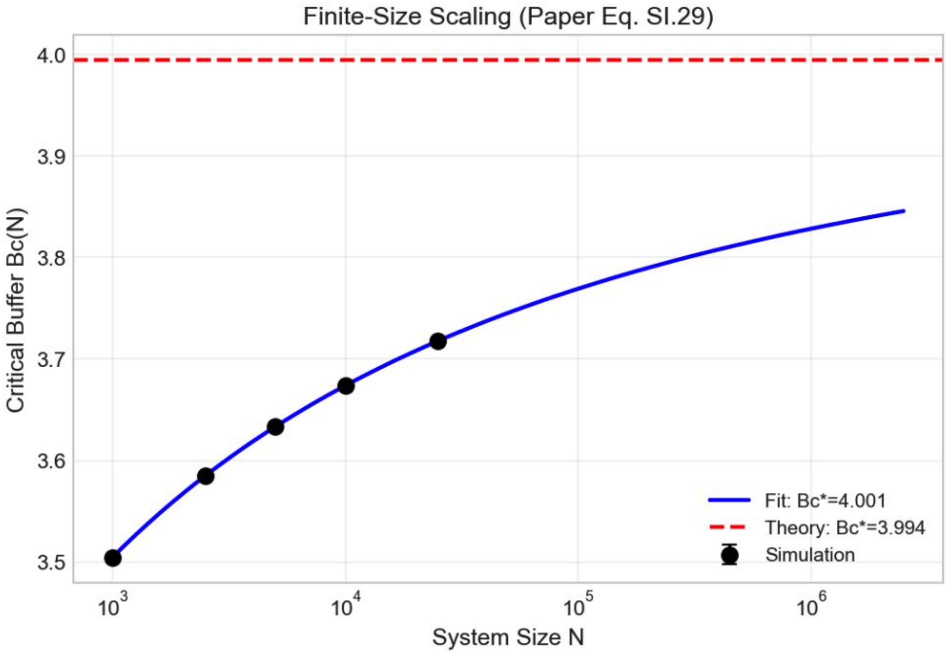
$$v = \begin{cases} 0 & \text{if } B > B_c^* \\ B_c^* - B & \text{if } B < B_c^* \end{cases}$$

- **Implication:** In the moving phase (B<B_c*), the slope of the velocity is constant: dB/dv=−1

✅

Finite-size scaling

Fitted parameters:
Bc*(∞) = 4.00139 ± 0.00380
Theory prediction: Bc* = 3.99431
Deviation: 1.86 σ



- The critical buffer size B_c* is determined by the mean-field connectivity K:

$$B_c^* \exp(1 - B_c^*) = \frac{1}{K}$$

- The analytical solution is given by the Lambert W function:

$$B_c^* = -W_{-1}\left(-\frac{1}{eK}\right)$$

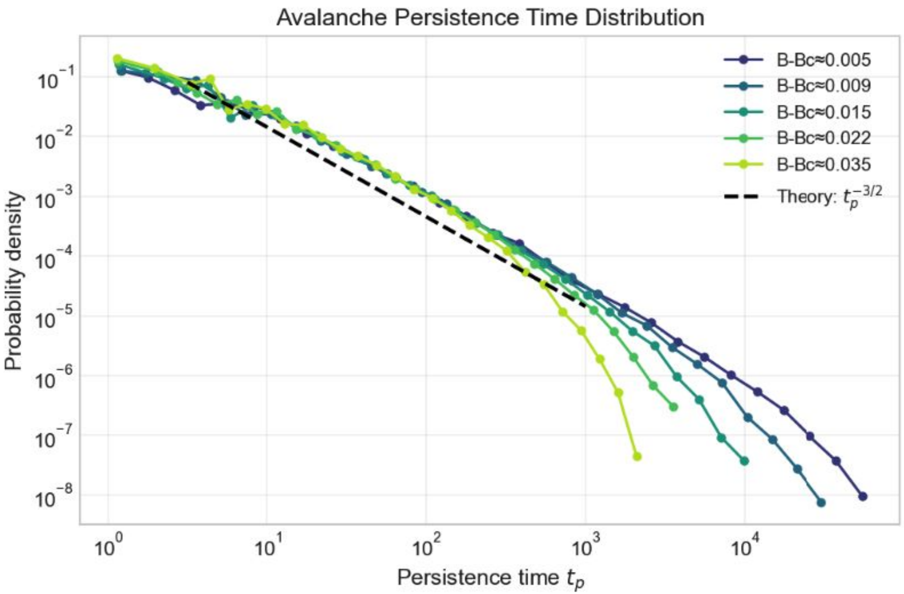
- For a system of finite size N, the critical buffer B_c(N) deviates from the theoretical limit B_c* according to a correction:

$$B_c(N) = B_c^* - \frac{1}{(a + b \ln N)^2}$$

- This predicts a very slow convergence to the thermodynamic limit, scaling as (lnN)**(−2). ✅

Avalanche Persistence

Persistence time power-law test:
B=3.638: exponent = 1.41 ± 0.03 (theory: 1.5)
$R^2 = 0.991$



- Near criticality, the distribution of persistence times (avalanches) follows a power law characteristic of an unbiased random walk:

$$P(t_p) \sim t_p^{-3/2}$$

- **Finite-Size Cutoff:** in simulations, the system boundaries truncate the distribution tail, distorting the slope
- **Tail Undersampling:** Long-duration avalanches are rare events; difficult to gather enough data points

~✅

Conclusion (spoiler: the pro's got the math right)

- Analytical solution to mean field theory has been supported:

Test	Theory	Result	Status
Slope dv/dB	-1	-1.0000 ± 0.0002	✓
B_c*	3.994	4.001 ± 0.004	✓ (1.86σ)
Avalanche exponent	1.5	1.41 ± 0.03	~✓

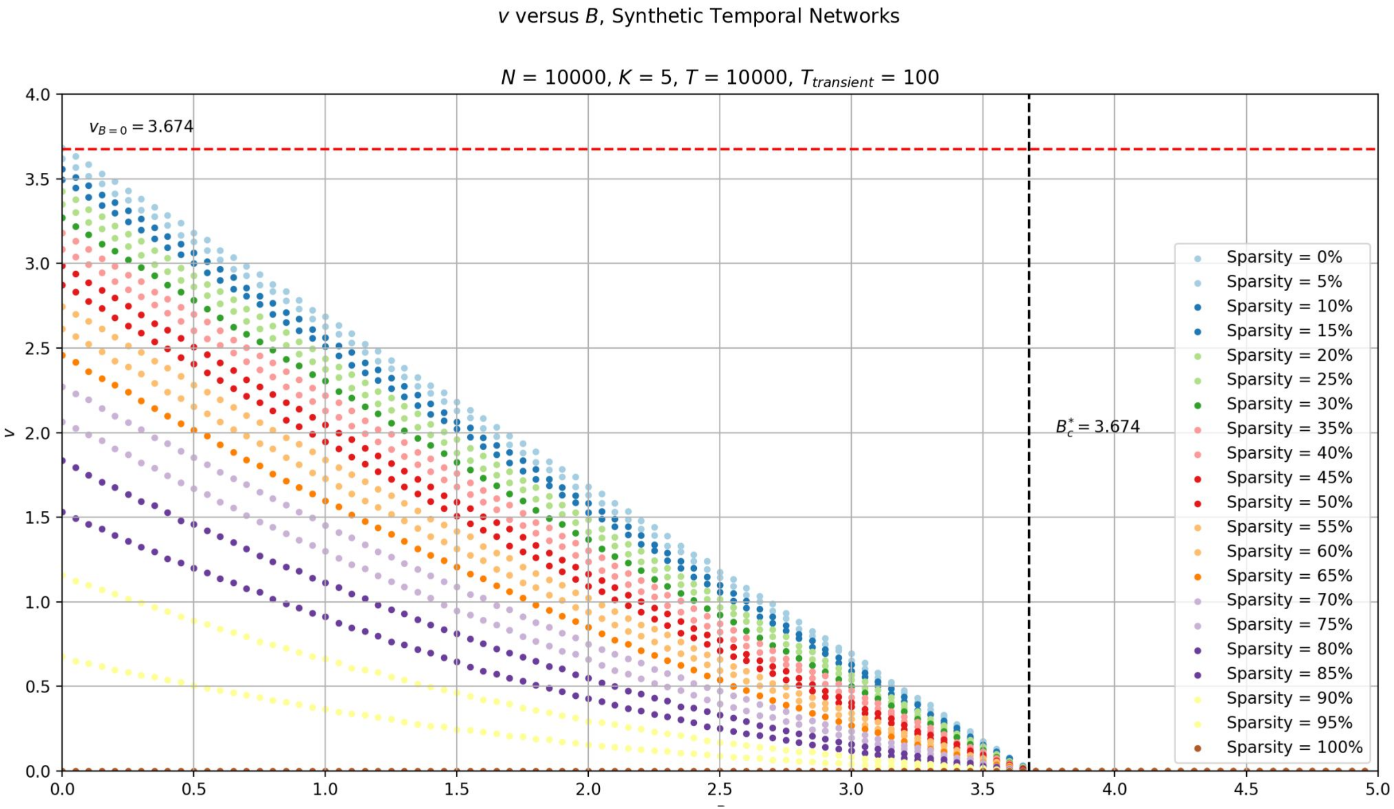
- Deviations explained by finite-N effects
- Mean Field model accurately describes timeliness criticality
- Limitations:
  a. What about real world networks? → Laszlo
  b. Exponential random noise may be unrealistic → Dan
  c. Actors may adjust buffers dynamically and locally → Matteo

Moving to real-world scenarios: do similar effects occur?

- Current: every node participates each iteration with delay probability: eps ~ Exp[-eps]
    - Introduction of sparsity
    - Hypothesis: smoothed out second order phase transition

- Current: fully random <k> connections each iteration
    - We examine real-world temporal networks into the model

Results: STNs with varying sparsity



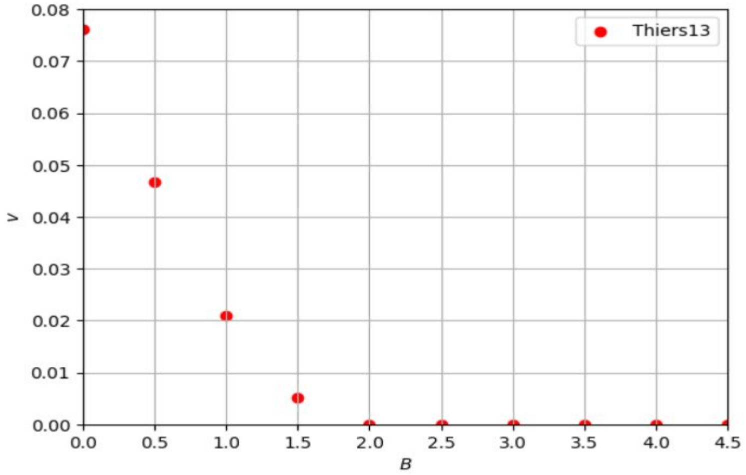
Note:

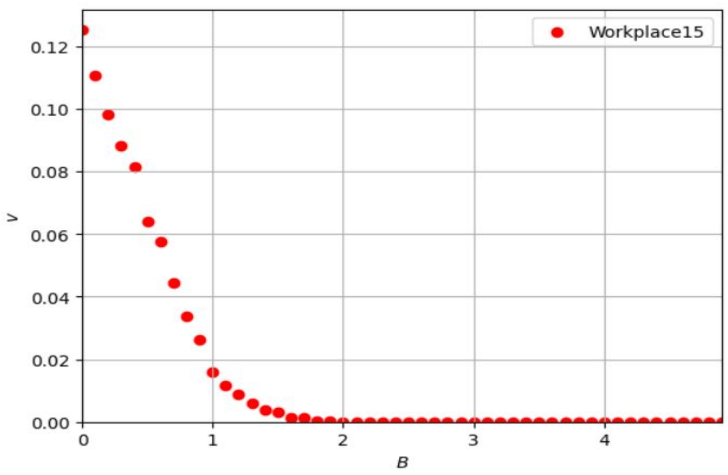
- B_c unaffected
- Smoothed out second-order phase transition
- Lower mean delay at v(B = 0)

