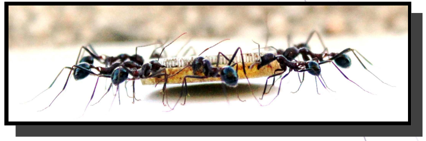
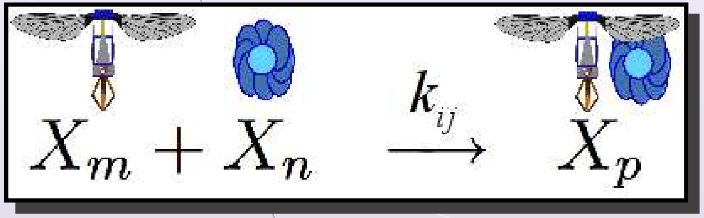
SCHOOL OF Life Sciences
ARIZONA STATE UNIVERSITY



# Stochastic Robotics: Complexity, Compositionality, and Scalability

Dr. Theodore (Ted) P. Pavlic
tpavlic@asu.edu

Friday, February 22, 2013



# Overview

The International Journal of Robotics Research
30(6) 713–729

DOI: 10.1177/0278364911403018
ijr.sagepub.com
SAGE

**A compositional framework for programming stochastically interacting robots**

**Nils Napp and Eric Klavins**

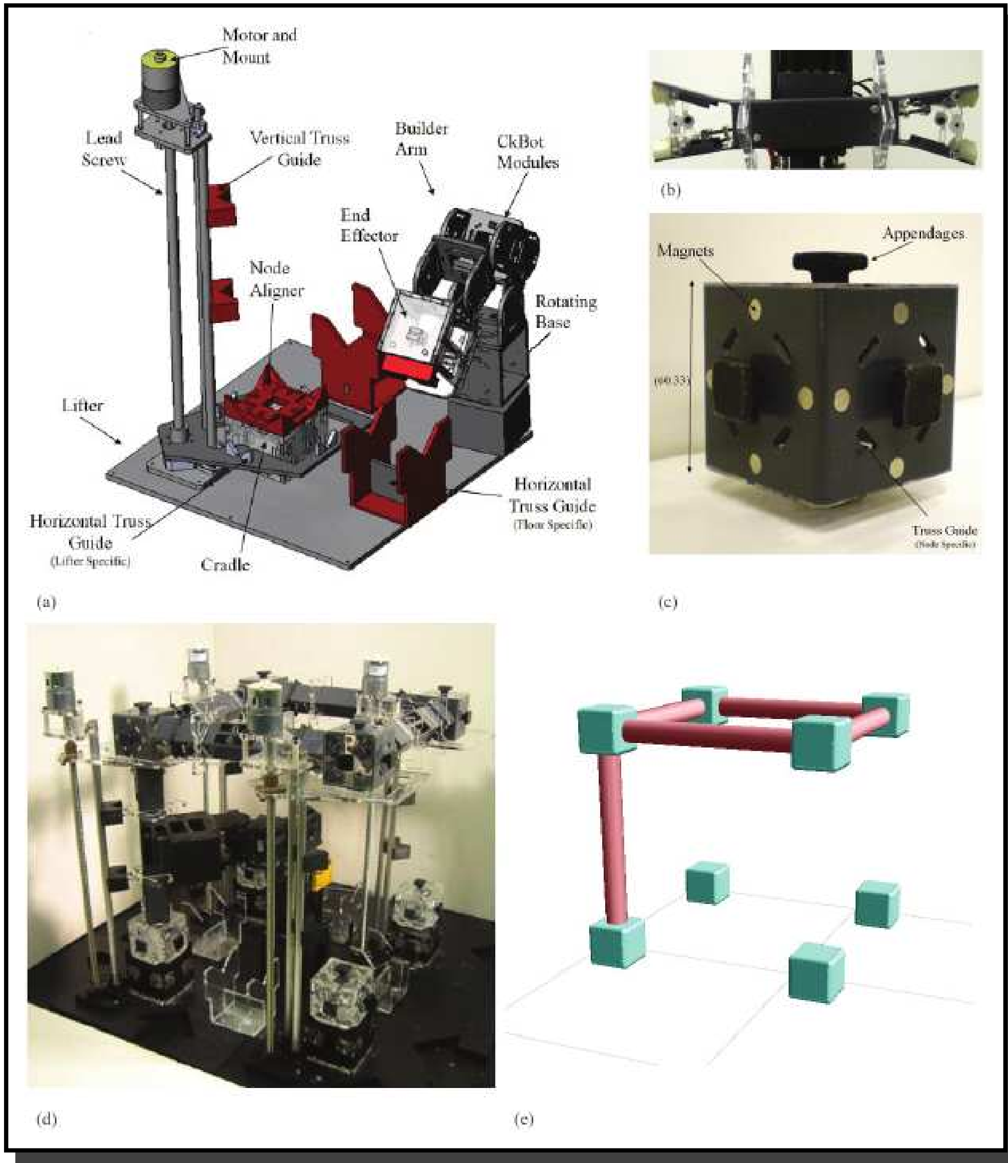
**Abstract**

*Large collections of simple, interacting robots can be difficult to program due to issues of concurrency and intermittent, probabilistic failures. Here, we present* Guarded Command Programming with Rates, *a formal framework for programming such multi-robot systems. Within this framework, we model robot behavior as a stochastic process and express concurrency and program composition using simple operations. In particular, we show how composition and other operations on programs can be used to specify increasingly complex behaviors of multi-robot systems and how stochasticity can be used to create programs that can tolerate failure of individual robots. Finally, we demonstrate our approach by encoding algorithms for routing parts in an abstract model of the Stochastic Factory Floor testbed (Galloway et al. 2010).*

N. Napp and E. Klavins, “A compositional framework for programming stochastically interacting robots,” *Int. J. Robot. Res. [Special Issue Stochasticity in Robot. Bio-Systems Part 2]*, vol. 30, no. 6, pp. 713–729, May 2011.

# Application: Factory Floor testbed
(Galloway et al. 2010)

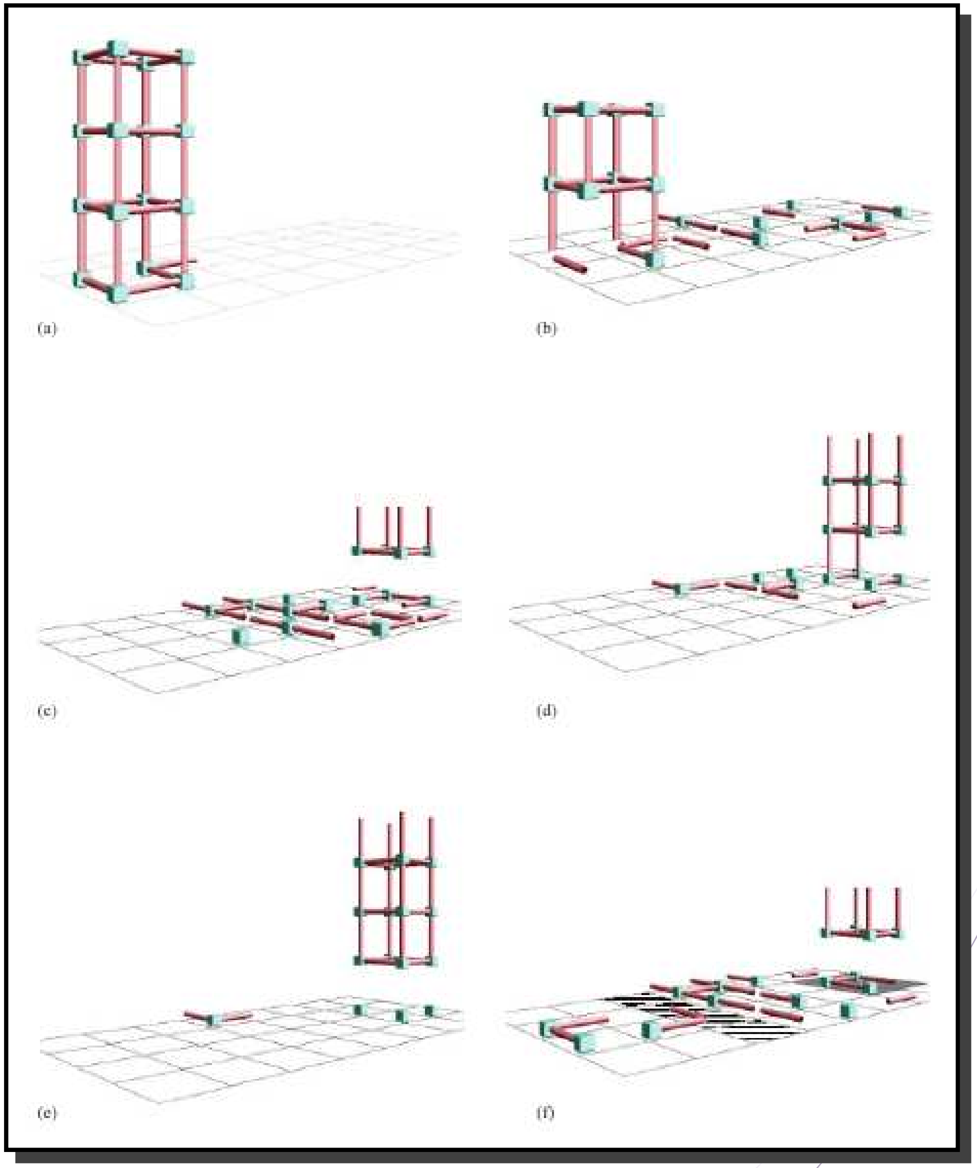




# Application: Factory Floor testbed

(Galloway et al. 2010)

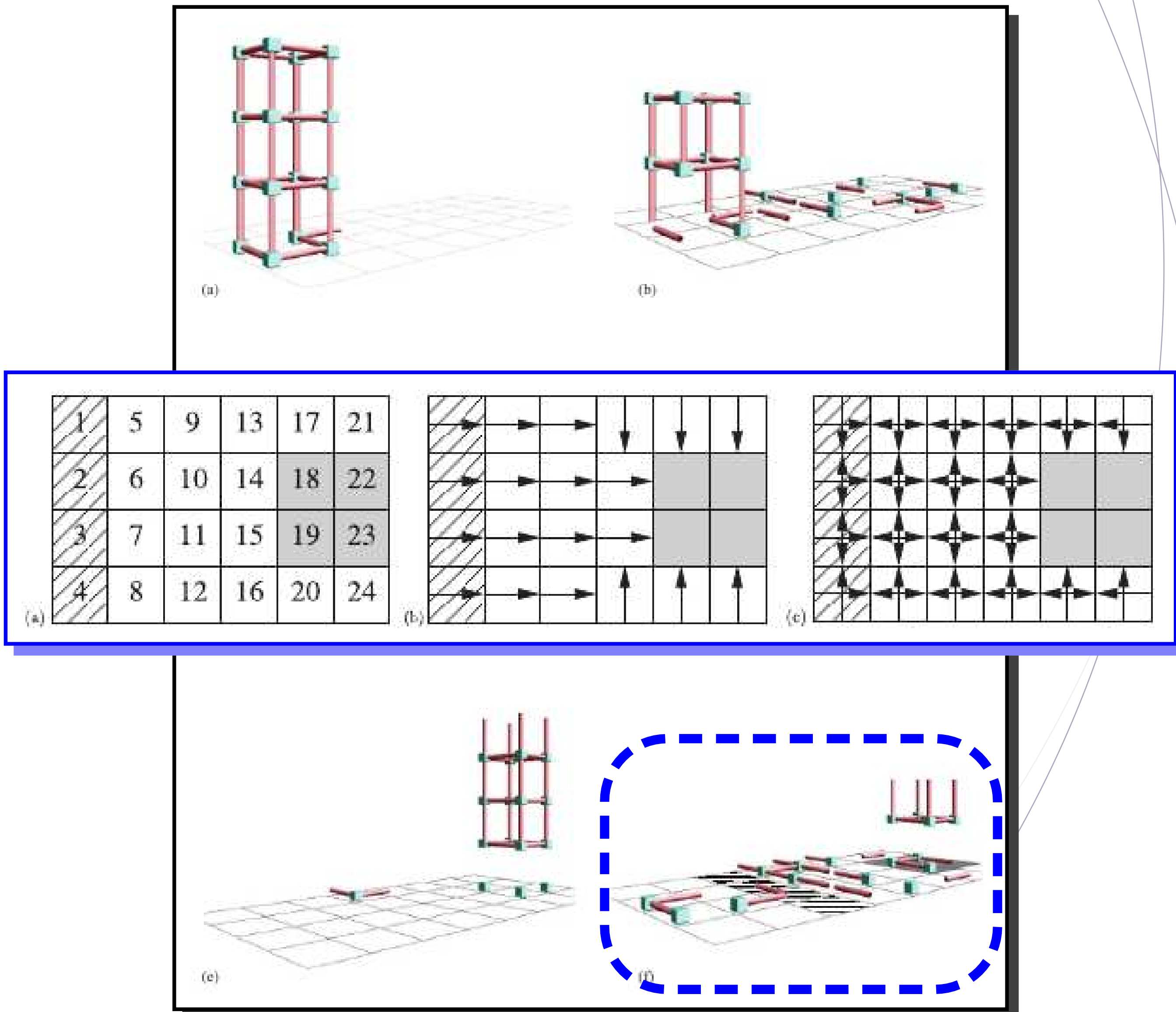




# Application: Factory Floor testbed routing

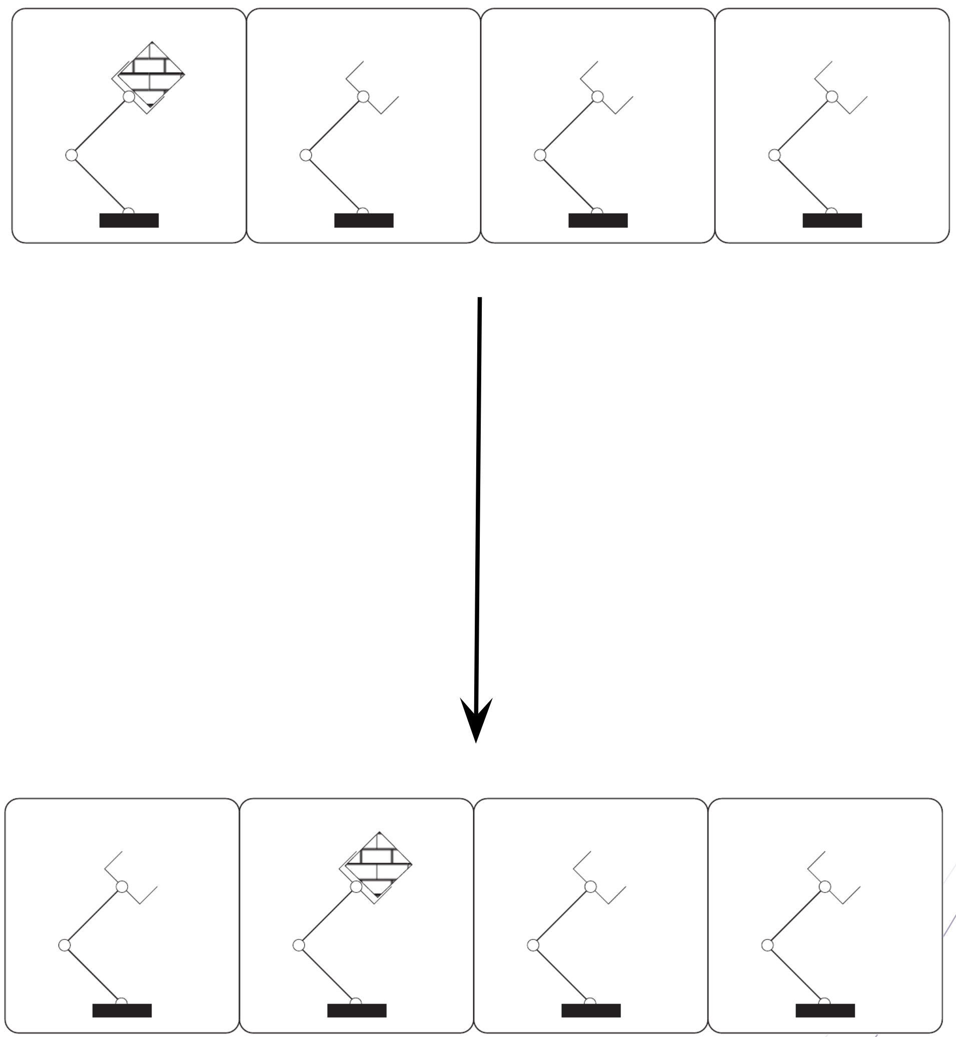
(Galloway et al. 2010; Napp and Klavins 2011)





