

# e-Merge-ANT: Spring 2001

Stephen Fitzpatrick, Cordell Green & Lambert Meertens  
Kestrel Institute, Palo Alto  
ants.kestrel.edu

ANTs PI Meeting  
Lake Tahoe, CA, 30 April-2 May 2001

- Status
- Architecture
  - decentralized resource management through graph coloring
- Soft graph coloring algorithms
  - decentralized, iterative-repair, anytime, approximate colorers
- Experimental results
  - dynamics, problem complexity, scalability, robustness

## Previous Results

Framework for distributed resource management as scheduling



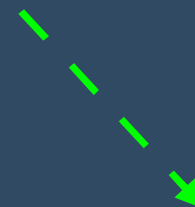
Algorithm for distributed scheduling:  
Self-Induced Colorer



Challenge problem demo on simulator & hardware

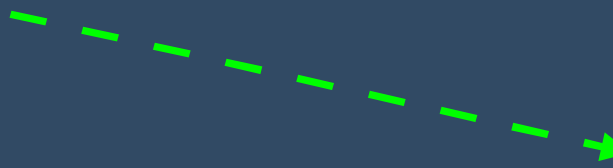
## New Results

Framework for assessing performance  
Experimental investigation of performance  
Faster, cheaper algorithms:  
Fixed-Probability Colorer &  
Conservative Fixed-Probability Colorer  
Initial theoretical analysis of performance  
Visualization



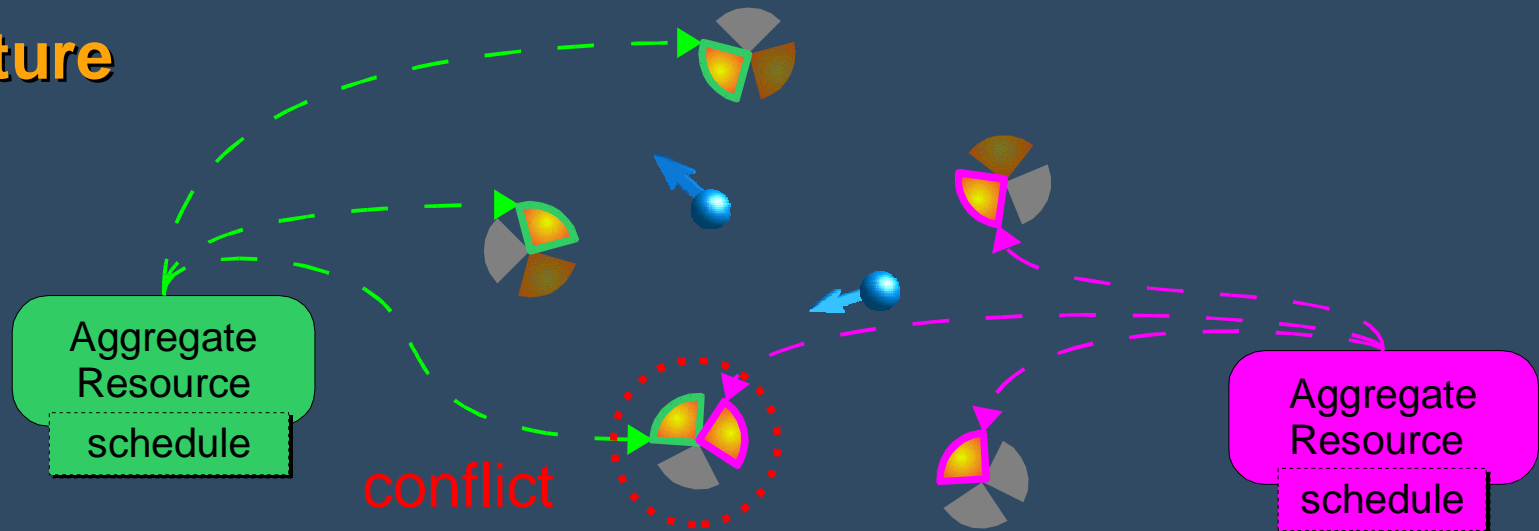
## Autumn Demo

Integrate new algorithms with hardware & new simulator





# Architecture



- Three sensors must collaborate to triangulate a target
  - define *aggregate sensors* representing “triangulators”
- Aggregate sensors may share physical radar units
  - each physical radar unit contains three heads
  - constraint: only one head can be sampled at a time
  - each radar unit may service several aggregate sensors
- A *sampling conflict* may occur
  - when two or more aggregate sensors try to use the same radar unit simultaneously
- Avoid conflicts by scheduling
  - assign time slots for aggregate sensors to use samplers



# Scheduling as Graph Coloring

## Challenge Problem

## Resource Management

## Graph Coloring

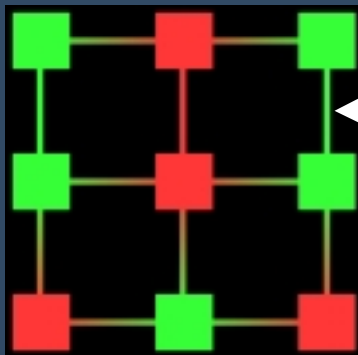
aggregate sensors ← resources → nodes

sample only one head per unit ← mutual exclusion constraints → edges

time slots ← reservations → colors

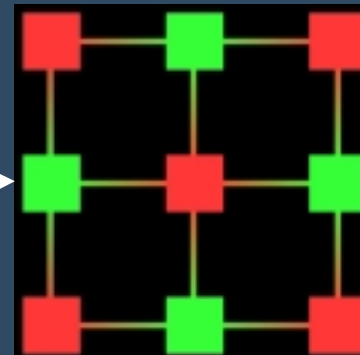
feasible scan schedule ← feasible schedule ← proper coloring

minimum scan period ← minimum feasible schedule length → chromatic number



a conflict

a proper 2-coloring





# Decentralized Graph Coloring

- Each node is to choose its own color
- Iterative algorithms:
  - begin with a random coloring
  - iteratively improve
    - each node chooses a color that minimizes its conflicts with its neighbors
- Need to coordinate choice of colors
  - if two neighbors simultaneously choose colors, they may choose the same color
  - i.e., they may introduce a conflict
- Number of colors fixed in advance
  - e.g., by considering latency constraints
    - scan range, speed of target, measurement duration