Verification on real world networks

Applied to real-world social networks from sociopatterns.org, agent temporal contact events measured

- Events are analogous to supply-propagation between firms
- High school proximity network (Thiers13)
- Office building face-to-face contacts (Workplace15)



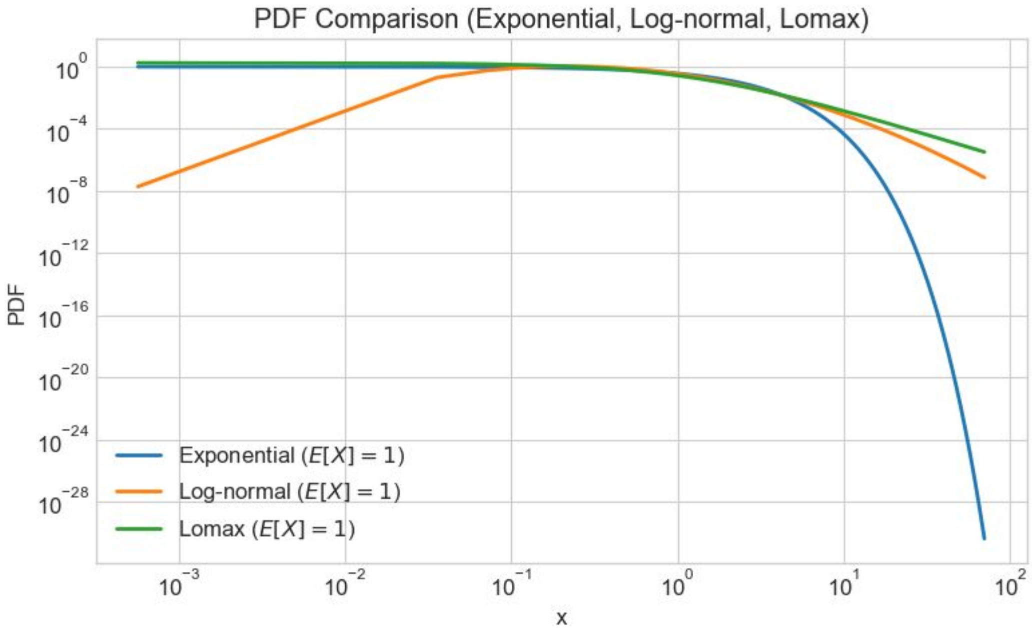


Note: smoothed out phase transition and lower B_C - affected by <k> among other factors

- Very low v(B=0)

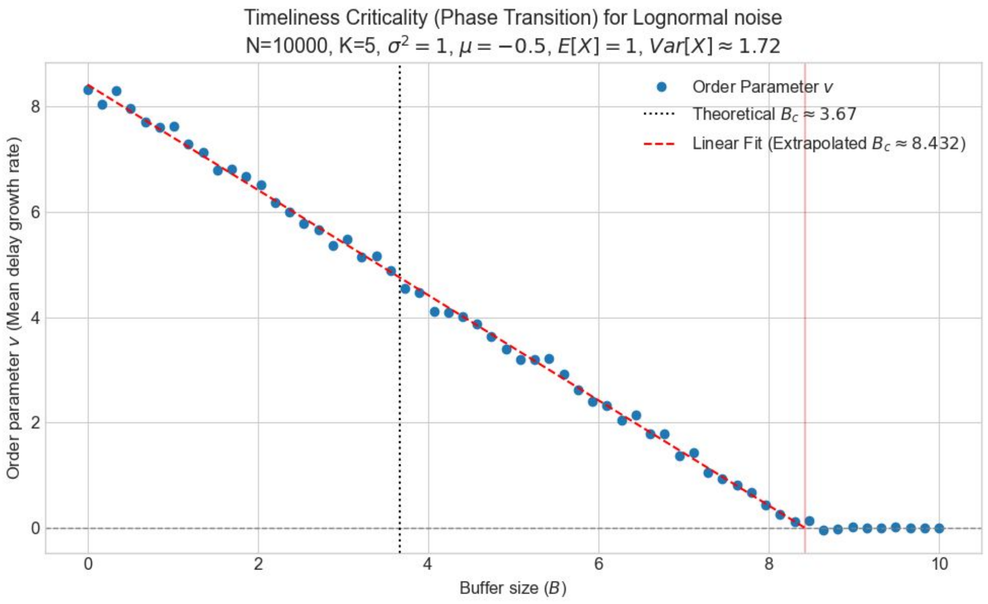
Suggests further research into e.g. existing supply chain networks

Different approaches to model the delays



- So far: delays as Markovian events with an exponential distr. noise term
- What would happen if extreme delays are more likely to happen?
- → need “heavy-tailed” distribution: Log-normal and Lomax

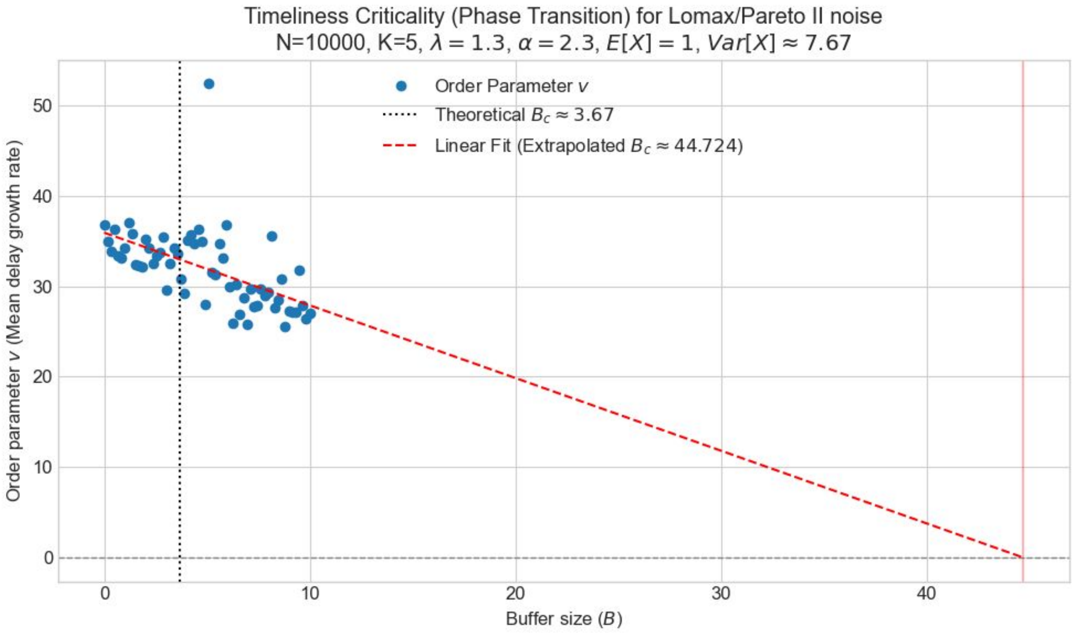
Phase transitions for different types of noise: Lognormal



- For Lognormal noise we observe that the transition stays linear
- Decays at a slower pace
- Increased variability, since extreme values are more probable

$$p(x) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\left(-\frac{(\ln x - \mu)^2}{2\sigma^2}\right)$$

Phase transitions for different types of noise: Lomax



- Lomax distribution has high variance → increased velocity and even slower decay → larger buffer.
- Decay is no longer as linear

$$p(x) = \frac{\alpha}{\lambda}\left(1 + \frac{x}{\lambda}\right)^{-(\alpha+1)}, \qquad x \geq 0,$$

