Outline

- Internet congestion control
  - Distributed optimization with separable objective function
- Two miracles in resource allocation in wireless networks
  - Distributed optimization with un-separable objective function, and without message passing
  - Power control
  - Carrier Sensing Multiple Access
- Parallel computations
  - Joint consensus and gradient descent methods
  - Just gradient descent
- Colorings
  - Combinatorial optimization: a sampling approach
- Distributed gradient free optimization

Coloring
Distributed combinatorial optimization:
A sampling approach

Simulation

Consists in producing samples from a distribution π over Ω

Example: Markov Chain Monte Carlo (MCMC) method, design a Markov chain whose stationary distribution is π
Reversible Markov chains: Metropolis, Glauber Dynamics

Optimization via simulation

Objective: maximize $U(s)$ over $s \in \Omega$

Solution: sample from $\pi^{\lambda}(s) = \frac{1}{Z(\lambda)} \lambda^{U(s)}, \quad s \in \Omega$

(fugacity λ has to be large enough)

Glauber Dynamics algorithm: construct a Markov chain where transitions from s to s' occur at rate $P(s, s') = \pi^{\lambda}(s')$
(the chain is reversible and has stationary distribution π^{λ})

Distributed optimization

Objective: solve the following optimization problem

$$\max \; U(s) = \sum_i U_i(s)$$

over $s = (s_1, \dots, s_N) \in \Omega = \Omega_1 \times \cdots \times \Omega_N$ finite

Distributed pay-off based solution:

- time is divided into periods;
- $s(t)$ is the variable in period t;
- at the end of period t, agent i observes her pay-off $U_i\big(s(t)\big)$, and decides to update her action $s_i(t) \to s_i(t+1)$.

For which choice of objective function can we design a decentralized pay-off based solution?

Separable objective functions

Problems admitting decentralized pay-off based solutions:

1. Fully separable objective

$$\max \sum_i U_i(s_i)$$

2. Separable objective with (un-separable) constraints

$$\max \sum_i U_i(s_i) 1_{\{s \in \Omega_f\}}$$

$$\Omega_f \subset \Omega$$

Glauber Dynamics

Single-site Glauber Dynamics algorithm:

At the beginning of period t:

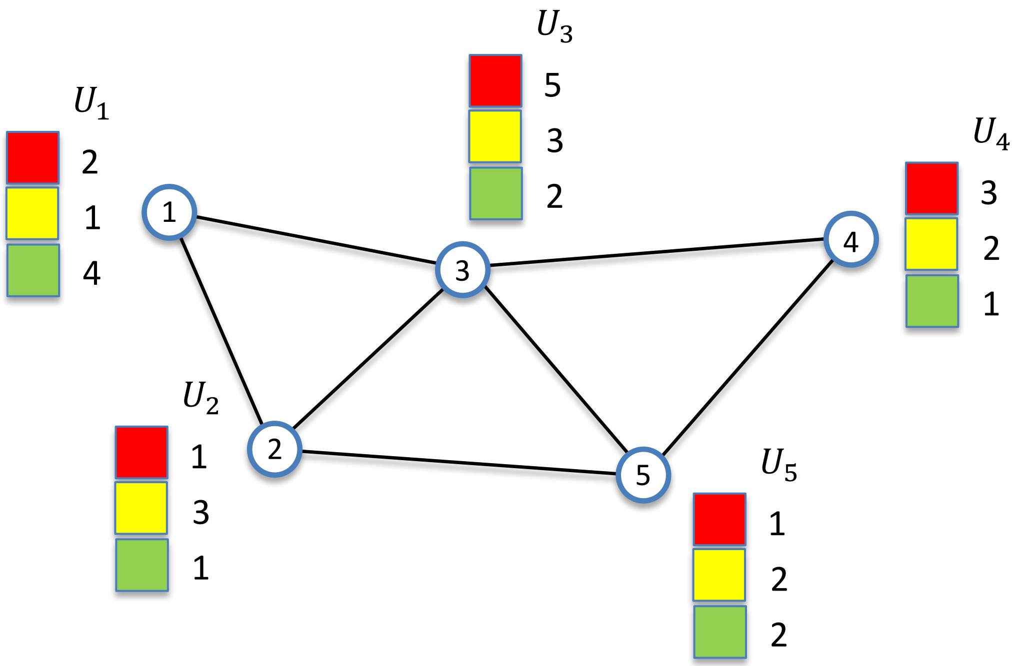
1. select an agent uniformly at random, say i
2. the agent updates her action to $s_i(t+1)$ according to distribution:

$$\alpha(s_i') = \frac{1_{\left(s_i', s_{-i}(t)\right)\in\Omega_f}\lambda^{U_i(s_i')}}{\sum_a 1_{\left(a, s_{-i}(t)\right)\in\Omega_f}\lambda^{U_i(a)}}$$

Under the above algorithm, $s(t)$ is a reversible Markov chain with steady state distribution

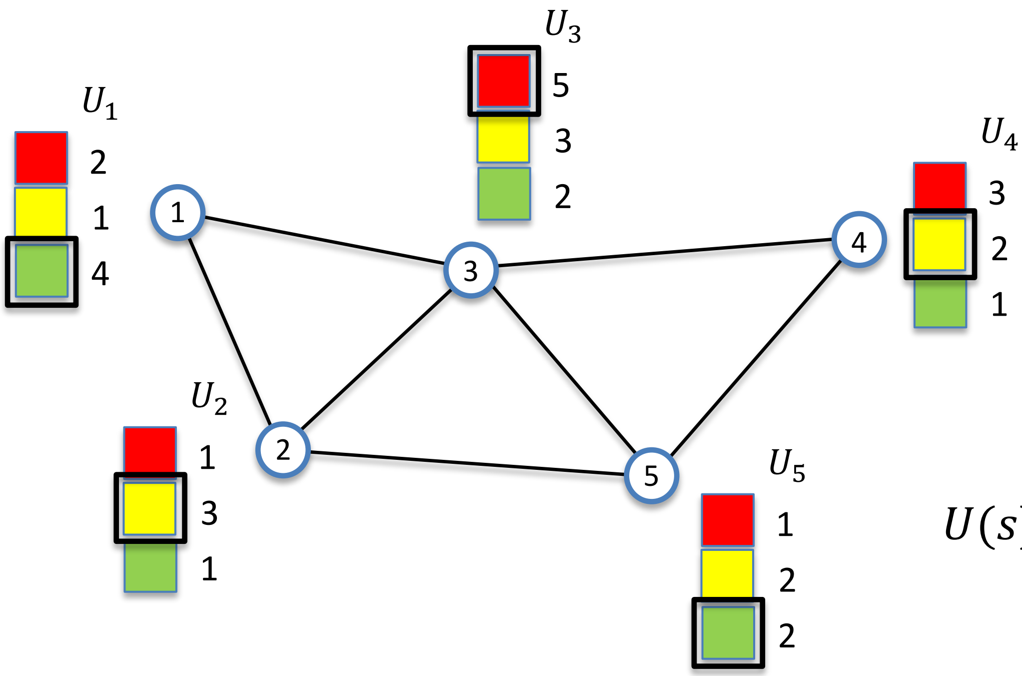
$$\pi^{\lambda}(s) \sim 1_{s\in\Omega_f}\lambda^{\sum_i U_i(s_i)}$$