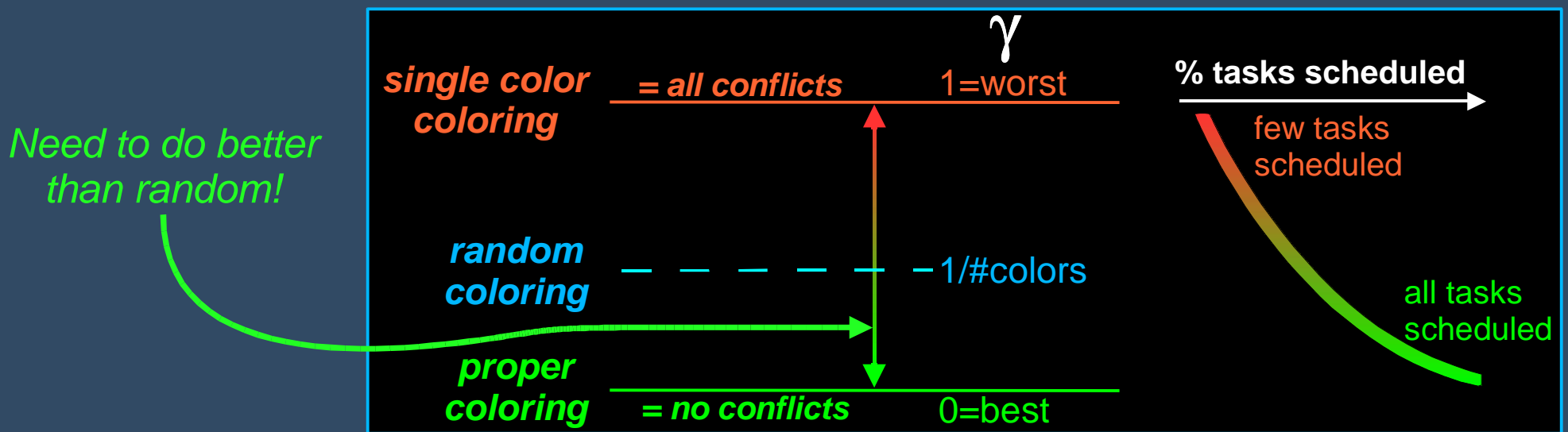
## Old Algorithm: Self-Induced Colorer

- Previously, we used *self-induced* coloring to coordinate color choices (to reduce the introduction of conflicts)
  - a node chooses a color for itself only when its *current* color is “active”
- Demoed with simulator/hardware
  - gave good performance in challenge problem simulator
- Missing: quantitative evaluation for large graphs ...



# Soft Graph Coloring

- Generalize the metric on colorings from proper/non-proper to ..
- *Degree of conflict*  $\gamma$ 
  - $\gamma = (\text{number of conflicts}) / (\text{total number of edges})$ 
    - range is [0,1]: 0 is best, 1 is worst
  - independent of graph size
  - suitable metric for off-line analysis of progress of anytime colorer
    - a random coloring with C colors has an expected score of  $\gamma = 1/C$
    - this acts as a baseline for assessing algorithms
  - applicable even in over-constrained scenarios
    - e.g., 3-coloring a 4-colorable graph





## New Algorithm: Fixed-Probability Colorer

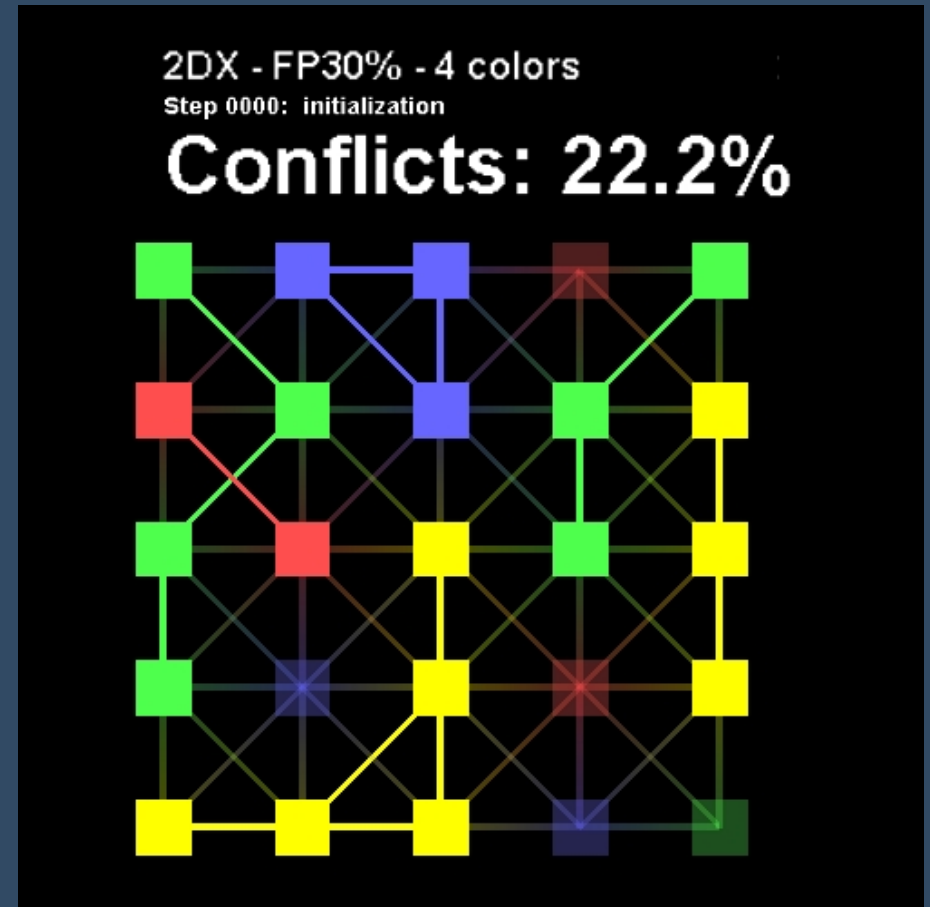
- FP is a soft graph colorer
  - decentralized, iterative, anytime, local-repair algorithm
- Iterated, synchronized steps, each having three phases:
  - a. Probabilistic activation:
    - at each step, each node activates at random with a fixed, uniform probability (the *activation probability*)
    - in contrast, in SI, nodes activate color by color
      - SI is less likely to introduce conflicts than FP but has lower parallelism
  - b. Select color using local repair/optimization:
    - when a node activates, it chooses a color that minimizes its conflicts with its neighbors
      - based on its current knowledge of its neighbors' colors
  - c. Local communication:
    - when a node changes color, it informs its neighbors



