Abstraction-Based Segmental Simulation of Chemical Reaction Networks

Computational Methods in Systems Biology (CMSB 2022)

Martin Helfrich Milan Češka Jan Křetínský Štefan Martiček

September 14, 2022

Technical University of Munich
European Research Council
Established by the European Commission
BRNO UNIVERSITY OF TECHNOLOGY

Introduction

Chemical Reaction Networks (CRN):

- Model real-world biochemical systems
- Many applications (e.g. in medicine & molecular programming)

Introduction

Chemical Reaction Networks (CRN):

- Model real-world biochemical systems
- Many applications (e.g. in medicine & molecular programming)

Transient analysis:

- *"How does the system evolve?"*
- Hard because of complex dynamics, state-space explosion, stochasticity, stiffness, and multimodality
- Two approaches:
  - Direct (numerical)
  - Indirect (using many trajectories)

Introduction

Chemical Reaction Networks (CRN):

- Model real-world biochemical systems
- Many applications (e.g. in medicine & molecular programming)

Transient analysis:

- *"How does the system evolve?"*
- Hard because of complex dynamics, state-space explosion, stochasticity, stiffness, and multimodality
- Two approaches:
    - Direct (numerical)
    - Indirect (using many trajectories)

This Work

- **Goal**: compute many simulations fast

Introduction

Chemical Reaction Networks (CRN):

- Model real-world biochemical systems
- Many applications (e.g. in medicine & molecular programming)

Transient analysis:

- *"How does the system evolve?"*
- Hard because of complex dynamics, state-space explosion, stochasticity, stiffness, and multimodality
- Two approaches:
    - Direct (numerical)
    - Indirect (using many trajectories)

This Work

- **Goal**: compute many simulations fast
- **Idea**: using memorization

CRNs

Example: Viral Infection

Species	$\mathrm{RNA}, \mathrm{DNA}, \mathrm{V}, \mathrm{P}$
Initial state	$(1 \times \mathrm{RNA})$
End time	200s
Reactions	$\mathrm{DNA} + \mathrm{P} \xrightarrow{0.00001125 \cdot \mathrm{DNA} \cdot \mathrm{P}} \mathrm{V}$ $\mathrm{RNA} \xrightarrow{1000 \cdot \mathrm{RNA}} \mathrm{RNA} + \mathrm{P}$ $\mathrm{DNA} \xrightarrow{0.025 \cdot \mathrm{DNA}} \mathrm{DNA} + \mathrm{RNA}$ $\mathrm{RNA} \xrightarrow{1 \cdot \mathrm{RNA}} \mathrm{DNA} + \mathrm{RNA}$ $\mathrm{RNA} \xrightarrow{0.25 \cdot \mathrm{RNA}} \emptyset$ $\mathrm{P} \xrightarrow{1.9985 \cdot \mathrm{P}} \emptyset$

- Evolution governed by Chemical Master Equation
- Gives rise to discrete-space continuous-time Markov chain (CTMC)

Simulation

Gillespie's stochastic simulation algorithm (SSA) [3]

- Sample one reaction at a time

Gillespie's stochastic simulation algorithm (SSA) [3]

- Sample one reaction at a time

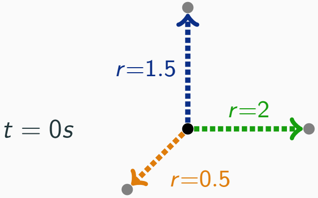
$t = 0s$ s_{init}

Start in initial state.

Simulation

Gillespie's stochastic simulation algorithm (SSA) [3]

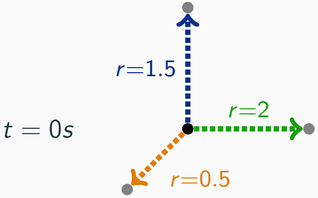
- Sample one reaction at a time



Compute rate of all reactions according to their propensity functions.

Gillespie's stochastic simulation algorithm (SSA) [3]

- Sample one reaction at a time



Time until the next reaction: $\Delta t \sim EXP(0.5+2+1.5)$
Probability of reactions: $\frac{0.5}{4}, \frac{2}{4}, \frac{1.5}{4}$

Simulation

Gillespie's stochastic simulation algorithm (SSA) [3]

- Sample one reaction at a time

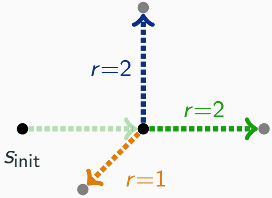
$t = 0.18s$

Simulation

Gillespie's stochastic simulation algorithm (SSA) [3]

- Sample one reaction at a time

$t = 0.18s$

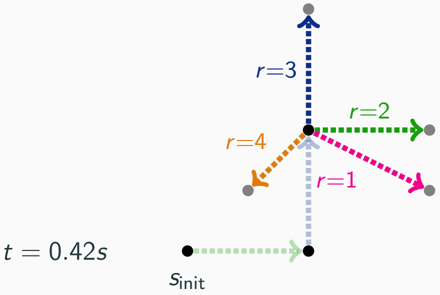


Time until the next reaction: $\Delta t \sim EXP(1+2+2)$
Probability of reactions: $\frac{1}{5}, \frac{2}{5}, \frac{2}{5}$

Simulation

Gillespie's stochastic simulation algorithm (SSA) [3]

- Sample one reaction at a time

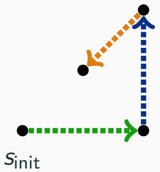


Time until the next reaction: $\Delta t \sim EXP(4+2+3+1)$
Probability of reactions: $\frac{4}{10}, \frac{2}{10}, \frac{3}{10}, \frac{1}{10}$

Gillespie's stochastic simulation algorithm (SSA) [3]

- Sample one reaction at a time
- May take a long time

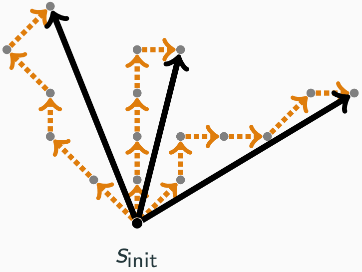
$t = 0.47s$



Segmental Simulation

Precompute k short trajectories (called segments) for each state.

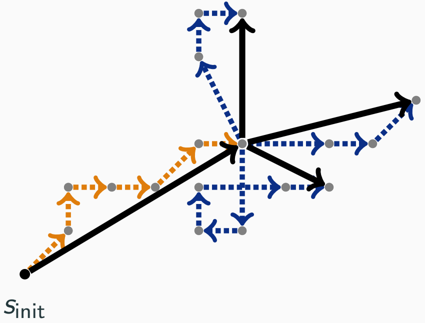
→ Simulate by sampling segments instead of single reactions.



Segmental Simulation

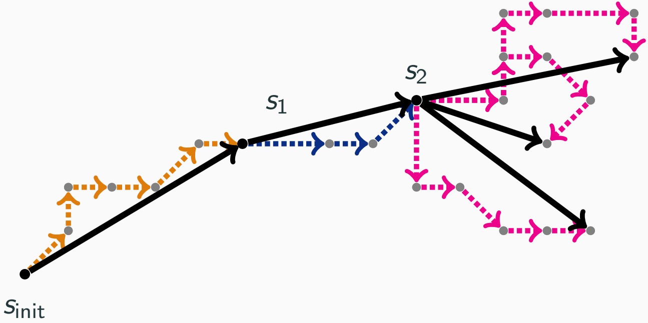
Precompute k short trajectories (called segments) for each state.