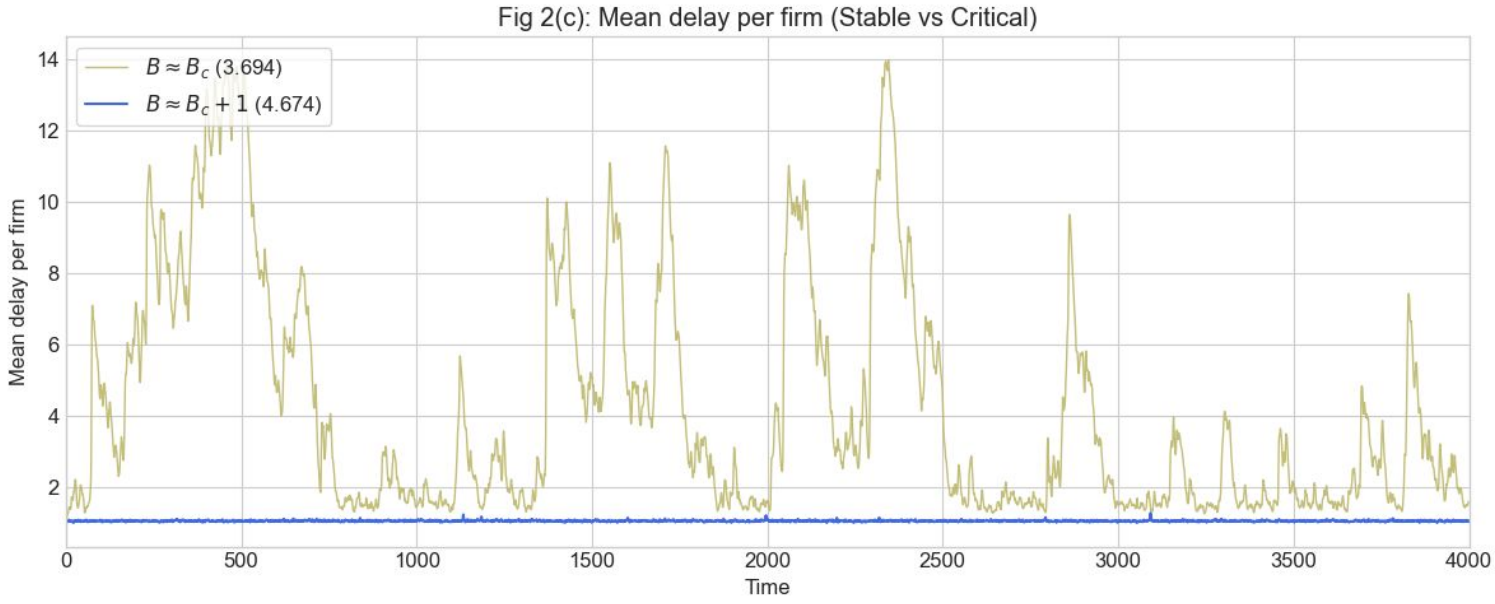
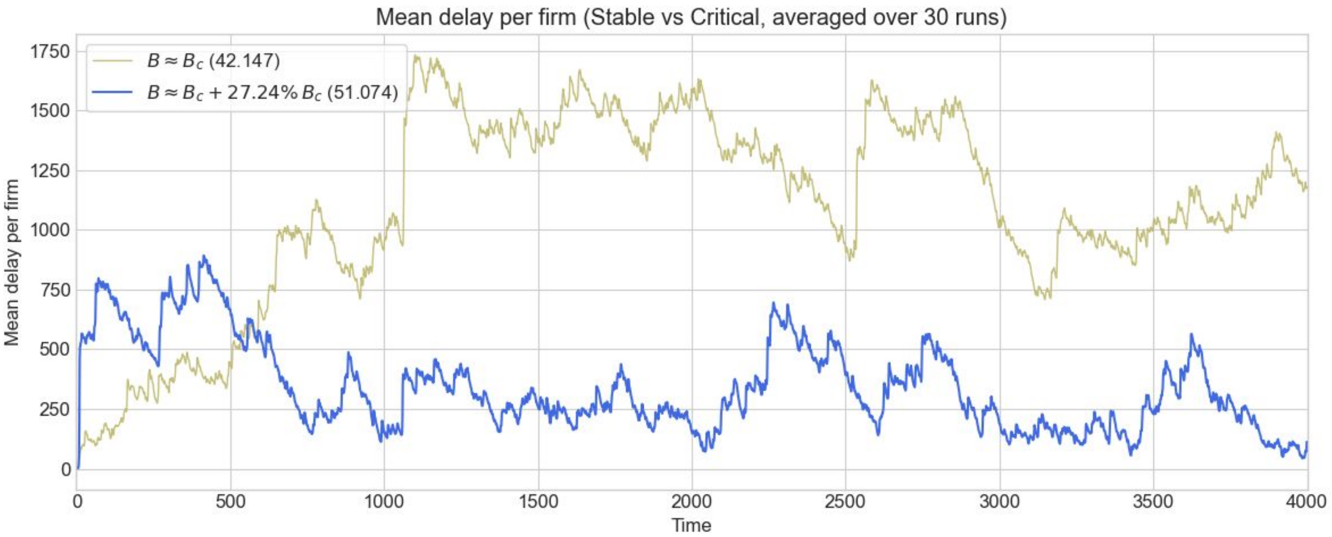
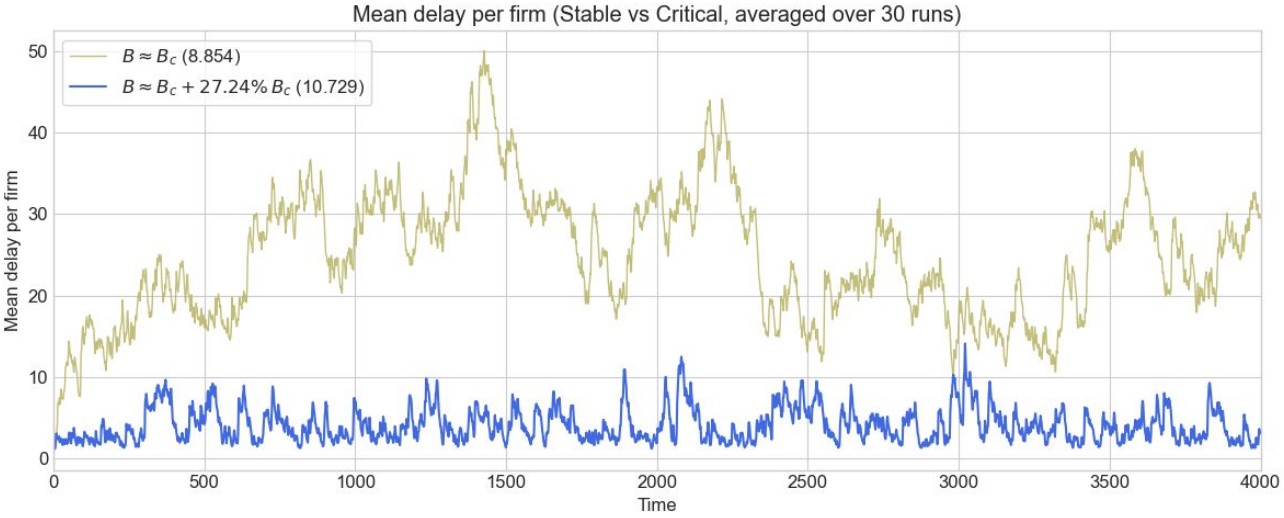
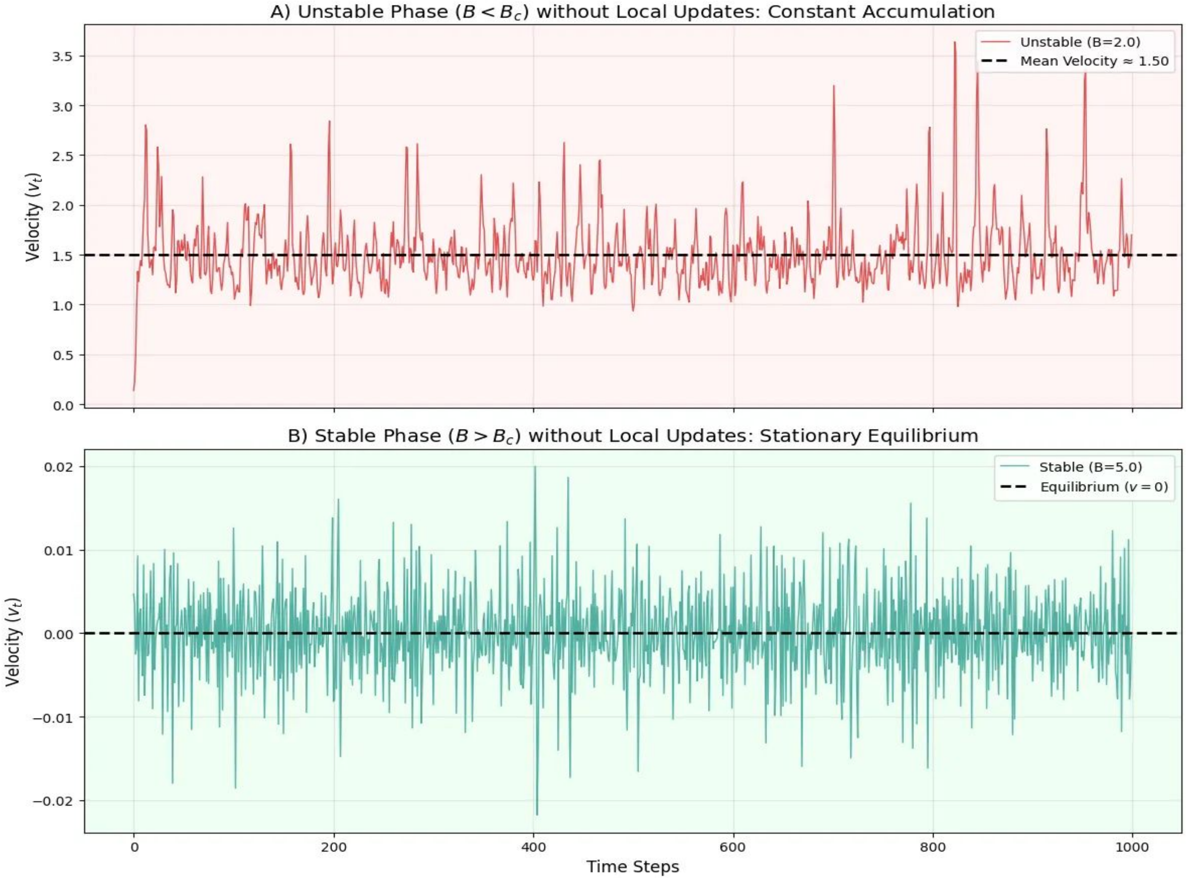


Fig 2(c): Mean delay per firm (Stable vs Critical)
$B \approx B_c$ (3.694)
$B \approx B_c + 1$ (4.674)
Mean delay per firm
Time
0
500
1000
1500
2000
2500
3000
3500
4000
2
4
6
8
10
12
14


Mean delay per firm (Stable vs Critical, averaged over 30 runs)
$B \approx B_c$ (42.147)
$B \approx B_c + 27.24\% \, B_c$ (51.074)
Mean delay per firm
Time


Mean delay per firm (Stable vs Critical, averaged over 30 runs)
$B \approx B_c$ (8.854)
$B \approx B_c + 27.24\% \, B_c$ (10.729)
Mean delay per firm
Time

The Static Trap: All or Nothing



- The Mean Field model system has no "brain" (No update rules).
- That’s why the outcome depends entirely on the starting Buffer: if it’s bigger than the critical size delays get absorbed, while, if it is smaller, delays accumulate.

The Solution: Self-Organized Criticality (SOC)

What if our system could adapt following some local update rules?
What if they enabled the system to autonomously gravitate towards a particular state, right in the middle between order and chaos, without any other external events and independently of the starting conditions?

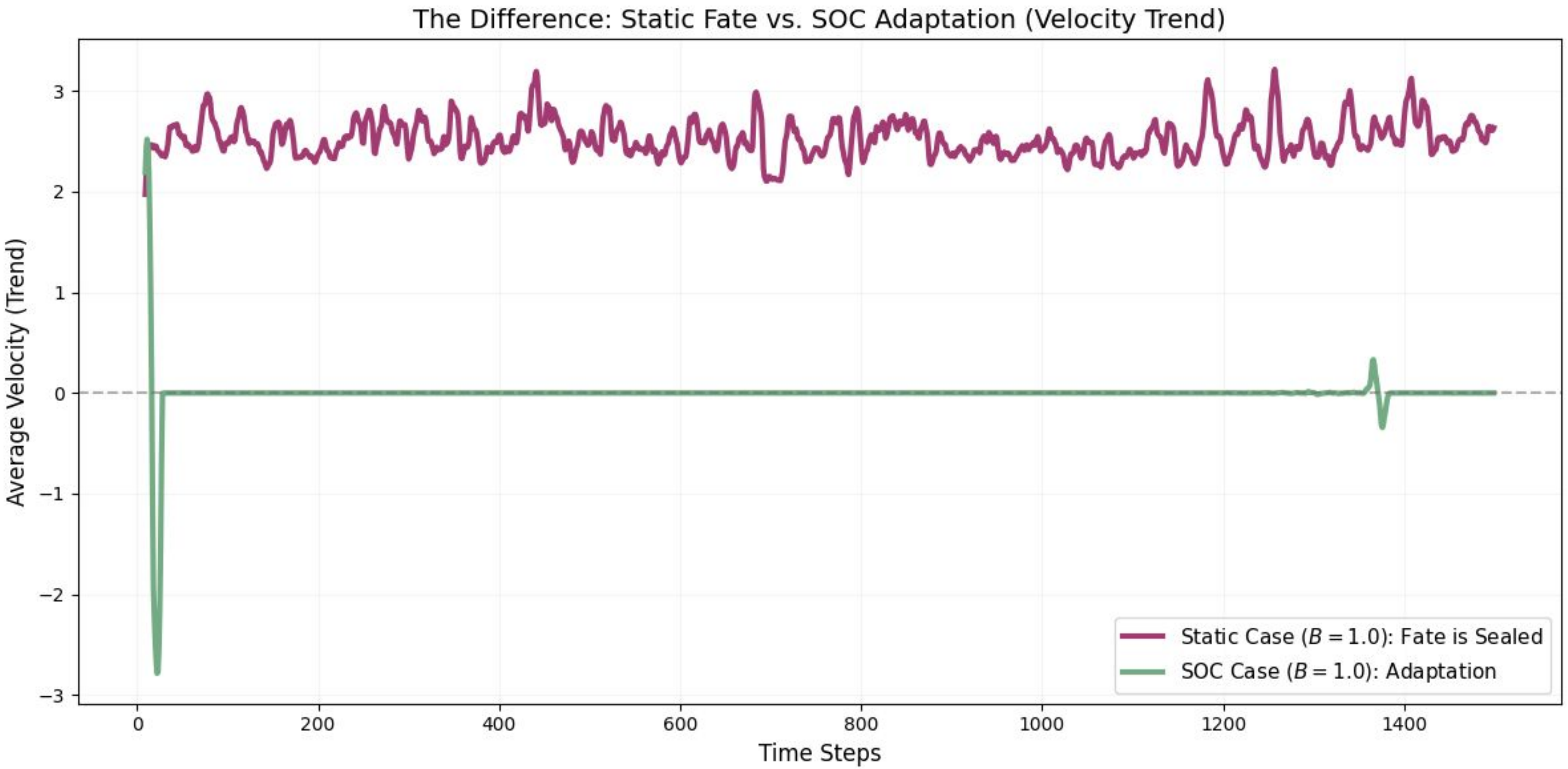
We just discovered Self-Organized Criticality!