# The FP Algorithm At Work

## Initialization

- 4 colors
- Topology:
  - each non-boundary node has 8 neighbors
- Edges:
  - bright = a conflict
  - faded = not a conflict
- Nodes:
  - bright = some incident edges are conflicts
  - faded = no incident edge is a conflict

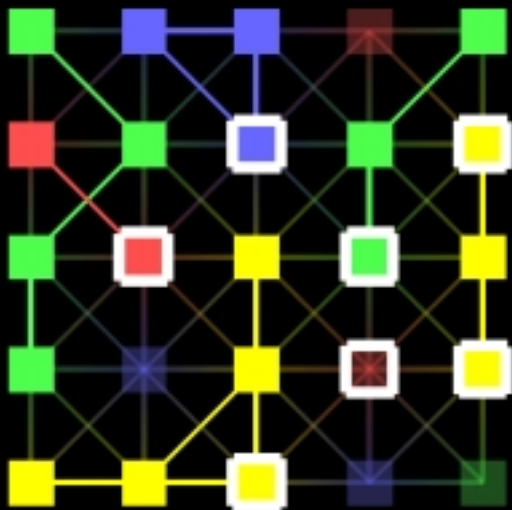




2DX - FP30% - 4 colors

Step 0001: Activate

Conflicts: 22.2%



2DX - FP30% - 4 colors

Step 0001: Color

Conflicts: 22.2%

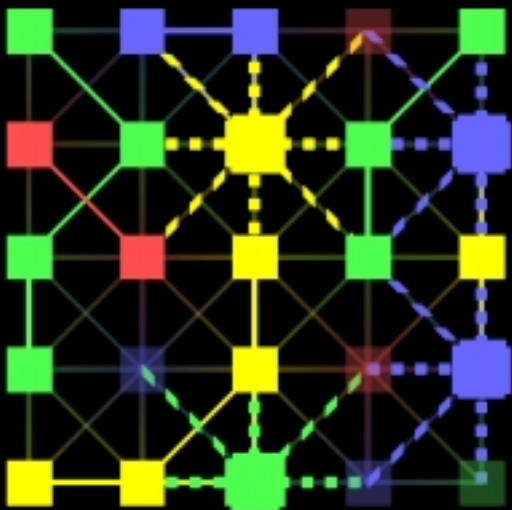


Step 1

2DX - FP30% - 4 colors

Step 0001: Communicate

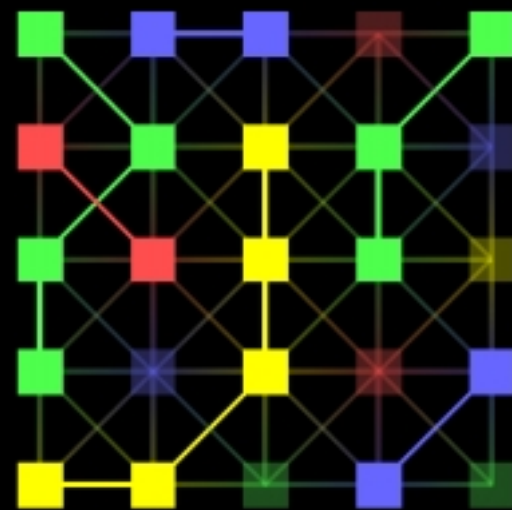
Conflicts: 22.2%



2DX - FP30% - 4 colors

Step 0001: Assess

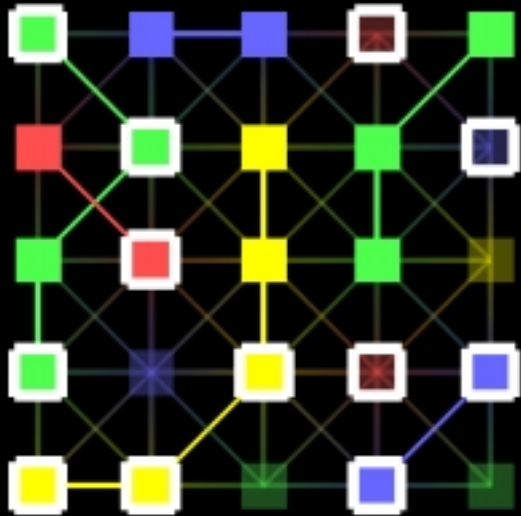
Conflicts: 16.7%





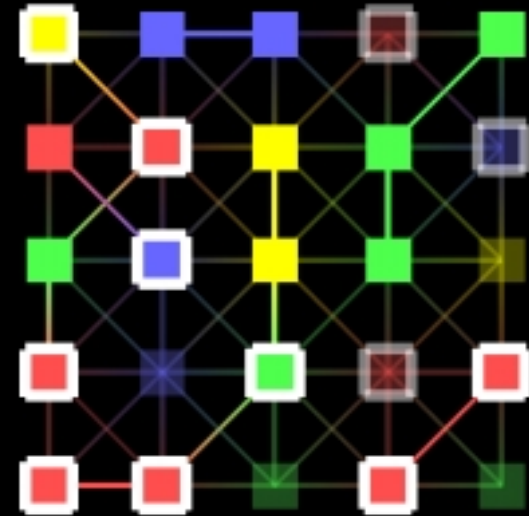
activate →

2DX - FP30% - 4 colors  
Step 0002: Activate  
**Conflicts: 16.7%**



→ color

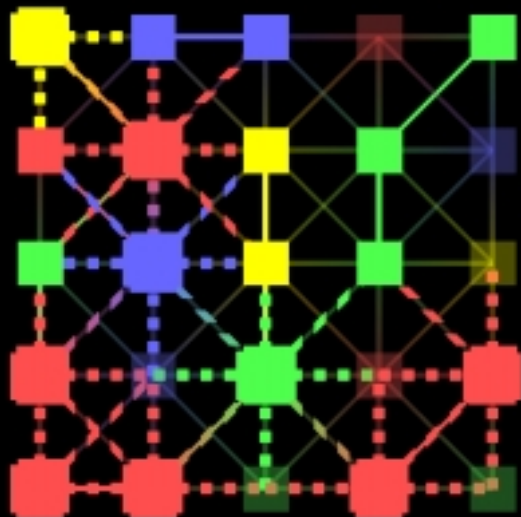
2DX - FP30% - 4 colors  
Step 0002: Color  
**Conflicts: 16.7%**



