Graph Dynamical Systems – A Mathematical Framework for Interaction-Based Systems, Their Analysis and Simulations

Henning S. Mortveit

Department of Mathematics &
NDSSL, Virginia Bioinformatics Institute
Virginia Polytechnic Institute and State University

Discrete Models in Systems Biology Workshop
SAMSI, 5 December, 2008
Outline

1 Background and Terminology
   - Graph Dynamical Systems
   - Applications Areas – Complex Systems
   - Research Theme/Approach
   - Examples of GDS Theory

2 Modeling and Applications
   - Gene Annotation
   - Epidemiology in Large Urban Areas
   - An HIV Model for Mutations

3 Summary, Links, References
Overview and Initial Remarks:

- **The Topic:** Modeling, simulation and analysis of phenomena/dynamics over networks.

- **Motivation:** Simulations used extensively to study distributed/networked systems. Tools/theory for validation and theoretical insight are needed.

- **Theoretical framework:** Graph Dynamical Systems. Brief introduction, their roles in applications and some examples of their theory.

- **Applications:**

  - **Gene annotation.** SDS-based predictions of gene functions – a cautionary tale.
  
  - **Epidemiology in large urban areas.** Individual-based modeling and implementation using mobility, contact graph dynamics, and disease propagation. The role of theory. Specialized hardware.
  
  - **An HIV model for mutations.**
Graph Dynamical Systems – GDS:

A Graph Dynamical System is a triple consisting of:

- A graph $Y$ with vertex set $v[Y] = \{1, 2, \ldots, n\}$.
Graph Dynamical Systems – GDS:

A Graph Dynamical System is a triple consisting of:

- A graph \( Y \) with vertex set \( v[Y] = \{1, 2, \ldots, n\} \).
- For each vertex \( i \) a state \( x_i \in K \) (e.g. \( K = \{0, 1\} \)) and a \( Y \)-local function \( F_i : K^n \rightarrow K^n \)

\[
F_i(x = (x_1, x_2, \ldots, x_n)) = (x_1, \ldots, x_{i-1}, f_i(x[i]), x_{i+1}, \ldots, x_n).
\]

(Local dynamics)
Graph Dynamical Systems – GDS:

- A Graph Dynamical System is a triple consisting of:
  
  - A graph $Y$ with vertex set $\nu(Y) = \{1, 2, \ldots, n\}$.
  - For each vertex $i$ a state $x_i \in K$ (e.g. $K = \{0, 1\}$) and a $Y$-local function $F_i : K^n \rightarrow K^n$
    
    $$F_i(x = (x_1, x_2, \ldots, x_n)) = (x_1, \ldots, x_{i-1}, f_i(x[i]), x_{i+1}, \ldots, x_n).$$

  - An update scheme that governs how the maps $F_i$ are assembled to a map $F : K^n \rightarrow K^n$.

(Local dynamics)
Graph Dynamical Systems – GDS:

A Graph Dynamical System is a triple consisting of:

- A graph $Y$ with vertex set $v[Y] = \{1, 2, \ldots, n\}$.
- For each vertex $i$ a state $x_i \in K$ (e.g. $K = \{0, 1\}$) and a $Y$-local function $F_i : K^n \rightarrow K^n$

$$F_i(x = (x_1, x_2, \ldots, x_n)) = (x_1, \ldots, x_{i-1}, f_i(x[i]), x_{i+1}, \ldots, x_n).$$

- An update scheme that governs how the maps $F_i$ are assembled to a map $F : K^n \rightarrow K^n$.

Typical choices of update schemes:

- **Parallel**: Generalized Cellular Automata

  $$F(x_1, \ldots, x_n)_i = f_i(x[i])$$

- **Sequential**: Sequential Dynamical Systems

  $$[F_Y, w] = F_{w(k)} \circ F_{w(k-1)} \circ \cdots \circ F_{w(1)}$$

  ($w = w(1) \cdots w(k)$ – a word on $v[Y]$)
An Example:

- **Graph** $Y = \text{Circ}_4$
- State set $K = \{0, 1\}$
- System state $x = (x_1, x_2, x_3, x_4)$
- Restricted vertex state $x[1] = (x_1, x_2, x_4)$
- **Vertex functions** $f_i = \text{nor}_3 : K^3 \rightarrow K$ by $\text{nor}_3(x, y, z) = (1 + x)(1 + y)(1 + z)$
- **$Y$-local maps** $\text{Nor}_1(x) = (\text{nor}_3(x[1]), x_2, x_3, x_4)$, etc.
- **Update sequence** $\pi = (1, 2, 3, 4)$
- **SDS map** $[\text{Nor}_Y, \pi] = \text{Nor}_4 \circ \text{Nor}_3 \circ \text{Nor}_2 \circ \text{Nor}_1$
An Example:

- **Graph** \( Y = \text{Circ}_4 \)
- **State set** \( K = \{0, 1\} \)
- **System state** \( x = (x_1, x_2, x_3, x_4) \)
- **Restricted vertex state** \( x[1] = (x_1, x_2, x_4) \)
- **Vertex functions**: \( f_i = \text{nor}_3 : K^3 \rightarrow K \) by \( \text{nor}_3(x, y, z) = (1 + x)(1 + y)(1 + z) \)
- **Y-local maps**: \( \text{Nor}_1(x) = (\text{nor}_3(x[1]), x_2, x_3, x_4) \), etc.
- **Update sequence** \( \pi = (1, 2, 3, 4) \)
- **SDS map**:
  \[
  [\text{Nor}_Y, \pi] = \text{Nor}_4 \circ \text{Nor}_3 \circ \text{Nor}_2 \circ \text{Nor}_1
  \]

**Sequential**:
\[
[\text{Nor}_Y, \pi](0, 0, 0, 0) = (1, 0, 1, 0)
\]

**Parallel**:
\[
\text{Nor}(0, 0, 0, 0) = (1, 1, 1, 1)
\]
An Example:

- **Graph** \( Y = \text{Circ}_4 \)
- **State set** \( K = \{0, 1\} \)
- **System state** \( x = (x_1, x_2, x_3, x_4) \)
- **Restricted vertex state** \( x[1] = (x_1, x_2, x_4) \)
- **Vertex functions**: \( f_i = \text{nor}_3 : K^3 \rightarrow K \) by
  \[ \text{nor}_3(x, y, z) = (1 + x)(1 + y)(1 + z) \]
- **Y-local maps**: \( \text{Nor}_1(x) = (\text{nor}_3(x[1]), x_2, x_3, x_4) \), etc.
- **Update sequence** \( \pi = (1, 2, 3, 4) \)
- **SDS map**:
  \[ [\text{Nor}_Y, \pi] = \text{Nor}_4 \circ \text{Nor}_3 \circ \text{Nor}_2 \circ \text{Nor}_1 \]

**Sequential**: \([\text{Nor}_Y, \pi](0, 0, 0, 0) = (1, 0, 1, 0)\)

**Parallel**: \(\text{Nor}(0, 0, 0, 0) = (1, 1, 1, 1)\)
Applications Areas – Complex Systems

- Examples of graph dynamical system applications:

  - Dynamical processes on networks: disease propagation on social contact graph, packet flow in cell phone communication, urban traffic and transportation.
  - Computational algorithms: Gauss-Seidel, gene annotation based on functional linkage networks, transport computations on irregular grids.
  - Computational paradigms: Distributed computing.
Applications Areas – Complex Systems

- Examples of graph dynamical system applications:
  
  - Dynamical processes on networks: disease propagation on social contact graph, packet flow in cell phone communication, urban traffic and transportation.
  
  - Computational algorithms: Gauss-Seidel, gene annotation based on functional linkage networks, transport computations on irregular grids.
  
  - Computational paradigms: Distributed computing.
Applications Areas – Complex Systems

▶ Examples of graph dynamical system applications:

- Dynamical processes on networks: disease propagation on social contact graph, packet flow in cell phone communication, urban traffic and transportation.
- Computational algorithms: Gauss-Seidel, gene annotation based on functional linkage networks, transport computations on irregular grids.
- Computational paradigms: Distributed computing.

▶ GDS map well to hardware – more later.

