Using Genetic Algorithms to solve the Minimum Labeling Spanning Tree Problem

Final Presentation

Oliver Rourke, oliverr@umd.edu

Advisor: Dr Bruce L. Golden, bgolden@rhsmith.umd.edu
R. H. Smith School of Business
Combinatorial optimization problem first proposed in 1996 [Chang:1996]

Connected Graph - set of vertices and edges.

Each edge has a label

Find the smallest set of labels which gives a connected sub-graph
An example of a labelled spanning tree, and subgraphs

Question: What is the smallest set of labels which induces a connected (sub-) graph?

Complete Graph $G$

Subgraph induced by $\{1, 2, 4\}$ - Connected

Subgraph induced by $\{1, 2\}$ - Not Connected
More about MLST

- NP-complete - 'perfect' algorithm impossible (?)
- Many heuristics have been used including:
  - Variable Neighborhood Search (VNS) - Best
  - Simulated Annealing
  - Pilot Method
  - Reactive Tabu Search
Introduction to Genetic Algorithms (GAs)

- Evolutionary-inspired heuristic for optimization problems
Introduction to Genetic Algorithms (GAs)

- Evolutionary-inspired heuristic for optimization problems
- Population = set of (valid) solutions
- Select, Breed, Replace
Evolutionary-inspired heuristic for optimization problems
Population = set of (valid) solutions
Select, Breed, Replace
Advantages:
- Flexible and adaptable
- Robust performance at global search
- Simple to parallelize
From Xiong, 2005

- Designed to be simple - no fine tuning
- One parameter - \( p \), population size
- Solution: List of labels (gives connected sub-graph)
- Gene: Label in the list
- Modified Genetic Algorithm (MGA), Xiong, 2006 - more intelligent crossover operator
GA Improvements

1: Coin toss: Make crossover/mutation stochastic
2: Keep equally fit offspring over parents
3: Favor mutation: Encourage retention of new material
Two distinct databases:

- 36 sets of instances (10 instances per set) from Cerulli et al. [2005], smaller graphs. Used for validation only.
- 5 sets of 100 randomly generated larger graphs (using technique from Xiong, 2005). Each set has 100 nodes, 0.2 edge density and either 25, 50, 100, 250 or 500 labels.
Serial Testing

- Conducted on Genome cluster at UMD
- All experiments using one processor, instances run sequentially
- Tested on own generated databases (same as above) with \( N = 100, p = 0.2, L = 25, 50, 100, 250, 500 \)
- Run 10 times with different random number seeds
- Stop: Max running time (\( L \times 20\) ms per instance) vs. Max generation count
### Serial GA changes - results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>% Above BKS</th>
<th>% Above Optimum (if known)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Original GA</strong></td>
<td>6.97</td>
<td>3.79</td>
<td>129.3</td>
</tr>
<tr>
<td>Crossover Coin toss</td>
<td>4.7</td>
<td>2.3</td>
<td>139.8</td>
</tr>
<tr>
<td>Mutation Coin Toss</td>
<td>5.1</td>
<td>3.0</td>
<td>130.5</td>
</tr>
<tr>
<td>Keep Equal</td>
<td>8.3</td>
<td>5.1</td>
<td>123.1</td>
</tr>
<tr>
<td>Favor Mutation</td>
<td>5.1</td>
<td>2.4</td>
<td>130.11</td>
</tr>
<tr>
<td>Everything</td>
<td>3.9</td>
<td>2.4</td>
<td>139.7</td>
</tr>
</tbody>
</table>

### Xiong’s Modified GA (MGA) + variants

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>% Above BKS</th>
<th>% Above Optimum (if known)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MGA</td>
<td>4.1</td>
<td>2.2</td>
<td>474.8</td>
</tr>
<tr>
<td>MGA with Everything</td>
<td>2.9</td>
<td>1.5</td>
<td>421.1</td>
</tr>
</tbody>
</table>
Results vs Iteration count (L=50)
Results vs Time (L=50)

- GA
- GA Stochastic
- MGA
- MGA Stochastic
- VNS
All Results vs Time

(a) $L=25$

(b) $L=50$
All Results vs Time

(c) $L=100$

(d) $L=250$
All Results vs Time

(e) $L=500$
Serial Algorithm → Parallel Algorithm

- Why?
  - Speedup
  - Larger Problems
  - Effectively use all available computational resources
Dividing up the Population

Figure: Three different types of GAs showing interaction between individuals (black dots) in the population. a) Panmictic b) Distributed c) Cellular [Alba:2008]
Figure: % above BKS and computational time for a variety of island sizes - no migration (run on 2.2GHz quad-core Intel Core i7)
Allocate different subpopulations to different processors
Communication between subpopulations - better results?
Serial Algorithm $\rightarrow$ Parallel Algorithm

- Allocate different subpopulations to different processors
- Communication between subpopulations - better results?
- Master-slave versus Direct communication (Who?)
- Message Passing versus Shared Memory (How?)
Parallel Structures (Who?)

(a) Master-slave  
(b) Direct communication

Figure: Different approaches to parallel programming
Communication Schemes (How?)

