Scalable, Anytime Constraint Optimization through Iterated, Peer-to-Peer Interaction in Sparsely-Connected Networks

Stephen Fitzpatrick & Lambert Meertens

Kestrel Institute

3260 Hillview Avenue, Palo Alto, California, U.S.A.

fitzpatrick@kestrel.edu & meertens@kestrel.edu

This work is sponsored in part by DARPA through the following programs:

ANTS: Autonomous Negotiating Teams http://ants.kestrel.edu/

NEST: Networked Embedded Software Technology http://consona.kestrel.edu/

Contracts #F30602-00-C-0014 & #F30602-01-C-0123 monitored by Air Force Research Laboratory

25 June 2002

IDPT 2002, Pasadena

Overview

- Context: autonomous coordination in large networks of simple sensors
  - scalable, robust, decentralized, adaptive
- Approach:
  - scheduling of sensor actions
  - locally computable metrics on schedules as basis for optimization
  - stochastic, distributed hill climbing to optimize schedules
- Abstract problem for investigation of algorithm dynamics
  - distributed, approximate graph coloring
- Summary of experimental results
- Conclusions

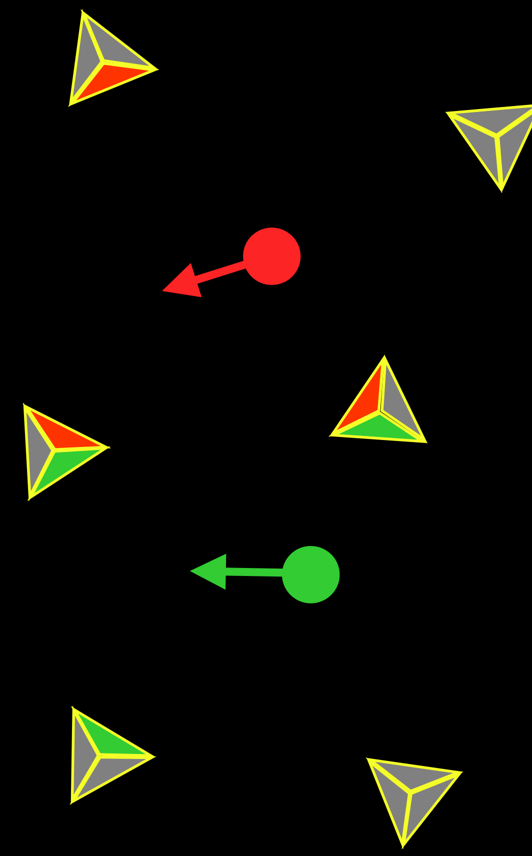
- Details of experimental results
  - if time permits

Large Networks of Simple Sensors

- Scenario: thousands of small, cheap sensors scattered over terrain
- Sensors equipped with low-power radio transmitters & receivers
  - permit broadcast communication between geographically close sensors
    - every node within range of a transmitting node may receive a message
  - communication should be minimized to reduce interference
  - latency is high enough that data/control variables are essentially distributed
- Autonomous coordination is required
  - sensors must be activated & deactivated appropriately to allow long periods of unattended operation with limited energy
  - the quality of data from a single sensor is low so multiple sensors must collaborate to acquire complimentary data
- Emphasis is on attaining good coordination quickly
  - soft real time adaptivity
  - long-term quality is secondary, though important for stable conditions
  - network load may vary dramatically

