

DISCRETE EVENT AND HYBRID SYSTEM MODELS AND METHODS FOR CYBER-PHYSICAL SYSTEMS

C. G. Cassandras

Division of Systems Engineering

**and Dept. of Electrical and Computer Engineering
and Center for Information and Systems Engineering
Boston University**

CONTROL AND OPTIMIZATION – CHALLENGES

1. **SCALABILITY**

2. **DECENTRALIZATION**



Distributed Algorithms

3. **COMMUNICATION**



**Event-driven (asynchronous)
Algorithms**

4. **NON-CONVEXITY**



**Global optimality,
escape local optima**

5. **EXLOIT DATA**



Data-Driven Algorithms

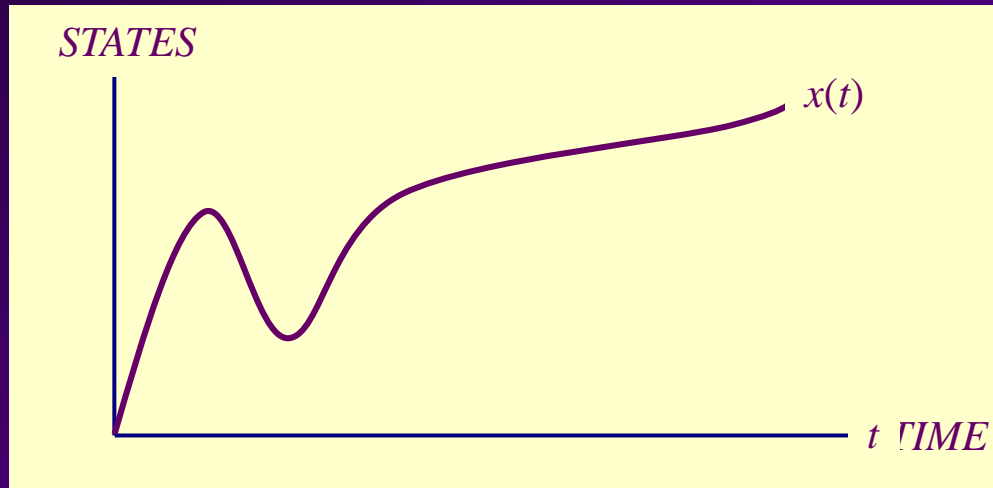
OUTLINE

- **MODELING:**
 - Discrete Event Systems (DES)
 - Hybrid Systems (HS)

- **CONTROL AND OPTIMIZATION:**
 - Event-Driven Distributed Algorithms
 - Data-Driven + Event-driven Algorithms:
The IPA Calculus
 - Global optimality, escaping local optima

TIME-DRIVEN v EVENT-DRIVEN SYSTEMS

TIME-DRIVEN SYSTEM



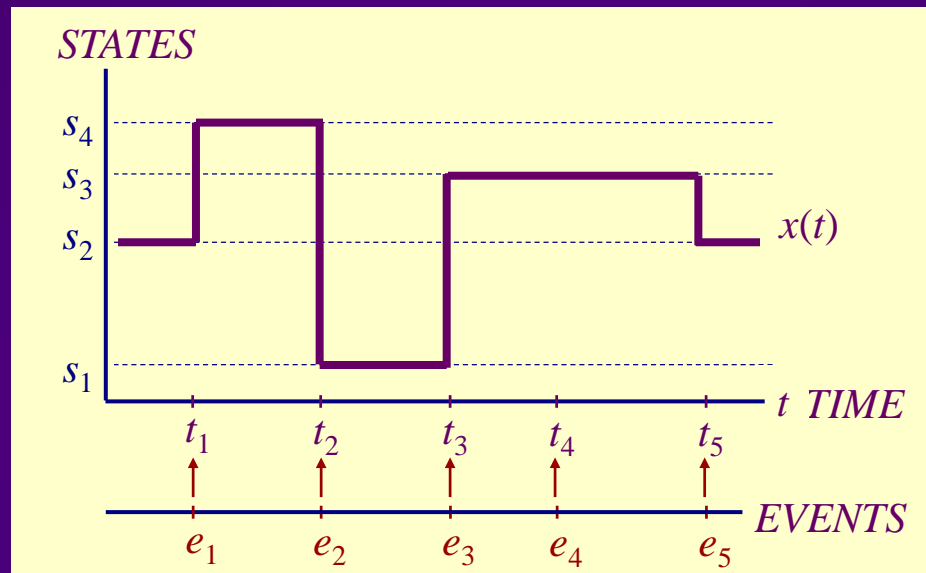
STATE SPACE:

$$X = \mathfrak{R}$$

DYNAMICS:

$$\dot{x} = f(x, t)$$

EVENT-DRIVEN SYSTEM



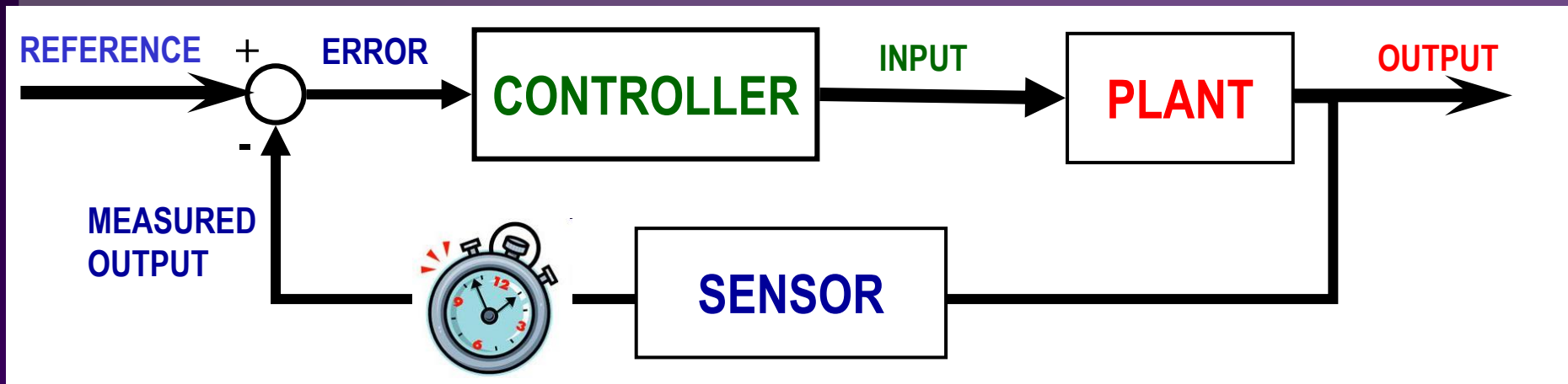
STATE SPACE:

$$X = \{s_1, s_2, s_3, s_4\}$$

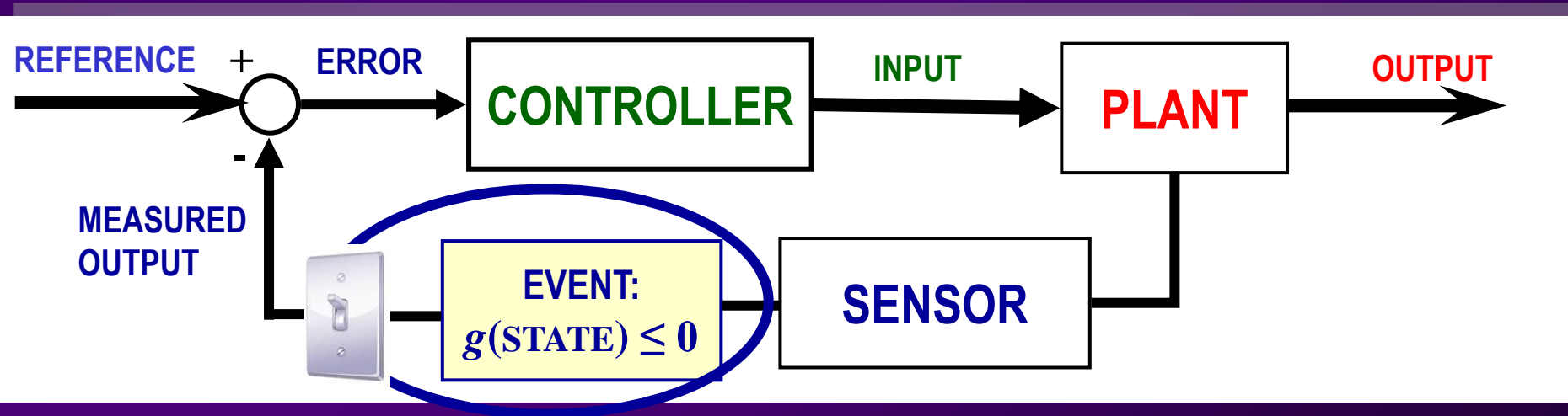
DYNAMICS:

$$x' = f(x, e)$$

TIME-DRIVEN v EVENT-DRIVEN CONTROL



EVENT-DRIVEN CONTROL: Act *only when needed* (or on **TIMEOUT**) - not based on a clock



REASONS FOR *EVENT-DRIVEN* MODELS, CONTROL, OPTIMIZATION

- Many systems are naturally **Discrete Event Systems (DES)** (e.g., Internet)
→ *all* state transitions are event-driven
- Most of the rest are **Hybrid Systems (HS)**
→ *some* state transitions are event-driven
- Many systems are **distributed**
→ components interact asynchronously (through events)
- Time-driven sampling inherently inefficient (“open loop” sampling)

REASONS FOR *EVENT-DRIVEN* MODELS, CONTROL, OPTIMIZATION

- Many systems are **stochastic**
→ actions needed in response to random events
- Event-driven methods provide significant advantages in **computation** and **estimation** quality
- System performance is often **more sensitive to event-driven** components than to time-driven components
- Many systems are **wirelessly networked** → energy constrained
→ time-driven communication consumes significant energy
UNNECESSARILY!

MODELING DES AND HS:

- Timed Automata**
- Hybrid Automata**

AUTOMATON

AUTOMATON: (E, X, Γ, f, x_0)

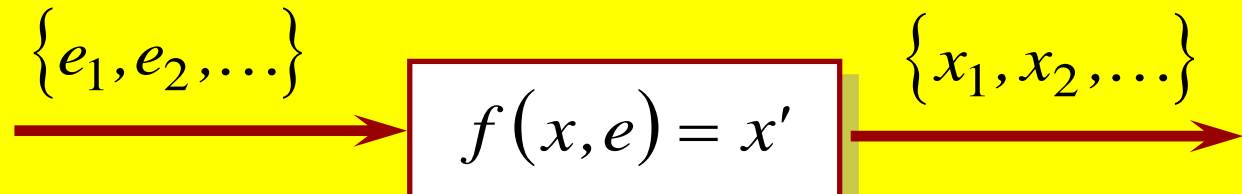
E : Event Set

X : State Space

$\Gamma(x)$: Set of *feasible* or *enabled* events at state x

f : State Transition Function $f: X \times E \rightarrow X$
(undefined for events $e \notin \Gamma(x)$)

x_0 : Initial State, $x_0 \in X$

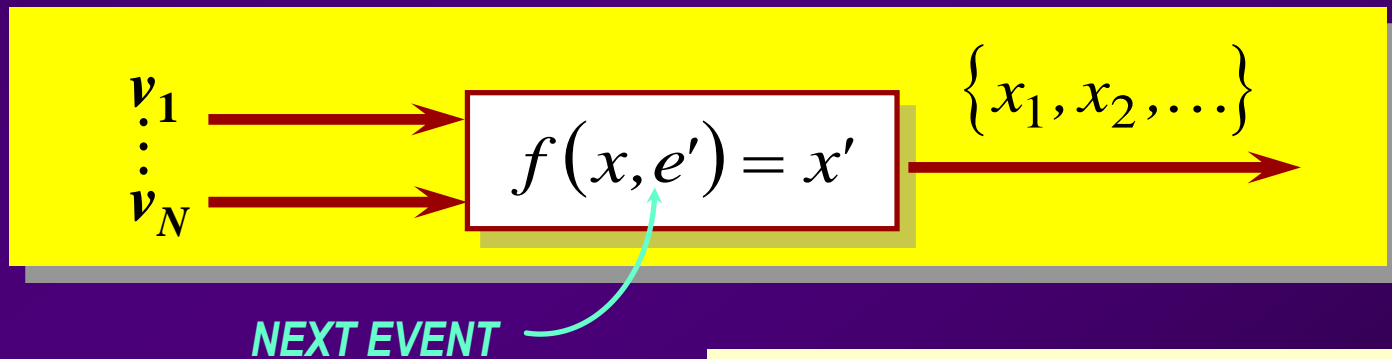


TIMED AUTOMATON

Add a **Clock Structure V** to the automaton: $(E, X, \Gamma, f, x_0, V)$
where:

$$V = \{v_i : i \in E\}$$

and v_i is a **Clock or Lifetime sequence**: $v_i = \{v_{i1}, v_{i2}, \dots\}$
one for each event i



Need an *internal mechanism* to determine
NEXT EVENT e' and hence
NEXT STATE $x' = f(x, e')$

HOW THE TIMED AUTOMATON WORKS...

➤ CURRENT STATE

$x \in X$ with feasible event set $\Gamma(x)$

➤ CURRENT EVENT

e that caused transition into x

➤ CURRENT EVENT TIME

t associated with e



Associate a

***CLOCK VALUE/RESIDUAL LIFETIME* y_i**

with each feasible event $i \in \Gamma(x)$

HOW THE TIMED AUTOMATON WORKS...

- **NEXT/TRIGGERING EVENT e' :**

$$e' = \arg \min_{i \in \Gamma(x)} \{y_i\}$$

- **NEXT EVENT TIME t' :**

$$t' = t + y^*$$

$$\text{where: } y^* = \min_{i \in \Gamma(x)} \{y_i\}$$

- **NEXT STATE x' :**

$$x' = f(x, e')$$

HOW THE TIMED AUTOMATON WORKS...

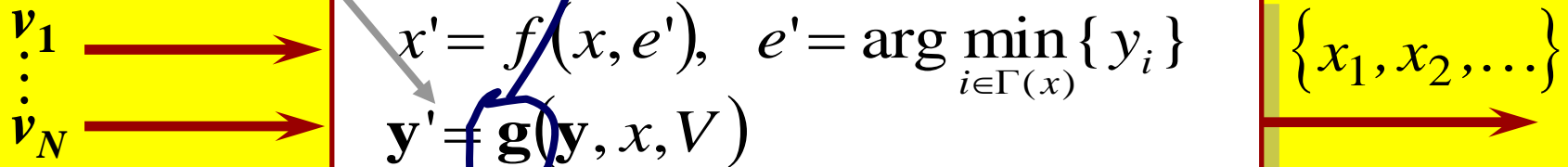


Determine new **CLOCK VALUES** y'_i
for every event $i \in \Gamma(x)$

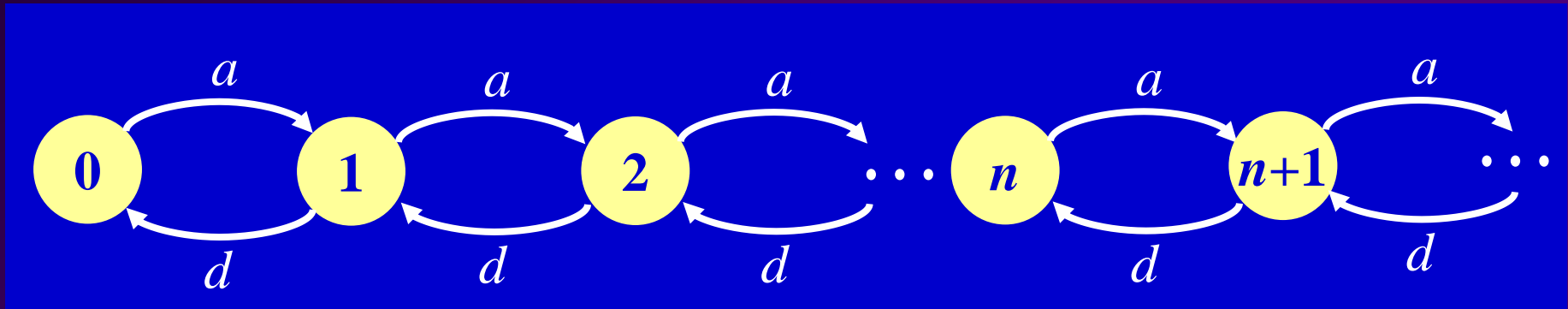
$$y'_i = \begin{cases} y_i - y^* & i \in \Gamma(x'), i \in \Gamma(x), i \neq e' \\ v_{ij} & i \in \Gamma(x') - \{\Gamma(x) - e'\} \\ 0 & \text{otherwise} \end{cases}$$

where : v_{ij} = new lifetime for event i

EVENT CLOCKS
ARE STATE VARIABLES



TIMED AUTOMATON - AN EXAMPLE



$$E = \{a, d\}$$

$$X = \{0, 1, 2, \dots\}$$

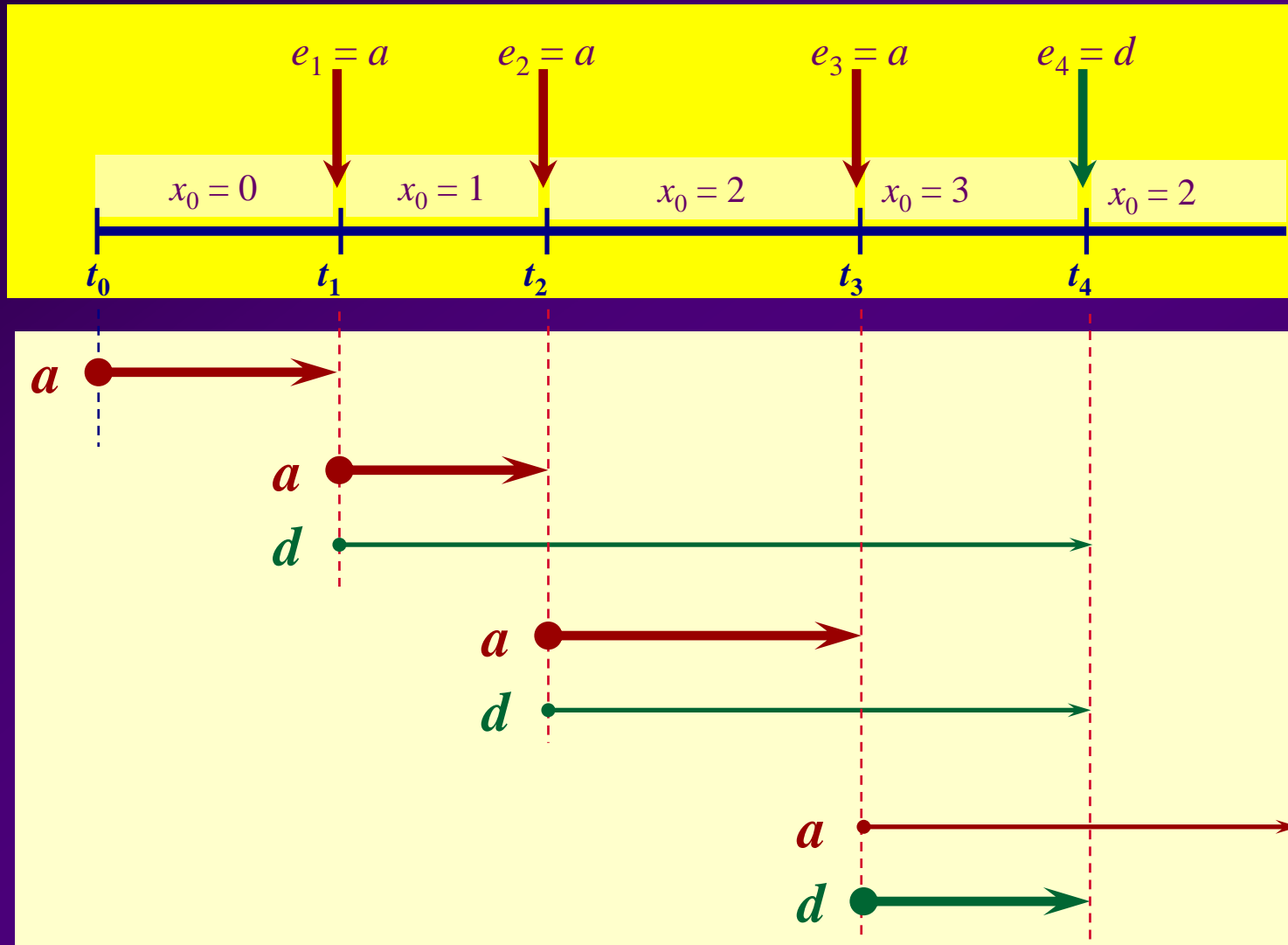
$$\Gamma(x) = \{a, d\}, \text{ for all } x > 0$$

$$\Gamma(0) = \{a\}$$

$$f(x, e') = \begin{cases} x + 1 & e' = a \\ x - 1 & e' = d, x > 0 \end{cases}$$

$$\text{Given input: } \mathbf{v}_a = \{v_{a1}, v_{a2}, \dots\}, \mathbf{v}_d = \{v_{d1}, v_{d2}, \dots\}$$

TIMED AUTOMATON - A STATE TRAJECTORY



STOCHASTIC TIMED AUTOMATON

- Same idea with the Clock Structure consisting of *Stochastic Processes*
- Associate with each event i a *Lifetime Distribution* based on which ν_i is generated



Generalized Semi-Markov Process (GSMP)

In a simulator, ν_i is generated through a pseudorandom number generator

HYBRID AUTOMATA

$$G_h = (Q, X, E, U, f, \phi, Inv, guard, \rho, q_0, \mathbf{x}_0)$$

Q : set of discrete states (modes)

X : set of continuous states (normally \mathbb{R}^n)

E : set of events

U : set of admissible controls

f : vector field, $f : Q \times X \times U \rightarrow X$

ϕ : discrete state transition function, $\phi : Q \times X \times E \rightarrow Q$

Inv : set defining an invariant condition (domain), $Inv \subseteq Q \times X$

$guard$: set defining a guard condition, $guard \subseteq Q \times Q \times X$

ρ : reset function, $\rho : Q \times Q \times X \times E \rightarrow X$

q_0 : initial discrete state

\mathbf{x}_0 : initial continuous state

HYBRID AUTOMATA

Key features:

Transition MAY occur

Guard condition:

Subset of X in which a transition from q to q' is enabled, defined through ϕ

Transition MUST occur

Invariant condition:
(domain)

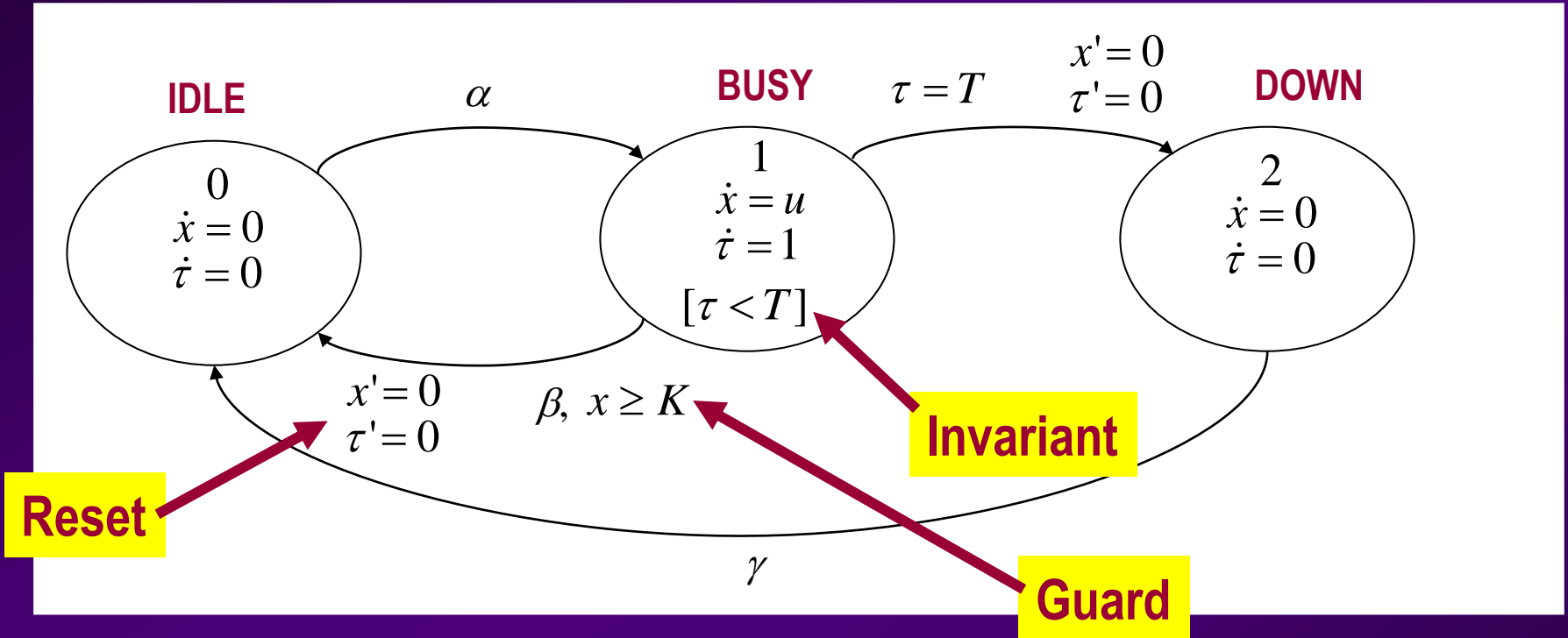
Subset of X to which x must belong in order to remain in q . If this condition no longer holds, a transition to some q' must occur, defined through ϕ

Reset condition:

New value x' at q' when transition occurs from (x, q)