Ex: Preferential graph colouring



Problem: find a proper colouring maximizing the sum of utilities

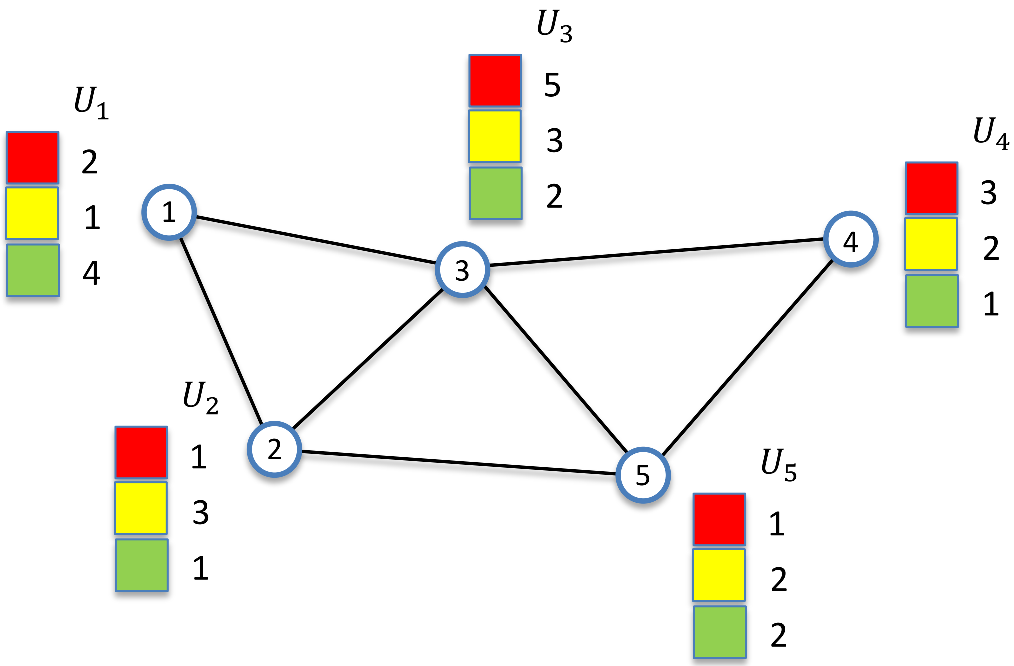
Ex: Preferential graph colouring



$$U(s) = \sum_i U_i(s_i) = 16$$

Problem: find a proper colouring maximizing the sum of utilities

Ex: Preferential graph colouring



Problem: find a proper colouring maximizing the sum of utilities
Application: optimal channel assignment in wireless networks

Mixing time

Definition: Let $s(t)$ be an irreducible positive recurrent Markov chain with stead-state distribution π. Let $\pi_{s_0}(t)$ be the distribution of $s(t)$ when starting at s_0. For $\epsilon > 0$,

$$t_{\mathrm{mix}}(\epsilon) = \inf\{t \geq 0: \forall s_0, ||\pi_{s_0}(t) - \pi||_{tv} < \epsilon\}$$

Phase transition: The chain is fast mixing if $t_{\mathrm{mix}}(\epsilon)$ is polynomial in the dimension of s. Usually, Glauber Dynamics algorithms are fast mixing if λ is small enough.

Literature: Mixing time of GD algorithms is generally an open problem; many interesting papers in maths and statistical physics (see *Markov chain and Mixing time*, **Levine-Peres-Wilmer**, 2008).

Mixing time

Consider GD algorithm for preferential colouring. Define the following metric on Ω: $\rho(r,s) = \sum_i 1_{\{r_i \neq s_i\}}$.

For all $r, s \in \Omega$, $H(r,s) = \sum_{i\sim j} \frac{h_{r,s}(i)}{g_{r,s}(i)}$

$$h_{r,s}(i) = \sum_{a\in A_r(i)\setminus \mathrm{A}_s(i)} \lambda^{U_i(a)} \vee \sum_{a\in A_s(i)\setminus \mathrm{A}_r(i)} \lambda^{U_i(a)}$$

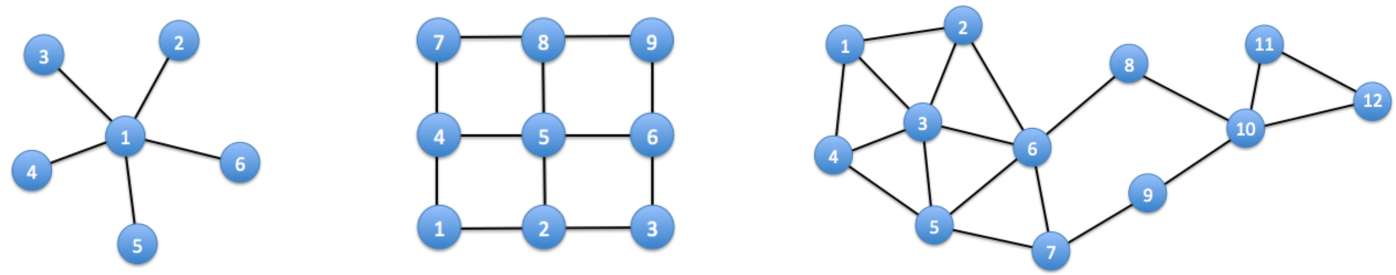
$$g_{r,s}(i) = \sum_{a\in A_r(i)} \lambda^{U_i(a)} \vee \sum_{a\in A_s(i)} \lambda^{U_i(a)}$$