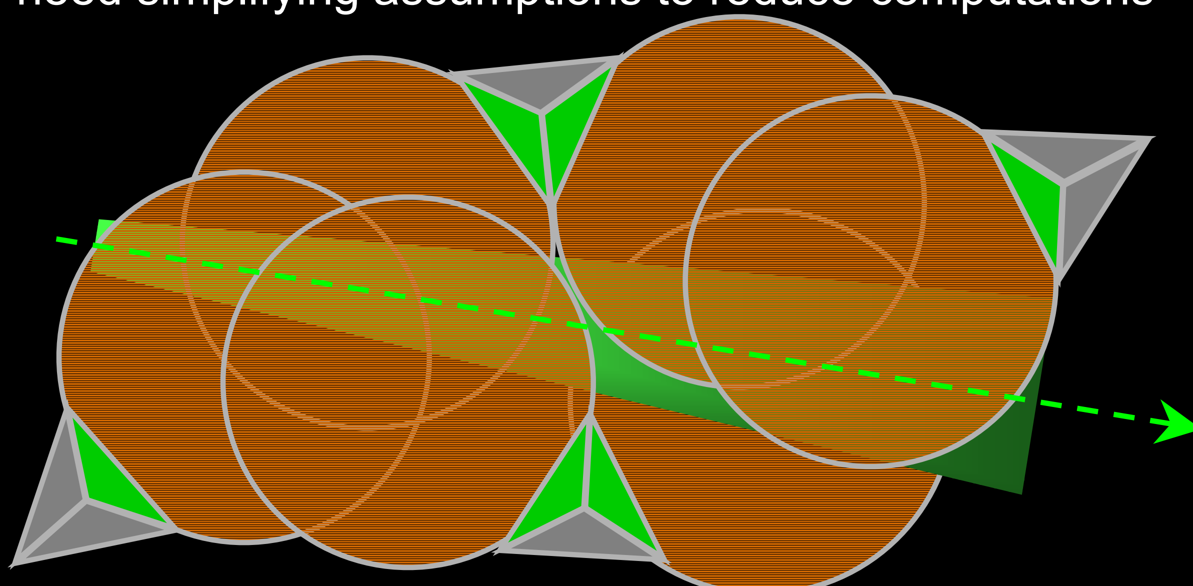
Example Application: Target Tracking

- Multiple targets moving through field of radars
- Each radar is capable of scanning in one of three directions at a time
  - a single scan requires 1 to 2 seconds
  - signal strength depends on distance and angle to target
- About three scans from different radars is required to accurately locate a target
  - scans should be approximately simultaneous
  - a given target should be rescanned every 2 seconds, approximately
- Coordination is required to ensure:
  - most radars can deactivate but targets are still detected
  - each target gets scanned adequately
  - radars scanning the same target do so approximately simultaneously to enhance data quality



Abstract Approach: Scan Scheduling

- Each radar's actions are scheduled over a reasonable period
  - targets are reasonably predictable for ~15 seconds
  - rescheduling allows radars to adapt to changes (e.g., a target turning)
- Metrics quantify schedule quality w.r.t. target behavior
  - high scores when targets are scanned simultaneously by about three radars
  - also take into account cost of scanning and constraint violations
- Objective is to determine scan schedules that optimize overall metric
  - an overall metric can be defined in terms of **expected values**
  - in practice, need simplifying assumptions to reduce computations



Distributed Computation of Scan Schedules

- Define a local, per-radar version of the quality metric
  - *assume* that each radar knows about targets in its vicinity
  - *assume* that each radar knows the scan schedules of nearby radars
  - then each radar can compute the quality of its scan schedule based on local information
- Each radar optimizes its own schedule w.r.t. its local quality metric
  - *assume* that a stable set of locally-good schedules is computed
    - given stable target behavior
  - then overall quality is expected to be good for practical sensor applications
    - can *not* in general make claims about true optimality
  - there may be pathological metrics for which achieving everywhere locally-good quality results in overall poor quality
    - probably will not occur in sensor domain
  - more practical concern is rate of convergence and stability
    - how to validate assumptions …

