Using Genetic Algorithms to solve the Minimum Labeling Spanning Tree Problem

Oliver Rourke, oliverr@umd.edu

Advisor: Dr Bruce L. Golden, bgolden@rhsmith.umd.edu
R. H. Smith School of Business

Abstract: Cellular Genetic Algorithms (CGAs) have shown themselves to be very powerful tools for combinatorial optimization. Through this project I hope to investigate CGAs, develop a parallel implementation of a CGA, use these techniques on the Minimum Labeling Spanning Tree Problem, and compare results with other heuristics.

Introduction to MLST

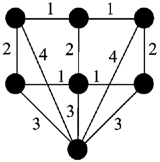
- First proposed in 1996 [Chang:1996]- variant on minimum weight spanning tree

- Connected Graph - set of vertices and edges.
- Each edge has a color
- Find the smallest set of colors which gives a connected sub-graph

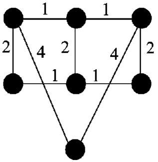
An example of a labelled spanning tree, and some feasible solutions [Xiong:2005]

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

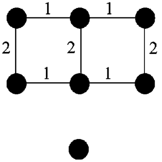
Complete Graph G



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



Subgraph induced by {1, 2} - Not Connected



Introduction to MLST

- First proposed in 1996 [Chang:1996]- variant on minimum weight spanning tree
- Shown to be NP-complete
- Two heuristics and an exhaustive search proposed in the original paper - heuristics achieved moderate success

Introduction to Genetic Algorithms (GAs)

- Evolutionary-inspired heuristic for optimization problems

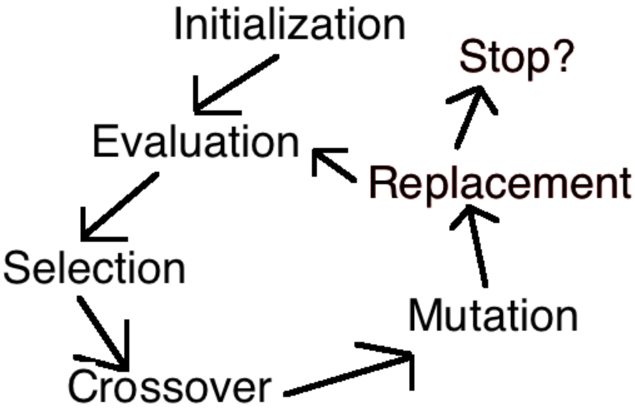
Introduction to Genetic Algorithms (GAs)

- Evolutionary-inspired heuristic for optimization problems
- Population = set of solutions
- Select, Breed, Replace

Introduction to Genetic Algorithms (GAs)

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

Key steps in a Genetic Algorithm



One Parameter GA for MLST - Serial

- From Xiong, 2005
- Designed to be simple - no fine tuning
- One parameter - p, population size
- Representation: List of labels
- Gene: Label in the list

Step 1: Initialization

- Create first generation of individuals - viable, varied

Step 1: Initialization

- Create first generation of individuals - viable, varied
- Initialization from Xiong:2005:
  For each individual in population:
      Individual = {}
      While Individual Is Not Viable:
          Individual.AddRandomColor()

Step 2: Evaluation

- Defined by problem
- For some problems can be extremely time consuming
- Multiple criteria
  $\rightarrow$Penalty functions?

Step 2: Evaluation

- Defined by problem
- For some problems can be extremely time consuming
- Multiple criteria
  $\rightarrow$Penalty functions?
- Evaluation in Xiong:2005:
  $\mathrm{Eval}(T) = \mathrm{len}(T)$

Step 3: Selection

- How? Random, Sweep, ...
- Favor strongest?

Step 3: Selection

- How? Random, Sweep, ...
- Favor strongest?
- Selection in Xiong:2005;
  for j = 1:Size(Population)
  Offspring(j) = Breed(Parent(j), parent(($j + k$) mod p))
  (where k is the generation number)

Step 4: Crossover

- Combine genes from parents to produce viable offspring
- Choose genes randomly? Follow order (pick 'strongest' genes first)?

