

Online Control of Cascading Power Failures

Daniel Bienstock, Columbia University

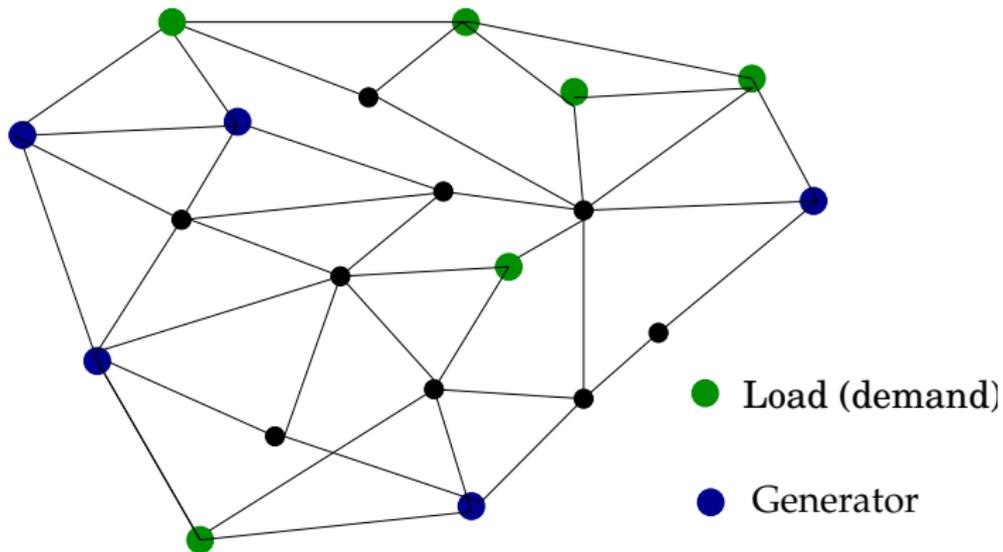
4-20-11

Credits

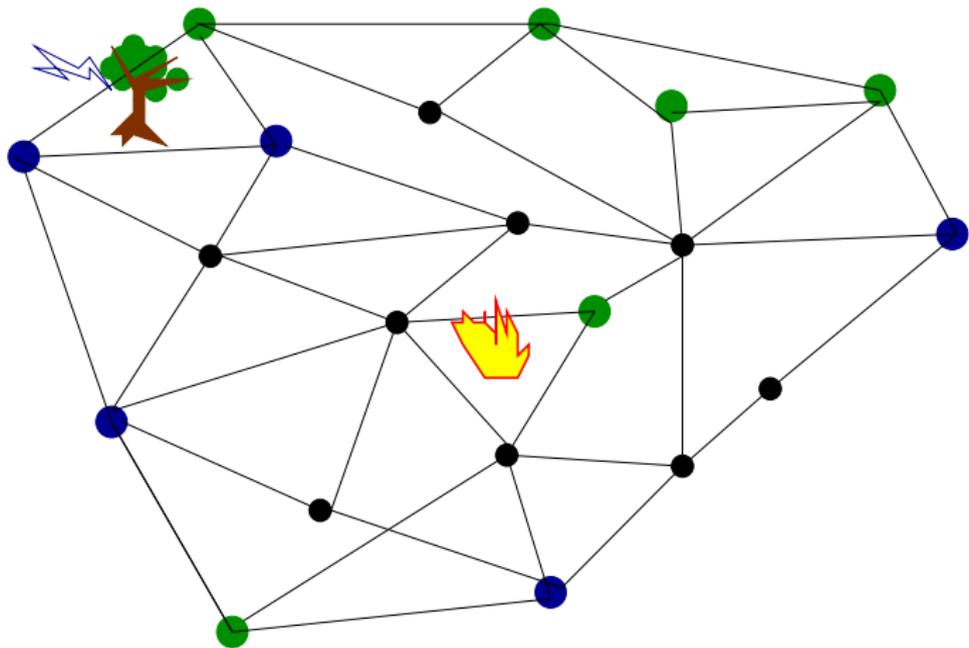
DOE grant with I. Hiskens (Michigan), I. Dobson J. Linderoth, S. Wright (Wisconsin)

PhD students: Sean Harnett (Applied Math), A. Verma (O.R.)

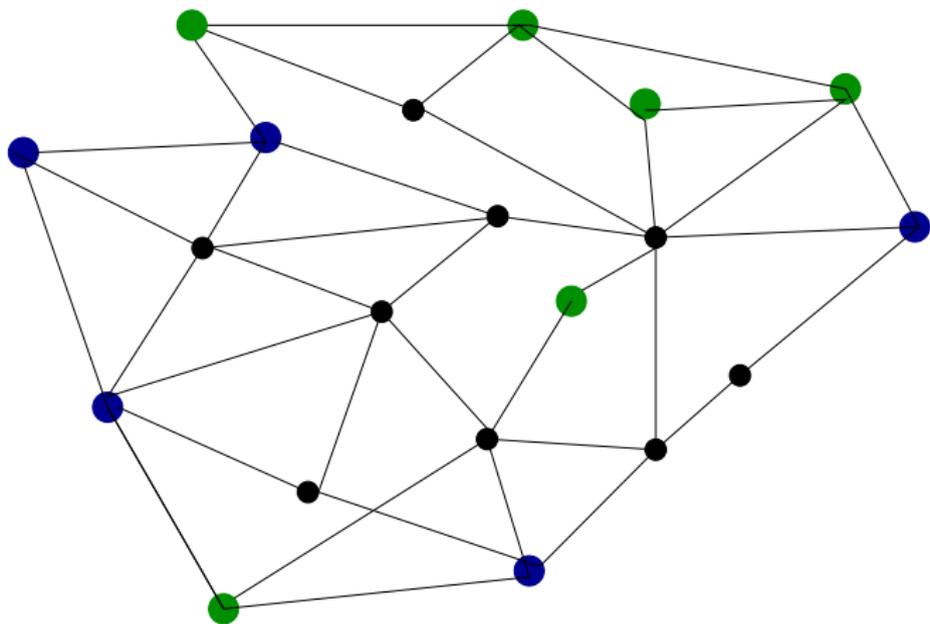
Cascade cartoon



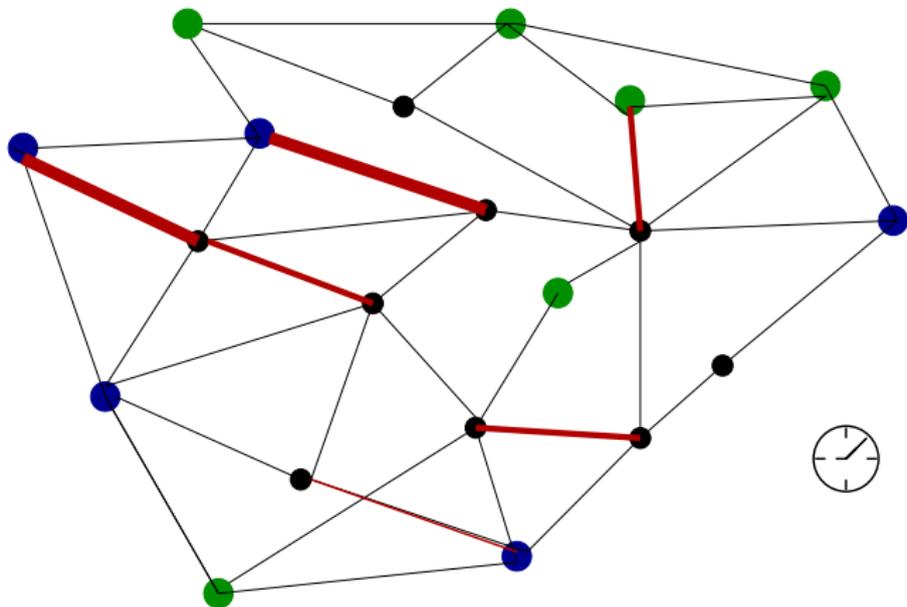
Cascade cartoon



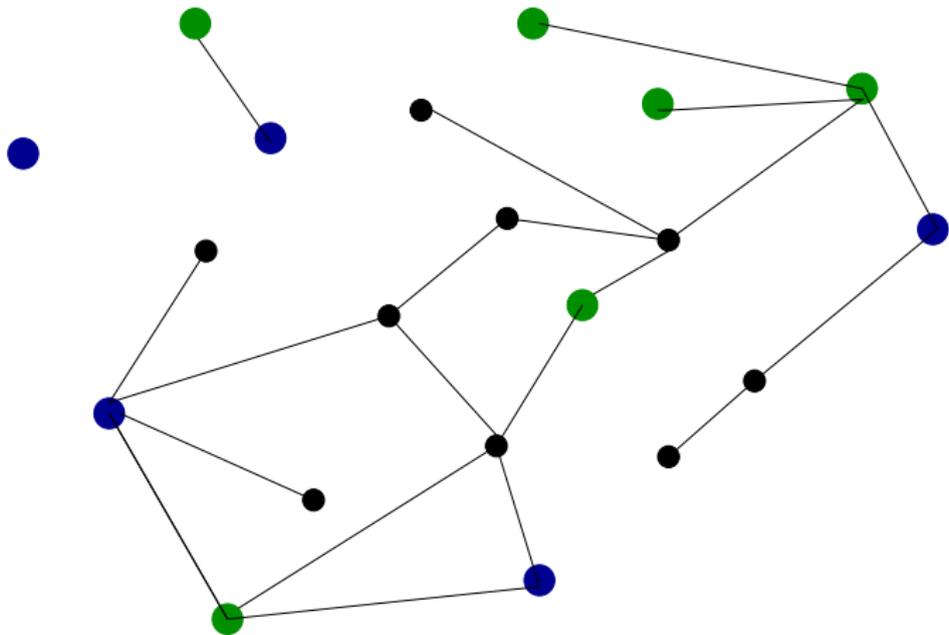
Cascade cartoon



Cascade cartoon



Cascade cartoon



Formal cascade model (Dobson et al)

→ Initial fault event takes place (an “act of God”).

Formal cascade model (Dobson et al)

→ Initial fault event takes place (an “act of God”).

For $r = 1, 2, \dots,$

1. Compute new power flows
2. Determine new set of outaged lines

More detailed cascade model

→ Initial fault event takes place (an “act of God”).

More detailed cascade model

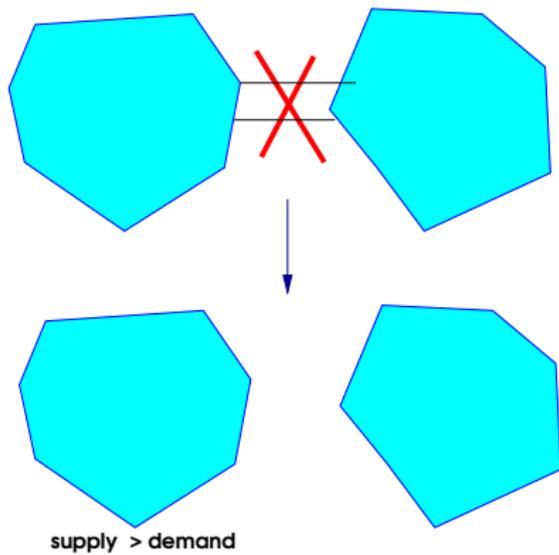
→ Initial fault event takes place (an “act of God”).