Why is it important?
It optimizes the trade-off between efficiency and resilience.

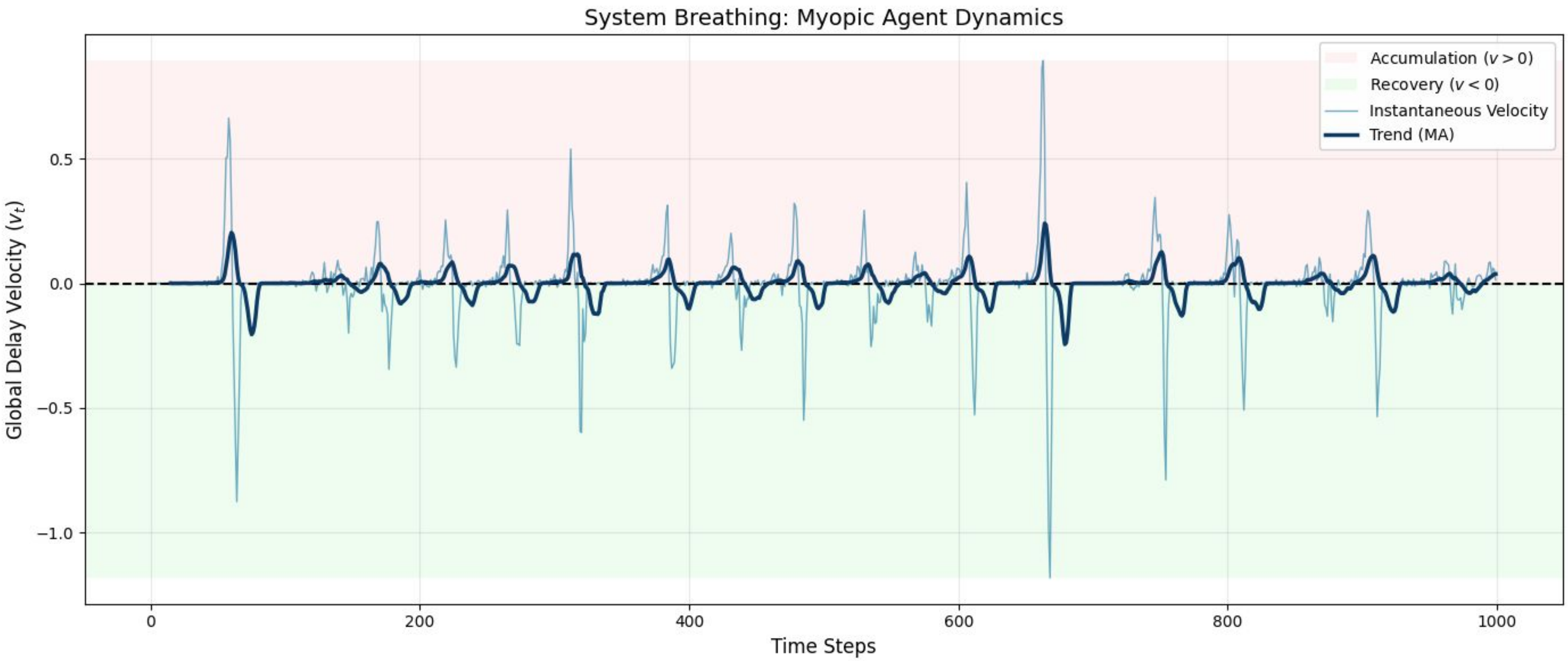
Local Rules

$$\text{Update} = \begin{cases} \underbrace{\text{Reaction}(\alpha) - \text{Memory}(\beta)}_{\text{Fix the Error}} & \text{if} \text{Delay} > 0 \ {\color{red}(\text{Recovery Phase})} \\ \underbrace{-\text{Efficiency}(\epsilon)}_{\text{Cut Costs}} & \text{if} \text{Delay} = 0 \ {\color{green}(\text{Optimization Phase})} \end{cases}$$

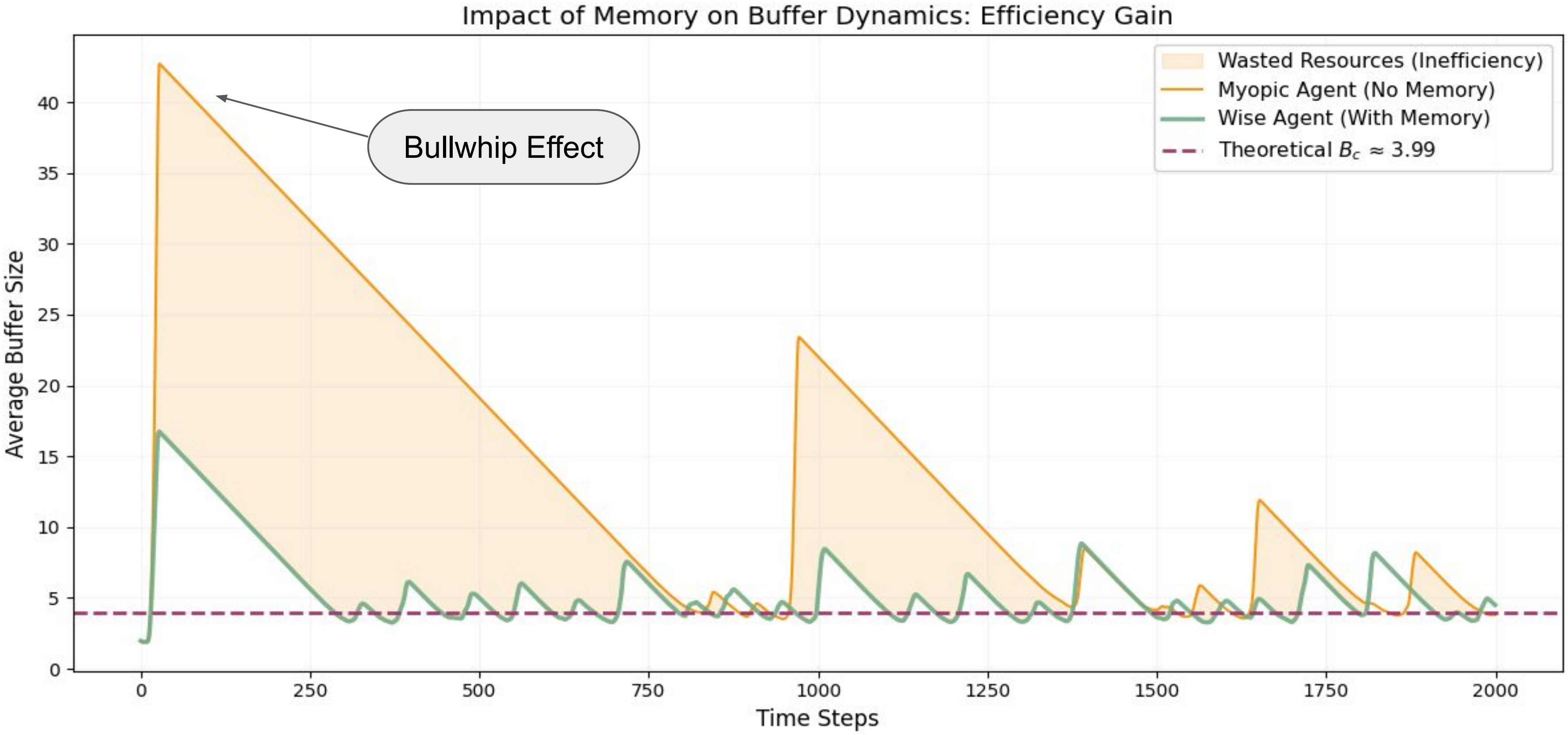
The Power Of Adaptation



Dynamic Stability: Panic vs. Recovery



The Cost of Panic vs The Value of Patience



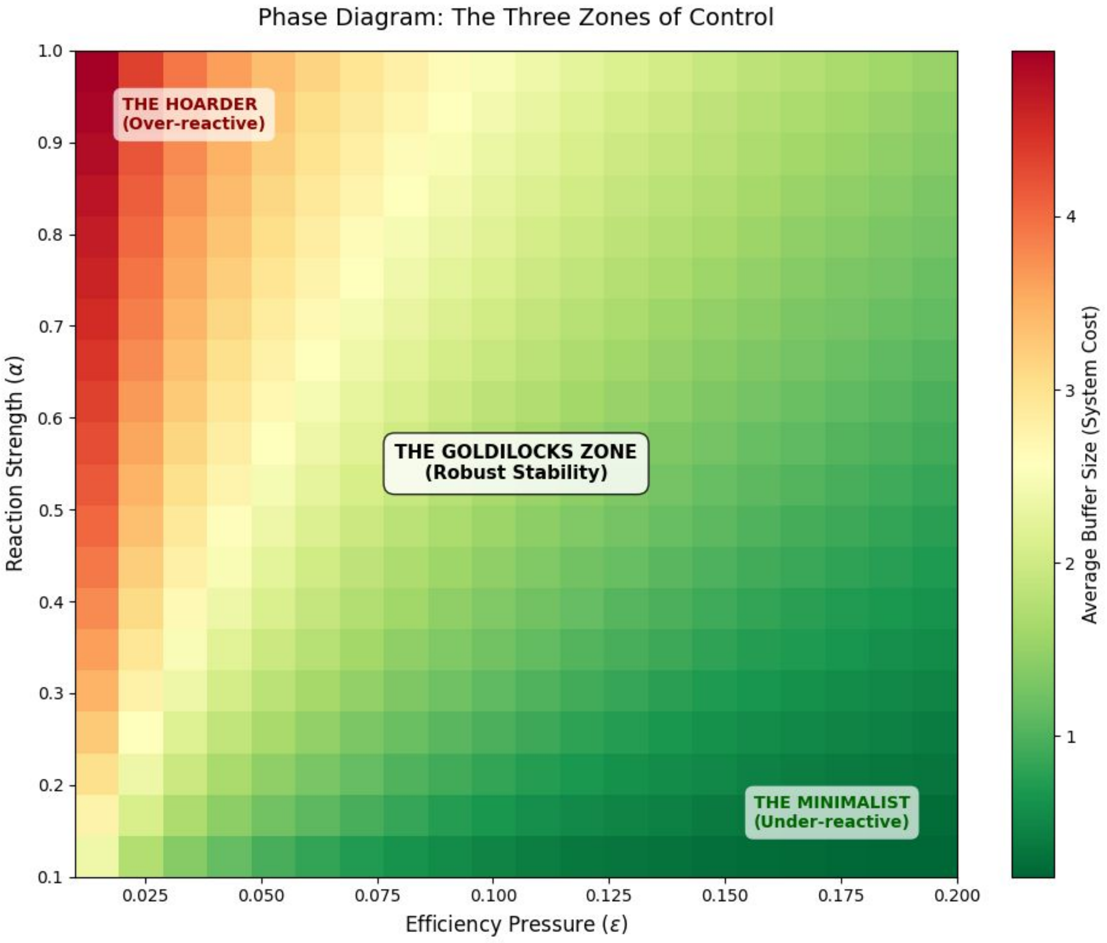
Do we always reach SOC?

High Reaction

Not all local rules are equal!

We can observe, in fact, three distinct zones depending on the parameters used.