Step 4: Crossover

- Combine genes from parents to produce viable offspring
- Choose genes randomly? Follow order (pick 'strongest' genes first)?
- Crossover in Xiong:2005:
  $S =$ Union of genes (colors) from both parents
  Sort(S) %According to frequency of labels in Graph
  $T = \{\}$
  while T Is Not Viable:
      T.AddLabel(NextLabel(S))
  return T

Crossover operator

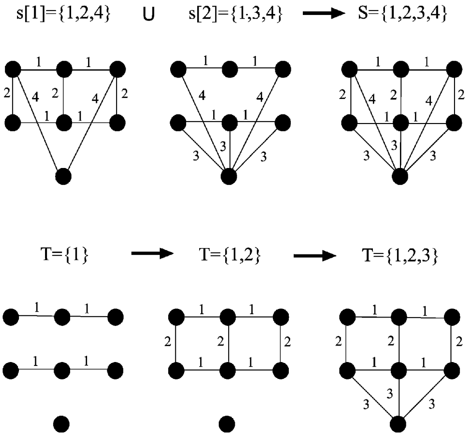


Figure: The crossover operator used in Xiong's GA [Xiong:2005]

Step 5: Mutation

- Introduce new genetic material
- Typically done with small probability

Step 5: Mutation

- Introduce new genetic material
- Typically done with small probability
- Mutation in Xiong:2005 (100% chance of mutation):

```
T.AddRandomColor
Sort(T) %According to frequency of labels in Graph
For Label in T(-1:-1:): %Reverse iterate
    T.Remove(Label)
    if T Is Not Viable:
       if T.Add(Label)
return T
```

Mutation operator

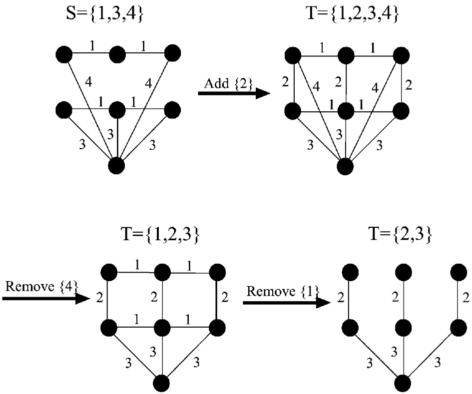


Figure: The mutation operator used in Xiong's GA [Xiong:2005]

Step 6: Replacement

- Find new generation from strongest offspring and parents
- Replace parents where warranted

Step 6: Replacement

- Find new generation from strongest offspring and parents
- Replace parents where warranted
- Replacement in Xiong:2005:
  If Eval(Offspring)<Eval(Parent):
      Parent.Replace(Offspring)

Step 7: Stopping Conditions

- Generations count/computational time
- Population Stagnant

Step 7: Stopping Conditions

- Generations count/computational time
- Population Stagnant
- Stopping Condition in Xiong:2005: Do p generations

GA improvements

- Improve Crossover/Mutation operators?
- Make crossover/mutation stochastic. Mix up ordering

GA improvements

- Improve Crossover/Mutation operators?
- Make crossover/mutation stochastic. Mix up ordering
- Favor retention of mutated genes?
- Keep equally good offspring?

GA improvements

- Improve Crossover/Mutation operators?
- Make crossover/mutation stochastic. Mix up ordering
- Favor retention of mutated genes?
- Keep equally good offspring?
- Divide up population space - promote diversity

3 Different types of GA

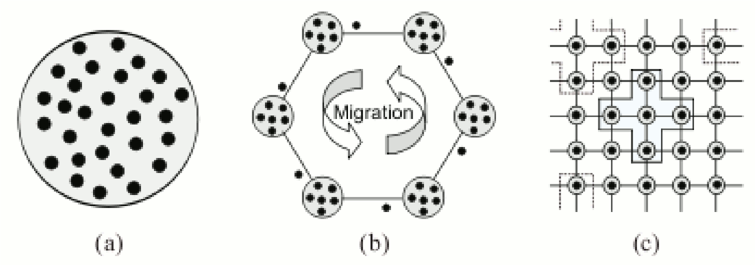


Figure: Three different types of GAs showing interaction between individuals (black dots) in the population. a)Panmictic b) Distributed c) Cellular [Alba:2008]

Genetic Algorithm $->$ Cellular Genetic Algorithm

- Modify Selection operator- limit to neighborhood on grid

Genetic Algorithm $->$ Cellular Genetic Algorithm

- Modify Selection operator- limit to neighborhood on grid
- Arrangement of entire population space
- Neighborhood size?