HYBRID AUTOMATA

Unreliable machine with timeouts

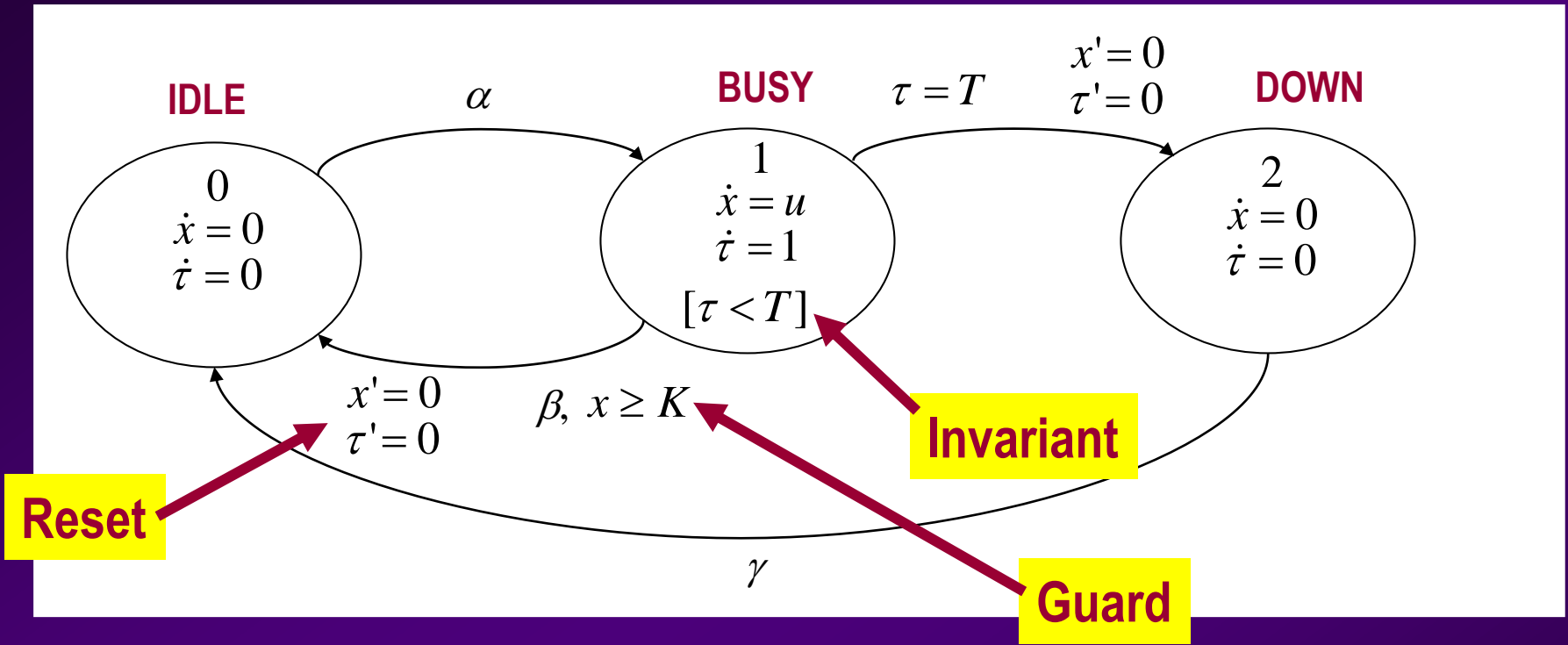


$x(t)$: physical state of part in machine

$\tau(t)$: clock

α : START, β : STOP, γ : REPAIR

HYBRID AUTOMATA

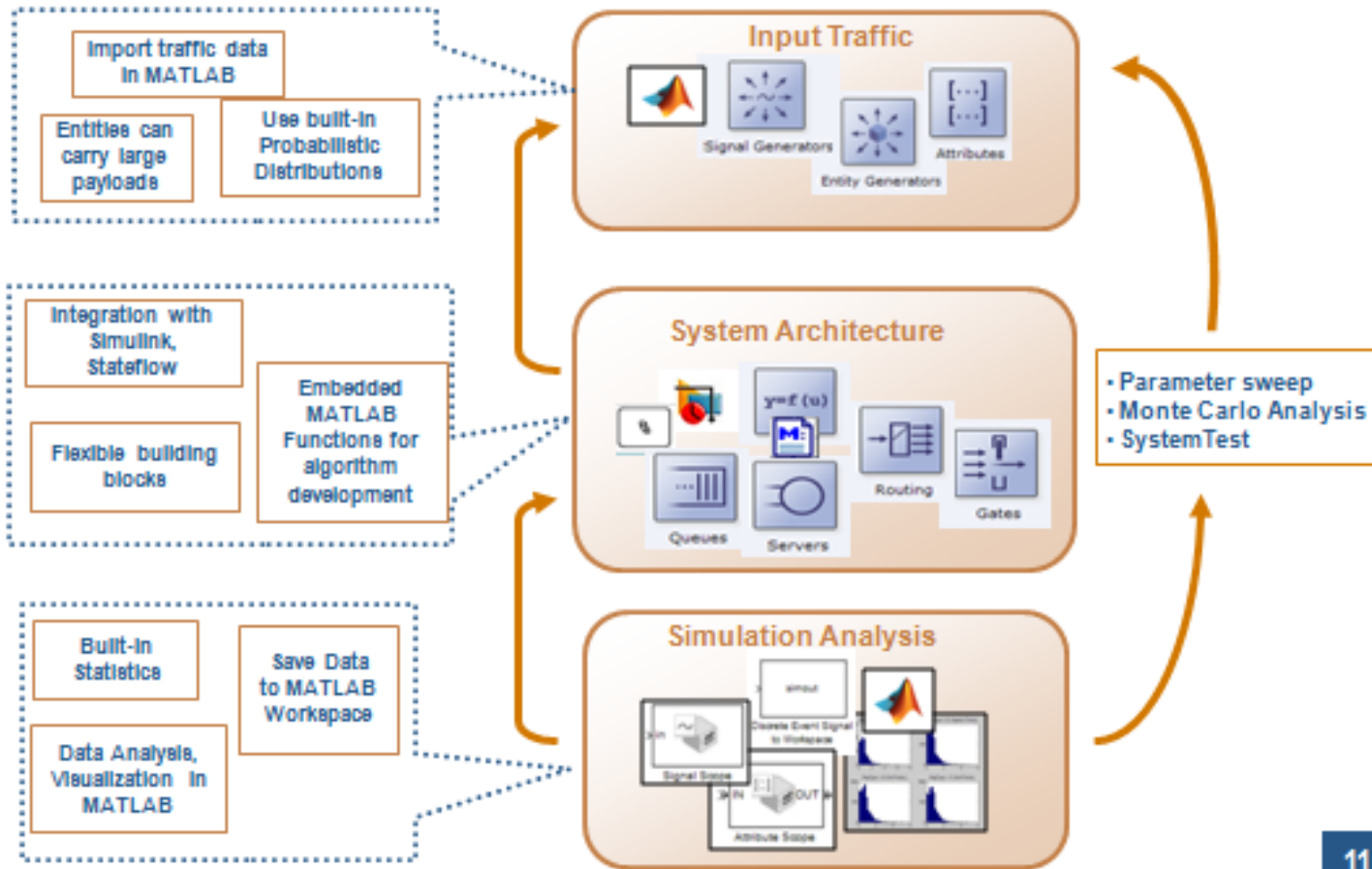


$$\phi(0; x, \tau; e) = \begin{cases} 1 & e = \alpha \\ 0 & \text{otherwise} \end{cases}$$

$$\phi(2; x, \tau; e) = \begin{cases} 0 & e = \gamma \\ 2 & \text{otherwise} \end{cases}$$

$$\phi(1; x, \tau; e) = \begin{cases} 2 & \tau = T \\ 0 & x \geq K, e = \beta \\ 1 & \text{otherwise} \end{cases}$$

SimEvents Key Features



SELECTED REFERENCES - MODELING

Timed Automata and Timed Petri Nets

- Alur, R., and D.L. Dill, “A Theory of Timed Automata,” Theoretical Computer Science, No. 126, pp. 183-235, 1994.
- Cassandras, C.G, and S. Lafortune, “Introduction to Discrete Event Systems,” Springer, 2008.
- Wang, J., Timed Petri Nets - Theory and Application, Kluwer Academic Publishers, Boston, 1998.

Hybrid Systems

- Bemporad, A. and M. Morari, “Control of Systems Integrating Logic Dynamics and Constraints,” Automatica, Vol. 35, No. 3, pp.407-427, 1999.
- Branicky, M.S., V.S. Borkar, and S.K. Mitter, “A Unified Framework for Hybrid Control: Model and Optimal Control Theory,” IEEE Trans. on Automatic Control, Vol. 43, No. 1, pp. 31-45, 1998.
- Cassandras, C.G., and J. Lygeros, “Stochastic Hybrid Systems,” Taylor and Francis, 2007.
- Grossman, R., A. Nerode, A. Ravn, and H. Rischel, (Eds), Hybrid Systems, Springer, New York, 1993.
- Hristu-Varsakelis, D. and W.S. Levine, Handbook of Networked and Embedded Control Systems, Birkhauser, Boston, 2005.

CONTROL AND OPTIMIZATION – CHALLENGES

1. **SCALABILITY**

2. **DECENTRALIZATION**



Distributed Algorithms

3. **COMMUNICATION**



**Event-driven (asynchronous)
Algorithms**

4. **NON-CONVEXITY**



**Global optimality,
escape local optima**

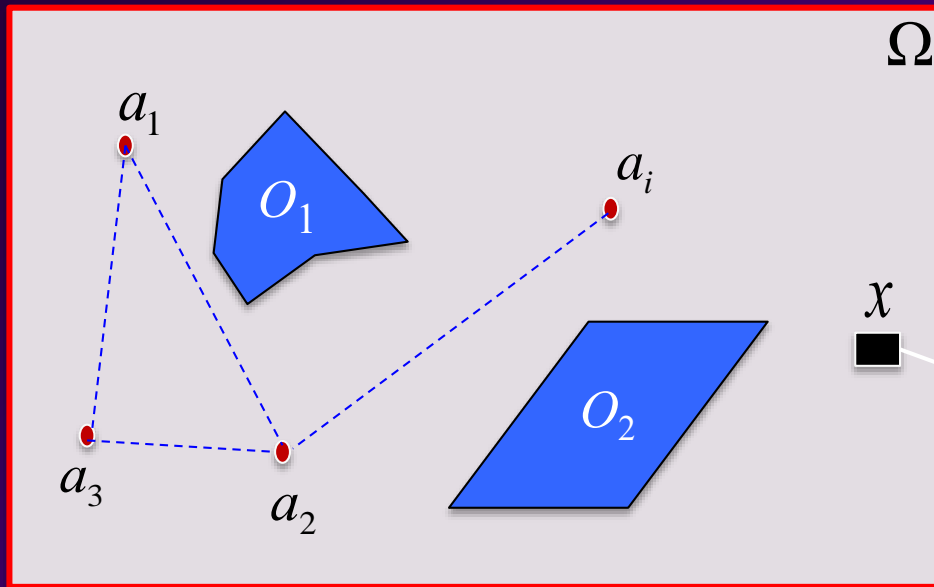
5. **EXPLOIT DATA**



Data-Driven Algorithms

**WHEN CAN WE
DECENTRALIZE ?**

MULTI-AGENT OPTIMIZATION: PROBLEM 1



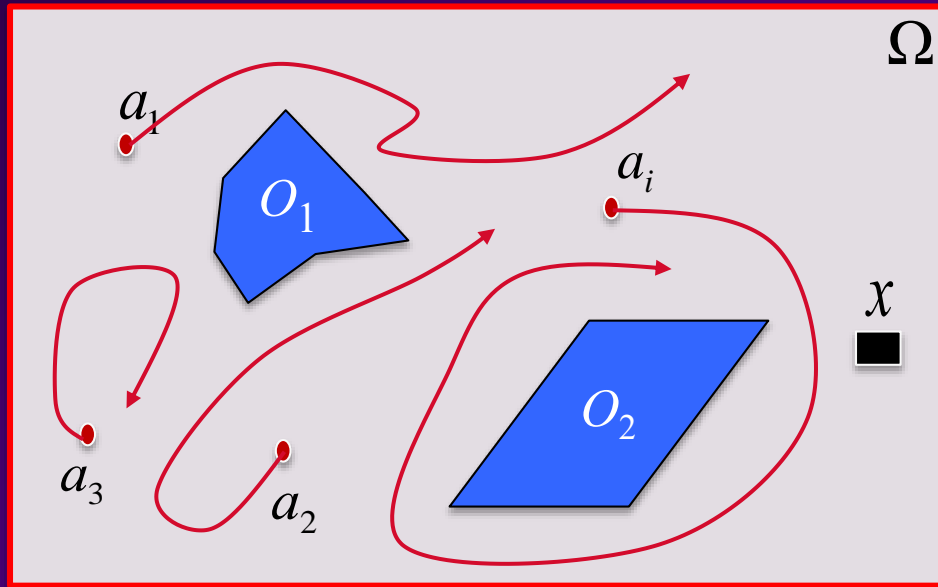
- s_i : agent state, $i = 1, \dots, N$
 $\mathbf{s} = [s_1, \dots, s_N]$
- O_j : obstacle (constraint)
- $R(x)$: property of point x
- $P(x, \mathbf{s})$: reward function

$$\max_{\mathbf{s}} H(\mathbf{s}) = \int_{\Omega} P(x, \mathbf{s}) R(x) dx$$

$$s_i \in F \subseteq \Omega, i = 1, \dots, N$$

GOAL: Find the best **state** vector $\mathbf{s} = [s_1, \dots, s_N]$ so that agents achieve a maximal **reward** from interacting with the mission space

MULTI-AGENT OPTIMIZATION: PROBLEM 2



$$\max_{\mathbf{u}(t)} J = \int_0^T \int_{\Omega} P(x, \mathbf{s}(u(t))) R(x) dx dt$$

May also have dynamics

$$s_i(t) \in F \subseteq \Omega, \quad i = 1, \dots, N$$

$$\dot{s}_i = f_i(s_i, u_i, t), \quad i = 1, \dots, N$$

GOAL: Find the best **state trajectories** $s_i(t)$, $0 \leq t \leq T$ so that agents achieve a maximal **reward** from interacting with the mission space

WHEN CAN WE DECENTRALIZE A MULTI-AGENT PROBLEM 1?

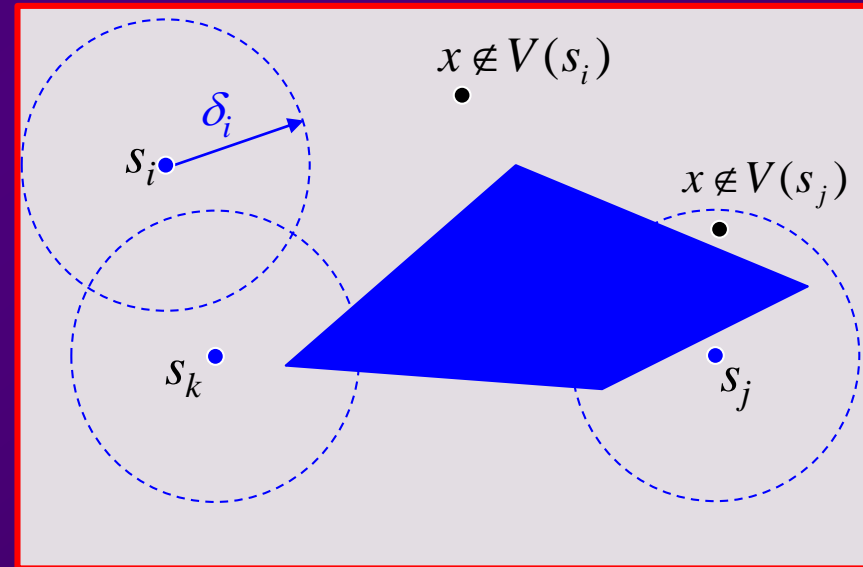
$$\max_{\mathbf{s}} H(\mathbf{s}) = \int_{\Omega} P(x, \mathbf{s}) R(x) dx$$

$$s_i \in F \subseteq \Omega, i = 1, \dots, N$$

Recall:

$$P(x, \mathbf{s}) = 1 - \prod_{i=1}^N [1 - \hat{p}_i(x, s_i)]$$

$$\hat{p}_i(x, s_i) = \begin{cases} p_i(x, s_i) & x \in V(s_i) \\ 0 & \text{otherwise} \end{cases}$$



Define agent i NEIGHBORHOOD:

$$B_i = \{b_i^k : \|s_i - s_k\| < 2\delta_i, k = 1, \dots, a, k \neq i\}$$

OBJECTIVE FUNCTION DECOMPOSITION

THEOREM: If $P(x, \mathbf{s}) = P(p_1, \dots, p_N)$ is a function of local reward functions p_i , then $H(\mathbf{s})$ can be expressed as:

$$H(\mathbf{s}) = H_1(\mathbf{s}_i^L) + H_2(\bar{\mathbf{s}}_i),$$

for any $i = 1, \dots, N$, where $\mathbf{s}_i^L = [s_i, s_{b_i^1}, \dots, s_{b_i^a}]$ and $\bar{\mathbf{s}}_i = [s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_N]$

State of i and its neighbors only

State of all agents except i

- Theorem implies

$$\frac{\partial H(\mathbf{s})}{\partial s_i} = \frac{\partial H_1(\mathbf{s}_i^L)}{\partial s_i}$$

- Distributed gradient-based algorithm:

$$s_i^{k+1} = s_i^k + \beta_k \frac{\partial H_1(\mathbf{s}_i^L)}{\partial s_i^k}$$

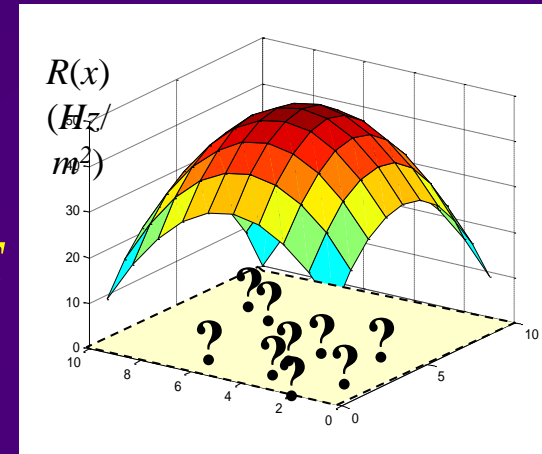
OBJECTIVE FUNCTION DECOMPOSITION

- Theorem 1 often applies and is easy to check for the “Problem 1” setting

EXAMPLE: Coverage Control Problems

COVERAGE: PROBLEM FORMULATION

- N mobile sensors, each located at $s_i \in \mathbb{R}^2$
- Data source at x emits signal with energy E
- Signal observed by sensor node i (at s_i)



■ SENSING MODEL:

$$p_i(x, s_i) \equiv P[\text{Detected by } i \mid A(x), s_i]$$

($A(x)$ = data source emits at x)

■ Sensing attenuation:

$p_i(x, s_i)$ monotonically decreasing in $d_i(x) \equiv \|x - s_i\|$

COVERAGE: PROBLEM FORMULATION

- Joint detection prob. assuming sensor independence ($\mathbf{s} = [s_1, \dots, s_N]$: node locations)

$$P(x, \mathbf{s}) = 1 - \prod_{i=1}^N [1 - p_i(x, s_i)]$$

Event sensing probability

- OBJECTIVE: Determine locations $\mathbf{s} = [s_1, \dots, s_N]$ to maximize total **Detection Probability**:

$$\max_{\mathbf{s}} \int_{\Omega} R(x) P(x, \mathbf{s}) dx$$

Theorem 1 applies

DISTRIBUTED COOPERATIVE SCHEME

- Set

$$H(s_1, \dots, s_N) = \int_{\Omega} R(x) \left\{ 1 - \prod_{i=1}^N [1 - p_i(x)] \right\} dx$$

- Maximize $H(s_1, \dots, s_N)$ by forcing nodes to move using gradient information:

$$\frac{\partial H}{\partial s_k} = \int_{\Omega} R(x) \prod_{i=1, i \neq k}^N [1 - p_i(x)] \frac{\partial p_k(x)}{\partial d_k(x)} \frac{s_k - x}{d_k(x)} dx$$

$$s_i^{k+1} = s_i^k + \beta_k \frac{\partial H}{\partial s_i^k} \rightarrow \text{Desired displacement} = V \cdot \Delta t$$

Cassandras and Li, EJC, 2005
Zhong and Cassandras, IEEE TAC, 2011

$$\frac{\partial H}{\partial s_k} = \int_{\Omega} R(x) \prod_{i=1, i \neq k}^N [1 - p_i(x)] \frac{\partial p_k(x)}{\partial d_k(x)} \frac{s_k - x}{d_k(x)} dx$$

... has to be autonomously evaluated by each node so as to determine how to move to next position:

$$s_i^{k+1} = s_i^k + \beta_k \frac{\partial H}{\partial s_i^k}$$

- Truncated $p_i(x) \Rightarrow \Omega$ replaced by node neighborhood Ω_i
- Discretize $p_i(x)$ using a local grid

CONTROL AND OPTIMIZATION – CHALLENGES

1. **SCALABILITY**

2. **DECENTRALIZATION**



Distributed Algorithms

3. **COMMUNICATION**



**Event-driven (asynchronous)
Algorithms**

4. **NON-CONVEXITY**



**Global optimality,
escape local optima**

5. **EXPLOIT DATA**



Data-Driven Algorithms

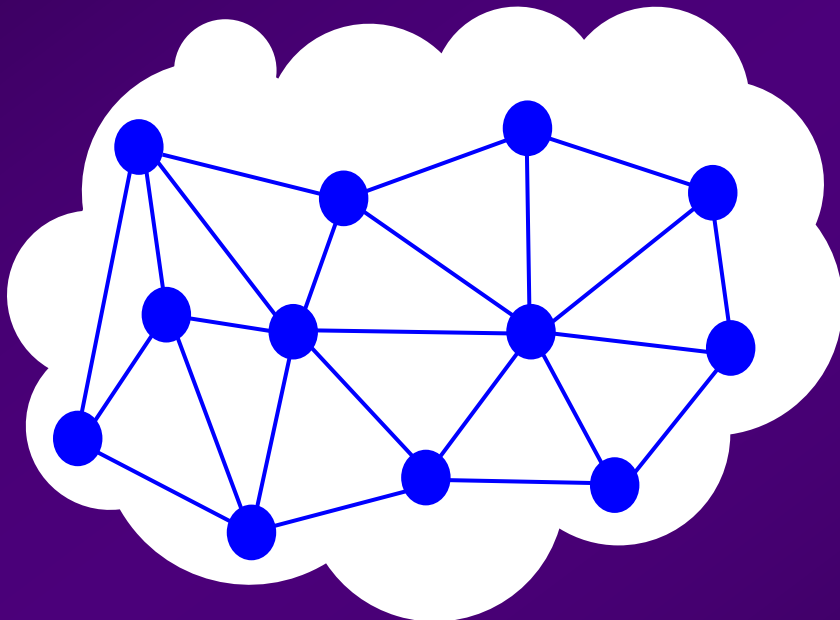
EVENT-DRIVEN DISTRIBUTED ALGORITHMS

DISTRIBUTED COOPERATIVE OPTIMIZATION

N system components
(processors, agents, vehicles, nodes),
one common objective:

$$\min_{s_1, \dots, s_N} H(s_1, \dots, s_N)$$

s.t. constraints on each s_i



$$\min_{s_1} H(s_1, \dots, s_N)$$

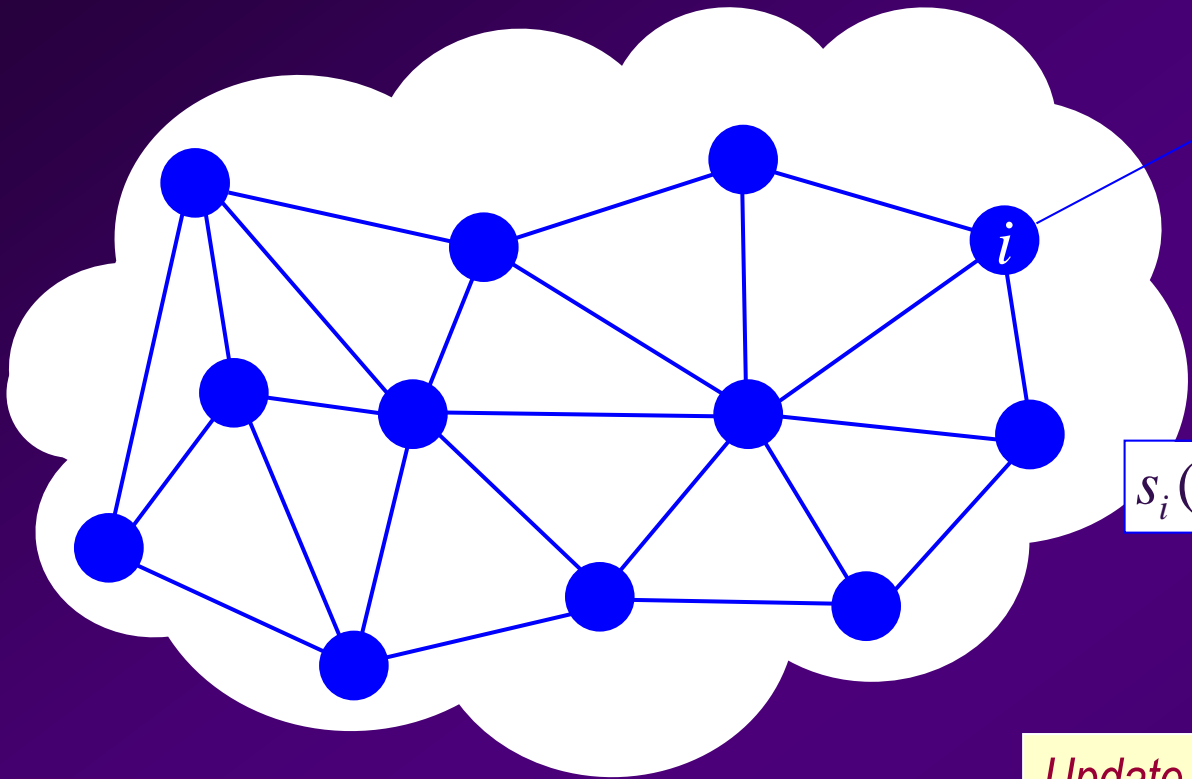
s.t. constraints on s_1

⋮

$$\min_{s_N} H(s_1, \dots, s_N)$$

s.t. constraints on s_N

DISTRIBUTED COOPERATIVE OPTIMIZATION



Controllable state
 $s_i, i = 1, \dots, n_i$



$$s_i(k+1) = s_i(k) + \alpha_i d_i(\mathbf{s}(k))$$

Step Size

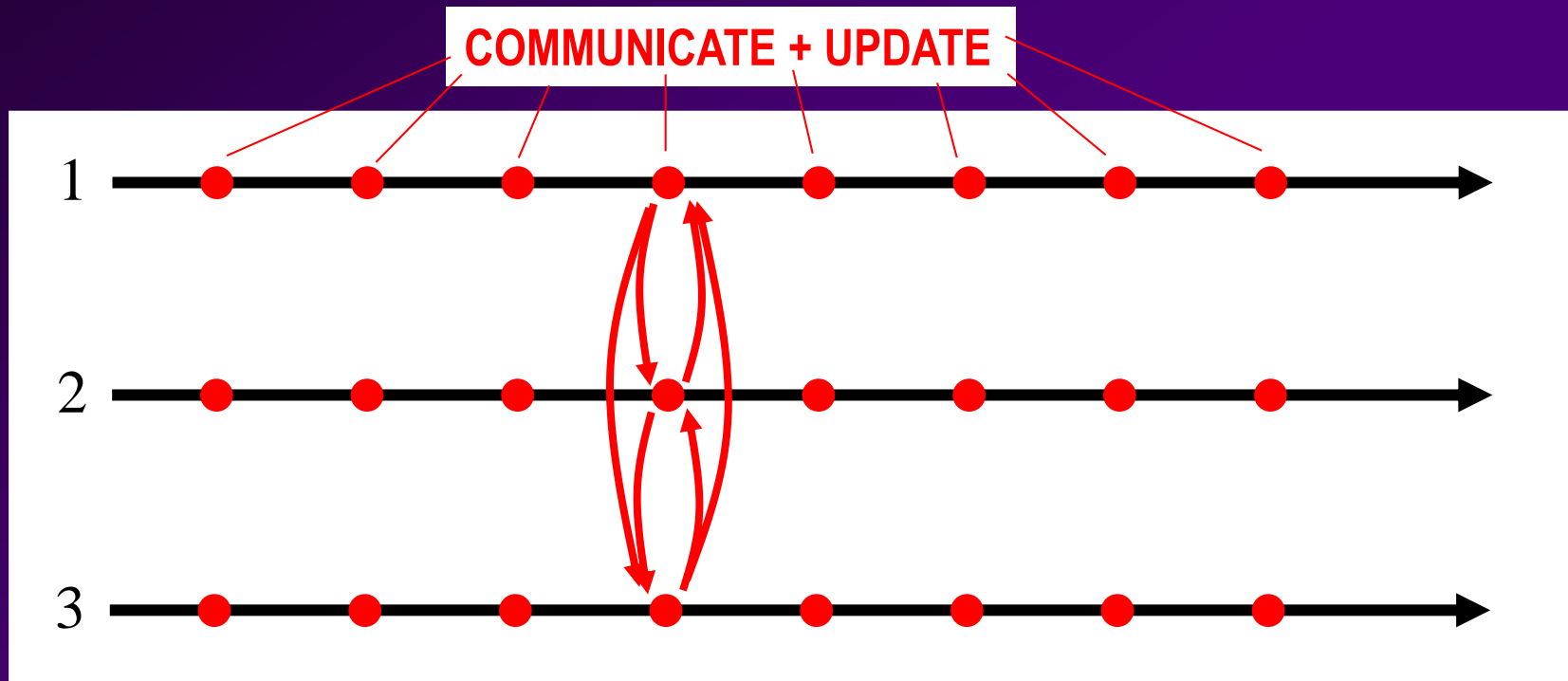
Update Direction, usually
 $d_i(\mathbf{s}(k)) = -\nabla_i H(\mathbf{s}(k))$

i requires knowledge of all s_1, \dots, s_N

Inter-node communication

$$\begin{aligned} \min_{s_i} & H(s_1, \dots, s_N) \\ \text{s.t.} & \text{ constraints on } s_i \end{aligned}$$

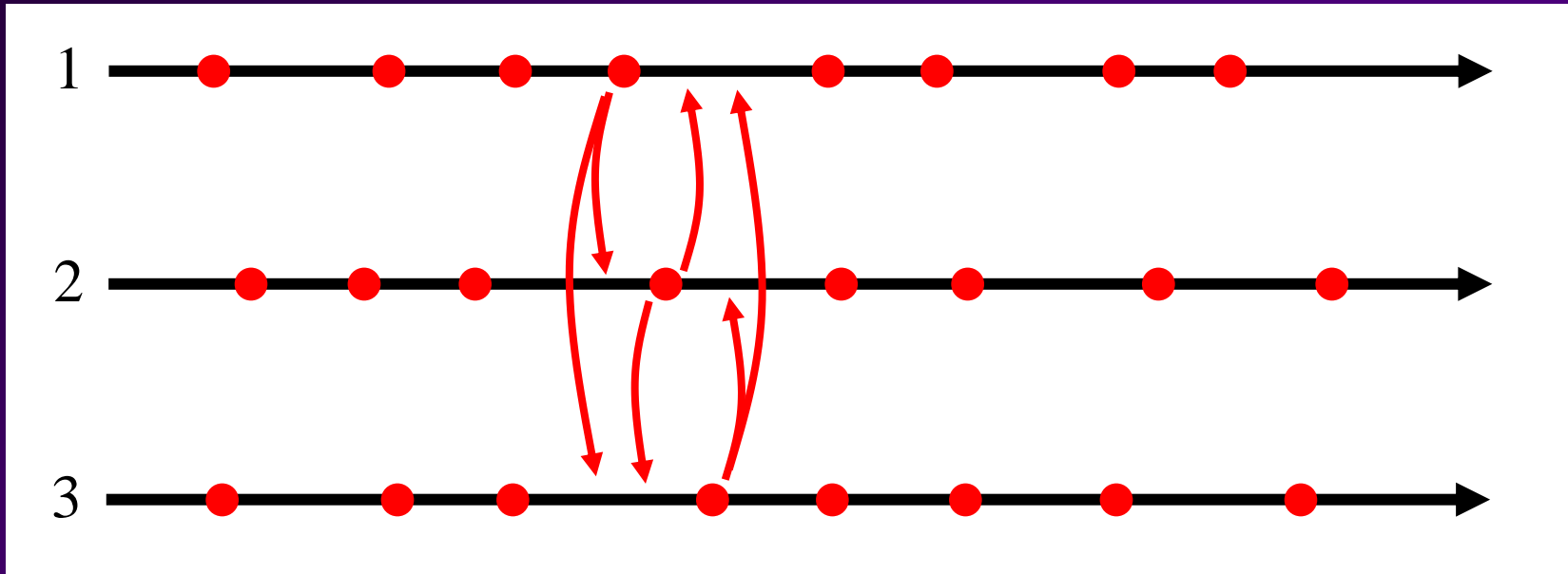
SYNCHRONIZED (TIME-DRIVEN) COOPERATION



Drawbacks:

- Excessive communication (critical in wireless settings!)
- Faster nodes have to wait for slower ones
- Clock synchronization infeasible
- Bandwidth limitations
- Security risks

ASYNCHRONOUS COOPERATION



- Nodes not synchronized, delayed information used

Update frequency for each node
is bounded
+
technical conditions

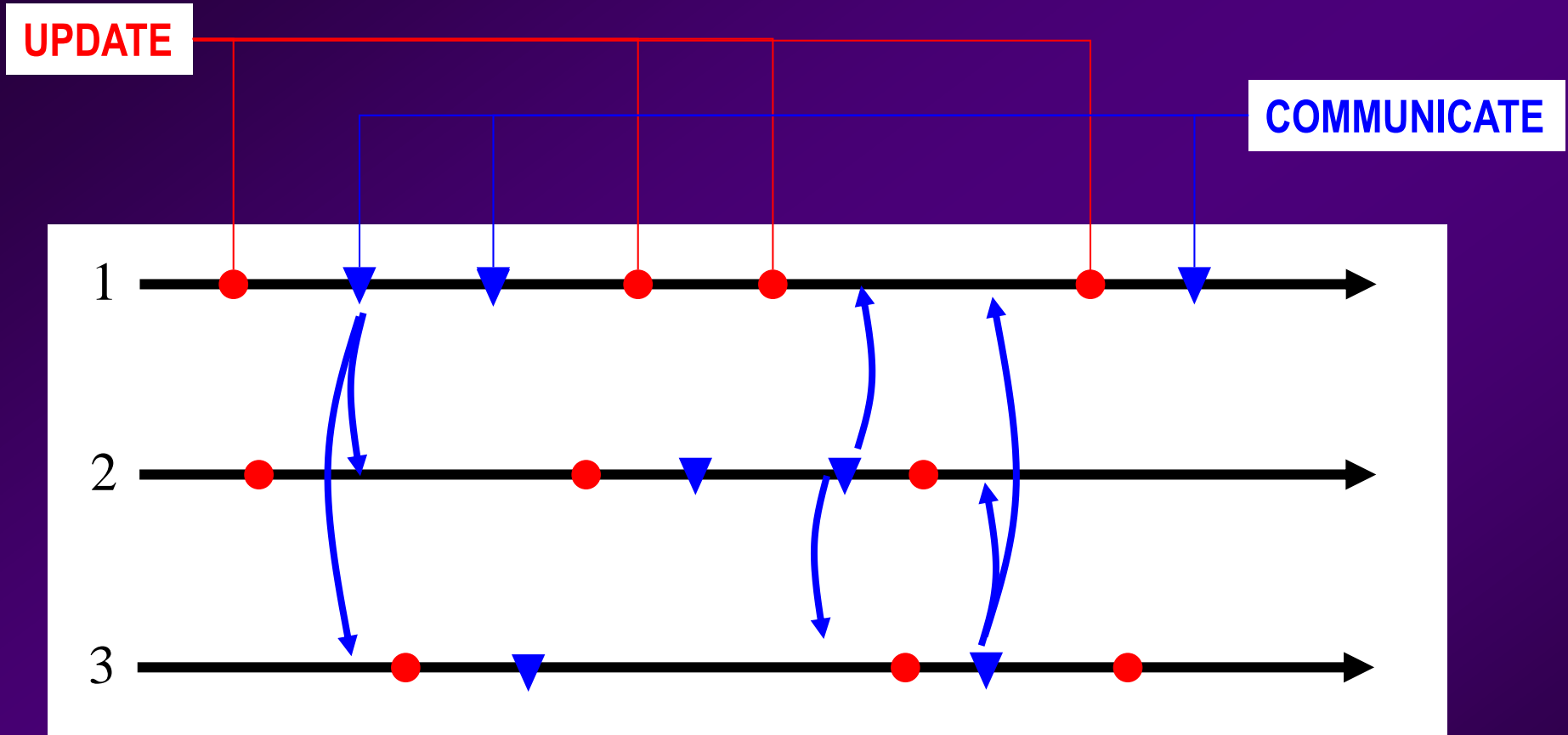
⇒

$$s_i(k+1) = s_i(k) + \alpha_i d_i(\mathbf{s}(k))$$

converges

Bertsekas and Tsitsiklis, 1997

ASYNCHRONOUS (EVENT-DRIVEN) COOPERATION



- UPDATE at i : locally determined, arbitrary (possibly periodic)
- COMMUNICATE from i : only when absolutely necessary

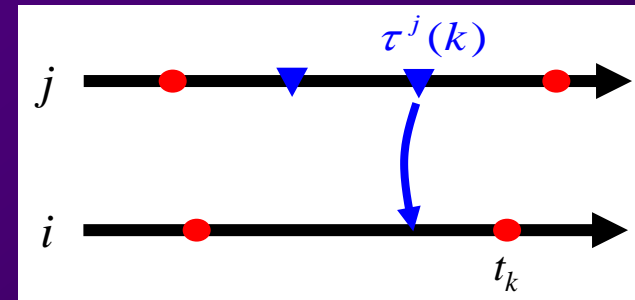
WHEN SHOULD A NODE COMMUNICATE?

Node state at any time t : $x_i(t)$
Node state at t_k : $s_i(k)$ } $\Rightarrow s_i(k) = x_i(t_k)$

AT UPDATE TIME t_k : $s_j^i(k)$: node j **state** estimated by node i

Estimate examples:

$\rightarrow s_j^i(k) = x_j(\tau^j(k))$ Most recent value



$\rightarrow s_j^i(k) = x_j(\tau^j(k)) + \frac{t_k - \tau^j(k)}{\Delta_j} \cdot \alpha_i \cdot d_j(x_j(\tau^j(k)))$ Linear prediction

WHEN SHOULD A NODE COMMUNICATE?

AT ANY TIME t :

- $x_i^j(t)$: node i state estimated by node j
- If node i knows how j estimates its state, then it can evaluate $x_i^j(t)$
- Node i uses
 - its own **true state**, $x_i(t)$
 - the **estimate that j uses**, $x_i^j(t)$

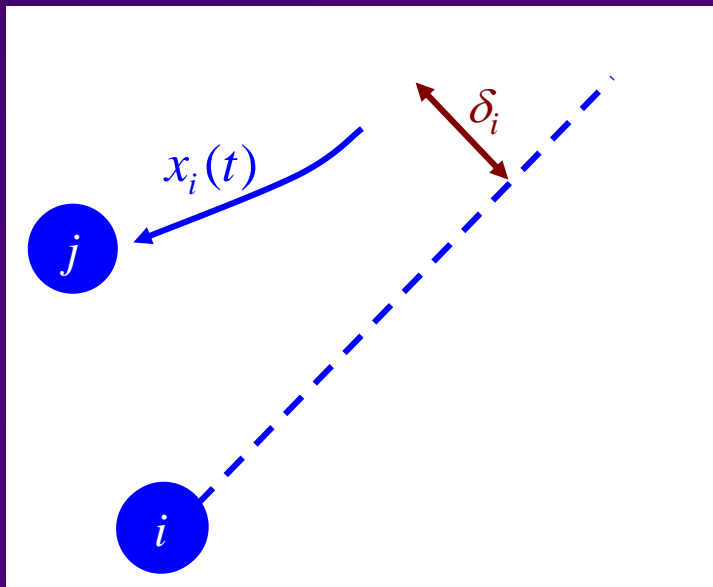
... and evaluates an ERROR FUNCTION $g(x_i(t), x_i^j(t))$

Error Function examples: $\|x_i(t) - x_i^j(t)\|_1$, $\|x_i(t) - x_i^j(t)\|_2$

WHEN SHOULD A NODE COMMUNICATE?

Compare ERROR FUNCTION $g(x_i(t), x_i^j(t))$ to THRESHOLD δ_i

Node i communicates its state to node j only when it detects that its *true state* $x_i(t)$ deviates from j ' estimate of it $x_i^j(t)$ so that $g(x_i(t), x_i^j(t)) \geq \delta_i$



\Rightarrow **Event-Driven** Control

THRESHOLD PROCESS

$$K_\delta > 0$$

Update Direction, usually

$$d_i(\mathbf{s}^i(k)) = -\nabla_i H(\mathbf{s}^i(k))$$

$$\delta_i(k) = \begin{cases} K_\delta \|d_i(\mathbf{s}^i(k))\| & \text{if } k \in C^i \\ \delta_i(k-1) & \text{otherwise} \end{cases}$$

$$\delta_i(0) = K_\delta \|d_i(\mathbf{s}^i(0))\|$$

Intuition:

near convergence

(small $d_i(\mathbf{s}^i(k))$),

better estimates are needed

CONVERGENCE

Asynchronous distributed state update process at each i :

$$s_i(k+1) = s_i(k) + \alpha \cdot d_i(\mathbf{s}^i(k))$$

*Estimates of other nodes,
evaluated by node i*

$$\delta_i(k) = \begin{cases} K_\delta \|d_i(\mathbf{s}^i(k))\| & \text{if } k \text{ sends update} \\ \delta_i(k-1) & \text{otherwise} \end{cases}$$

ASSUMPTION 1: There exists a positive integer B such that for all $i = 1, \dots, N$ and $k \geq 0$ at least one of the elements of the set $\{k-B+1, k-B+2, \dots, k\}$ belongs to C^i .

INTERPRETATION: Each node updates its state at least once during a period in which B state update events take place (no time bound)

ASSUMPTION 2: The objective function $H(\mathbf{s})$, $\mathbf{s} \in \mathfrak{R}^m$, $m = \sum_{i=1}^N n_i$ satisfies:

(a) $H(\mathbf{s}) \geq 0$, for all $\mathbf{s} \in \mathfrak{R}^m$

(b) $H(\cdot)$ continuously differentiable and $\nabla H(\cdot)$ Lipschitz continuous, i.e., there exists K_1 such that for all $\mathbf{x}, \mathbf{y} \in \mathfrak{R}^m$

$$\|\nabla H(\mathbf{x}) - \nabla H(\mathbf{y})\| \leq K_1 \|\mathbf{x} - \mathbf{y}\|$$

ASSUMPTION 3: There exist positive constants K_2, K_3 such that for all $i = 1, \dots, N$ and $k \in C^i$

$$(a) d_i(k)' \nabla_i H(s^i(k)) \leq -\|d_i(k)\|^2 / K_3$$

$$(b) K_2 \nabla_i H(s^i(k)) \leq \|d_i(k)\|$$

NOTE: Very mild condition, immediately satisfied with $K_2 = K_3 = 1$ when we use the usual update direction given by $d_i(k) = -\nabla_i H(s^i(k))$

ASSUMPTION 4: There exists a positive constant K_4 such that The ERROR FUNCTION satisfies

$$\|x_i(t) - x_i^j(t)\| \leq K_4 g(x_i(t) - x_i^j(t))$$

NOTE: Very mild condition, immediately satisfied with $K_4 = 1$ when we use the common choice $g(x_i(t) - x_i^j(t)) = \|x_i(t) - x_i^j(t)\|$

THEOREM: Under A1-A4, there exist positive constants α and K_δ such that

$$\lim_{k \rightarrow \infty} \nabla H(\mathbf{s}(k)) = 0$$

Zhong and Cassandras, IEEE TAC, 2010

INTERPRETATION:

- *Event-driven optimization achievable with reduced communication requirements \Rightarrow energy savings*
- *No loss of performance*

THEOREM: Under A1-A4, there exist positive constants α and K_δ such that

$$\lim_{k \rightarrow \infty} \nabla H(\mathbf{s}(k)) = 0$$

BYPRODUCT OF PROOF:

obtaining the *largest* possible K_δ and hence the *smallest* possible number of communication events:

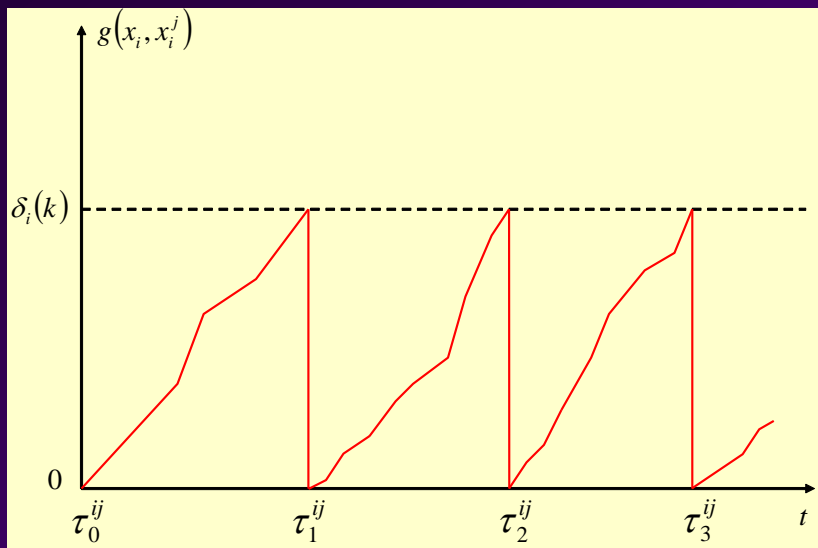
$$K_\delta < \frac{1}{(1+B)K_4\sqrt{m}} \left(\frac{2}{K_1K_3} - \alpha \right)$$

Comm.
frequency

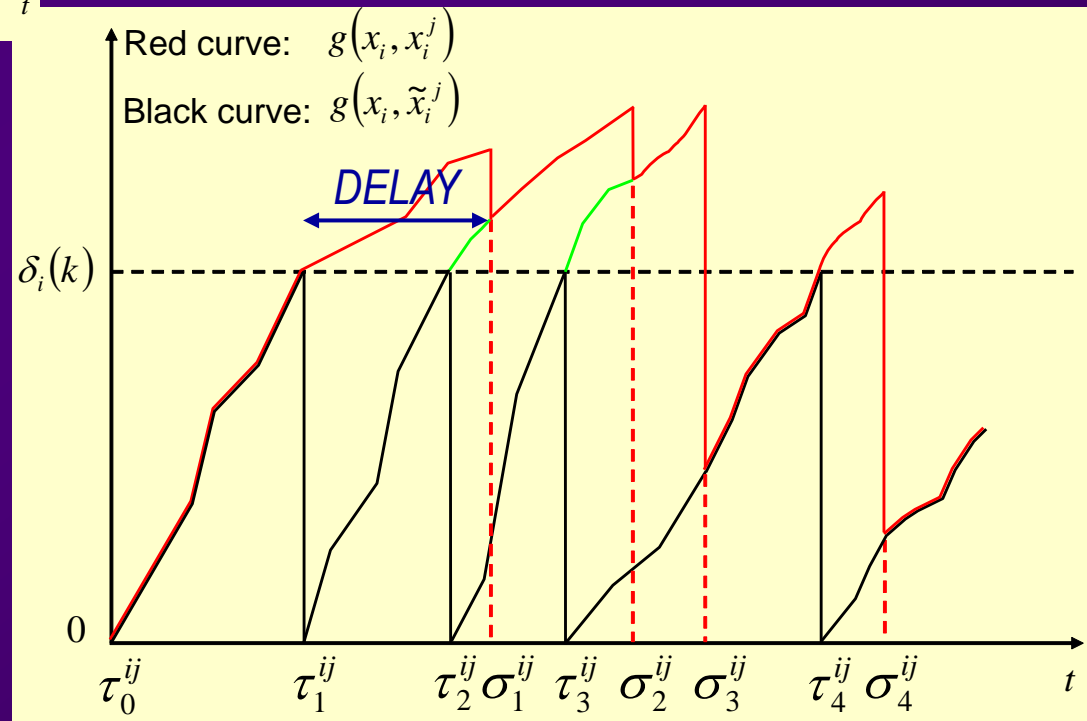
$$0 < \alpha < 2 / K_1K_3$$

State dim. \sim network dim.

COONVERGENCE WHEN DELAYS ARE PRESENT



Error function trajectory with NO DELAY



COONVERGENCE WHEN DELAYS ARE PRESENT

Add a boundedness assumption:

ASSUMPTION 5: There exists a non-negative integer D such that if a message is sent before t_{k-D} from node i to node j , it will be received before t_k .

INTERPRETATION: at most D state update events can occur between a node sending a message and all destination nodes receiving this message.

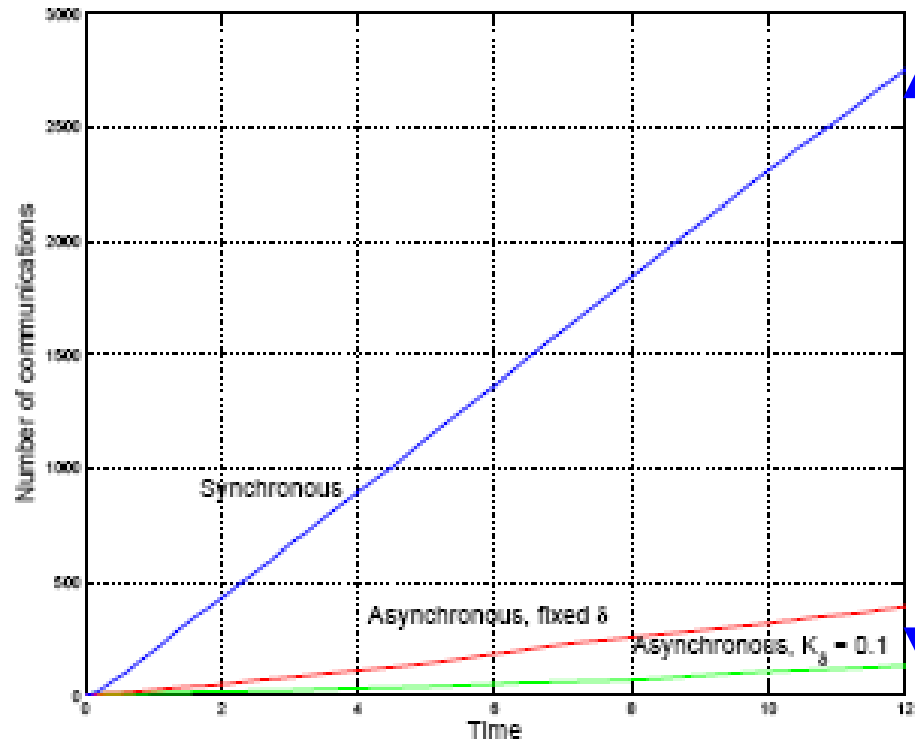
THEOREM: Under A1-A5, there exist positive constants α and K_δ such that

$$\lim_{k \rightarrow \infty} \nabla H(\mathbf{s}(k)) = 0$$

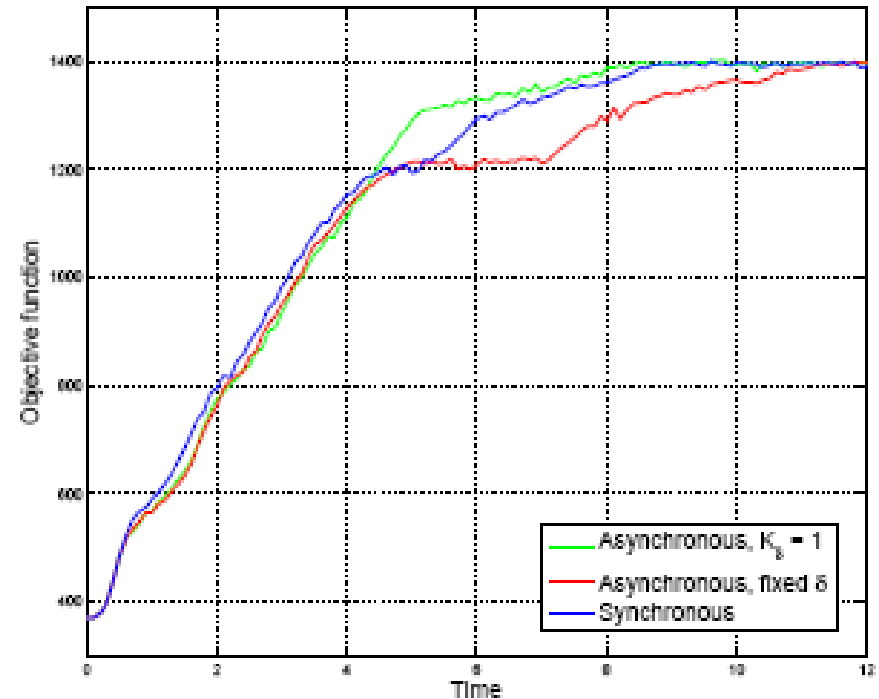
NOTE: The requirements on α and K_δ depend on D and they are tighter.