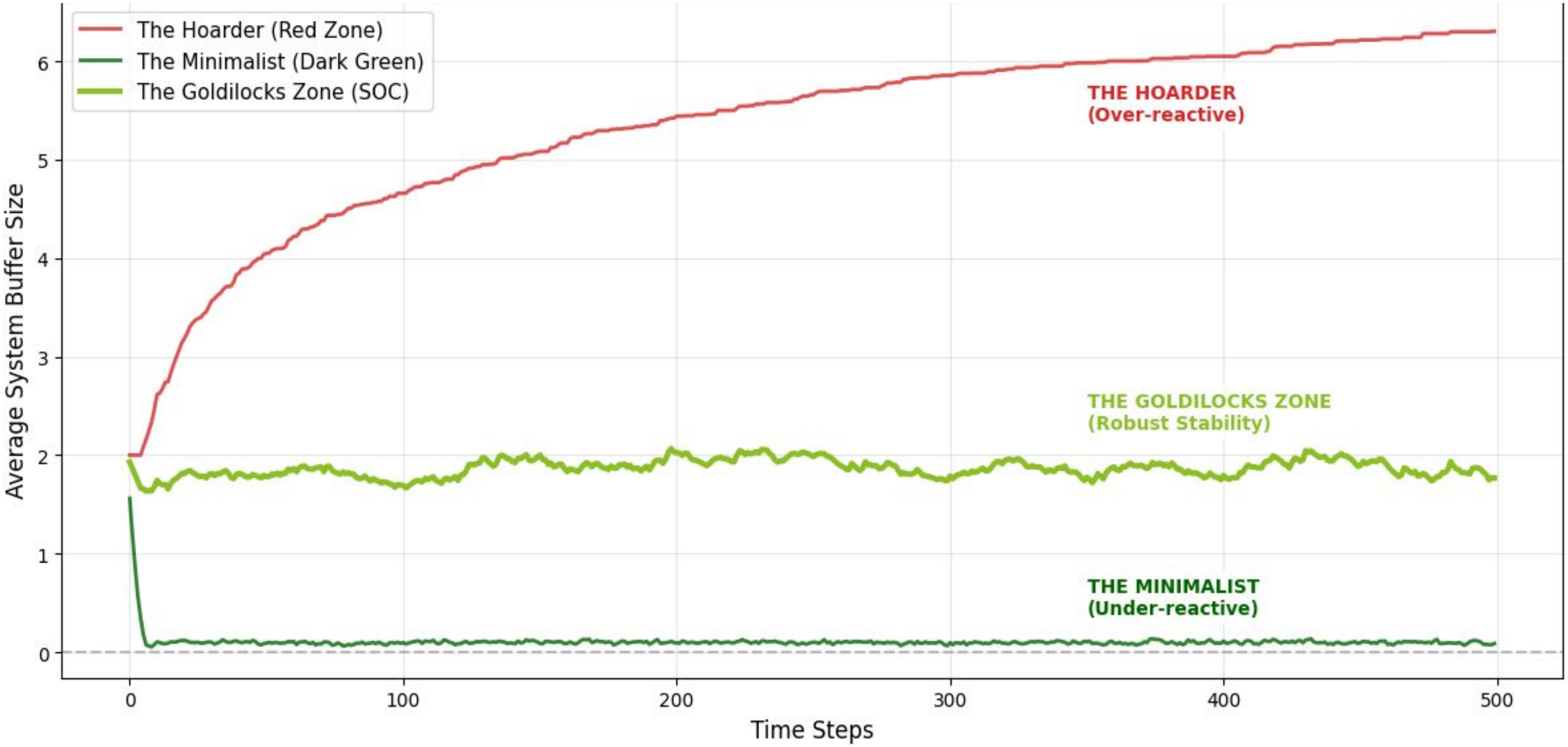
Low Reaction



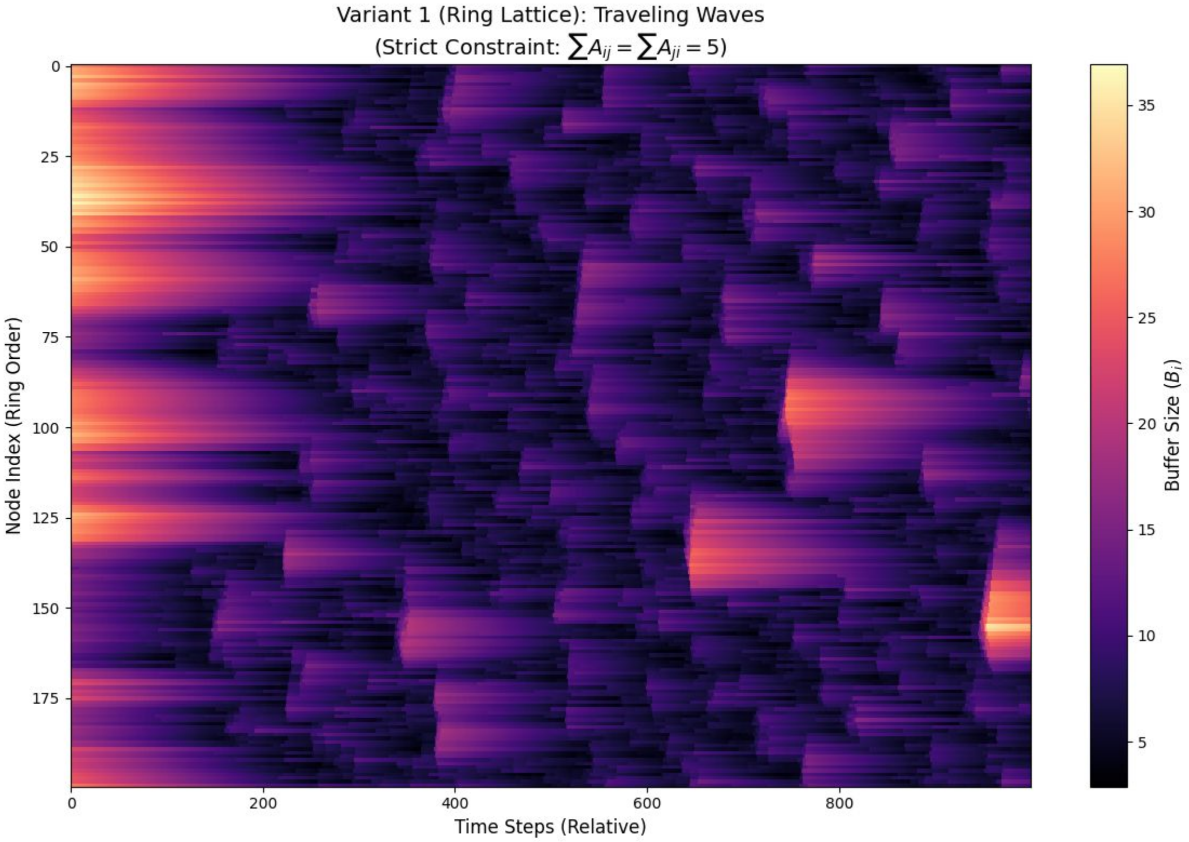
Zero Efficiency

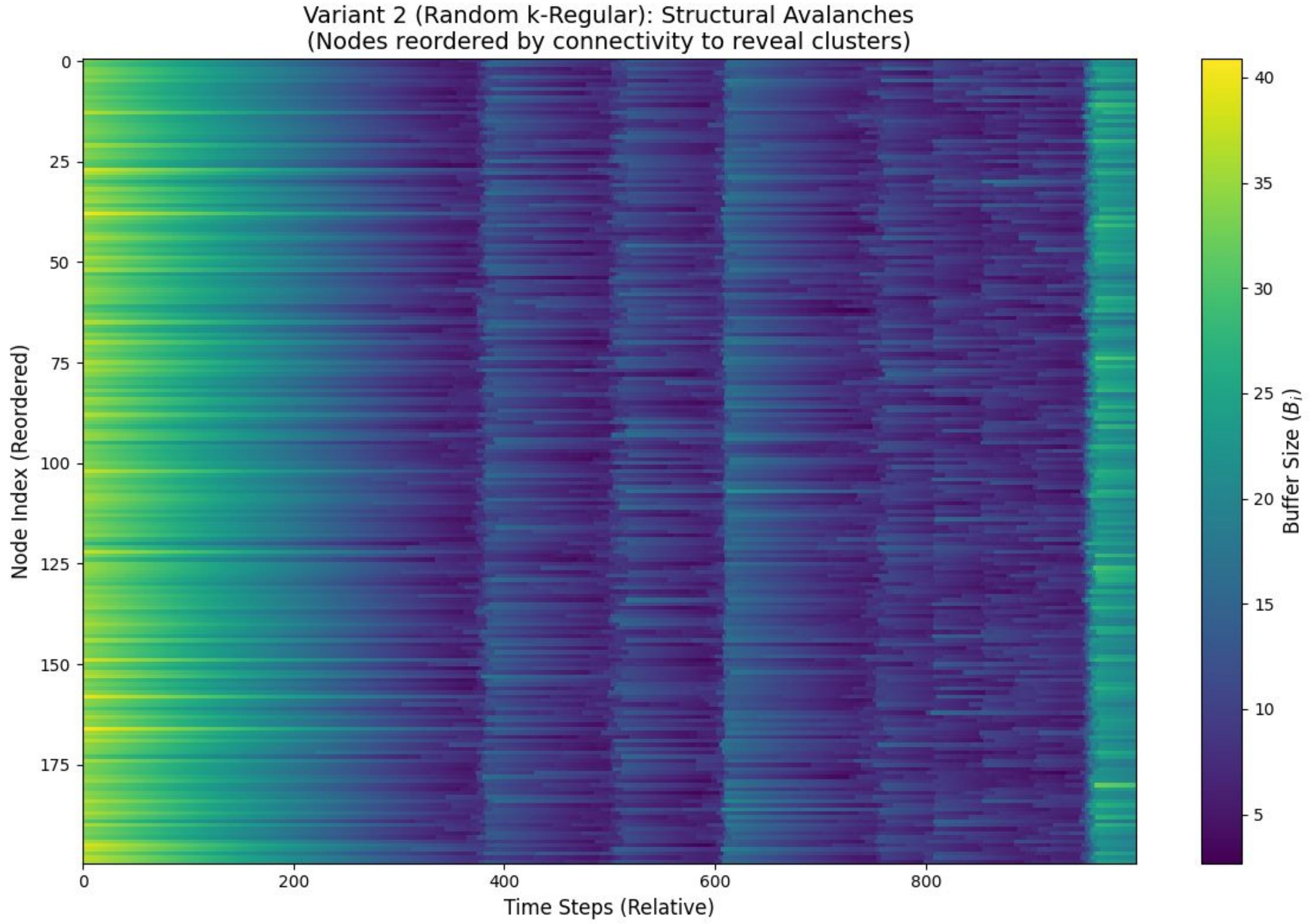
Aggressive Cost-Cutting

Time Evolution of the Three Zones.



Now that we know the rules, let's study networks with a fixed geometric structure…





Variant 2 (Random k-Regular): Structural Avalanches
(Nodes reordered by connectivity to reveal clusters)
Node Index (Reordered)
Time Steps (Relative)
Buffer Size (B_i)
0
25
50
75
100
125
150
175
200
400
600
800
5
10
15
20
25
30
35
40

The Local Rules decide *if* we survive, but the topology decides *how* we fail, either through smooth waves or sudden crashes.

Conclusion

- Analytical mean field theory was validated through simulation → delay propagation shows second-order transition at critical buffer size
- Increasing sparsity smooths out second-order transition → as do real networks
- Changing the delays distribution destroys the previous assumption of stability around the critical buffersize
- Local rules allow for the emergence of SOC, but do not guarantee it → Memory is essential to dumpen fluctuations

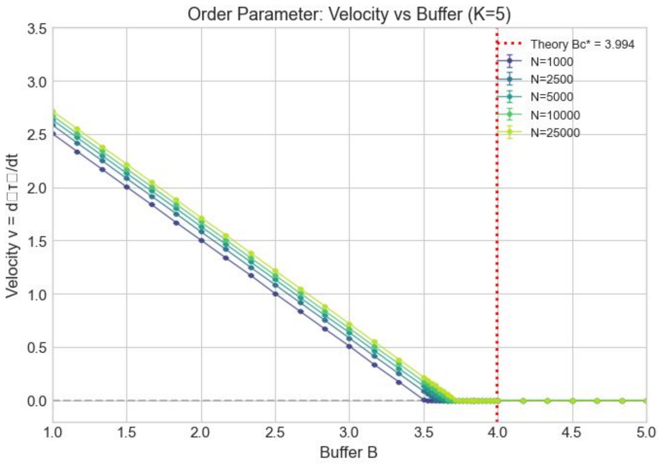
Thank you for listening!

