# Asynchronous Execution and Communication Latency in Distributed Constraint Optimization

Stephen Fitzpatrick & Lambert Meertens

Kestrel Institute

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

fitzpatrick@kestrel.edu & meertens@kestrel.edu

http://ants.kestrel.edu/ & http://consona.kestrel.edu/

Outline:

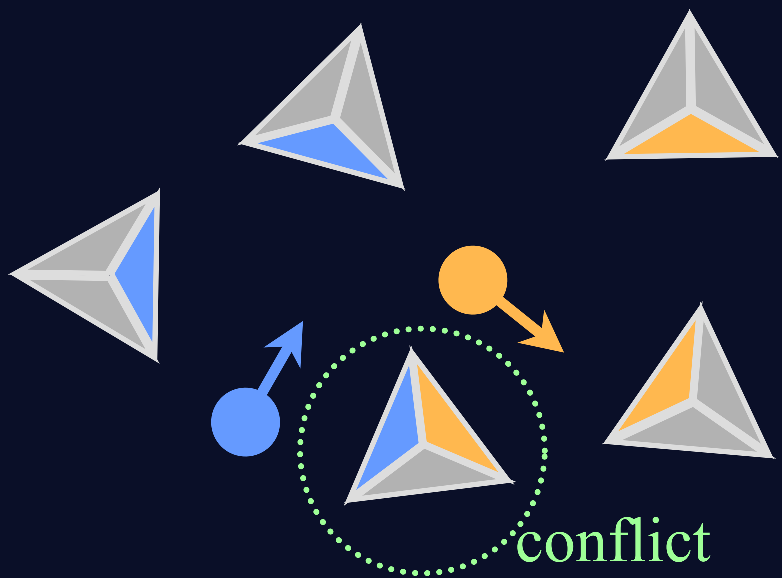
- Motivation: real-time coordination of sensors in a high-latency network
- Modeling coordination as graph colouring
- Soft graph colouring for real-time responsiveness
- A class of distributed anytime algorithms (synchronous)
- Convergence
- Tightness of constraints: conservative variant
- Scalability and robustness
- Asynchronous execution
- Very high communication latencies

*DCR-2002, 16 July, Bologna*

*This work is sponsored in part by DARPA through the `Autonomous Negotiating Teams' program under contract #F30602-00-C-0014 and the `Networked Embedded Software Technology' program under contract #F30602-01-C-0123, both monitored by the Air Force Research Laboratory. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.*

# Motivation: Large Networks of Short-Range Sensors

- Short-range, directional radars
  - each can scan 1 of its 3 sectors at a time
  - each scan acquires range & radial velocity
  - battery-operated – conservation important
- Collaboration needed for tracking
  - 3 approximately-simultaneous scans needed for trilateralization
- Low-power radio communication
  - low bandwidth, high latency
  - reveals positions of radars – minimize

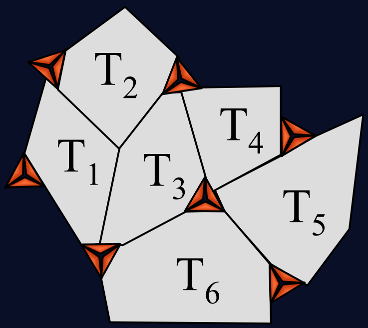



- Coordination mechanism organizes collaboration
  - optimizes simultaneous scanning, minimizes costs
- Must be:
  - scalable (e.g., to $10^5$ sensors)
  - real-time adaptive (e.g., new targets are detected, existing targets disappear)
  - robust (e.g., hardware may fail)

# Inter-Sensor Collaboration

- Main requirement: scan each target simultaneously with 3 radars
  - define virtual resources: *trackers*
  - each tracker is comprised of 3 sectors on nearby radars
    - $T_i \equiv \{R_{i1}:S_{i1}, R_{i2}:S_{i2}, R_{i3}:S_{i3}\}$
  - each tracker can track a single target over some contiguous region
- Main constraint: each radar can scan only 1 sector at a time
  - if two trackers use different sectors on the same radar, they are mutually exclusive
    - $\text{mutually\_exclusive}(T_1, T_2) \Leftrightarrow \exists\, j,k \in \{1, 2, 3\}: R_{1j}=R_{2k} \wedge S_{1j} \neq S_{2k}$
- Compute a cyclic schedule of tracker usage
  - worst-case assumption: all trackers need to be used
  - mutually exclusive trackers cannot be used in the same time slot
  - number of time slots determined by target speed, scan time & revisit period

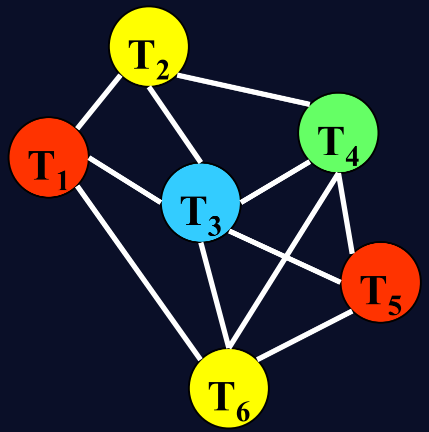
| timeslot # | scan start time (seconds) | scan end time (seconds) | T1 | T2 | T3 | T4 | T5 | T6 |
|---|---|---|---|---|---|---|---|---|
| 1 | 0.0 | 2.0 | X | | | | X | |
| 2 | 2.0 | 4.0 | | X | | | | X |
| 3 | 4.0 | 6.0 | | | X | | | |
| 4 | 6.0 | 8.0 | | | | X | | |



# Modeling Coordination as Graph Colouring

- Each tracker can be mapped to a *node* in an undirected graph
- Each mutual exclusion constraint then maps to an *edge*
  - nodes that are *adjacent* in the graph are mutually exclusive/cannot be used simultaneously
  - two nodes are said to be neighbors iff they are adjacent
- A *proper k-colouring* of the graph's nodes maps to a feasible schedule
  - time slot ⇔ integer in $Z_k$ ⇔ colour

