Experiments on Dense Graphs with a Stochastic, Peer-to-Peer Colorer

Stephen Fitzpatrick & Lambert Meertens

fitzpatrick@kestrel.edu & meertens@kestrel.edu

Kestrel Institute
3260 Hillview Avenue,
Palo Alto, California, U.S.A.
http://ants.kestrel.edu/

Motivation: large coordination problems in soft real time
Framework: distributed constraint optimization
– specialized to distributed, approximate graph coloring
Normalized metric: degree of conflict
Algorithm: peer-to-peer constraint maximization
Experimental results
Motivation: Large Networks of Simple Sensors

• Scenario: many small, cheap sensors scattered over terrain
• Sensors equipped with low-power radio transmitters & receivers
  – permit broadcast communication between geographically close sensors
    • every sensor within range of a transmitting sensor may receive a message
    – 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
Challenges

• Scalability
  – up to $10^5$ sensors

• Real-time adaptivity
  – sensor coordination must keep pace with target behavior
  – good collaboration soon is better than excellent collaboration eventually
  – 5 seconds

• Wide load range
  – number of targets may quickly change from none to many

• Robustness
  – failure of even a significant fraction of the sensors must not cause catastrophic failure of the whole system

• Communication efficiency
  – transmission consumes energy and reveals location
  – 1 message per sensor per second
Distributed Constraint Optimization

• Set of labeled vertices $v_i$
  – domains $\Delta v_i$

• Set of labeled hyper-edges $E \equiv \{ j \rightarrow e_j \}$
  – a hyper-edge is an order sequence of vertices
    • or their labels
  – $e_j \equiv (v_{j1}, v_{j2}, \ldots, v_{jr})$
  – where $jr$ is the edge’s rank

• Each edge is labeled with a penalty function
  – $f_j: \Delta v_{j1} \times \Delta v_{j2} \times \ldots \times \Delta v_{jr} \rightarrow [0,1]$

• Each vertex is to choose a value to minimize the mean penalty ("degree of conflict")
  – $\gamma \equiv \Sigma_j f_j / |E|$
Examples

• Vertex k-Coloring
  – $\Delta v_i \equiv \{1 \ldots k\}$
  – rank of each edge is 2
  – penalty functions are all the equality function
    $\delta_k(x,y) \equiv$ if $x=y$ then 1 else 0
  – penalty functions are symmetric

• Leader election under broadcast communication
  – $\Delta v_i \equiv \{\text{Off}, \text{On}\}$
  – a hyper-edge connects each vertex to all other vertices within a given distance
  – penalty function: let $n$ be number of vertices with value On in edge $j$
    • $f_j(n=0) = 1$
    • $f_j(n=1) = 0$
    • $f_j(n>1) = 1-1/n^2$
  – penalty functions are symmetric
Normalized Metric

• Expected value of $\gamma$ over random assignments
  – $[\gamma] = \sum_j f_j / |E|$  
  – related to the tightness of the constraint

• Normalize: $\Gamma \equiv \gamma / [\gamma]$  
  – $\Gamma=0$ is typically perfect  
    • not achievable in over-constrained systems  
  – $\Gamma=1$ is as good/bad as random  
    • in a distributed system, a random assignment requires no coordination or communication  
  – $\Gamma>1$ is worse than random  
    • indicates a problem with coordination

Vertex k-Coloring

$[\delta_k] = 1/k$
$[\gamma]= 1/k$

loose constraint
independent of graph density

$\Gamma = k\gamma$

<table>
<thead>
<tr>
<th>$\delta_3$</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Algorithm Overview

• Local degree of conflict $\gamma_i \equiv \sum_{j \in \Delta E(i)} f_j / |E(i)|$
  – where $E(i)$ is the subset of the hyper-edges involving vertex $i$

• Main idea: each vertex continually adjusts its own value to minimize its own $\gamma_i$
  – each vertex communicates changes to its neighbors
  – per vertex costs vary with number of neighbors (for bounded domain)
  – robust due to highly distribution and local interaction
  – anytime algorithm generically suited to soft real time
  – convergence to stable solution rather than termination

• Assumption: if every vertex minimizes $\gamma_i$ then overall solution will be good
  – good enough for sensor coordination
  – though probably not a true minimum
Fixed Probability Algorithm
(synchronous, conservative version)

• The vertices repeatedly execute the following steps in lockstep
• Every vertex determines simultaneously whether or not to activate
  – it activates iff $\gamma_i > 0$ and random[0,1) < p
    • where the activation probability p is a fixed number in [0,1]
• If a vertex activates, it attempts to minimize its local degree of conflict
  – according to what it believes are the values of adjacent vertices
  – the method of minimization depends on the nature of the domain
• All vertices that have changed value inform adjacent vertices
  – communication latency is always 1

Vertex k-Coloring
Vertex computes a histogram of neighbors’ colors and chooses a minimum
Effect of Activation Probability

- Activation probability $p$ can be adjusted to balance speed of adaptation against coherence.
- High $p$ causes simultaneous changes by neighbors
  - incoherence due to outdated information
- Low $p$ causes slow adaptation

- 500 vertices
- Mean degree 14.0
- 4-colorable graphs in 2-D space
  randomly partition the vertices into 4 equivalence classes
  randomly add edges between vertices in different classes (that are sufficiently close)

CFP 0.1 (bottom) & CFP 0.9 (top)
Effect of Density

- For sparse graphs, regions of agreement quickly grow
  - but may not entirely reconcile with each other
  - most easily seen in 2-colorings of regular graphs
- As the density increases, the coupling between regions increases
  - initially, reconciling regions becomes more difficult so conflicts increase
  - eventually, the graphs have a small diameter so everything is local and proper colorings crystallize

Γ vs. time

- 900 vertices
- 10-colorable graphs (no spatial aspect)
- edge density varying from ~0.01 to ~0.89
- CFP 0.2
Effect of Density (cont.)

- Can summarize results for a given run by summing $\Gamma$
  - equivalent to area under curves in preceding plots

- Moderate activation probabilities (~0.25) provide good overall performance
  - even for high density graphs

- 900 vertices
- 10-colorable graphs (no spatial aspect)

![Graph showing total $\Gamma$ (summed over 10000 steps) vs. mean degree for different activation probabilities (p=0.1, p=0.2, p=0.3, p=0.4, p=0.5).]
Communication Costs

- Single-step communication cost: fraction of vertices that change color
  - in a distributed system, each color change must be communicated
- For low density, costs vary linearly (approx.) with activation probability
  - more activity leads to more change
- For high density, costs increase more rapidly with activation probability
  - can be viewed as overhead caused by incoherence

- 900 vertices
- 10-colorable graphs (no spatial aspect)

Total communication cost (summed over 10000 steps)
Comparison with Sequential Algorithms

- 900 vertices
- 4-colorable graphs (no spatial aspect)

Non-strict sequential hill-climber
  - 5% tolerance

Greedy heuristic
  - order vertices by decreasing degree
Conclusions

• CFP coordination is simple to implement and cheap to use
  – random number generator probably does not need to be high quality
• Challenge is to adjust the activation probability
  – for many problems, an experimental approach is probably feasible
  – but ideally an optimal probability would be computed from graph characteristics
• Quality of solutions obtained by local optimization can be good
  – for sparse graphs, quality rapidly increases towards optimal
    • well suited to real-time systems
  – for dense graphs, final quality is optimal but initial improvement is poor
    • typically not well suited to real-time systems
• More complex algorithms?
  – could probably do better by coercing larger regions
  – would be difficult to achieve scalable, real-time results