(a) Message Passing

(b) Shared Memory

Figure: Different approaches to inter-processor communication
Local communication

- Arrange subpopulations on grid, define neighborhood on grid [Scharrenbroich: 'CGA-inspired' distributed GA]
- Carry strongest individuals between 'neighboring' subpopulations at certain points in algorithm
Mesh diagrams

Figure: Different mesh arrangement with one possible neighborhood definition in (b)
Local communication

- Arrange subpopulations on grid, define neighborhood on grid [Scharrenbroich: 'CGA-inspired' distributed GA]
- Carry strongest individuals between 'neighboring' subpopulations at certain points in algorithm
- How?
  - Replace weakest individuals locally?
  - Place in 'waiting room' where they can be accessed, not directly replacing...
- When?
  - Regular intervals?
  - When population stagnates
Direct Communication results

- At best: negligible improvement
- Why? Problem, population size, number of processors...
Direct Communication results

- At best: negligible improvement
- Why? Problem, population size, number of processors...
- OR Straight up bad idea (in this case)
Global Communication

- All subpopulations connected through common 'vault'
- Strongest unique solutions found to date stored in vault
Figure: 1D Mesh with vault included
Vault implications

- Communication from subpopulations to vault - best, unique individuals
Vault implications

- Communication from subpopulations to vault - best, unique individuals
- Communication out of vault - occasionally breed with a randomly selected individual out of vault (modify selection)
Vault implications

- Communication from subpopulations to vault - best, unique individuals
- Communication out of vault - occasionally breed with a randomly selected individual out of vault (modify selection)
- Evolution can lose local optima - vault will maintain global optima
  - Simulated Annealing type selection
  - Respawning (Individual?Subpopulation?)
- Conducted on Genome cluster at UMD
- All experiments involving 32 threads with 32 processors and 32 subpopulations (or separate VNS trials)
- Tested on own generated databases (same as above) with N = 100, p = 0.2, L = 25, 50, 100, 250, 500
- Run 10 times with different random number seeds
- Max running time = L*20ms per instance (for each processor)
All Results vs Time

(a) $L=25$

(b) $L=50$
All Results vs Time

(c) $L=100$

(d) $L=250$
All Results vs Time

\[ L = 500 \]
Vault evolution (separate exploration of search space with investigation of interesting areas)
Vault extensions

- Vault evolution (separate exploration of search space with investigation of interesting areas)
- Replace unique with different enough (Hamming or Levenshtein distance...)

- C++ with pthreads
- Run on Genome cluster at UMD - Quad-Core AMD Opteron® Processor 8382 (2.6GHz)
Genetic Algorithm: Compared with original code from Xiong et al. [2005]. When random seed set correctly and random numbers sampled in the same order, returned the same results.

VNS: Results compared with the results reported in Consoli (2009). Similar results achieved (no statistically significant difference)
Parallel Validation

- Remove all inter-processor communication, record results from each processor individually. Check similar to serial code.
- Verify the sending and receiving for each type of message on both ends
- Investigate speed-up (on 32 processor machine, 32 subpopulations for parallel code):

<table>
<thead>
<tr>
<th>Instance Set</th>
<th>L=100</th>
<th>L=500</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time for serial (s)</td>
<td>2.38</td>
<td>5.42</td>
</tr>
<tr>
<td>Parallel-No Comm (s)</td>
<td>3.75</td>
<td>8.25</td>
</tr>
<tr>
<td>Parallel-Synchronized Iterations (s)</td>
<td>4.35</td>
<td>10.89</td>
</tr>
<tr>
<td>Parallel-Synchronized+Vault (s)</td>
<td>4.51</td>
<td>12.03</td>
</tr>
</tbody>
</table>
Goals

- Create own competitive, efficient, serial GA code
Genetic Algorithm for the MLST

Oliver Rourke

The MLST
Genetic Algorithms
My Serial GAs
Parallel GAs
Software/Validation etc...

Goals

- Create own competitive, efficient, serial GA code
- Convert to an efficient parallel GA, first synchronous and later asynchronous.
Goals

- Create own competitive, efficient, serial GA code
- Convert to an efficient parallel GA, first synchronous and later asynchronous.
- Fine tune parallel GA (investigate migration operators)
Goals

- Create own competitive, efficient, serial GA code
- Convert to an efficient parallel GA, first synchronous and later asynchronous.
- Fine tune parallel GA (investigate migration operators)
- Run optimized code on large array of processors
 Deliverables

- Efficient, competitive serial GA code for the MLST
- Efficient, asynchronous and synchronous parallel GA code for the MLST
- Results from running code on appropriate machines
- Report, presentation
A sincere thank you to Dr Golden for so much invaluable advice, patient listening and watching over the whole project.

Thanks also go to Drs Ide and Balan for help with presenting work and other advice, to Dr Zimin for help with the Genome cluster and many pointers to do with parallel computing, and to all of you for listening and for your questions.
Bibliography

- Canyurt, O. And Hajela, P., Cellular Genetic algorithm technique for the multicriterion design optimization, Struct. Multidisc Optim 20, 2010
- Fujimoto, N. and Tsutsui, S., A Highly-Parallel TSP Solver for a GPU Computing Platform, LNCS 6046, 2011
Bibliography (cont.)