Theorem If $\theta = 1 - \max_{r,s:\rho(r,s)=1} H(r,s) > 0$, then

$$t_{\mathrm{mix}}(\epsilon) \leq 1 + \frac{N}{\theta} \log N + \log \epsilon^{-1}$$

Numerical experiments

- Networks

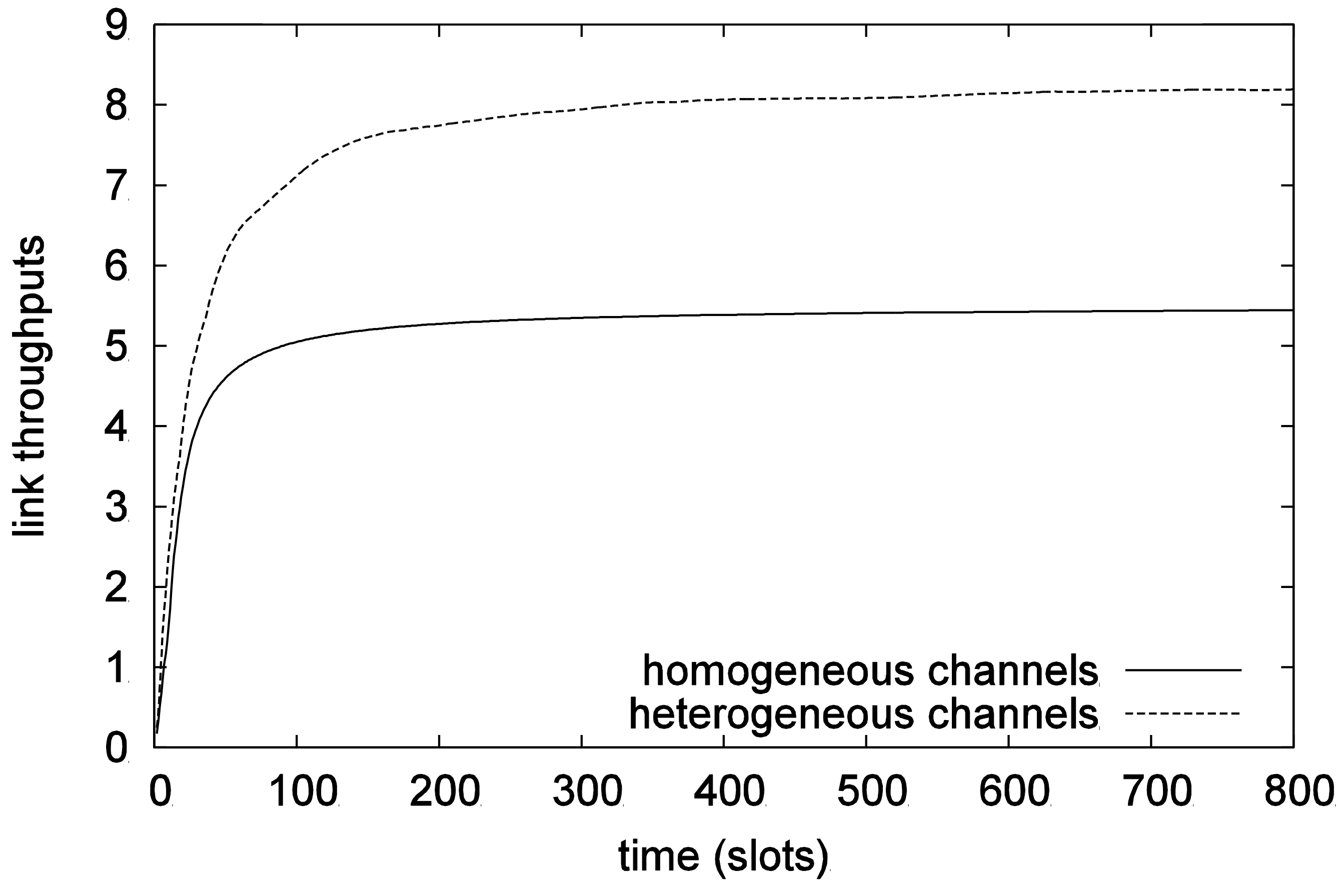


- Homogeneous vs heterogeneous (unif. on [1,10]) channels

- Logarithmic utility

- Performance metrics
  - Cumulative average throughput per link
  - Convergence time: the time it takes so that all cumulative throughputs to be within 5% of their limits
    Notice: different than mixing time.