$\rightarrow$ Simulate by sampling segments instead of single reactions.



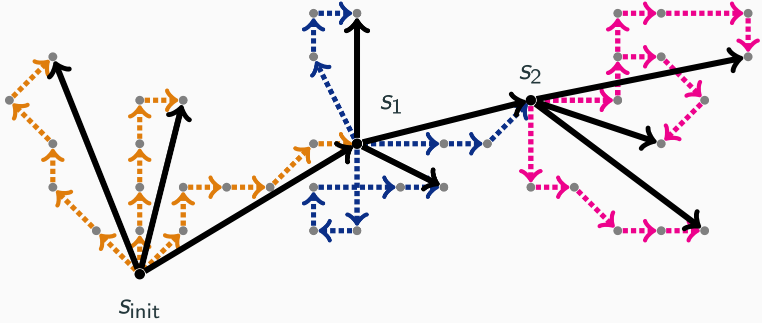
Segmental Simulation

Precompute k short trajectories (called segments) for each state.
$\rightarrow$ Simulate by sampling segments instead of single reactions.



Segmental Simulation

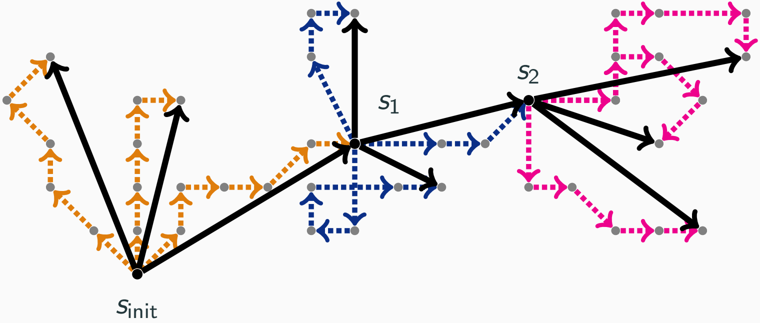
Precompute k short trajectories (called segments) for each state.
$\rightarrow$ Simulate by sampling segments instead of single reactions.



- much faster!

Segmental Simulation

Precompute k short trajectories (called segments) for each state.
→ Simulate by sampling segments instead of single reactions.



- much faster!
- Problem: many states → too inefficient

Abstraction-Based Segmental Simulation

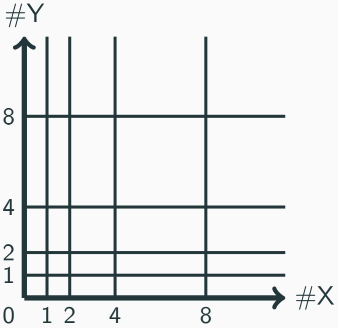
- Idea: Do not treat every state separately!

Abstraction-Based Segmental Simulation

- Idea: Do not treat every state separately!
- States with similar species counts have similar propensities
  $\rightarrow$ their behave similarly

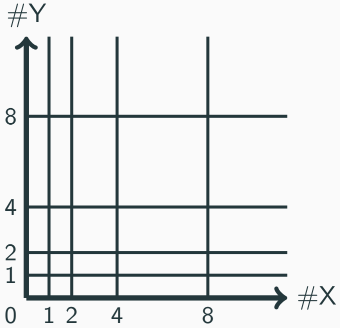
Abstraction-Based Segmental Simulation

- Idea: Do not treat every state separately!
- States with similar species counts have similar propensities
  $\rightarrow$ their behave similarly
- Population-level abstraction: split state-space into regions (called abstract states)



Abstraction-Based Segmental Simulation

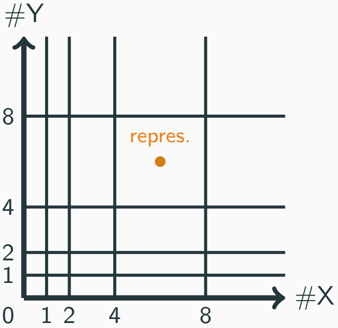
- Idea: Do not treat every state separately!
- States with similar species counts have similar propensities
  $\rightarrow$ their behave similarly
- Population-level abstraction: split state-space into regions (called abstract states)



- Population levels grow exponentially

Abstraction-Based Segmental Simulation

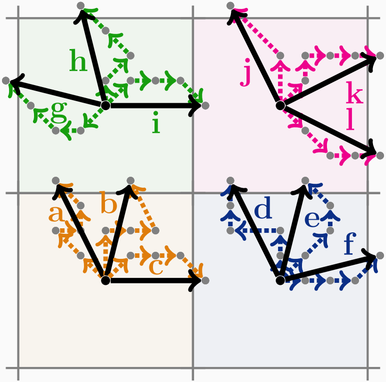
- Idea: Do not treat every state separately!
- States with similar species counts have similar propensities
  $\rightarrow$ their behave similarly
- Population-level abstraction: split state-space into regions (called abstract states)



- Population levels grow exponentially
- Choose representative for each abstract state (usually center)

Abstraction-Based Segmental Simulation

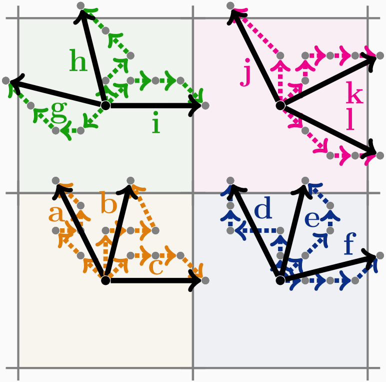
Only precompute k segments for each representative.



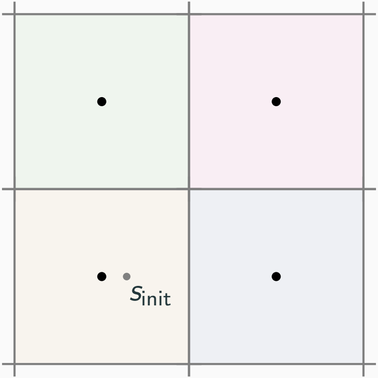
MEMORY
($k = 3$)

Segments end when they leave the abstract state.
→ Intuition: "*significant change*"

Only precompute k segments for each representative.



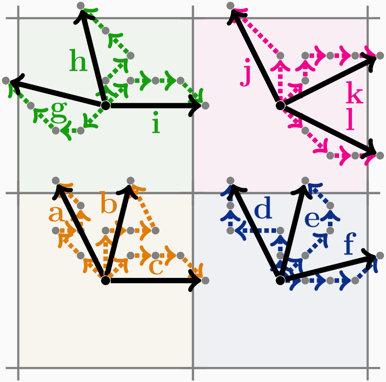
MEMORY
$(k = 3)$



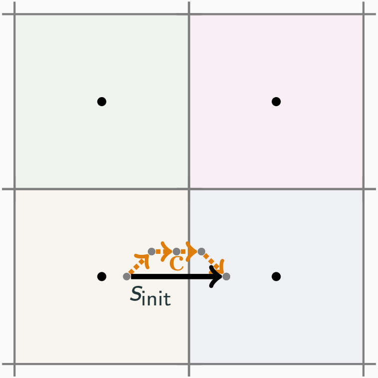
SIMULATION

Abstraction-Based Segmental Simulation

Only precompute k segments for each representative.