Step 2

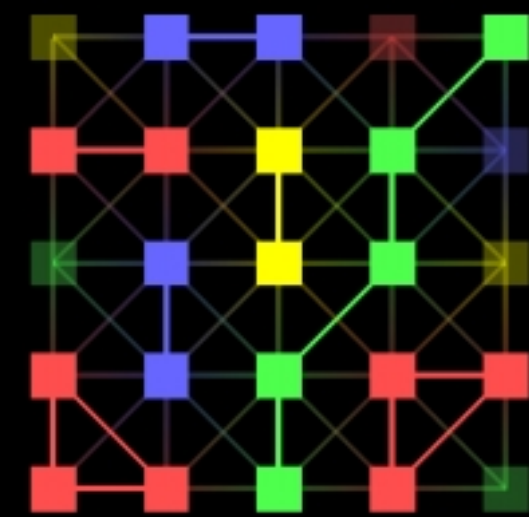
← send

2DX - FP30% - 4 colors  
Step 0002: Communicate  
**Conflicts: 16.7%**



→ assess

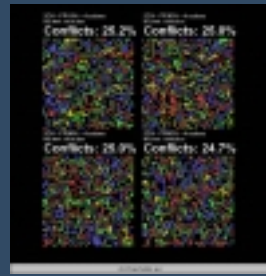
2DX - FP30% - 4 colors  
Step 0002: Assess  
**Conflicts: 19.4%**



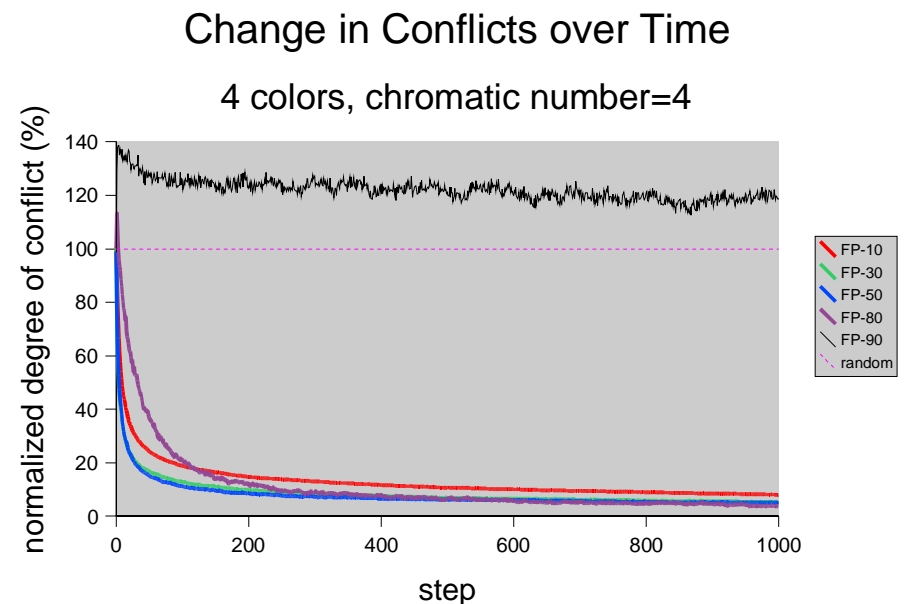


## Convergence: Typical Behaviors

- FP converges rapidly for wide range of activation probabilities
  - 30% seems to be a good choice for a wide range of graphs
- If the activation probability is too high, FP does not converge
  - neighbors simultaneously update colors (introducing conflicts)
  - more complex graphs require lower activation probabilities
- If the probability is too low, FP converges too slowly
  - in particular, early reduction of conflicts is slow
- Need to balance speed against convergence



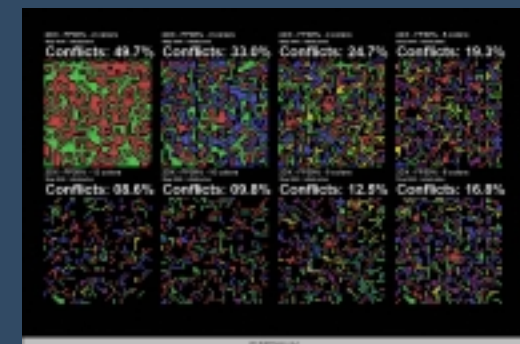
- *Normalized degree of conflict*  $\Gamma = \gamma C$ 
  - coloring is easier with more colors
  - scale  $\gamma$  by the number of colors  $C$
  - simplifies analysis of experimental data
  - a random coloring has an expected value of  $\Gamma$  of 1





## Problem Complexity: Constraint Tightness

- The chromatic number seems to be a critical threshold for problem complexity
- FP performs “well” when critically or slightly under-constrained
  - #colors equal to or slightly greater than chromatic number
  - FP usually achieves proper coloring when under-constrained
- FP performs “reasonably” & *behaves* well when over-constrained
  - #colors < chromatic number
  - reduces conflicts significantly below random level
  - doesn't fall down & doesn't blow up
- FP's performance when loosely-constrained is counter-intuitive
  - performance is not as good as might be expected on easy problems