Continual Data Push

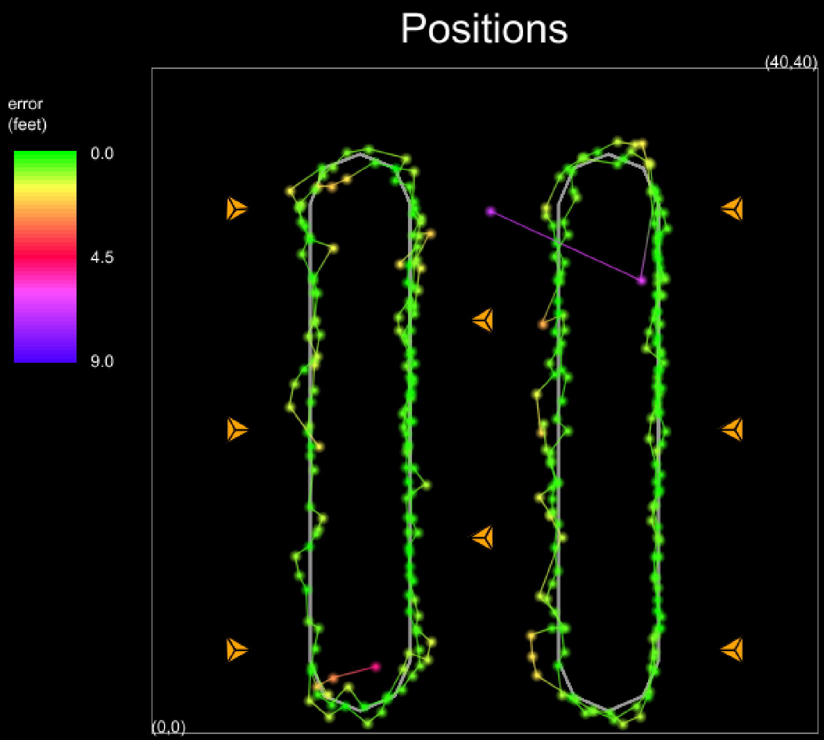
- *Assumption:* each radar knows about targets in its vicinity
  - when a radar acquires data, it broadcasts it to nearby radars
  - each radar can combine data to produce local target estimates
- *Assumption*: each radar knows the scan schedules of nearby radars
  - when a radar computes its own scan schedule, it broadcasts the new schedule to nearby radars

Distributed Hill Climbing

- *Assumption*: a stable set of locally-good schedules is computed
  - if schedules are recomputed too frequently, then incoherence results
    - because of communication latency, each radar is using out-of-date information in making its own decisions
    - some out-of-date information can be tolerated, but there is a limit
  - if schedules are recomputed too infrequently, then radars cannot keep pace with changes in target behavior
  - need to balance coherence against adaptivity
- Stabilization technique: stochastic activation
  - each radar is periodically given a chance to reschedule
  - but it reschedules only if a random number falls below some fixed, uniform activation probability
- The activation probability allows coherence and adaptivity to be balanced
  - it was expected that the ideal activation probability would depend on, e.g., density of the network
  - but so far a value of ~0.3 has worked well for sparse networks

Experimental Results with Simulator

- Visualization shows two targets being tracked simultaneously
- Radars adapt to target positions
  - middle radars multi-task between targets
- Proof of concept demonstration
  - large scale, quantitative experiments planned
  - meanwhile …



- each blob is an estimated target position
  - green indicates a good estimate
- each target follows an oval track
  - just visible under estimated positions

Distributed, Anytime, Approximate Graph Coloring

- Want an abstract problem with similar properties to sensor coordination
  - for experimental investigation of dynamics without details of scanning
- Distributed, approximate k-coloring of a graph's nodes:
  - each node in a given graph is to be assigned one of k colors
  - such that the fraction of conflicts is minimized
    - where a conflict is an edge that connects nodes of the same color
- Clean metric: (normalized) degree of conflict

$$\Gamma \equiv k \times |\{\{u,v\} \mid \{u,v\} \in E \wedge C_u = C_v\}| \,/\, |E|$$

where u,v are nodes, E is the set of undirected edges and C_u is u's color

  - for a proper coloring $\Gamma=0$; for a random coloring $\Gamma=1$
- Same basic algorithm as for sensor coordination
  - called Fixed Probability (FP)
  - each node undergoes periodic-stochastic activation
  - when activated, a node chooses an optimal color for itself
    - based on what it knows of its neighbors' colors
  - when a node changes color, it broadcasts its new color to adjacent nodes

Summary of Experimental Results

- Activation probabilities of around 0.2 to 0.3 are typically good for sparse graphs
  - higher probabilities lead to incoherence
- The algorithm is scalable in costs and quality of solution
  - per-node, per-step costs depend on edge density
- The algorithm is robust against topological changes and message noise and loss
- Execution does not need to be strictly synchronous
  - the communication latency determines an upper bound on the activation probability
- For very high density graphs, a phase transition is observed
  - proper colorings quickly obtained

Conclusion

- Sensor coordination and graph coloring can both be viewed as distributed constraint optimization
  - where a constraint exists between variables that can influence each other
- The FP algorithm can be view as distributed hill climbing
  - where the variables are essentially distributed (not *parallel* hill climbing)
  - and where the hill climbing metric can be decomposed into local terms
- This problem class and algorithm seem well suited to soft-real-time applications in which approximate solutions are OK
  - most of the computational cost of combinatorial problems is typically incurred in obtaining the last 5% of a solution
- Ongoing research:
  - formally specifying problems as soft, global constraints
  - refining soft, global constraints into soft, local constraints
    - automated support
  - synthesizing executable code from soft, local constraints
    - automated support