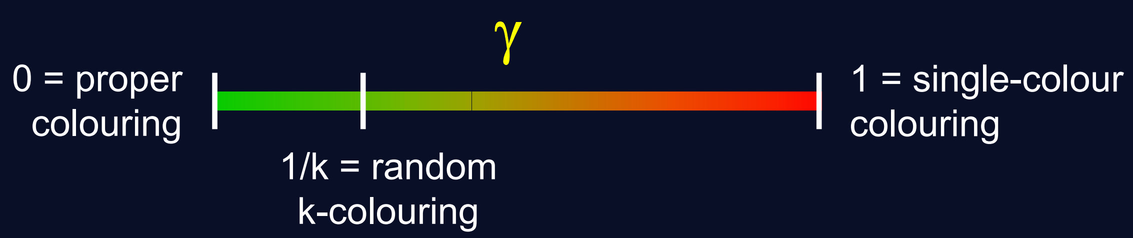
| timeslot # | scan start time (seconds) | scan end time (seconds) | T1 | T2 | T3 | T4 | T5 | T6 |
|---|---|---|---|---|---|---|---|---|
| 1 | 0.0 | 2.0 | | | | | | |
| 2 | 2.0 | 4.0 | | | | | | |
| 3 | 4.0 | 6.0 | | | | | | |
| 4 | 6.0 | 8.0 | | | | | | |



# Soft Graph Colouring

- An edge connecting nodes of the same colour represents a *conflict*
  - some radar has been scheduled to scan two sectors simultaneously
- For real-time adaptation, the number of conflicts must be quickly reduced
  - fast reduction to acceptable levels is more important than total elimination
- Define the *degree of conflict* as the fraction of edges that are conflicts
  - let E be the set of edges and $C_v$ the colour of node v

$$\gamma \equiv \frac{\left|\left\{\{u,v\} \in E \mid C_u = C_v\right\}\right|}{|E|}$$



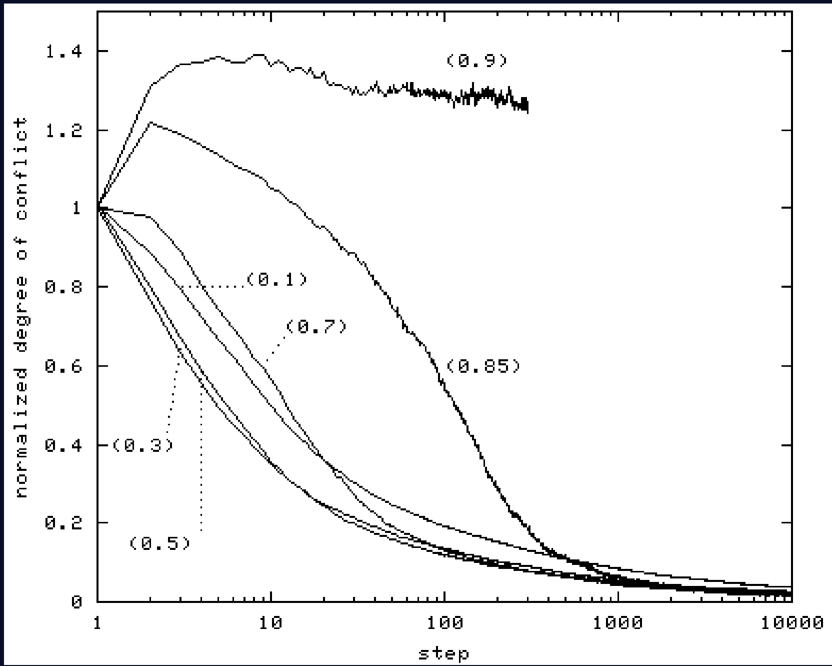

- Normalize: $\Gamma \equiv k\gamma$
  - random k-colouring has an expected $\Gamma$ of 1
- Assessment of coordination mechanism is based on how quickly it reduces $\Gamma$ after random initialization

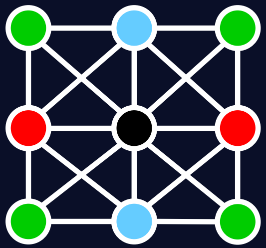
# A Class of Distributed Anytime Algorithms (synchronous)

- Main idea: each node repeatedly chooses its own colour to minimize its conflicts with neighbouring nodes
- Fixed Probability algorithm FP(p) …
  - Initialization:
    - each node chooses a random colour and informs its neighbours
  - Synchronized infinite loop:
    - probabilistic activation
      - a node activates if a randomly generated number falls below some fixed activation level p
    - if a node activates, it non-deterministically chooses its next colour
      - it computes a histogram of colour usage among its neighbours, based on what they last told it
      - it then chooses any colour that is least used in the histogram
      - if the chosen colour differs from its current colour, it tells its neighbours
- Convergence?
  - under the right conditions, the total number of conflicts reduces over time and *may* converge to 0 …