# Performance of FP against Activation Prob. & #Colors

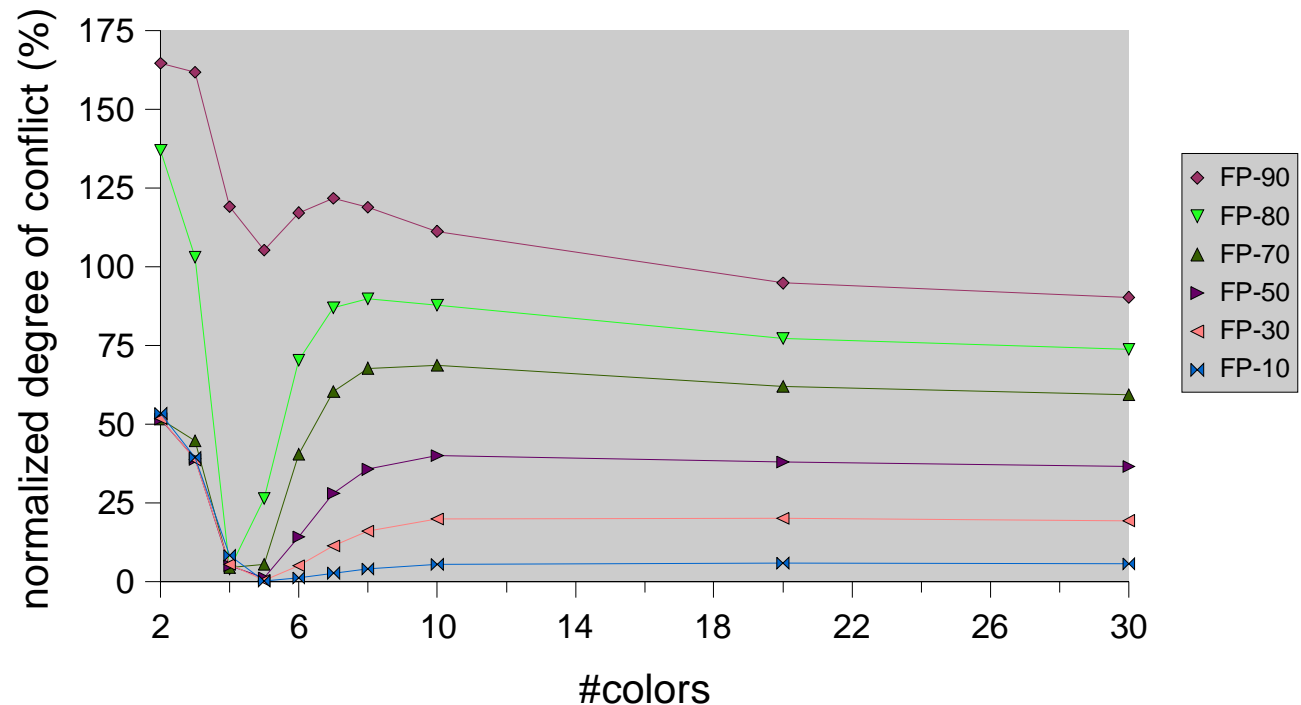
- When loosely constrained, FP partly acts like a random colorer
  - most colors are unused in a given neighborhood
  - a node chooses randomly from the unused colors
  - so at every activation, a node is highly likely to change color

## Loosely constrained FP

- $\Gamma$  does **not** converge to zero
- simple analysis predicts  $\Gamma \rightarrow C\sigma\theta/(2-\theta)$ 
  - $C$  is the number of colors
  - $\theta$  is the activation threshold
  - $\sigma$  is the probability that two neighbors will choose the same color if they activate simultaneously
- experiments give a good fit for  $\sigma=1/(C-C_0)$ 
  - $C_0$  is the chromatic number

## Performance of FP vs Tightness of Constraints

chromatic number=4, after 1000 steps

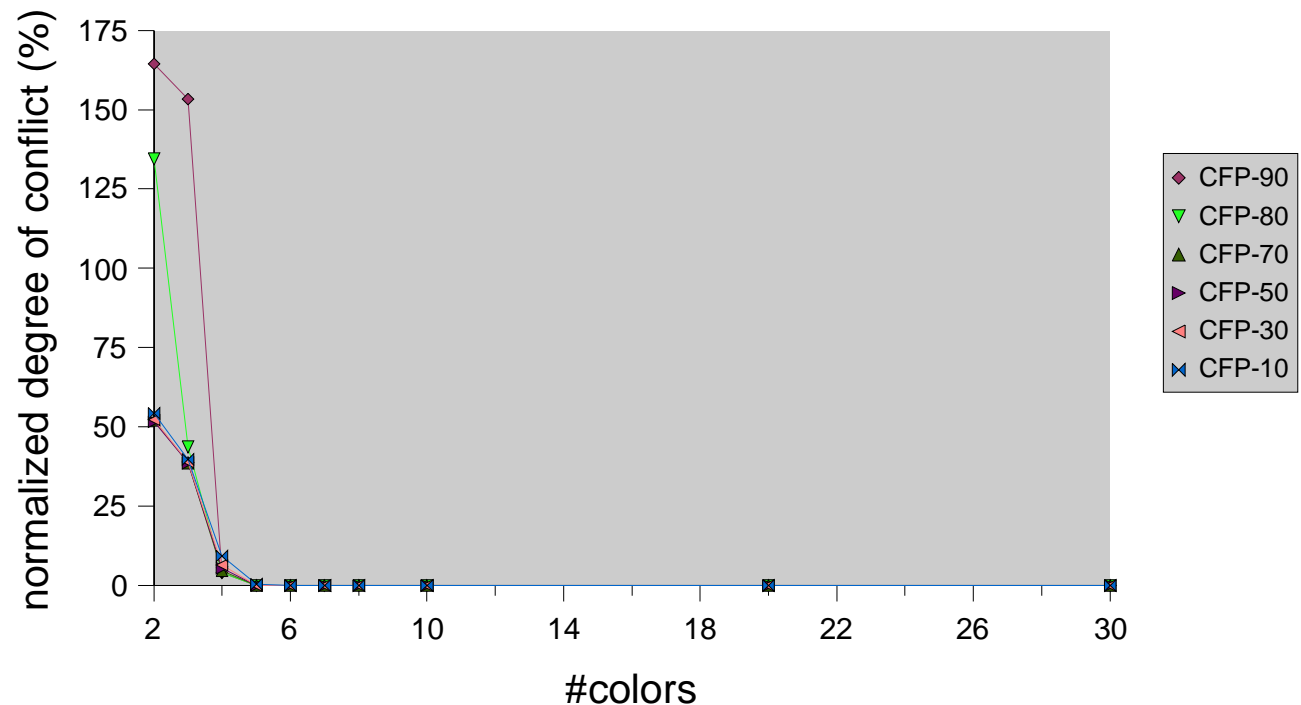


# New Algorithm: Conservative Fixed-Probability Colorer

- CFP is a more “conservative” variant of FP
  - now an activated node will change color only if it has conflicts with its neighbors
- CFP has better performance when under/loosely-constrained
  - proper coloring rapidly achieved