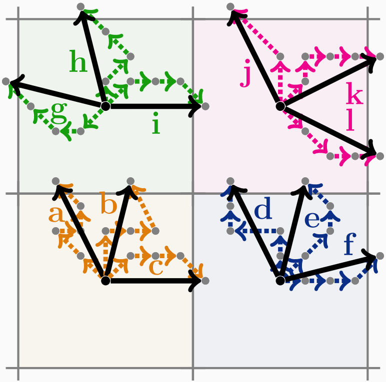
MEMORY
($k = 3$)



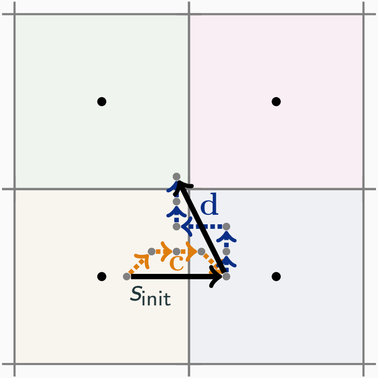
SIMULATION

Abstraction-Based Segmental Simulation

Only precompute k segments for each representative.



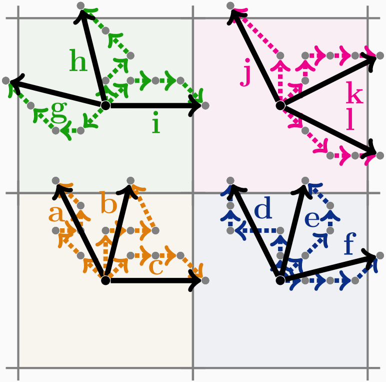
MEMORY
$(k = 3)$



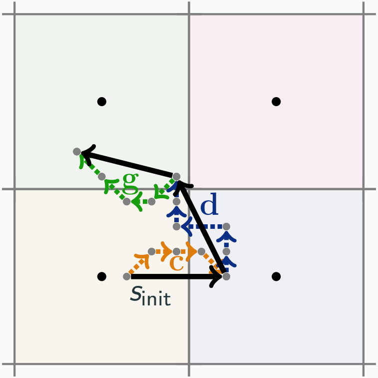
SIMULATION

Abstraction-Based Segmental Simulation

Only precompute k segments for each representative.



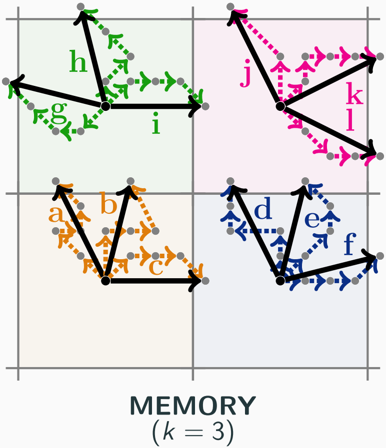
MEMORY
$(k = 3)$

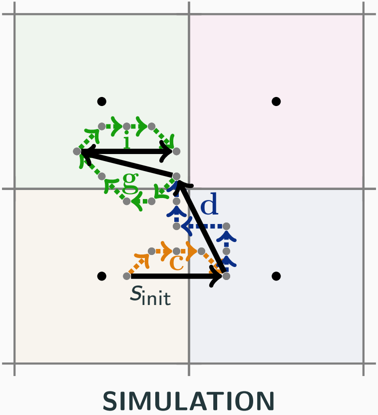


SIMULATION

Abstraction-Based Segmental Simulation

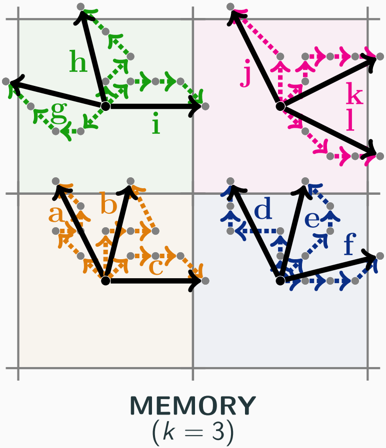
Only precompute k segments for each representative.

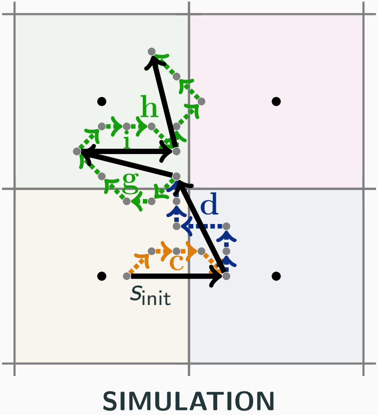




Abstraction-Based Segmental Simulation

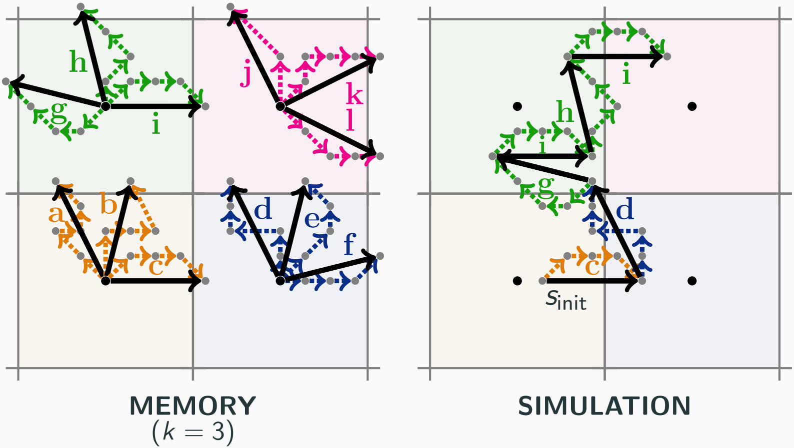
Only precompute k segments for each representative.





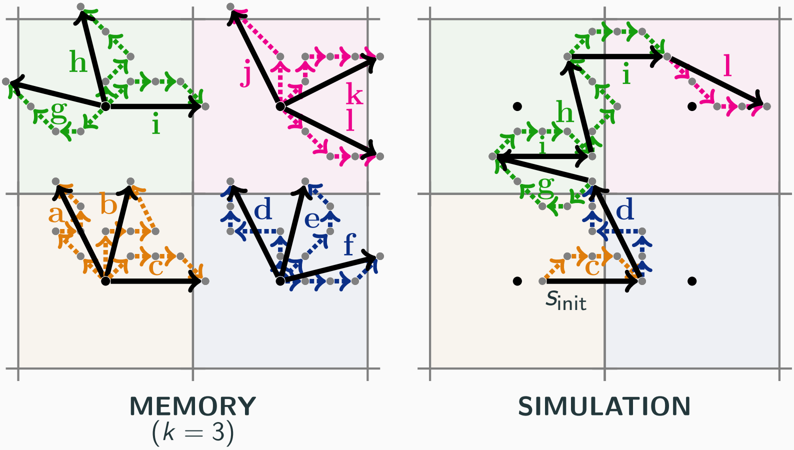
Abstraction-Based Segmental Simulation

Only precompute k segments for each representative.