Genetic Algorithm $->$ Cellular Genetic Algorithm

- Modify Selection operator- limit to neighborhood on grid
- Arrangement of entire population space
- Neighborhood size?
- Choosing within neighborhood:
  - Step through neighborhood
  - Randomly choose one
  - Pick 'strongest' neighbor?

Serial Cellular Genetic Algorithm $- >$ Parallel Cellular Genetic Algorithm

- Why?
  - Speedup
  - Larger Problems

Serial Cellular Genetic Algorithm $->$ Parallel Cellular Genetic Algorithm

- Why?
  - Speedup
  - Larger Problems
- Allocate nodes to separate processors

Serial Cellular Genetic Algorithm −> Parallel Cellular Genetic Algorithm

- Why?
  - Speedup
  - Larger Problems
- Allocate nodes to separate processors
- Master-slave vs. direct communication

Parallel Structures

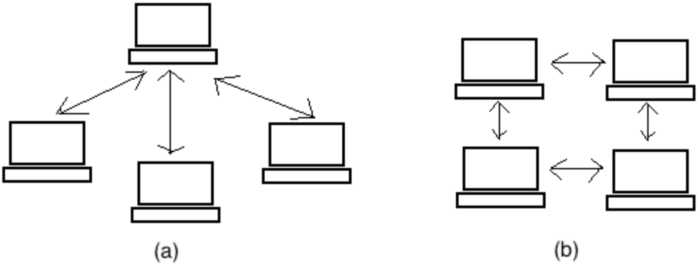


Figure: Different approaches to parallel programming. (a) Master/Slave configuration and (b) Inter processor communication

Cellular Genetic Algorithm $->$ Parallel Cellular Genetic Algorithm

- Why?
  - Speedup
  - Larger Problems
- Allocate nodes to separate processors
- Master-slave vs. direct communication

Cellular Genetic Algorithm $->$ Parallel Cellular Genetic Algorithm

- Why?
  - Speedup
  - Larger Problems
- Allocate nodes to separate processors
- Master-slave vs. direct communication
- Lock nodes when in use. Queues?

Cellular Genetic Algorithm $->$ Parallel Cellular Genetic Algorithm

- Why?
  - Speedup
  - Larger Problems
- Allocate nodes to separate processors
- Master-slave vs. direct communication
- Lock nodes when in use. Queues?
- Synchronous (simultaneous) vs asynchronous

Hardware/Software/Databases

Language - C++ with MPI (Message Passing Interface)
Hardware - array of processors at UMD
Database: Randomly generated labeled spanning trees

Validation/Testing

- Comparing my serial CGA with other heuristics and with global optimum (if known, e.g. through exhaustive search)
- Compare parallel results with serial CGA, ensure as expected (feasible, function in the right range)
- Calculate speedup of parallel vs. serial, asynchronous vs. synchronous [Fujimoto:2011, Vidal:2010, Drummond:2001, Groer:2010]

Schedule: Part I

Part 1: Creating my serial Cellular Genetic Algorithm
Tasks:

- Adding improvements to the Genetic Algorithm
- Modifying selection operator/imposing grid structure so becomes CGA

Timing: Sept - Oct 2010
Result: Competitive, efficient serial CGA code
Validation: Compare computational effort, results with other heuristics (e.g. GA from Xiong, 2005)

Schedule: Part 2

Part 2: Going parallel

Tasks:

- Initially converting to synchronous code - direct communication, locking nodes ...
- Converting synchronous code to asynchronous code

Timing: Nov 2010- Jan 2011

Result: Efficient, parallel, asynchronous CGA code using direct communication

Validation:

- Check results match serial code
- Check speed-up rate of synchronous code over serial code (hopefully equal to number of processors)
- Check speed-up of asynchronous code over synchronous code

Schedule: Part 3

Part 3: Fine tuning/Polishing

Tasks:

- Determine optimum parameters, neighborhood/population space arrangement etc.
- Further optimize code if possible

Timing: Feb 2011

Result: Efficient, competitive, parallel, asynchronous CGA code using direct communication

Validation: Compare with earlier version of algorithm/with other algorithms used in literature

Schedule: Part 4

Part 4: Running on massive array/Reporting

Tasks:

- Run on powerful array of processors
- Prepare final report/presentation

Timing: Mar 2011-

Result:

- Results for larger problems than attempted earlier (incl % optimal, speed-up results ...)
- Parallel, asynchronous, competitive Cellular Genetic Algorithm code for the MLST using direct processor-processor communication
- Final report/presentation

Bibliography

- Alba, E. and Dorronsoro, B., Cellular Genetic Algorithms, Springer, NY, 2008
- Back, T., Hammel., U and Schwefel, H., Evolutionary Computation: Comments on the History and Current State, IEEE Transactions on evolutionary computation, Vo. 1, No 1, 1997
- Canyurt, O. And Hajela, P., Cellular Genetic algorithm technique for the multicriterion design optimization, Struct. Multidisc Optim 20, 2010
- Chang, R. and Leu, S., The minimum labeling spanning trees, Information Processing Letters 63, 1997
- Drummond, L., Ochi, L. and Vianna, D., An Asynchronous parallel metaheuristic for the period vehicle routing problem, Future Generation Computer Systems 17, 2001
- Groer, C., Golden, B. and Wasil, E., A Parallel Algorithm for the Vehicle Routing Problem, INFORMS Journal on Computing, 2010
- Fujimoto, N. and Tsutsui, S., A Highly-Parallel TSP Solver for a GPU Computing Platform, LNCS 6046, 2011
- Huy, N. et al., Adaptive Cellular Memttic Algorithms, Evolutionary Computation 17(2), 2009
- Karova, M., Smarkov, V. and Penev, S, Genetic operators crossover and mutation in solving the TSP problem, International conference on computer systems and technologies, 2005. Katayama, Hirabayashi, Naruhusa, Performance Analysis for Crossover Operators of Genetic Algorithm, Systems and Computers in Japan, Vol 30., No 2., 1999
- Papaioannou, G. and Wilson, J., The evolution of cell formation problem methodologies based on recent studies (1997-2008): Review and directions for future research, European Journal of Operational Research 206, 2010
- Paszkowicz, W., Properties of a Genetic algorithm extended by a random self-learning operator and asymmetric mutations: A convergence study for a task of powder-pattern indexing, Analytics Chimica Acta 566 (2006)
- Sarma J., and De Jong, K., An analysis of the effect of the neighborhood size and shape on local selection algorithms. In H.M. Voigt, W. Ebeling, I. Rechenberg, and H.P. Schwefel, editors, Proc. of the International Confer- ence on Parallel Problem Solving from Nature IV (PPSN-IV), volume 1141 of Lecture Notes in Computer Science (LNCS), pages 236244. Springer-Verlag, Heidelberg, 1996.

Bibliography (cont.)

- Simoncini D., et al., From Cells to Islands: An Unified Model of Cellular Parallel Genetic Algorithms, ACRI, 2006.
- Seredynski, F. and Zomaya, A., Sequential and Parallel Automata-Based Scheduling Algorithms, IEE Transactions on Parallel and Distributed Systems, Vol 13, No 10, 2002
- Serpell, M. and Smith, E., Self-Adaptation of Mutation Operator and Probability for Permutation Representation in Genetic Algorithms, Evolutionary Computation 18(3), 2010
- Vidal, P. and Alba, E., A Multi-GPU Implementation of a Cellular Genetic Algorithm, IEEE 2010
- Xiong, Y., Golden, B. and Wasil, E., A One-Parameter Genetic Algorithm for the Minimum labeling Spanning Tree problem, IEEE Transactions on evolutionary computation, Vol. 9, No. 1, 2005
- Xiong, Y., Golden, B., and Wasil, E., Improved Heuristic for the Minimum Label Spanning Tree Problem, IEEE Transactions on evolutionary computing, Vol 10., No. 6, 2006