▶ GDS useful for formal/mathematical system specification - avoids the “(c++/java/etc)-implementation-defines-behavior” mess. Analysis. UML, StateCharts.
Comments Regarding GDS and Their Analysis:

- Graph Dynamical Systems and Finite Dynamical Systems (FDS): Discrete time & finite state.
- Some consequences of this:
  - Classical dynamical systems theory and tools do rarely apply (e.g., no more derivatives, Lyapunov functions, etc.)
  - FDS theory (so far) is largely rooted in discrete mathematics, algebra, combinatorics, graph theory and probability theory.
- A Goal: Connect to existing mathematical theory and tools, e.g. theory of Coxeter groups and computational algebra.
GDS Research Theme:

- GDS research theme: Derive phase space properties based on graph, vertex functions and update sequence (as opposed to exhaustive computation). Compare “scale-free” graphs etc.

- Similar work by Golubitsky and Stewart 6-7 years ago for systems of ODEs. Studied relations between “ODE graph structure” and properties of solutions and invariant subspaces.
GDS Research Theme:

- GDS research theme: Derive phase space properties based on graph, vertex functions and update sequence (as opposed to exhaustive computation). Compare “scale-free” graphs etc.

- Similar work by Golubitsky and Stewart 6-7 years ago for systems of ODEs. Studied relations between “ODE graph structure” and properties of solutions and invariant subspaces.

- Example: SDS-map \([\text{Nor}_{\text{Circ}_4}, \pi]\) and dependency on update sequence:
Example: Phase Space Equivalences of SDS:

- Compare SDS \((Y, F_Y, \pi)\) and \((Y, F_Y, \sigma)\) for \(\pi, \sigma \in S_Y\) and their maps for a fixed graph \(Y\) and fixed \(Y\)-local functions \(F_Y\).

- **A: Functional Equivalence.** The number of distinct SDS maps.
- **B: Dynamical Equivalence** The number of SDS maps up to topological conjugation/phase space isomorphism.
- **C: Cycle Equivalence.** The number of cycle equivalence classes of SDS maps.
Cycle Equivalence of SDS Maps:

**Theorem**

For any $\pi \in S_Y$, the SDS maps $[F_Y, \pi]$ and $[F_Y, \sigma_s(\pi)]$ are cycle equivalent.  

($\sigma_s$ – cyclic shift of length $s$ – Macauley/Mortveit arXiv:0709.0291)
Cycle Equivalence of SDS Maps:

Theorem

\[ \text{For any } \pi \in S_Y, \text{ the SDS maps } [F_Y, \pi] \text{ and } [F_Y, \sigma_s(\pi)] \text{ are cycle equivalent.} \]

(\( \sigma_s \) – cyclic shift of length s – Macauley/Mortveit arXiv:0709.0291)

Proposition

\[ \text{If } |K| = 2 \text{ then } [F_Y, \pi] \text{ and } [F_Y, \rho(\pi)] \text{ are cycle equivalent.} \]

(\( \rho \) - reflection – Macauley/Mortveit arXiv:0709.0291)
Cycle Equivalence of SDS Maps:

**Theorem**

For any \( \pi \in S_Y \), the SDS maps \([F_Y, \pi]\) and \([F_Y, \sigma_s(\pi)]\) are cycle equivalent.

(\(\sigma_s\) – cyclic shift of length \(s\) – Macauley/Mortveit arXiv:0709.0291)

**Proposition**

If \(|K| = 2\) then \([F_Y, \pi]\) and \([F_Y, \rho(\pi)]\) are cycle equivalent.

(\(\rho\) - reflection – Macauley/Mortveit arXiv:0709.0291)

- More generally: \([F_Y, \pi]\) and \([F_Y, \sigma_s(\pi)]\) are cycle equivalent if the induced acyclic orientations \(O_{\pi_Y}^{\pi}\) and \(O_{\pi_Y}^{\pi}\) are click-equivalent. (An example of how to relate graph structure, update sequences and dynamics.)
A Bound for the Number of Cycle Classes – $\kappa(Y)$:

- The number of cycle equivalence classes for permutation SDS is bounded above by the number of click-equivalence classes of acyclic orientations of the SDS graph $Y$. Denoted $\kappa(Y)$. (Base case: $\kappa(\text{Line}_2) = 1$.)