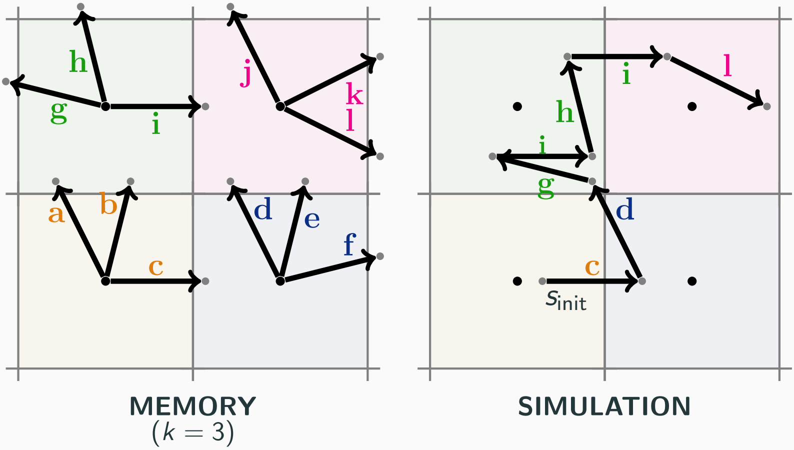
Abstraction-Based Segmental Simulation

Only precompute k segments for each representative.



Abstraction-Based Segmental Simulation

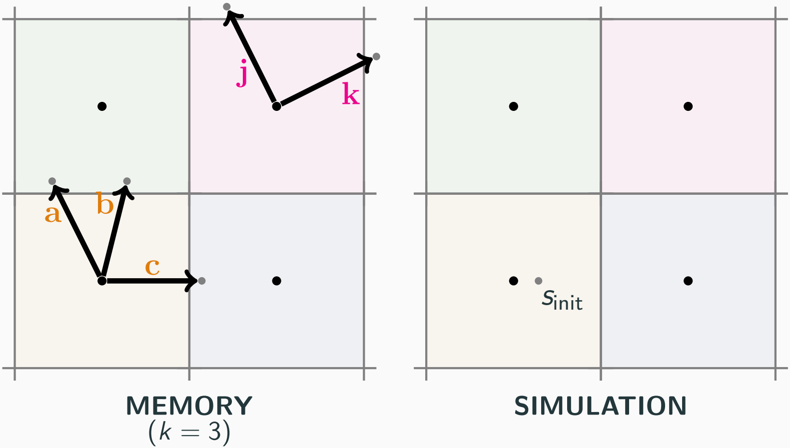
Only precompute k segments for each representative.



To save memory: Work with summaries instead of segments.

Abstraction-Based Segmental Simulation

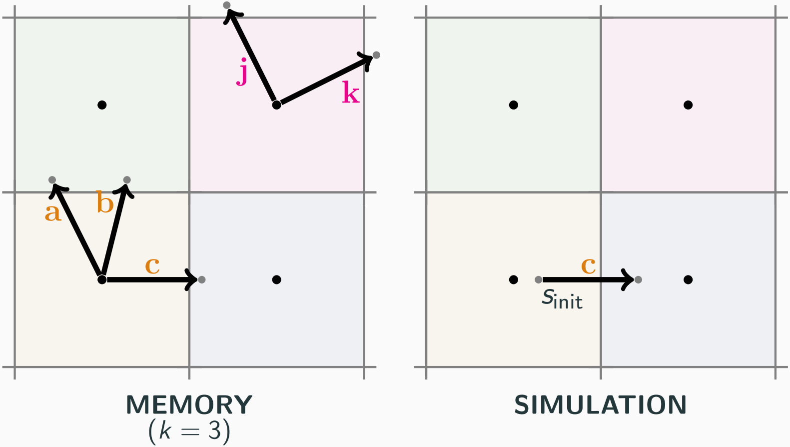
Only precompute k segments for each representative.



Lazy: Do not precompute but fill memory on-the-fly!

Abstraction-Based Segmental Simulation

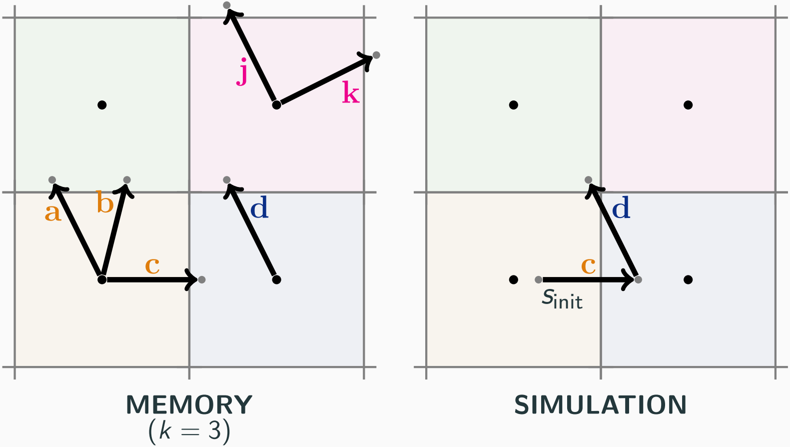
Only precompute k segments for each representative.



Lazy: Do not precompute but fill memory on-the-fly!

Abstraction-Based Segmental Simulation

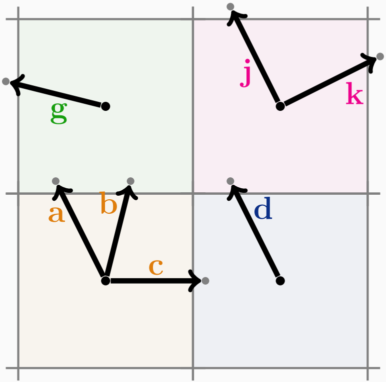
Only precompute k segments for each representative.

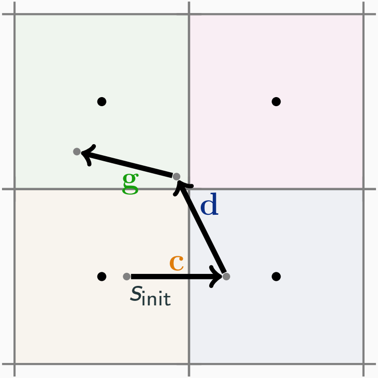


Lazy: Do not precompute but fill memory on-the-fly!

Abstraction-Based Segmental Simulation

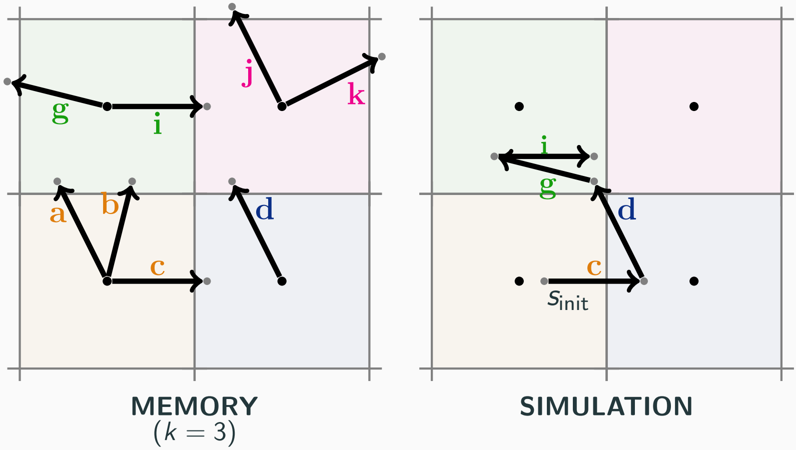
Only precompute k segments for each representative.





Lazy: Do not precompute but fill memory on-the-fly!