# Application: Factory Floor testbed routing
(Napp and Klavins 2011)

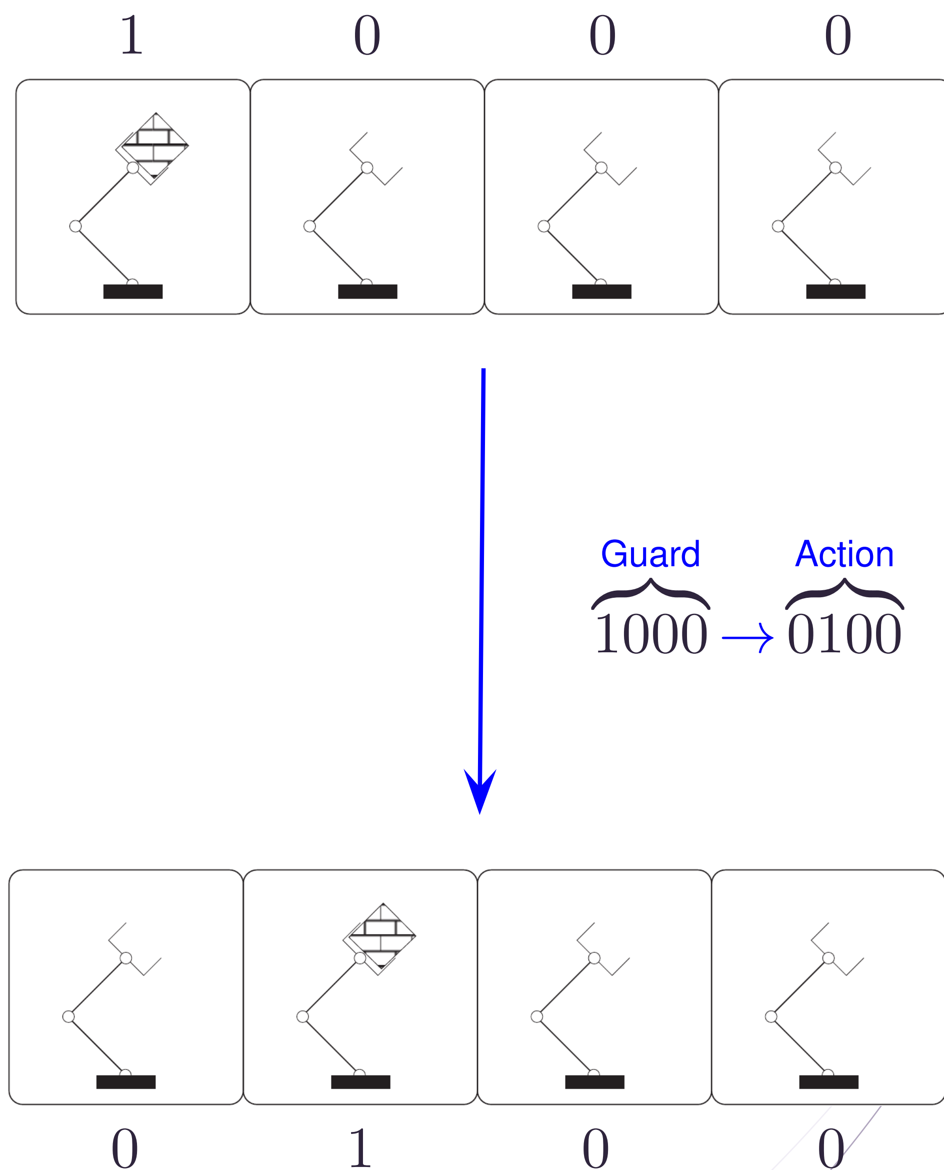




# Application: Factory Floor testbed routing

(Napp and Klavins 2011)





# Application: Factory Floor testbed routing

(Napp and Klavins 2011)

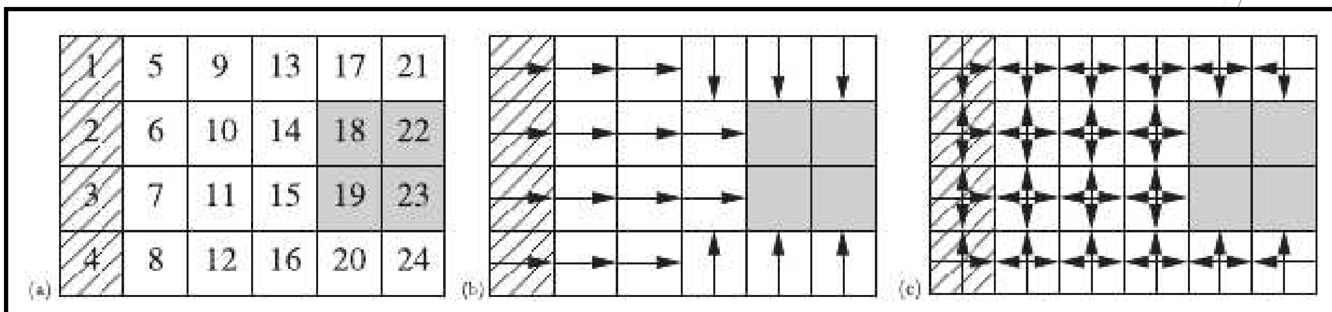


Weakly related to GCP by Dijkstra (1975).

*Non-deterministic* reasoning about programs (predicate transformers).

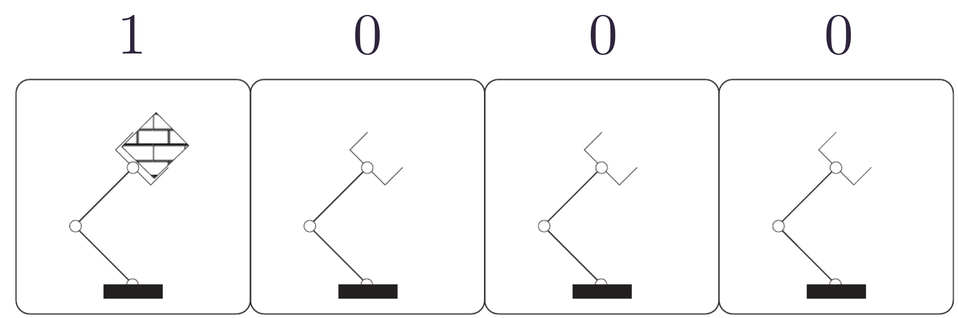
$$x \geq y \to m := x \,\square\, y \geq x \to m := y$$

$$\texttt{state} = 0100 \to \texttt{state} = 1000$$
$$\square\, \texttt{state} = 0100 \to \texttt{state} = 0010$$

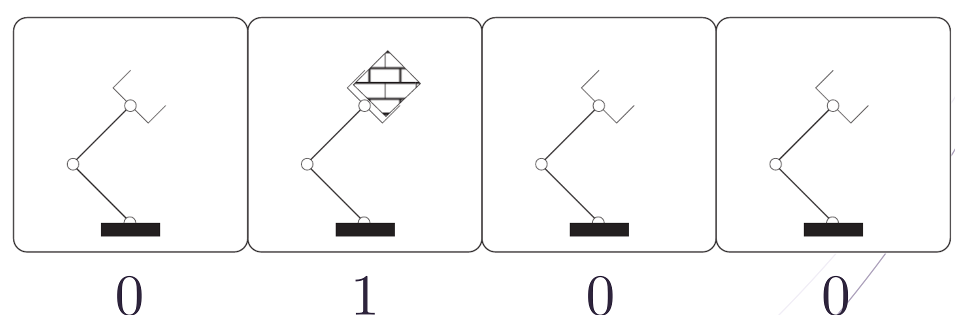


# Application: Factory Floor testbed routing
(Napp and Klavins 2011)



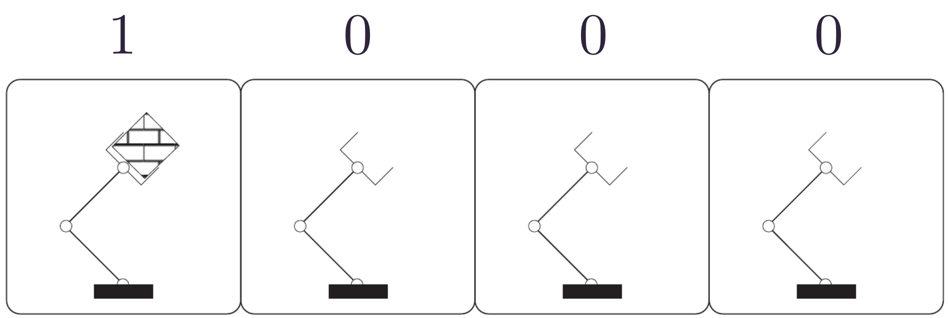



$$\overbrace{1000}^{\text{Guard}} \rightarrow \overbrace{0100}^{\text{Action}}$$ (global state)



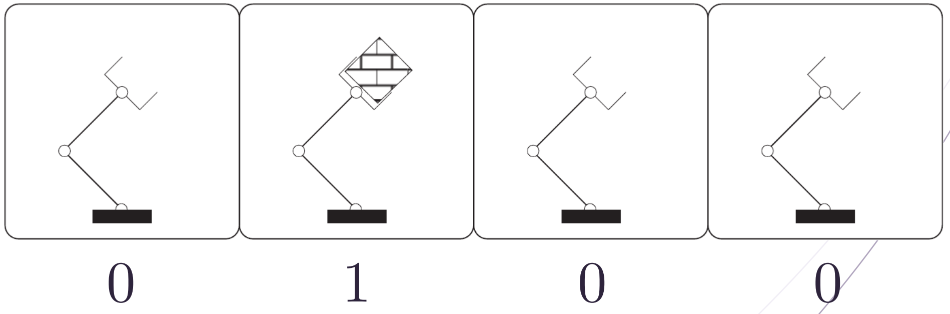
# Application: Factory Floor testbed routing
(Napp and Klavins 2011)





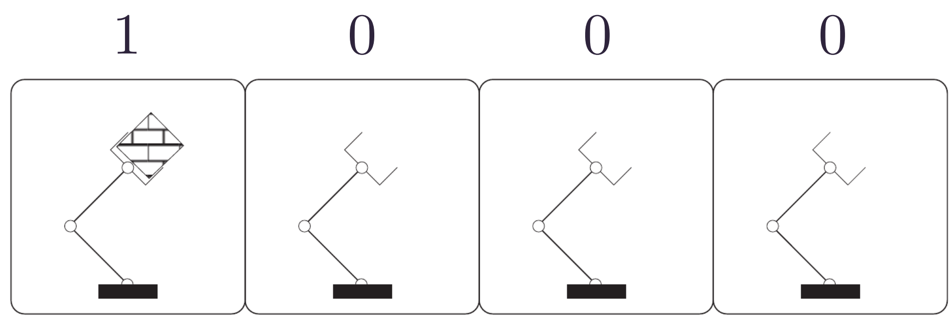

$$\overbrace{1000}^{\text{Guard}} \rightarrow \overbrace{0100}^{\text{Action}}$$ (global state)

$$s_1\overline{s}_2 \rightarrow \overline{s}_1 s_2$$ (site encoded)



# Application: Factory Floor testbed routing
(Napp and Klavins 2011)

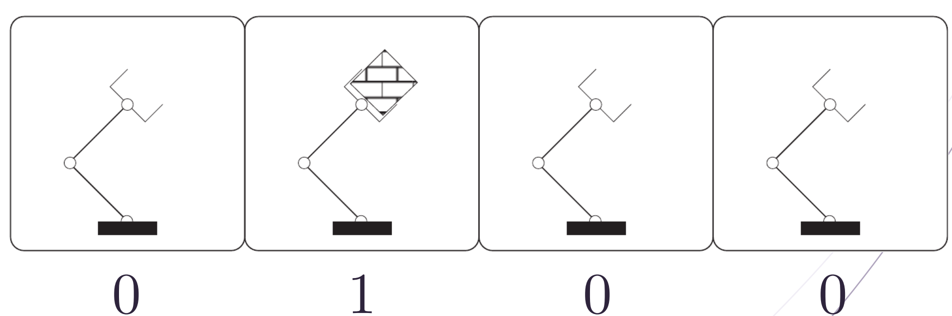





$$\overbrace{1000}^{\text{Guard}} \rightarrow \overbrace{0100}^{\text{Action}}$$ (global state)

$$s_1\overline{s}_2 \rightarrow \overline{s}_1 s_2$$ (site encoded)

$$s_i\overline{s}_{i+1} \rightarrow \overline{s}_i s_{i+1}$$ (behavioral ND program)



For compositionality, augment GCP with exponential rates.

$$s_i \overline{s}_{i+1} \rightarrow \overline{s}_i s_{i+1} \qquad \text{(GCP)}$$

$$s_i \overline{s}_{i+1} \overset{k}{\rightharpoonup} \overline{s}_i s_{i+1} \qquad \text{(GCPR)}$$

So a GCPR command is now an edge of a **Markov chain.** Rates are either chosen *programmatically* or are used to model error rates.

# Guarded Command Programming with Rates

(Napp and Klavins 2011)



For compositionality, augment GCP with exponential rates.

$$s_i \overline{s}_{i+1} \rightarrow \overline{s}_i s_{i+1} \qquad \text{(GCP)}$$

$$s_i \overline{s}_{i+1} \overset{k}{\rightharpoonup} \overline{s}_i s_{i+1} \qquad \text{(GCPR)}$$

So a GCPR command is now an edge of a **Markov chain.** Rates are either chosen *programmatically* or are used to model error rates.

- A GCPR $\Psi = \{(g_1, a_1, r_1), \ldots, (g_n, a_n, r_n)\}$ is a set of commands that are each made up of a guard, action, and rate.

# Guarded Command Programming with Rates
(Napp and Klavins 2011)



For compositionality, augment GCP with exponential rates.

$$s_i \overline{s}_{i+1} \rightarrow \overline{s}_i s_{i+1} \qquad \text{(GCP)}$$

$$s_i \overline{s}_{i+1} \overset{k}{\rightharpoonup} \overline{s}_i s_{i+1} \qquad \text{(GCPR)}$$

So a GCPR command is now an edge of a **Markov chain.** Rates are either chosen *programmatically* or are used to model error rates.

- A GCPR $\Psi = \{(g_1, a_1, r_1), \ldots, (g_n, a_n, r_n)\}$ is a set of commands that are each made up of a guard, action, and rate.
- A GCPR $\Psi$ can be *scaled* by $\sigma \in \mathbb{R}_{\geq 0}$ such that

$$\sigma\Psi = \bigcup_{(g,a,r)\in\Psi} (g, a, \sigma r).$$

# Guarded Command Programming with Rates

(Napp and Klavins 2011)



For compositionality, augment GCP with exponential rates.

$$s_i \overline{s}_{i+1} \rightarrow \overline{s}_i s_{i+1} \qquad \text{(GCP)}$$

$$s_i \overline{s}_{i+1} \overset{k}{\rightharpoonup} \overline{s}_i s_{i+1} \qquad \text{(GCPR)}$$

So a GCPR command is now an edge of a **Markov chain.** Rates are either chosen *programmatically* or are used to model error rates.

- A GCPR $\Psi = \{(g_1, a_1, r_1), \ldots, (g_n, a_n, r_n)\}$ is a set of commands that are each made up of a guard, action, and rate.
- A GCPR $\Psi$ can be *scaled* by $\sigma \in \mathbb{R}_{\geq 0}$ such that

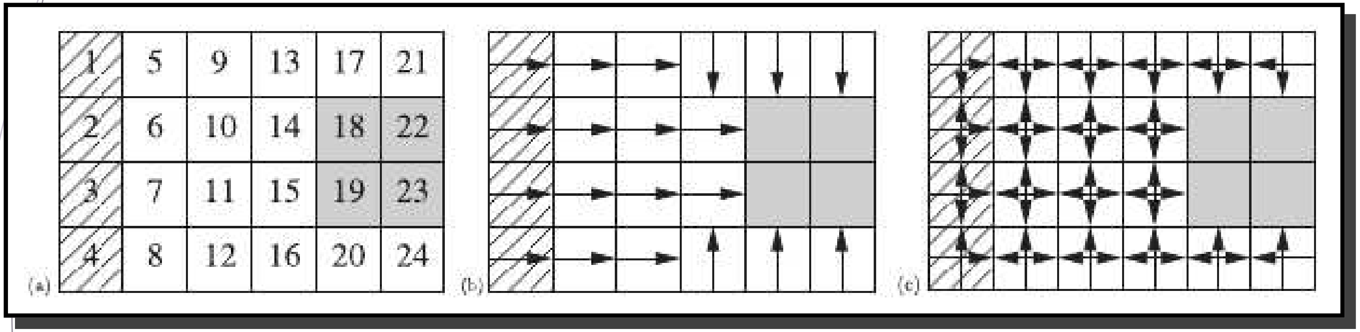
$$\sigma\Psi = \bigcup_{(g,a,r)\in\Psi} (g, a, \sigma r).$$

- The *composition* $\Psi \cup \Phi$ of GCPR $\Psi$ and GCPR $\Phi$ is the union of the two programs.