Grid 16 links – 6 channels



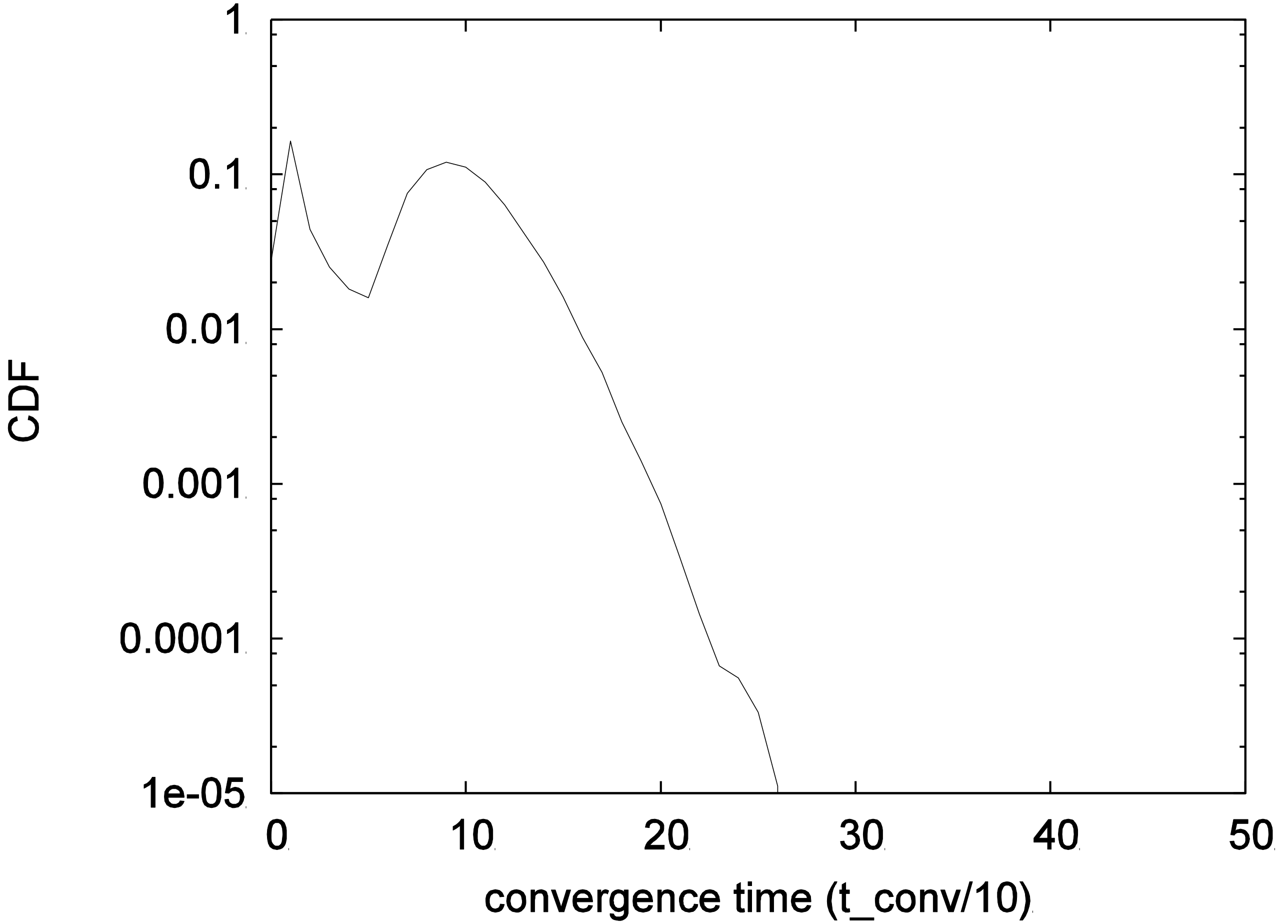
Star – 3 channels

- Convergence time: averaged over 9000 simulations

Number of links	**3**	**6**	**10**
Homogeneous channels	29.2	73.1	143.9
Heterogeneous channels	36.5	834.7	1756.1

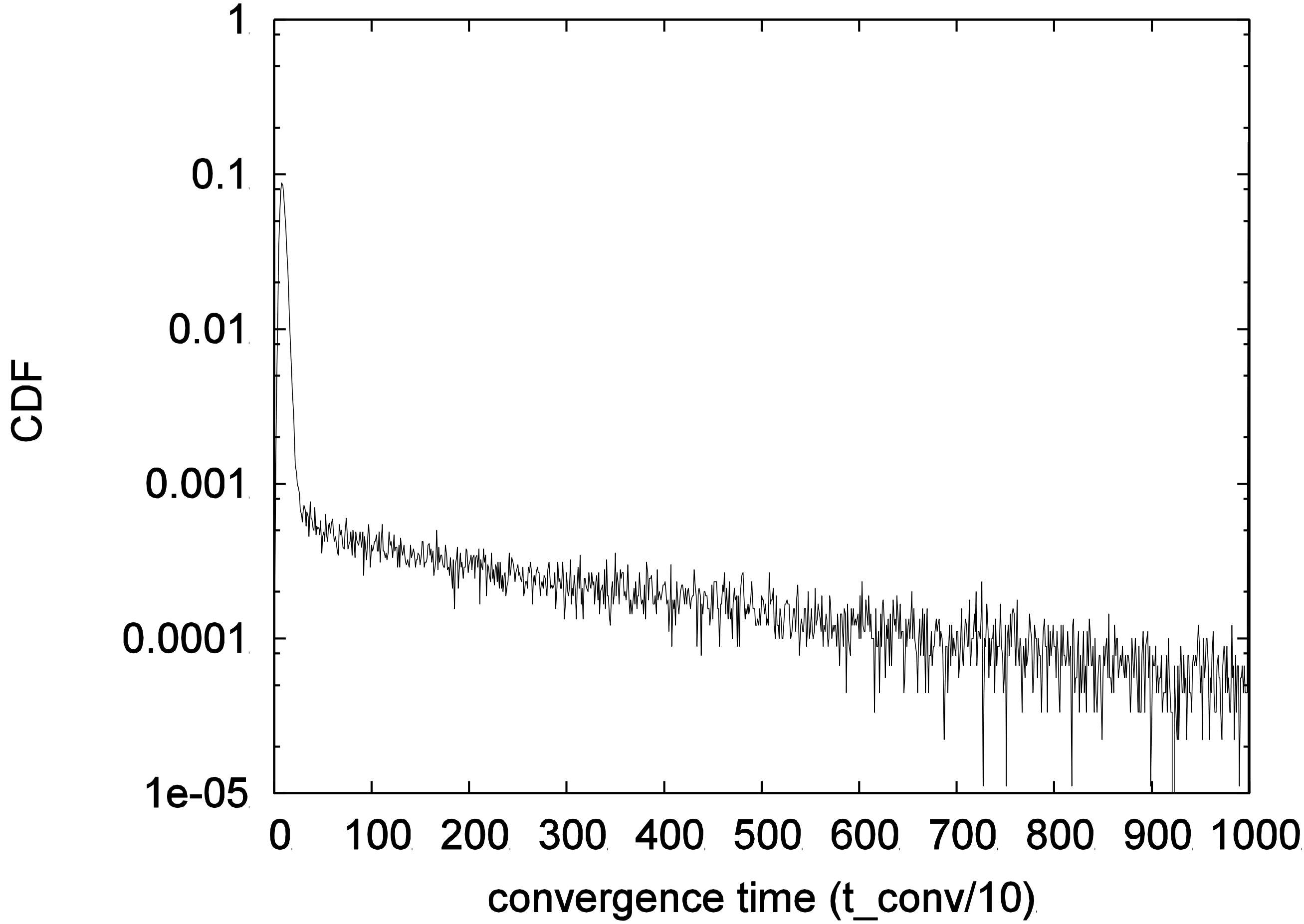
Star – 3 channels

- Homogeneous channels



Star – 3 channels

- Heterogeneous channels



General objective function

Objective: solve the following optimization problem

$$\max \ U(s) = \sum_i U_i(s)$$

$$\text{over } s = (s_1, \ldots, s_N) \in \Omega = \Omega_1 \times \cdots \times \Omega_N$$

Glauber Dynamics does not work, because updating one variable s_i does impact the utilities perceived by all agents.