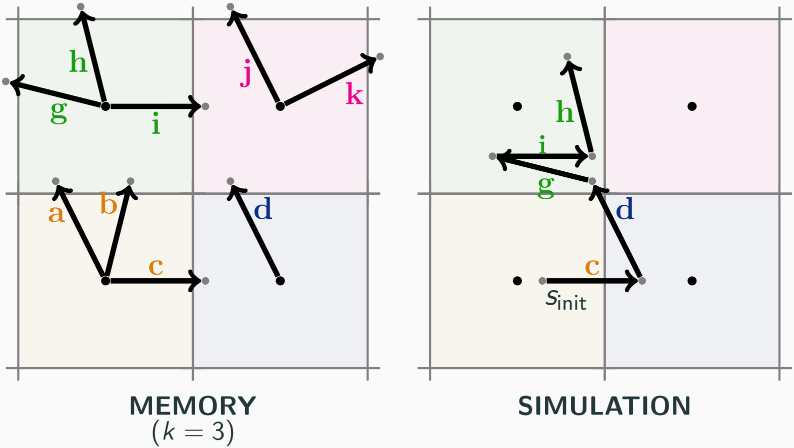
Only precompute k segments for each representative.



Lazy: Do not precompute but fill memory on-the-fly!

Abstraction-Based Segmental Simulation

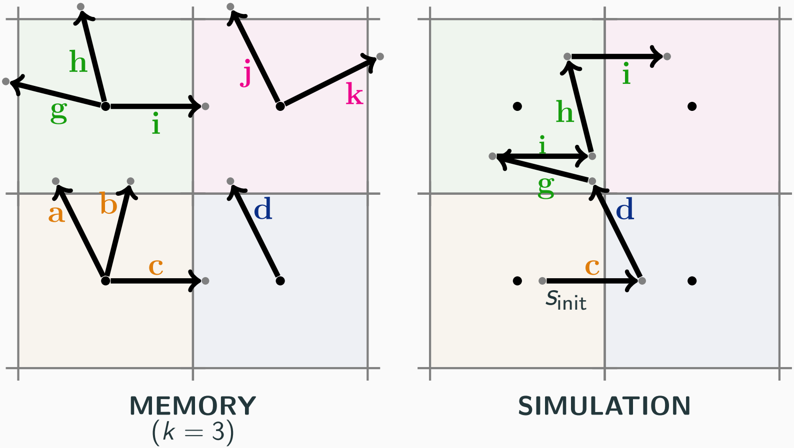
Only precompute k segments for each representative.



Lazy: Do not precompute but fill memory on-the-fly!

Abstraction-Based Segmental Simulation

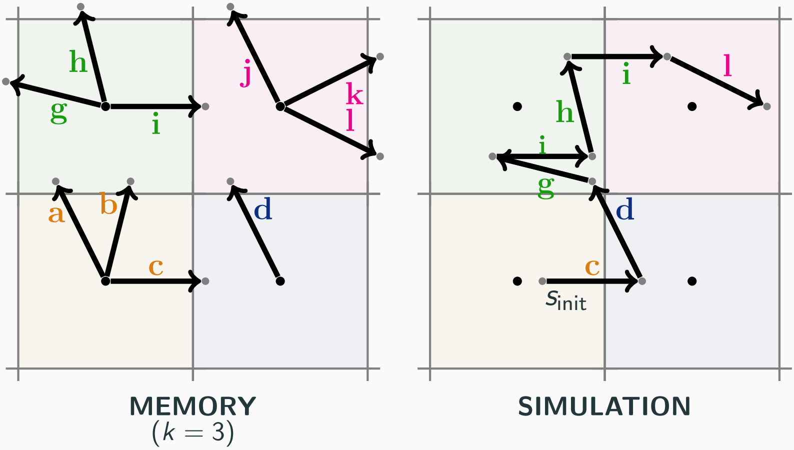
Only precompute k segments for each representative.



Lazy: Do not precompute but fill memory on-the-fly!

Abstraction-Based Segmental Simulation

Only precompute k segments for each representative.



Lazy: Do not precompute but fill memory on-the-fly!

Introduced Inaccuracy

There are two error sources:

Introduced Inaccuracy

There are two error sources:

1. Limited number of memorized segments:
   - Cannot faithfully represent actual segment distribution
   - Error vanishes for $k \to \infty$

Introduced Inaccuracy

There are two error sources:

1. Limited number of memorized segments:
   - Cannot faithfully represent actual segment distribution
   - Error vanishes for $k \to \infty$
2. Using representative's segments
   - Similar species counts $\to$ similar propensities $\to$ similar segments
   - Error gets smaller if we add more population levels

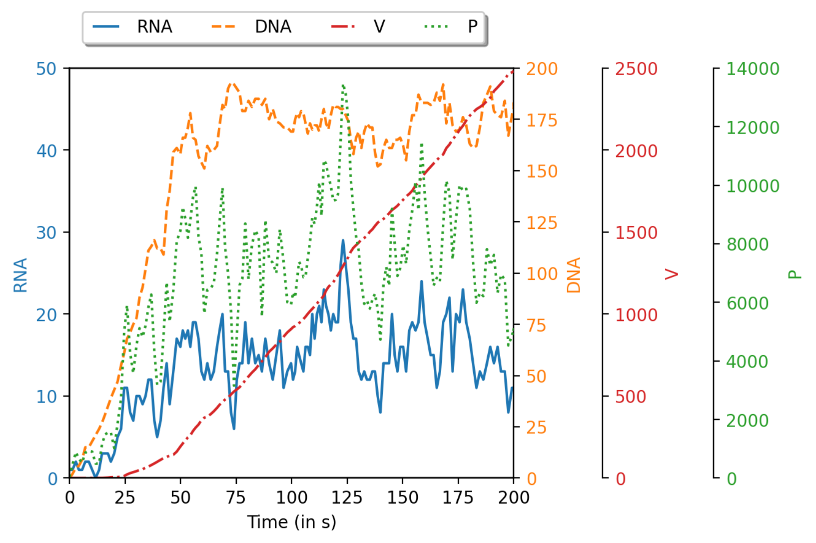
Evaluation - Accuracy

Example: Viral Infection

Species	$\mathrm{RNA}, \mathrm{DNA}, \mathrm{V}, \mathrm{P}$
Initial state	$(1 \times \mathrm{RNA})$
End time	200s
Reactions	$\mathrm{DNA} + \mathrm{P} \xrightarrow{0.00001125 \cdot \mathrm{DNA} \cdot \mathrm{P}} \mathrm{V}$ $\mathrm{RNA} \xrightarrow{1000 \cdot \mathrm{RNA}} \mathrm{RNA} + \mathrm{P}$ $\mathrm{DNA} \xrightarrow{0.025 \cdot \mathrm{DNA}} \mathrm{DNA} + \mathrm{RNA}$ $\mathrm{RNA} \xrightarrow{1 \cdot \mathrm{RNA}} \mathrm{DNA} + \mathrm{RNA}$ $\mathrm{RNA} \xrightarrow{0.25 \cdot \mathrm{RNA}} \emptyset$ $\mathrm{P} \xrightarrow{1.9985 \cdot \mathrm{P}} \emptyset$