Zhong and Cassandras, IEEE TAC, 2010

SYNCHRONOUS v ASYNCHRONOUS OPTIMAL COVERAGE PERFORMANCE



Energy savings + Extended lifetime



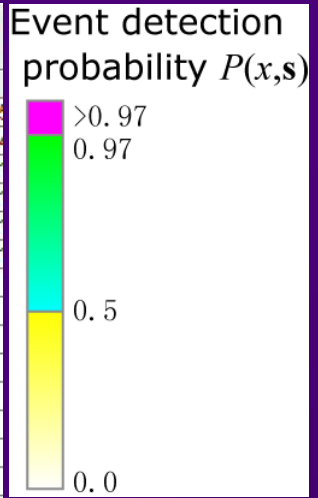
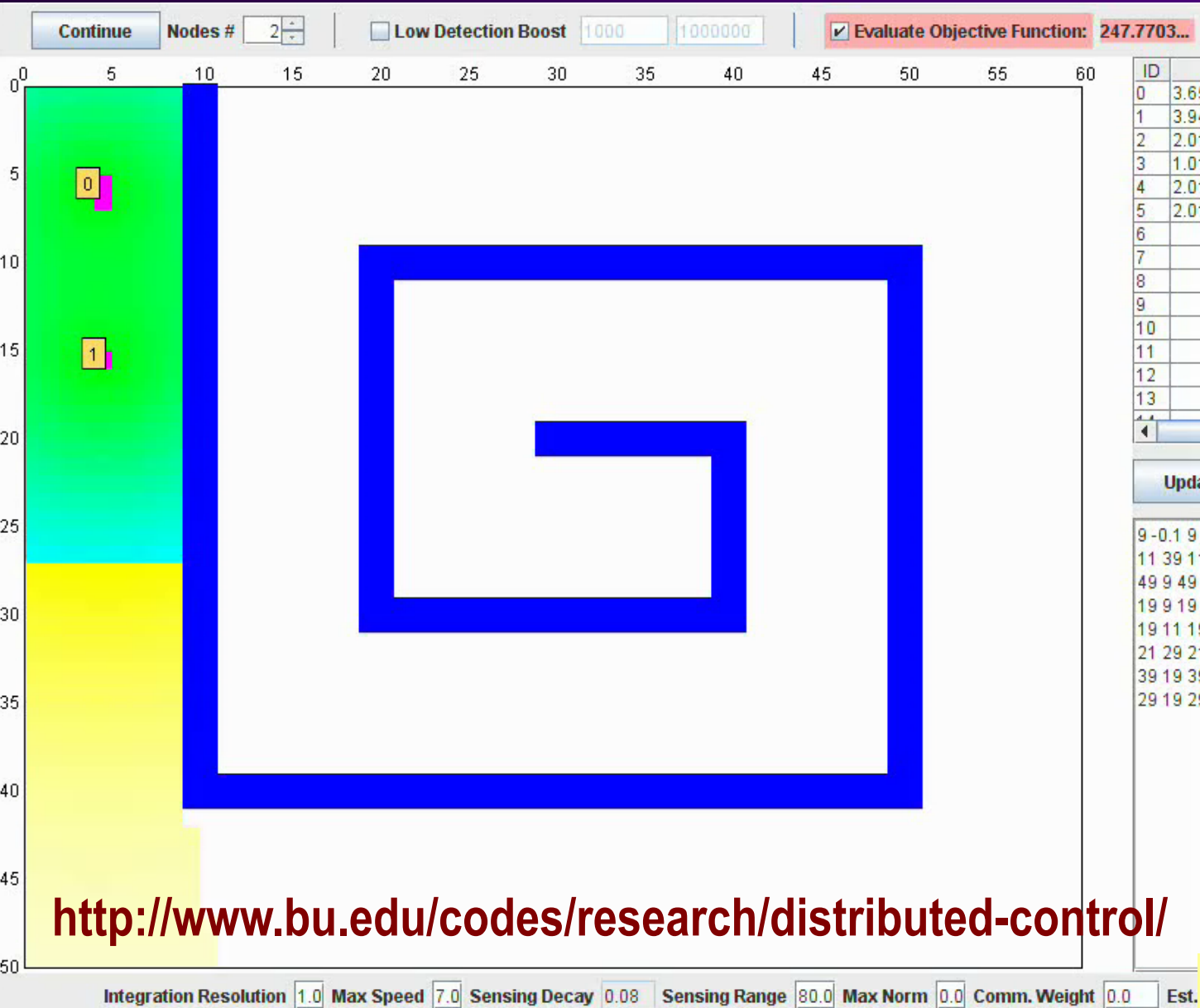
SYNCHRONOUS v ASYNCHRONOUS:

No. of communication events
for a deployment problem *with obstacles*

SYNCHRONOUS v ASYNCHRONOUS:

Achieving optimality
in a problem *with obstacles*

OPTIMAL COVERAGE IN A MAZE



ID	
0	3.63
1	3.94
2	2.01
3	1.01
4	2.01
5	2.01
6	
7	
8	
9	
10	
11	
12	
13	

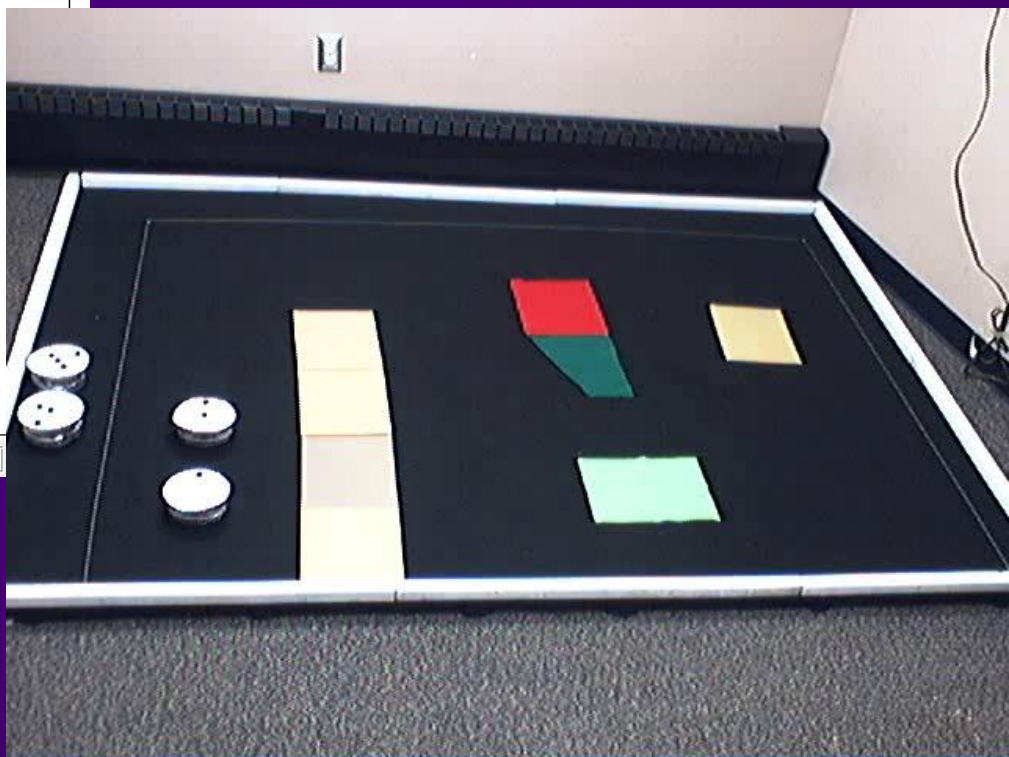
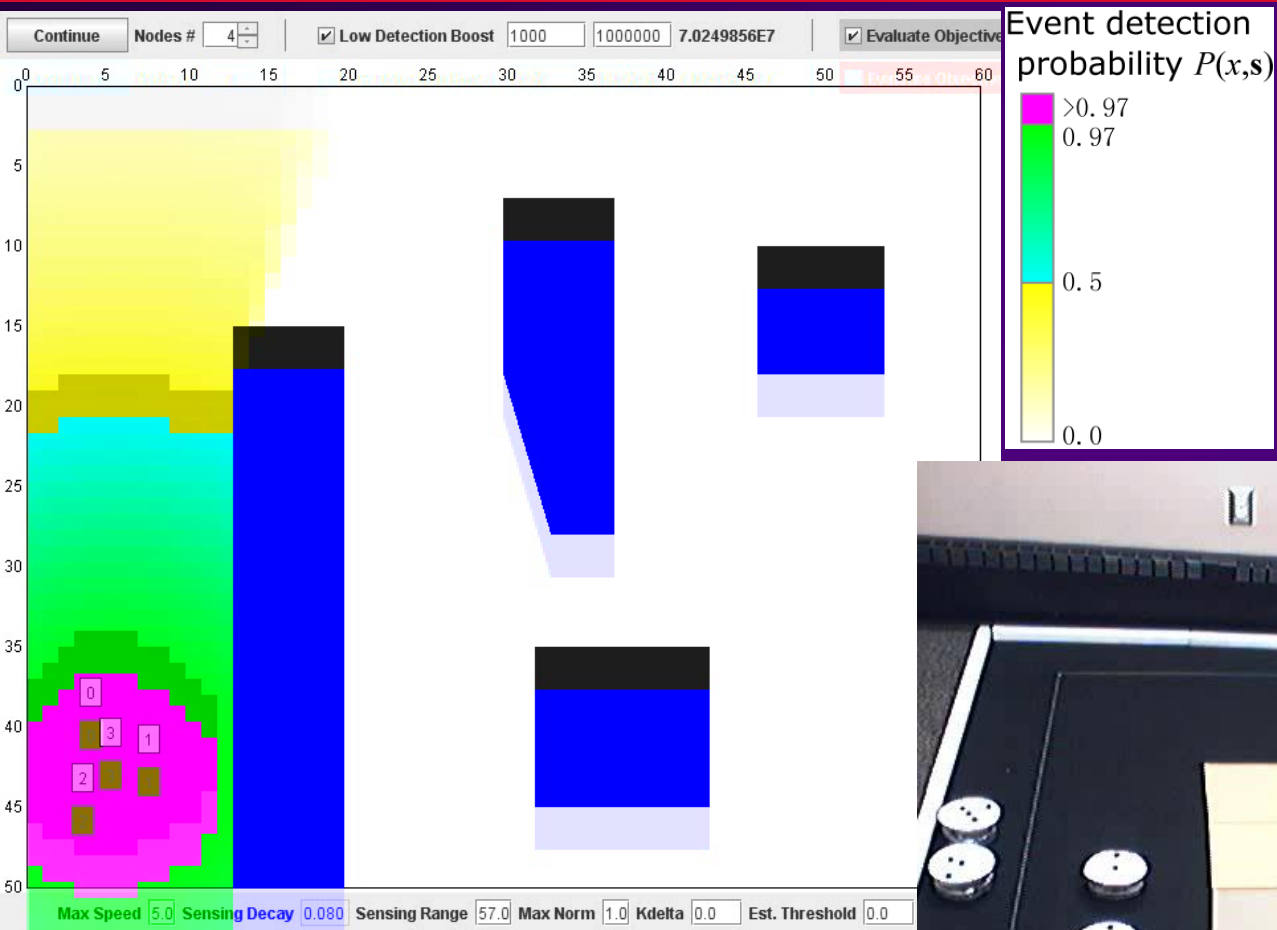
Update

9	-0.1 9
11	39 1
49	9 49
19	9 19
19	11 19
21	29 2
39	19 39
29	19 29

<http://www.bu.edu/codes/research/distributed-control/>

Zhong and Cassandras, 2008

DEMO: OPTIMAL DISTRIBUTED DEPLOYMENT WITH OBSTACLES – SIMULATED AND REAL



**IT IS HARD TO
DECENTRALIZE
PROBLEM 2 ...**

MORE ON THAT LATER...

CONTROL AND OPTIMIZATION – CHALLENGES

1. **SCALABILITY**

2. **DECENTRALIZATION**



Distributed Algorithms

3. **COMMUNICATION**



**Event-driven (asynchronous)
Algorithms**

4. **NON-CONVEXITY**



**Global optimality,
escape local optima**

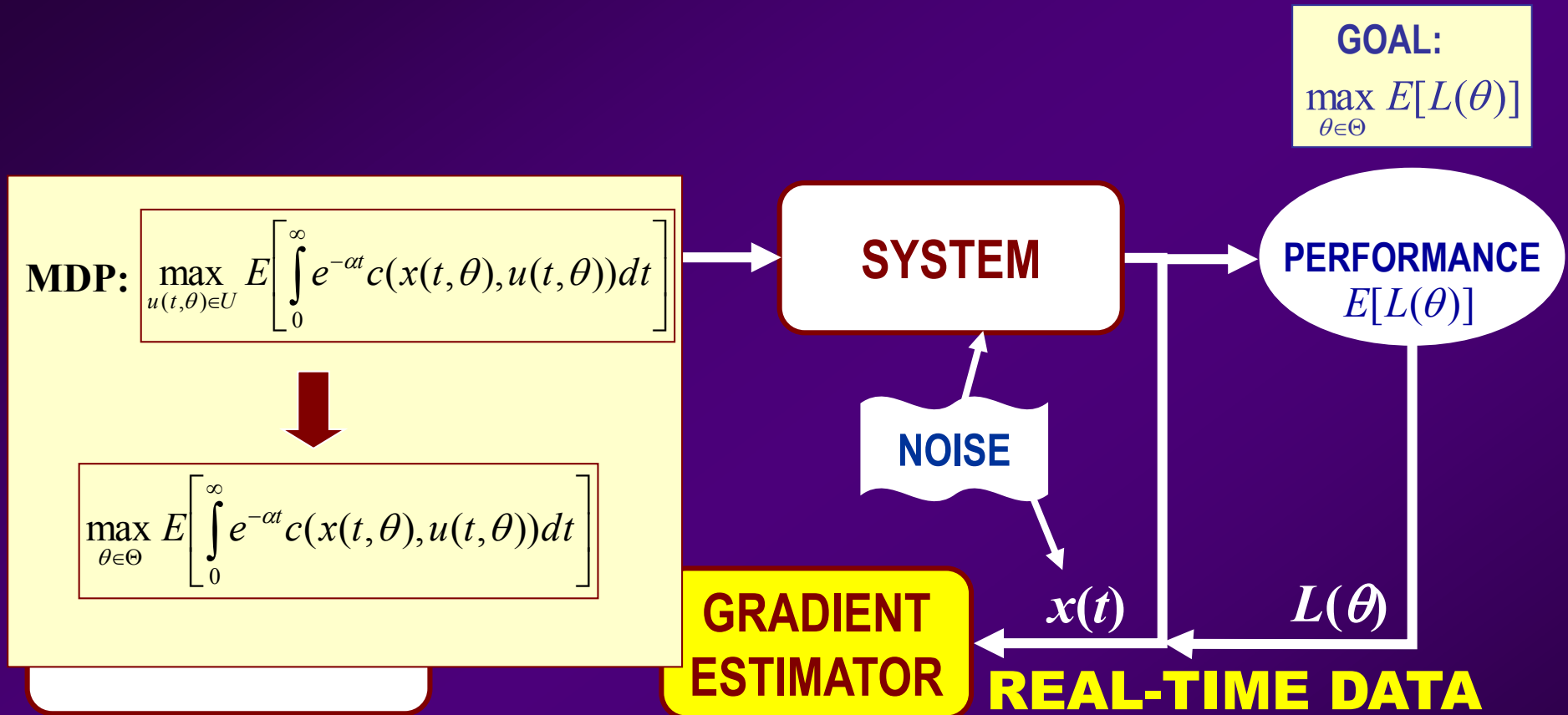
5. **EXLOIT DATA**



Data-Driven Algorithms

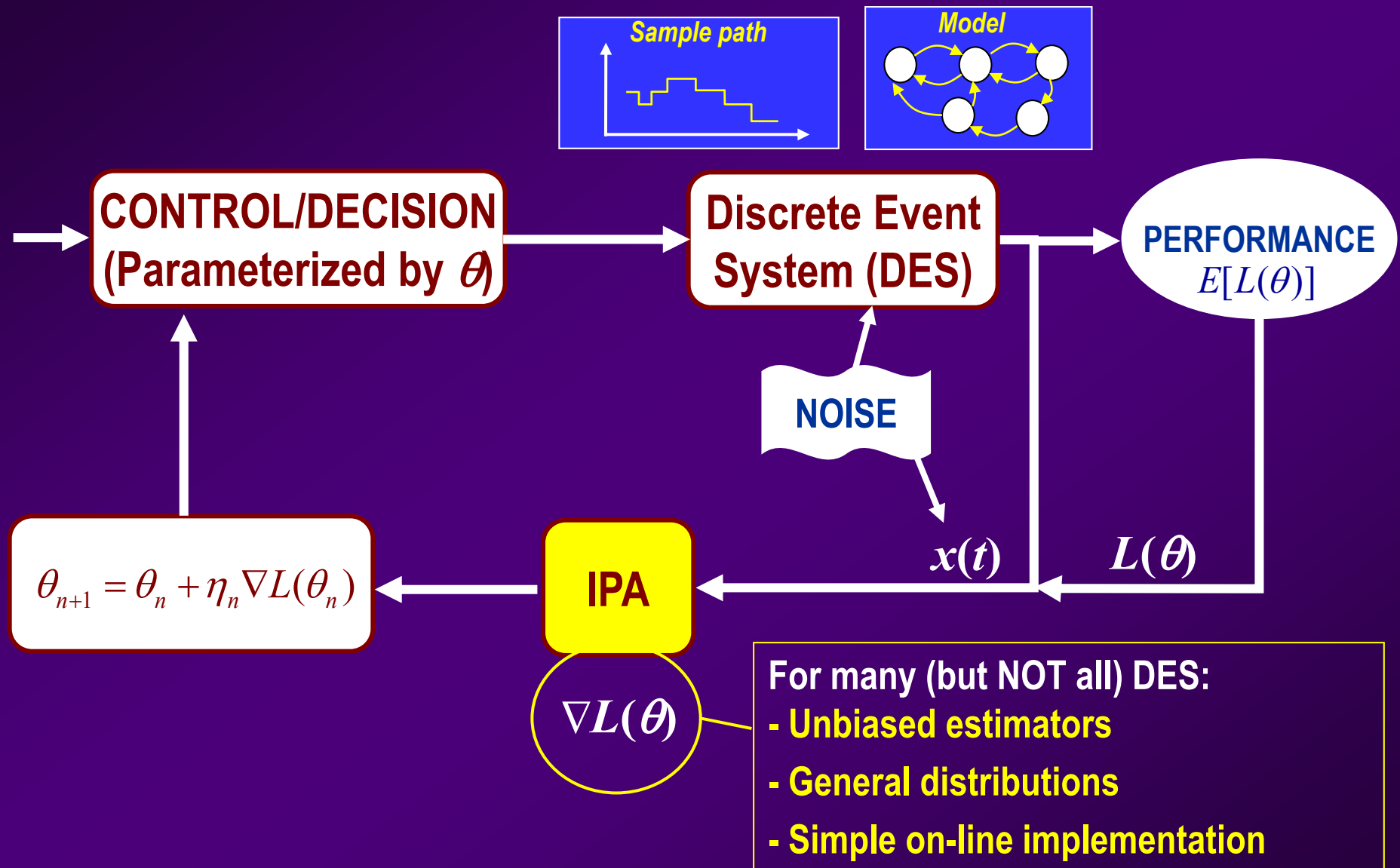
DATA-DRIVEN + EVENT-DRIVEN ALGORITHMS

DATA-DRIVEN STOCHASTIC OPTIMIZATION

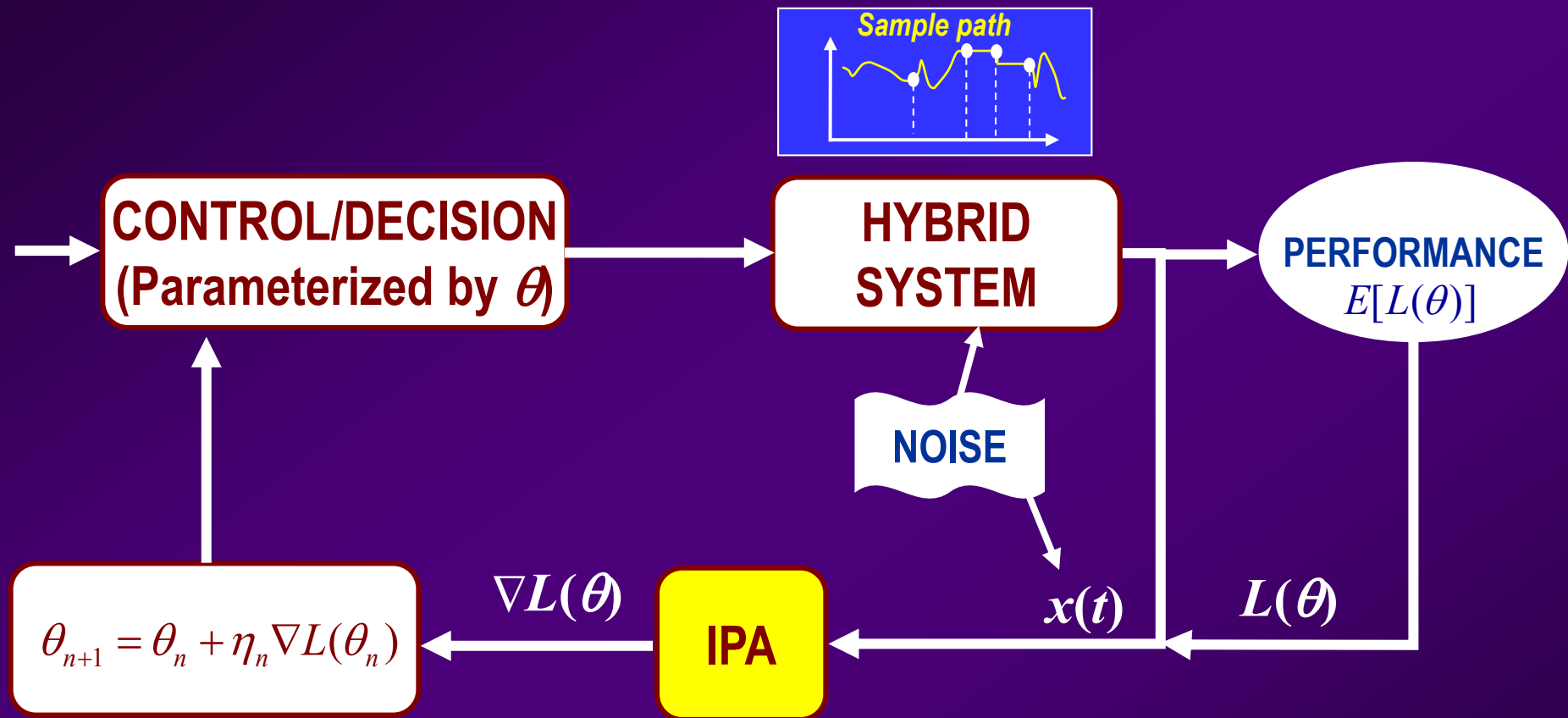


- DIFFICULTIES:**
- $E[L(\theta)]$ NOT available in closed form
 - $\nabla L(\theta)$ not easy to evaluate
 - $\nabla L(\theta)$ may not be a good estimate of $\nabla E[L(\theta)]$

DATA-DRIVEN STOCHASTIC OPTIMIZATION IN *DES*: INFINITESIMAL PERTURBATION ANALYSIS (IPA)



REAL-TIME STOCHASTIC OPTIMIZATION: *HYBRID SYSTEMS*



A general framework for an IPA theory in Hybrid Systems

PERFORMANCE OPTIMIZATION AND IPA

Performance metric
(objective function):

$$J(\theta; x(\theta, 0), T) = E[L(\theta; x(\theta, 0), T)]$$

IPA goal:

- Obtain unbiased estimates of $\frac{dJ(\theta; x(\theta, 0), T)}{d\theta}$, normally $\frac{dL(\theta)}{d\theta}$
- Then: $\theta_{n+1} = \theta_n + \eta_n \frac{dL(\theta_n)}{d\theta}$