Performance of CFP vs. Tightness of Constraints  
chromatic number=4, after 1000 steps

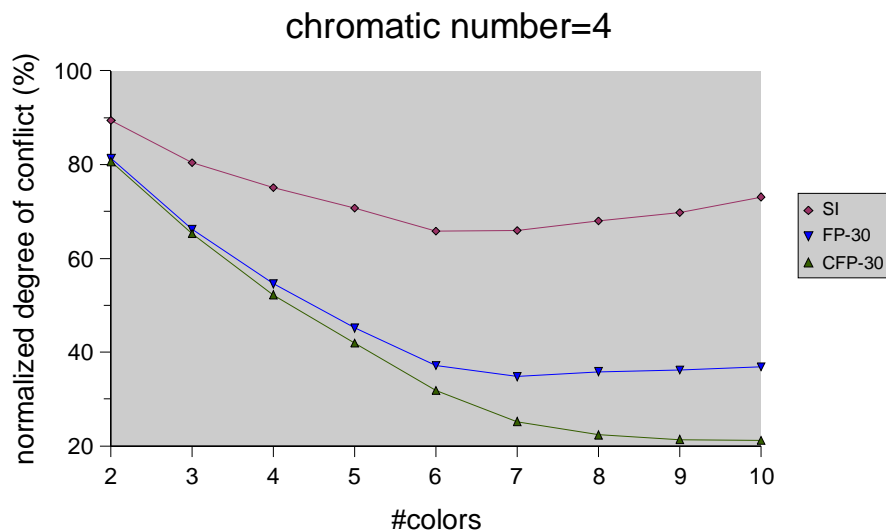




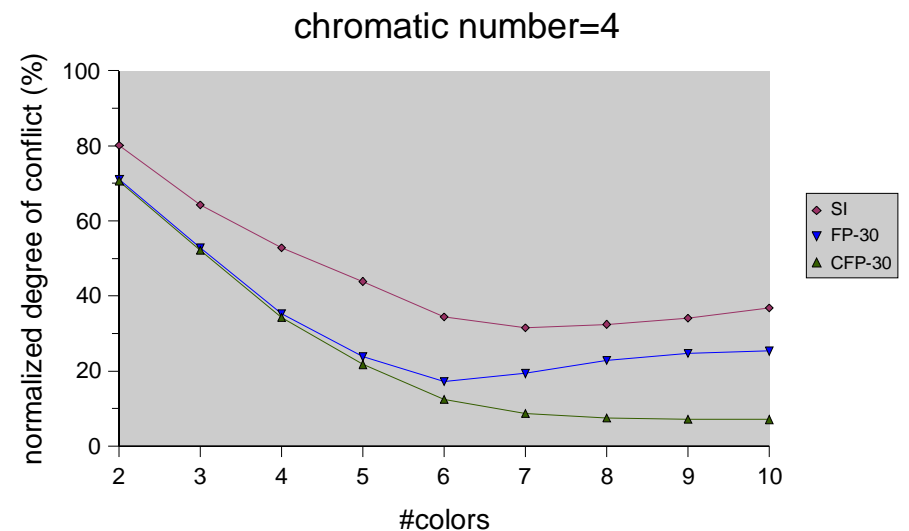
## Short-Term Response: Conflicts

- CFP quickly reduces conflicts when critically constrained, under-constrained or loosely constrained
  - ⇒ adaptation to changing tasks/resources
  - CFP is an *anytime* algorithm
    - tracking proceeds simultaneously with coloring
    - appropriate metric: the *mean* of the degree of conflict
- CFP reduces conflicts below random when over-constrained

Short-Term Response: Mean Conflicts/10 Steps



Short-Term Response: Mean Conflicts/30 Steps



for the challenge problem



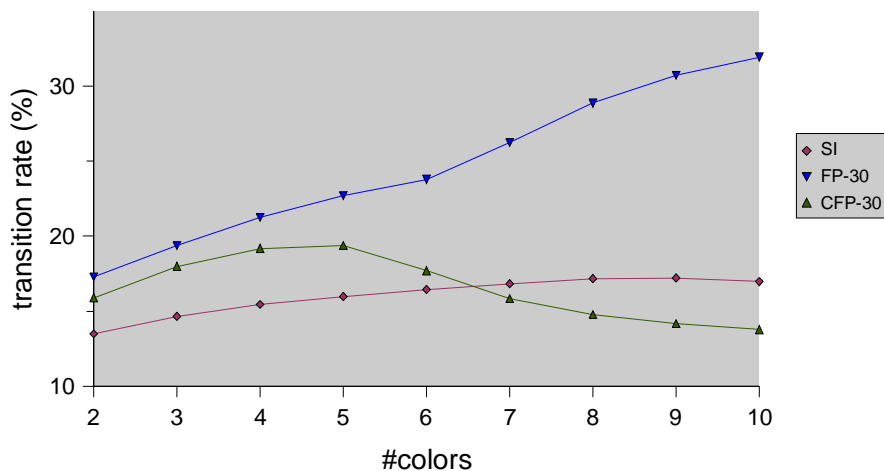
# Short-Term Response: Communication

- CFP has low communication costs

- For a single step, the transition rate  $\tau$  is the fraction of nodes that change color
  - $\tau$  is independent of the interconnection complexity of the graph
  - it simplifies comparison of experimental data over multiple graphs

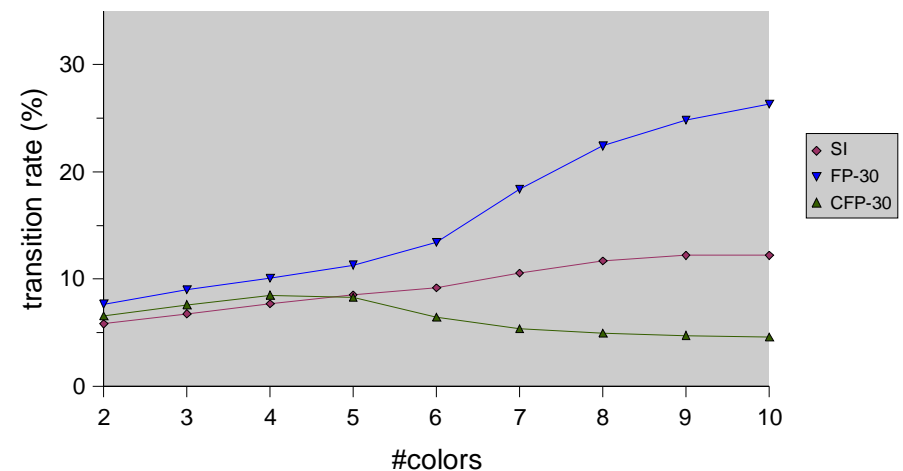
Short-Term Response: Mean Comm./10 Steps