# Effect of Activation Level on Convergence of FP

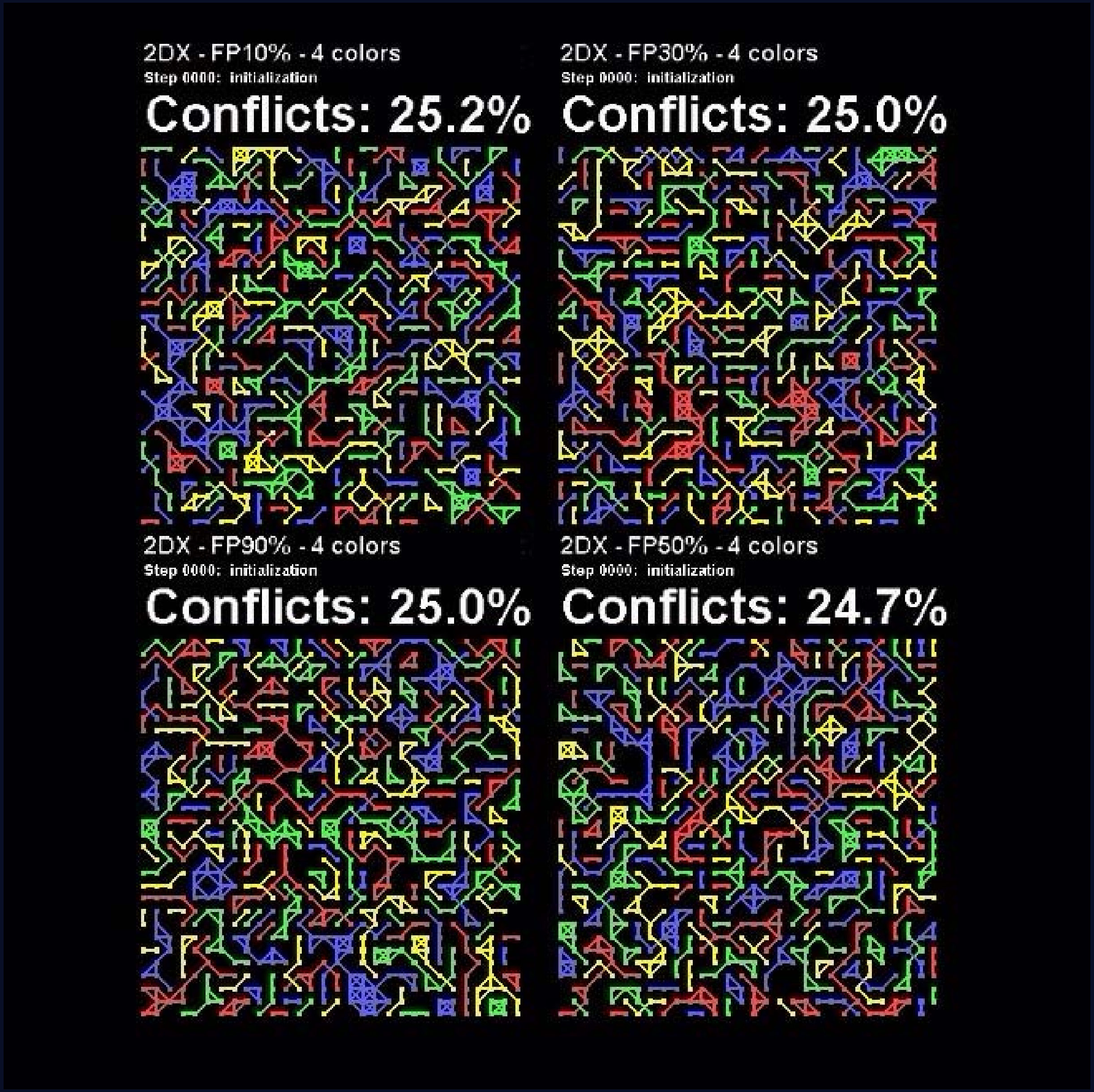
- Measure (normalized) degree of conflict after each synchronous step
  - experiment performed in simulator
- When activation level is too high, thrashing occurs
  - too many neighbours are simultaneously updating colours
  - because of out-of-date information, they make mutually harmful decisions
- When activation level is too low, adaptivity is hindered
  - extreme case is sequential execution
- Need compromise between speed and coherence
  - an activation level of 0.3 seems to be reasonable for sparse graphs
  - this level was used for experiments reported in following slides



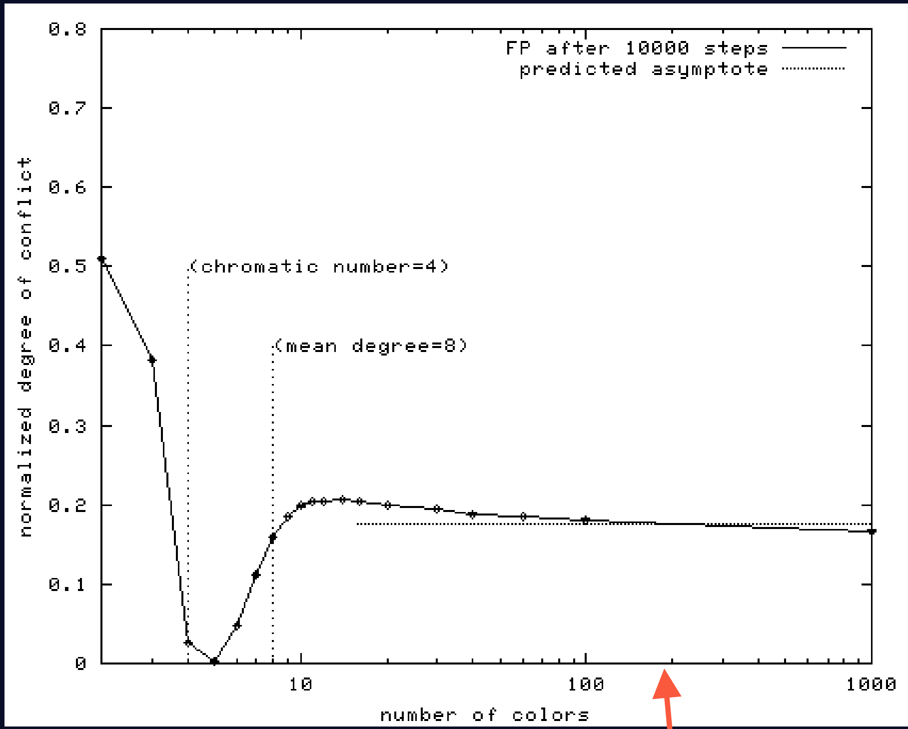
- experimental results shown for 2D grids
  - number of colours = chromatic number = 4
  - 500-5000 nodes
- experiments also performed with random graphs having higher, known chromatic numbers