Related Work

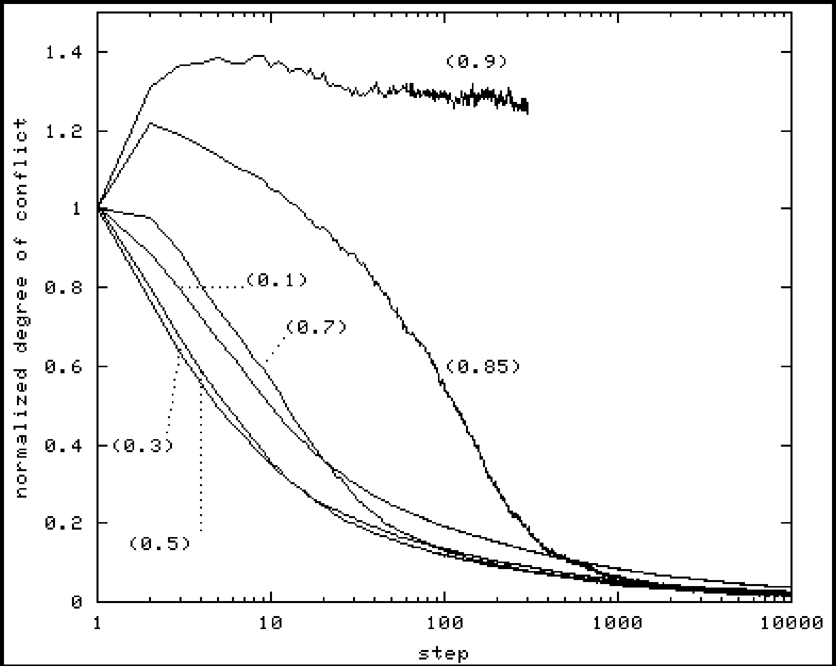
- Stochastic activation is a simple technique to enhance coherency of distributed solutions
  - more sophisticated techniques may produce better results
  - but would need to show they are worth the effort/cost
  - of interest: locally adapting the activation probability for highly irregular networks
- Washington University at St. Louis (Zhang et al.) is conducting experiments comparing the FP algorithm with Distributed Breakout (Yokoo et al.)
  - http://www.cs.wustl.edu/~zhang/projects/dcmp/index.html
- A deterministic FP algorithm was published by Fabiunke
  - deterministic version can cause short-term increases in conflicts
  - when combined with randomization, can reach proper colorings

Extra Slides

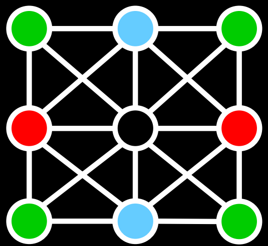
Details of Experimental Results

Experimental Results: Activation Probability

- Synchronous execution
- As expected, high activation probabilities result in incoherence
  - in extreme cases, thrashing results: constant change with no improvement