chromatic number=4



Short-Term Response: Mean Comm./30 Steps

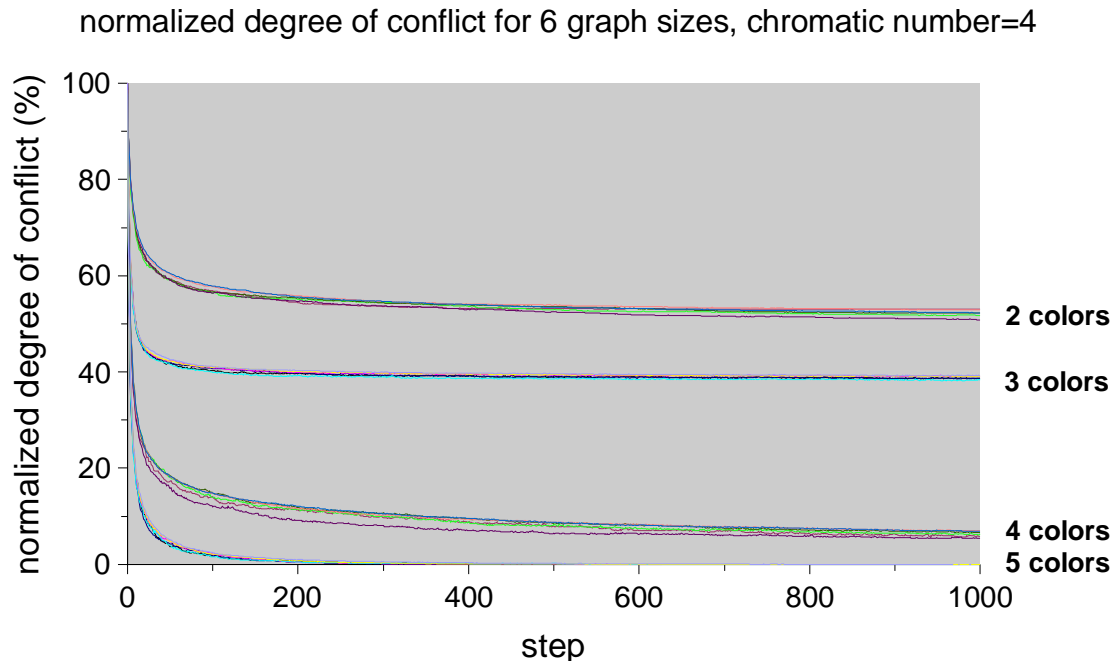
chromatic number=4



# Scalability

- CFP is scalable
  - per-node costs are independent of the number of nodes
  - per-node communication, storage & computation costs proportional to number of neighbors, not number of nodes
- Rate of conflict reduction for CFP is independent of graph size
  - for large graphs of similar structure, degree of conflict does not vary much with graph size

## Scalability of CFP (30%)



For each color, plot shows results for 6 graphs averaged over 3 runs per graph

- 625 nodes
- 900 nodes
- 1000 nodes
- 1520 nodes
- 3600 nodes
- 4970 nodes



## Fault Tolerance – Dynamic Topology

- CFP gracefully adapts to faulty nodes
  - low rates of node turnover, applied continuously, slightly reduce the quality of colorings
  - CFP recovers robustly from moderate rates of node turnover applied intermittently
    - number of conflicts jumps, but quickly falls
- Tested using a simple scheme to simulate a dynamic hardware configuration (e.g., nodes dying and reviving)
  - varies the topology without drastically altering the complexity (i.e., the chromatic number)
  - simplifies analysis
    - Construct a graph
    - Remove  $R$  randomly-chosen nodes (and incident edges)
    - Every  $P$  steps
      - remove a further  $R$  randomly-chosen nodes (and incident edges)
      - from the pool of  $2R$  removed nodes, reinsert  $R$  randomly chosen nodes
      - reinsert all previously removed edges whose end nodes are now present in the graph