For $r = 1, 2, \dots,$

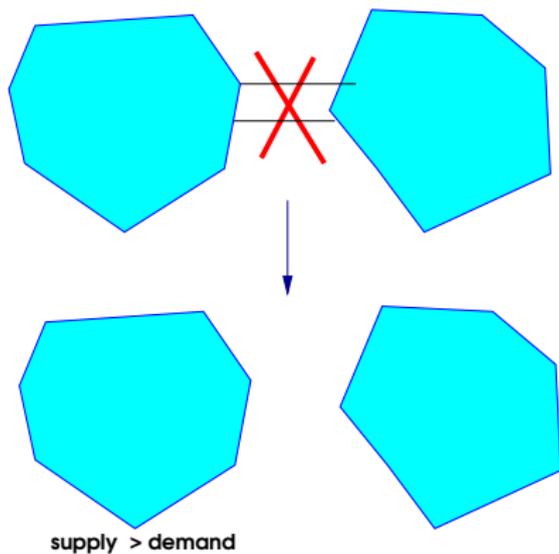
(round r of cascade)

1. Reconfigure demands and generator output levels.

Islanding

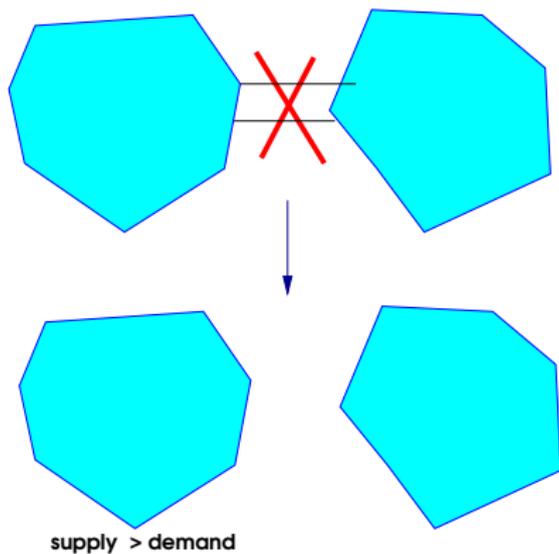


Islanding



→ proportional adjustment of generator outputs/loads

Islanding



- proportional adjustment of generator outputs/loads
- to be added: generator ramp-up (down?) rates, other limits

More detailed cascade model

→ Initial fault event takes place (an “act of God”).

For $r = 1, 2, \dots,$

(round r of cascade)

1. Reconfigure demands and generator output levels.

More detailed cascade model

→ Initial fault event takes place (an “act of God”).

For $r = 1, 2, \dots,$

(round r of cascade)

1. Reconfigure demands and generator output levels.
2. New power flows are instantiated.

Which power flow model?

→ This talk: linearized flows

Which power flow model?

→ This talk: linearized flows

→ Working on: AC power flows

Which power flow model?

→ This talk: linearized flows

→ Working on: AC power flows

Low (2011): some (all?) AC power flow problems can be (at least approximately) modeled using SDP

Which power flow model?

→ This talk: linearized flows

→ Working on: AC power flows

Low (2011): some (all?) AC power flow problems can be (at least approximately) modeled using SDP

▶ Efficient in the large scale setting?

Which power flow model?

→ This talk: linearized flows

→ Working on: AC power flows

Low (2011): some (all?) AC power flow problems can be (at least approximately) modeled using SDP

- ▶ Efficient in the large scale setting?
- ▶ However, “local” version seems workable – can either find a nearby solution, or **prove** that none exists (cannot be done with Newton-Raphson)

More detailed cascade model

→ Initial fault event takes place (an “act of God”).

For $r = 1, 2, \dots,$

(round r of cascade)

1 Reconfigure demands and generator output levels.

More detailed cascade model

→ Initial fault event takes place (an “act of God”).

For $r = 1, 2, \dots,$

(round r of cascade)

- 1 Reconfigure demands and generator output levels.
- 2 New power flows are instantiated.

More detailed cascade model

→ Initial fault event takes place (an “act of God”).

For $r = 1, 2, \dots,$

(round r of cascade)

- 1 Reconfigure demands and generator output levels.
- 2 New power flows are instantiated.
- 3 The next set of line outages takes place. If none do, STOP

Outage mechanism

Notation: f_k^r = flow on line k in round r

Outage mechanism

Notation: \mathbf{f}_k^r = flow on line \mathbf{k} in round \mathbf{r}

Set $\tilde{\mathbf{f}}_k^r = \alpha |\mathbf{f}_k^r| + (1 - \alpha) \tilde{\mathbf{f}}_k^{r-1}$, where $0 \leq \alpha \leq 1$.

Outage mechanism

Notation: \mathbf{f}_k^r = flow on line \mathbf{k} in round \mathbf{r}

Set $\tilde{\mathbf{f}}_k^r = \alpha |\mathbf{f}_k^r| + (1 - \alpha) \tilde{\mathbf{f}}_k^{r-1}$, where $0 \leq \alpha \leq 1$.

($\rightarrow \tilde{\mathbf{f}}_k^r$ = running average of $|\mathbf{f}_k|$)

Outage mechanism

Notation: \mathbf{f}_k^r = flow on line \mathbf{k} in round \mathbf{r}

Set $\tilde{\mathbf{f}}_k^r = \alpha |\mathbf{f}_k^r| + (1 - \alpha) \tilde{\mathbf{f}}_k^{r-1}$, where $0 \leq \alpha \leq 1$.

($\rightarrow \tilde{\mathbf{f}}_k^r$ = running average of $|\mathbf{f}_k|$)

$\rightarrow \mathbf{k}$ outages if $\tilde{\mathbf{f}}_k^r > \mathbf{u}_k$.

Outage mechanism

Notation: \mathbf{f}_k^r = flow on line \mathbf{k} in round \mathbf{r}

Set $\tilde{\mathbf{f}}_k^r = \alpha |\mathbf{f}_k^r| + (1 - \alpha) \tilde{\mathbf{f}}_k^{r-1}$, where $0 \leq \alpha \leq 1$.

($\rightarrow \tilde{\mathbf{f}}_k^r$ = running average of $|\mathbf{f}_k|$)

$\rightarrow \mathbf{k}$ outages if $\tilde{\mathbf{f}}_k^r > \mathbf{u}_k$. **or:** \mathbf{e} outages if $\tilde{\mathbf{f}}_k^r \geq \mathbf{u}_k$

Controlled cascades

→ Initial outage event takes place.

Controlled cascades

→ Initial outage event takes place. **Compute control.**

Controlled cascades

→ Initial outage event takes place. **Compute control.**

For $r = 1, 2, \dots, R - 1$

1. Reconfigure demands and generator output levels.

Controlled cascades

→ Initial outage event takes place. **Compute control.**

For $r = 1, 2, \dots, R - 1$

1. Reconfigure demands and generator output levels.
2. New power flows are instantiated.

Controlled cascades

→ Initial outage event takes place. **Compute control.**

For $r = 1, 2, \dots, R - 1$

1. Reconfigure demands and generator output levels.
2. New power flows are instantiated.
- 3a. **Take measurements and apply control to shed demand.**

Controlled cascades

→ Initial outage event takes place. **Compute control.**

For $r = 1, 2, \dots, R - 1$

1. Reconfigure demands and generator output levels.
2. New power flows are instantiated.
- 3a. **Take measurements and apply control to shed demand.**
- 3b. **Reconfigure generator outputs;**

Controlled cascades

→ Initial outage event takes place. **Compute control.**

For $r = 1, 2, \dots, R - 1$

1. Reconfigure demands and generator output levels.
2. New power flows are instantiated.
- 3a. **Take measurements and apply control to shed demand.**
- 3b. **Reconfigure generator outputs; get new power flows.**

Controlled cascades

→ Initial outage event takes place. **Compute control.**

For $r = 1, 2, \dots, R - 1$

1. Reconfigure demands and generator output levels.
2. New power flows are instantiated.
- 3a. **Take measurements and apply control to shed demand.**
- 3b. **Reconfigure generator outputs; get new power flows.**
4. The next set of outages takes place.

Controlled cascades

→ Initial outage event takes place. **Compute control.**

For $r = 1, 2, \dots, R - 1$

1. Reconfigure demands and generator output levels.
2. New power flows are instantiated.
- 3a. **Take measurements and apply control to shed demand.**
- 3b. **Reconfigure generator outputs; get new power flows.**
4. The next set of outages takes place.

At round R , reduce demands so as to remove any line overloads.

Adaptive affine controls

For each demand bus $n \mathbf{v}$, and round \mathbf{r} , control triple $\mathbf{c}_v^r, \mathbf{b}_v^r, \mathbf{s}_v^r$

Adaptive affine controls

For each demand bus $n \mathbf{v}$, and round \mathbf{r} , control triple c_v^r, b_v^r, s_v^r

→ Parameterized by integers $\mathbf{r} > 0$ and $\delta > 0$.

Adaptive affine controls

For each demand bus $n \mathbf{v}$, and round \mathbf{r} , control triple $\mathbf{c}_v^r, \mathbf{b}_v^r, \mathbf{s}_v^r$

→ Parameterized by integers $\mathbf{r} > \mathbf{0}$ and $\delta > \mathbf{0}$.

At round \mathbf{r} ,

- ▶ Let $\kappa^\delta = \max$ **overload** of any line within radius δ of \mathbf{v}

Adaptive affine controls

For each demand bus \mathbf{v} , and round \mathbf{r} , control triple $\mathbf{c}_v^r, \mathbf{b}_v^r, \mathbf{s}_v^r$

→ Parameterized by integers $\mathbf{r} > \mathbf{0}$ and $\delta > \mathbf{0}$.

At round \mathbf{r} ,

- ▶ Let $\kappa^\delta = \max$ **overload** of any line within radius δ of \mathbf{v}
- ▶ If $\kappa^\delta > \mathbf{c}_v^r$, demand at \mathbf{v} reduced (scaled) by a factor

$$\min \left\{ 1, \{ \mathbf{b}_v^r + \mathbf{s}_v^r (\mathbf{c}_v^r - \kappa^\delta) \}^+ \right\}.$$

Adaptive affine controls

For each demand bus $n \mathbf{v}$, and round \mathbf{r} , control triple $\mathbf{c}_v^r, \mathbf{b}_v^r, \mathbf{s}_v^r$

→ Parameterized by integers $\mathbf{r} > \mathbf{0}$ and $\delta > \mathbf{0}$.

At round \mathbf{r} ,

- ▶ Let $\kappa^\delta = \max$ **overload** of any line within radius δ of \mathbf{v}
- ▶ If $\kappa^\delta > \mathbf{c}_v^r$, demand at \mathbf{v} reduced (scaled) by a factor

$$\min \left\{ \mathbf{1}, \left\{ \mathbf{b}_v^r + \mathbf{s}_v^r (\mathbf{c}_v^r - \kappa^\delta) \right\}^+ \right\}.$$

Example: $(\mathbf{1}, \mathbf{1}, \mathbf{s})$ control; scale = $\min \left\{ \mathbf{1}, \left\{ \mathbf{1} + \mathbf{s} (\mathbf{1} - \kappa) \right\}^+ \right\}$.

Adaptive affine controls

For each demand bus $n \mathbf{v}$, and round \mathbf{r} , control triple $\mathbf{c}_v^r, \mathbf{b}_v^r, \mathbf{s}_v^r$

→ Parameterized by integers $\mathbf{r} > \mathbf{0}$ and $\delta > \mathbf{0}$.