NOTATION:

$$x'(t) = \frac{\partial x(\theta, t)}{\partial \theta}, \quad \tau'_k = \frac{d\tau_k(\theta)}{d\theta}$$

THE IPA CALCULUS

IPA: *THREE FUNDAMENTAL EQUATIONS*

System dynamics over $(\tau_k(\theta), \tau_{k+1}(\theta)]$: $\dot{x} = f_k(x, \theta, t)$

NOTATION: $x'(t) = \frac{\partial x(\theta, t)}{\partial \theta}$, $\tau'_k = \frac{\partial \tau_k(\theta)}{\partial \theta}$

1. Continuity at events: $x(\tau_k^+) = x(\tau_k^-)$

Take $d/d\theta$:

$$x'(\tau_k^+) = x'(\tau_k^-) + [f_{k-1}(\tau_k^-) - f_k(\tau_k^+)]\tau'_k$$

If no continuity, use reset condition $\Rightarrow x'(\tau_k^+) = \frac{d\rho(q, q', x, v, \delta)}{d\theta}$

IPA: *THREE FUNDAMENTAL EQUATIONS*

2. Take $d/d\theta$ of system dynamics $\dot{x} = f_k(x, \theta, t)$ over $(\tau_k(\theta), \tau_{k+1}(\theta))$:

$$\frac{dx'(t)}{dt} = \frac{\partial f_k(t)}{\partial x} x'(t) + \frac{\partial f_k(t)}{\partial \theta}$$

Solve $\frac{dx'(t)}{dt} = \frac{\partial f_k(t)}{\partial x} x'(t) + \frac{\partial f_k(t)}{\partial \theta}$ over $(\tau_k(\theta), \tau_{k+1}(\theta))$:

$$x'(t) = e^{\int_{\tau_k}^t \frac{\partial f_k(u)}{\partial x} du} \left[\int_{\tau_k}^t \frac{\partial f_k(v)}{\partial \theta} e^{-\int_{\tau_k}^v \frac{\partial f_k(u)}{\partial x} du} dv + x'(\tau_k^+) \right]$$

initial condition from 1 above

NOTE: If there are no events (pure time-driven system),
IPA reduces to this equation

IPA: *THREE FUNDAMENTAL EQUATIONS*

3. Get τ'_k depending on the event type:

- **Exogenous** event: By definition, $\tau'_k = 0$

- **Endogenous** event: occurs when $g_k(x(\theta, \tau_k), \theta) = 0$

$$\tau'_k = - \left[\frac{\partial g}{\partial x} f_k(\tau_k^-) \right]^{-1} \left(\frac{\partial g}{\partial \theta} + \frac{\partial g}{\partial x} x'(\tau_k^-) \right)$$

- **Induced** events:

$$\tau'_k = - \left[\frac{\partial y_k(\tau_k)}{\partial t} \right]^{-1} y'_k(\tau_k^+)$$

IPA: *THREE FUNDAMENTAL EQUATIONS*

Ignoring resets and induced events:

$$1. \quad x'(\tau_k^+) = x'(\tau_k^-) + [f_{k-1}(\tau_k^-) - f_k(\tau_k^+)] \cdot \tau'_k$$

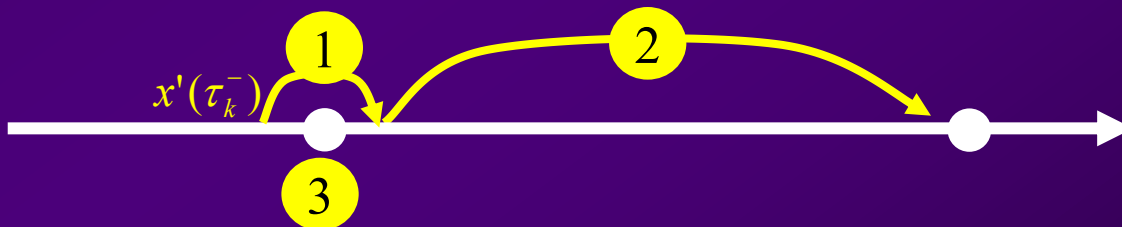
$$2. \quad x'(t) = e^{\int_{\tau_k}^t \frac{\partial f_k(u)}{\partial x} du} \left[\int_{\tau_k}^t \frac{\partial f_k(v)}{\partial \theta} e^{-\int_{\tau_k}^v \frac{\partial f_k(u)}{\partial x} du} dv + x'(\tau_k^+) \right]$$

$$3. \quad \tau'_k = 0 \quad \text{or} \quad \tau'_k = - \left[\frac{\partial g}{\partial x} f_k(\tau_k^-) \right]^{-1} \left(\frac{\partial g}{\partial \theta} + \frac{\partial g}{\partial x} x'(\tau_k^-) \right)$$

Recall:

$$x'(t) = \frac{\partial x(\theta, t)}{\partial \theta}$$

$$\tau'_k = \frac{\partial \tau_k(\theta)}{\partial \theta}$$



Cassandras et al, *Europ. J. Control*, 2010

IPA PROPERTIES

Back to performance metric: $L(\theta) = \sum_{k=0}^N \int_{\tau_k}^{\tau_{k+1}} L_k(x, \theta, t) dt$

NOTATION: $L'_k(x, \theta, t) = \frac{\partial L_k(x, \theta, t)}{\partial \theta}$

Then: $\frac{dL(\theta)}{d\theta} = \sum_{k=0}^N \left[\tau'_{k+1} \cdot L_k(\tau_{k+1}) - \tau'_k \cdot L_k(\tau_k) + \int_{\tau_k}^{\tau_{k+1}} L'_k(x, \theta, t) dt \right]$

What happens
at event times

What happens
between event times

IPA PROPERTY 1: **ROBUSTNESS**

THEOREM 1: If either 1,2 holds, then $dL(\theta)/d\theta$ depends only on information available at event times τ_k :

1. $L(x, \theta, t)$ is independent of t over $[\tau_k(\theta), \tau_{k+1}(\theta)]$ for all k
2. $L(x, \theta, t)$ is only a function of x and for all t over $[\tau_k(\theta), \tau_{k+1}(\theta)]$:

$$\frac{d}{dt} \frac{\partial L_k}{\partial x} = \frac{d}{dt} \frac{\partial f_k}{\partial x} = \frac{d}{dt} \frac{\partial f_k}{\partial \theta} = 0$$

$$\frac{dL(\theta)}{d\theta} = \sum_{k=0}^N \left[\tau'_{k+1} \cdot L_k(\tau_{k+1}) - \tau'_k \cdot L_k(\tau_k) + \int_{\tau_k}^{\tau_{k+1}} \cancel{L'_k(x, \theta, t)} dt \right]$$

IMPLICATION: - Performance sensitivities can be obtained from information limited to event times, which is easily observed
- **No need to track system in between events !**

IPA PROPERTY 1: **ROBUSTNESS**

EXAMPLE WHERE THEOREM 1 APPLIES (simple tracking problem):

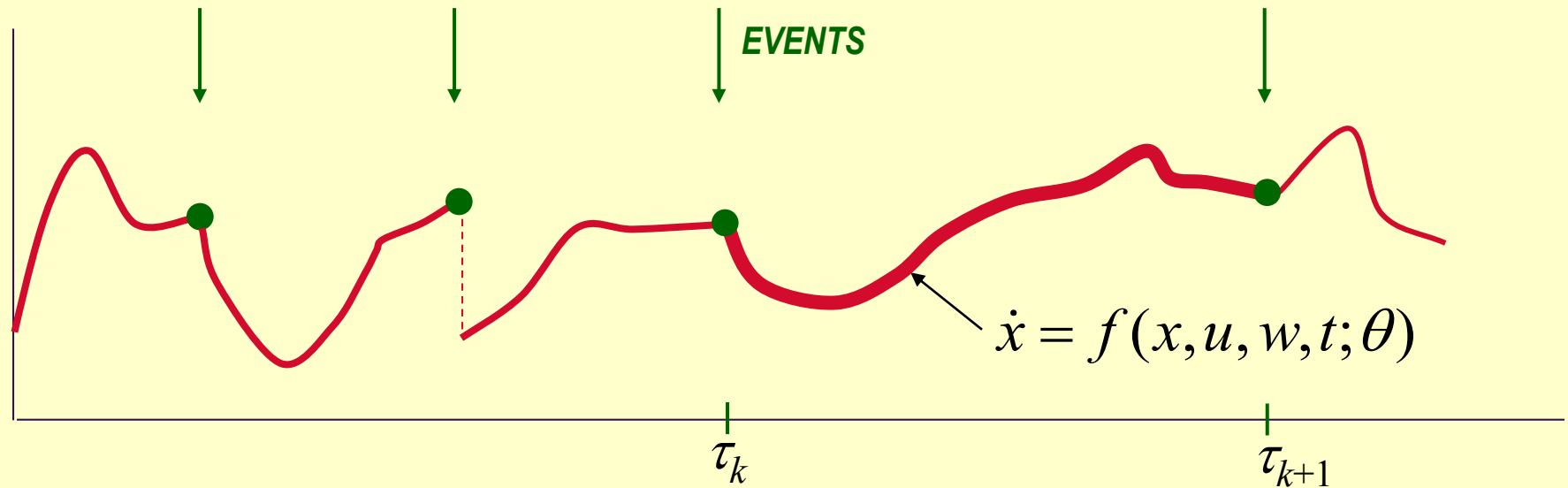
$$\begin{aligned} \min_{\theta, \phi} E \left[\int_0^T [x(t) - g(\phi)] dt \right] &\Rightarrow \frac{\partial L}{\partial x} = 1 \\ \text{s.t. } \dot{x}_k &= a_k x_k(t) + u_k(\theta_k) + w_k(t) \Rightarrow \frac{\partial f_k}{\partial x_k} = a_k, \quad \frac{\partial f_k}{\partial \theta_k} = \frac{du_k}{d\theta_k} \\ k &= 1, \dots, N \end{aligned}$$

NOTE: THEOREM 1 provides *sufficient* conditions only.
IPA still depends on info. limited to event times if

$$\begin{aligned} \dot{x}_k &= a_k x_k(t) + u_k(\theta_k, t) + w_k(t) \\ k &= 1, \dots, N \end{aligned}$$

for “nice” functions $u_k(\theta_k, t)$, e.g., $b_k \theta t$

IPA PROPERTY 1: **ROBUSTNESS**



Evaluating $x(t; \theta)$ requires full knowledge of w and f values (obvious)

However, $\frac{dx(t; \theta)}{d\theta}$ may be **independent** of w and f values (**NOT** obvious)

It often depends only on:

- event times τ_k
- possibly $f(\tau_{k+1}^-)$

IPA PROPERTY 2: **DECOMPOSABILITY**

THEOREM 2: Suppose an endogenous event occurs at τ_k with switching function $g(x, \theta)$.

If $f_k(\tau_k^+) = 0$, then $x'(\tau_k^+)$ is independent of f_{k-1} .

If, in addition, $\frac{dg}{d\theta} = 0$ then $x'(\tau_k^+) = 0$

IMPLICATION: Performance sensitivities are often reset to 0
 \Rightarrow sample path can be conveniently **decomposed**

IPA PROPERTY 3: **SCALABILITY**

IPA scales with the **EVENT SET**, not the STATE SPACE !

As a complex system grows with the addition of more states, the number of **EVENTS** often remains unchanged or increases at a much lower rate.

EXAMPLE: A queueing network may become very large, but the basic events used by IPA are still “arrival” and “departure” at different nodes.

IPA estimators are **EVENT-DRIVEN**

IPA PROPERTIES

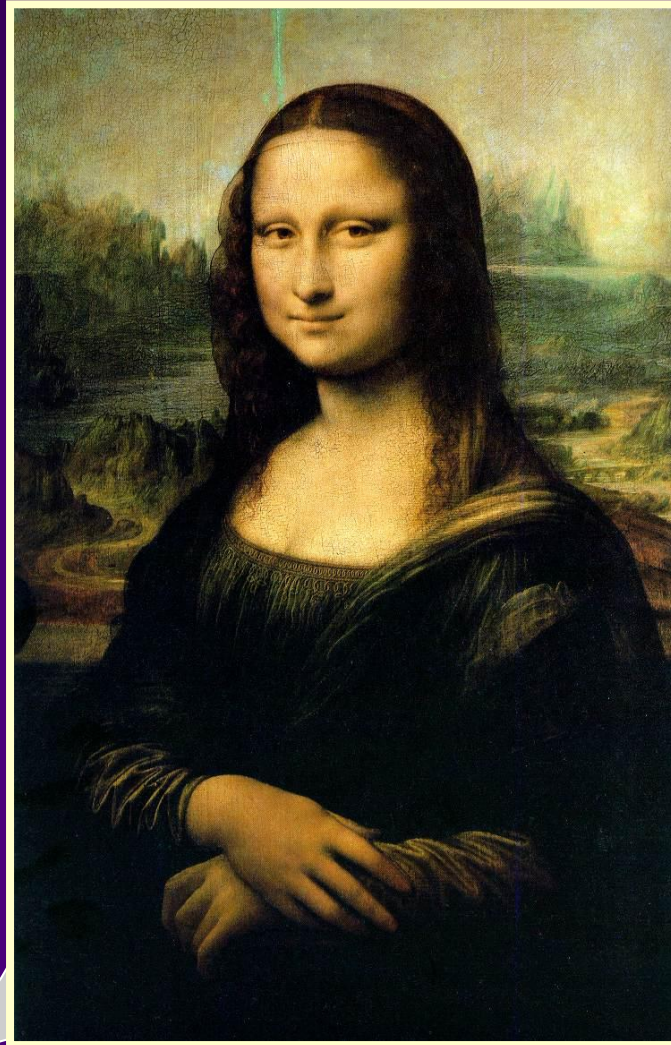
In many cases:

- **No need for a detailed model** (captured by f_k) to describe state behavior in between events
- This explains why **simple abstractions of a complex stochastic system** can be adequate to perform sensitivity analysis and optimization, as long as event times are accurately observed and local system behavior at these event times can also be measured.
- This is true in **abstractions of DES as HS** since:
Common performance metrics (e.g., workload) satisfy **THEOREM 1**

WHAT IS THE RIGHT ABSTRACTION LEVEL ?



TOO FAR...
model not
detailed enough



JUST RIGHT...
good model



TOO CLOSE...
too much
undesirable
detail

A SMART CITY CPS APPLICATION:

ADAPTIVE

TRAFFIC LIGHT CONTROL

TRAFFIC LIGHT CONTROL - BACKGROUND

A basic binary switching control (GREEN – RED) problem with a long history...

- Mixed Integer Linear Programming (MILP) [*Dujardin et al, 2011*]
- Extended Linear Complementarity Problem (ELCP) [*DeSchutter, 1999*]
- MDP and Reinforcement Learning [*Yu et al., 2006*]
- Game Theory [*Alvarez et al., 2010*]
- Evolutionary algorithms [*Taale et al., 1998*]
- Fuzzy Logic [*Murat et al., 2005*]
- Expert Systems [*Findler and Stapp, 1992*]
- **Perturbation Analysis**

TRAFFIC LIGHT CONTROL - BACKGROUND

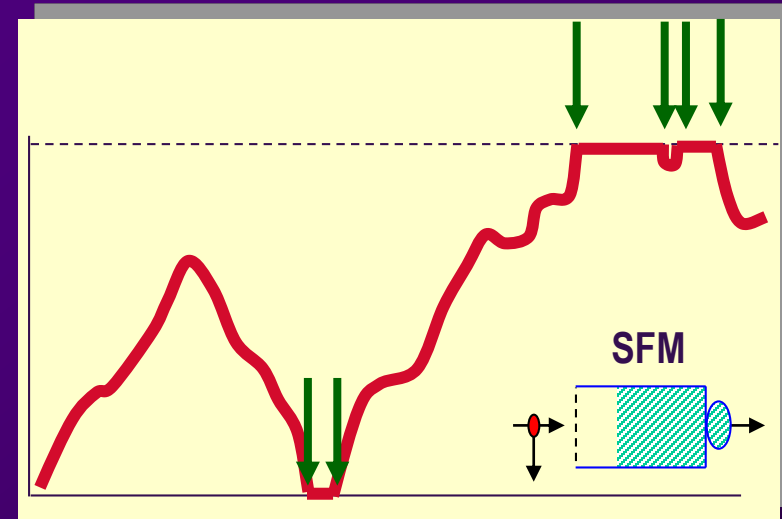
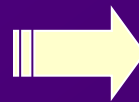
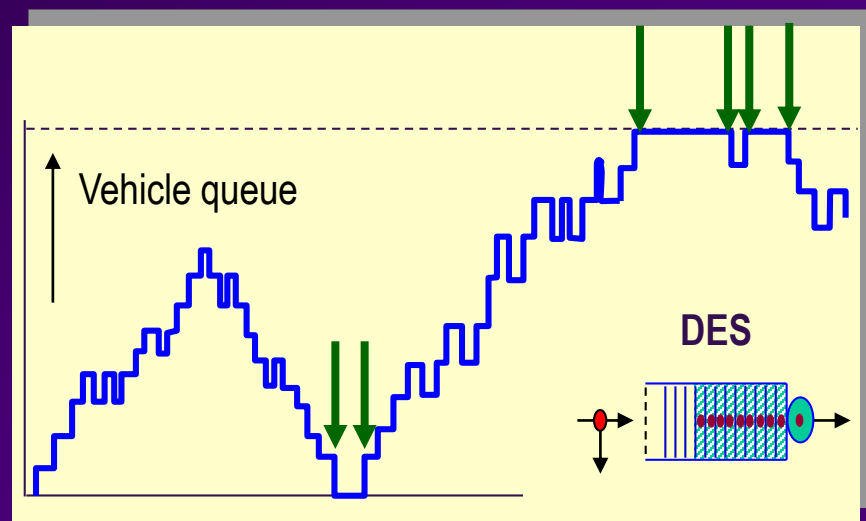
- **Perturbation Analysis** [Panayiotou et al., 2005]

[Geng and Cassandras, 2012]

Single
Intersection



Use a Hybrid System Model: **Stochastic Flow Model (SFM)**



Aggregate states into *modes* and keep only events causing mode transitions

SINGLE-INTERSECTION MODEL



Traffic light control:

$$\theta = [\theta_1, \theta_2, \theta_3, \theta_4]$$

GREEN light cycle
at queue $n = 1, 2, 3, 4$

OBJECTIVE:

Determine θ to minimize
total weighted vehicle queues

$$\min_{\theta} J_T(\theta) = \frac{1}{T} E \left[\sum_{n=1}^4 \int_0^T w_n x_n(\theta, t) dt \right]$$

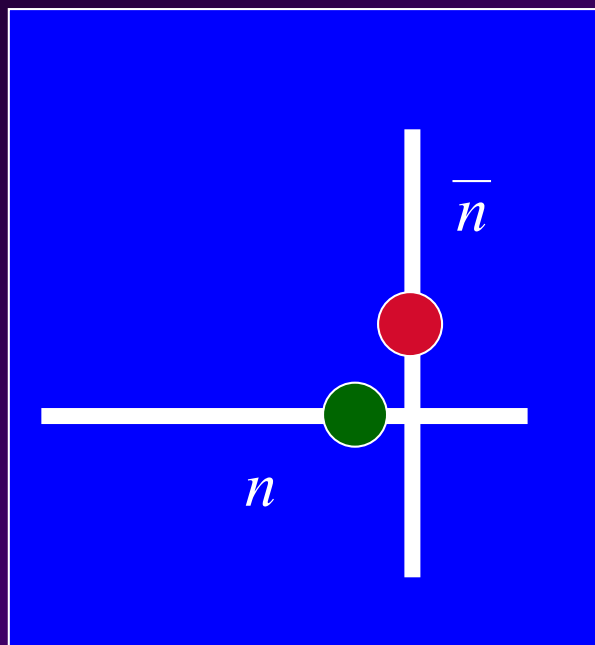
SINGLE-INTERSECTION MODEL

$$\min_{\theta} J_T(\theta) = \frac{1}{T} E \left[\sum_{n=1}^4 \int_0^T w_n x_n(\theta, t) dt \right] = \frac{1}{T} E[L_T(\theta)]$$

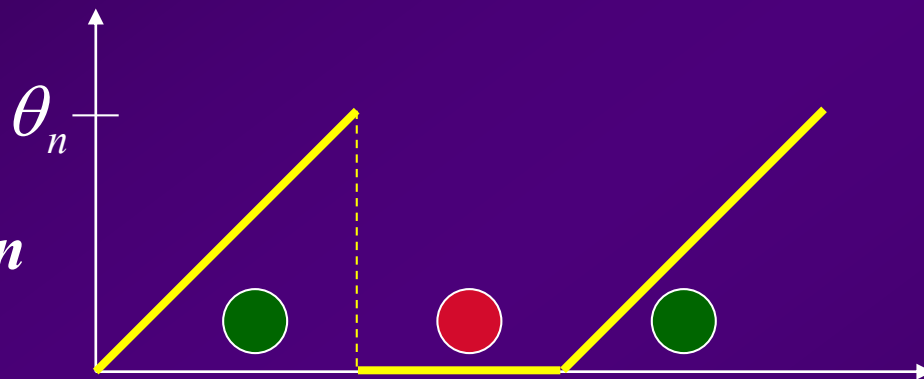
IPA APPROACH:

- Observe events and event times, estimate $\frac{dJ_T(\theta)}{d\theta}$ through $\frac{dL_T(\theta)}{d\theta}$
- Then, $\theta_{n+1} = \theta_n + \eta_n \frac{dL_T(\theta_n)}{d\theta}$

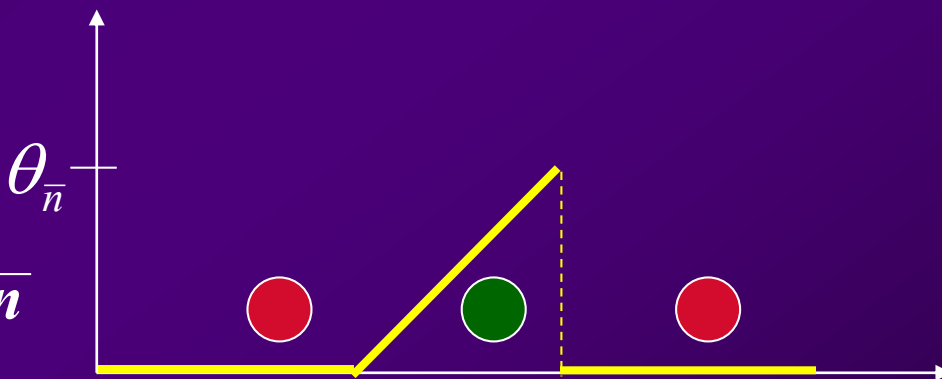
HYBRID SYSTEM STATE DYNAMICS



GREEN n



GREEN \bar{n}



$$\dot{z}_n(t) = \begin{cases} 1 & \text{if } 0 < z_n(t) < \theta_n \text{ or } z_{\bar{n}}(t) = \theta_{\bar{n}} \\ 0 & \text{otherwise} \end{cases}$$

GREEN light “clock”

$z_n(t^+) = 0$ if $z_n(t) = \theta_n$ → Control: GREEN light cycle

HYBRID SYSTEM STATE DYNAMICS

$$\dot{z}_n(t) = \begin{cases} 1 & \text{if } 0 < z_n(t) < \theta_n \text{ or } z_{\bar{n}}(t) = \theta_{\bar{n}} \\ 0 & \text{otherwise} \end{cases}$$

$$z_n(t^+) = 0 \text{ if } z_n(t) = \theta_n$$

[RESOURCE DYNAMICS]

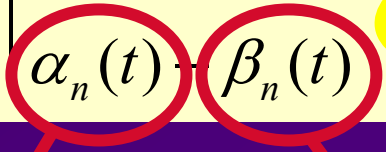
Define: $G_n(t) = \begin{cases} 1 & \text{if } 0 < z_n(t) < \theta_n \text{ or } z_{\bar{n}}(t) = \theta_{\bar{n}} \\ 0 & \text{otherwise} \end{cases}$

GREEN + queue n

$$\dot{x}_n(t) = \begin{cases} \alpha_n(t) \\ 0 \\ \alpha_n(t) \cdot \beta_n(t) \end{cases}$$

IPA ROBUSTNESS:
 $\alpha_n(t), \beta_n(t)$ DO NOT HAVE TO BE KNOWN!