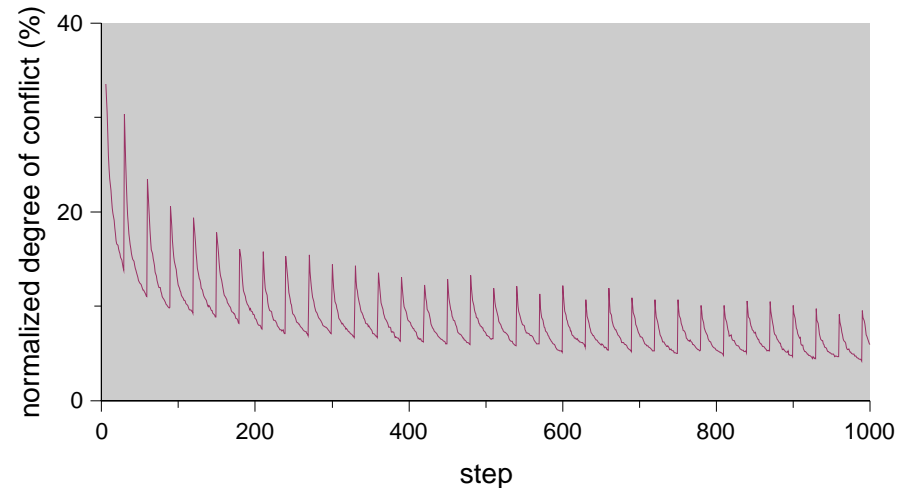


# Dynamic Topology – Effect

Intermittent:  
turnover rate  $R=20\%$   
applied every 30 steps

## Effect of Dynamic Node Set on CFP (30%)

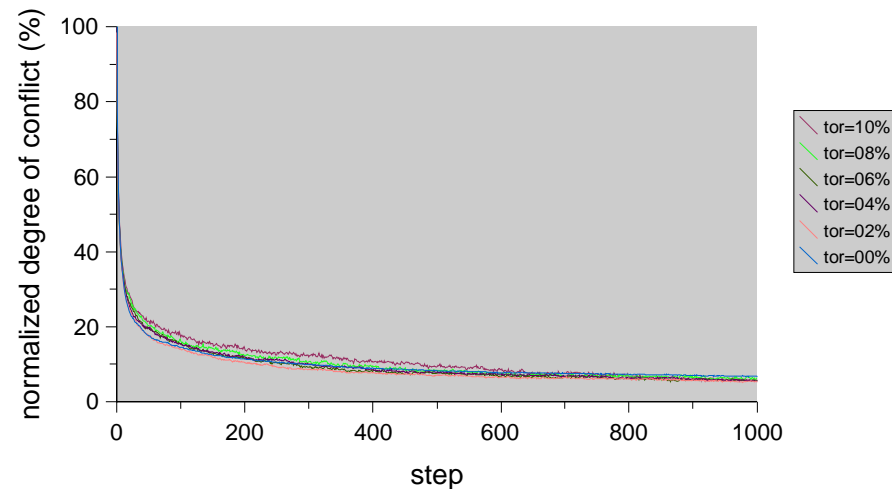
chromatic number=4, #colors=4; turnover=20%/30 steps



Continuous:  
turnover applied  
every step

## Effect of Dynamic Node Set on CFP (30%)

chromatic number=4; #colors=4; continuous turnover



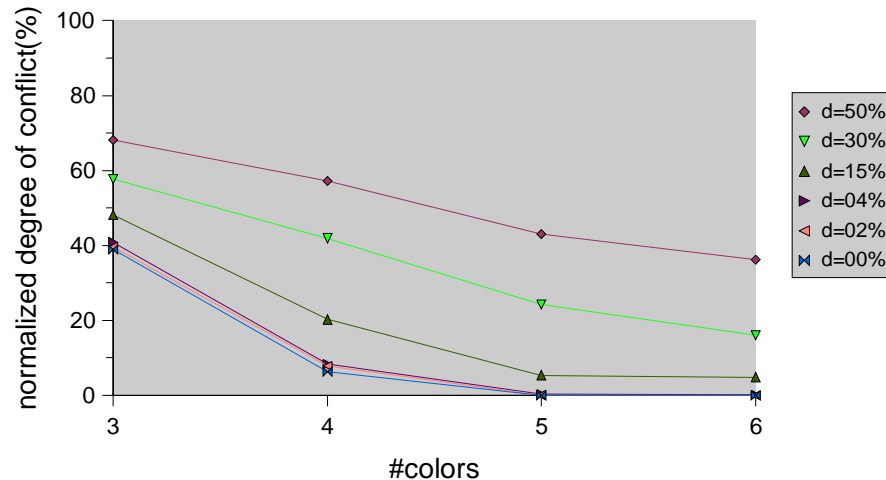


# Communication Noise and Loss

- CFP is tolerant of (low-level) communication noise and loss
  - low-level noise or lossiness increases the degree of conflict incrementally
- Model communication noise as follows:
  - each color-change message is subjected to a random process:
    - with probability  $d$ , the message is dropped
    - with probability  $r$ , the color is randomized
    - with probability  $1-d-r$ , the message is passed through unaltered

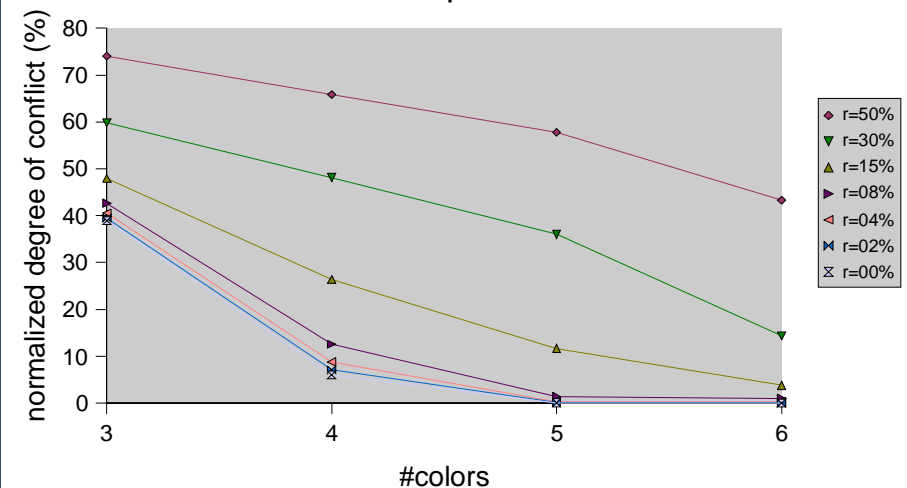
Effect of Communication Loss on CFP (30%)

conflicts after 1000 steps, chromatic number=4



Effect of Random Noise on CFP (30%)

conflicts after 1000 steps, chromatic number=4





## Summary of Results for CFP (*et al.*)

- **Theoretical models**
  - convergence when under-constrained
  - convergence when critically constrained
  - loose upper-bound on communication costs
  - scalability (constant parallel complexity)
- **Experimental results**
  - reasonable activation probability for wide range of graphs
  - convergence
  - rapid short-term reduction in conflicts when under-constrained
  - rapid short-term reduction in conflicts when critically constrained
  - good behavior when over-constrained
  - low communication costs
  - scalability
  - robustness against node/communication failure
- **Simple algorithm for decentralized, anytime graph coloring**
  - promises fast, cheap, robust resource management



## Future Work

- Does “fast” translate into “fast enough”?
  - need to test algorithm on challenge problem
  - other resource types may give more complex search spaces and may need more complex interaction between schedulers
- Application-specific models of performance
  - explicit relationship between  $\gamma$  and “quality of solution”
- Different classes of local, iterative-repair algorithm
  - e.g., activation based on local measures of degree of conflict
    - measures maintained by diffusion scheme
  - dynamic determination of chromatic number & #colors
    - we already have prototype algorithm
- Open problems (for dynamics/complexity groups?):
  - reliable performance predictors from simple graph metrics
    - e.g., chromatic number, degree of interconnection
    - metrics need to be locally & cheaply computable for use at run-time
  - convergence models for over-constrained coloring
  - improved analysis  $\rightarrow$  improved algorithms