At round \mathbf{r} ,

- ▶ Let $\kappa^\delta = \max$ **overload** of any line within radius δ of \mathbf{v}
- ▶ If $\kappa^\delta > \mathbf{c}_v^r$, demand at \mathbf{v} reduced (scaled) by a factor

$$\min \left\{ \mathbf{1}, \left\{ \mathbf{b}_v^r + \mathbf{s}_v^r (\mathbf{c}_v^r - \kappa^\delta) \right\}^+ \right\}.$$

Example: $(\mathbf{1}, \mathbf{1}, \mathbf{s})$ control; scale = $\min \left\{ \mathbf{1}, \left\{ \mathbf{1} + \mathbf{s} (\mathbf{1} - \kappa) \right\}^+ \right\}$.

Problem: choose control so as to maximize demand at end of round \mathbf{R} .

General methodology: simulation-based optimization

Given a control vector $\tilde{\mathbf{u}} = (\mathbf{c}_v^r, \mathbf{b}_v^r, \mathbf{s}_v^r)$ (over all \mathbf{v} and \mathbf{r}),

$\Upsilon(\tilde{\mathbf{u}})$ = total demand satisfied at cascade end
(yield)

- ▶ Maximization of $\Upsilon(\tilde{\mathbf{u}})$ should be (very?) fast

General methodology: simulation-based optimization

Given a control vector $\tilde{\mathbf{u}} = (\mathbf{c}_v^r, \mathbf{b}_v^r, \mathbf{s}_v^r)$ (over all \mathbf{v} and \mathbf{r}),

$\Upsilon(\tilde{\mathbf{u}})$ = total demand satisfied at cascade end
(yield)

- ▶ Maximization of $\Upsilon(\tilde{\mathbf{u}})$ should be (very?) fast
- ▶ Optimization should be robust (noisy process)

General methodology: simulation-based optimization

Given a control vector $\tilde{\mathbf{u}} = (\mathbf{c}_v^r, \mathbf{b}_v^r, \mathbf{s}_v^r)$ (over all \mathbf{v} and \mathbf{r}),

$\Upsilon(\tilde{\mathbf{u}})$ = total demand satisfied at cascade end
(yield)

- ▶ Maximization of $\Upsilon(\tilde{\mathbf{u}})$ should be (very?) fast
- ▶ Optimization should be robust (noisy process)
- ▶ From a strict perspective, $\Upsilon(\tilde{\mathbf{u}})$ is not even continuous

General methodology: simulation-based optimization

Given a control vector $\tilde{\mathbf{u}} = (\mathbf{c}_v^r, \mathbf{b}_v^r, \mathbf{s}_v^r)$ (over all \mathbf{v} and \mathbf{r}),

$\Upsilon(\tilde{\mathbf{u}})$ = total demand satisfied at cascade end
(yield)

- ▶ Maximization of $\Upsilon(\tilde{\mathbf{u}})$ should be (very?) fast
- ▶ Optimization should be robust (noisy process)
- ▶ From a strict perspective, $\Upsilon(\tilde{\mathbf{u}})$ is not even continuous

$\Upsilon(\tilde{\mathbf{u}})$ is obtained through a simulation

Derivative-free optimization

Conn, Scheinberg, Vicente, others

Rough description:

- ▶ *Sample* a number of control vectors $\tilde{\mathbf{u}}$
- ▶ Use sample points to construct a convex approximation to \mathcal{R}
- ▶ Optimize this approximation; this yields a new sample point

Scalability to large dimensionality?

Derivative-free optimization

Conn, Scheinberg, Vicente, others

Rough description:

- ▶ *Sample* a number of control vectors $\tilde{\mathbf{u}}$
- ▶ Use sample points to construct a convex approximation to \mathcal{T}
- ▶ Optimize this approximation; this yields a new sample point

Scalability to large dimensionality? (Not?)

“First order” method

Given a control vector $\tilde{\mathbf{u}}$

1. Estimate the “gradient” $\mathbf{g} = \nabla \Upsilon(\tilde{\mathbf{u}})$ through finite differences.

“First order” method

Given a control vector $\tilde{\mathbf{u}}$

1. Estimate the “gradient” $\mathbf{g} = \nabla \Upsilon(\tilde{\mathbf{u}})$ through finite differences.

Requires $\mathbf{O}(1)$ simulations per demand node.

“First order” method

Given a control vector $\tilde{\mathbf{u}}$

1. Estimate the “gradient” $\mathbf{g} = \nabla \Upsilon(\tilde{\mathbf{u}})$ through finite differences.

Requires $\mathbf{O}(1)$ simulations per demand node.

2. Estimate step size $\operatorname{argmax} \Upsilon(\tilde{\mathbf{u}} + \sigma \mathbf{g})$

“First order” method

Given a control vector $\tilde{\mathbf{u}}$

1. Estimate the “gradient” $\mathbf{g} = \nabla \Upsilon(\tilde{\mathbf{u}})$ through finite differences.

Requires $\mathbf{O}(1)$ simulations per demand node.

2. Estimate step size $\operatorname{argmax} \Upsilon(\tilde{\mathbf{u}} + \sigma \mathbf{g})$

→ Easily parallelizable

Solving the optimal scaling problem, fast

For each r , and for each *component* (“island”) K at round r ,

$$(c_s^r, b_s^r, s_s^r) = (c_t^r, b_t^r, s_t^r)$$

for every s, t in K

Solving the optimal scaling problem, fast

For each r , and for each *component* (“island”) \mathbf{K} at round r ,

$$(c_s^r, b_s^r, s_s^r) = (c_t^r, b_t^r, s_t^r)$$

for every s, t in \mathbf{K}

Then, equivalent problem:

- ▶ In round r , choose $\alpha^r(\mathbf{K}) \leq 1$ for each component \mathbf{K}
- ▶ If bus $v \in$ component \mathbf{K} , then scale its demand by $\alpha^r(\mathbf{K})$

Solving the optimal scaling problem

Notation:

- ▶ $\hat{\beta}$ = supply/demand vector at time 0
- ▶ $\Upsilon^R(\beta)$ = total demand using optimal control, at end of round R , if the supply/demand vector is β at time 0

Solving the optimal scaling problem

Notation:

- ▶ $\hat{\beta}$ = supply/demand vector at time 0
- ▶ $\Upsilon^R(\beta)$ = total demand using optimal control, at end of round R , if the supply/demand vector is β at time 0
- ▶ For each $t \geq 0$, compute $\Theta^R(t) \doteq \Upsilon^R(t\beta)$

Solving the optimal scaling problem

Notation:

- ▶ $\hat{\beta}$ = supply/demand vector at time 0
- ▶ $\Upsilon^R(\beta)$ = total demand using optimal control, at end of round R , if the supply/demand vector is β at time 0
- ▶ For each $t \geq 0$, compute $\Theta^R(t) \doteq \Upsilon^R(t\beta)$

Theorem:

$\Theta^R(t)$ is piecewise linear nondecreasing with $O(m^{R-1}/R!)$ breakpoints.

Solving the optimal scaling problem

Notation:

- ▶ $\hat{\beta}$ = supply/demand vector at time 0
- ▶ $\Upsilon^R(\beta)$ = total demand using optimal control, at end of round R , if the supply/demand vector is β at time 0
- ▶ For each $t \geq 0$, compute $\Theta^R(t) \doteq \Upsilon^R(t\beta)$

Theorem:

$\Theta^R(t)$ is piecewise linear nondecreasing with $O(m^{R-1}/R!)$ breakpoints.

Actually, $O(f(R) m^2)$ breakpoints.

Solving the optimal scaling problem

Notation:

- ▶ $\hat{\beta}$ = supply/demand vector at time 0
- ▶ $\Upsilon^R(\beta)$ = total demand using optimal control, at end of round R , if the supply/demand vector is β at time 0
- ▶ For each $t \geq 0$, compute $\Theta^R(t) \doteq \Upsilon^R(t\beta)$

Theorem:

$\Theta^R(t)$ is piecewise linear nondecreasing with $O(m^{R-1}/R!)$ breakpoints.

Actually, $O(f(R) m^2)$ breakpoints.

Well, probably $O(R m)$ breakpoints.

Implementation

- (1) Solve scaling problem
- (2) Grid search around control found by scaling problem
- (3) (optional) First-order method on demand-quantile control
- (4) (optional) Then switch to first-order method

Implementation

- (1) Solve scaling problem
 - (2) Grid search around control found by scaling problem
 - (3) (optional) First-order method on demand-quantile control
 - (4) (optional) Then switch to first-order method
- ▶ Parallel implementation using Unix sockets
 - ▶ 24 cores (3 × 8-core Intel i7 systems)

Implementation

- (1) Solve scaling problem
 - (2) Grid search around control found by scaling problem
 - (3) (optional) First-order method on demand-quantile control
 - (4) (optional) Then switch to first-order method
- ▶ Parallel implementation using Unix sockets
 - ▶ 24 cores (3 × 8-core Intel i7 systems) soon (?) more cores, cloud, BlueGene, gpu (?)
 - ▶ Gurobi, Cplex used to solve linear systems

Implementation

- (1) Solve scaling problem
 - (2) Grid search around control found by scaling problem
 - (3) (optional) First-order method on demand-quantile control
 - (4) (optional) Then switch to first-order method
- ▶ Parallel implementation using Unix sockets
 - ▶ 24 cores (3 × 8-core Intel i7 systems) soon (?) more cores, cloud, BlueGene, gpu (?)
 - ▶ Gurobi, Cplex used to solve linear systems
 - ▶ **(1)** and **(2)** on grids with 10^4 s lines/buses require seconds

Implementation

- (1) Solve scaling problem
 - (2) Grid search around control found by scaling problem
 - (3) (optional) First-order method on demand-quantile control
 - (4) (optional) Then switch to first-order method
- ▶ Parallel implementation using Unix sockets
 - ▶ 24 cores (3 × 8-core Intel i7 systems) soon (?) more cores, cloud, BlueGene, gpu (?)
 - ▶ Gurobi, Cplex used to solve linear systems
 - ▶ **(1)** and **(2)** on grids with 10^4 s lines/buses require seconds
 - ▶ Five gradient steps of general method: \sim one hour wallclock

Experiments on the **Eastern Interconnect**:
approximately 15K buses and 23K lines (Powerworld 03sfeq)

Experiments on the **Eastern Interconnect**:
approximately 15K buses and 23K lines (Powerworld 03sfeq)

- ▶ Cascade initiated by disabling many high flow lines (but retaining connectivity)

Experiments on the **Eastern Interconnect**: approximately 15K buses and 23K lines (Powerworld 03sfeq)

- ▶ Cascade initiated by disabling many high flow lines (but retaining connectivity)

Control subject to two constraints:

- ▶ Control can only operate in rounds 1 - 10
- ▶ Length of cascade limited to $R = 20$ rounds

Experiments on the **Eastern Interconnect**: approximately 15K buses and 23K lines (Powerworld 03sfeq)

- ▶ Cascade initiated by disabling many high flow lines (but retaining connectivity)

Control subject to two constraints:

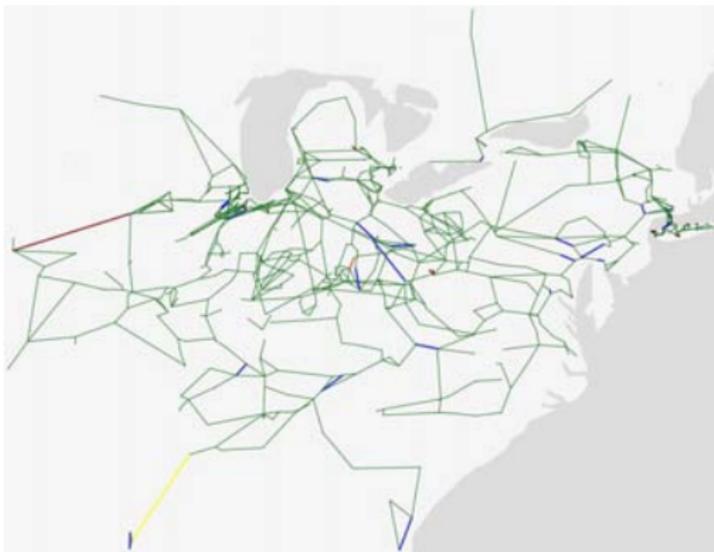
- ▶ Control can only operate in rounds 1 - 10
- ▶ Length of cascade limited to $R = 20$ rounds

All cascades evaluated on three criteria:

- ▶ Yield
- ▶ Length of cascade (= no. of rounds until stable)
- ▶ Number of outaged lines

r	No control		Control 1		Control 2	
	lines out	yield %	lines out	yield %	lines out	yield %
1	24	100	3	81	3	81
2	56	99	5	81	5	81
3	127	99	3	81	3	81
4	132	98	4	81	0	75
5	253	96	6	81		
6	467	94	20	81		
7	433	91	63	81		
8	483	90	33	81		
9	389	87	49	81		
10	419	85	65	81		
15	263	70	0	80		
16	357	67				
17	495	65				
18	335	63				
19	101	61				
20	15	61				
24	1	61				
25	0	61				

Start



green = normal operation
blue = disabled by contingency
black = outage
red, yellow = overload

Round 1

no control



yield = 100%, 24 outaged lines

control



yield 100%, 3 outaged lines

green = normal operation
blue = disabled by contingency
black = outage
red, yellow = overload

Round 4

no control



yield = 98.06%, 339 outaged lines

control



stable, yield = 75%, 11 outaged lines

green = normal operation
blue = disabled by contingency
black = outage
red, yellow = overload

Round 5

no control



yield = 96.39%, 592 outaged lines

control



stable at round 4, yield = 75%, 11 outaged lines

green = normal operation
blue = disabled by contingency
black = outage
red, yellow = overload

Round 6

no control



yield = 93.82%, 1059 outaged lines

control



stable at round 4, yield = 75%, 11 outaged lines

green = normal operation
blue = disabled by contingency
black = outage
red, yellow = overload

Round 7

no control



yield = 91.08%, 1492 outaged lines

control



stable at round 4, yield = 75%, 11 outaged lines

green = normal operation
blue = disabled by contingency
black = outage
red, yellow = overload

Round 8

no control



yield = 89.17%, 1975 outaged lines

control



stable at round 4, yield = 75%, 11 outaged lines

green = normal operation
blue = disabled by contingency
black = outage
red, yellow = overload

Round 9

no control



yield = 87.17%, 2364 outaged lines

control



stable at round 4, yield = 75%, 11 outaged lines

green = normal operation
blue = disabled by contingency
black = outage
red, yellow = overload

Round 10

no control



yield = 85.16%, 2783 outaged lines

control



stable at round 4, yield = 75%, 11 outaged lines

green = normal operation
blue = disabled by contingency
black = outage
red, yellow = overload

Round 11

no control



yield = 80.95%, 3318 outaged lines

control



stable at round 4, yield = 75%, 11 outaged lines

green = normal operation
blue = disabled by contingency
black = outage
red, yellow = overload

Round 12

no control



yield = 78.22%, 3686 outaged lines

control



stable at round 4, yield = 75%, 11 outaged lines

green = normal operation
blue = disabled by contingency
black = outage
red, yellow = overload

Round 13

no control



yield = 74.77%, 4063 outaged lines

control



stable at round 4, yield = 75%, 11 outaged lines

green = normal operation
blue = disabled by contingency
black = outage
red, yellow = overload

Round 14

no control



yield = 72.02%, 4356 outaged lines

control

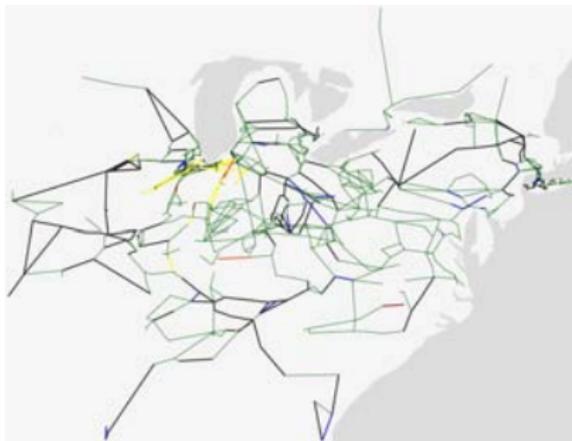


stable at round 4, yield = 75%, 11 outaged lines

green = normal operation
blue = disabled by contingency
black = outage
red, yellow = overload

Round 15

no control



yield = 70.45%, 4619 outaged lines

control

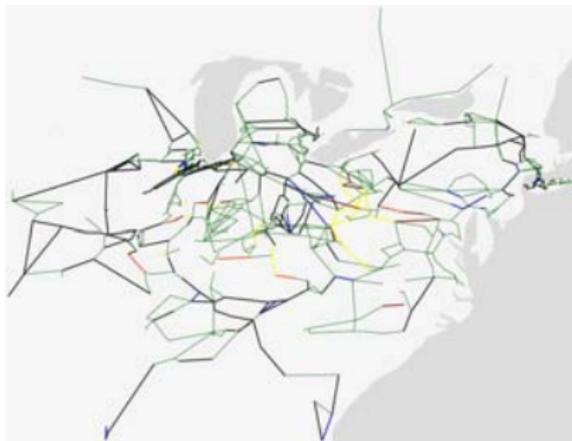


stable at round 4, yield = 75%, 11 outaged lines

green = normal operation
blue = disabled by contingency
black = outage
red, yellow = overload

Round 16

no control



yield = 66.78%, 4976 outaged lines

control



stable at round 4, yield = 75%, 11 outaged lines

green = normal operation
blue = disabled by contingency
black = outage
red, yellow = overload

Round 17

no control



yield = 65.09%, 5471 outaged lines

control



stable at round 4, yield = 75%, 11 outaged lines

green = normal operation
blue = disabled by contingency
black = outage
red, yellow = overload

Round 18

no control



yield = 62.89%, 5806 outaged lines

control



stable at round 4, yield = 75%, 11 outaged lines

green = normal operation
blue = disabled by contingency
black = outage
red, yellow = overload

Round 19

no control



yield = 61.46%, 5907 outaged lines

control



stable at round 4, yield = 75%, 11 outaged lines

green = normal operation
blue = disabled by contingency
black = outage
red, yellow = overload

Round 25

no control



stable, yield 60.78%, 5959 outaged lines

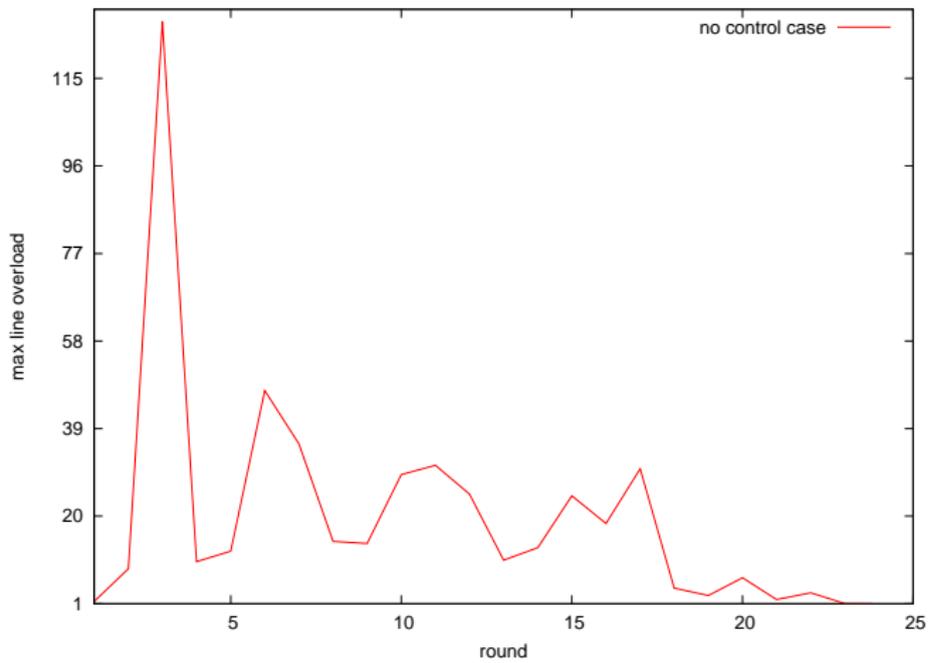
control



stable at round 4, yield = 75%, 11 outaged lines

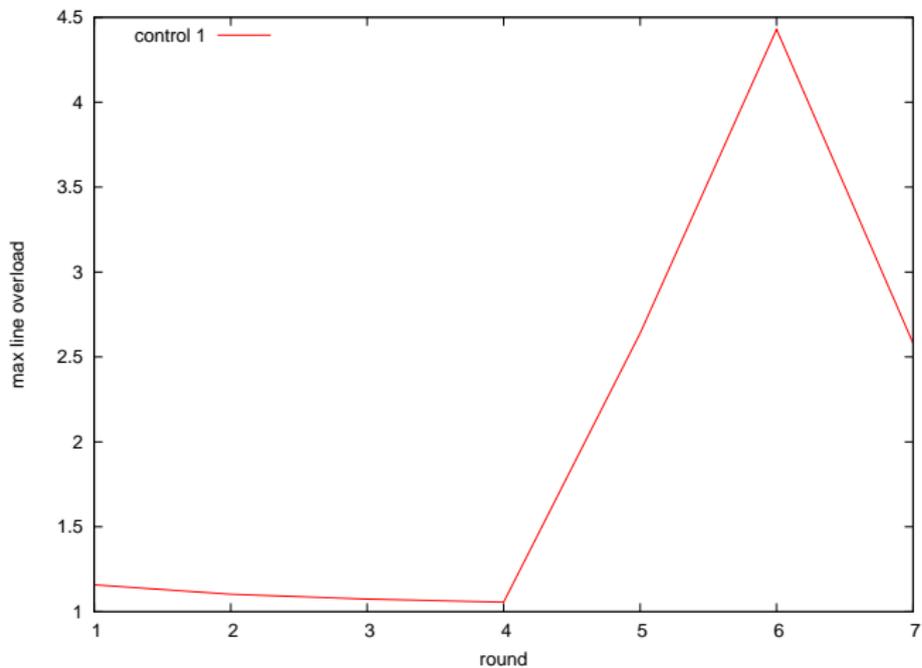
green = normal operation
blue = disabled by contingency
black = outage
red, yellow = overload