# Animation: Activation Threshold



# Effect of Tightness of Constraints

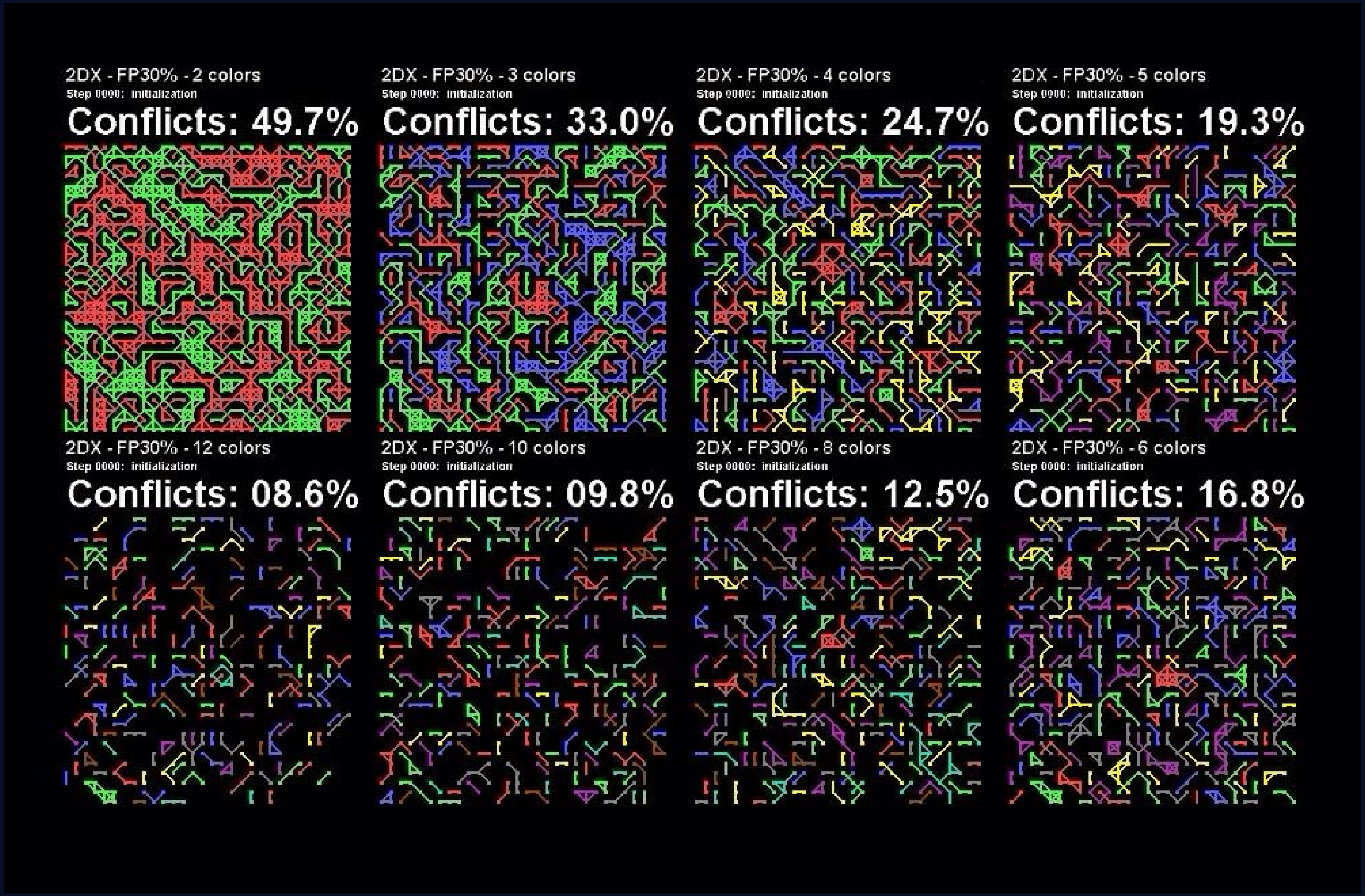
- Performance of FP is good on over-constrained problems
  - where #colours<chromatic number
  - for 2D & 3D grids, observed convergence value of degree of conflict is close to theoretical minimum
- Performance of FP is poor on loosely constrained problems
  - where #colours>>chromatic number
  - intuitively, these are easy problems
- When loosely constrained, each colour choice is essentially random
  - for each given node, most colours are not used by any neighbour
  - FP chooses randomly from among the unused colours
  - asymptotic value predicted as $\alpha/(2-\alpha)$ where $\alpha$ is the activation level




this is *not* a time axis

- experimental results shown for 2D grids
- chromatic number = 4

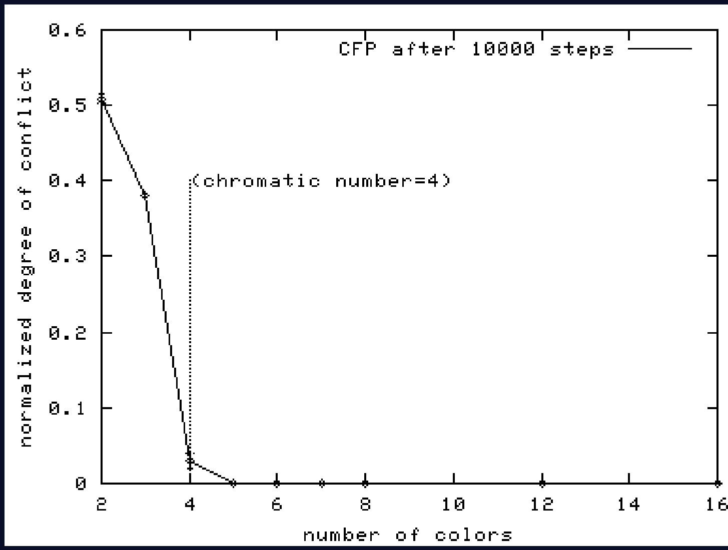
# Animation: Tightness of Constraints



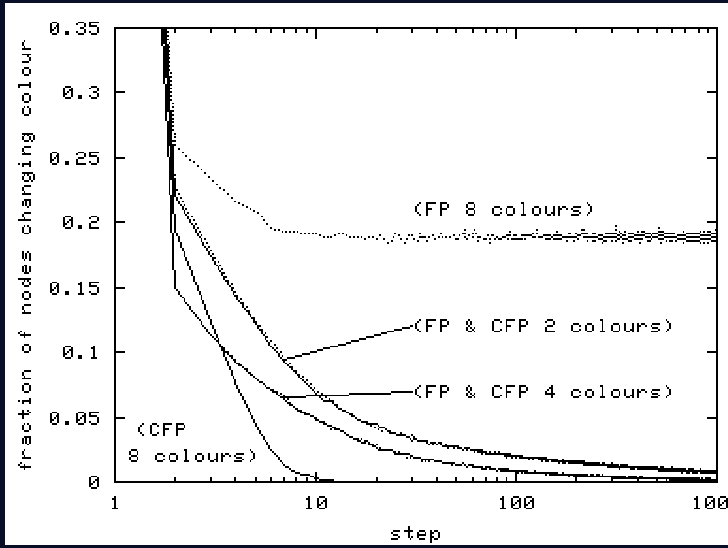
# CFP: Conservative Variant

- Colour choice is non-deterministic
- But activation is restricted
  – in addition to passing the test for random number<activation level, a node may activate *only* if it has a conflict with any neighbour
- Conservative variant has good performance overall
  – communication costs are also better than FP's for loosely constrained problems
    - under FP, node activity continues unabated forever
    - under CFP, node activity decreases with the degree of conflict