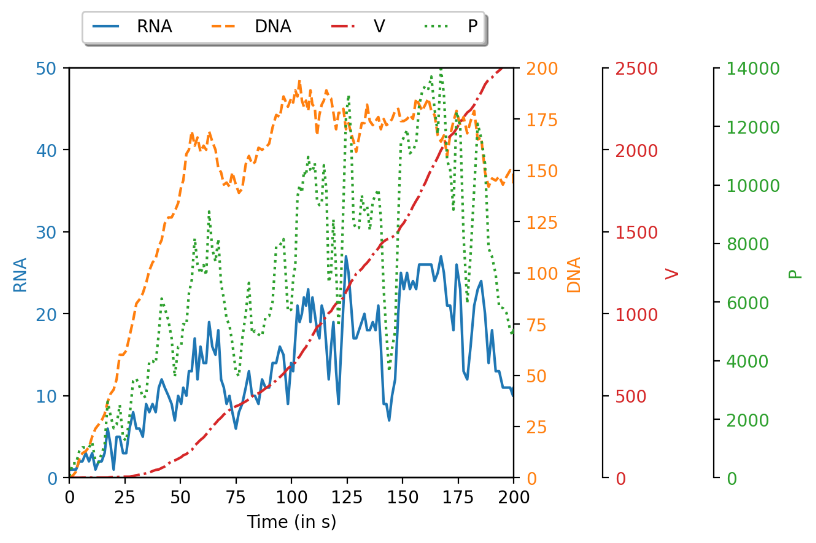
Evaluation - Accuracy

SSA - Simulation 1



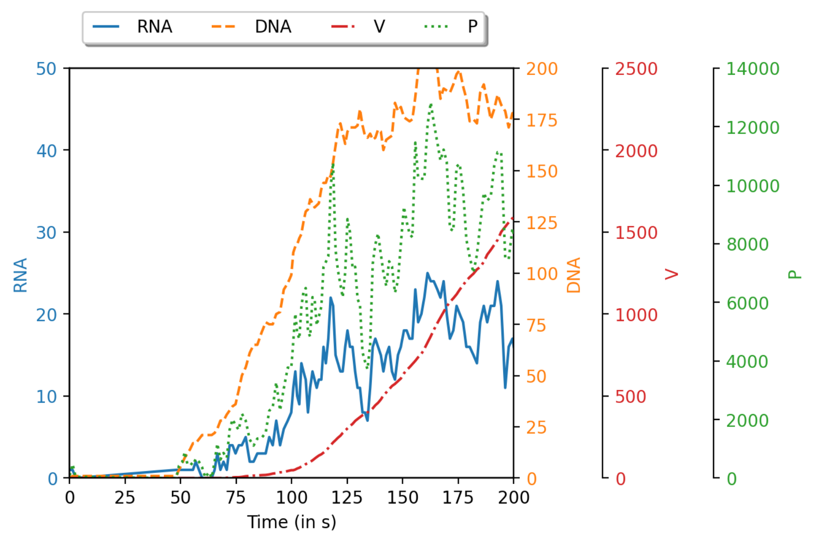
Evaluation - Accuracy

SSA - Simulation 2



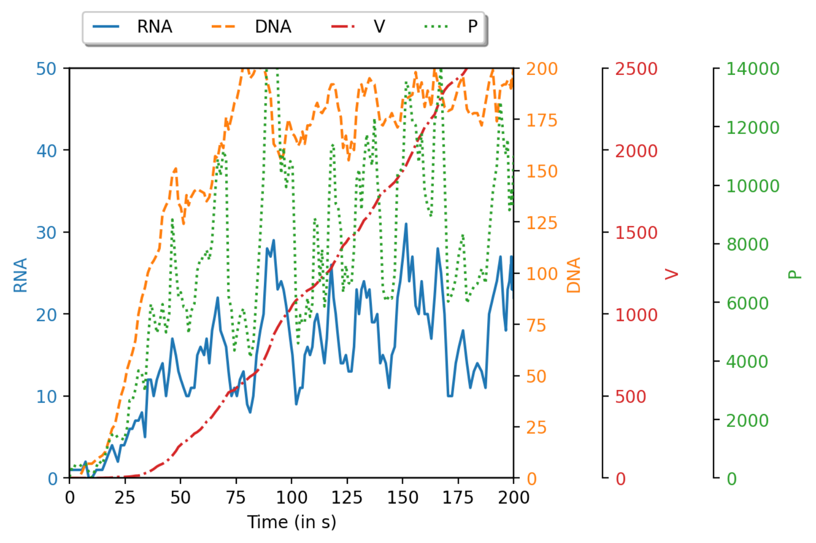
Evaluation - Accuracy

SSA - Simulation 3



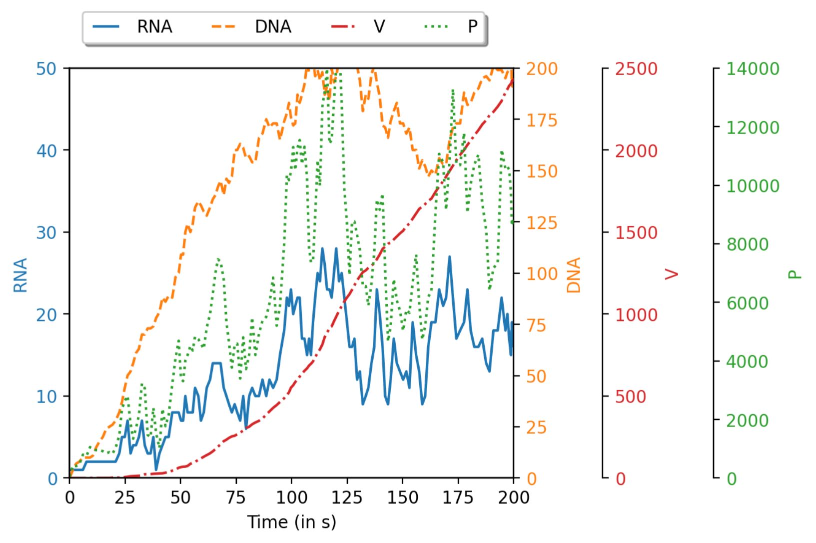
Evaluation - Accuracy

SEG - Simulation 4



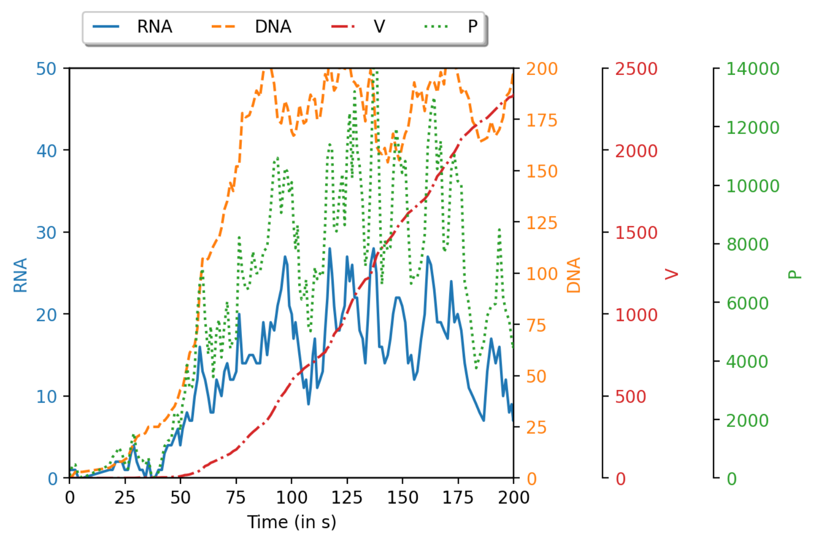
Evaluation - Accuracy

SEG - Simulation 5

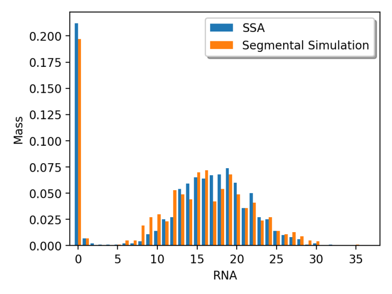


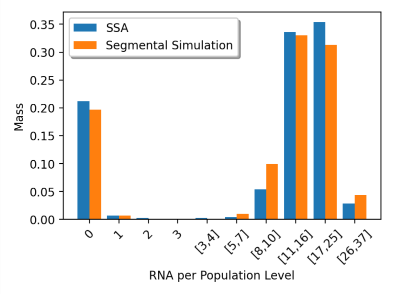
Evaluation - Accuracy

SEG - Simulation 6



Evaluation - Accuracy

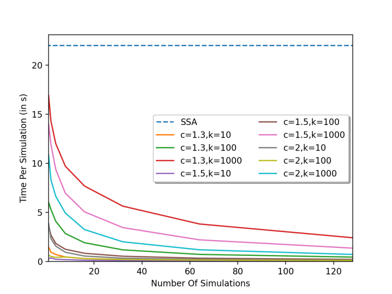


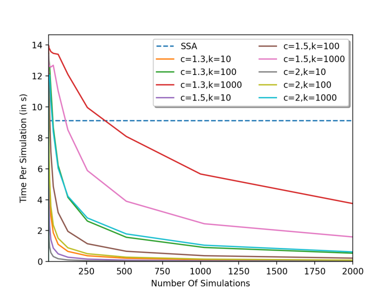


	Mean	Var
SSA	13.6	2878
SEG	13.5	2685

Evaluation - Performance

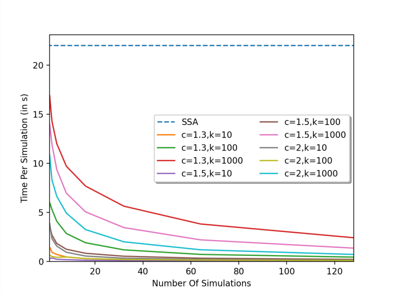
Speed-up:

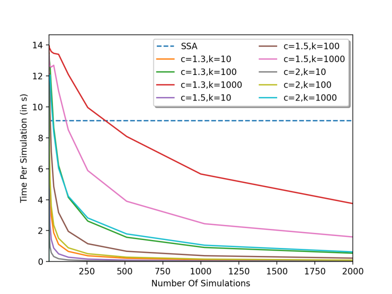




Evaluation - Performance

Speed-up:

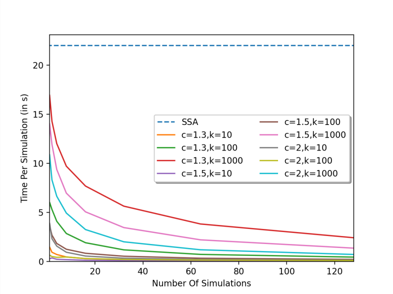


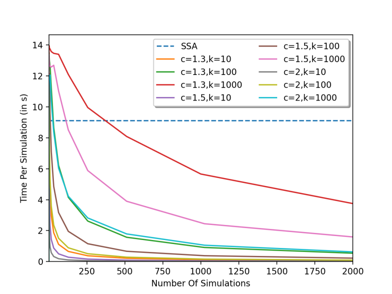


- Depends on model and target accuracy

Evaluation - Performance

Speed-up:

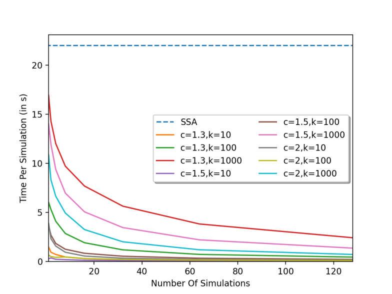


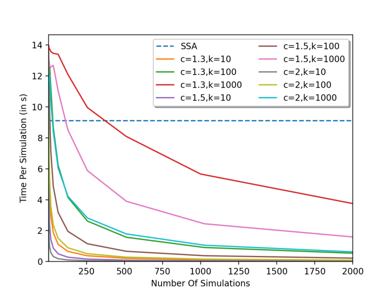


- Depends on model and target accuracy
- Accelerates with number of simulations

Evaluation - Performance

Speed-up:

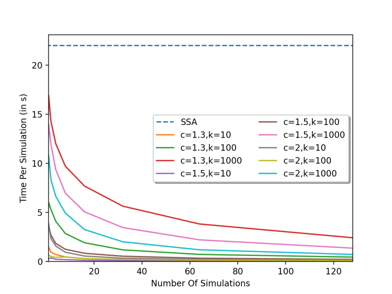


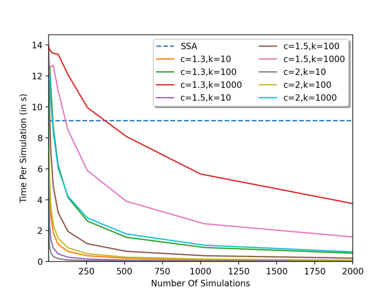


- Depends on model and target accuracy