Why: overloads in no-control case



r	No control		Control 1		Control 2	
	lines out	yield %	lines out	yield %	lines out	yield %
1	24	100	3	(.81) 81	3	(.81) 81
2	56	99	5	81	5	81
3	127	99	3	81	3	81
4	132	98	4	81	0	(.92) 75
5	253	96	6	81		
6	467	94	20	81		
7	433	91	63	81		
8	483	90	33	81		
9	389	87	49	81		
10	419	85	65	81		
15	263	70	0	80		
16	357	67				
17	495	65				
18	335	63				
19	101	61				
20	15	61				
24	1	61				
25	0	61				

Why: overloads under control 1



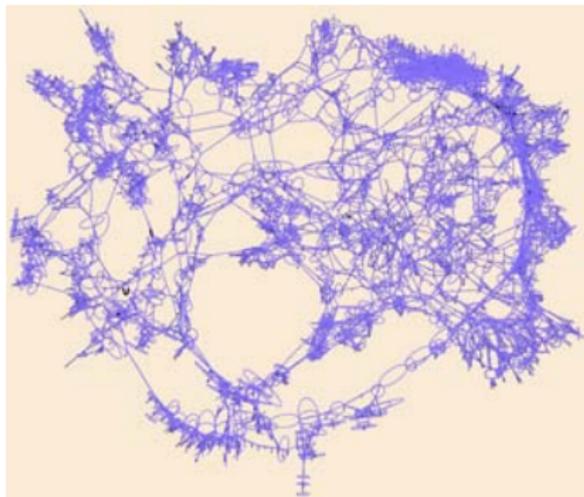
A very different cascade

r	No control				Control			
	κ	O	I	Y	κ	O	I	Y
1	40.96	86	1	100	40.96	86	1	100
2	8.60	187	8	99	8.60	165	8	* 96
3	55.51	365	20	98	61.74	303	17	96
4	67.14	481	70	95	66.63	408	44	94
5	94.61	692	149	93	131.08	492	94	93
6	115.53	403	220	91	112.58	416	146	90
7	66.12	336	333	89	99.62	326	191	* 78
8	47.83	247	414	87	60.95	227	248	77
9	7.16	160	457	85	32.50	72	279	76
10	7.06	245	542	84	9.50	43	292	76
15	5.03	64	721	81	1.34	1	312	75
16	84.67	72	743	80	1.13	1	312	75
17	32.15	52	756	80	1.38	2	312	75
18	6.50	43	763	80	1.26	1	312	75
19	9.97	85	812	80	0.99	0	312	75
34	0.99	0	995	78				

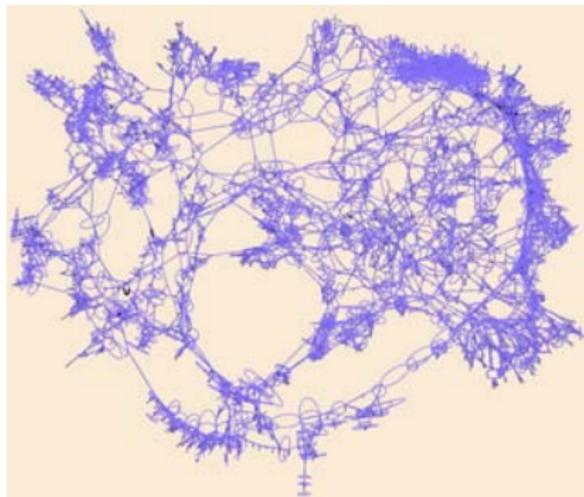
κ = max line overload, O = outages, I = islands, Y = yield (%)