conflicts

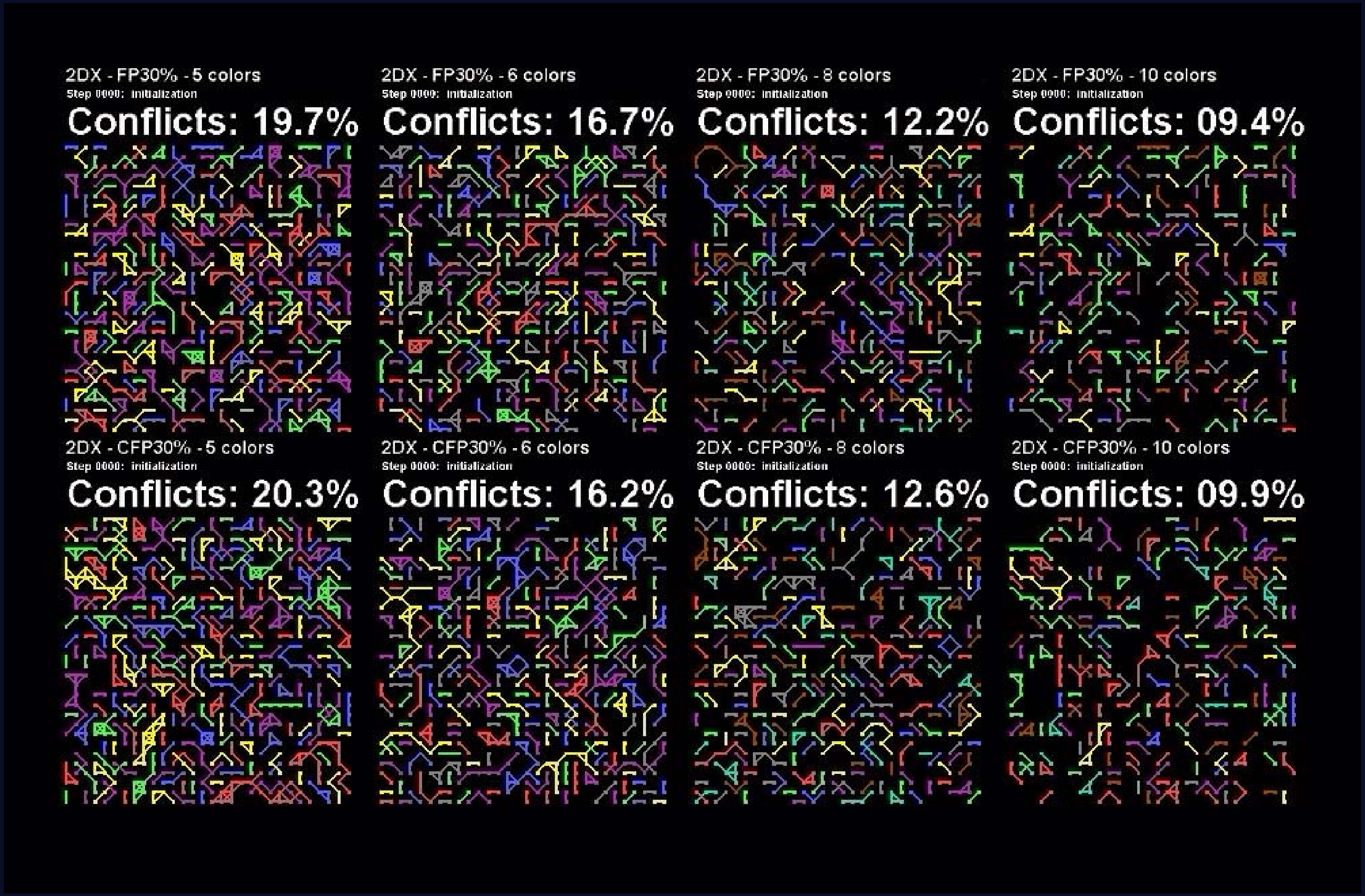



communication rate



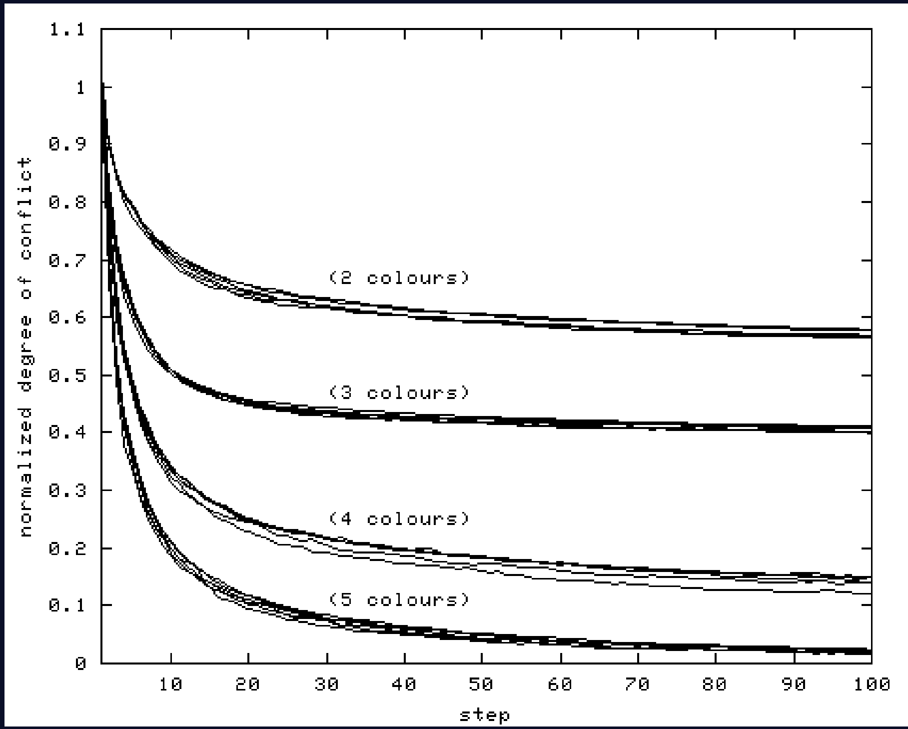

- experimental results shown for 2D grids
- chromatic number = 4

# Animation: FP vs. CFP



# Scalability

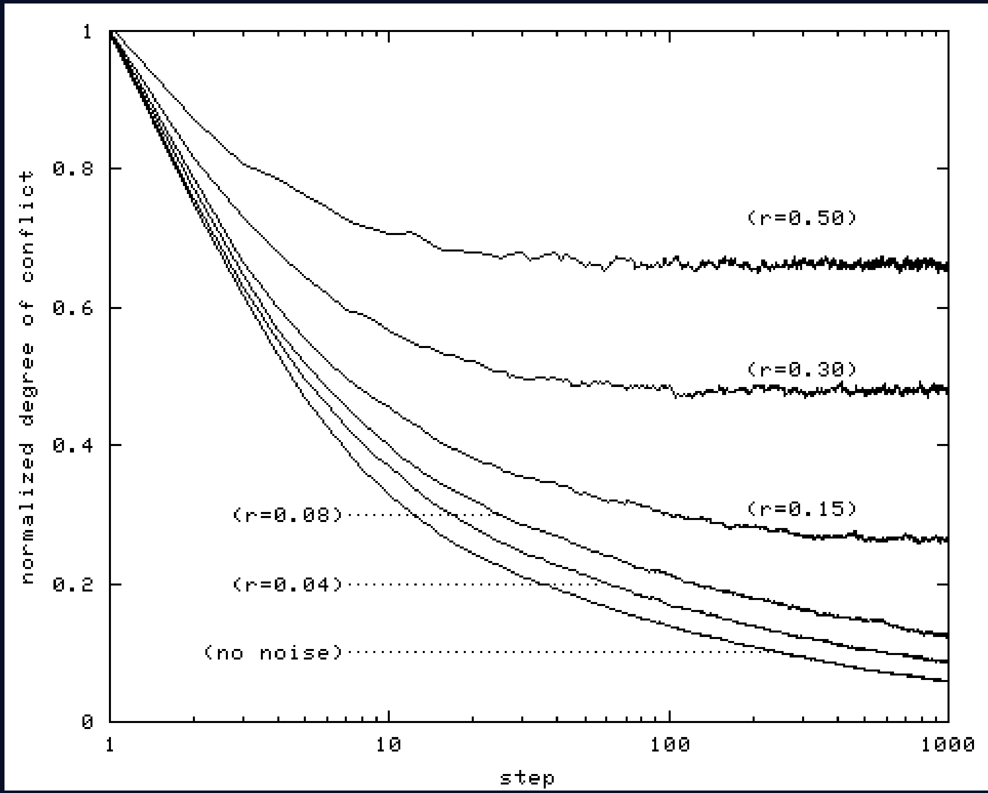
- The algorithm is scalable in cost
  - per node, per step costs depend on (mean) degree of the graph
  - they do not depend on the number of nodes
    - to the extent that the mean degree is independent of the number of nodes
- The algorithm is scalable in performance
  - for large graphs, the reduction in normalized degree of conflict over steps shows little variation for graphs of different sizes