**Theorem**

$$
\kappa(Y) = \begin{cases} 
\kappa(Y_1)\kappa(Y_2), & e \text{ is a bridge linking } Y_1 \text{ and } Y_2, \\
\kappa(Y'_e) + \kappa(Y''_e), & e \text{ is a cycle-edge}.
\end{cases}
$$

A Bound for the Number of Cycle Classes – $\kappa(Y)$:

- The number of cycle equivalence classes for permutation SDS is bounded above by the number of click-equivalence classes of acyclic orientations of the SDS graph $Y$. Denoted $\kappa(Y)$. (Base case: $\kappa(\text{Line}_2) = 1.$)

**Theorem**

\[
\kappa(Y) = \begin{cases} 
\kappa(Y_1)\kappa(Y_2), & \text{if } e \text{ is a bridge linking } Y_1 \text{ and } Y_2, \\
\kappa(Y'_e) + \kappa(Y''_e), & \text{if } e \text{ is a cycle-edge}.
\end{cases}
\]


**Remarks:**

- $\kappa(Y) = \beta(Y) –$ the Möbius invariant of the intersection lattice of $Y$: $\beta(Y) = |\mu_{LY}(\hat{0}, \hat{1})|$.
- $\kappa(Y)$ is a Tutte-invariant with $x = 1$ and $y = 0$, and thus $\kappa(Y) = T_Y(1, 0)$. 
Example (Enumeration and Graph Automorphisms)

Let $Y = Q_2^3$, the binary 3-cube:

$$\kappa(Q_2^3) = \kappa(Q_2^2) + \kappa(Q_2^2) = \kappa(Q_2^2) + 2\kappa(Q_2^2) + \kappa(Q_2^2)$$

$$= \kappa(Q_2^2) + 2\kappa(Q_2^2) + 2\kappa(Q_2^2) + \kappa(Q_2^2) + \kappa(Q_2^2)$$

$$= \kappa(Q_2^2) + 4\kappa(Q_2^2) + 2\kappa(Q_2^2) + \kappa(Q_2^2) + \kappa(Q_2^2)$$

$$= 27 + 64 + 16 + 12 + 14 = 133$$

► If the vertex functions are all the same (a homogeneous SDS) then $\bar{\kappa}(Y) = 8$ is a sharp upper bound for the number of cycle classes of permutation SDS. Compare to $8! = 40320$ possible update sequences.
Terminology:

- Let $W(S)$ be a Coxeter system where $W$ is the Coxeter group, $S = \{s_1, s_2, \ldots, s_n\}$ is the set of generators, relations $(s_is_j)^{m_{ij}}$, Coxeter matrix $M = [m_{ij}]$, and Coxeter graph $\Gamma$.
- The set of Coxeter elements is $C(W, S) = \{\prod s_{\pi(i)} \mid \pi \in S_n\}$.
- The SDS graph $Y$ corresponds to the (unlabeled) Coxeter graph $\Gamma$.

Let $\pi \in S_Y$.

\[
[F_Y, \pi] = F_{\pi(n)} \circ F_{\pi(n-1)} \circ \cdots \circ F_{\pi(1)}
\]

Permutation SDS map

\[
\prod_{k=1}^{n} s_{\pi(k)} = s_{\pi(n)} s_{\pi(n-1)} \cdots s_{\pi(1)}
\]

Coxeter element

- For example: cycle equivalent SDS correspond to conjugate Coxeter elements.
Modeling and Applications
Gene Annotation Using Functional Linkage Networks:

Goal: Determine if a collection of genes exhibit a given biological functions. (Based on [3]).

Modeling approach:

- Represent genes as vertices and connect genes if there is sufficient experimental/computational evidence indicating that they share biological functions.
- Each edge has a weight $w \in [-1, 1]$ encoding the degree of co-expression.
- A new biological function is given, and it is known that some subset of the genes expresses this function.
- Assign a state $+1$ to genes that express the new function and $-1$ to genes that do not express function $f$. Assign state 0 to states of remaining vertices.
Gene Annotation Using Functional Linkage Networks:

- **Goal:** Determine if a collection of genes exhibit a given biological functions. (Based on [3]).

**Modeling approach:**

- Represent genes as vertices and connect genes if there is sufficient experimental/computational evidence indicating that they share biological functions.
- Each edge has a weight \( w \in [-1, 1] \) encoding the degree of co-expression.
- A new biological function is given, and it is known that some subset of the genes expresses this function.
- Assign a state +1 to genes that express the new function and −1 to genes that do not express function \( f \). Assign state 0 to states of remaining vertices.