[USER DYNAMICS]



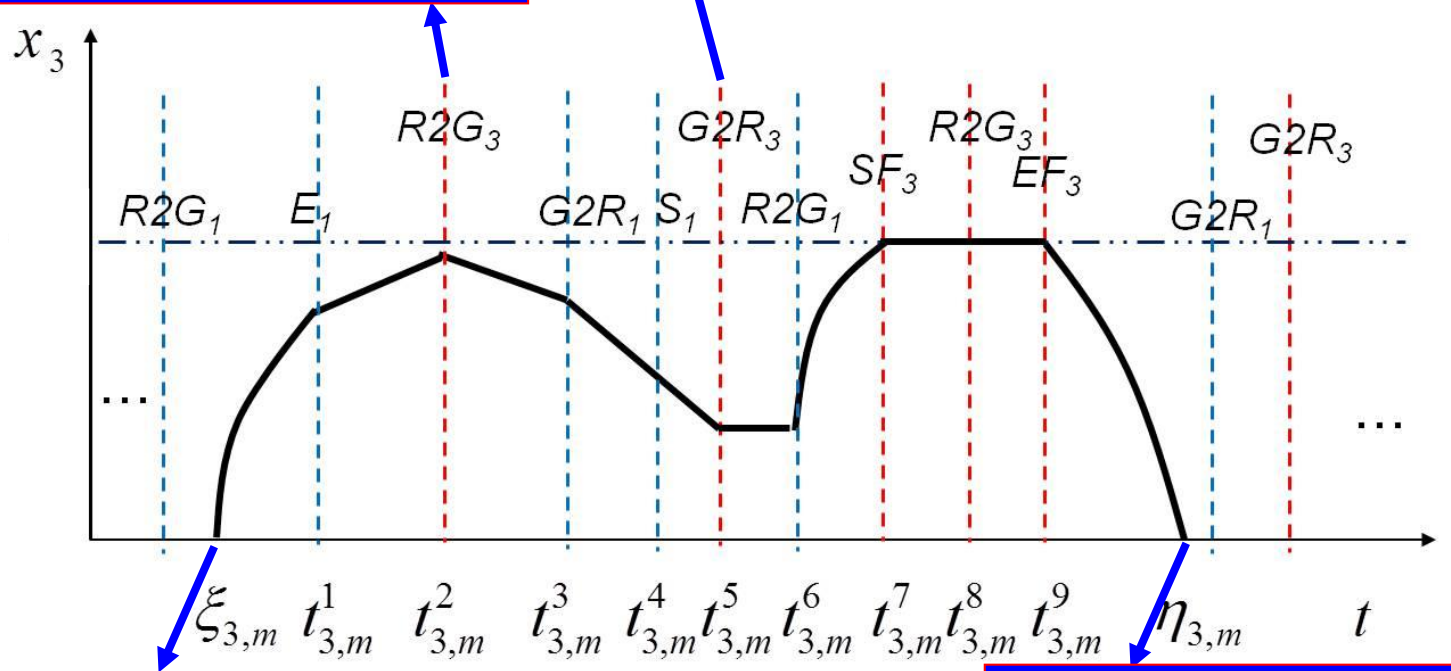
Vehicle departure rate process

Vehicle arrival rate process

EVENTS IN THE TLC MODEL

Event **R2G**
 RED light switches to GREEN
endogenous

Event **G2R**
 GREEN light switches to RED
endogenous



Event **S**
 Non-Empty-Period (NEP) starts
endogenous or exogenous

Event **E**
 Non-Empty-Period (NEP) ends
endogenous

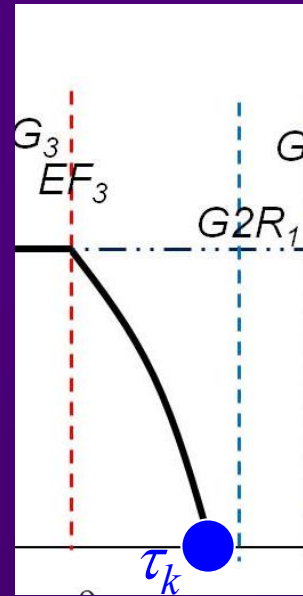
APPLY IPA EQUATIONS FOR θ AND s VECTORS

FOR EXAMPLE: Endogenous event with

$$g_k(x(\theta, \tau_k), \theta) = x_n(\theta, t) = 0$$

$$\tau'_k = - \left[\frac{\partial g}{\partial x} f_k(\tau_k^-) \right]^{-1} \left(\frac{\partial g}{\partial \theta} + \frac{\partial g}{\partial x} x'(\tau_k^-) \right)$$

$$\tau'_{k,i} = \frac{-x'_{n,i}(\tau_k^-)}{\alpha_n(\tau_k) - \beta_n(\tau_k)}$$



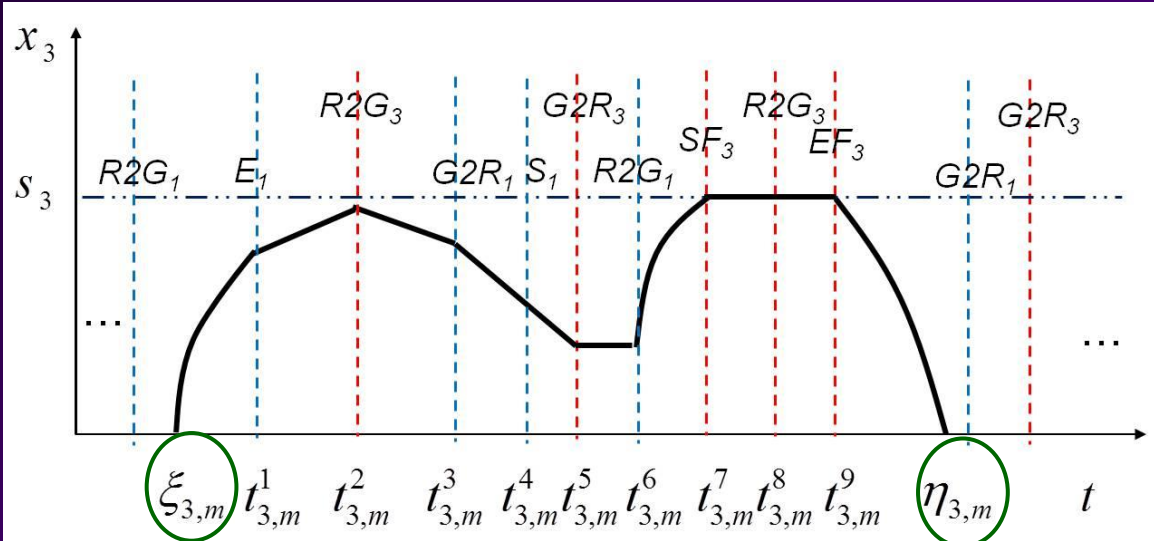
$$x'(\tau_k^+) = x'(\tau_k^-) + [f_{k-1}(\tau_k^-) - f_k(\tau_k^+)] \tau'_k$$

$$x'_{n,i}(\tau_k^+) = x'_{n,i}(\tau_k^-) - \frac{[\alpha_n(\tau_k) - \beta_n(\tau_k)] x'_{n,i}(\tau_k^-)}{\alpha_n(\tau_k) - \beta_n(\tau_k)}$$

$$= 0$$

Perturbation in queue n
RESET to 0 when NEP ends

COST DERIVATIVE IN m th NEP



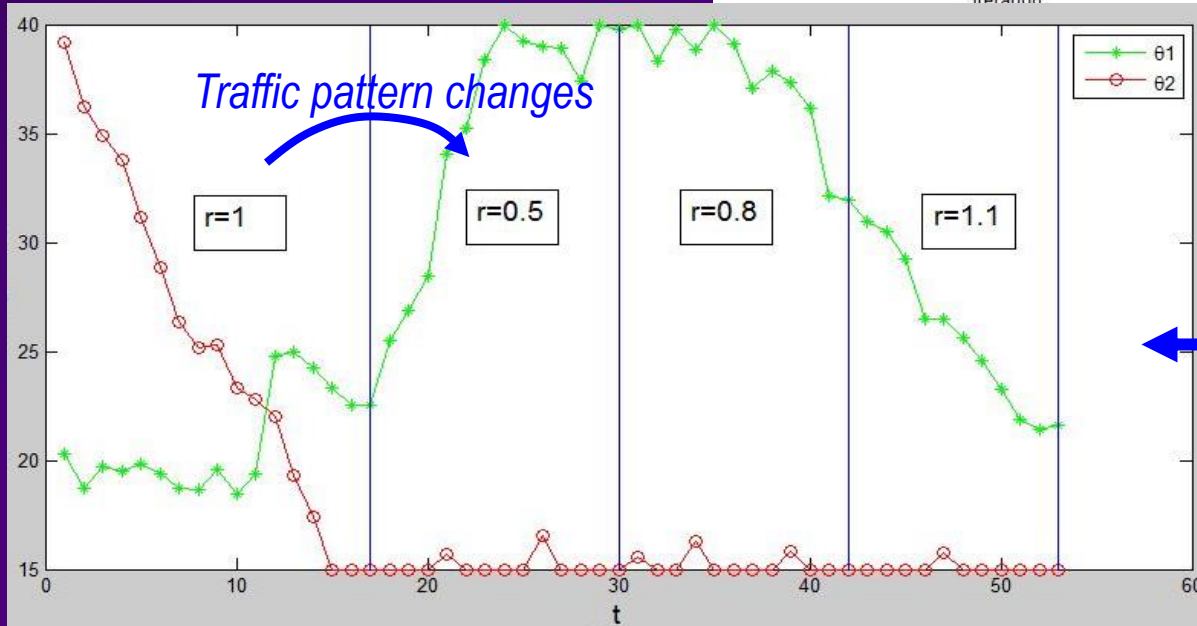
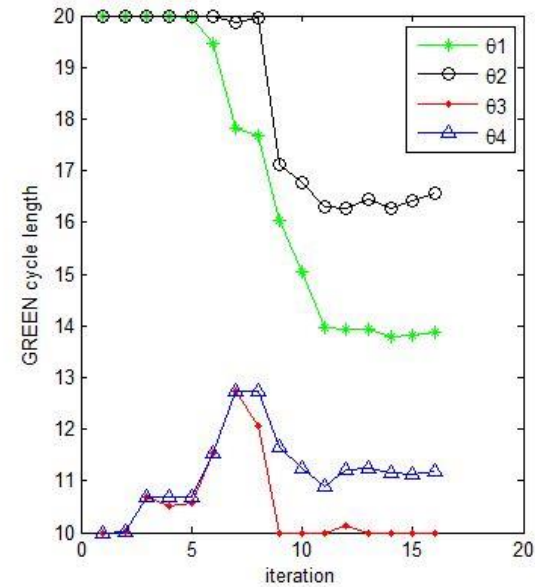
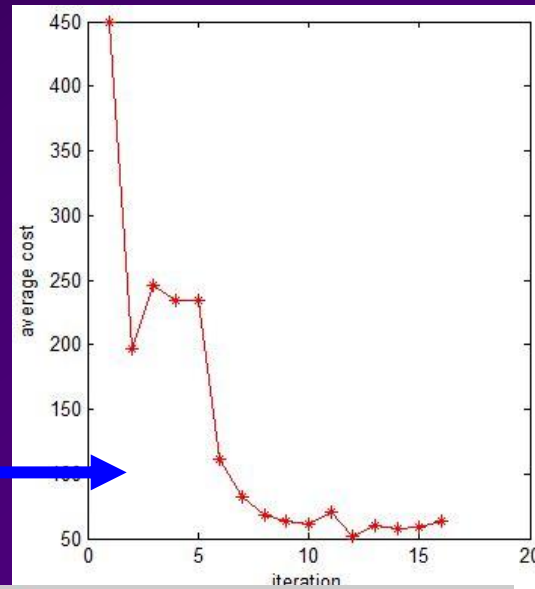
$$L_{n,m} = \int_{\xi_{n,m}(\theta)}^{\eta_{n,m}(\theta)} x_n(\theta, t) dt$$

$$\begin{aligned} \frac{dL_{n,m}(\theta, s)}{d\theta_i} &= x'_{n,i}((\xi_{n,m})^+) \cdot (t_{n,m}^1 - \xi_{n,m}) \\ &+ x'_{n,i}((t_{n,m}^{J_{n,m}})^+) \cdot (\eta_{n,m} - t_{n,m}^{J_{n,m}}) \\ &+ \sum_{j=2}^{J_{n,m}} x'_{n,i}((t_{n,m}^j)^+) \cdot (t_{n,m}^j - t_{n,m}^{j-1}) \end{aligned}$$

NOTES: - Need only TIMERS, COUNTERS and state derivatives
 - Scalable in number of EVENTS – not states!

TYPICAL SIMULATION RESULTS

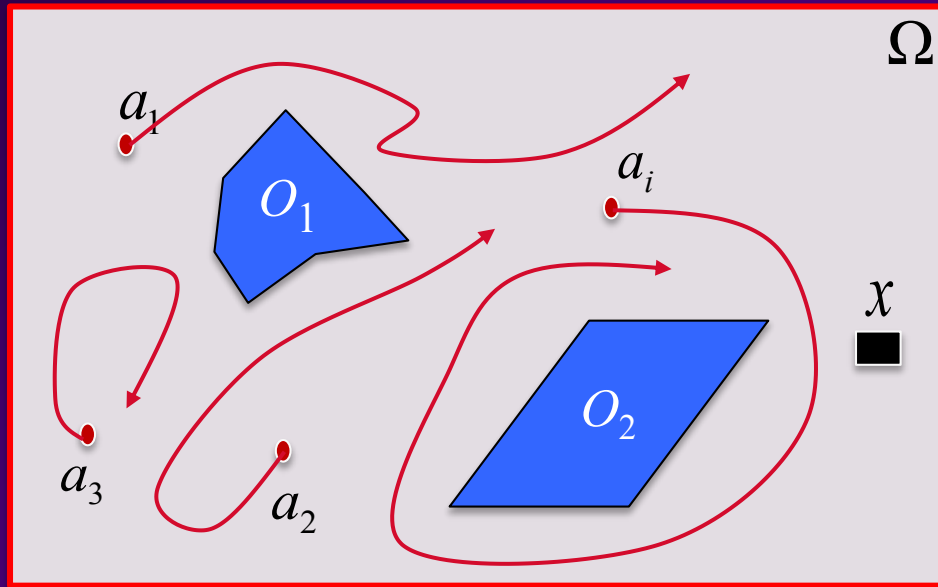
9-fold cost reduction



Adaptivity

**IT IS HARD TO
DECENTRALIZE
PROBLEM 2 ...**

MULTI-AGENT OPTIMIZATION: PROBLEM 2



$$\max_{\mathbf{u}(t)} J = \int_0^T \int_{\Omega} P(x, \mathbf{s}(u(t))) R(x) dx dt$$

May also have dynamics

$$s_i(t) \in F \subseteq \Omega, \quad i = 1, \dots, N$$

$$\dot{s}_i = f_i(s_i, u_i, t), \quad i = 1, \dots, N$$

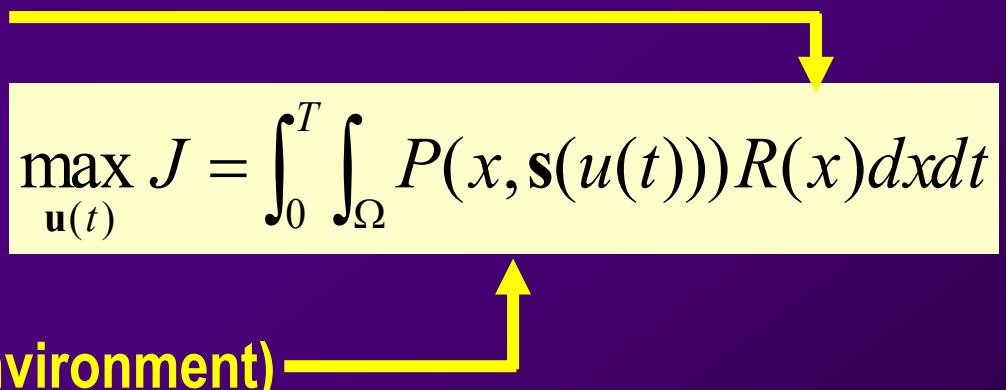
GOAL: Find the best **state trajectories** $s_i(t)$, $0 \leq t \leq T$ so that agents achieve a maximal **reward** from interacting with the mission space

PERSISTENT MONITORING PROBLEM

GOAL: Find the best **state trajectories** $s_i(t)$, $0 \leq t \leq T$ so that agents achieve a maximal **reward** from interacting with the mission space

Need three model elements:

1. ENVIRONMENT MODEL


$$\max_{\mathbf{u}(t)} J = \int_0^T \int_{\Omega} P(x, \mathbf{s}(u(t))) R(x) dx dt$$

2. SENSING MODEL

(how agents interact with environment)

3. AGENT MODEL

$$\dot{s}_i = f_i(s_i, u_i, t), \quad i = 1, \dots, N$$

PERSISTENT MONITORING PROBLEM

Start with 1-dimensional mission space $\Omega = [0, L]$

AGENT DYNAMICS:

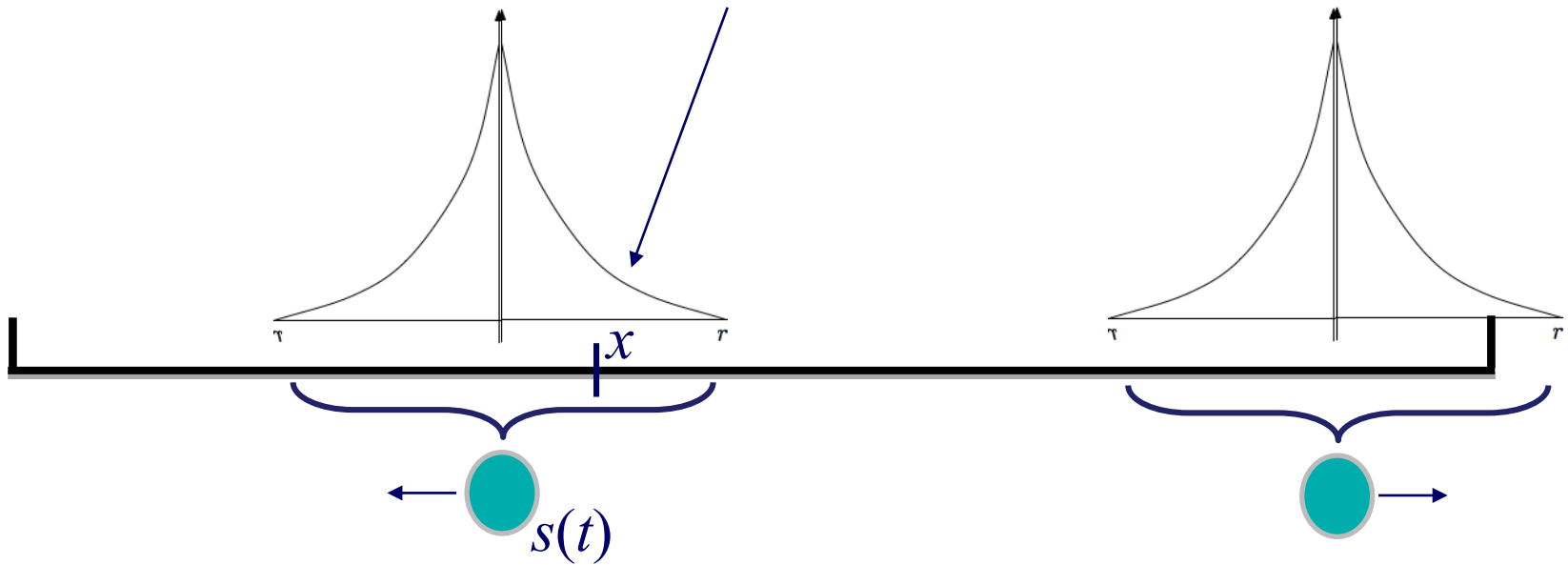
$$\dot{s}_j = u_j, \quad |u_j(t)| \leq 1$$

Analysis still holds for:

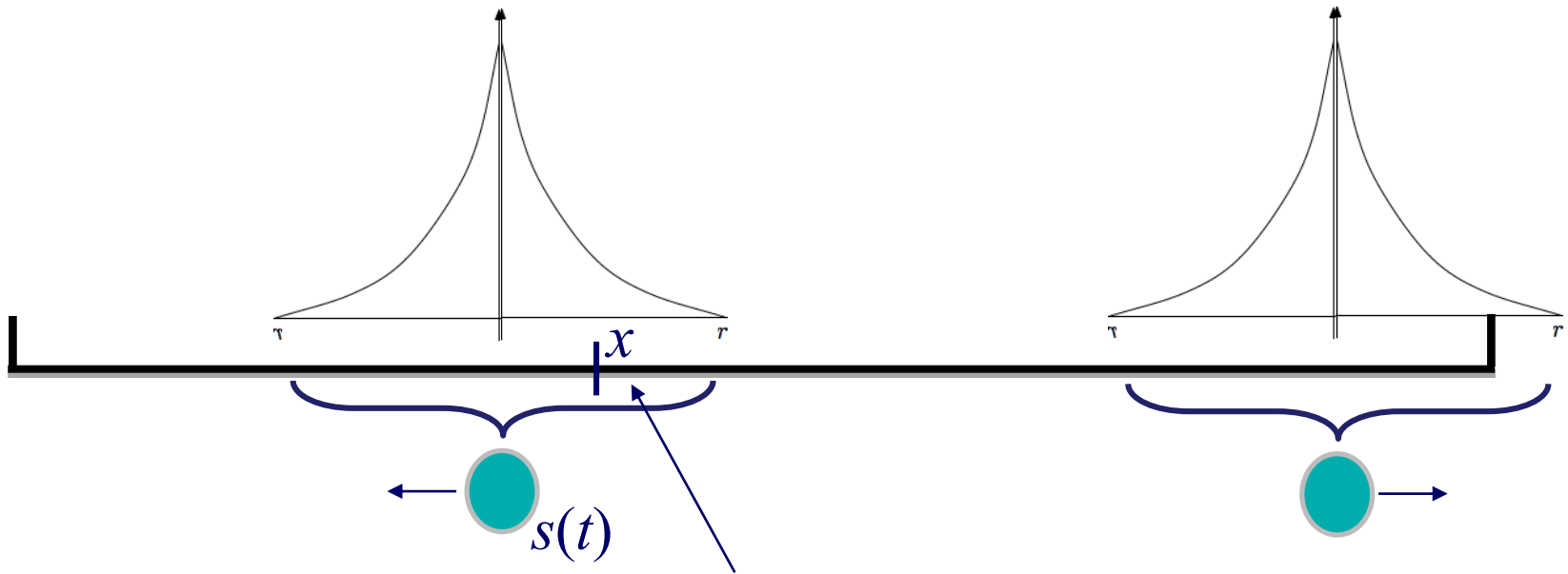
$$\dot{s}_j = g_j(s_j) + bu_j, \quad |u_j(t)| \leq 1$$

PERSISTENT MONITORING PROBLEM

SENSING MODEL: $p(x,s)$ Probability agent at s senses point x



PERSISTENT MONITORING PROBLEM



ENVIRONMENT MODEL: Associate to x *Uncertainty Function* $R(x,t)$

Use:

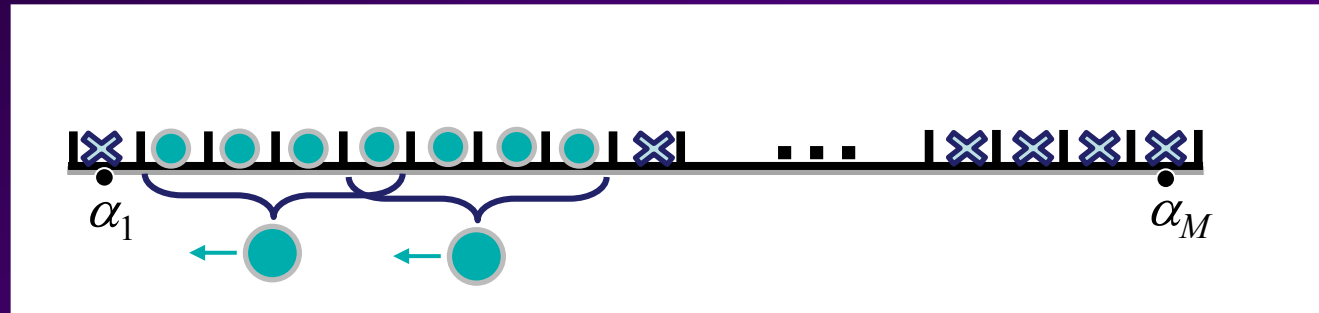
$$\dot{R}(x,t) = \begin{cases} 0 & \text{if } R(x,t) = 0, A(x) < Bp(x,s(t)) \\ A(x) - Bp(x,s(t)) & \text{otherwise} \end{cases}$$

If x is a known “target”:

$$\dot{R}_x(t) = f_x(R,s,t) + \textit{noise}$$

PERSISTENT MONITORING PROBLEM

Partition mission space $\Omega = [0, L]$ into M intervals:



For each interval $i = 1, \dots, M$ define **Uncertainty Function** $R_i(t)$:

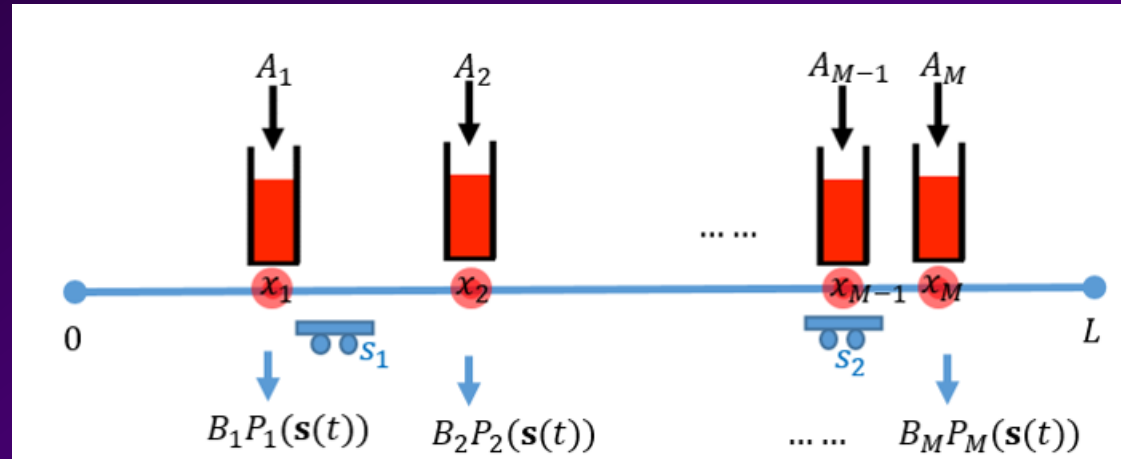
$$\dot{R}_i(t) = \begin{cases} 0 & \text{if } R_i(t) = 0, A_i < BP_i(\mathbf{s}(t)) \\ A_i - BP_i(\mathbf{s}(t)) & \text{otherwise} \end{cases}$$

$$P_i(\mathbf{s}) = 1 - \prod_{j=1}^N [1 - p_i(s_j)]$$

$$p_i(s_j) \equiv p_j(\alpha_i, s_j)$$

where $P_i(\mathbf{s}) =$ joint prob. i is sensed by agents located at $\mathbf{s} = [s_1, \dots, s_N]$