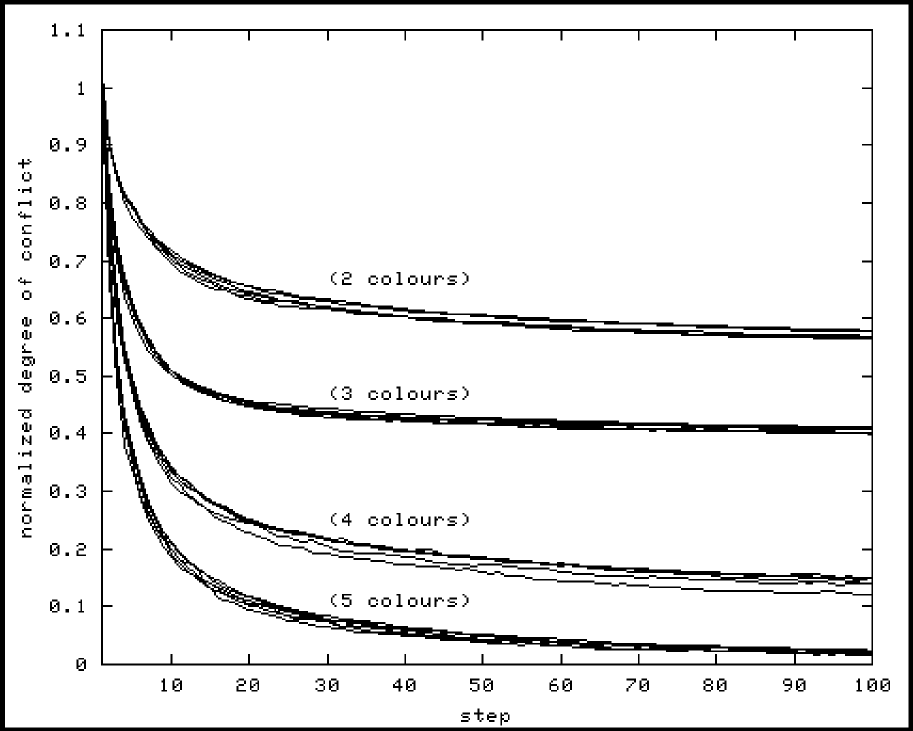


- plot shows effect of various activation probabilities
- results are for regular 2D grids
  - edges along x & y axes and diagonals
  - number of colours = chromatic number = 4
  - 500-5000 nodes
- experiments also performed with random graphs having higher, known chromatic numbers



Scalability

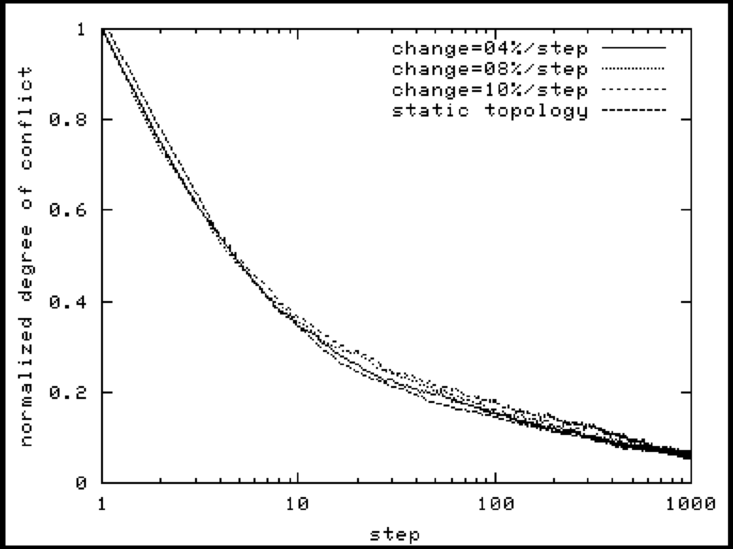
- Per-node, per-step costs are independent of the number of nodes
  - for a given edge density
- Quality of solution is independent of the number of nodes



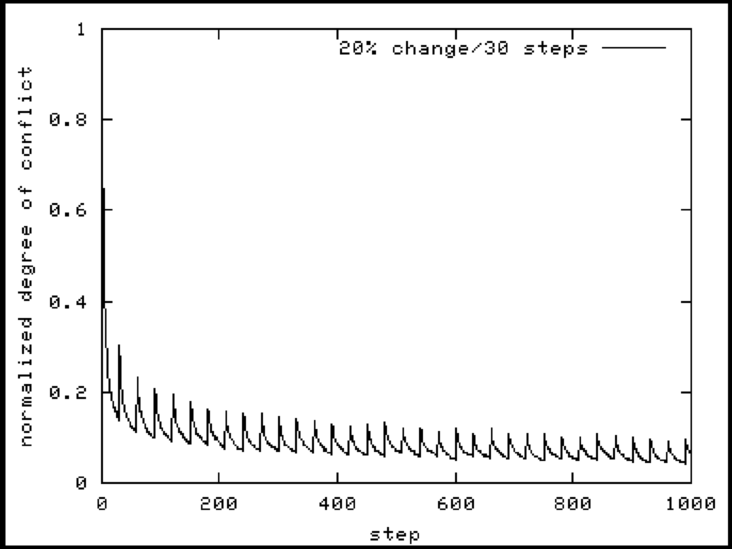
- results shown are for FP(0.3) on 2D grids
- 6 graphs of different sizes (500-5000 nodes)
  - each graph has chromatic number 4
  - each was coloured using 2, 3, 4 & 5 colours