**Goal:** Given the functional linkage network (FLN) and the initial configuration \( x = (x_v)_{v \in V[Y]} \) assign states ±1 to all the genes with state 0 so as to minimize the energy function

\[
E(x, Y) = - \sum_{\{u,v\} \in E[Y]} w_{\{u,v\}} x_u x_v .
\]

**Observation:** Locally, neighbor genes with identical states should be connected by a positive edge and neighbor genes with opposite states should be connected by a negative edge.
Gene Annotation Using Functional Linkage Networks - (II):

The algorithm in [3] computes an approximation to such a minimal state through an iterative sequential scheme.

1. Select (at random) a permutation $\pi$ of the set $V$ of vertices initially in state 0.
2. Repeat: Update the state of each of vertex $v \in V$ asynchronously using the update sequence $\pi$ and the vertex function

$$\text{sign} \sum_{\{u,v\}} w_{\{v,u\}} x_u .$$

The algorithm stops when successive configurations are identical.

▶ **Observation:** This algorithm corresponds to a fixed point computation of a threshold SDS, $[\text{FLN}_Y, \pi]$ say.

▶ **Fact:** The SDS $[\text{FLN}_Y, \pi]$ has only fixed points as attractors (i.e. no periodic orbits of period $\geq 2$.) Proof idea: Every state transition lowers the energy $E$ (previous slide).
Gene Annotation – A Cautionary Tale:

- Sensitivity of $\omega$-limit set w.r.t. update sequence and/or initial condition?

  - Have constructed threshold-SDS with the following properties: They have $2^{n+1}$ states, $2^n$ fixed points (predictions), with an initial condition that can reach $2^n - n$ of these fixed points depending on choice of update sequence (with A. Kumar and M. Macauley). **Exponential instability!** The same systems also display strong sensitivity with respect to initial conditions.
Gene Annotation – A Cautionary Tale:

- Sensitivity of $\omega$-limit set w.r.t. update sequence and/or initial condition?

- Have constructed threshold-SDS with the following properties: They have $2^{n+1}$ states, $2^n$ fixed points (predictions), with an initial condition that can reach $2^n - n$ of these fixed points depending on choice of update sequence (with A. Kumar and M. Macauley). **Exponential instability**! The same systems also display strong sensitivity with respect to initial conditions.

- The results carry over to graphs that are trees and also to the random graph $G_{n,p}$ (with high probability) for certain ranges of $p$.

- This instability may not be present in the FLN-algorithm (due to e.g. graph structure, nature of edge weights etc.).

- Through the SDS identification this issue became apparent right away.

- **Bottom line:** Need better understanding of the interplay between structure and dynamics.
Modeling and Analysis of Epidemics and Interventions for Urban Areas:

- Ordinary differential equation (ODE) based \textit{SIR}-type models are often used to study disease evolution for groups/aggregates of a large, “well-mixed” population.

- \textbf{Problems:}
  - The well-mixing assumption is often questionable.
  - Interventions are often targeted at individuals. Hard to study one or more such interventions in an aggregated model.

- Diseases such as influenza spread from person-to-person based on contact. It is natural to consider individual-based models defined over a graph.

- Will present the EpiSims model in the following.
But First: What Should a Model Tell Us?

▶ Some questions studied in NDSSL using EpiSims:

- Given limited resources (time, medication), how should vaccines be administered?
- Who should be isolated/quarantined? (High degree vertices is not necessarily the answer.)
- What are the effects of limited compliance?
- Will closing elementary schools limit the outbreak?
- What observables can be used for efficient and accurate early detection?
- How can one protect critical sub-populations such as police, hospital personnel and military forces?

▶ Note: All the questions above translate into questions about the social contact graph: What are effective (and realistic) intervention strategies for *shattering* the contact graph?
EpiSims – Scope and Scale:

- Large urban areas (E.g., Chicago has about 9 million people - NDSSL is currently targeting the US population.) Note about scaling & model complexity.

- A collection of individuals forming the population. (Attributes include: age, health status, gender, household composition)

- A set of activities for all individuals (consistent across, e.g. household).

- A disease. Examples Influenza, small-pox, pandemic flu.

- **The Paradigm:** People perform activities which brings them to locations for certain time periods. The co-occupancy of locations can be used to construct the social contact graph (at a specific time). This graph and, e.g. the duration of contact governs disease dynamics (at that point in time).
EpiSims – Construction of Population:

- EpiSims uses a *synthetic population*.