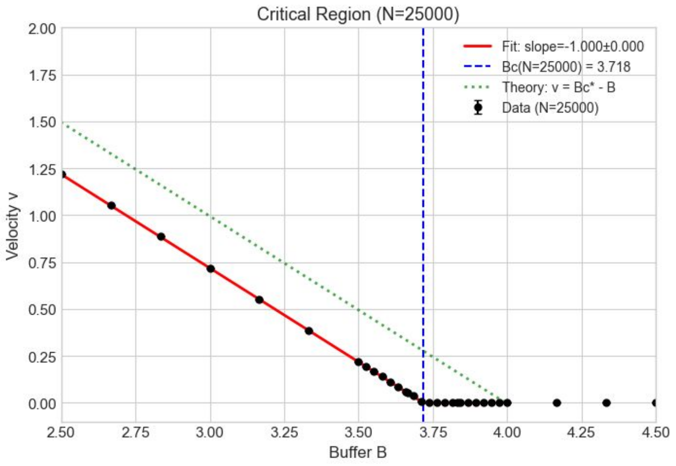
References

- Moran, José, Frank P. Pijpers, Utz Weitzel, Jean-Philippe Bouchaud, and Debabrata Panja. ‘Critical Fragility in Socio-Technical Systems’. *Proceedings of the National Academy of Sciences* 122, no. 9 (2025): e2415139122. https://doi.org/10.1073/pnas.2415139122.
- Moran, José, Matthijs Romeijnders, Pierre Le Doussal, et al. ‘Timeliness Criticality in Complex Systems’. *Nature Physics* 20, no. 8 (2024): 1352–58. https://doi.org/10.1038/s41567-024-02525-w.
- Modeling Managerial Behavior: Misperceptions of Feedback in a Dynamic Decision Making Experiment, Author(s): John D. Sterman, Source: Management Science, Vol. 35, No. 3 (Mar., 1989), pp. 321-339, Published by: INFORMS, Stable URL: http://www.jstor.org/

Appendix

Appendix





==

HYPOTHESIS TEST H1: Slope dv/dB = -1

==

N	Slope	SE	t-stat	p-value	Result
1000	-0.9996	0.0007	0.64	0.5249	✓ ACCEPT H0
2500	-1.0000	0.0004	0.04	0.9654	✓ ACCEPT H0
5000	-1.0003	0.0003	-1.06	0.2881	✓ ACCEPT H0
10000	-0.9997	0.0003	0.94	0.3464	✓ ACCEPT H0
25000	-1.0000	0.0002	-0.13	0.8955	✓ ACCEPT H0

Conclusion: H0 states slope = -1 (theory prediction)
If $p > 0.05$, we cannot reject H0, supporting the theory.

Weighted average slope: -1.0000 ± 0.0001
Deviation from -1: 0.0000

Appendix

Bc estimates by system size:

N	Bc(N)	Error
1000	3.5048	0.0003
2500	3.5845	0.0003
5000	3.6333	0.0002
10000	3.6739	0.0002
25000	3.7176	0.0002

==

FINITE-SIZE SCALING FIT (Paper Eq. SI.29)

Bc(N) = Bc* - 1/(a + b*ln(N))²

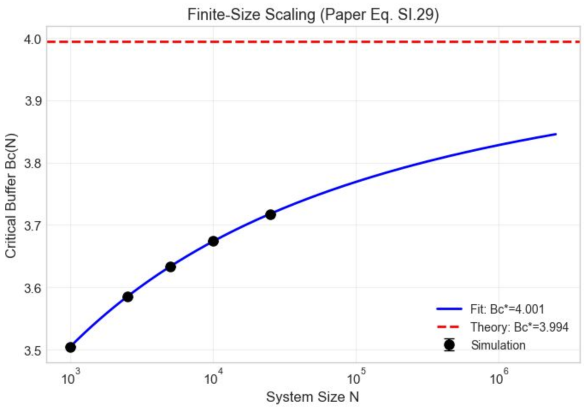
==

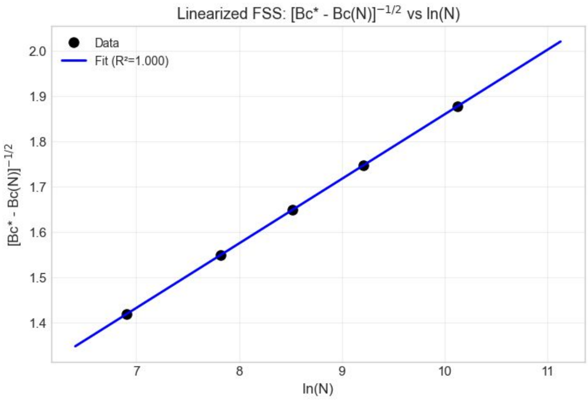
Fitted parameters:

- Bc*(∞) = 4.00139 ± 0.00380
- a = 0.4347 ± 0.0097
- b = 0.1425 ± 0.0022
- χ² = 1.91