# Analysis of GCPR using Master Equation

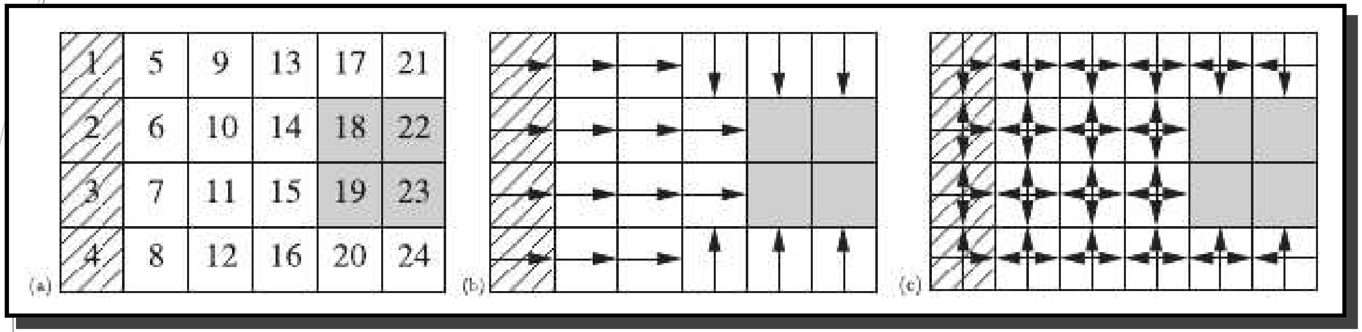
(Napp and Klavins 2011)






- Desired *and undesired behaviors* compose a *single* Markov process.

# Analysis of GCPR using Master Equation
## (Napp and Klavins 2011)

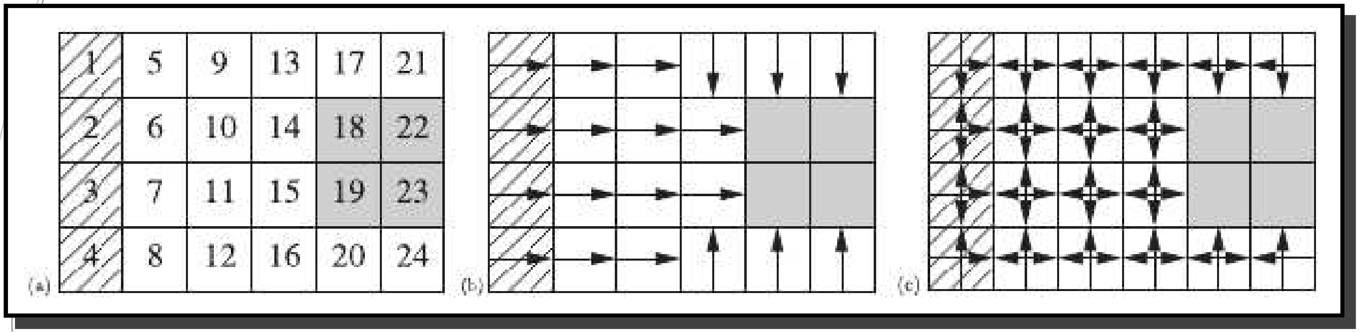



- Desired *and undesired behaviors* compose a *single* Markov process.
- Slow, robust behaviors can be *mixed* with fast, idealized behaviors.

# Analysis of GCPR using Master Equation

## (Napp and Klavins 2011)

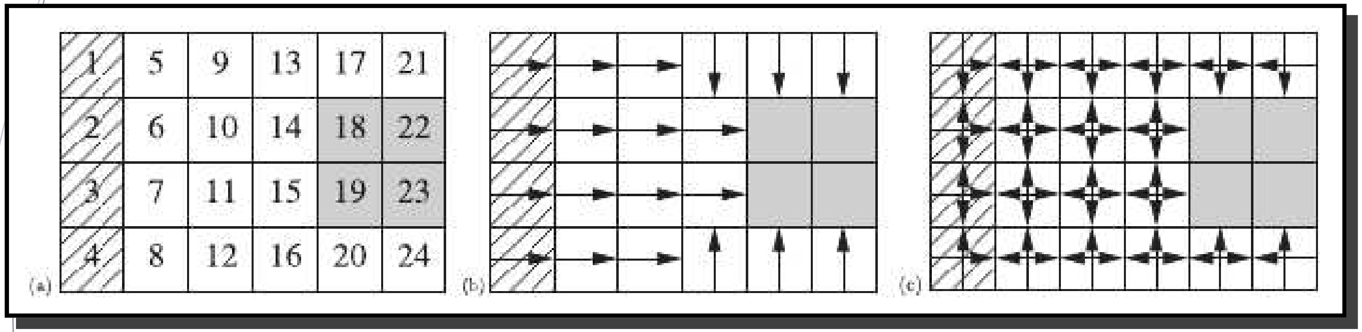





- Desired *and undesired behaviors* compose a *single* Markov process.
- Slow, robust behaviors can be *mixed* with fast, idealized behaviors.
- Rates and *scalars* are chosen to ensure adequate performance and error resilience.

# Analysis of GCPR using Master Equation
(Napp and Klavins 2011)






- Desired *and undesired behaviors* compose a *single* Markov process.
- Slow, robust behaviors can be *mixed* with fast, idealized behaviors.
- Rates and *scalars* are chosen to ensure adequate performance and error resilience.
- System described by linear master equation:

$$\dot{\vec{p}} = \vec{p}Q$$

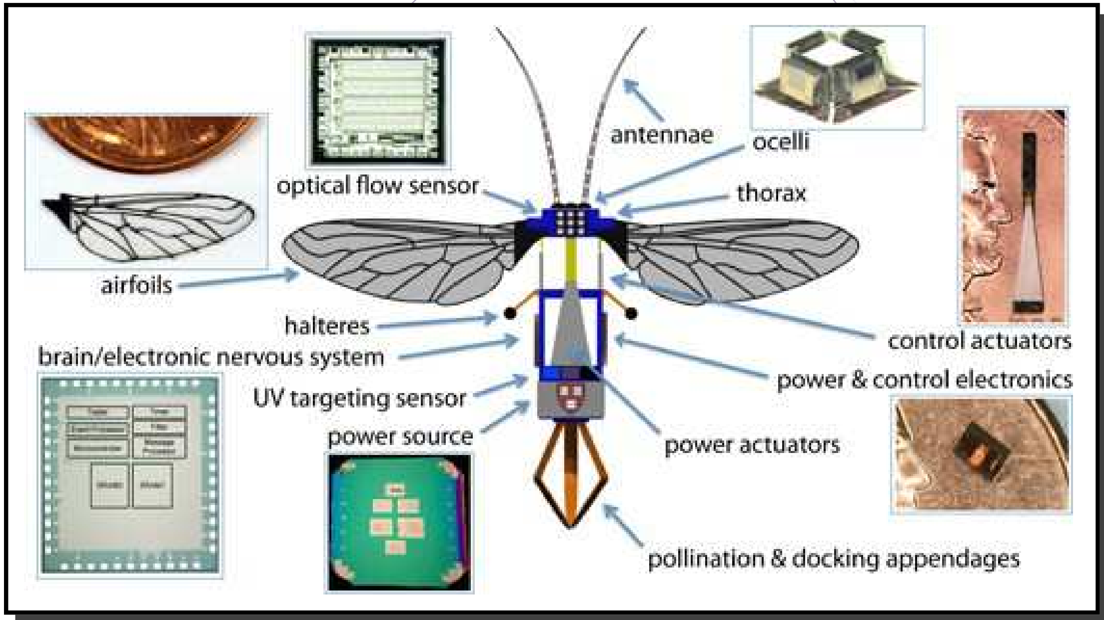
where $p_j$ is probability of state $j$ and $Q$ is graph Laplacian.

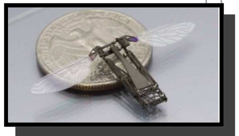
- ☐ Correctness: steady-state probability $p^*$
- ☐ Performance: spectrum of $Q$ (i.e., $\lambda_2$)

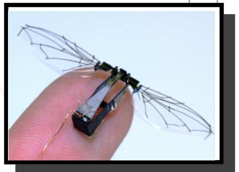
# Application: Artificial Pollination by RoboBees

(Berman et al. 2011a,b)








S. Berman, V. Kumar, and R. Nagpal, “Design of control policies for spatially inhomogeneous robot swarms with application to commercial pollination,” in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, May 9–13, 2011.

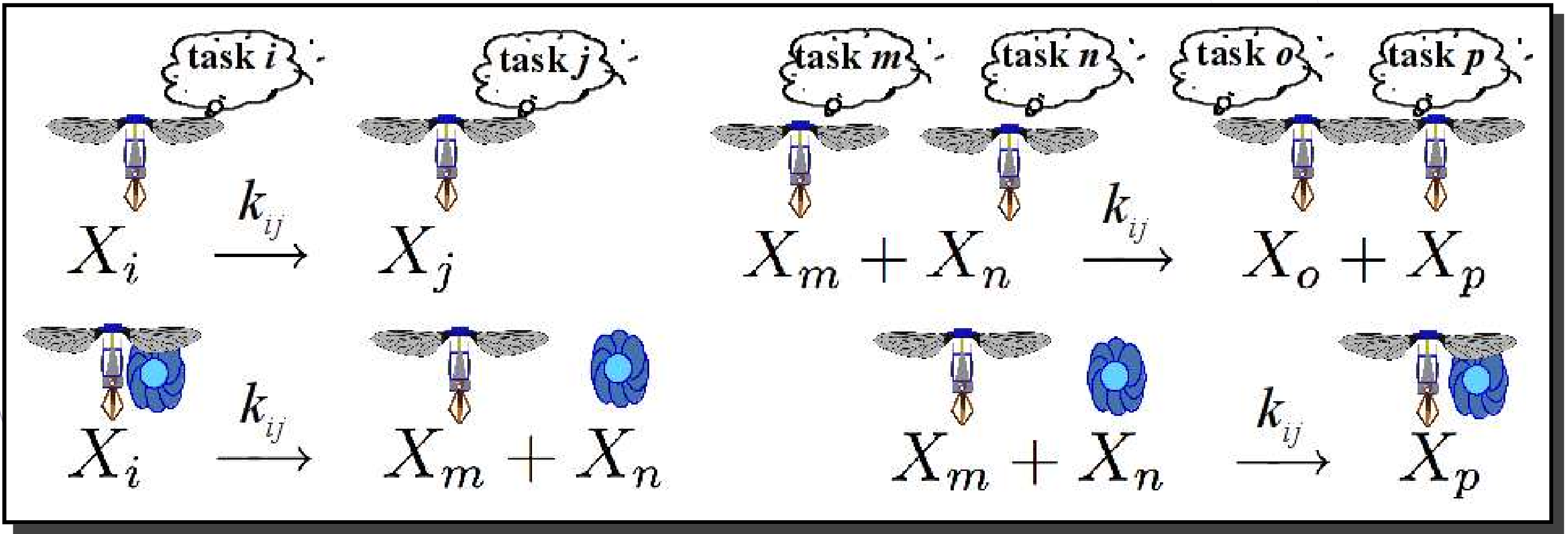
S. Berman, R. Nagpal, and Á. Halász, “Optimization of stochastic strategies for spatially inhomogeneous robot swarms: a case study in commerical pollination,” in *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, CA, USA, September 25–30, 2011, pp. 3923–3930.


`http://robobees.seas.harvard.edu/`

# RoboBees as Chemical Reaction Networks

(Berman et al. 2011a,b)





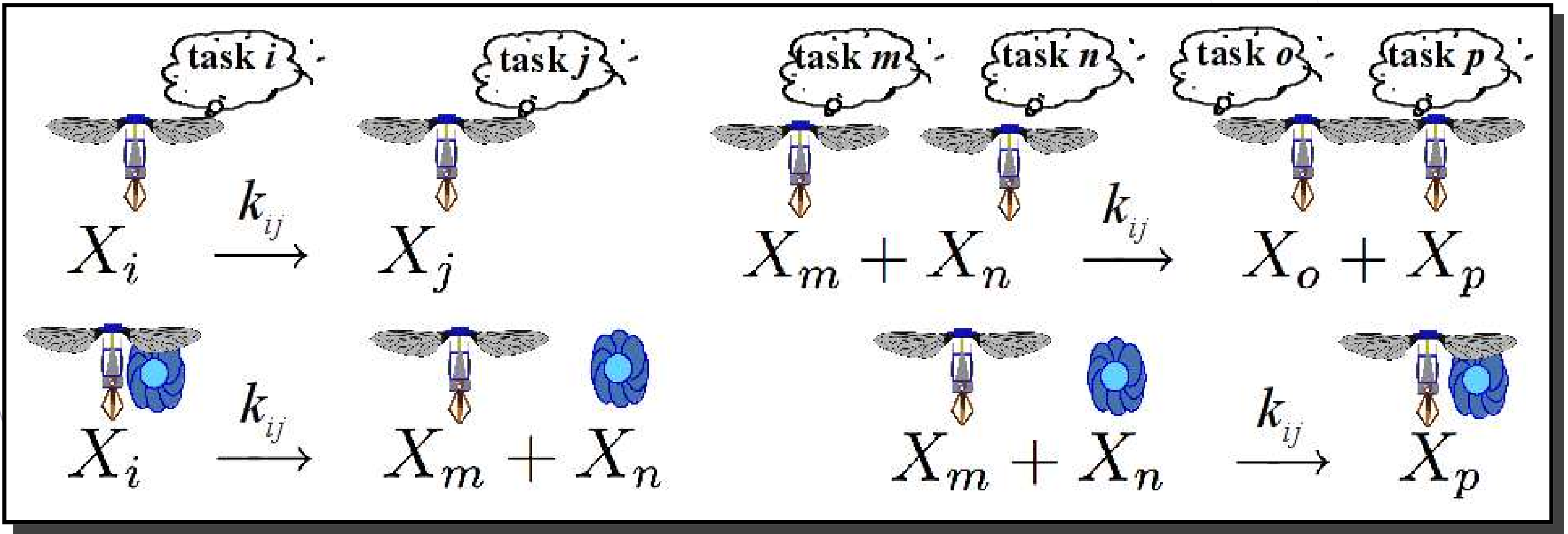

$$X_i \xrightarrow{k_{ij}} X_j \qquad X_m + X_n \xrightarrow{k_{ij}} X_o + X_p$$

$$X_i \xrightarrow{k_{ij}} X_m + X_n \qquad X_m + X_n \xrightarrow{k_{ij}} X_p$$

# RoboBees as Chemical Reaction Networks

(Berman et al. 2011a,b)

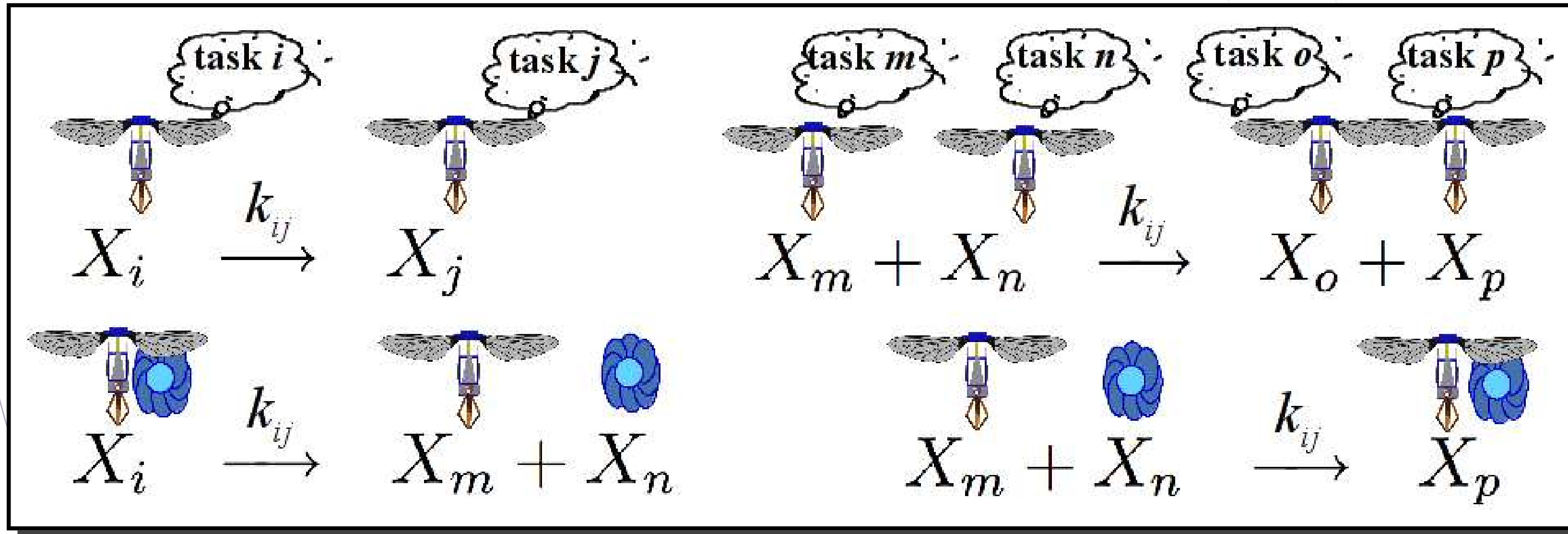





- Reactions model behavior switches; choose rates programmatically.