- Accelerates with number of simulations
- Can already be faster than SSA in first simulation

Evaluation - Performance

Speed-up:





- Depends on model and target accuracy
- Accelerates with number of simulations
- Can already be faster than SSA in first simulation
- Memorization: trade-off between speed and memory

Evaluation - Comparison

Oversimplified comparison with other approaches:

Approach	Speed-up	Accuracy
SSA [3]	1x	perfect
τ-leaping [2]	~5x	very good
Hybrid Simulation [4]	~50x	good
Deep Learning[1] [1]	~100x	good
Segmental Simulation[2]	~200x	good

[1] requires precomputed data and long training period
[2] significant memory requirement

Future Work

- Handle larger models: adaptive memory allocation

Future Work

- Handle larger models: adaptive memory allocation
- Formal error bounds

Future Work

- Handle larger models: adaptive memory allocation
- Formal error bounds
- Segmental simulation as general framework for accelerating simulations

Future Work

- Handle larger models: adaptive memory allocation
- Formal error bounds
- Segmental simulation as general framework for accelerating simulations

Thank you!

References

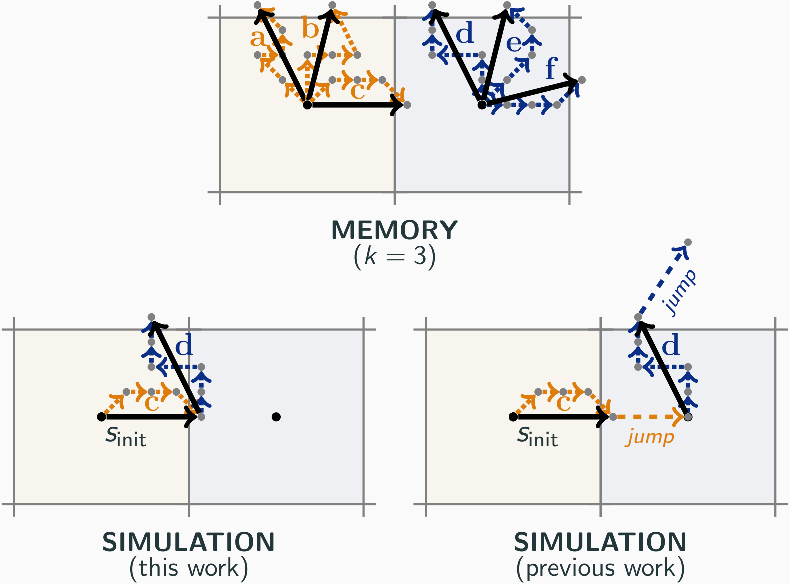
[1] Cairoli, F., Carbone, G., Bortolussi, L.: Abstraction of markov population dynamics via generative adversarial nets. In: CMSB'21. pp. 19–35. Springer (2021)