General objective function

Objective: solve the following optimization problem

$$\max \;\; U(s) = \sum_i U_i(s)$$

$$\text{over } s = (s_1, \dots, s_N) \in \Omega = \Omega_1 \times \cdots \times \Omega_N$$

Can we design a decentralized pay-off based solution?

General objective function

Objective: solve the following optimization problem

$$\max \ U(s) = \sum_i U_i(s)$$

over $s = (s_1, \dots, s_N) \in \Omega = \Omega_1 \times \cdots \times \Omega_N$

Can we design a decentralized pay-off based solution?

Yes:

Achieving Pareto optimality through distributed learning, **Marden-Young-Pao**, Discussion Paper Series 557, University of Oxford, July 2011

Perturbed Markov chains

Idea from **Young**, *The evolution of conventions*, Econometrica 1993

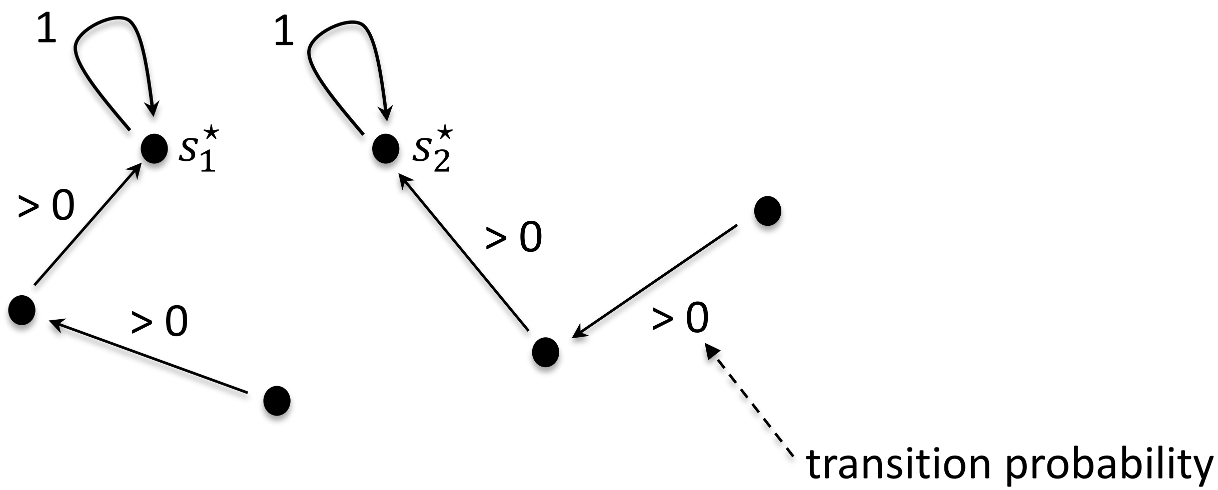
Step 1. Construct a Markov chain absorbed in states maximizing social welfare

Step 2. Perturb the Markov chain to achieve irreducibility

Step 3. Show that in steady-state, the perturbed Markov chain concentrates on socially optimal states

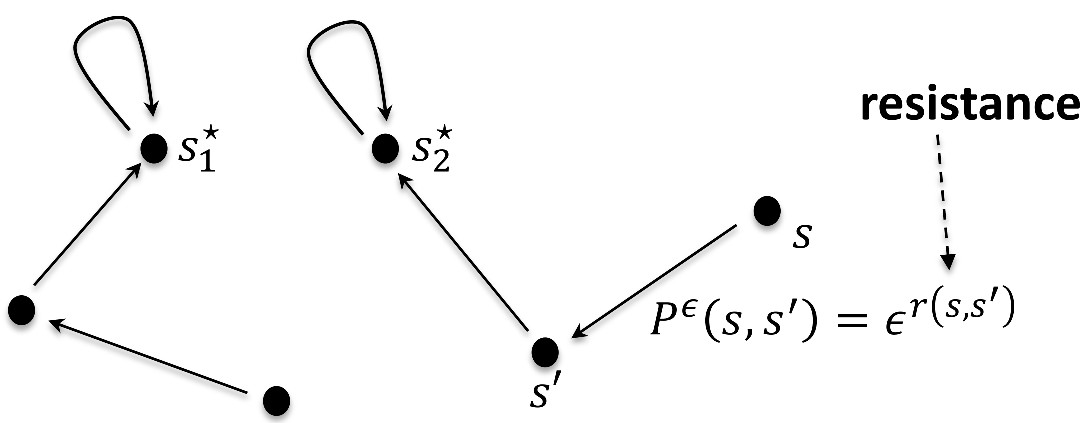
Transient Markov chain

Let $\Omega^\star$ be the set of socially optimal states.