Robustness against Node "Failure"

- Maintain a pool of R randomly selected nodes that have been removed from the graph
  - with period P, restore half of the removed nodes and remove others
  - also remove/restore edges incident to removed/restored nodes
- If the fraction of edges removed is small, the chromatic number of the graph probably does not change
  - changing the chromatic number might cause effects unrelated to robustness



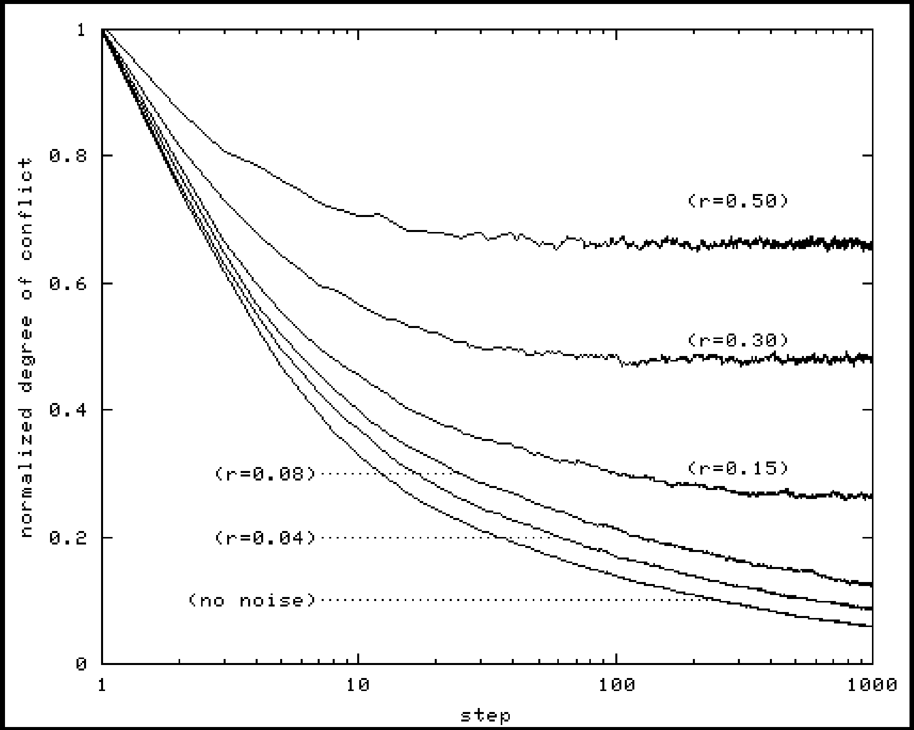
continuous change: P=1, small R
little effect



intermittent change: P=30, large R
spikes in the number of conflicts

Robust Against Communication Noise

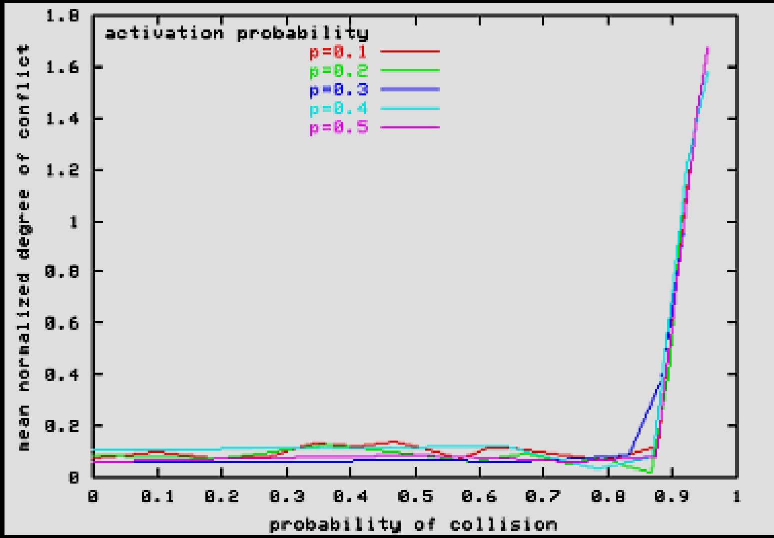
- Subject each color-change message to a probabilistic process that may
  - randomize the color (noise)
  - discard the message (loss)
  - pass the message through unchanged
- Small amounts of noise/loss cause small increases in conflicts