[2] Cao, Y., Gillespie, D.T., Petzold, L.R.: Efficient step size selection for the tau-leaping simulation method. The Journal of chemical physics **124**(4), 044109 (2006)

[3] Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. The journal of physical chemistry **81**(25), 2340–2361 (1977)

[4] Hepp, B., Gupta, A., Khammash, M.: Adaptive hybrid simulations for multiscale stochastic reaction networks. The Journal of chemical physics **142**(3), 034118 (2015)

Importance of Concrete State Information



Importance of Concrete State Information

- Only abstract states $\rightarrow$ rounding
- Rounding looses progress in all but one dimension

Importance of Concrete State Information

- Only abstract states $\rightarrow$ rounding
- Rounding looses progress in all but one dimension

Example: Rounding Problem

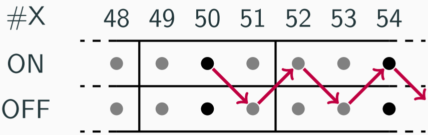
Species	$\mathrm{ON}, \mathrm{OFF}, \mathrm{X}$
Initial state	$(1 \times \mathrm{ON}, 50 \times \mathrm{X})$
Reactions	$\mathrm{ON} \rightarrow \mathrm{OFF} + \mathrm{X}$ $\mathrm{OFF} \rightarrow \mathrm{ON} + \mathrm{X}$

Importance of Concrete State Information

- Only abstract states → rounding
- Rounding looses progress in all but one dimension

Example: Rounding Problem

Species	$\mathrm{ON}, \mathrm{OFF}, \mathrm{X}$
Initial state	$(1 \times \mathrm{ON}, 50 \times \mathrm{X})$
Reactions	$\mathrm{ON} \rightarrow \mathrm{OFF} + \mathrm{X}$ $\mathrm{OFF} \rightarrow \mathrm{ON} + \mathrm{X}$



SIMULATION
(this work)

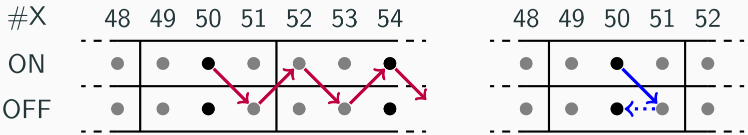
X does grow. ✓

Importance of Concrete State Information

- Only abstract states → rounding
- Rounding looses progress in all but one dimension

Example: Rounding Problem

Species	$\mathrm{ON}, \mathrm{OFF}, \mathrm{X}$
Initial state	$(1 \times \mathrm{ON}, 50 \times \mathrm{X})$
Reactions	$\mathrm{ON} \to \mathrm{OFF} + \mathrm{X}$ $\mathrm{OFF} \to \mathrm{ON} + \mathrm{X}$



SIMULATION (this work)

SIMULATION (previous work)

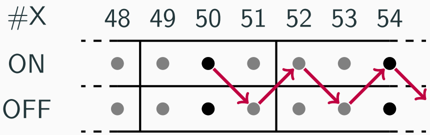
X does grow. ✓

Importance of Concrete State Information

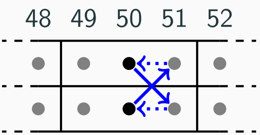
- Only abstract states → rounding
- Rounding looses progress in all but one dimension

Example: Rounding Problem

Species	$\mathrm{ON}, \mathrm{OFF}, \mathrm{X}$
Initial state	$(1 \times \mathrm{ON}, 50 \times \mathrm{X})$
Reactions	$\mathrm{ON} \rightarrow \mathrm{OFF} + \mathrm{X}$ $\mathrm{OFF} \rightarrow \mathrm{ON} + \mathrm{X}$



SIMULATION
(this work)



SIMULATION
(previous work)

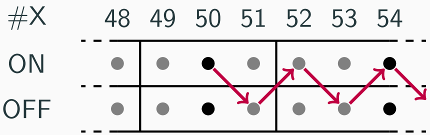
X does grow. ✓

Importance of Concrete State Information

- Only abstract states → rounding
- Rounding looses progress in all but one dimension

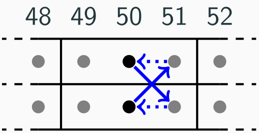
Example: Rounding Problem

Species	$\mathrm{ON}, \mathrm{OFF}, \mathrm{X}$
Initial state	$(1 \times \mathrm{ON}, 50 \times \mathrm{X})$
Reactions	$\mathrm{ON} \rightarrow \mathrm{OFF} + \mathrm{X}$ $\mathrm{OFF} \rightarrow \mathrm{ON} + \mathrm{X}$



SIMULATION
(this work)

X does grow.



SIMULATION
(previous work)

X does NOT grow.

Lazy Algorithm

Inputs : $\mathcal{N}$ (CRN), k (number of segments), c (partitioning parameter), t_{end} (time horizon), s_{init} (initial state) and m (number of simulations)

Output: list of m segmental simulations

```
simulations := [ ];
memory := {};                    // mapping each abstract state to a list of segments
for 1 to m do
    s := s_init;  t := 0;  simulation := [(s, t)];
    while t < t_end do
        a := abstractState_c(s);
        if |memory(a)| < k then
            segment := sampleNewSegm(N, a.representative);    // sample new segment
            memory(a).add(segment);                            // save it for reuse
        else
            segment := chooseUniformlyFrom(memory(a));         // reuse old segment
        end
        // apply segment's relative effects
        s := s + segment.Δ_state;  t := t + segment.Δ_time;
        simulation.add((s, t));
    end
    simulations.add(simulation);
end
return simulations
```

More Data - Speed

Mod.	SSA	SEG k=10			SEG k=100			SEG k=1000		
		c=2	c=1.5	c=1.3	c=2	c=1.5	c=1.3	c=2	c=1.5	c=1.3
PP	0.014s	70x	70x	70x	70x	70x	23x	28x	23x	12x
VI	0.88s	730x	380x	180x	100x	48x	17x	8.6x	4.8x	2.9x
TS	22s	360x	360x	340x	390x	350x	280x	250x	190x	110x
RP	9.1s	760x	540x	320x	300x	140x	62x	54x	21x	7.4x

Table 1: Average run-time of one SSA simulation and the speedup factor of segmental simulation when computing 10,000 simulations with different abstraction parameters.

More Data - Memory

Mod.	SEG k=10			SEG k=100			SEG k=1000		
	c=2	c=1.5	c=1.3	c=2	c=1.5	c=1.3	c=2	c=1.5	c=1.3
PP	25kb	61kb	130kb	250kb	570kb	1.3mb	2.2mb	4.8mb	11mb
VI	210kb	730kb	2.0mb	1.8mb	4.8mb	13mb	11mb	25mb	53mb
TS	1.2mb	3.0mb	8.7mb	15mb	37mb	85mb	100mb	250mb	550mb
RP	3.8mb	12mb	34mb	43mb	120mb	300mb	310mb	760mb	1.0gb

Table 2: Size of segmental abstraction after 10,000 simulations for different parameters.