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

# Robust against Communication Noise

- Each colour-change message subjected to random process:
  - probability r, colour randomized
  - probability d, message lost
  - otherwise, message unchanged
- For small amounts of noise, incremental increases in degree of conflict are observed
  - no catastrophic failure



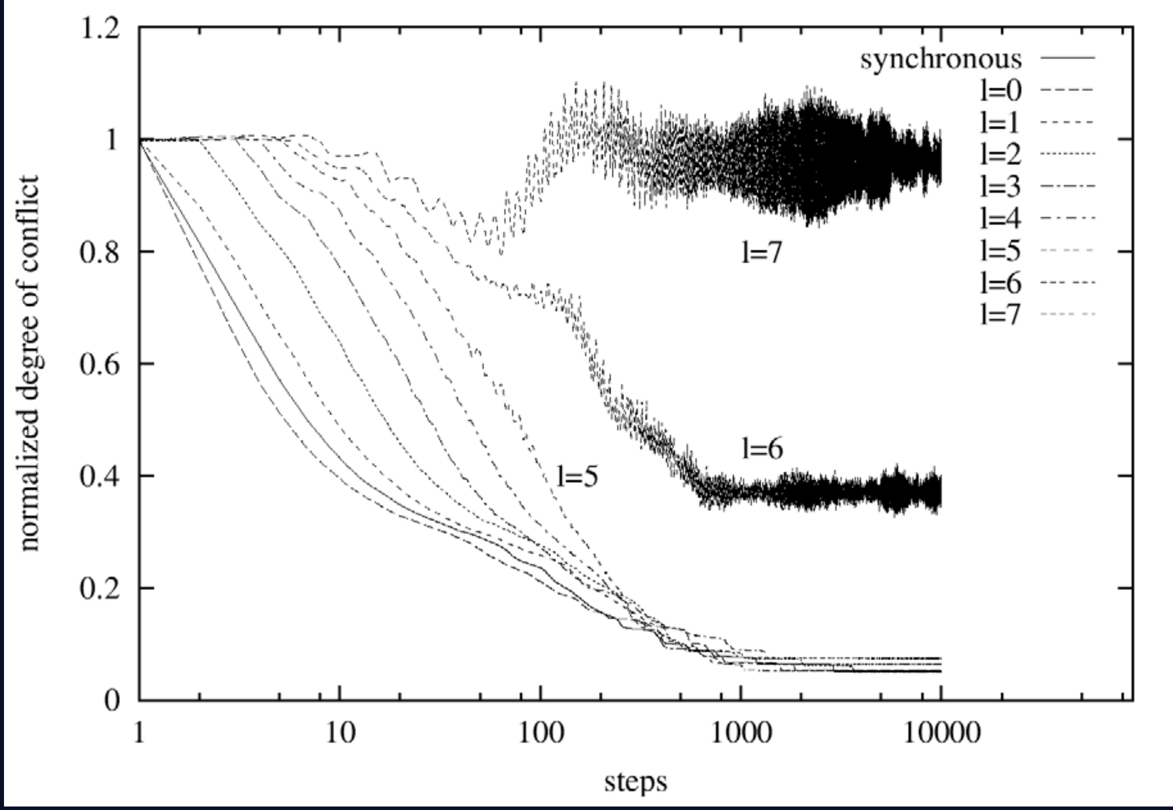

- results shown are for CFP(0.3) on 2D grids with 4 colours subject to various amounts of message randomization
- similar results were obtained for small amounts of message loss