# RoboBees as Chemical Reaction Networks

(Berman et al. 2011a,b)

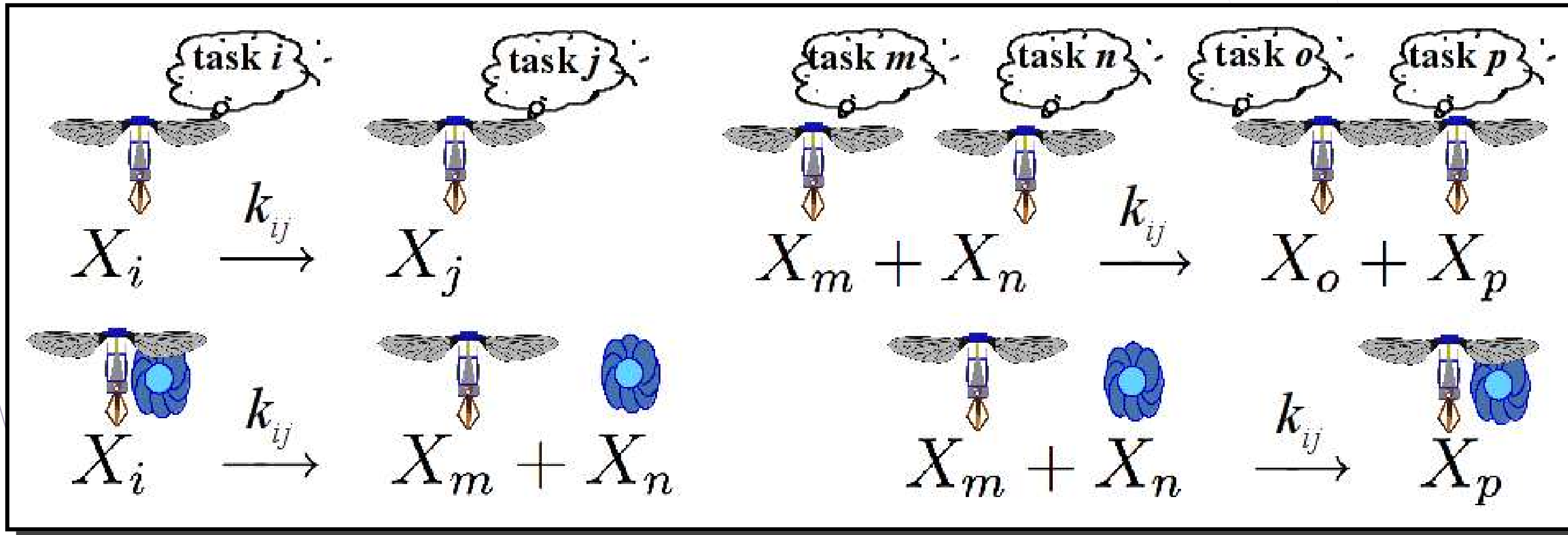



- Reactions model behavior switches; choose rates programmatically.
- Motion governed by drift–diffusion process; choose field and diffusion coefficient programmatically.

$$\vec{x}_i(t+\delta_t) = \vec{x}_i(t) + \vec{v}(\vec{x}_i, t)\delta_t + \sqrt{2D\delta_t}\vec{Z}(t), \qquad Z_j(t) \sim \mathcal{N}(0,1)$$

# RoboBees as Chemical Reaction Networks
(Berman et al. 2011a,b)






- Reactions model behavior switches; choose rates programmatically.
- Motion governed by drift–diffusion process; choose field and diffusion coefficient programmatically.

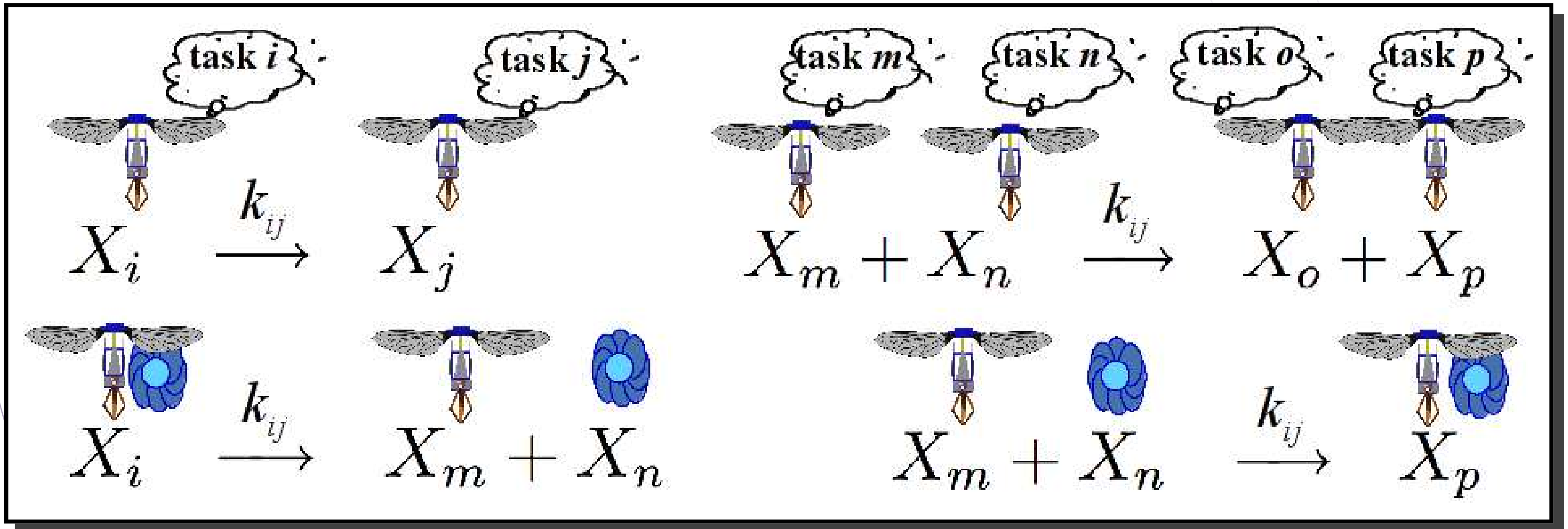
$$\vec{x}_i(t+\delta_t) = \vec{x}_i(t) + \vec{v}(\vec{x}_i, t)\delta_t + \sqrt{2D\delta_t}\vec{Z}(t), \qquad Z_j(t) \sim \mathcal{N}(0,1)$$

- *Macroscopic* design with advection–diffusion–reaction equations.

# RoboBees as Chemical Reaction Networks

(Berman et al. 2011a,b)






- Reactions model behavior switches; choose rates programmatically.
- Motion governed by drift–diffusion process; choose field and diffusion coefficient programmatically.

$$\vec{x}_i(t+\delta_t) = \vec{x}_i(t) + \vec{v}(\vec{x}_i, t)\delta_t + \sqrt{2D\delta_t}\vec{Z}(t), \qquad Z_j(t) \sim \mathcal{N}(0,1)$$

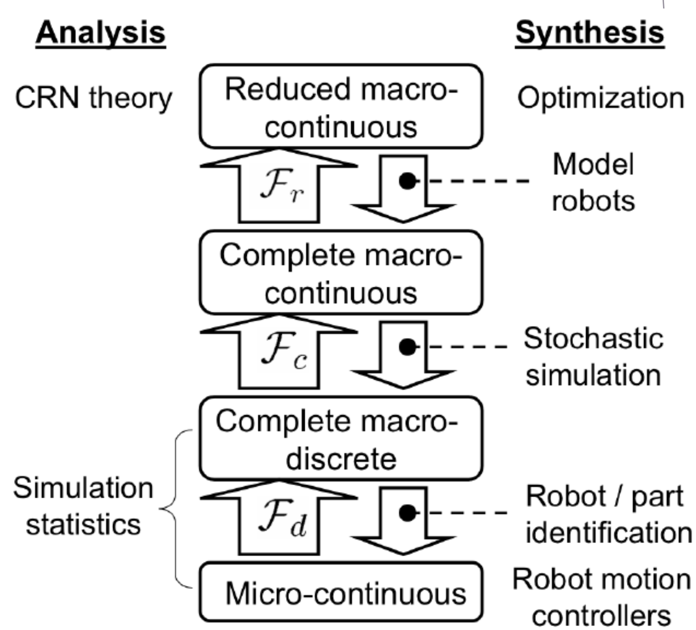
- *Macroscopic* design with advection–diffusion–reaction equations.
- **Goal:** Desired flower coverage (1500 robots, 3 minute bouts, wind).

# Application: Swarm Robotic Assembly
(Matthey et al. 2009)



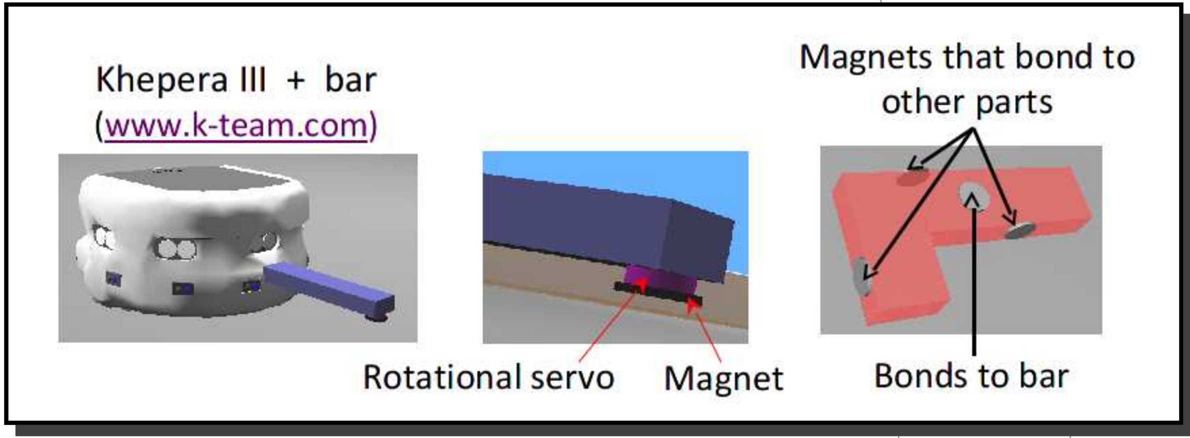
**Stochastic Strategies for a Swarm Robotic Assembly System**

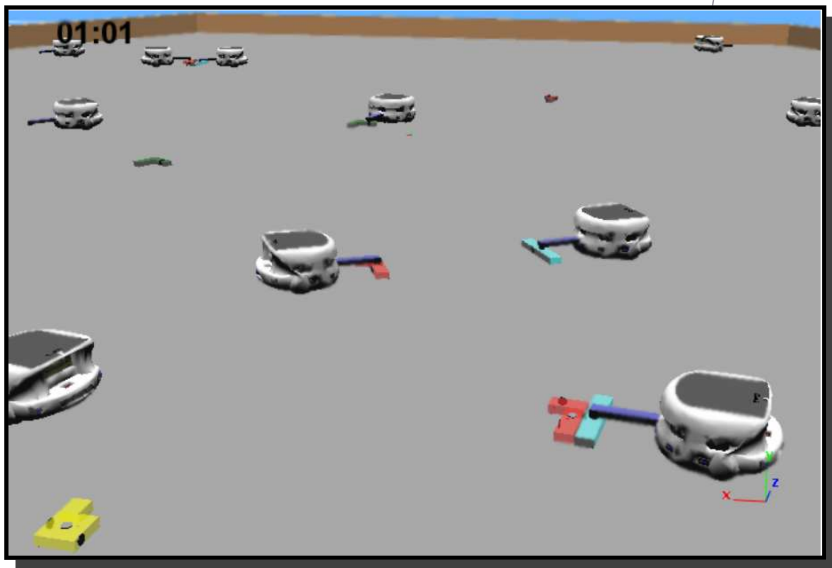
Loïc Matthey, Spring Berman and Vijay Kumar




L. Matthey, S. Berman, and V. Kumar, "Stochastic strategies for a swarm robotic assembly system," in *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 12–17, 2009, pp. 1953–1958

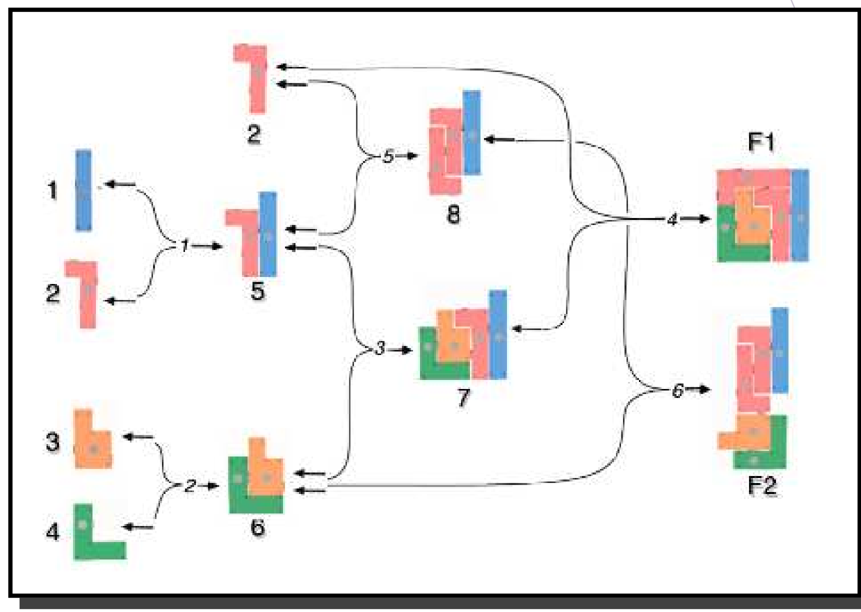
# Application: Swarm Robotic Assembly
(Matthey et al. 2009)