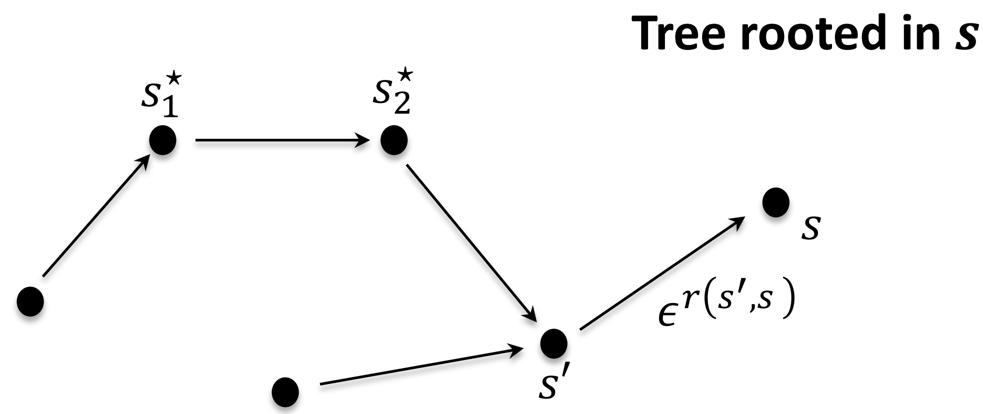
Resistance, rooted trees, potential

Step 2. Irreducible perturbed Markov chain



Resistance, rooted trees, potential

Step 2. Irreducible perturbed Markov chain

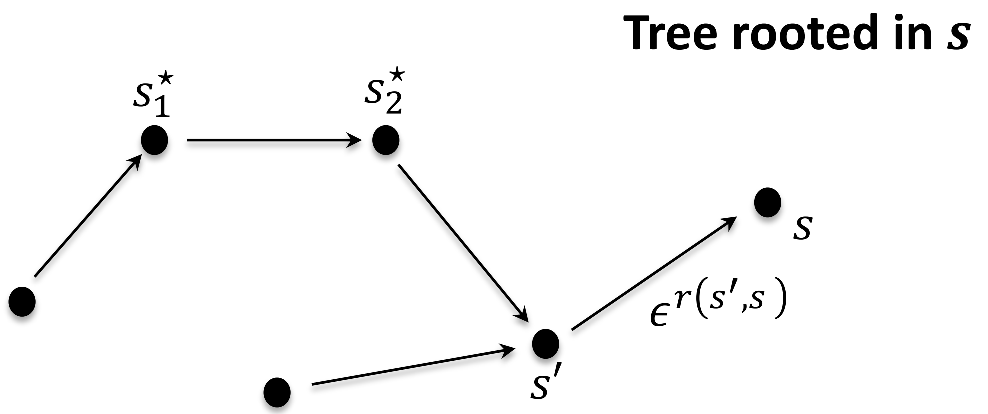


Steady-state distribution: $\pi^{\epsilon}(s) \sim \sum_{T \in \mathrm{Trees}_s} \epsilon^{\sum_{(s_1,s_2)\in T} r(s_1,s_2)}$

Potential of s: $\gamma(s) = \min_{T \in \mathrm{Trees}_s} \sum_{(s_1,s_2)\in T} r(s_1, s_2)$

Resistance, rooted trees, potential

Lemma When $\epsilon \to 0$, π^ϵ concentrates on states with minimal potential.



Steady-state distribution: $\pi^\epsilon(s) \sim \sum_{T \in \mathrm{Trees}_s} \epsilon^{\sum_{(s_1,s_2)\in T} r(s_1,s_2)}$

Potential of s: $\gamma(s) = \min_{T \in \mathrm{Trees}_s} \sum_{(s_1,s_2)\in T} r(s_1, s_2)$

Algorithm

Challenge: each agent must be aware of the way social welfare evolves when updating her action

Idea: enrich the *state* of agent

$$x_i(t) = (\bar{s}_i(t), \bar{u}_i(t), m_i(t))$$

Baseline action

Baseline utility

Mood $\in \{C, D\}$

Algorithm: phase 1

At the beginning of each time period t: $k > N$,

- If $m_i(t) = C$, select a new action a according to

  $$\beta_i(a) = \frac{\epsilon^k}{A-1} \text{ if } a \neq \bar{s}_i(t)$$

- If $m_i(t) = D$, select a new action a uniformly at random

Algorithm: phase 2

At the end of each time period t: agent i observes her received utility $u_i(t) = U_i\big(s(t)\big) \in [0,1]$, and updates her state as:

- If $m_i(t) = C$,
  - if $(s_i(t), u_i(t)) = (\bar{s}_i(t), \bar{u}_i(t))$, $x_i(t+1) = x_i(t)$;
  - else $(\bar{s}_i(t+1), \bar{u}_i(t+1)) = (s_i(t), u_i(t))$

    $m_i(t+1) = C$ w.p. $\epsilon^{1-u_i(t)}$

- If $m_i(t) = D$, $(\bar{s}_i(t+1), \bar{u}_i(t+1)) = (s_i(t), u_i(t))$

  $m_i(t+1) = C$ w.p. $\epsilon^{1-u_i(t)}$

Convergence

Theorem For any $\delta > 0$, there exists ϵ such that:

$$\liminf_{t\to\infty} \frac{1}{t}\sum_{i=0}^{t-1} 1_{\{s(i)\in\Omega^\star\}} \geq 1-\delta$$

Proof. Show that a state has baseline actions in $\Omega^\star$ if and only if it has minimum potential.

Application

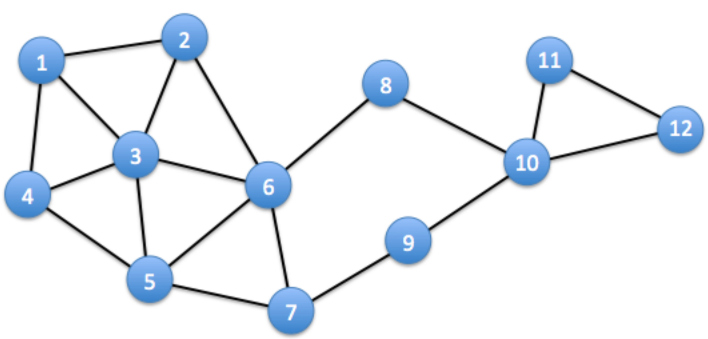
Wireless networks with infrequent channel switching.
Between two updates, MAC protocols (e.g. CSMA) share resources in time.