# Asynchronous Execution

- The synchronous FP algorithm requires synchronization, which may:
  - require overhead (e.g. communication cost)
  - slow down the process (wait for the slowest message and node)
  - slow down convergence — or not
- For asynchronous FP the essential idea is the same as for synchronous version, except that execution is asynchronous:
  - *Non-synchronized* infinite loop (but same rate for all nodes):
    - probabilistic activation
      - a node activates if a randomly generated number falls below some fixed activation level p
    - if a node activates, it non-deterministically chooses its next colour
      - it computes a histogram of colour usage among its neighbours, based on what *it last heard from them*
      - it then chooses any colour that is least used in the histogram
      - if the chosen colour differs from its current colour, it tells its neighbours
- Asynchrony may help in symmetry breaking, but communication latency may cause ill-advised changes

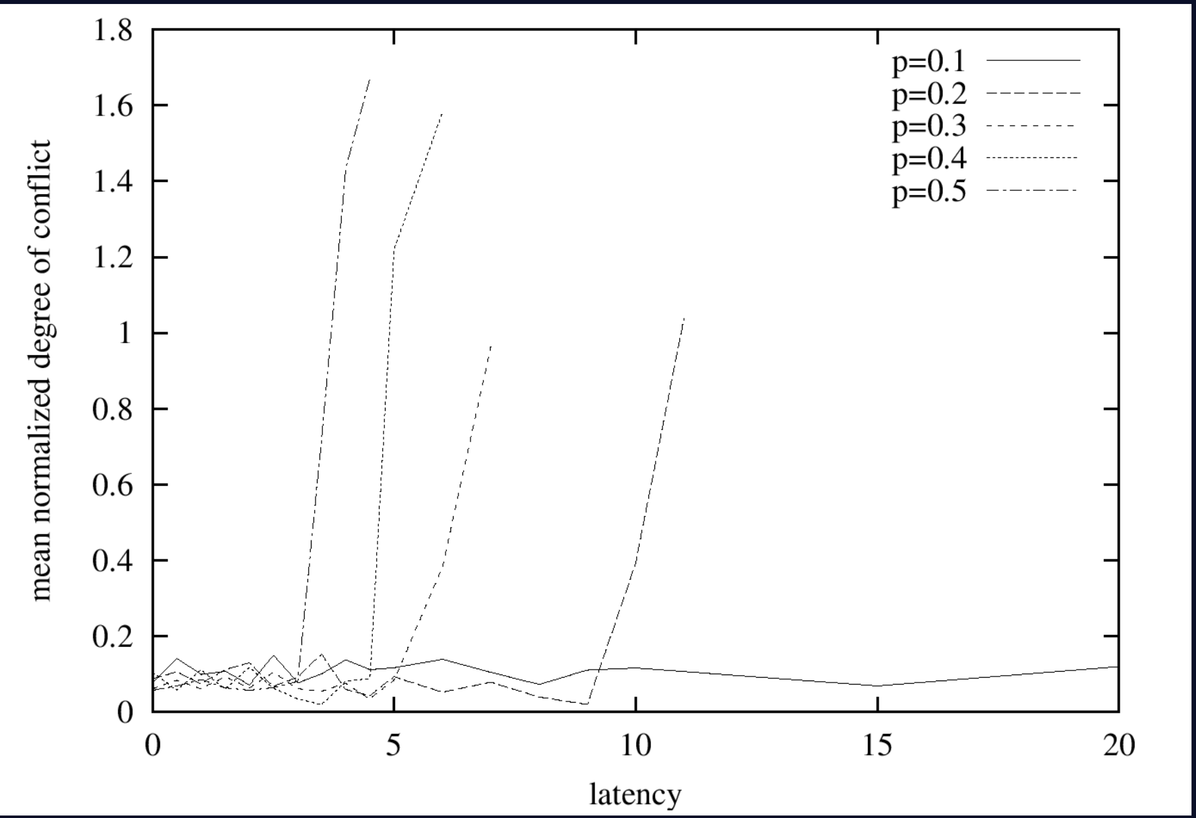
# Effect of Communication Latency

- Performance of asynchronous FP is reasonable for moderate latencies
  - short-term performance degrades (as expected)
  - long-term result quite good
- Performance is even better than synchronous FP when latency < 0.5 time units
- Performance sharply becomes very poor for higher latencies
  - divergence
  - latency = 7 not better than random colouring




- experimental results averaged for 20 random graphs
- p = 0.3
- mean degree = 10
- chromatic number = 3

# Communication Latency and Activation Probability




- Sharp performance drop for higher latencies: the threshold latency decreases as activation probability increases
- This is due to higher probability of “collision” : a colour-change message still travelling along an edge when decision is taken

- degree of conflict averaged over 10,000 steps
- mean degree = 10
- chromatic number = 3

# Effect of Collision Probability

- For activation probability p and latency L,(an upper bound on) the probability of collision is about

  $1 - (1 - p)^L$

- Performance drop indeed depends on collision probability: fine up to about 0.8; bad at 0.9 and higher
- So given latency L, a safe activation probability is:

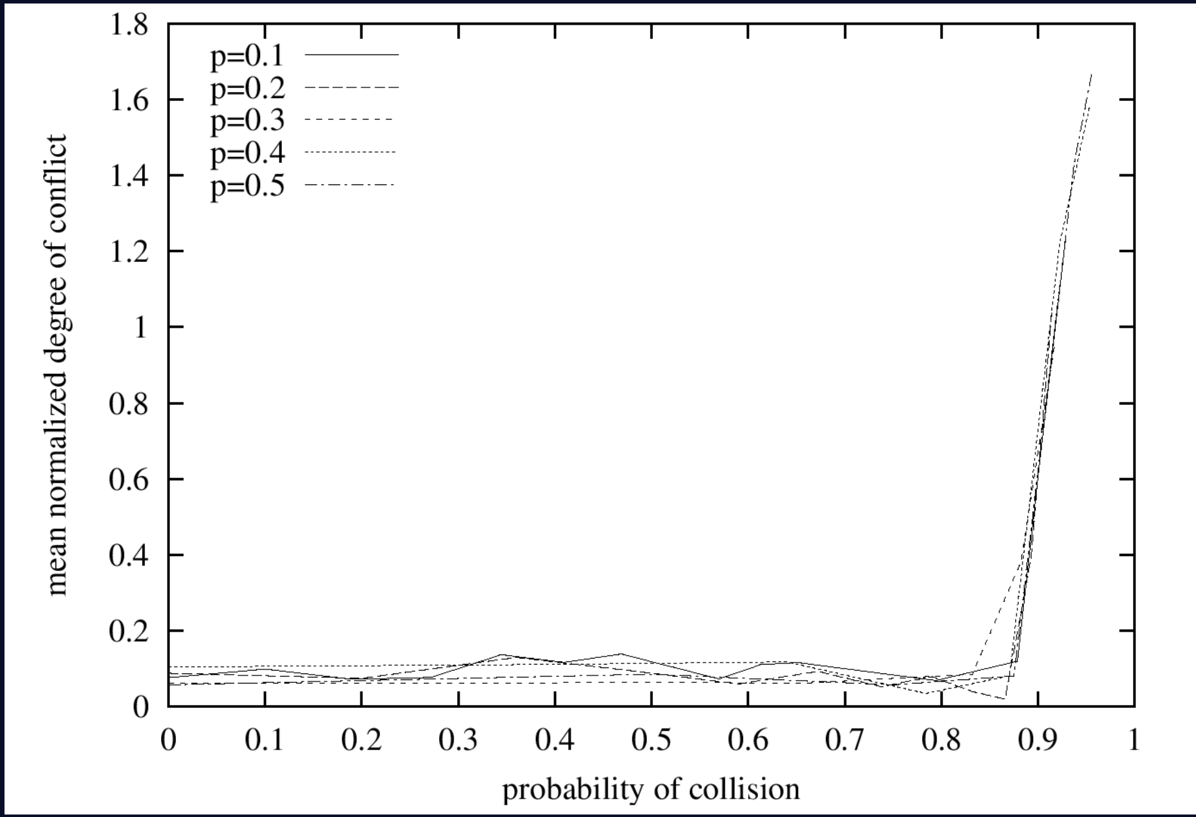
  $p \leq 1 - 0.2^{1/L}$

$L = 1 \rightarrow p \leq 0.80$

$L = 2 \rightarrow p \leq 0.55$

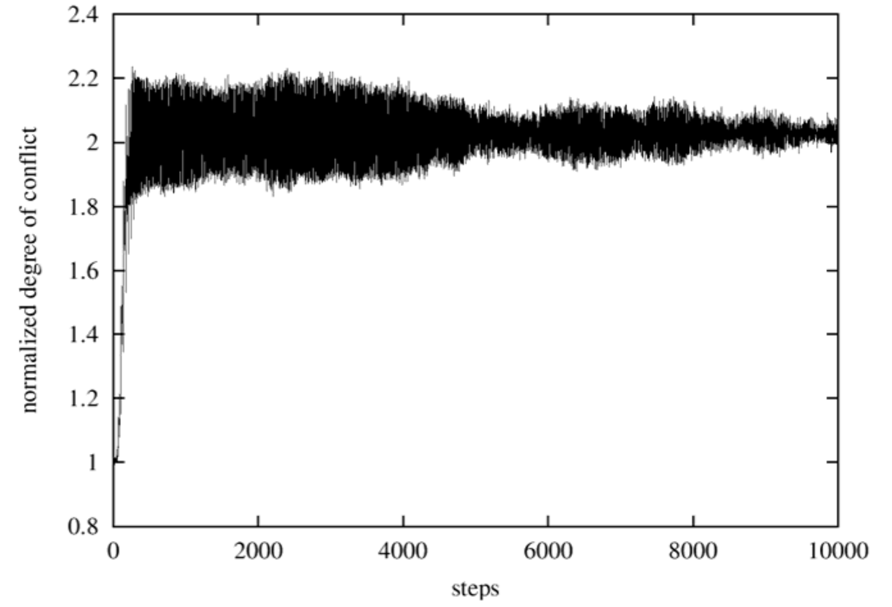
$L = 4 \rightarrow p \leq 0.42$

$L = 8 \rightarrow p \leq 0.18$



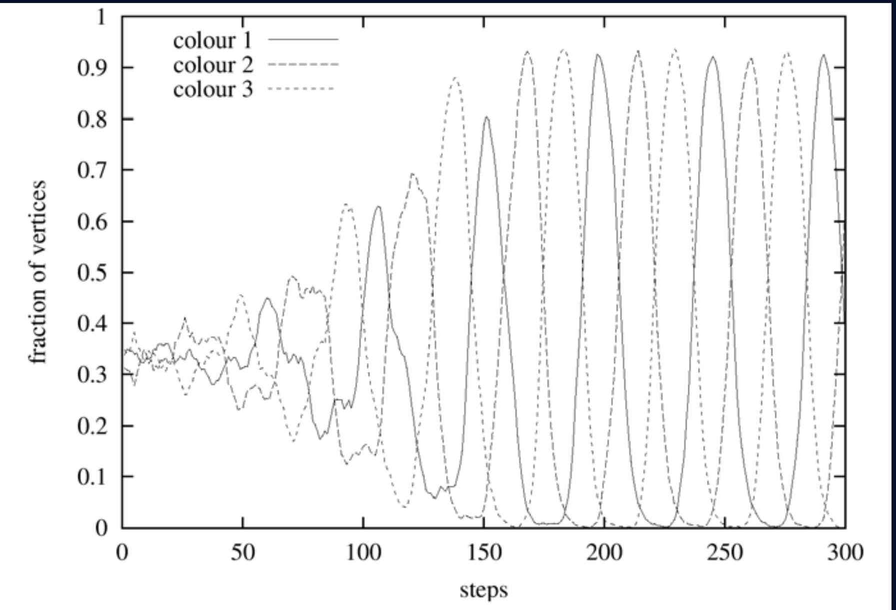

- degree of conflict averaged over 10,000 steps
- mean degree = 10
- chromatic number = 3

# Very High Latencies




- p = 0.3
- L = 15

- Surprise: for very high latencies, the normalized degree of conflict Γ tends to a mean value of approximately 2




- p = 0.3
- L = 10

- For very high latencies, the control mechanism gets caught in an out-of-phase, oscillating trajectory, with period > 2L

# Conclusion

- The FP algorithm is simple but effective for distributed, real-time, approximate colouring of sparse graphs
  - scalable, low-cost, robust
- Basic framework of stochastic activation & local optimization seems appropriate for other distributed constraint problems
  - graph colouring serves as a clean, archetypal problem
- The algorithm has also been tested with dense, random graphs
  - interesting, but different, results
  - proper k-colourings quickly obtained for very dense k-colourable graphs
    - local constraints guide colouring to a unique, proper colouring
- Asynchronous execution and communication latency are handled well
  - provided that the activation probability does not exceed a critical level
- Further work on algorithm
  - non-uniform activation levels, perhaps determined dynamically from local metrics