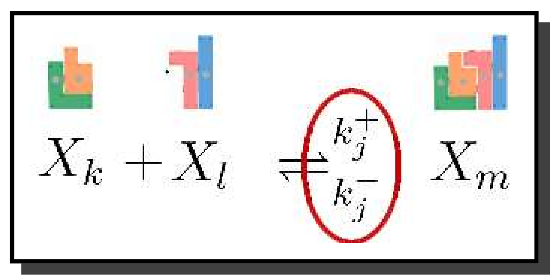


# Swarm Robotic Assembly as Chemical Reaction Network

(Matthey et al. 2009)



$$X_k + X_l \underset{k_j^-}{\overset{k_j^+}{\rightleftharpoons}} X_m$$

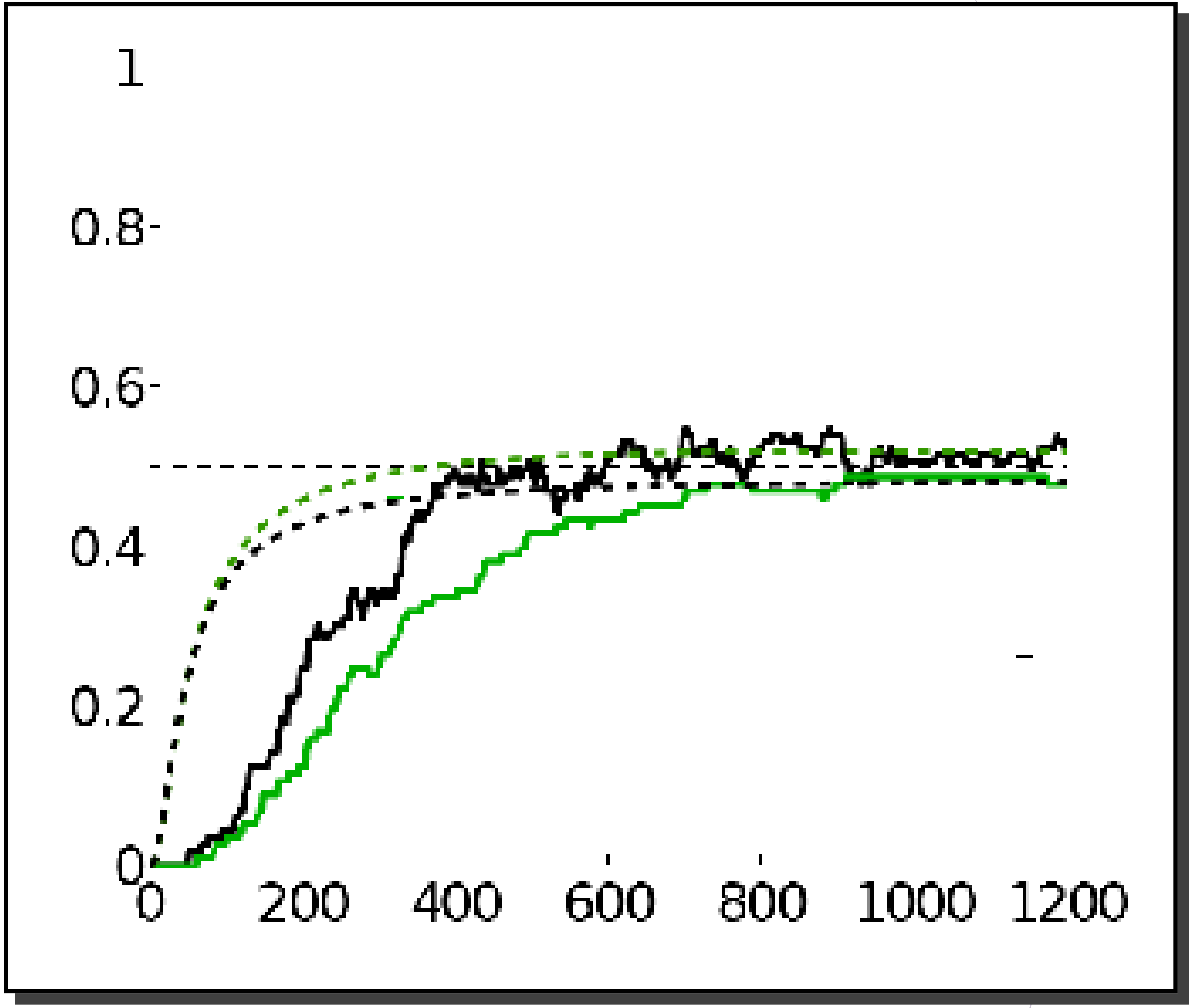
**Goal:** Equal number of both parts assembled (i.e., $x_{F1} = x_{F2}$).

# Swarm Robotic Assembly as Chemical Reaction Network

**(Matthey et al. 2009)**



**Results from CME Design:** Micro-/Macro-scopic Fraction of Parts 1 and 2



# Cooperative Transport in Ants
(Kumar et al. 2013)



## A Stochastic Hybrid System Model of Collective Transport in the Desert Ant *Aphaenogaster cockerelli*


Ganesh P. Kumar*
School of Computing, Informatics and Decision Systems Engineering
Arizona State University
Tempe, AZ, USA
Ganesh.P.Kumar@asu.edu

Aurélie Buffin
School of Life Sciences
Arizona State University
Tempe, AZ, USA
baurelie@asu.edu

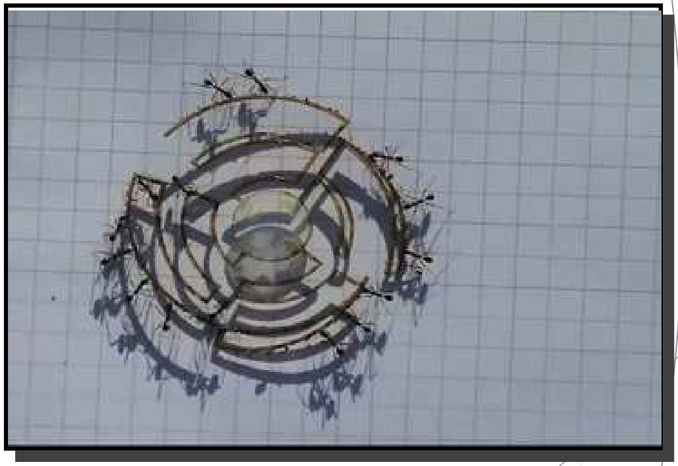
Theodore P. Pavlic
School of Life Sciences
Arizona State University
Tempe, AZ, USA
tpavlic@asu.edu

Stephen C. Pratt
School of Life Sciences
Arizona State University
Tempe, AZ, USA
Stephen.Pratt@asu.edu

Spring M. Berman
School for Engineering of Matter, Transport and Energy
Arizona State University
Tempe, AZ, USA
Spring.Berman@asu.edu


G. P. Kumar, A. Buffin, T. P. Pavlic, S. C. Pratt, and S. M. Berman, “A stochastic hybrid system model of collective transport in the desert ant *Aphaenogaster cockerelli*,” in *Proceedings of the 16th ACM International Conference on Hybrid Systems: Computation and Control*, Philadelphia, PA, April 8–11, 2013.

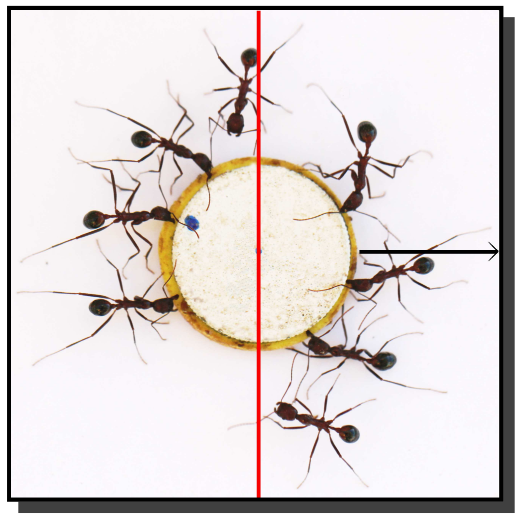
*Aphaenogaster cockerelli* ants

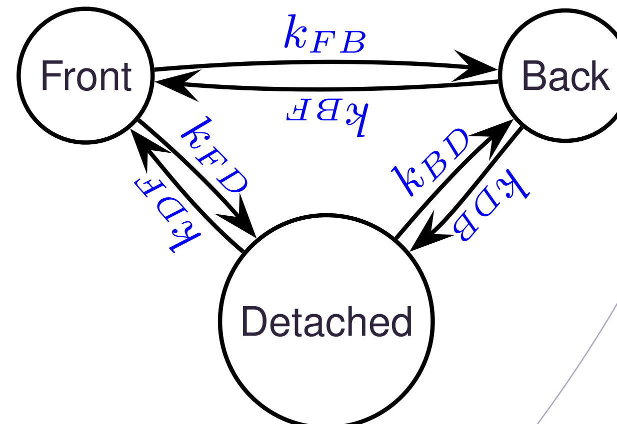


# Stochastic Hybrid System Model for Ants

(Kumar et al. 2013)








$$
\begin{cases}
\dot{x}_L = v_L \\
m_L \dot{v}_L = N_F \underbrace{K(v_L^d - v_L)}_{\text{Individual Behavior}} + \underbrace{\mu \,\mathrm{sgn}(v_L)(m_L g - (N_F + N_B) F_L)}_{\text{Friction}}
\end{cases}
$$

Generalized Lie derivative of moments along trajectories of SHS systems:

- For function $\psi$,

$$L\psi(\vec{x}) \triangleq \frac{\partial \psi}{\partial x_L}\dot{x}_L + \frac{\partial \psi}{\partial v_L}\dot{v}_L + \sum_{\substack{i,j\in\{F,B,D\}\\ i\neq j}} (\psi(\phi_{ij}(\vec{x})) - \psi(\vec{x}))k_{ij}N_i$$

  and

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{E}(\psi(\vec{x})) = \mathrm{E}(L\psi(\vec{x})).$$

So arbitrary moment dynamics can be derived from SHS model.

Generalized Lie derivative of moments along trajectories of SHS systems:

- For function $\psi$,

$$L\psi(\vec{x}) \triangleq \frac{\partial \psi}{\partial x_L}\dot{x}_L + \frac{\partial \psi}{\partial v_L}\dot{v}_L + \sum_{\substack{i,j\in\{F,B,D\}\\ i\neq j}} (\psi(\phi_{ij}(\vec{x})) - \psi(\vec{x}))k_{ij}N_i$$

  and

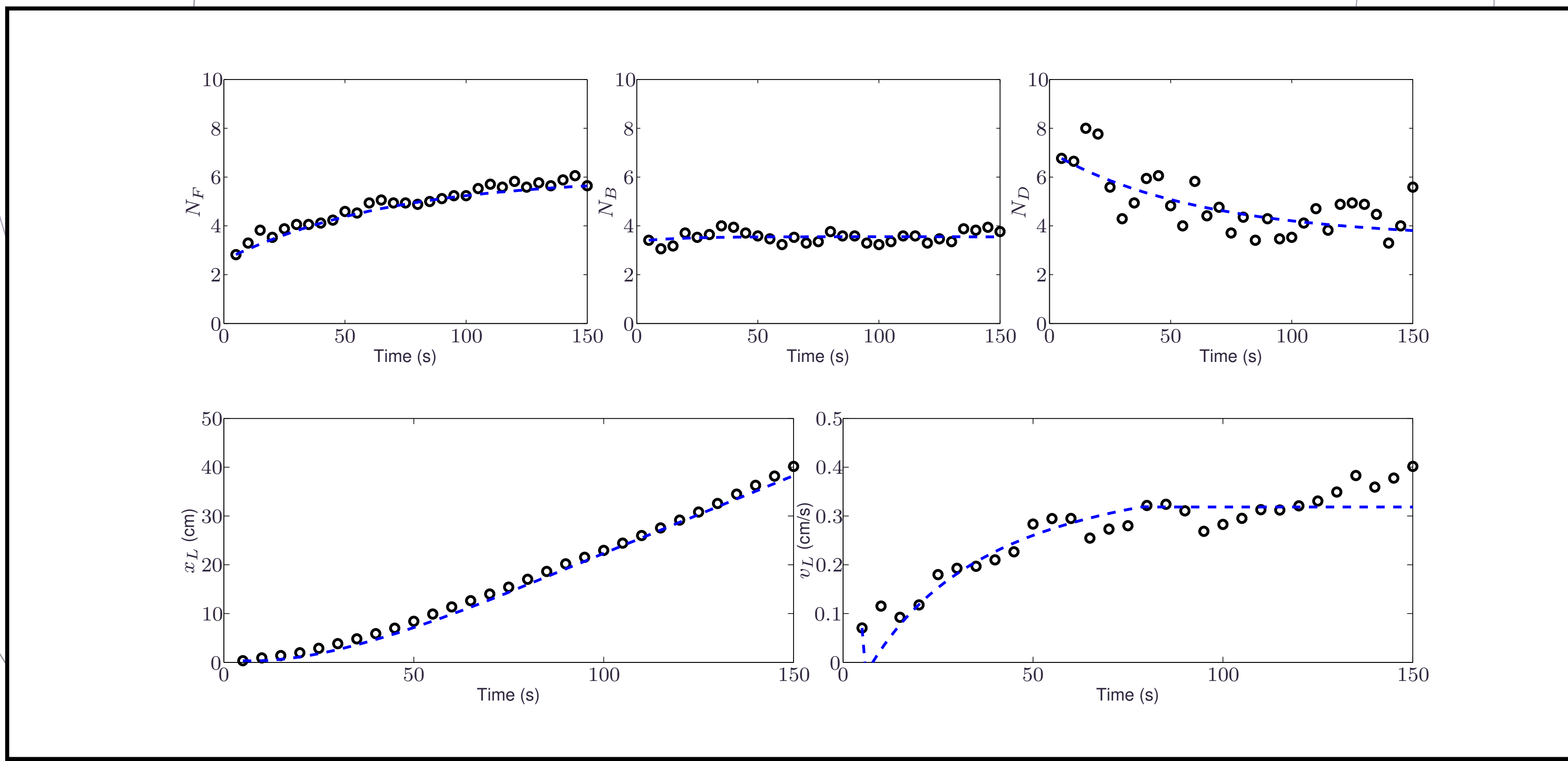
$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{E}(\psi(\vec{x})) = \mathrm{E}(L\psi(\vec{x})).$$

So arbitrary moment dynamics can be derived from SHS model.

Behavioral switching rates and control parameters can be found by fitting moment dynamics to statistics.

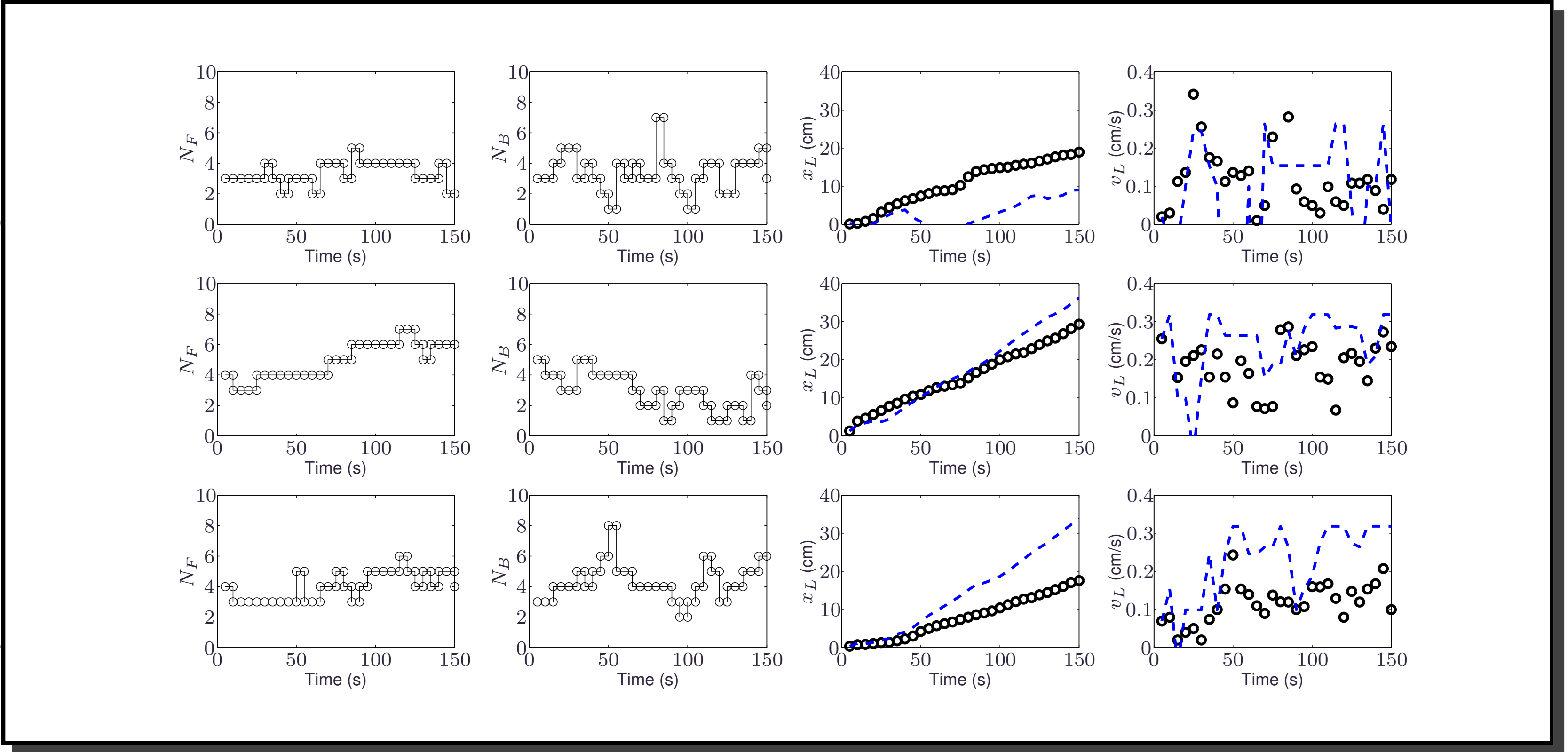
# Fitting Results
(Kumar et al. 2013)



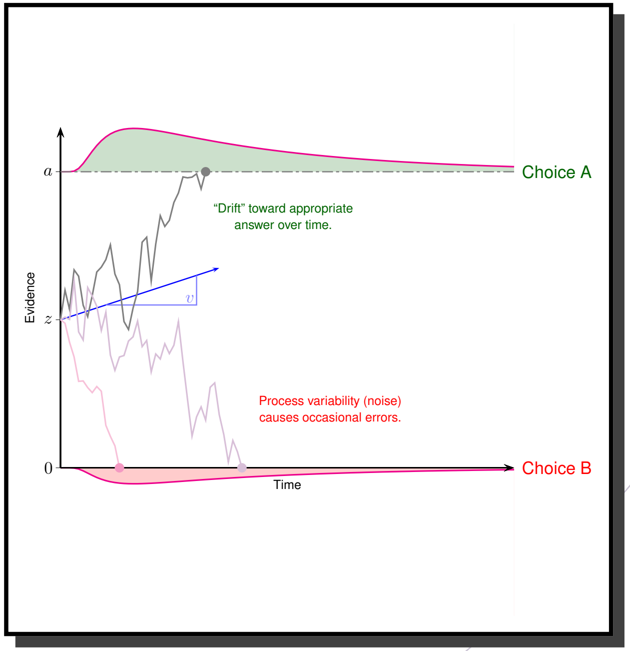


# Fitting Results
## (Kumar et al. 2013)





T. P. Pavlic, S. C. Pratt, "Speed–accuracy tradeoffs in *Temnothorax rugatulus* ants: sequential-sampling models of quorum detection while house hunting" (IUSSI-NAS 2012, SMB 2013).