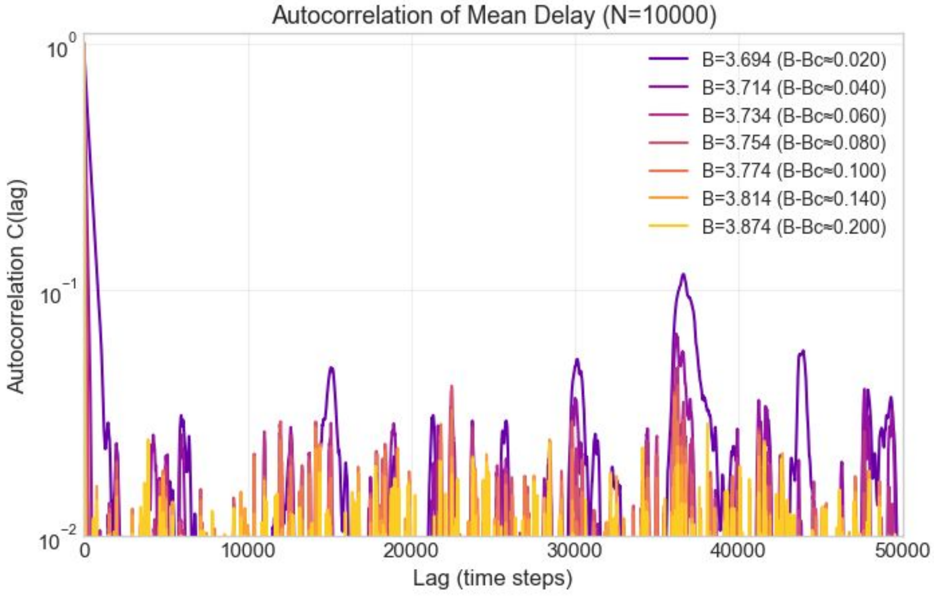
Theory prediction: Bc* = 3.99431

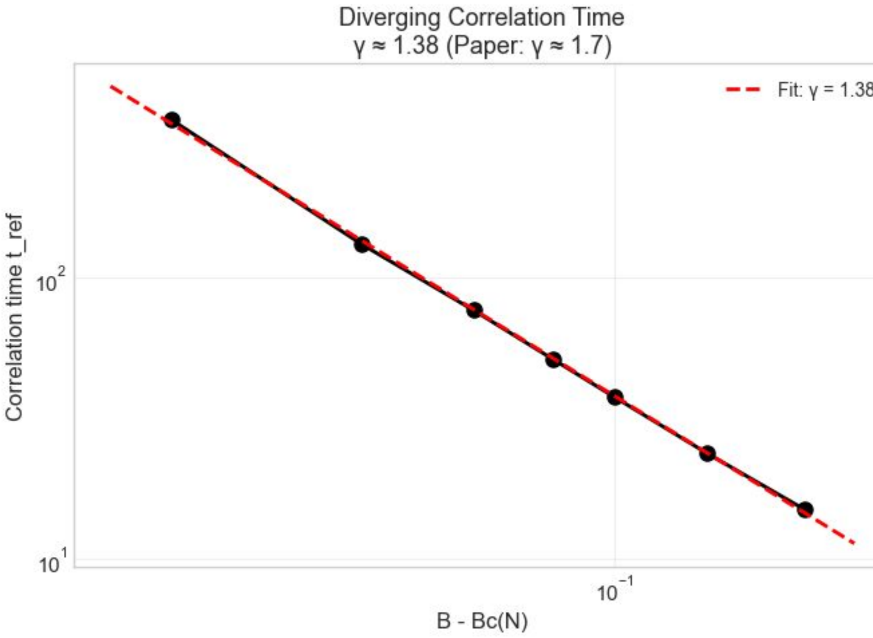
Deviation: 1.86 σ





Appendix

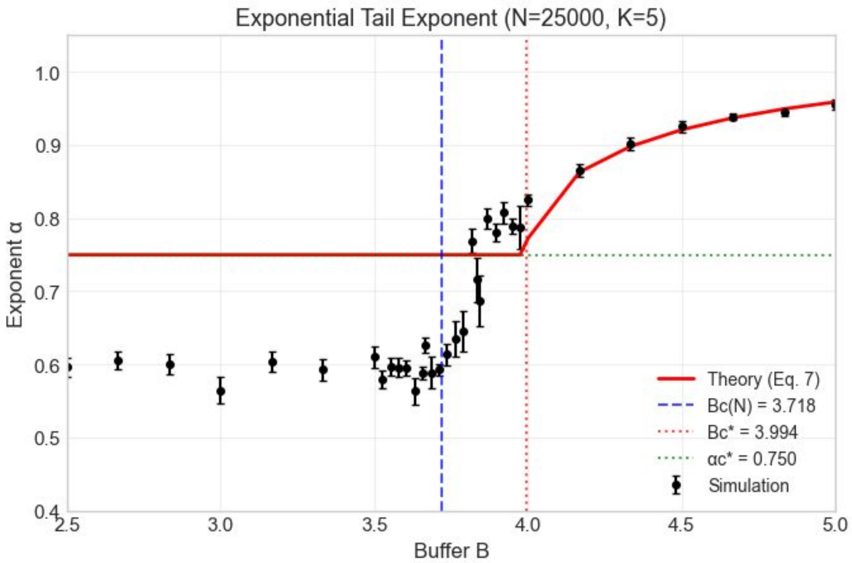


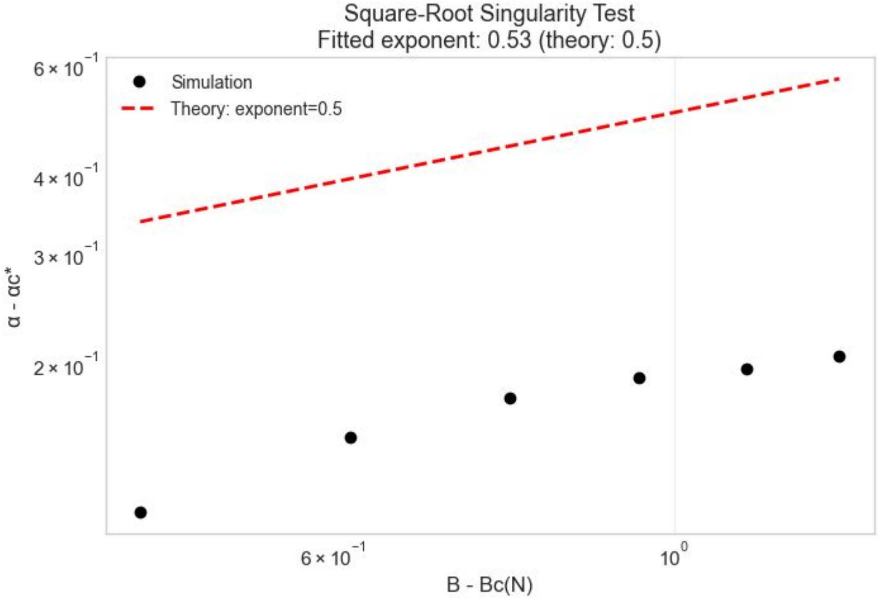


Correlation time exponent: $\gamma = 1.38$

Paper reports $\gamma \approx 1.69$ for K=5

Appendix





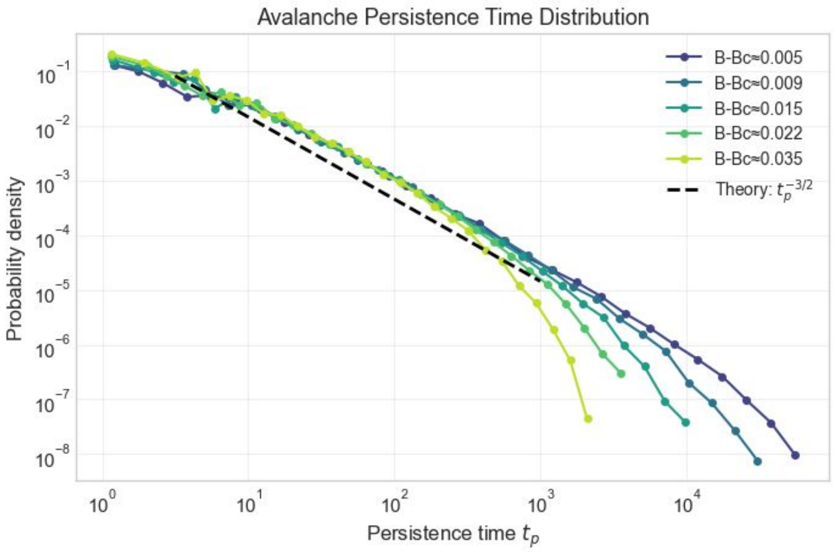
Square-root singularity test:

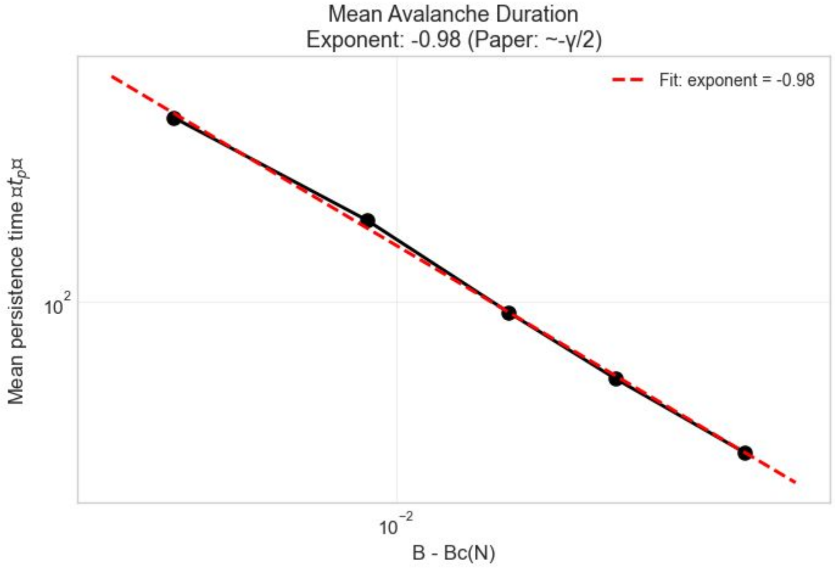
Fitted exponent: 0.529

Theory predicts: 0.5

$R^2 = 0.944$

Appendix





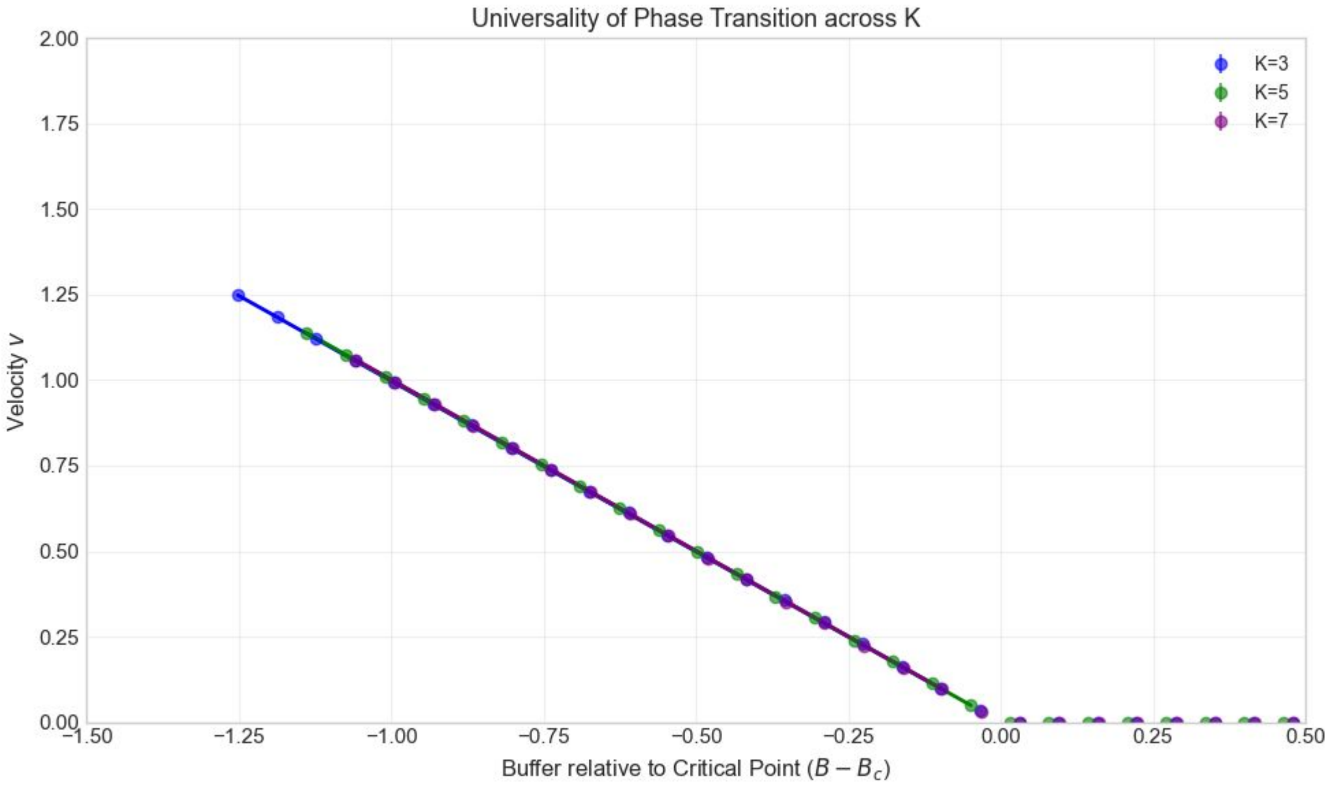
Persistence time power-law test:

B=3.638: exponent = 1.41 ± 0.03 (theory: 1.5), R^2 = 0.991

B=3.642: exponent = 1.40 ± 0.03 (theory: 1.5), R^2 = 0.992

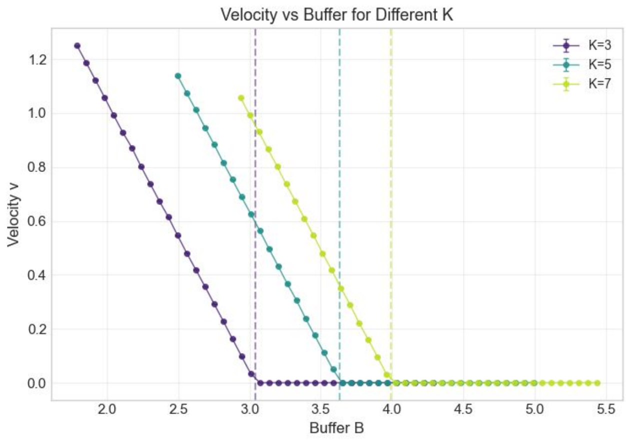
B=3.648: exponent = 1.40 ± 0.03 (theory: 1.5), R^2 = 0.991

Appendix

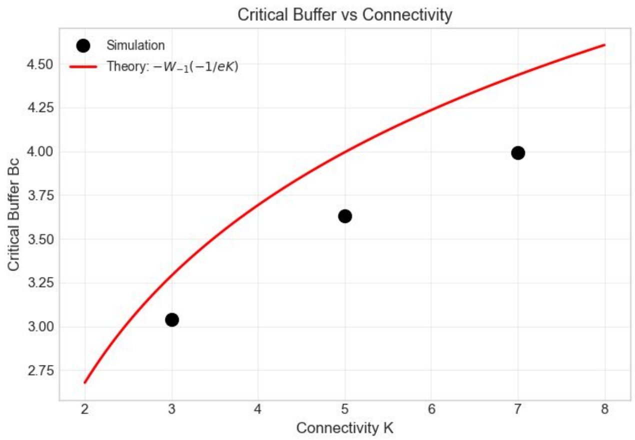


Universality of Phase Transition across K
K=3
K=5
K=7
Velocity v
Buffer relative to Critical Point ($B - B_c$)
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−1.50
−1.25
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50

Appendix



Velocity vs Buffer for Different K
K=3
K=5
K=7
Velocity v
Buffer B
1.2
1.0
0.8
0.6
0.4
0.2
0.0
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5



Critical Buffer vs Connectivity
Simulation
Theory: $-W_{-1}(-1/eK)$
Critical Buffer Bc
Connectivity K
4.50
4.25
4.00
3.75
3.50
3.25
3.00
2.75
2
3
4
5
6
7
8