$$U_i(s) = U(\mu_{i,s_i} \times \phi_i(s))$$

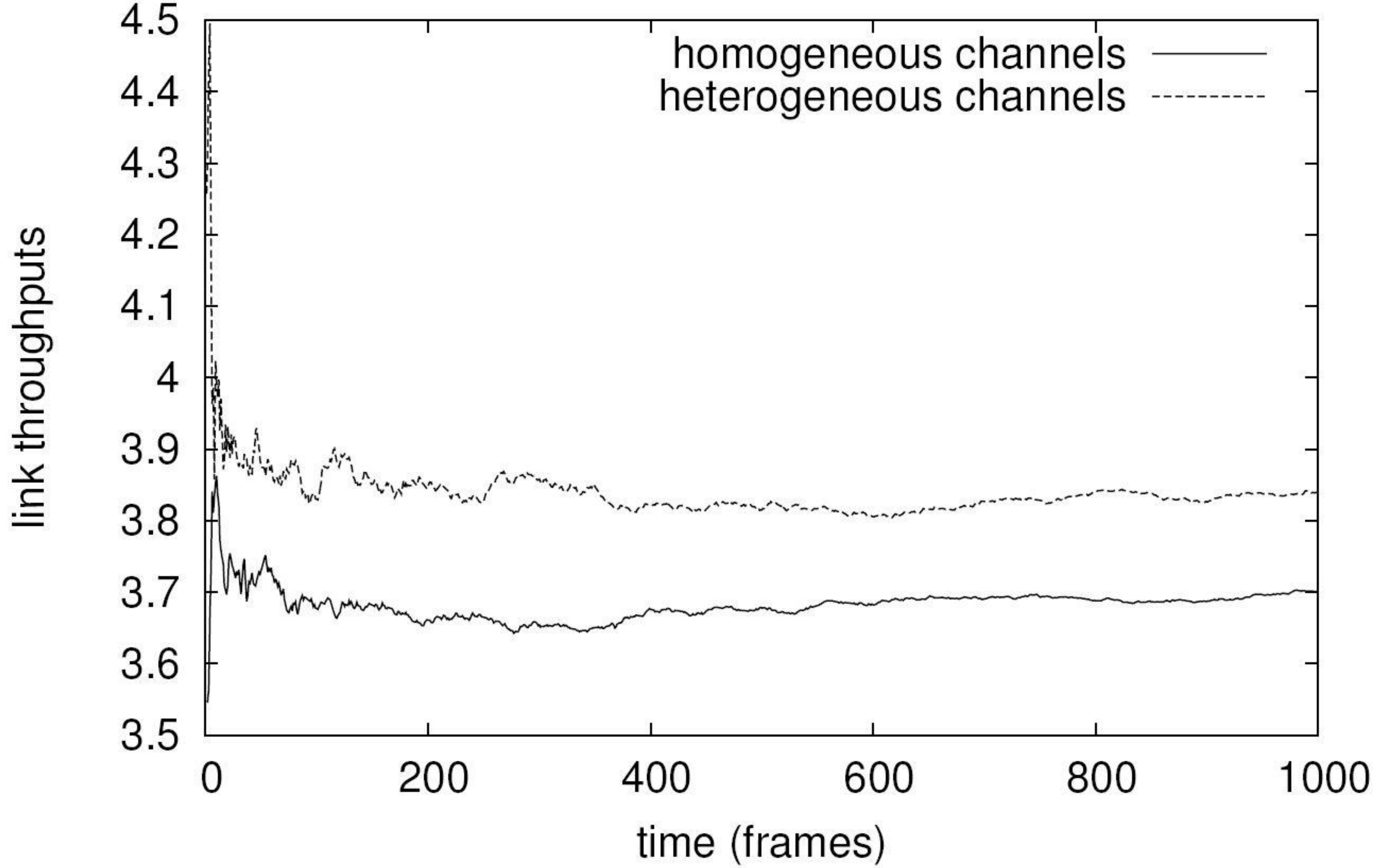
Channel condition

Results from MAC (e.g. depends on the number of agents using the same channel)

Interference graph:



Random network – 3 channels



Outline

- Internet congestion control
  - Distributed optimization with separable objective function
- Two miracles in resource allocation in wireless networks
  - Distributed optimization with un-separable objective function, and without message passing
  - Power control
  - Carrier Sensing Multiple Access
- Parallel computations
  - Joint consensus and gradient descent methods
  - Just gradient descent
- Colorings
  - Combinatorial optimization: a sampling approach
- Distributed gradient free optimization

General objective function

Objective: solve the following convex optimization problem

$$\min \;\; f(x) = \sum_i f_i(x)$$

over $x = (x_1, \dots, x_N) \in \mathrm{X} = \mathrm{X}_1 \times \cdots \times \mathrm{X}_N$

Can we design a decentralized pay-off based solution?

General objective function

Objective: solve the following convex optimization problem

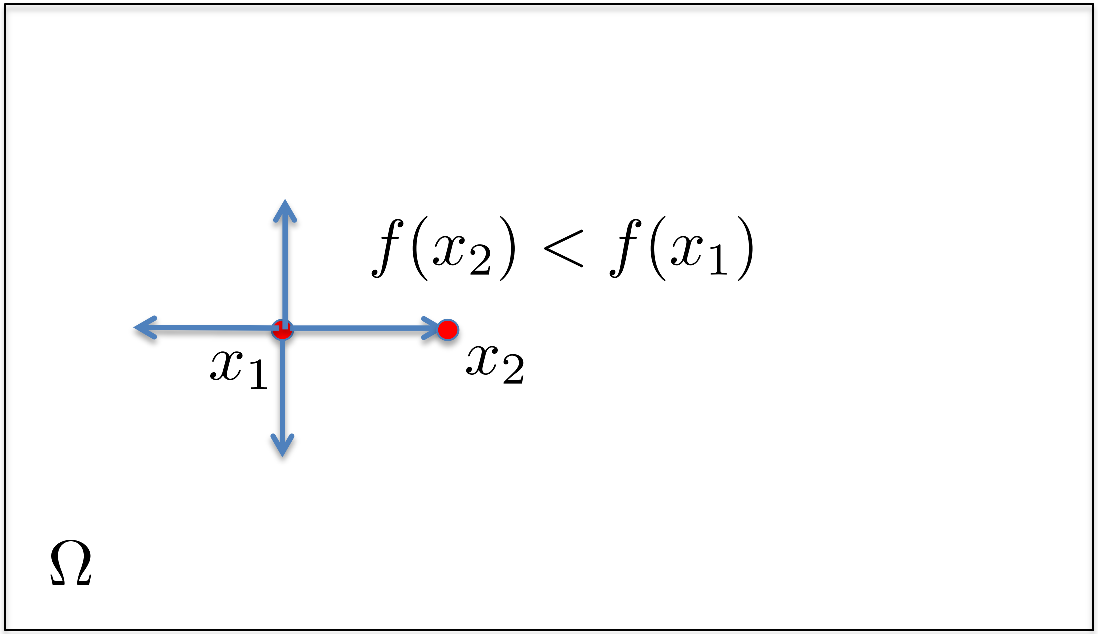
$$\min \;\; f(x) = \sum_i f_i(x)$$

over $x = (x_1, \ldots, x_N) \in \mathrm{X} = \mathrm{X}_1 \times \cdots \times \mathrm{X}_N$

Can we design a decentralized pay-off based solution?