# Cooperative Task-Processing Networks*

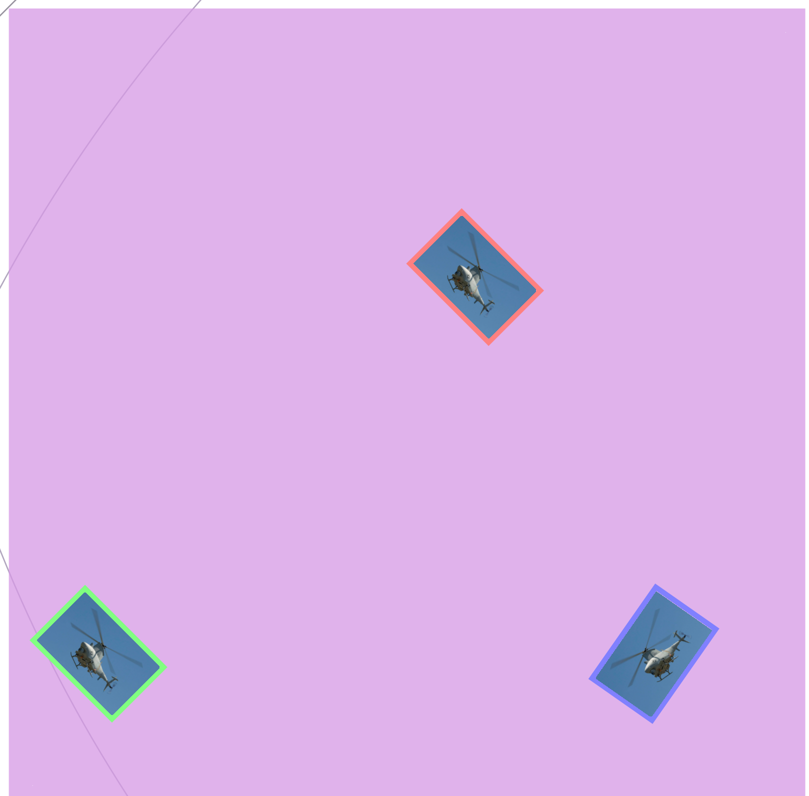
Theodore P. Pavlic[1] and Kevin M. Passino[2]

[1] Department of Computer Science and Engineering (pavlic.3@osu.edu)
[2] Department of Electrical and Computer Engineering (passino@ece.osu.edu)
The Ohio State University, Columbus, OH 43210 USA

T. P. Pavlic and K. M. Passino, “Cooperative task-processing networks,” in *Proceedings of the Second International Workshop on Networks of Cooperating Objects, CONET 2011*, Chicago, IL, USA, April 11, 2011.

# Application: Cooperative patrol
(Pavlic and Passino 2011)



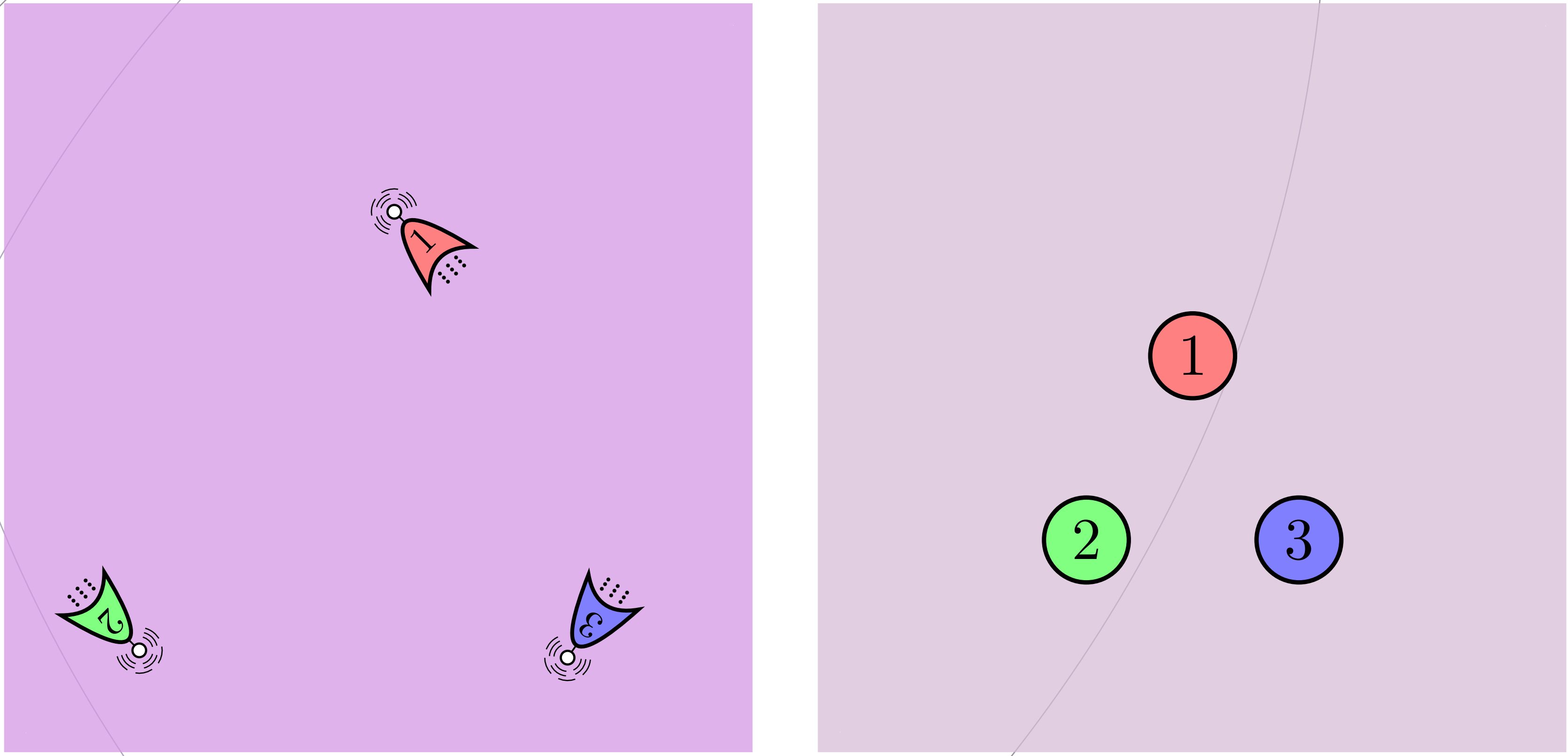


Example AAV Application:

Three MQ-8 Firescouts on patrol

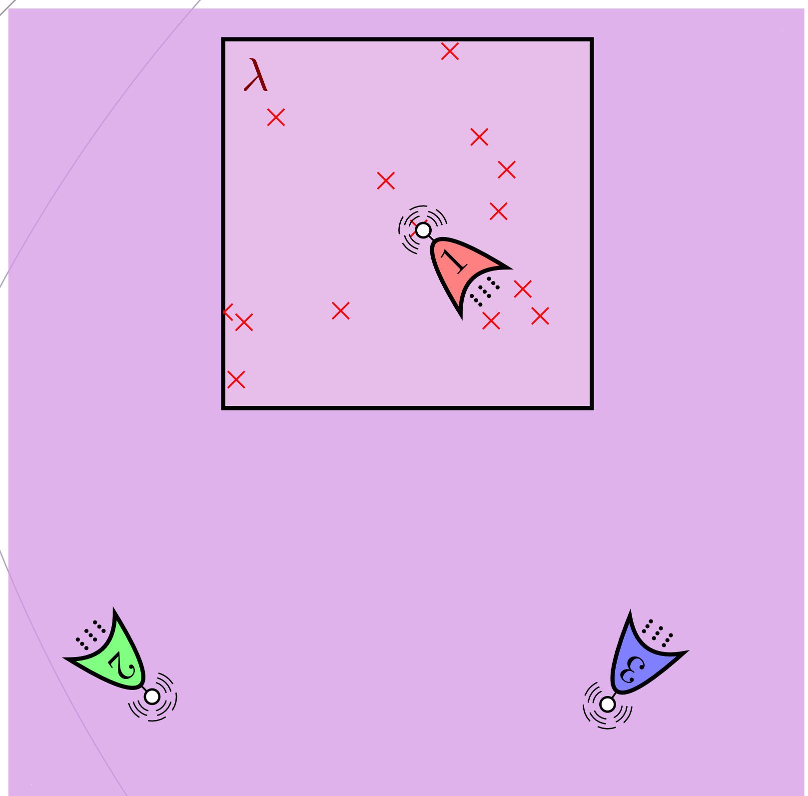
# Application: Cooperative patrol
(Pavlic and Passino 2011)

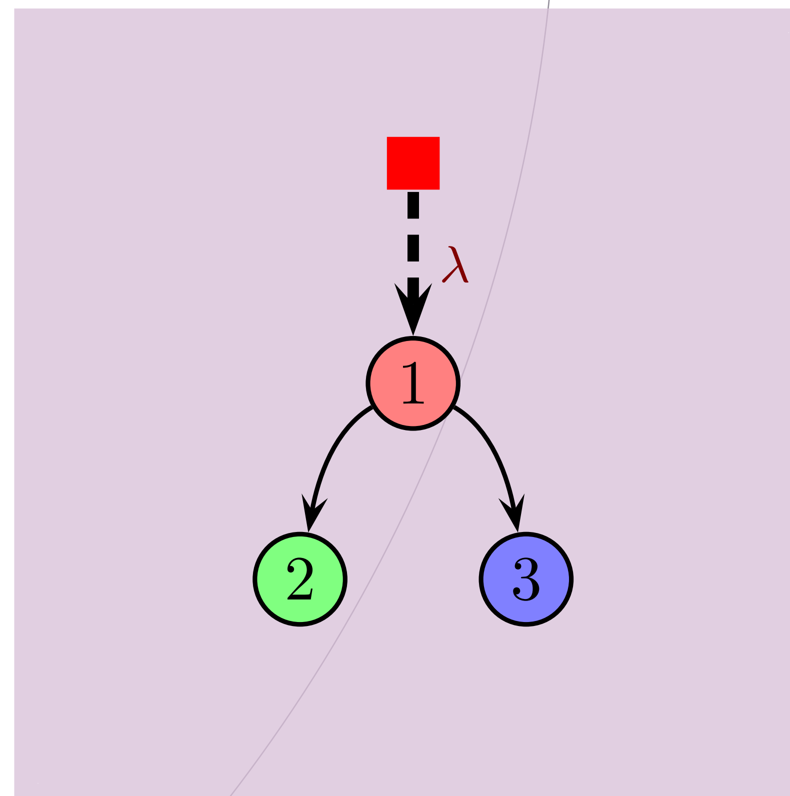




# Application: Cooperative patrol
(Pavlic and Passino 2011)



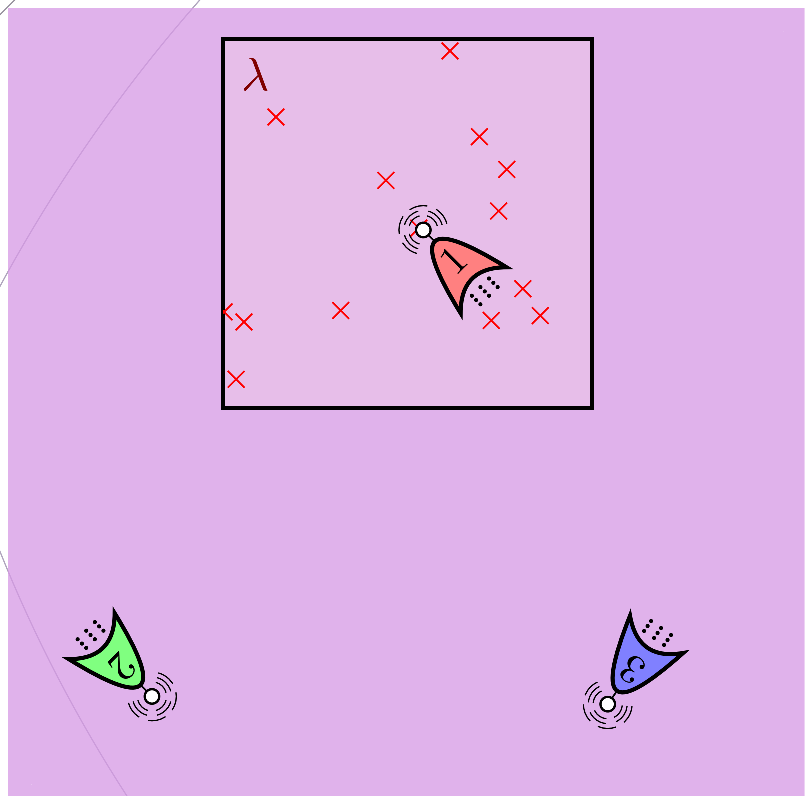


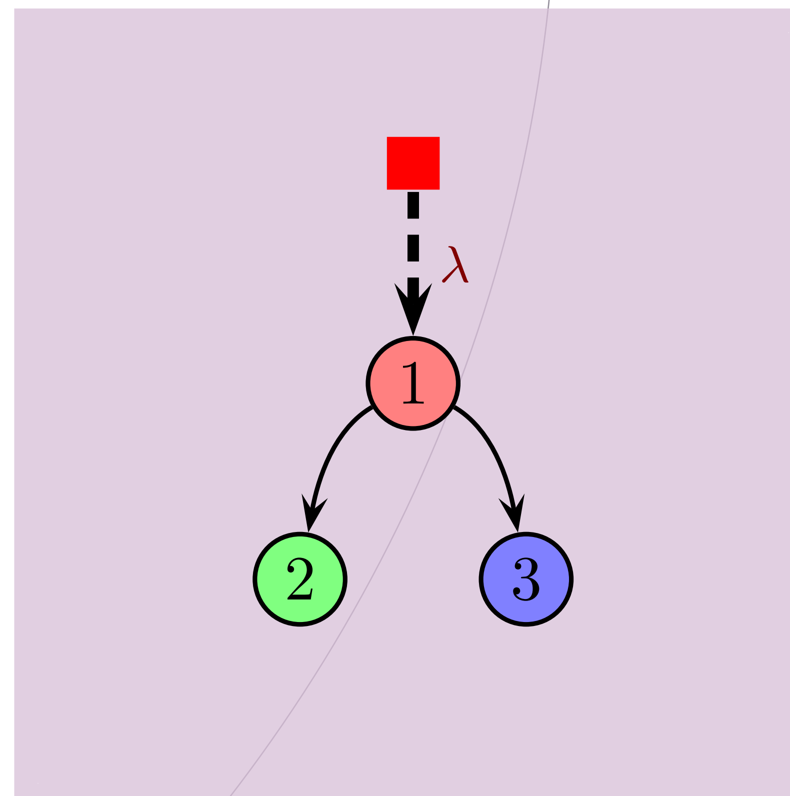



# Application: Cooperative patrol

(Pavlic and Passino 2011)