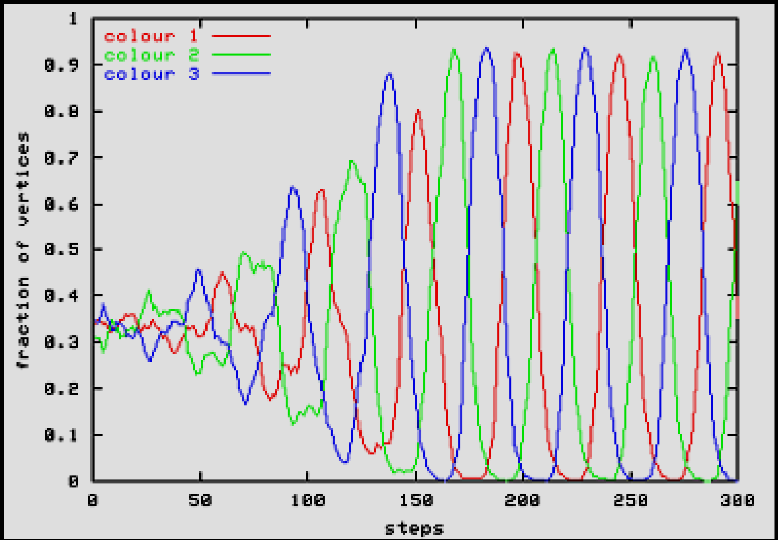


- results shown are for FP(0.3) on 2D grids with 4 colours subject to various amounts of message randomization
- similar results were obtained for small amounts of message loss

Effect of Asynchronous Execution/Latency

- Periodic but asynchronous coloring
  - simplifies implementation on distributed hardware
- Asynchronous execution is OK provided that the activation probability α is low with respect to communication latency L
  - "collision probability" along an edge = $1-(1-\alpha)^L$ < threshold
- Academic interest: extremely high communication latencies cause a "resonance" effect
  - each color is adopted in turn by almost every node simultaneously

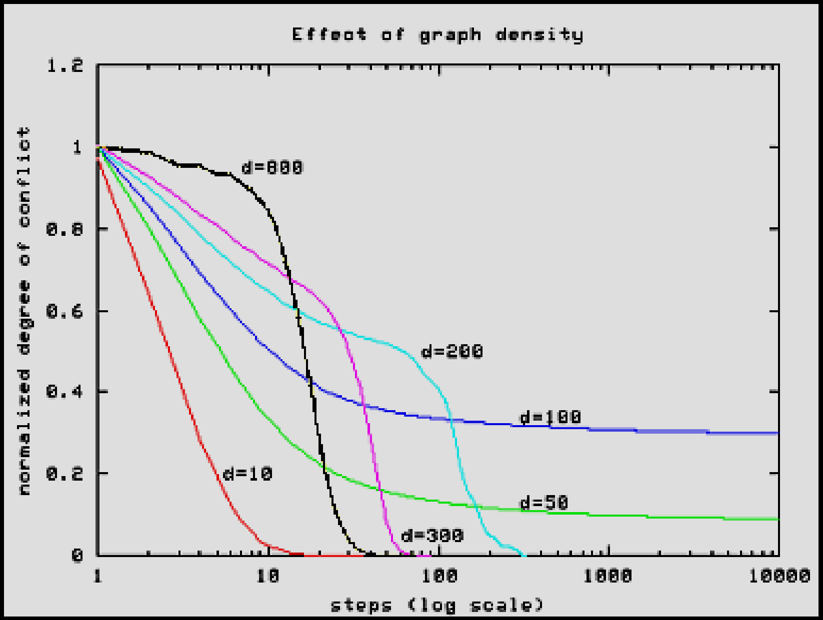




very high latency

Possible Phase Transition w.r.t. Network Density

- For high-density graphs, the degree of conflict increases with the density for a while
- For very-high-density graphs, all conflicts are rapidly eliminated
  - presumably due to large number of backbone variables that implicitly guide the search



- random 20-colorable graphs
- size ~ 2000 nodes
- d is the mean degree