  - Impossible to get the exact description of a population for a given urban area. Confidentiality issues.
  - Approach: Construct a synthetic population based on extensive survey data and data bases (commercial and public).
  - Match and calibrate/validate against real population on disease-relevant attributes such as demographics, age-distribution, income-distribution, household composition, commute distance, work and school locations.
  - Synthetic population is *statistically indistinguishable* from the real population data as measured by the survey.
EpiSims – Construction of Activities and Locations:

How do people move around? By performing their *activities*.

- Individuals of the population are assigned activities for e.g. a 24 hour period.
- Activities statistically match activity surveys from the real population.
- Activities are generated to be consistent across e.g. households.
- Example: 8AM – Work, 11:30AM – Doctor visit, 5:00PM – Shopping, 6PM – Home.
- Locations are generated based on public and commercial databases, and activities are mapped to locations.
EpiSims – Construction of Activities and Locations:

How do people move around? By performing their activities.

- Individuals of the population are assigned activities for e.g. a 24 hour period.
- Activities statistically match activity surveys from the real population.
- Activities are generated to be consistent across e.g. households.
- Example: 8AM – Work, 11:30AM – Doctor visit, 5:00PM – Shopping, 6PM – Home.
- Locations are generated based on public and commercial databases, and activities are mapped to locations.
EpiSims – Population Mobility – TRANSIMS:

- Population dynamics created by, e.g. TRANSIMS (a component in NDSSL’s Simfrasstructure framework) based on the population and the set of activities.

  - Dynamics is infrastructure constrained and traffic rules are followed.
  - Handles all normal types of transportation (car, bus, train, walk, etc.)
  - Track people on a per second basis.
EpiSims – Population Mobility – TRANSIMS:

- Population dynamics created by, e.g. TRANSIMS (a component in NDSSL's Simfrrastructure framework) based on the population and the set of activities.
  - Dynamics is infrastructure constrained and traffic rules are followed.
  - Handles all normal types of transportation (car, bus, train, walk, etc.)
  - Track people on a per second basis.

- TRANSIMS consists of two main components:
  - Router: Creates travel routes based on activities. A regular-expression constrained version of Dijkstra’s shortest path algorithm.
  - Micro-simulator: Maps travelers through transportation infrastructure on a per second basis using routes. (An example of a composed probabilistic synchronous graph dynamical system.)

- Calibration and validation: TRANSIMS has been calibrated against real traffic data from several cities.

- Other models: Random way-point/random walk. Common in, e.g. wireless communication, but lack realism.
EpiSims – Population Mobility – TRANSIMS:

- Population dynamics created by, e.g. TRANSIMS (a component in NDSSL’s Simfrasstructure framework) based on the population and the set of activities.

  - Dynamics is infrastructure constrained and traffic rules are followed.
  - Handles all normal types of transportation (car, bus, train, walk, etc.)
  - Track people on a per second basis.

- TRANSIMS consists of two main components:

  - **Router**: Creates travel routes based on activities. A regular-expression constrained version of Dijkstra’s shortest path algorithm.
  - **Micro-simulator**: Maps travelers through transportation infrastructure on a per second basis using routes. (An example of a composed probabilistic synchronous graph dynamical system.)

- Calibration and validation: TRANSIMS has been calibrated against real traffic data from several cities.

- Other models: Random way-point/random walk. Common in, e.g. wireless communication, but lack realism.
EpiSims – An Example Route:
TRANSIMS—Network Representation & Dynamics:

- TRANSIMS uses a discretization of the real road network and transportation infrastructure.
  
  - Each lane of all roads (or links) are discretized into 7.5m long cells.
  - A cell can hold zero or one vehicles.
  - Discrete vehicle speeds \{0, 1, 2, 3, 4, 5\} (cells/sec)

This is a cell-based approach. An alternative is a vehicle-based approach, but that is trickier to compute/implement. (Why?)
TRANSIMS– Network representation & Dynamics:

- Intersections require for example:
  - Permissible link continuations
  - Traffic signaling
  - Traffic signal synchronization

Structure of a fairly simple intersection.
TRANSIMS– Driving Rules:

- **Lane change decision** ($\Phi_1$): If a lane change is desirable for a car in cell $i$ (front-gap $\delta(i)$ of cell $i$ is less then $v_{\text{max}}$ and less than neighbor front-gap) and permissible (neighbor back-gap at least $v_{\text{max}}$) then set the lane-change flag.
- **Lane Change** ($\Phi_2$): Execute lane change for each cell.
- **Velocity update** ($\Phi_3$) for each car by the sequence
  - $v \mapsto \min(v + 1, \delta(i), v_{\text{max}})$ (acceleration)
  - With probability $p_{\text{break}}$ let $v \mapsto \max(v - 1, 0)$ (random slow-down).
- **Position update** ($\Phi_4$): Move each car $v$ cells forward.