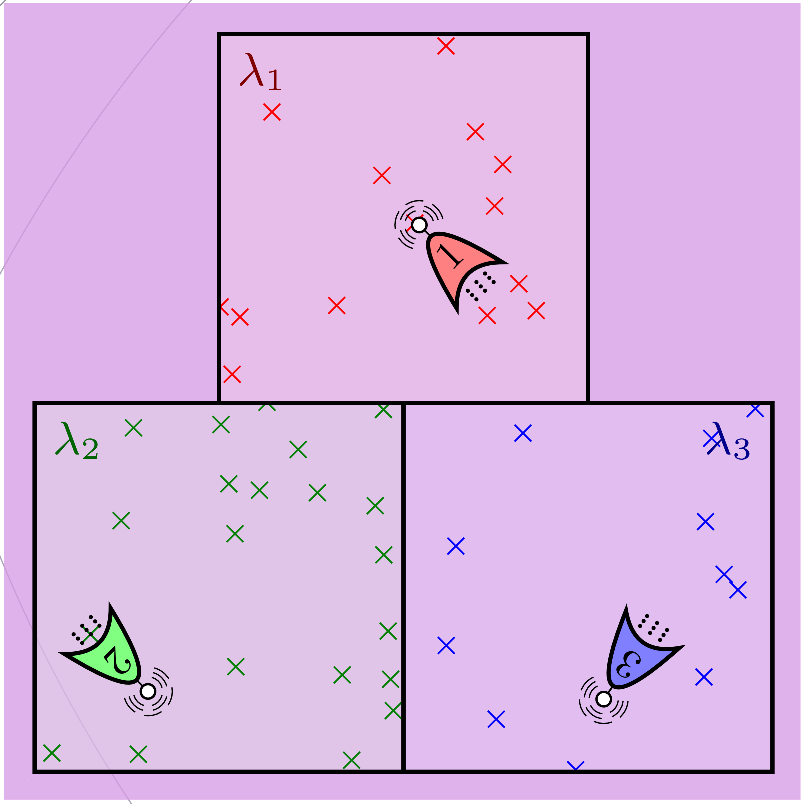



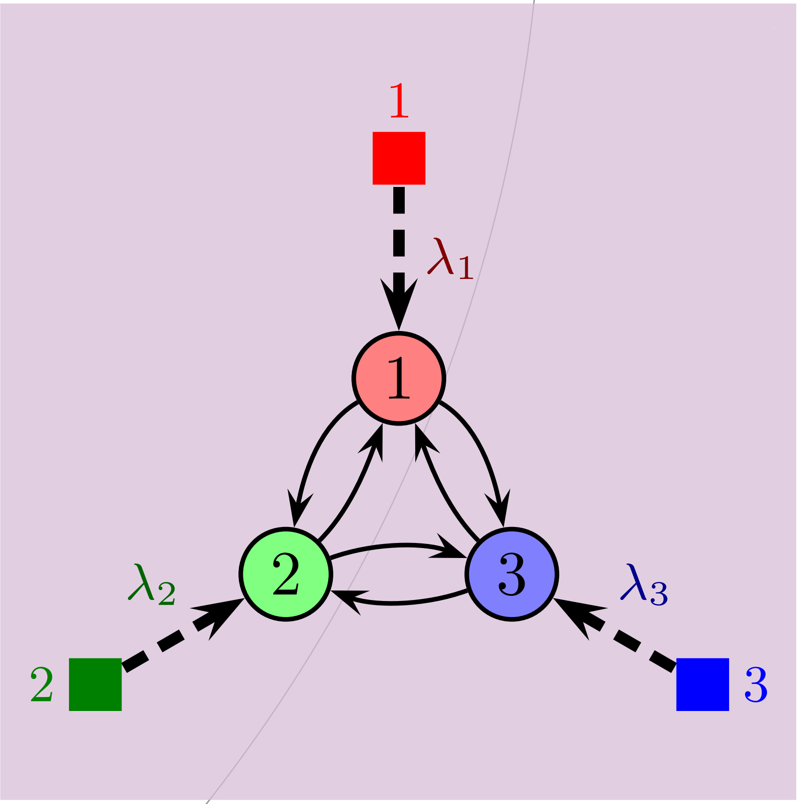

- Task-processing network (TPN): *conveyor* 1 and *cooperators* 2 and 3

# Application: Cooperative patrol

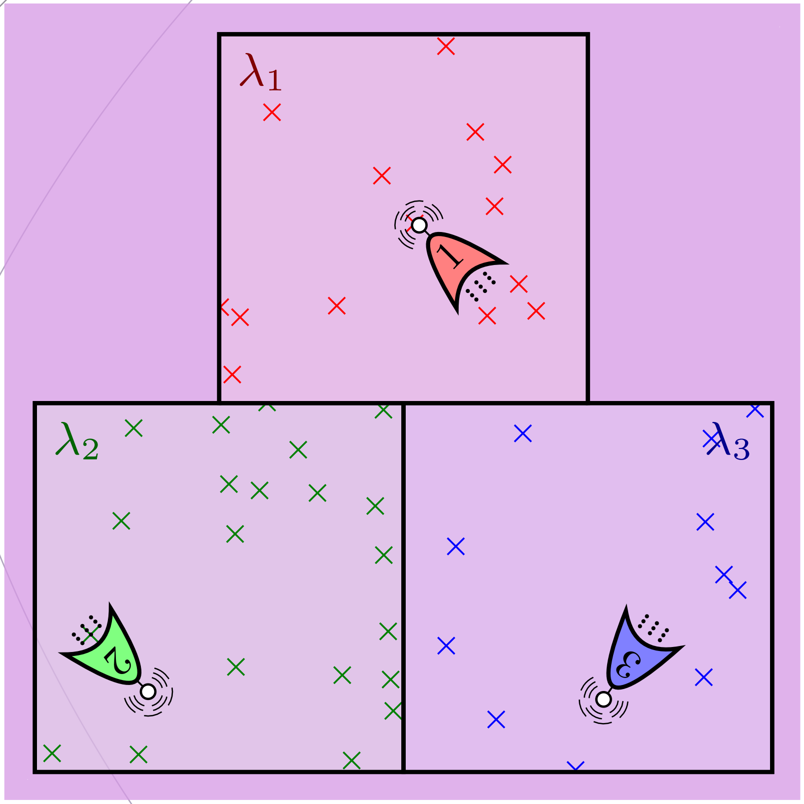
(Pavlic and Passino 2011)

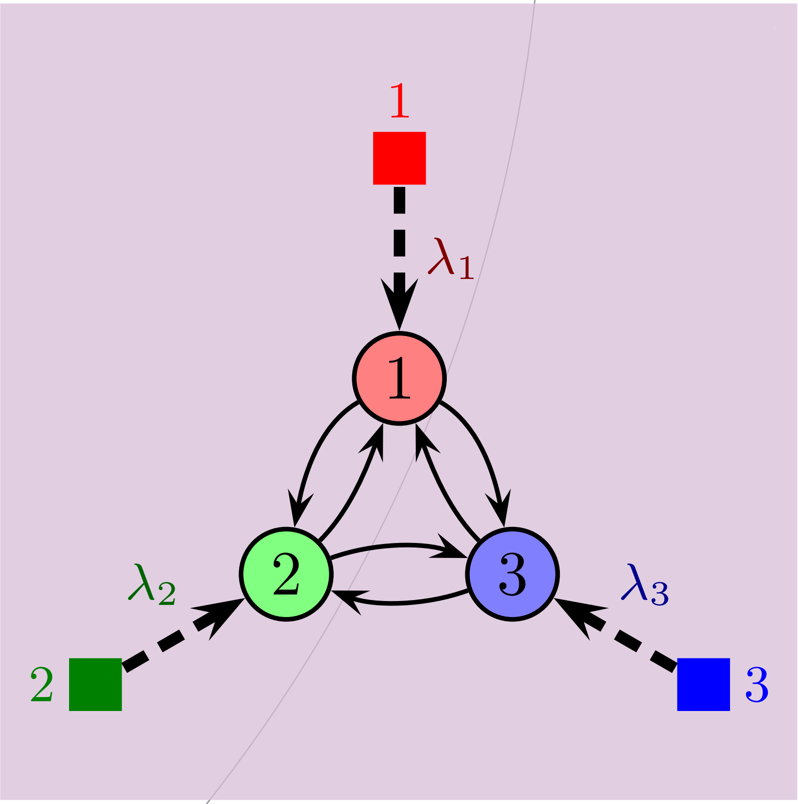





- Task-processing network (TPN): *conveyor* 1 and *cooperators* 2 and 3
- Each can be both conveyor and cooperator simultaneously




- Task-processing network (TPN): *conveyor* 1 and *cooperators* 2 and 3
- Each can be both conveyor and cooperator simultaneously
- Policy should be decentralized but still share load

For cooperator $i \in \mathcal{C}$, its local rate of gain

$$U_i(\vec{\gamma}) \triangleq \overbrace{-c_i \underbrace{\left( \prod_{j \in \mathcal{C}_i} (1-\gamma_j) \right)}_{\Pr(\text{No } \mathcal{C}_i \text{ volunteers} | i \text{ advertisement})}}^{\text{Conveyor costs}} + \underbrace{\gamma_i \sum_{j \in \mathcal{V}_i} \left( \qquad - \overbrace{P_s(j|i)}^{\Pr(i \text{ awarded task from } j | i \text{ volunteers})} c_{ij} \right)}_{\text{Cooperator part:}}$$

Costs of local processing on $i \in \mathcal{V}$:

$$c_i \triangleq \sum_{k \in \mathcal{Y}_i} \lambda_i^k c_i^k$$

Costs and benefits to $i \in \mathcal{C}$ for volunteering for tasks exported from $j \in \mathcal{V}_i$:

$$c_{ij} \triangleq \sum_{k \in \mathcal{Y}_j} \lambda_j^k c_{ij}^k$$

# Cooperation game
## (Pavlic and Passino 2011)

For cooperator $i \in \mathcal{C}$, its local rate of gain

$$U_i(\vec{\gamma}) \triangleq \overbrace{-c_i \underbrace{\left( \prod_{j \in \mathcal{C}_i} (1-\gamma_j) \right)}_{\Pr(\text{No } \mathcal{C}_i \text{ volunteers} | i \text{ advertisement})}}^{\text{Conveyor costs}} + \underbrace{\gamma_i \sum_{j \in \mathcal{V}_i} \left( p_{ij}(Q_j) - \overbrace{P_s(j|i)}^{\Pr(i \text{ awarded task from } j | i \text{ volunteers})} c_{ij} \right)}_{\text{Cooperator part: } \gamma_i,\, Q_j \text{ vary with } \gamma_i}$$

**Costs of local processing on $i \in \mathcal{V}$:**

$$c_i \triangleq \sum_{k \in \mathcal{Y}_i} \lambda_i^k c_i^k$$

**Costs and benefits to $i \in \mathcal{C}$ for volunteering for tasks exported from $j \in \mathcal{V}_i$:**

$$c_{ij} \triangleq \sum_{k \in \mathcal{Y}_j} \lambda_j^k c_{ij}^k$$

$$p_{ij}(Q_j) \triangleq \sum_{k \in \mathcal{Y}_j} \lambda_j^k q_{ij}^k p_j^k(Q_j)$$

Payment functions added as stabilizing controls (“quantity” $Q_i \triangleq \sum_{j \in \mathcal{C}_i} \gamma_j$).

# Cooperation game
(Pavlic and Passino 2011)

For cooperator $i \in \mathcal{C}$, its local rate of gain

Decreasing-cost externality

$$U_i(\vec{\gamma}) \triangleq \overbrace{-c_i \underbrace{\left( \prod_{j \in \mathcal{C}_i} (1-\gamma_j) \right)}_{\Pr(\text{No } \mathcal{C}_i \text{ volunteers} | i \text{ advertisement})}}^{\text{Conveyor costs}} + \underbrace{\gamma_i \sum_{j \in \mathcal{V}_i} \left( p_{ij}(Q_j) - \overbrace{P_s(j|i)}^{\Pr(i \text{ awarded task from } j | i \text{ volunteers})} c_{ij} \right)}_{\text{Cooperator part: } \gamma_i,\, Q_j \text{ vary with } \gamma_i}$$

**Costs** of **local processing** on $i \in \mathcal{V}$:

$$c_i \triangleq \sum_{k \in \mathcal{Y}_i} \lambda_i^k c_i^k$$

Cournot oligopolies on a graph ($\propto$ jury volunteering)

**Costs** and **benefits** to $i \in \mathcal{C}$ for **volunteering** for tasks exported from $j \in \mathcal{V}_i$:

$$c_{ij} \triangleq \sum_{k \in \mathcal{Y}_j} \lambda_j^k c_{ij}^k$$

$$p_{ij}(Q_j) \triangleq \sum_{k \in \mathcal{Y}_j} \lambda_j^k q_{ij}^k p_j^k(Q_j)$$

Payment functions added as stabilizing controls ("quantity" $Q_i \triangleq \sum_{j \in \mathcal{C}_i} \gamma_j$).

- Totally asynchronous parallel computation of $\vec{\gamma}^*$ by local gradient ascent:

# Totally asynchronous solver
(Pavlic and Passino 2011)



- Totally asynchronous parallel computation of $\vec{\gamma}^*$ by local gradient ascent:
  - Agents iterate asynchronously.
  - Each agent operates on a possibly outdated copy of $\vec{\gamma}$.

# Totally asynchronous solver
(Pavlic and Passino 2011)



- Totally asynchronous parallel computation of $\vec{\gamma}^*$ by local gradient ascent:
  - Agents iterate asynchronously.
  - Each agent operates on a possibly outdated copy of $\vec{\gamma}$.
  - If synchronous transition mapping is a contraction with respect to maximum norm

$$\|\vec{\gamma}\|_\infty \triangleq \max_{i \in \mathcal{C}} \{|\gamma_i|\},$$

    then a unique Nash equilibrium exists and is asymptotically stable by totally asynchronous distributed gradient ascent iterations (Bertsekas and Tsitsiklis 1997).

# Totally asynchronous solver

## (Pavlic and Passino 2011)

- Totally asynchronous parallel computation of $\vec{\gamma}^*$ by local gradient ascent:
  - Agents iterate asynchronously.
  - Each agent operates on a possibly outdated copy of $\vec{\gamma}$.
  - If synchronous transition mapping is a contraction with respect to maximum norm

$$\|\vec{\gamma}\|_\infty \triangleq \max_{i \in \mathcal{C}} \{|\gamma_i|\},$$

    then a unique Nash equilibrium exists and is asymptotically stable by totally asynchronous distributed gradient ascent iterations (Bertsekas and Tsitsiklis 1997).
- Constraints on topology and payment functions ensure contraction.
  - Diagonal-dominance/convexity argument.
  - Network structure ensures dominance.

# Totally asynchronous projected gradient ascent



Define $T : [0,1]^n \to [0,1]^n$ by $T(\vec{\gamma}) \triangleq (T_1(\vec{\gamma}), T_2(\vec{\gamma}), \ldots, T_n(\vec{\gamma}))$ where, for each $i \in \mathcal{C}$,

$$T_i(\vec{\gamma}) \triangleq \min\{\gamma_i^{\text{max}}, \max\{0, \gamma_i + \sigma_i \nabla_i U_i(\vec{\gamma})\}\}$$

Projected gradient ascent

# Totally asynchronous projected gradient ascent



Define $T : [0,1]^n \to [0,1]^n$ by $T(\vec{\gamma}) \triangleq (T_1(\vec{\gamma}), T_2(\vec{\gamma}), \ldots, T_n(\vec{\gamma}))$ where, for each $i \in \mathcal{C}$,

$$T_i(\vec{\gamma}) \triangleq \min\{\gamma_i^{\text{max}}, \max\{0, \gamma_i + \sigma_i \nabla_i U_i(\vec{\gamma})\}\},$$

where

$$\frac{1}{\sigma_i} \geq 2|\mathcal{V}_i| \max_{k \in \mathcal{V}_i} |p'_{ik}(0)|$$

for all $\vec{\gamma} \in [0,1]^n$.