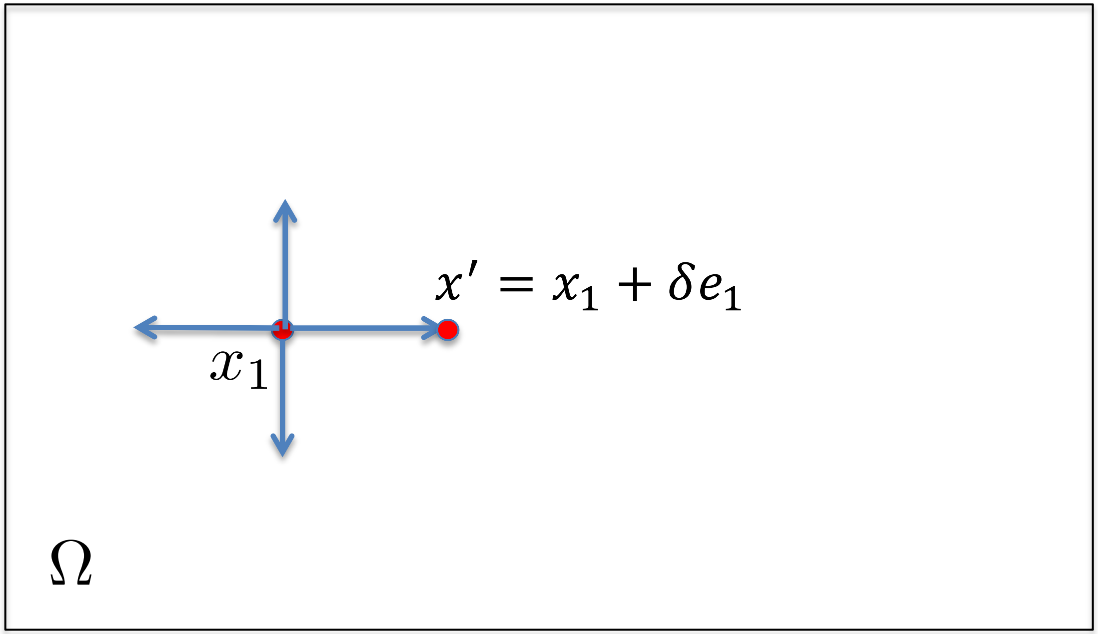
Yes.
Agents communicate through the impact of their actions on the pay-offs of other agents.
Sampling methods are used to aggregate agents' "feelings".

Random local search



- Let us do it in a distributed manner

Pick an agent at random, say 1



- Agent 1 randomly propose an update of her action:

$$x' = x_1 + \delta e_1$$

Play the updated state

$$x' = x_1 + \delta e_1$$

- Agent *i* observes her cost: $f_i(x')$, and compute the corresponding change: $\Delta f_i = f_i(x') - f_i(x_1)$

Accepted or rejected move

$$x' = x_1 + \delta e_1$$

- Agent *i* observes her cost: $f_i(x')$, and compute the corresponding change: $\Delta f_i = f_i(x') - f_i(x_1)$

- Agent *i* accepts the move with probability: $\epsilon^{\Delta f_i + 1}$

- The move is accepted only if ALL agents accept it!

  This happens with probability proportional to: $\epsilon^{\Delta f}$

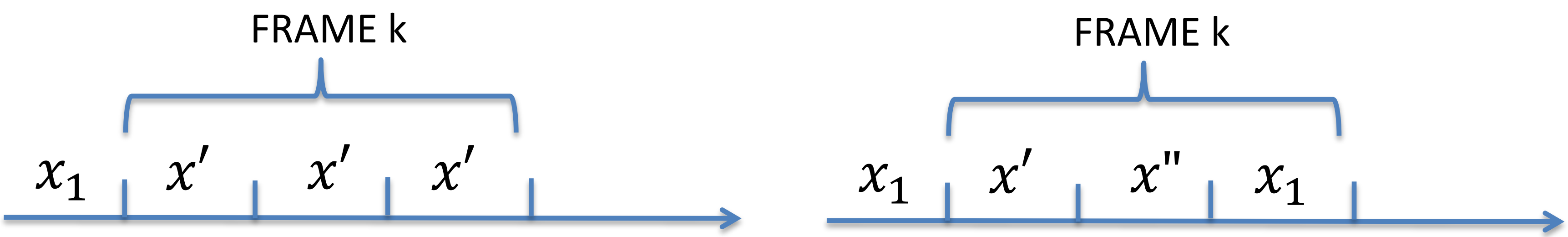
Acceptance notification

- Inter-dependence assumption: We assume that in any state, any agent has an action whose effect is recognizable by all agents:

$$\forall x, i, j,\ \ f_i(x) \neq f_i(x_{-j}, O_j)$$

- If all agents accept the move then they keep playing the same action, $x" = x'$. Otherwise agent 1 is informed about the rejection, $x" = (x'_{-j}, 0_j)$.

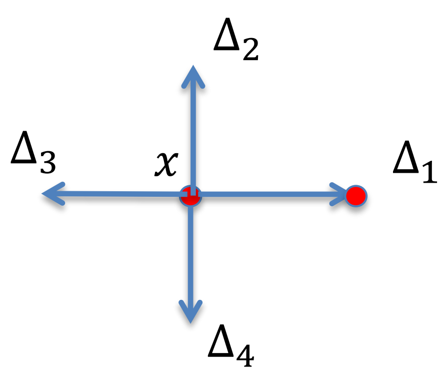
Resulting Dynamics



Accepted move

Rejected move

- We can look at the times where the state changes: induced Markov chain. Let e.g. $\Delta_1 = f(x + \delta e_1) - f(x)$.



$$P[X_{k+1} - X_k = \delta e_1] = \frac{\epsilon^{\Delta_1}}{\epsilon^{\Delta_1} + \epsilon^{\Delta_2} + \epsilon^{\Delta_3} + \epsilon^{\Delta_4}}$$

When ε is small, we move in the steepest descent direction.

Summary
Distributed optimization

- Constrained convex separable problem: GD can be implemented if “prices” are communicated to agents
- Sometimes these “prices” can be guessed via observing pay-offs – GD without message passing
- Convex non-separable problem: message passing across agents helps to implement GD descent
- But, all kinds of problems can be solved in a distributed manner **without any signaling**

- **…** what matters is the convergence time