PERSISTENT MONITORING (PM) WITH KNOWN TARGETS



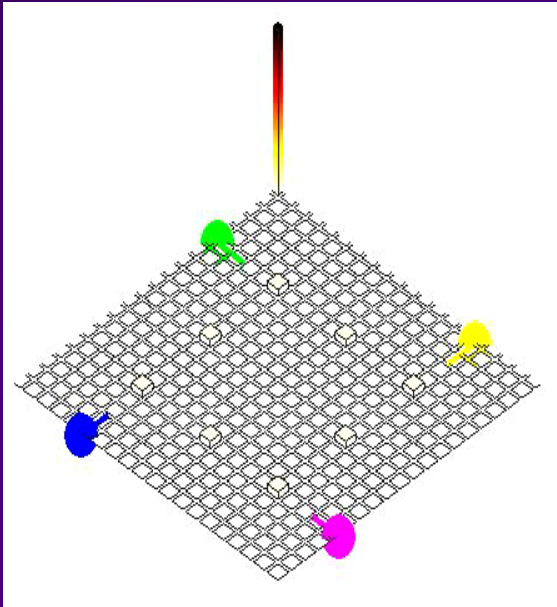
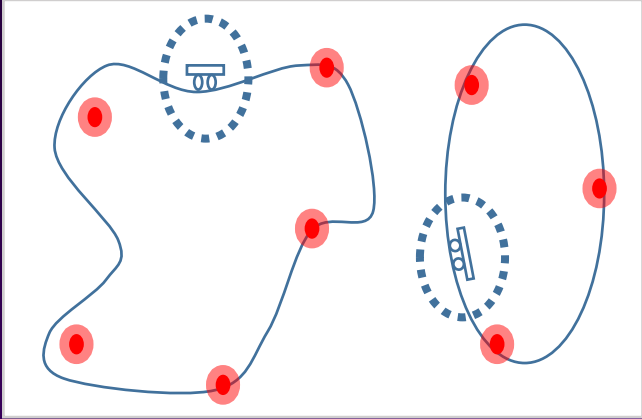
$$\min_{u_1, \dots, u_N} J = \frac{1}{T} \int_0^T \sum_{i=1}^M R_i(t) dt$$

s.t.

$$\dot{s}_j = u_j, \quad |u_j(t)| \leq 1, \quad 0 < a \leq s_j(t) \leq b < L$$

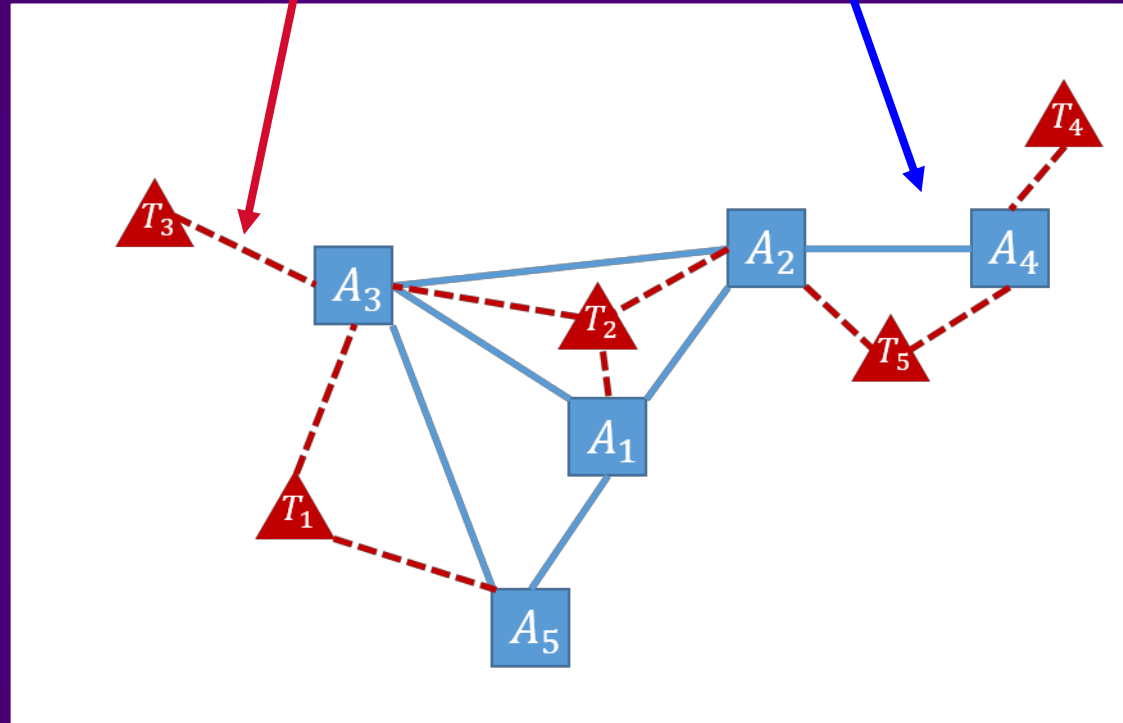
$$\dot{R}_i(t) = \begin{cases} 0 & \text{if } R_i(t) = 0, A_i < B P_i(\mathbf{s}(t)) \\ A_i - B P_i(\mathbf{s}(t)) & \text{otherwise} \end{cases}$$

PERSISTENT MONITORING WITH KNOWN TARGETS



**Agent-Target
Interaction Network
(time-varying)**

**Agent Network
(time-varying)**



**Hard to decentralize a controller that involves
time-varying agent-environment interactions**

THREE TYPES OF NEIGHBORHOODS

The agent neighborhood of an agent (conventional)

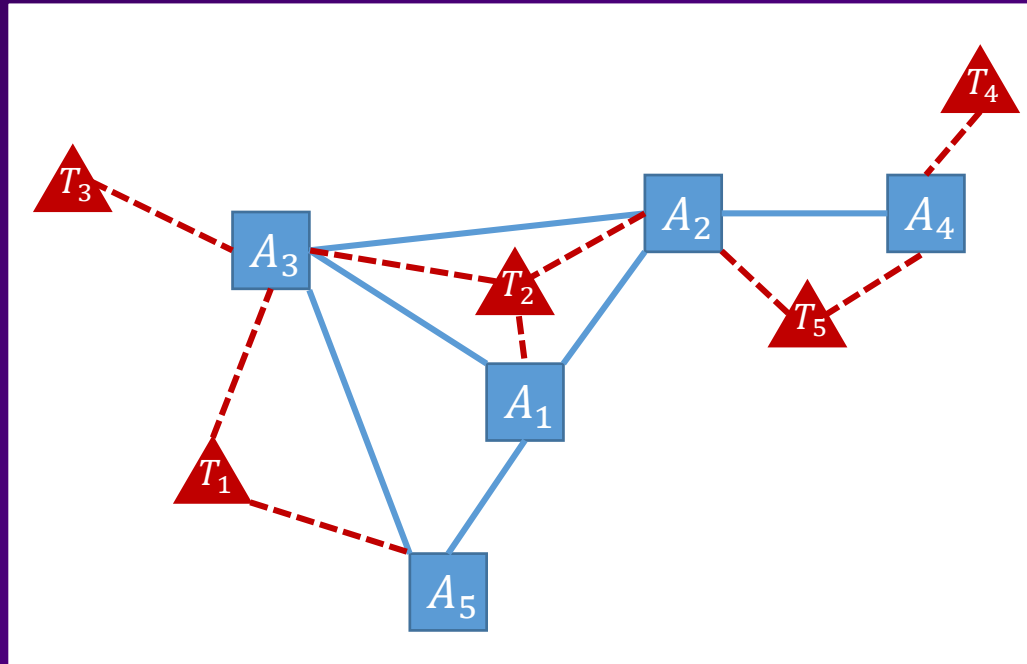
The agent neighborhood of agent j is the set

$$\mathcal{A}_j(t) = \{k : \|s_k(t) - s_j(t)\| \leq r_c, k \neq j, k = 1, \dots, N\}.$$

The target neighborhood of an agent

The target neighborhood of agent j is the set

$$\mathcal{T}_j(t) = \{i : |x_i - s_j(t)| \leq r_j, i = 1, \dots, M\}.$$



The agent neighborhood of an target

The agent neighborhood of target i is the set

$$\mathcal{B}_i(t) = \{j : |s_j(t) - x_i| \leq r_j, j = 1, \dots, N\}.$$

PM WITH KNOWN TARGETS – 1D CASE

We have shown that:

1. Optimal Trajectories are bounded:

$$x_1 \leq s_j^*(t) < x_M \quad j = 1, \dots, N$$

2. Existence of finite dwell times at target on optimal trajectories:

Under certain conditions: $s_j^*(t) = x_k$ and $u_j^*(t) = 0$ for $t \in [t_1, t_2]$

3. Under the constraint $s_j(t) < s_{j+1}(t)$, on an optimal trajectory:

$$s_j(t) \neq s_{j+1}(t)$$

Zhou et al, IEEE CDC, 2016

OPTIMAL CONTROL SOLUTION

Optimal trajectory is fully characterized by TWO parameter vectors:

$$\theta_j = [\theta_{j1} \cdots \theta_{jS}] \quad j = 1, \dots, N$$

$$w_j = [w_{j1} \cdots w_{jS}] \quad j = 1, \dots, N$$

Switching points

Waiting times at
switching points, $w_{jk} \geq 0$

$$J(\boldsymbol{\theta}, \mathbf{w}) = \frac{1}{T} \sum_{k=0}^K \int_{\tau_k(\boldsymbol{\theta}, \mathbf{w})}^{\tau_{k+1}(\boldsymbol{\theta}, \mathbf{w})} \sum_{i=1}^M R_i(t) dt$$

τ_k : k th event time

⇒ Under optimal control, this is a HYBRID SYSTEM

HYBRID SYSTEM EVENTS

Type 1:
switches in $\dot{R}_i(t)$

Type 2:
switches in
agent sensing

Type 3:
switches in
 $\dot{s}_j(t)$

Type 4:
changes in
neighbor sets

TABLE I: Events in agent-target system

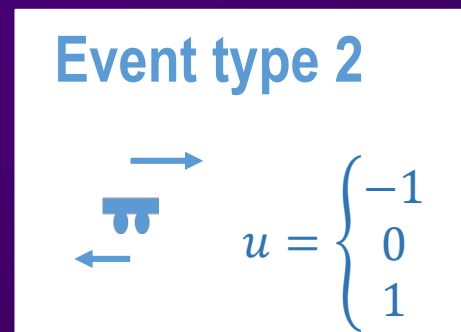
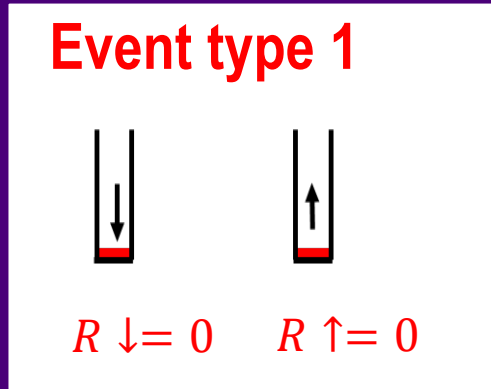
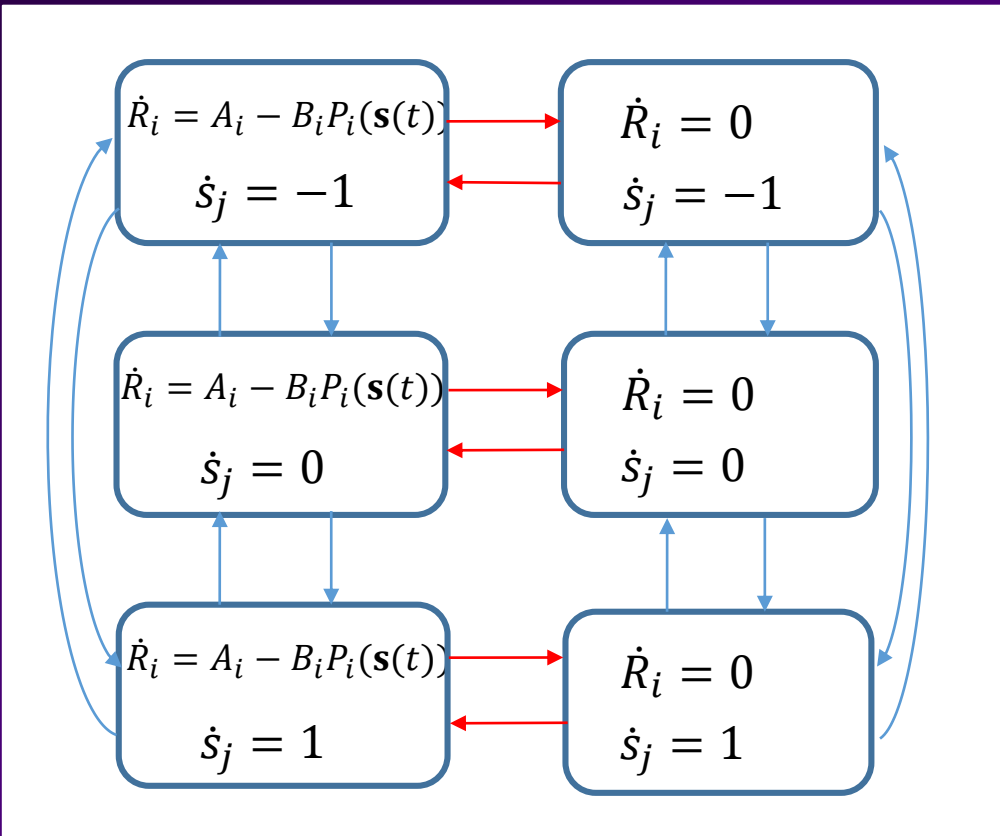
Event Name	Description
ρ_i^0	$R_i(t)$ hits 0
ρ_i^+	$R_i(t)$ leaves 0
π_{ij}^0	$p_{ij}(s_j(t))$ hits 0
π_{ij}^+	$p_{ij}(s_j(t))$ leaves 0
$\nu_j^{(1,0)}$	$u_j(t)$ switches from 1 to 0
$\nu_j^{(-1,0)}$	$u_j(t)$ switches from -1 to 0
$\nu_j^{(0,1)}$	$u_j(t)$ switches from 0 to 1
$\nu_j^{(0,-1)}$	$u_j(t)$ switches from 0 to -1
$\nu_j^{(1,-1)}$	$u_j(t)$ switches from 1 to -1
$\nu_j^{(-1,1)}$	$u_j(t)$ switches from -1 to 1
Δ_{ij}^+	$\mathcal{N}_{ij}(\tau^+) = \mathcal{N}_{ij}(\tau^-) \cup \{k\}, k \notin \mathcal{N}_{ij}(\tau^-)$
Δ_{ij}^-	$\mathcal{N}_{ij}(\tau^+) = \mathcal{N}_{ij}(\tau^-) \setminus \{k\}, k \in \mathcal{N}_{ij}(\tau^-)$

Note: events in the table include all $i = 1, \dots, M$ and $j = 1, \dots, N$

HYBRID SYSTEM EVENTS: EXAMPLE

$$\dot{R}_i(t) = \begin{cases} 0 & \text{if } R_i(t) = 0, A_i < BP_i(\mathbf{s}(t)) \\ A_i - BP_i(\mathbf{s}(t)) & \text{otherwise} \end{cases}$$

A simple example: 1 agent 1 target



$$\dot{s}_j = u_j, \quad |u_j(t)| \leq 1$$

IPA GRADIENTS

Objective function gradient:

$$\nabla J(\boldsymbol{\theta}, \mathbf{w}) = \frac{1}{T} \sum_{i=1}^M \sum_{k=0}^K \int_{\tau_k(\boldsymbol{\theta}, \mathbf{w})}^{\tau_{k+1}(\boldsymbol{\theta}, \mathbf{w})} \nabla R_i(t) dt$$

$$\nabla R_i(t) = \begin{bmatrix} \frac{\partial R_i(t)}{\partial \boldsymbol{\theta}} & \frac{\partial R_i(t)}{\partial \mathbf{w}} \end{bmatrix}^T$$

where $\nabla R_i(t)$ is obtained using the **IPA Calculus**

$$1. \quad x'(\tau_k^+) = x'(\tau_k^-) + [f_{k-1}(\tau_k^-) - f_k(\tau_k^+)] \cdot \tau'_k$$

$$2. \quad x'(t) = e^{\int_{\tau_k}^t \frac{\partial f_k(u)}{\partial x} du} \left[\int_{\tau_k}^t \frac{\partial f_k(v)}{\partial \theta} e^{-\int_{\tau_k}^v \frac{\partial f_k(u)}{\partial x} du} dv + x'(\tau_k^+) \right]$$

$$3. \quad \tau'_k = 0 \quad \text{or} \quad \tau'_k = - \left[\frac{\partial g}{\partial x} f_k(\tau_k^-) \right]^{-1} \left(\frac{\partial g}{\partial \theta} + \frac{\partial g}{\partial x} x'(\tau_k^-) \right)$$

← $\nabla R_i(t)$ is updated on an
EVENT-DRIVEN basis
 τ_k : k th event time

AGENT AND TARGET EVENTS

TABLE I: Events in agent-target system

Event Name	Description
ρ_i^0	$R_i(t)$ hits 0
ρ_i^+	$R_i(t)$ leaves 0
π_{ij}^0	$p_{ij}(s_j(t))$ hits 0
π_{ij}^+	$p_{ij}(s_j(t))$ leaves 0
$\nu_j^{(1,0)}$	$u_j(t)$ switches from 1 to 0
$\nu_j^{(-1,0)}$	$u_j(t)$ switches from -1 to 0
$\nu_j^{(0,1)}$	$u_j(t)$ switches from 0 to 1
$\nu_j^{(0,-1)}$	$u_j(t)$ switches from 0 to -1
$\nu_j^{(1,-1)}$	$u_j(t)$ switches from 1 to -1
$\nu_j^{(-1,1)}$	$u_j(t)$ switches from -1 to 1
Δ_{ij}^+	$\mathcal{N}_{ij}(\tau^+) = \mathcal{N}_{ij}(\tau^-) \cup \{k\}, k \notin \mathcal{N}_{ij}(\tau^-)$
Δ_{ij}^-	$\mathcal{N}_{ij}(\tau^+) = \mathcal{N}_{ij}(\tau^-) \setminus \{k\}, k \in \mathcal{N}_{ij}(\tau^-)$

AGENT
Event Set \mathcal{E}^A

TARGET
Event Set \mathcal{E}^T

Note: events in the table include all $i = 1, \dots, M$ and $j = 1, \dots, N$

LOCAL EVENT SETS FOR AGENTS

\mathcal{E}_j^A : Subset of \mathcal{E}^A that contains only events related to agent j

\mathcal{E}_j^T : Subset of \mathcal{E}^T that contains only events related to agent j

Definition

The local event set of any agent j is the union of agent events \mathcal{E}_j^A and target events \mathcal{E}_i^T for all $i \in \mathcal{T}_j(t)$:

$$\mathcal{E}_j(t) = \mathcal{E}_j^A \cup \bigcup_{i \in \mathcal{T}_j(t)} \mathcal{E}_i^T$$

HOW CAN WE DECENTRALIZE ?

DECENTRALIZATION: Each agent should be able to evaluate

$$\nabla J(\boldsymbol{\theta}, \mathbf{w}) = \frac{1}{T} \sum_{i=1}^M \sum_{k=0}^K \int_{\tau_k(\boldsymbol{\theta}, \mathbf{w})}^{\tau_{k+1}(\boldsymbol{\theta}, \mathbf{w})} \nabla R_i(t) dt$$

...based only on LOCAL events (i.e., events it can observe)

Can this gradient be evaluated by every agent j using ONLY local events in $\mathcal{E}_j(t)$?

“ALMOST DECENTRALIZATION”

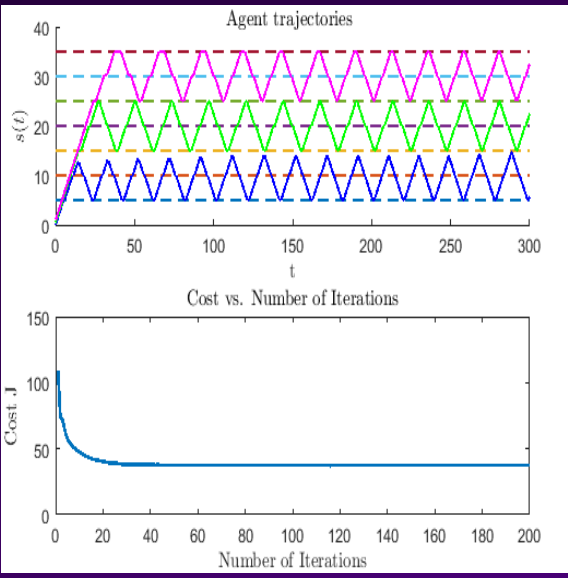
THEOREM: Any centralized solution of the trajectory optimization problem can be recovered through

$$\left[\theta_j^{l+1}, \mathbf{w}_j^{l+1}\right]^T = \left[\theta_j^l, \mathbf{w}_j^l\right]^T - [\alpha_\theta^l, \alpha_w^l] \nabla_j J(\bar{\theta}, \bar{\mathbf{w}})$$

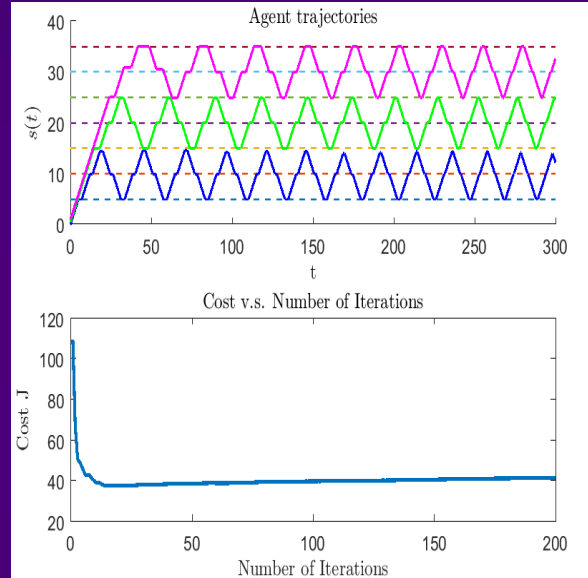
In which each agent j optimizes its trajectory under the following conditions:

1. Initial trajectory parameters $(\theta_j^0, \mathbf{w}_j^0)$
2. The LOCAL information set $I_j(t) = \mathcal{E}_j(t) \cup_{k \in \mathcal{N}_{i_j}(t), i \in \mathcal{T}_j(t)} \mathcal{E}_k(t)$.
3. The subset of the GLOBAL information set $\{\rho_i^0, i \notin \mathcal{T}_j(t)\}$

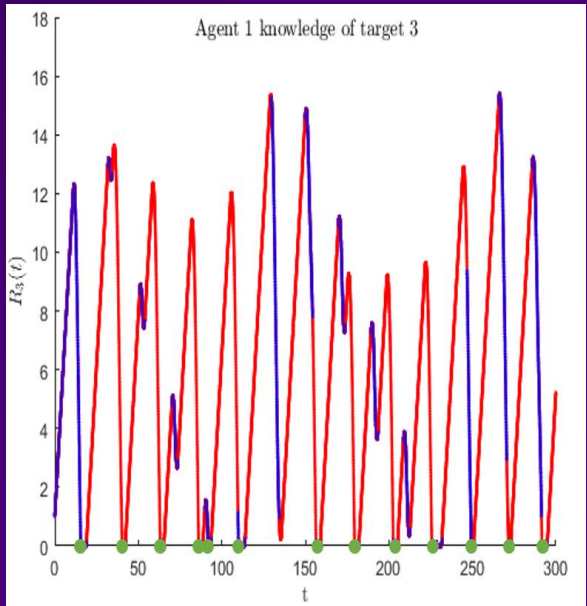
“ALMOST DECENTRALIZATION”: EXAMPLE



“Almost decentralized” solution, $J^* = 37.38$



Fully decentralized solution (ignoring non-local events), $J^* = 41.66$



Red: true state of target 3
Blue: state of target 3 observed by agent 1 when in its neighborhood
Green dots: instants when agent 1 receives non-local events

NOTE: Agent does NOT need to reconstruct the full target state!