# Totally asynchronous projected gradient ascent



Define $T : [0,1]^n \to [0,1]^n$ by $T(\vec{\gamma}) \triangleq (T_1(\vec{\gamma}), T_2(\vec{\gamma}), \ldots, T_n(\vec{\gamma}))$ where, for each $i \in \mathcal{C}$,

$$T_i(\vec{\gamma}) \triangleq \min\{\gamma_i^{\text{max}}, \max\{0, \gamma_i + \sigma_i \nabla_i U_i(\vec{\gamma})\}\},$$

where

$$\frac{1}{\sigma_i} \geq 2|\mathcal{V}_i| \max_{k \in \mathcal{V}_i} |p'_{ik}(0)|$$

for all $\vec{\gamma} \in [0,1]^n$. If

$$\min_{j \in \mathcal{V}_i} |p'_{ij}(|\mathcal{C}_j|)| > \left(|\mathcal{V}_i| - \frac{1}{2}\right) \max_{j \in \mathcal{V}_i} |c_{ij}| \qquad \text{for all } i \in \mathcal{C},$$

then the totally asynchronous distributed iteration (TADI) sequence $\{\vec{\gamma}(t)\}$ generated with mapping $T$ and the outdated estimate sequence $\{\vec{\gamma}^i(t)\}$ for all $i \in \mathcal{C}$ each converge to the unique Nash equilibrium $\vec{\gamma}^*$ of the cooperation game.

# Totally asynchronous projected gradient ascent

Define $T : [0,1]^n \to [0,1]^n$ by $T(\vec{\gamma}) \triangleq (T_1(\vec{\gamma}), T_2(\vec{\gamma}), \ldots, T_n(\vec{\gamma}))$ where, for each $i \in \mathcal{C}$,

$$T_i(\vec{\gamma}) \triangleq \min\{\gamma_i^{\text{max}}, \max\{0, \gamma_i + \sigma_i \nabla_i U_i(\vec{\gamma})\}\},$$

where

$$\frac{1}{\sigma_i} \geq 2|\mathcal{V}_i| \max_{k \in \mathcal{V}_i} |p'_{ik}(0)|$$

for all $\vec{\gamma} \in [0,1]^n$. If

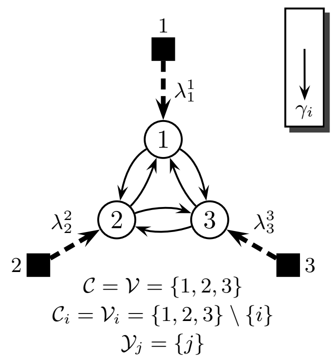
$$\overbrace{\min_{j \in \mathcal{V}_i} |p'_{ij}(|\mathcal{C}_j|)|}^{\text{Benefit}} > \overbrace{\left(|\mathcal{V}_i| - \frac{1}{2}\right)}^{1/(\text{Relatedness})} \overbrace{\max_{j \in \mathcal{V}_i} |c_{ij}|}^{\text{Cost}} \quad \text{for all } i \in \mathcal{C},$$

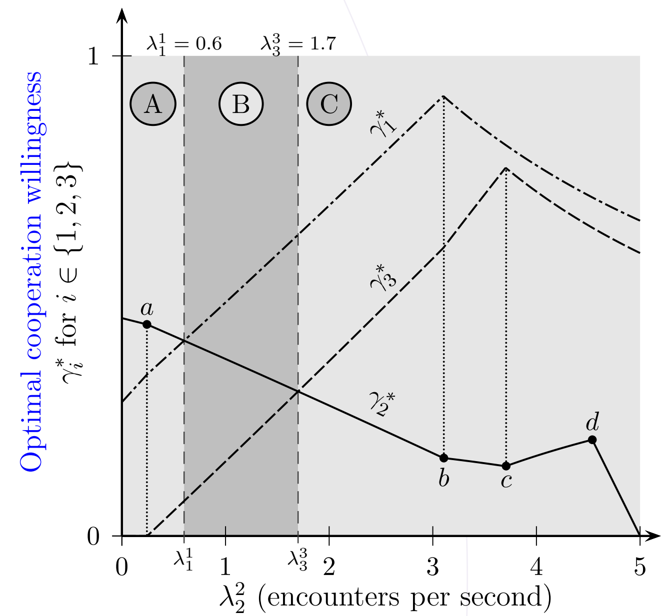
∼ Hamilton's rule on networks

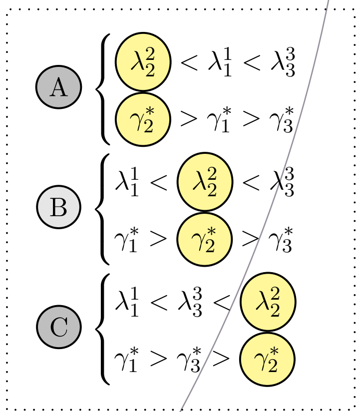
then the totally asynchronous distributed iteration (TADI) sequence $\{\vec{\gamma}(t)\}$ generated with mapping $T$ and the outdated estimate sequence $\{\vec{\gamma}^i(t)\}$ for all $i \in \mathcal{C}$ each converge to the unique Nash equilibrium $\vec{\gamma}^*$ of the cooperation game.

# Results: Cyclic feedback

(Pavlic and Passino 2011)

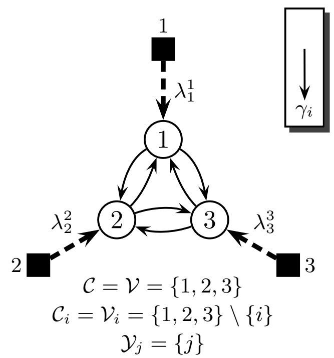


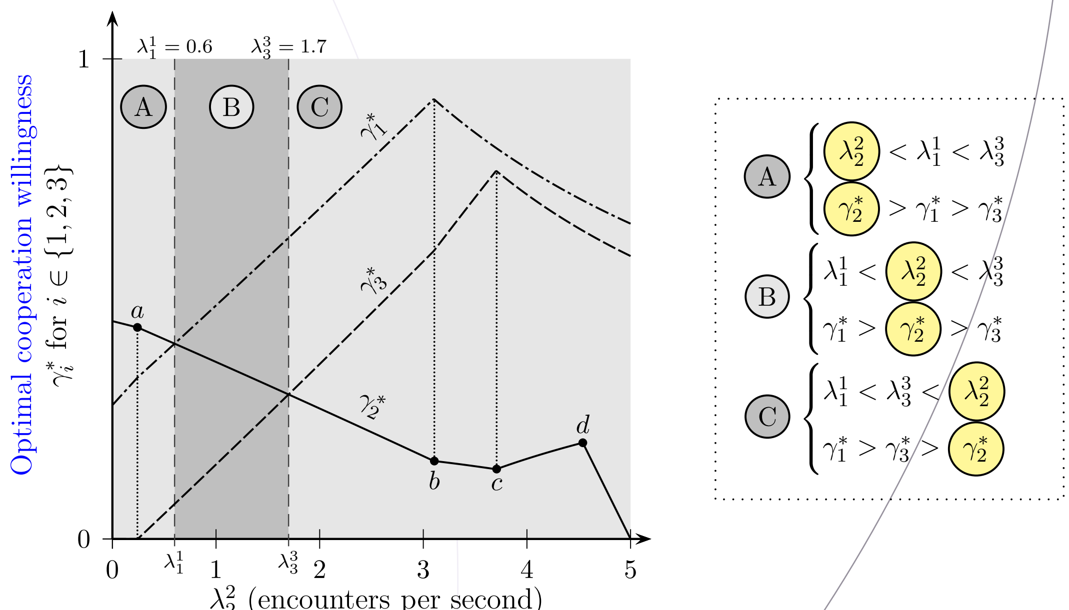




## Equilibrium in AAV patrol scenario

(simulation results above match analytic predictions)



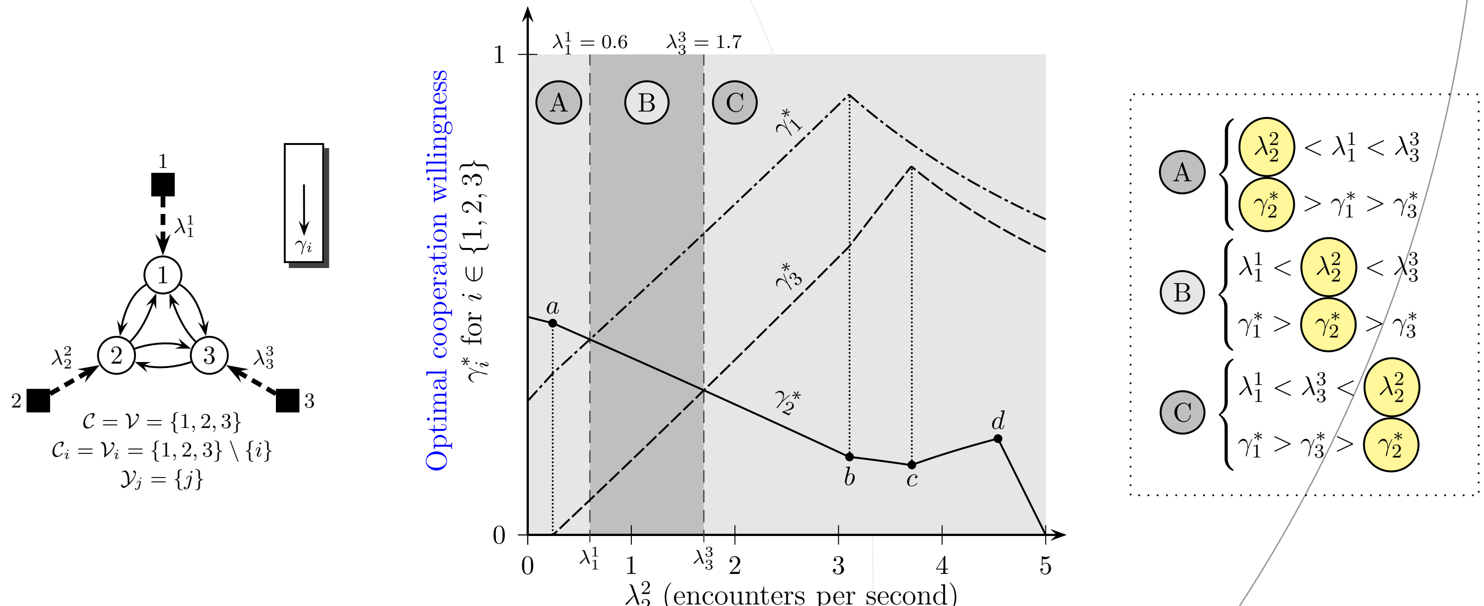




Equilibrium in AAV patrol scenario
(simulation results above match analytic predictions)

- Increases in one encounter rate (e.g., $\lambda_2$) cause equilibrium shift so neighbors (e.g., 1 and 3) help more and agent (e.g., 2) helps less

# Results: Cyclic feedback
(Pavlic and Passino 2011)

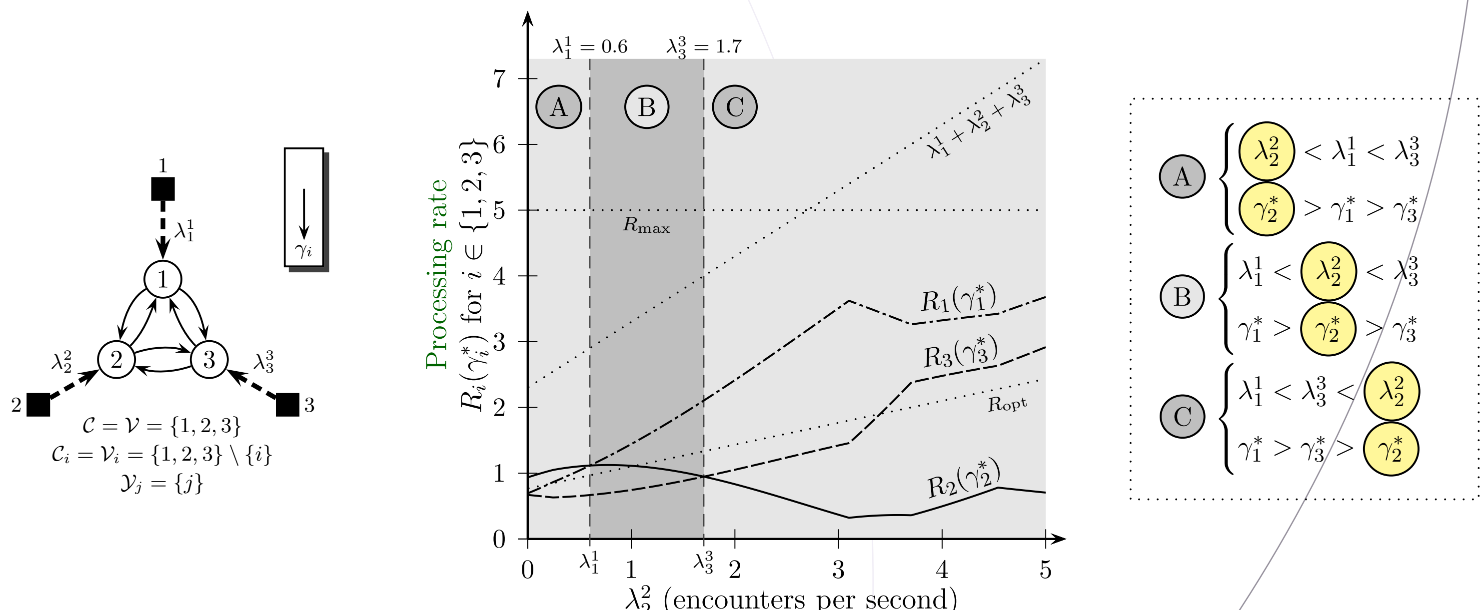





Equilibrium in AAV patrol scenario
(simulation results above match analytic predictions)

- Increases in one encounter rate (e.g., $\lambda_2$) cause equilibrium shift so neighbors (e.g., 1 and 3) help more and agent (e.g., 2) helps less
- Emergence due to market coupling from network cycles

# Results: Cyclic feedback
(Pavlic and Passino 2011)




Equilibrium in AAV patrol scenario
(simulation results above match analytic predictions)

- Increases in one encounter rate (e.g., $\lambda_2$) cause equilibrium shift so neighbors (e.g., 1 and 3) help more and agent (e.g., 2) helps less
- Emergence due to market coupling from network cycles

# Stochastic Robotics: Conclusions



- Stochastic programming can model failures and blend programs for optimal performance (Napp and Klavins 2011)

# Stochastic Robotics: Conclusions



- Stochastic programming can model failures and blend programs for optimal performance (Napp and Klavins 2011)
- Chemical reaction networks provide reduced-order analysis frameworks for design and control (Berman et al. 2011a,b; Matthey et al. 2009)

# Stochastic Robotics: Conclusions

- Stochastic programming can model failures and blend programs for optimal performance (Napp and Klavins 2011)
- Chemical reaction networks provide reduced-order analysis frameworks for design and control (Berman et al. 2011a,b; Matthey et al. 2009)
- Stochastic hybrid system models combine stochastic behavioral switching with continuous-time dynamics (Kumar et al. 2013)

# Stochastic Robotics: Conclusions



- Stochastic programming can model failures and blend programs for optimal performance (Napp and Klavins 2011)
- Chemical reaction networks provide reduced-order analysis frameworks for design and control (Berman et al. 2011a,b; Matthey et al. 2009)
- Stochastic hybrid system models combine stochastic behavioral switching with continuous-time dynamics (Kumar et al. 2013)
- Numerical solvers stabilize adaptive mixed-Nash equilibria on networks (Pavlic and Passino 2011)

# Stochastic Robotics: Conclusions



- Stochastic programming can model failures and blend programs for optimal performance (Napp and Klavins 2011)
- Chemical reaction networks provide reduced-order analysis frameworks for design and control (Berman et al. 2011a,b; Matthey et al. 2009)
- Stochastic hybrid system models combine stochastic behavioral switching with continuous-time dynamics (Kumar et al. 2013)
- Numerical solvers stabilize adaptive mixed-Nash equilibria on networks (Pavlic and Passino 2011)
- Acknowledgments:

# Stochastic Robotics: Conclusions

Introduction
Guarded Command Programming with Rates
Reaction Networks
Cooperative Transport
Cooperative Task Processing
Conclusions

- Stochastic programming can model failures and blend programs for optimal performance (Napp and Klavins 2011)
- Chemical reaction networks provide reduced-order analysis frameworks for design and control (Berman et al. 2011a,b; Matthey et al. 2009)
- Stochastic hybrid system models combine stochastic behavioral switching with continuous-time dynamics (Kumar et al. 2013)
- Numerical solvers stabilize adaptive mixed-Nash equilibria on networks (Pavlic and Passino 2011)
- Acknowledgments:
- Questions? Comments?