round 1

no control

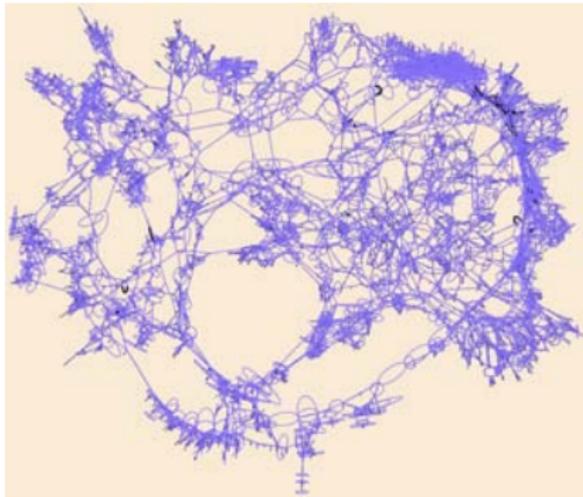


control

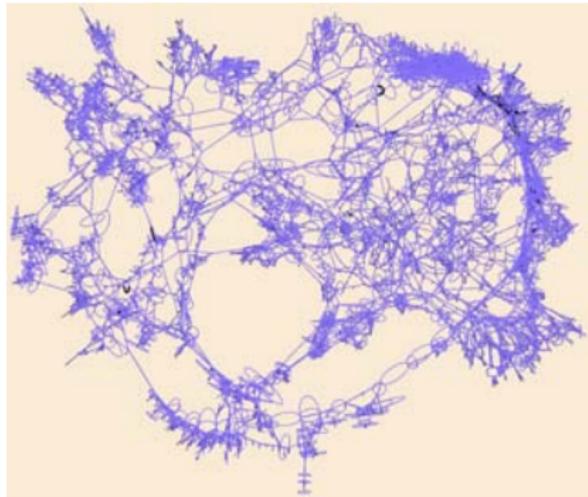


round 2

no control

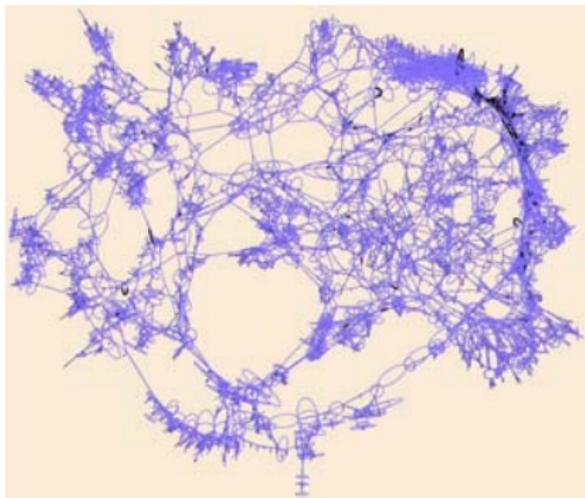


control

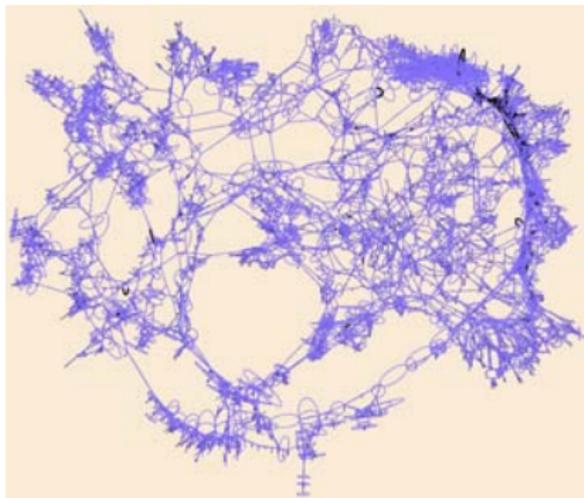


round 3

no control

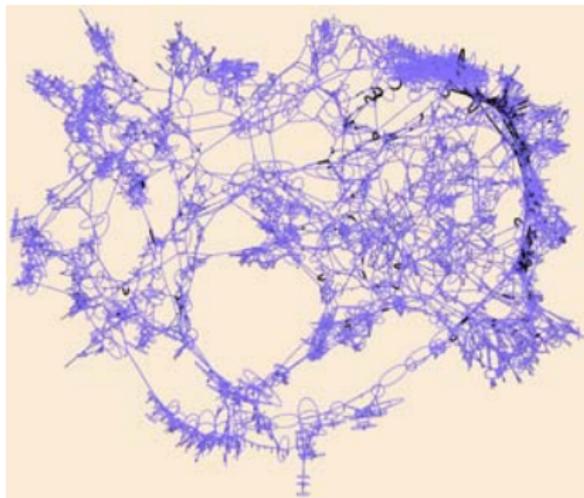


control

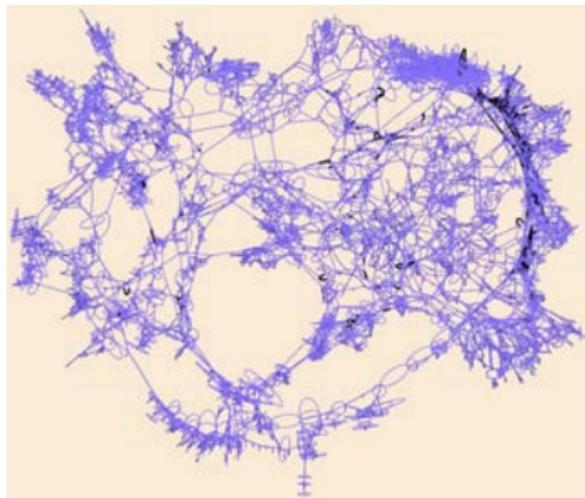


round 4

no control

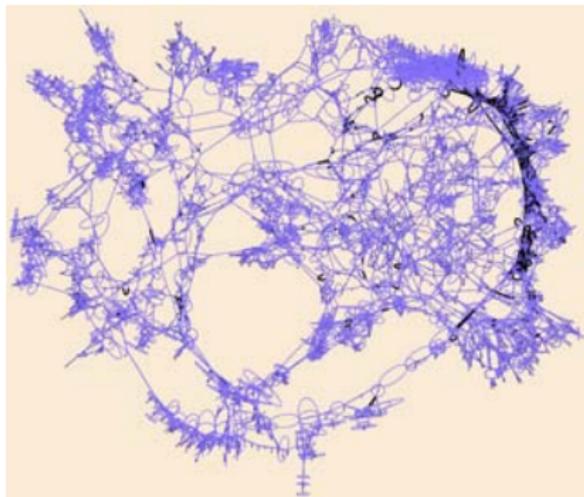


control

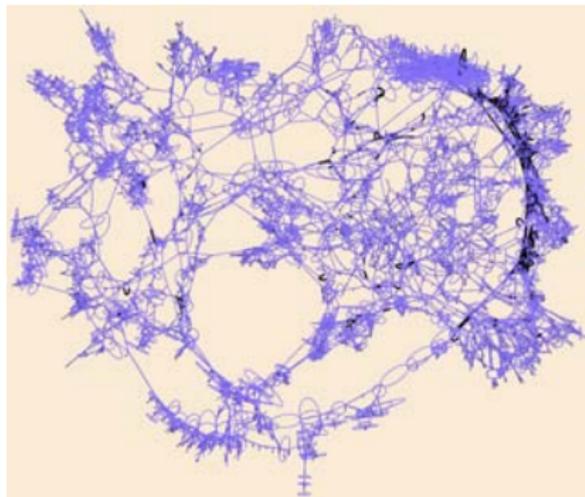


round 5

no control

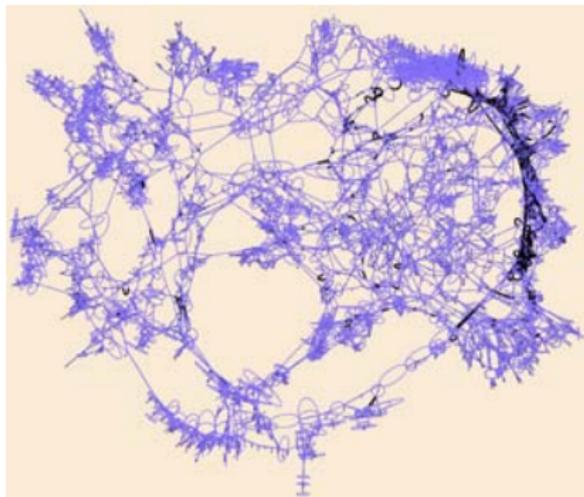


control

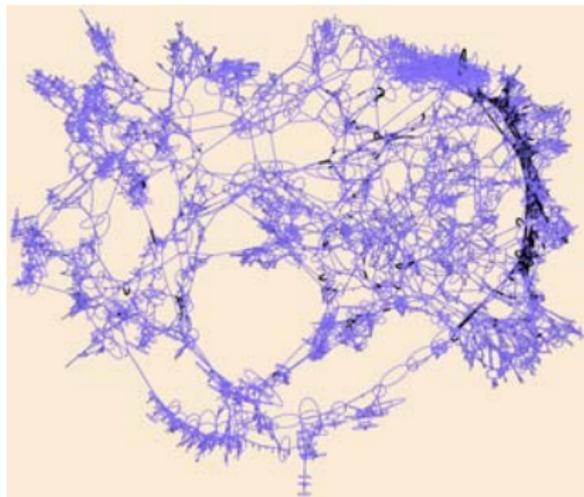


round 6

no control

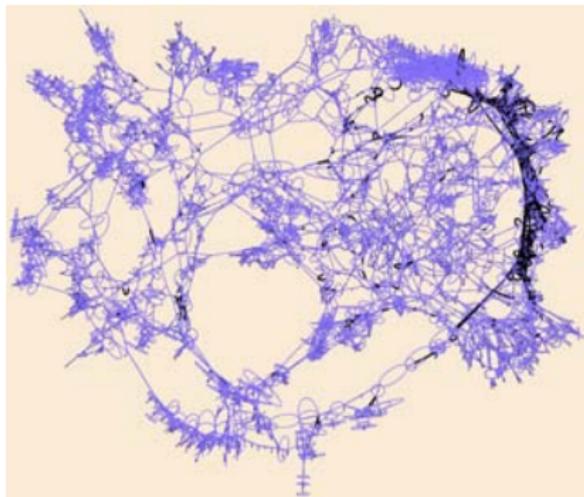


control

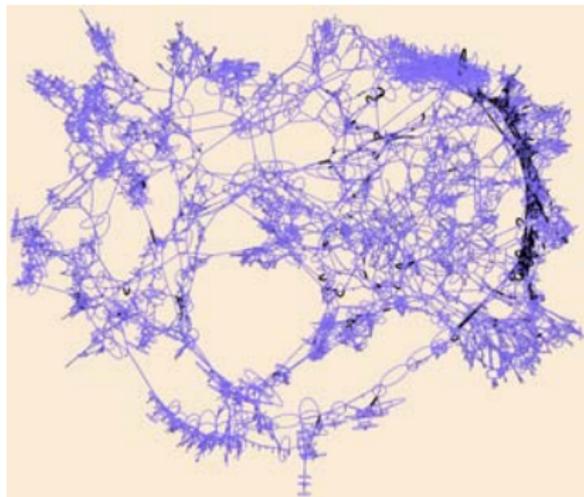


round 7

no control

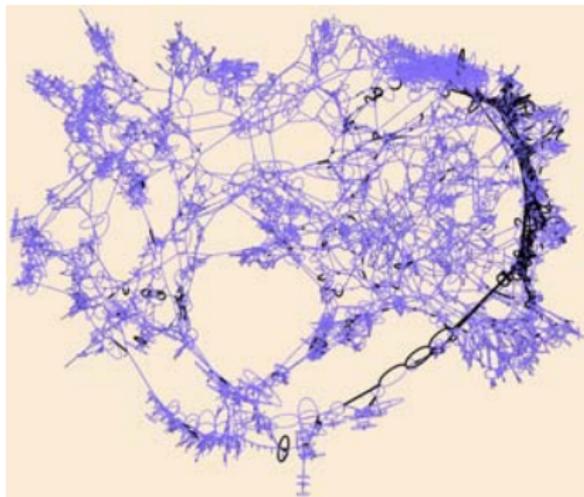


control

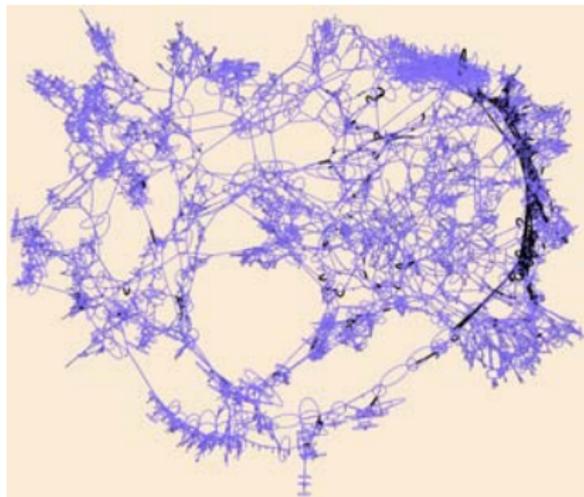


round 8

no control

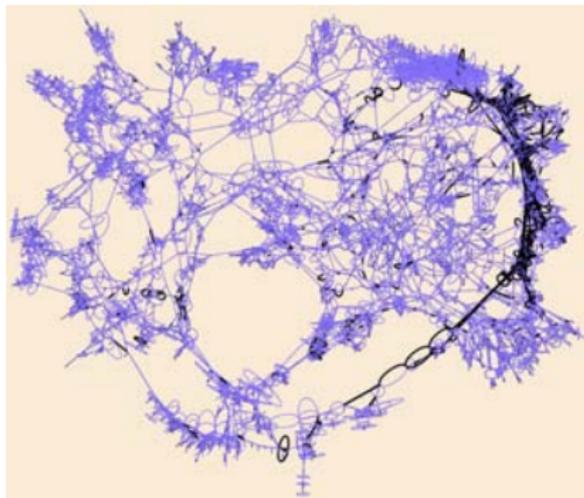


control

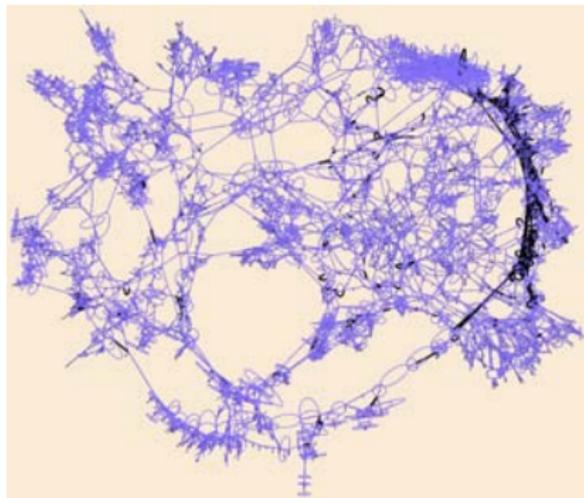


round 9

no control

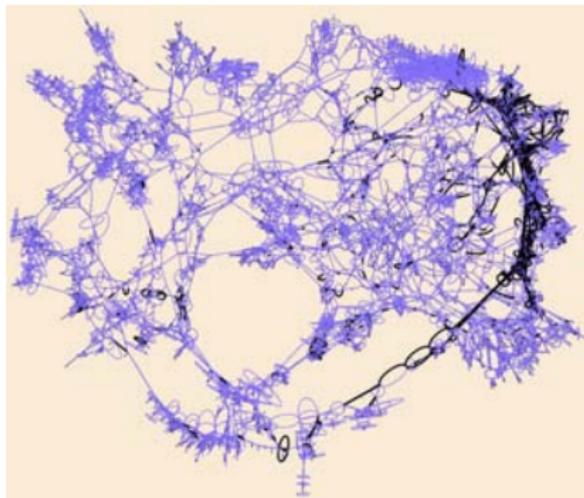


control

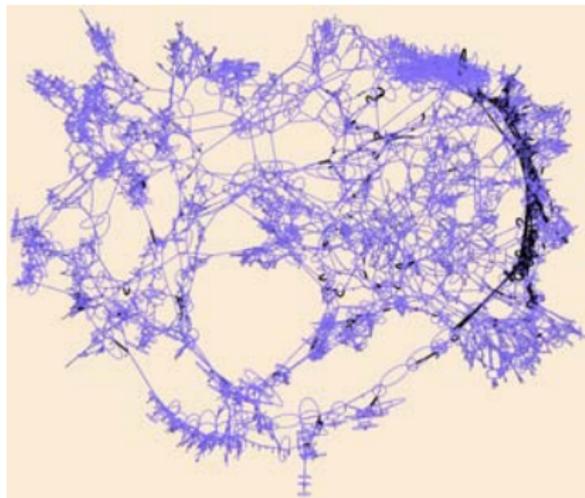


round 10

no control

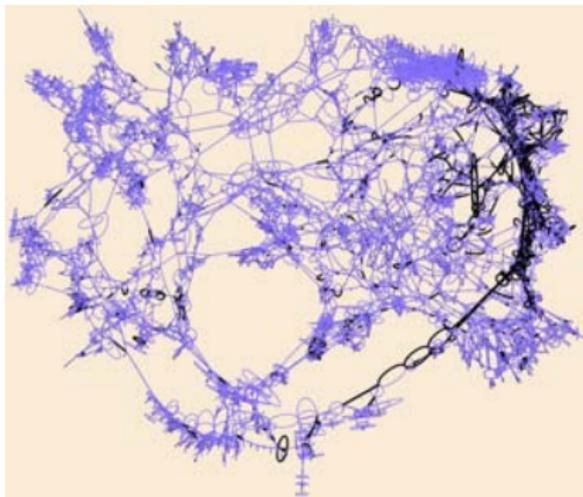


control

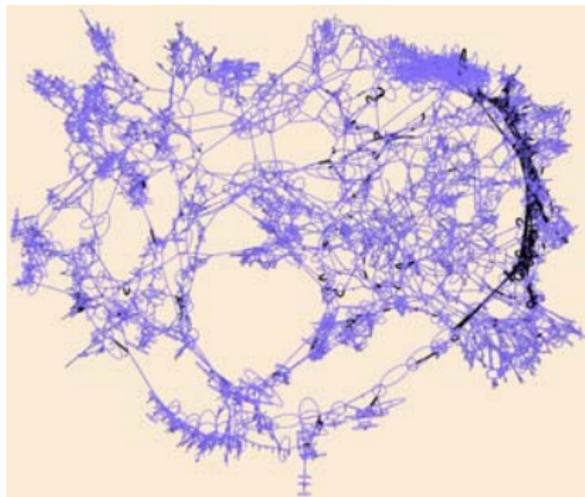


round 11

no control

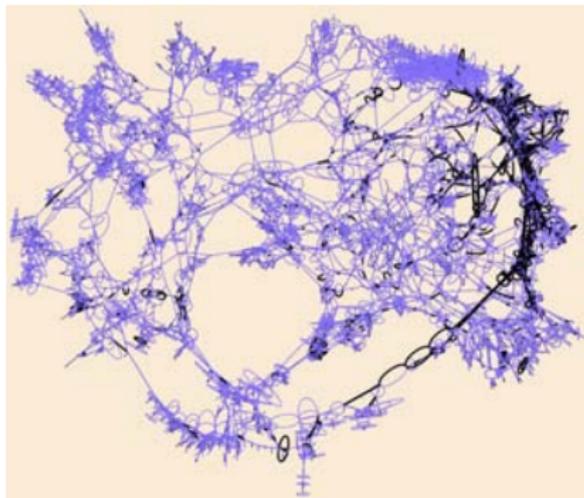


control

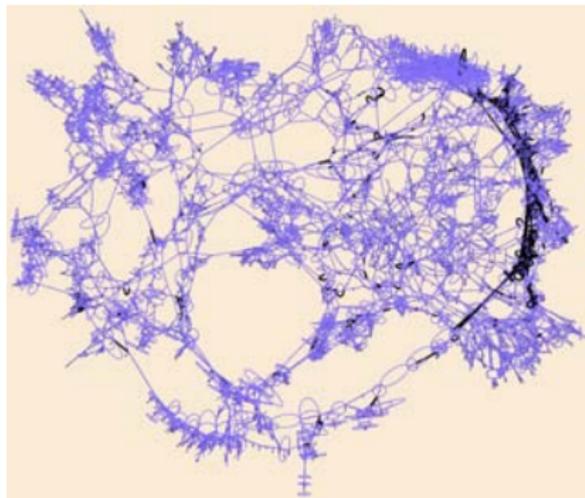


round 12

no control

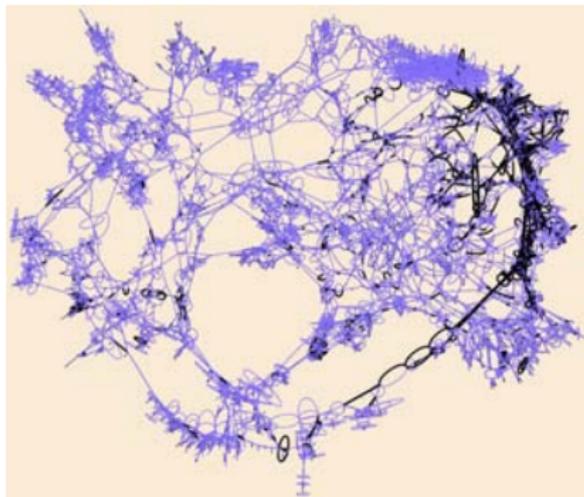


control

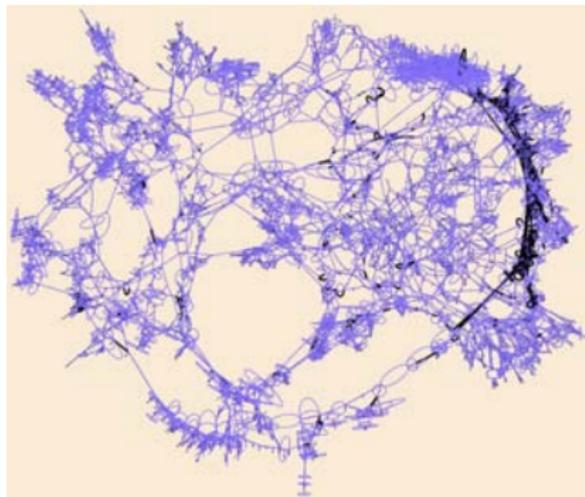


round 13

no control

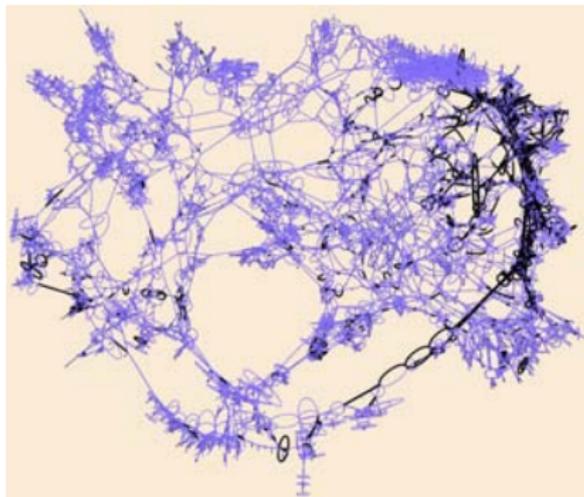


control

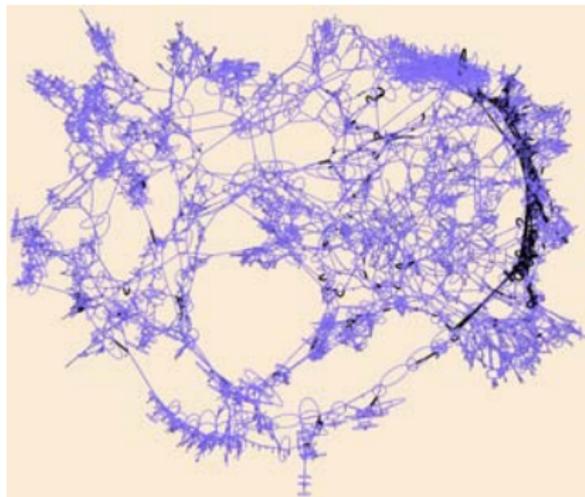


round 14

no control

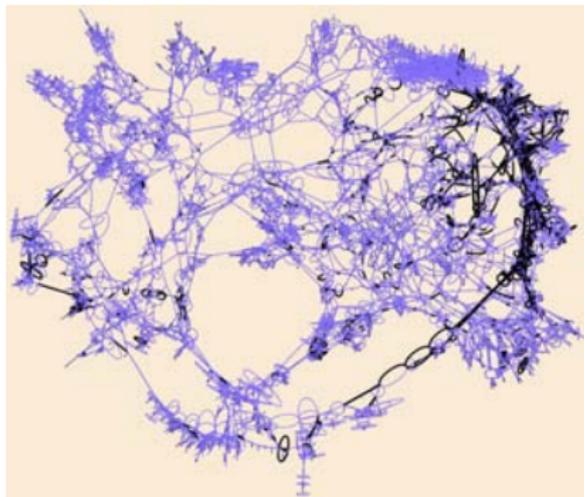


control

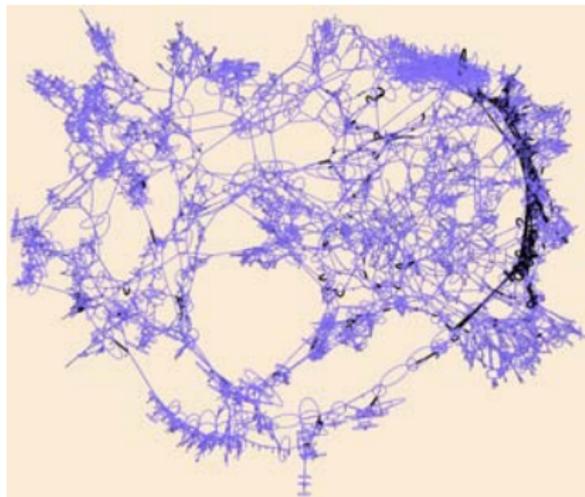


round 15

no control

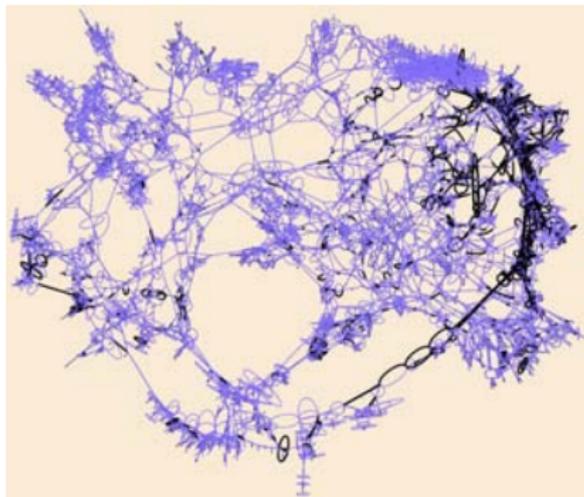


control

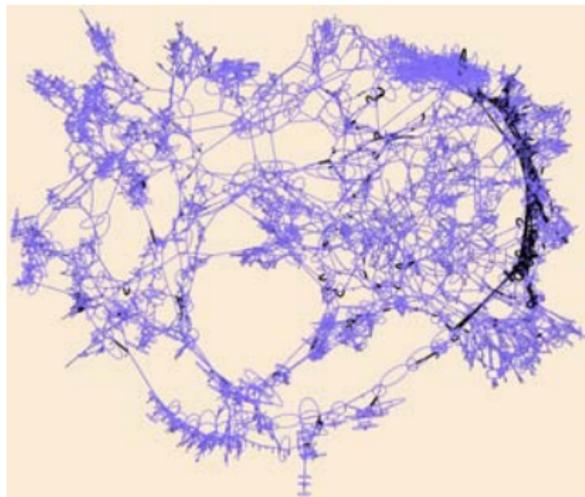


round 16

no control

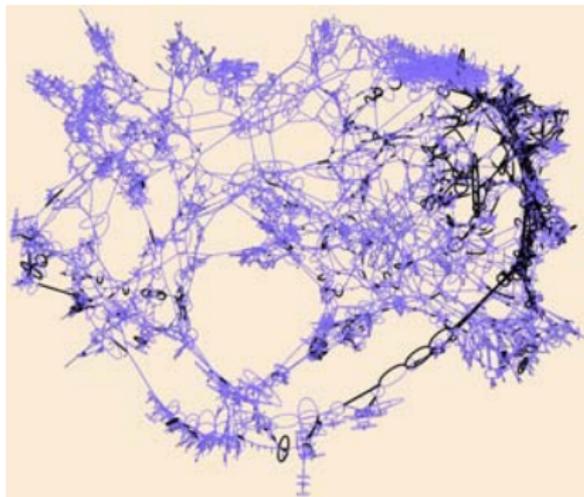


control

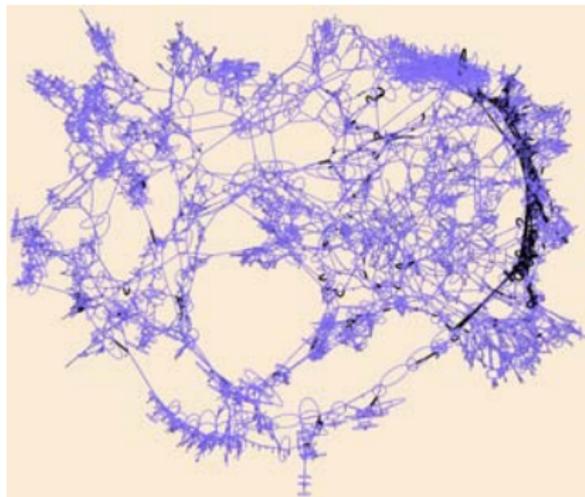


round 17

no control