Appendix

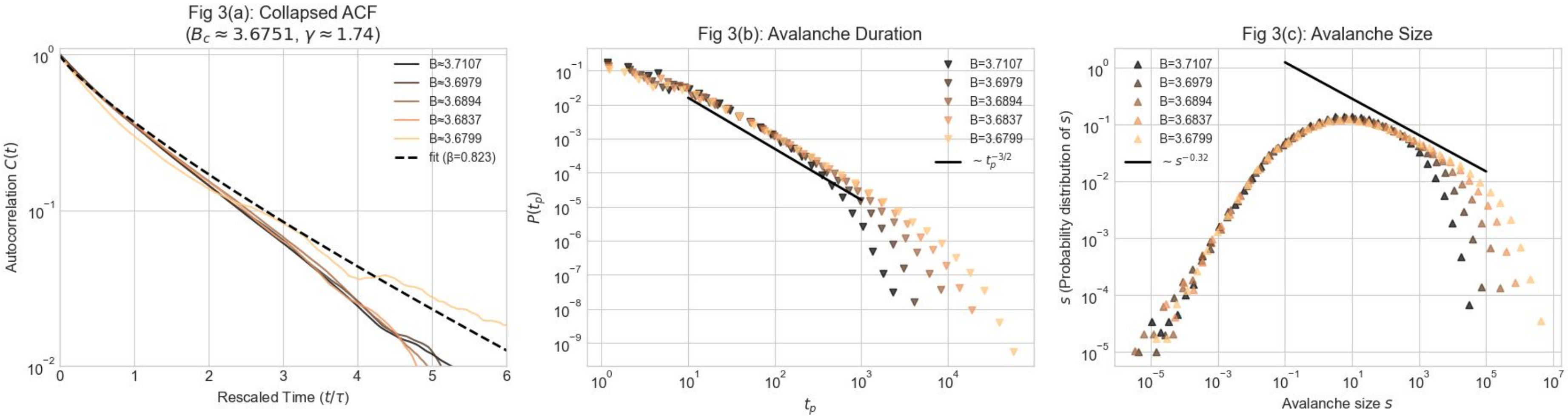
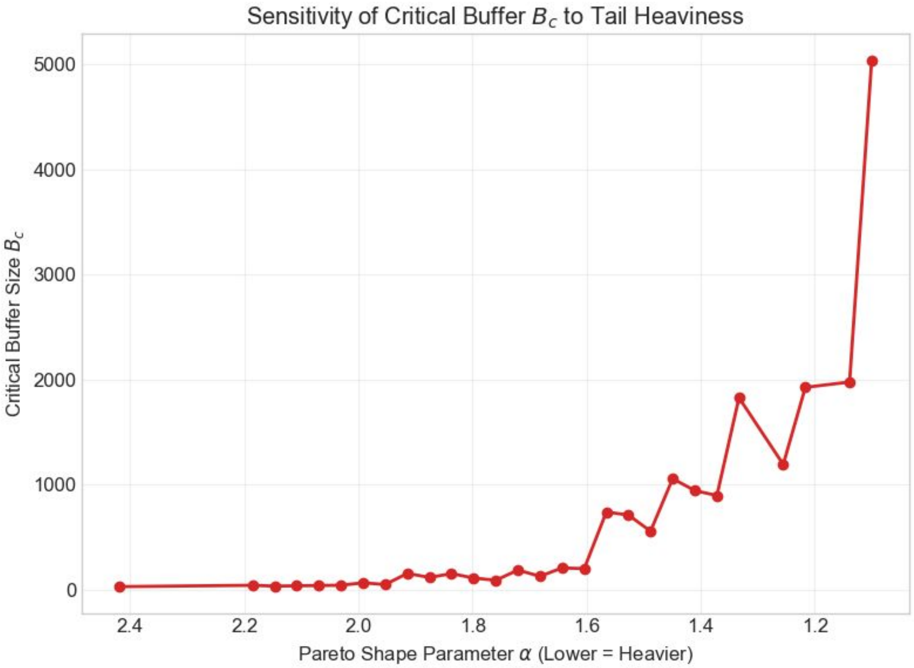
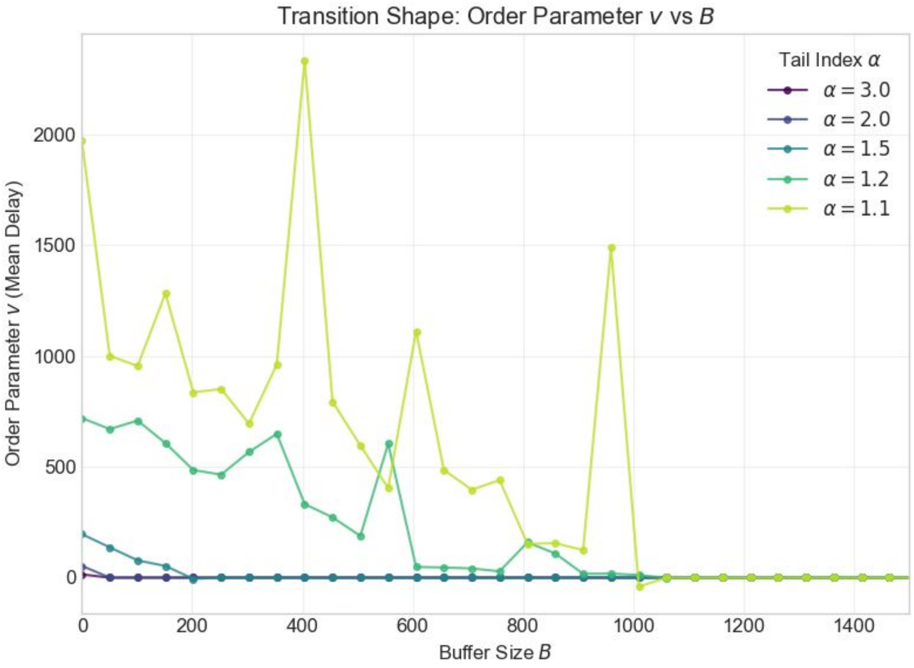


Fig 3(a): Collapsed ACF
($B_c \approx 3.6751$, $\gamma \approx 1.74$)
Autocorrelation $C(t)$
Rescaled Time (t/τ)
B≈3.7107
B≈3.6979
B≈3.6894
B≈3.6837
B≈3.6799
fit (β=0.823)
Fig 3(b): Avalanche Duration
$P(t_p)$
t_p
B=3.7107
B=3.6979
B=3.6894
B=3.6837
B=3.6799
$\sim t_p^{-3/2}$
Fig 3(c): Avalanche Size
s (Probability distribution of s)
Avalanche size s
B=3.7107
B=3.6979
B=3.6894
B=3.6837
B=3.6799
$\sim s^{-0.32}$

Appendix



Sensitivity of Critical Buffer B_c to Tail Heaviness
Critical Buffer Size B_c
5000
4000
3000
2000
1000
0
2.4
2.2
2.0
1.8
1.6
1.4
1.2
Pareto Shape Parameter α (Lower = Heavier)



Transition Shape: Order Parameter v vs B
Order Parameter v (Mean Delay)
Tail Index α
$\alpha = 3.0$
$\alpha = 2.0$
$\alpha = 1.5$
$\alpha = 1.2$
$\alpha = 1.1$
2000
1500
1000
500
0
0
200
400
600
800
1000
1200
1400
Buffer Size B

Appendix

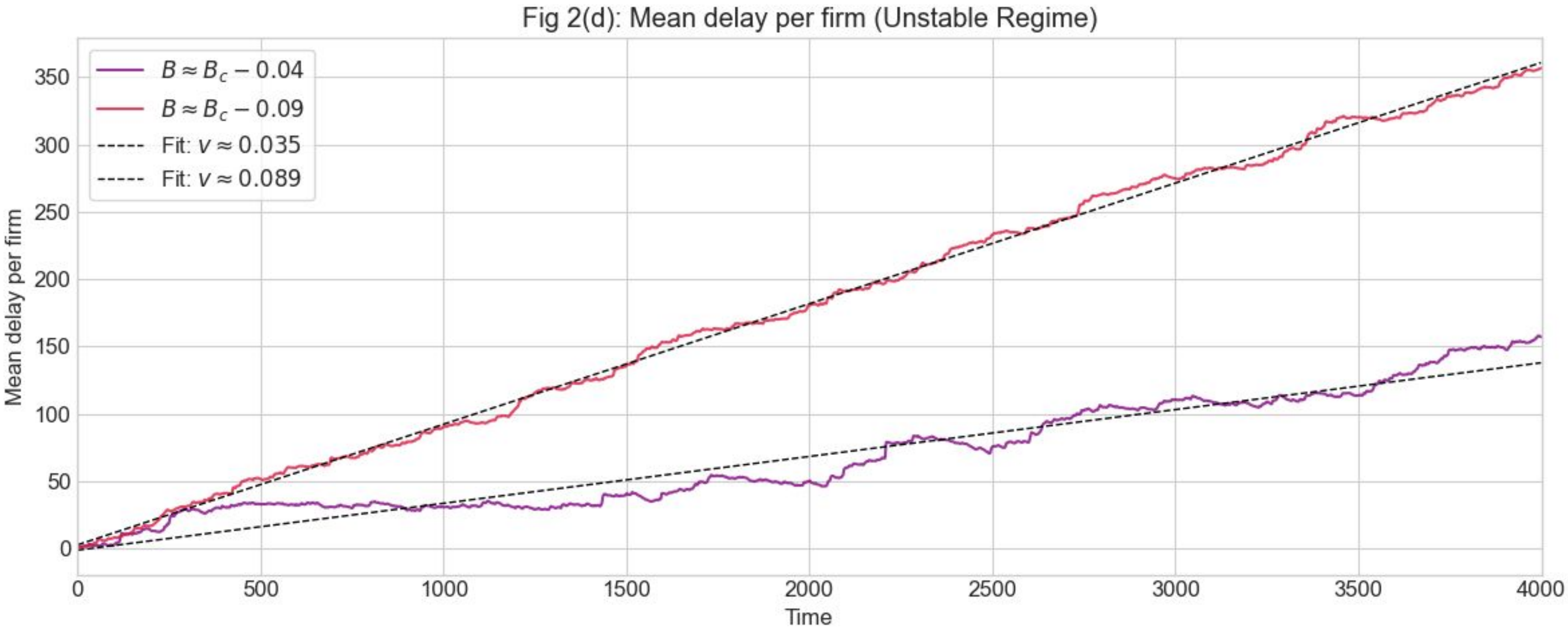


Fig 2(d): Mean delay per firm (Unstable Regime)
$B \approx B_c - 0.04$
$B \approx B_c - 0.09$
Fit: $v \approx 0.035$
Fit: $v \approx 0.089$
Mean delay per firm
Time
0
50
100
150
200
250
300
350
0
500
1000
1500
2000
2500
3000
3500
4000

Appendix: Summary of Hypothesis tests

H1: Velocity slope dv/dB = -1

Weighted average slope: -1.0000 ± 0.0001
Theory predicts: -1.0
Deviation: 0.0000
RESULT: ✓ SUPPORTED (within 2σ of theory)

H2: Critical buffer Bc* from FSS extrapolation

Extrapolated Bc*: 4.00139 ± 0.00380
Theory predicts: 3.99431
Deviation: 1.86σ
RESULT: ✓ CONSISTENT with theory

H3: Alpha exponent behavior

Theory αc* = 0.7496
Simulation shows expected behavior:
- α ≈ constant (αc*) for B < Bc
- α increases for B > Bc

Square-root singularity: See Figure

H4: Finite-Size Scaling

Paper form: $Bc(N) = Bc^* - 1/(a + b^*\ln(N))^2$
Fitted a = 0.4347, b = 0.1425
Paper reports: a ≈ 0.47, b ≈ 0.14 for K=5
RESULT: ✓ Logarithmic convergence confirmed

H5: Avalanche statistics

Persistence time distribution:
- Theory predicts P(tp) ~ tp^(-3/2)
- Simulation shows power-law tail near criticality

Mean persistence time diverges as B → Bc+

Appendix: Data Quality Assessment

Simulation Parameters:
  System sizes tested: [1000, 2500, 5000, 10000, 25000]
  Time steps per run: 50000
  Ensemble size: 10 trials
  Burn-in fraction: 50%

Statistical Reliability:
  N= 1000: Mean SEM(v) = 0.0012
  N= 2500: Mean SEM(v) = 0.0011
  N= 5000: Mean SEM(v) = 0.0010
  N= 10000: Mean SEM(v) = 0.0009
  N= 25000: Mean SEM(v) = 0.0008

R^2 values for linear fits (v vs B):
  N= 1000: R^2 = 1.0000
  N= 2500: R^2 = 1.0000
  N= 5000: R^2 = 1.0000
  N= 10000: R^2 = 1.0000
  N= 25000: R^2 = 1.0000

Appendix: Synthetic Temporal Networks with sparsity (STNs)

How do we emulate it? Array based network emulation

Most important steps:

Generate and shuffle delay array

```
# Shuffle the delays from the last iteration k times.
for i in range(k):
    np.random.shuffle(delays_last_iteration)
    # Reformat them each time into the selected delays array, this ensures exactly k connections per node.
    # While making sure that each node has exactly k connections.
    delays_selected[i*n:(i+1)*n] = delays_last_iteration
```

Select affected nodes

```
# The probability of a node being selected is 1-sparsity.
n_selected = np.random.choice([0, 1], size=n, p=[sparsity, 1-sparsity])
delays_last_iteration_orginal = delays_last_iteration.copy()
```

Propagate delay to next iteration

```
# Only propagate the delays of the nodes that are selected.
delays_current_iteration = np.where(n_selected == 1, np.where(max_values < 0, 0, max_values) + eps, delays_last_iteration_orginal)
```