Constructed from four cellular automata (CA) / SDS:
TRANSIMS—Driving Rules:

constructed from four cellular automata (CA) / SDS:

- **Lane change decision** ($\Phi_1$): If a lane change is desirable for a car in cell $i$ (front-gap $\delta(i)$ of cell $i$ is less than $v_{max}$ and less than neighbor front-gap) and permissible (neighbor back-gap at least $v_{max}$) then set the lane-change flag.

- **Lane Change** ($\Phi_2$): Execute lane change for each cell.

- **Velocity update** ($\Phi_3$) for each car by the sequence
  - $v \mapsto \min(v + 1, \delta(i), v_{max})$ (acceleration)
  - With probability $p_{break}$ let $v \mapsto \max(v - 1, 0)$ (random slow-down).

- **Position update** ($\Phi_4$): Move each car $v$ cells forward.

The micro-simulator is a composed probabilistic GDS with map: $\Phi_4 \circ \Phi_3 \circ \Phi_2 \circ \Phi_1$
Transims short movie - click to play (Note: requires quick time plug-in)
TRANSIMS Micro-Simulator – Comments & Summary:

- Validation: TRANSIMS has been validated against real traffic data and also against special road configurations.
- Data preparation: Creating synthetic population, activities and preparing a network is a substantial part of the process.
- The TRANSIMS micro-simulator is a stochastic finite dynamical system (slow-down & lane-change.) Why: It happens in normal traffic, and it seems essential to produce realistic traffic.
EpiSims Continued ... Social Network/People-Location Graph:

- The information from TRANSIMS allows us to construct the *Social Contact Network*. 
EpiSims Continued ... Social Network/People-Location Graph:

- The information from TRANSIMS allows us to construct the Social Contact Network.
- Gives information about contacts, duration of contact and time at location.

![Diagram of Social Network/People-Location Graph]
EpiSims – Disease Dynamics – Basic Ideas:

- Disease progression/Vertex functions:

  - An individual with state $S$ is mapped to state $I$ with probability

    $$p = 1 - (1 - p)^k;$$

    $k$ – the number of infectious neighbors.

  - An individual in state $I$ is mapped to $R$.
  - All other states remain fixed.

- Evolution: The system evolves from some initial state $x(t = 0)$ using a synchronous update scheme, $x(1), x(2)$, etc.
EpiSims – Details:

- There are two basic processes in the disease evolution:
  - Step 1: A susceptible individual becomes infected. (external)
  - Step 2: The individual disease evolution. (internal)

Step 1 governs the transition $S \rightarrow I$: Bernoulli trials with probability

$$p_{i,s,a} = \left\{ 1 - \exp \left[ t \sum_{j \in T_1} \sum_{r \in R(j)} N_{j,r} \ln(1 - rs \rho_a(i,j)) \right] \right\} / M^\alpha .$$

Step 2, which governs the internal disease evolution, is described using a probabilistic finite state machine. Specified as disease manifestations – see next slide.
EpiSims – Disease Manifestation Examples:
EpiSims – Summary:

- EpiSims is a probabilistic graph dynamical system. Each simulation/run is a realization of a stochastic process. An experiment or case study samples this probability space.

- The implementation is configurable in terms of computing resources, but is typically run on cluster machines.

- METIS type algorithms are used in partitioning.

- TRANSIMS can be replaced by other mobility generators.

- EpiSims is a component in NDSSL’s Siminfrastructure framework. This is a framework that allows one to study co-evolving socio-technical systems (epidemics, electrical power networks, markets, tele-communication, transportation)

- Again – Research goal: Improve understanding of relations between structure (the contact graph) and dynamics.
GDS – Modeling & Specialized Architectures:

- GDS and (finite dynamical systems) are well-suited for specialized hardware architectures.