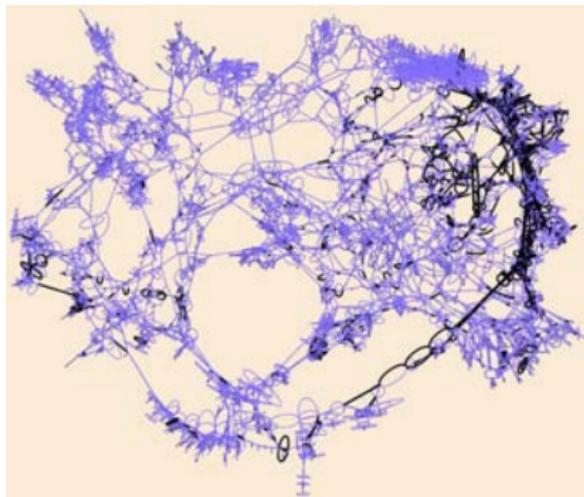


control

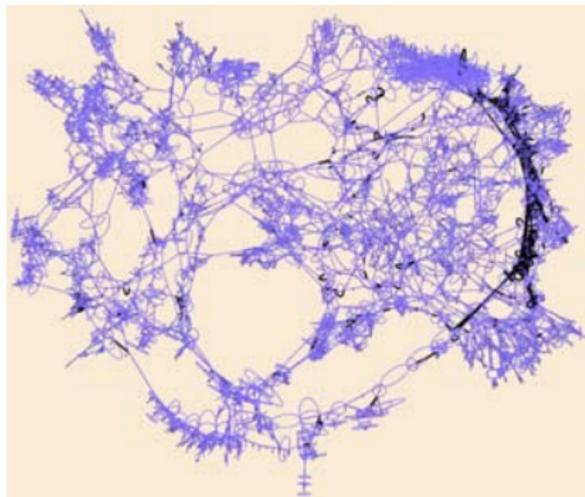


round 18

no control

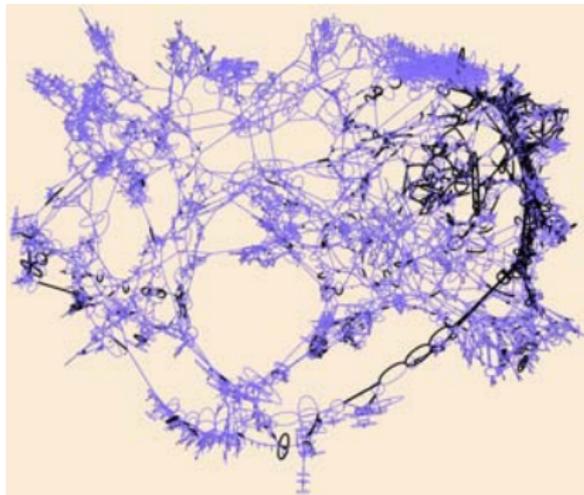


control

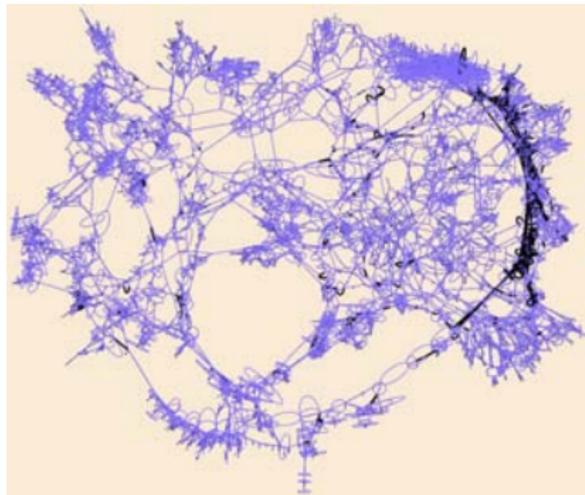


round 19

no control



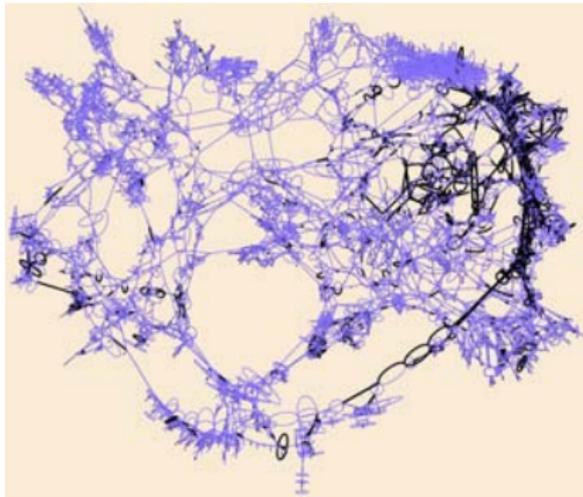
control



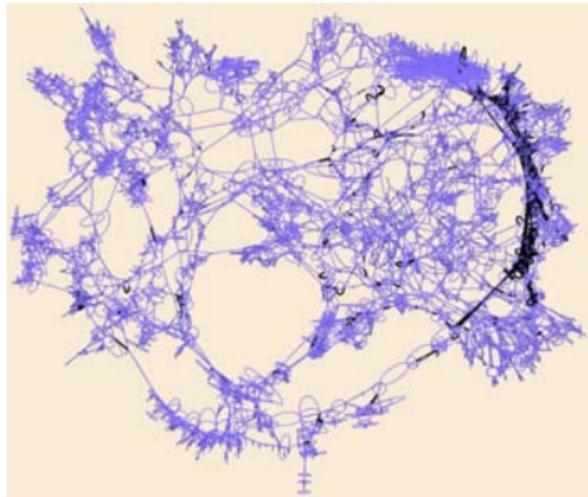
stable, yield = 75%, 2598 outaged lines

round 20

no control

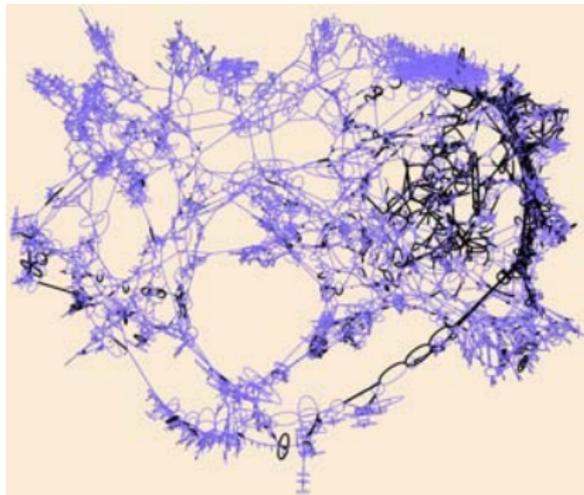


control



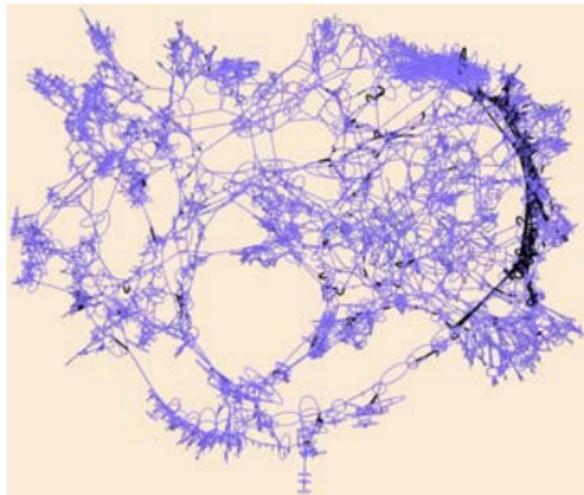
round 34

no control



stable, yield = 78%, 4425 outaged lines

control



(stable at round 19, yield = 75%, 2598 outaged lines)

Stochastic models: why?

- ▶ Noise should accumulate as the cascade unfolds
- ▶ Need a better way to account for line outages near the limit
 - ▶ Deterministic rule is too unforgiving and may not match anything “real”
 - ▶ Deterministic rule is numerically unstable

Stochastic line outage rule:

For each round \mathbf{r} , use a threshold $\mathbf{0} < \epsilon_{\mathbf{r}} < \mathbf{1}$.

Given a line \mathbf{k} (notation: $\tilde{f}_k =$ moving average of $|\text{flow}|$ on line k),

Stochastic line outage rule:

For each round r , use a threshold $0 < \epsilon_r < 1$.

Given a line k (notation: $\tilde{f}_k =$ moving average of $|\text{flow}|$ on line k),

- ▶ k is not outaged if $|\tilde{f}_k| < (1 - \epsilon_r)u_k$,
- ▶ k is outaged if $|\tilde{f}_k| > u_k$, and
- ▶ k is outaged with probability $1/2$, if $(1 - \epsilon_r)u_k \leq \tilde{f}_k \leq u_k$.

Stochastic line outage rule:

For each round r , use a threshold $0 < \epsilon_r < 1$.

Given a line k (notation: \tilde{f}_k = moving average of |flow| on line k),

- ▶ k is not outaged if $|\tilde{f}_k| < (1 - \epsilon_r)u_k$,
- ▶ k is outaged if $|\tilde{f}_k| > u_k$, and
- ▶ k is outaged with probability $1/2$, if $(1 - \epsilon_r)u_k \leq \tilde{f}_k \leq u_k$.

→ Example: $\epsilon_r = 0.01 + 0.05 * \lfloor r/10 \rfloor$