CONTROL AND OPTIMIZATION – CHALLENGES

1. **SCALABILITY**

2. **DECENTRALIZATION**



Distributed Algorithms

3. **COMMUNICATION**



**Event-driven (asynchronous)
Algorithms**

4. **NON-CONVEXITY**



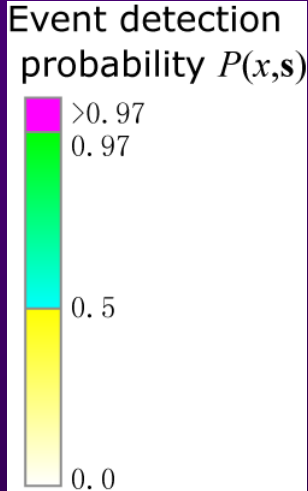
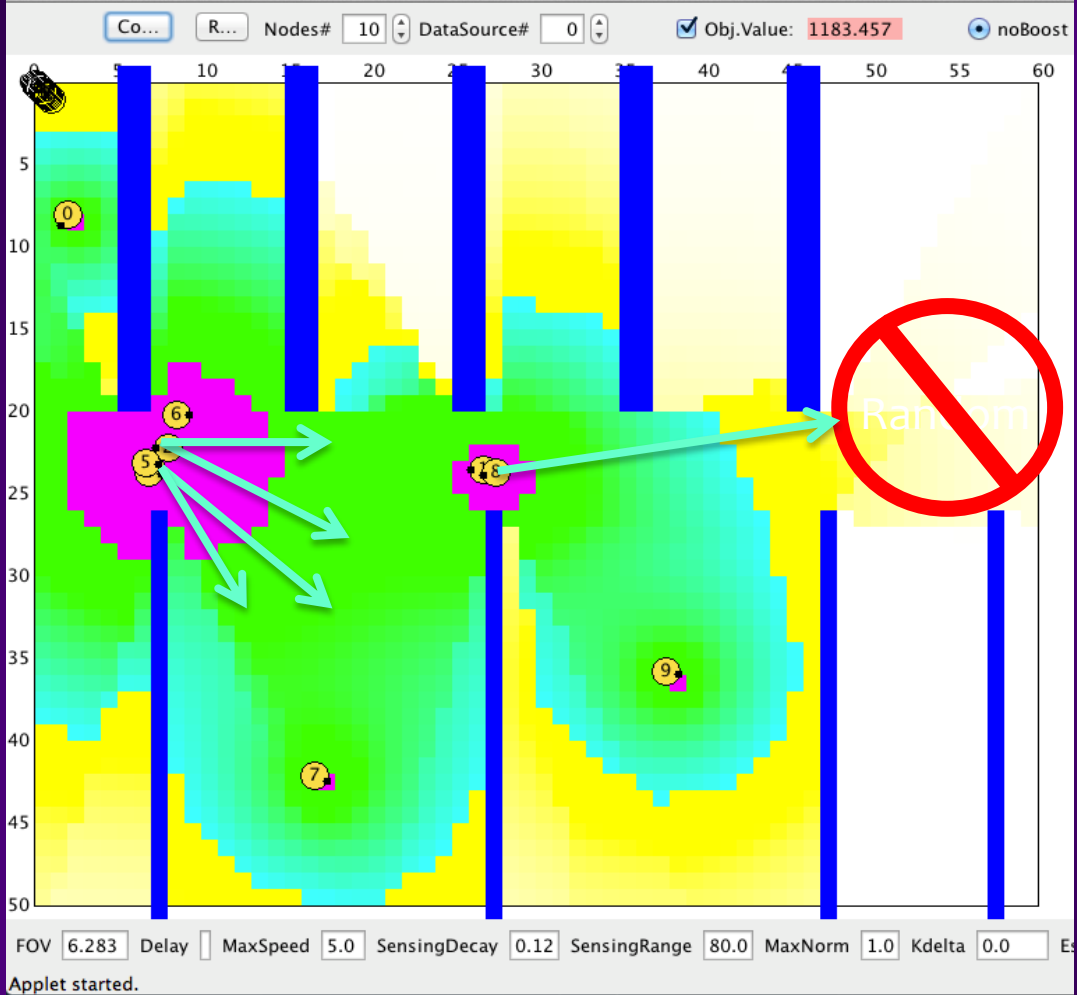
**Global optimality,
escape local optima**

5. **EXLOIT DATA**



Data-Driven Algorithms

LOCAL OPTIMUM EXAMPLE



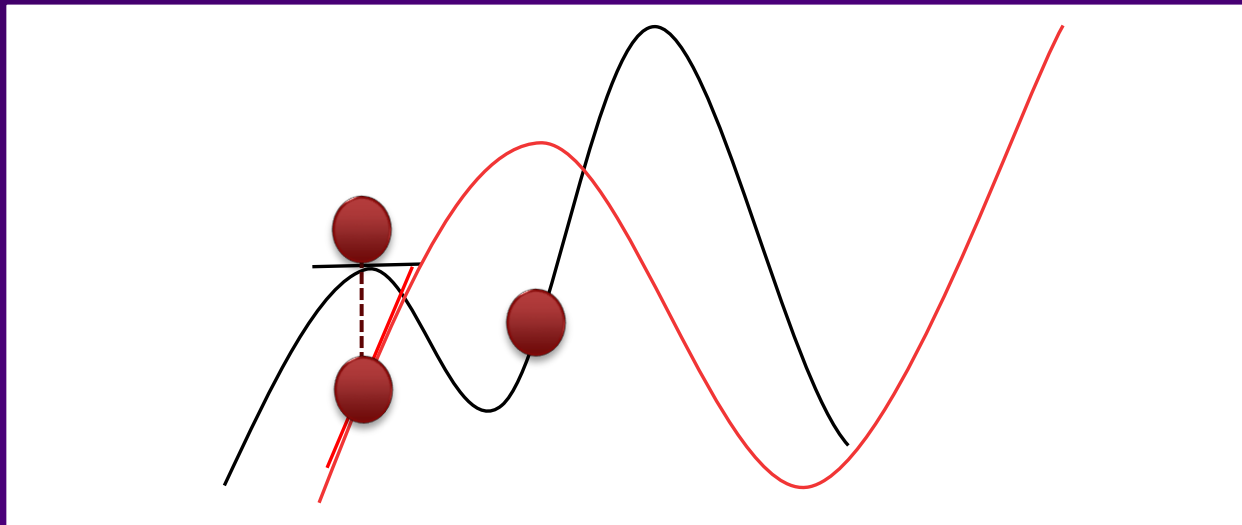
“BOOSTING FUNCTION” IDEA

At a local optimum s^1

$$\left. \frac{\partial H_i(\mathbf{s})}{\partial s_i} \right|_{s^1} = 0$$

Alter $H_i(\mathbf{s})$ to $\hat{H}_i(\mathbf{s})$

$$\left. \frac{\partial \hat{H}_i(\mathbf{s})}{\partial s_i} \right|_{s^1} \neq 0$$

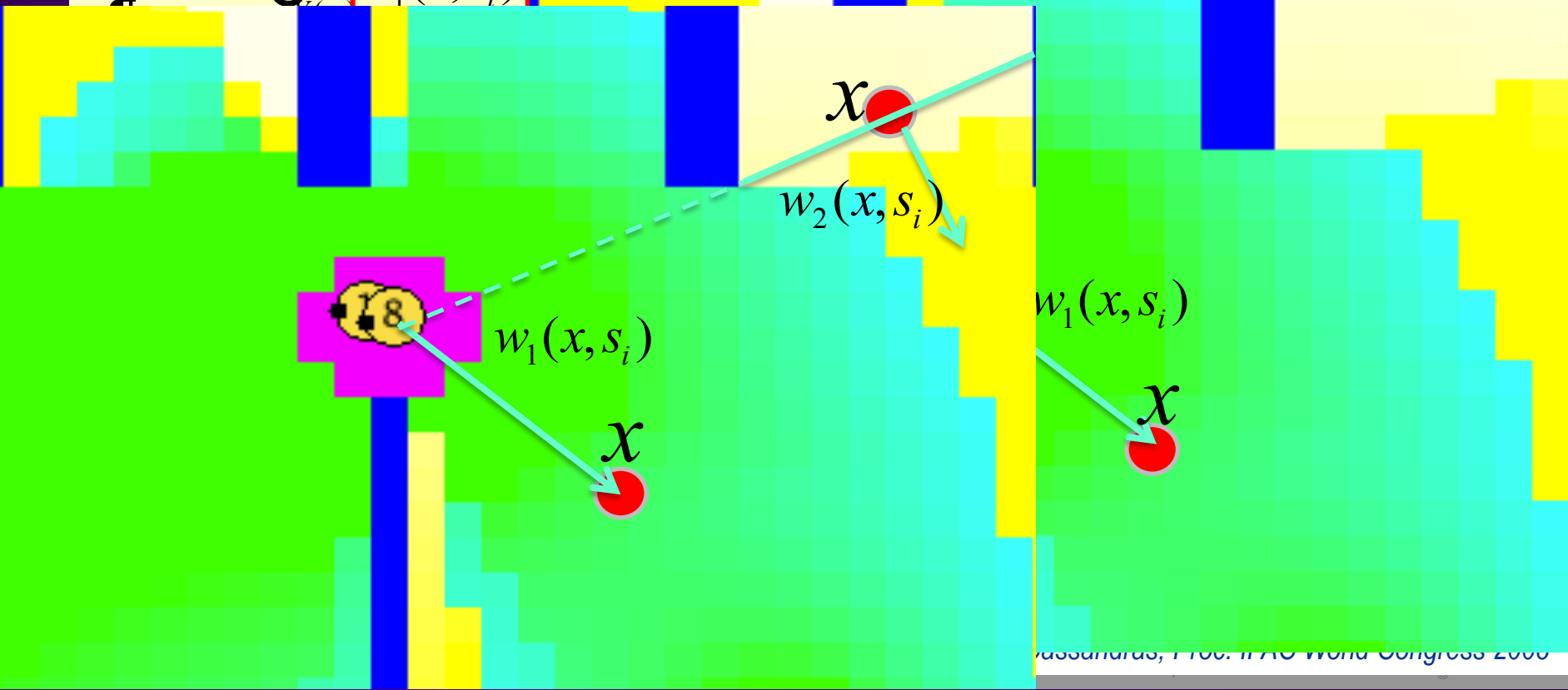


NOTE: Hard to find the proper $\hat{H}_i(\mathbf{s})$ \rightarrow try altering $\frac{\partial H_i(\mathbf{s})}{\partial s_i}$ directly

PARTIAL DERIVATIVE STRUCTURE

$$\frac{\partial H_i(s)}{\partial s_{ix}} = \int_{V(s_i)} w_1(x, s_i) \frac{(x - s_i)_x}{d_i(x)} dx + \sum_{j \in \Gamma_i} \text{sgn}(n_{jx}) \frac{\sin \theta_{ij}}{D_{ij}} \int_0^{z_{ij}} w_2(\rho_{ij}(r), s_i) r dr$$

$$\frac{\partial H_i(s)}{\partial s_{iy}} = \int_{V(s_i)} w_1(x, s_i) \frac{(x - s_i)_y}{d_i(x)} dx + \sum_{j \in \Gamma_i} \text{sgn}(n_{jy}) \frac{\cos \theta_{ij}}{D_{ij}} \int_0^{z_{ij}} w_2(\rho_{ij}(r), s_i) r dr$$



Cassandras, Proc. IEEE World Congress 2000

BOOSTING FUNCTION APPROACH

BOOSTING FUNCTION: Transform the derivative so its value is $\neq 0$ and provides a “boost” towards more likely optimum

$$\frac{\partial H_i(s)}{\partial s_{ix}} = \int_{V(s_i)} \boxed{w_1(x, s_i)} \frac{(x - s_i)_x}{d_i(x)} dx + \int_{j \in G_i} \text{sgn}(n_{jx}) \frac{\sin \theta_{ij}}{D_{ij}} \int_0^{z_{ij}} \boxed{w_2(r_{ij}(r), s_i)} r dr$$

$$\hat{w}_1(x, s_i) = g_i(w_1(x, s_i))$$

$$\hat{w}_2(x, s_i) = h_i(w_2(x, s_i))$$

Focus on linear forms:

$$\hat{w}_1(x, s_i) = \alpha_1(x, \mathbf{s}) w_1(x, s_i) + \beta_1(x, \mathbf{s})$$

$$\hat{w}_2(x, s_i) = \alpha_2(x, \mathbf{s}) w_2(x, s_i) + \beta_2(x, \mathbf{s})$$

THREE BOOSTING FUNCTIONS

1. P -boosting function
2. Neighbor-boosting function
3. Φ -boosting function

P-BOOSTING FUNCTION

Assign higher weights for low-coverage points

$$a_1(x, s) = kP(x, s)^{-\gamma}$$

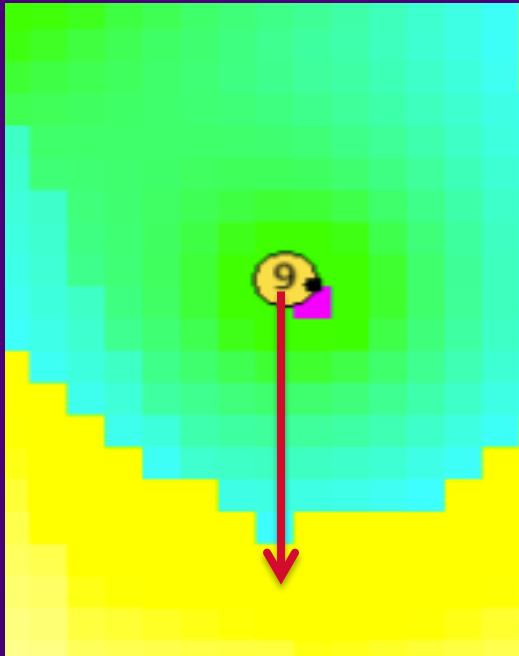
$$b_1(x, s) = 0$$

$$a_2(x, s) = 1$$

$$b_2(x, s) = 0$$

$$\hat{w}_1(x, s_i) = kP(x, s)^{-\gamma} w_1(x, s_i)$$

$$\hat{w}_2(x, s_i) = w_2(x, s_i)$$



$$P(x, s) = 1 - \prod_{i=1}^N [1 - \hat{p}_i(x, s_i)]$$

Boosted derivative:

$$\frac{\partial \hat{H}_i(s)}{\partial s_{ix}} = \int_{V(s_i)} \hat{w}_1(x, s_i) \frac{(x - s_i)_x}{d_i(x)} dx + \int_{j \in \mathcal{G}_i} \mathbf{a}_{j \hat{\mathbf{G}}} \text{sgn}(n_{jx}) \frac{\sin \alpha_{ij}}{D_{ij}} \int_0^{z_{ij}} \hat{w}_2(r_{ij}(r), s_i) r dr$$

NEIGHBOR-BOOSTING FUNCTION

Add repelling forces from agent's neighbors

$$a_1(x, s) = 1$$

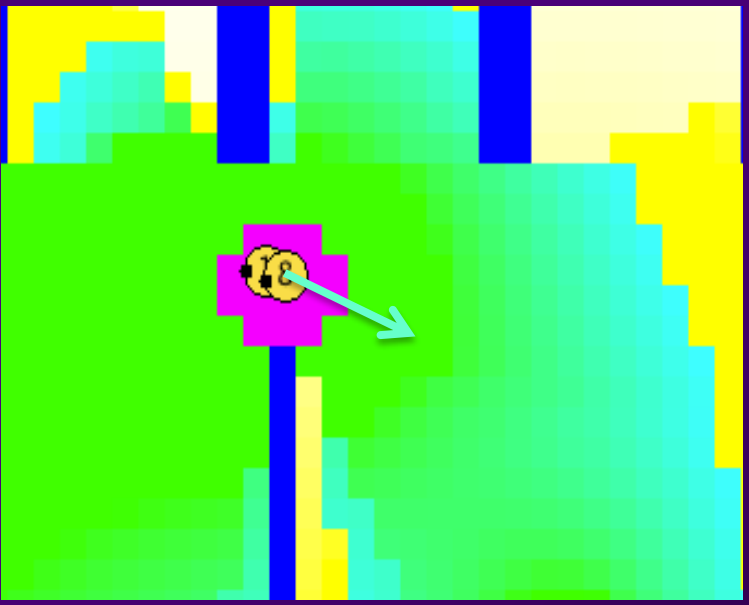
$$b_1(x, s) = \mathring{a} \sum_{j \in B_i} d(x - s_j) \frac{k_j}{\|s_i - x\|^g}$$

$$a_2(x, s) = 1$$

$$b_2(x, s) = 0$$

$$\hat{w}_1(x, s_i) = w_1(x, s_i) + \sum_{j \in B_i} \delta(x - s_j) \frac{k_j}{\|s_i - x\|^\gamma}$$

$$\hat{w}_2(x, s_i) = w_2(x, s_i)$$



Boosted derivative:

$$\frac{\mathring{H}_i(s)}{\mathring{S}_{ix}} = \mathring{0}_{V(s_i)} \hat{w}_1(x, s_i) \frac{(x - s_i)_x}{d_i(x)} dx + \mathring{a} \sum_{j \in G_i} \text{sgn}(n_{jx}) \frac{\sin \alpha_{ij}}{D_{ij}} \mathring{0}_0^{z_{ij}} \hat{w}_2(r_{ij}(r), s_i) r dr$$

Φ -BOOSTING FUNCTION

Boost weights for points poorly covered by agent's neighbors

$$a_1(x, s) = kF_i(x)^g$$

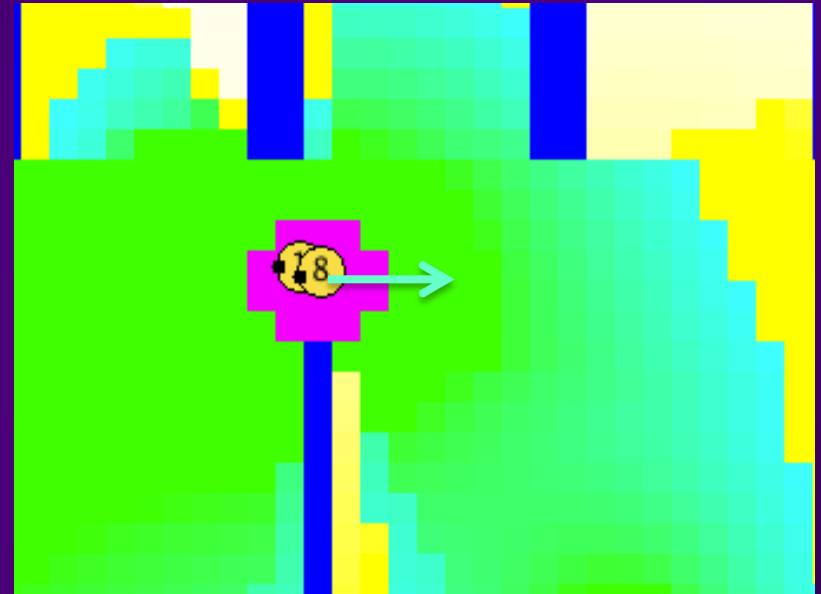
$$b_1(x, s) = 0$$

$$a_2(x, s) = 1$$

$$b_2(x, s) = 0$$

$$\hat{w}_1(x, s_i) = kF_i(x)^g w_1(x, s_i)$$

$$\hat{w}_2(x, s_i) = w_2(x, s_i)$$

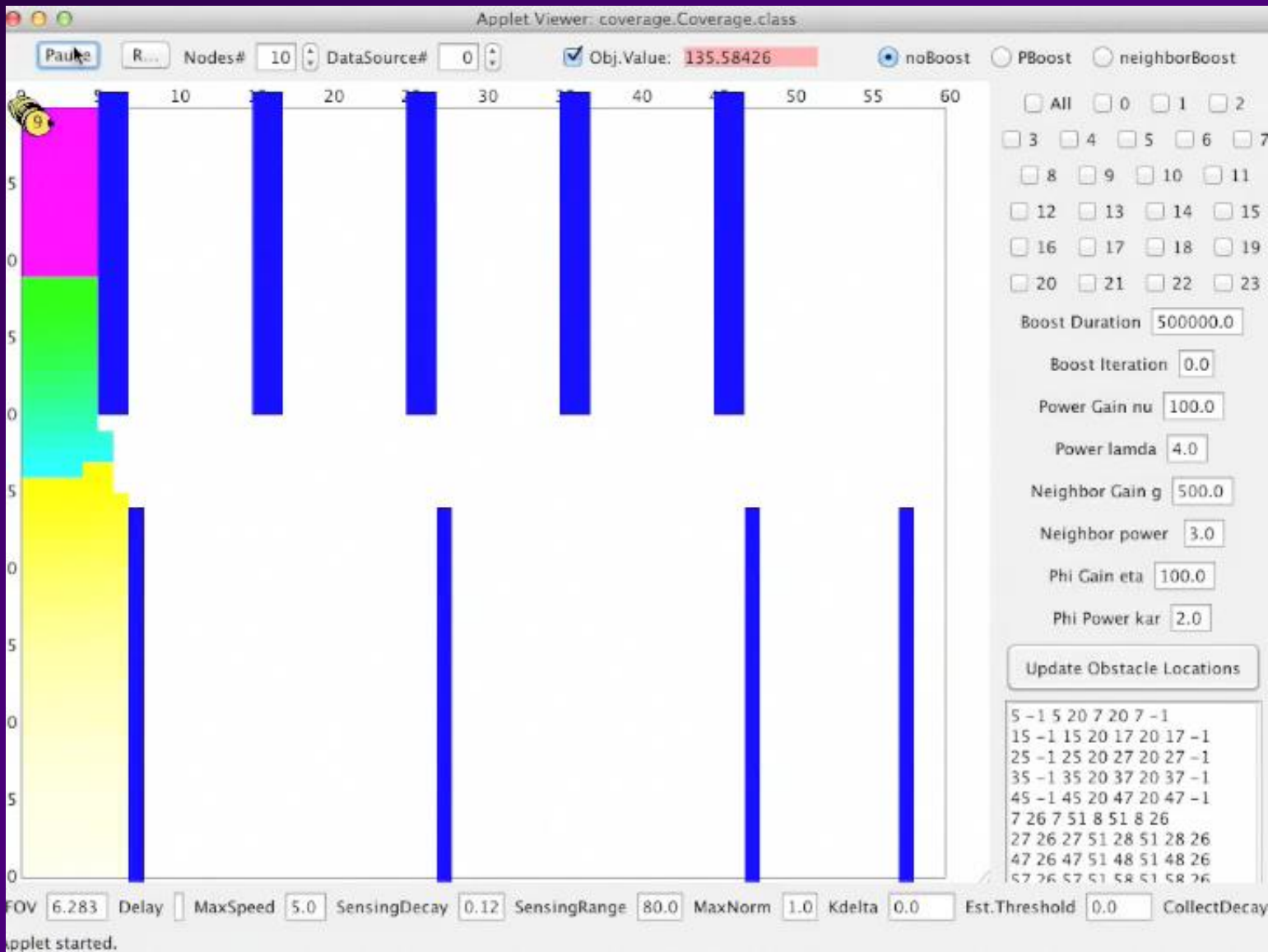


$$\Phi_i(x) = \prod_{k \in B_i} [1 - \hat{p}_k(x, s_k)]$$

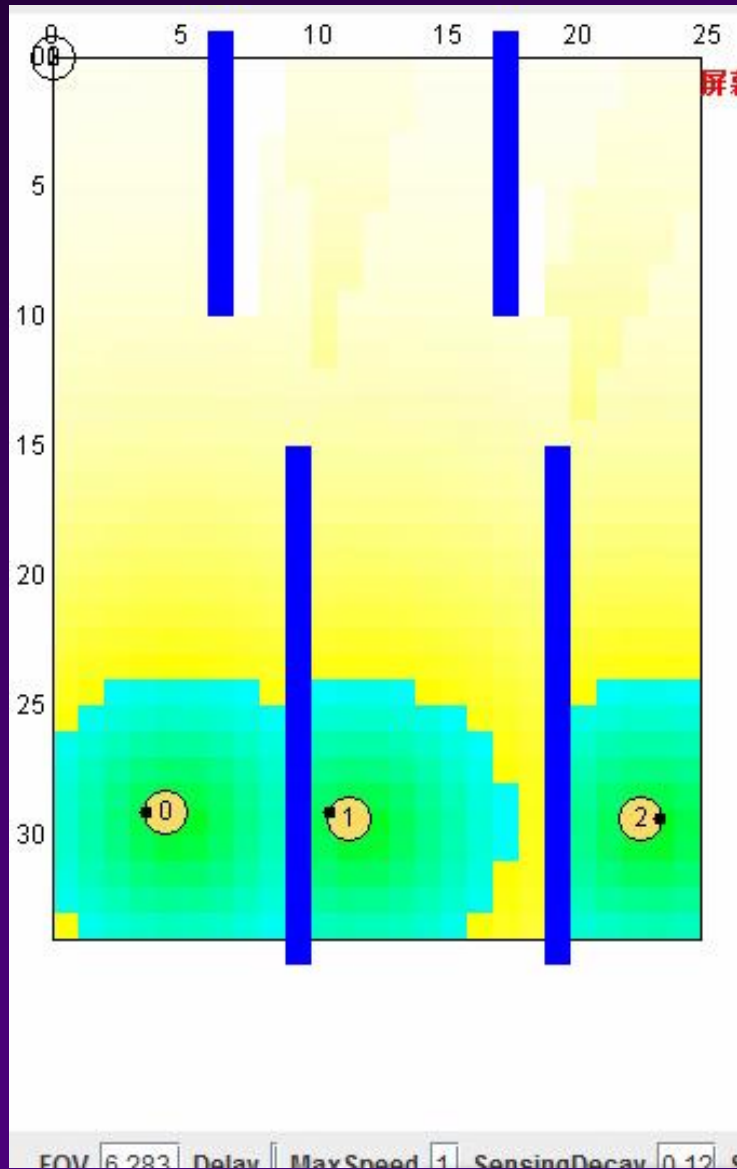
**Boosted
derivative:**

$$\frac{\partial \hat{H}_i(s)}{\partial s_{ix}} = \int_{V(s_i)} \hat{w}_1(x, s_i) \frac{(x - s_i)_x}{d_i(x)} dx + \sum_{j \in \Gamma_i} \text{sgn}(n_{jx}) \frac{\sin \theta_{ij}}{D_{ij}} \int_0^{z_{ij}} \hat{w}_2(\rho_{ij}(r), s_i) r dr$$

BOOSTING PROCESS EXAMPLE



COVERAGE EXAMPLE: SIMULATED v REAL



RELATED APPROACHES FOR GLOBAL OPTIMALITY

Simulated Annealing:

- Random perturbations for escaping local optima
- Can reach global optimum but very slow

Multistarts, Stochastic Comparison Algorithm (SCA):

- Random initial points
- SCA can reach global optimum but very slow

Submodularity, greedy algorithms:

- If $H(s)$ submodular, can obtain bounds
– sometimes very tight !

$f: 2^N \rightarrow \mathbb{R}$ submodular if $f(S \cup \{k\}) - f(S) \geq f(T \cup \{k\}) - f(T)$
for any $S \subset T \subset F, k \in F, k \notin T$

SUMMARY, RESEARCH DIRECTIONS

- Small, cheap cooperating devices cannot handle complexity
⇒ we need **DISTRIBUTED** control and optim. algorithms
- Cooperating agents operate autonomously (asynchronously)
⇒ we need **ASYNCHRONOUS (EVENT-DRIVEN)** control/optimization schemes
- Too much communication kills node energy sources
⇒ communicate **ONLY** when necessary
⇒ we need **EVENT-DRIVEN** control/optimization schemes
- Networks grow large, sensing tasks grow large
⇒ we need **SCALABLE** control and optim. algorithms