**Example:** A specialized version of the TRANSIMS micro-simulator was implemented as a hybrid CPU/FPGA computation using a streaming approach (PCI-X). Serves as prototype for the use of FPGAs for large-scale simulations.

**Visualization:** Be patient! Startup takes some time on a laptop!
GDS – Modeling & Specialized Architectures:

- GDS and (finite dynamical systems) are well-suited for specialized hardware architectures.

**Example:** A specialized version of the TRANSIMS micro-simulator was implemented as a hybrid CPU/FPGA computation using a streaming approach (PCI-X). Serves as prototype for the use of FPGAs for large-scale simulations.

- **FPGA:** Field Programmable Gate Array – a fairly new class of computing architecture.
- Characteristics of applications that map well to FPGA: High degree of parallelism, localized interaction, integer arithmetic, limited use of floating point arithmetic.

- **Visualization:** Be patient! Startup takes some time on a laptop!
GDS – Modeling & Specialized Architectures:

- GDS and (finite dynamical systems) are well-suited for specialized hardware architectures.

**Example:** A specialized version of the TRANSIMS micro-simulator was implemented as a hybrid CPU/FPGA computation using a streaming approach (PCI-X). Serves as prototype for the use of FPGAs for large-scale simulations.

- **FPGA:** Field Programmable Gate Array – a fairly new class of computing architecture.
- Characteristics of applications that map well to FPGA: High degree of parallelism, localized interaction, integer arithmetic, limited use of floating point arithmetic.
- **Problem size:** Portland has more than 125000 road links or 6.25 million road cells. Chicago has about 20 million road cells. (This shows the computational power of modeling directly using CA/SDS.)
- **Result:** Hybrid implementation initially about 20% faster than software reference model (on a Cray Octigabay). Not quite fast enough! Additional speedup was obtained later.

- **Visualization:** Be patient! Startup takes some time on a laptop!
GDS – Modeling & Specialized Architectures:

Examples:
- GDS and (finite dynamical systems) are well-suited for specialized hardware architectures.
- A specialized version of the TRANSIMS micro-simulator was implemented as a hybrid CPU/FPGA computation using a streaming approach (PCI-X). Serves as prototype for the use of FPGAs for large-scale simulations.
- An HIV model for mutations.

GDS architecture:
- Defined as a class of computing architecture that is well-suited for specialized hardware implementations.
- Field Programmable Gate Array (FPGA) is a class of computing architecture.

Characteristics of applications that map well to FPGAs:
- High degree of parallelism.
- Localized interaction.
- Integer arithmetic, limited use of floating point arithmetic.

Problem size:
- Portland has more than 125,000 road links or 6.25 million road cells. (This shows the computational power of modeling directly using CA/SDS.)
- Chicago has about 20 million road cells. (This shows the computational power of modeling directly using CA/SDS.)

Result:
- Hybrid implementation initially about 20% faster than software reference model (on a Cray Octogon). Not quite fast enough! Additional speedup was obtained later.

Lessons learnt:
- Both design and implementations of large-scale simulations like TRANSIMS need to be done with specialized hardware in mind. The "optimize-the-inner-loop" paradigm/afterthought will not cut it.
- Characteristics of applications that map well to FPGAs: High degree of parallelism, localized interaction, integer arithmetic, limited use of floating point arithmetic.

Visualization:
- Be patient! Startup takes some time on a laptop!
New Project: A Spatial Network Model for HIV:

► **Goal:** To create a network model for free virus and immune cells relevant in HIV infection throughout the entire body.

► **Later Goal:** Understand mechanisms of HIV mutations.

► **Entities:** Naïve CD4+ T-cells, memory T-cells, T1 helper-cells, T2 helper-cells, tissue-resident macrophages (RM), infiltrated macrophages (MDM), dendritic cells (DC), follicular dendritic cells (FDC), free virions (V), free antibody (Ab).

► **States:** Not susceptible; only naïve T-cell, Infected/susceptible, Latently infected; only memory T-cells, not susceptible/shedding.


► **Contact:** Kate Wendelsdorf, NDSSL/VBI at Virginia Tech (wkath83 -@- vbi.vt.edu)
Links & References

NDSSL: http://ndssl.vbi.vt.edu

Math homepage: http://www.math.vt.edu/people/hmortvei/research/