Technical note

- ▶ Using the above model, yield is *not* a differentiable function of control parameters
- ▶ As a result, no theoretical guarantee that stochastic gradients method will converge

Technical note

- ▶ Using the above model, yield is *not* a differentiable function of control parameters
- ▶ As a result, no theoretical guarantee that stochastic gradients method will converge

A smooth model:

Line k is outaged with probability $\mathcal{F}(\tilde{f}_k/u_k)$, where

- ▶ $\mathcal{F}(x) \rightarrow 1$ as $x \rightarrow +\infty$,
- ▶ $\mathcal{F}(x) \rightarrow 0$ as $x \rightarrow 0$,

Experiments using $\epsilon_r = 0.01 + 0.05 * \lfloor r/10 \rfloor$

- ▶ (Second cascade discussed above)
- ▶ Uncontrolled cascade is stable at round 34, with yield = 78% and 4425 outaged lines

Experiments using $\epsilon_r = 0.01 + 0.05 * \lfloor r/10 \rfloor$

- ▶ (Second cascade discussed above)
- ▶ Uncontrolled cascade is stable at round 34, with yield = 78% and 4425 outaged lines
- ▶ Compare to four control algorithms **c10**, **c15**, **c20**, **c25**
- ▶ Here, each control **cT** must achieve stability by round **T**, but can only shed load in rounds **1 - 10**.

Why?

Experiments using $\epsilon_r = 0.01 + 0.05 * \lfloor r/10 \rfloor$

- ▶ (Second cascade discussed above)
- ▶ Uncontrolled cascade is stable at round 34, with yield = 78% and 4425 outaged lines
- ▶ Compare to four control algorithms **c10**, **c15**, **c20**, **c25**
- ▶ Here, each control **cT** must achieve stability by round **T**, but can only shed load in rounds **1 - 10**.

Why?

Greater robustness is achieved by limiting the time frame

1000 runs

Option	DetY	MaxY	MinY	AveY	StddY
c10	37.49	38.93	0.00	7.54	9.55
c15	72.44	63.94	3.41	28.02	17.94
c20	75.19	73.04	0.00	32.24	21.30
c25	77.23	54.62	0.25	16.84	12.66
no control (34 rounds)	77.75	18.86	0.00	5.11	5.28