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

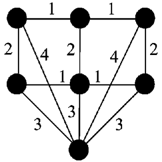
Introduction to Minimum Labelling Spanning Tree Problem (MLST)

- 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

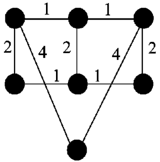
An example of a labelled spanning tree, and subgraphs

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

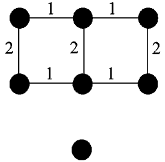
Complete Graph G



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
  - 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

Introduction to Genetic Algorithms (GAs)

- 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

One Parameter GA for MLST - Serial

- 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

Step 1: Initialization

- Create a varied, viable population for generation 1.

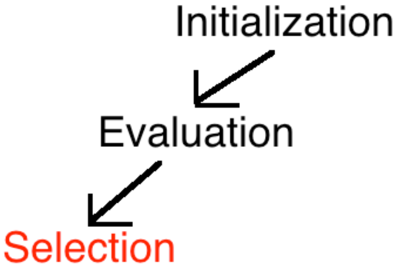
Initialization

Step 2: Evaluation

- Assess how 'good' each individual in the population is.

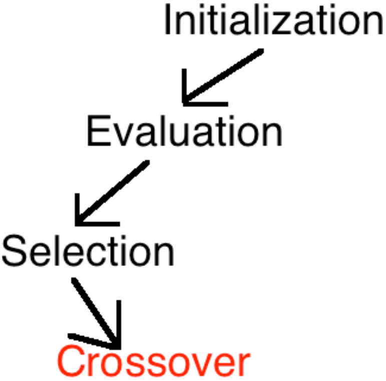
Step 3: Selection

- Choose pairs out of population to breed and create an offspring



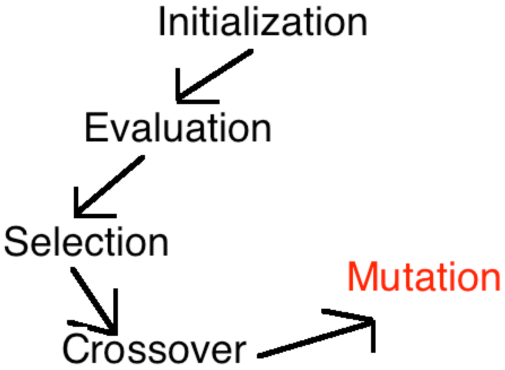
Step 4: Crossover

- Combine genes from parents to produce a viable offspring



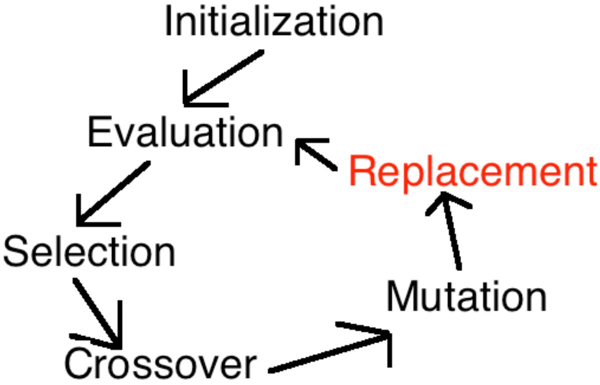
Step 5: Mutation

- Give some new genetic material to offspring



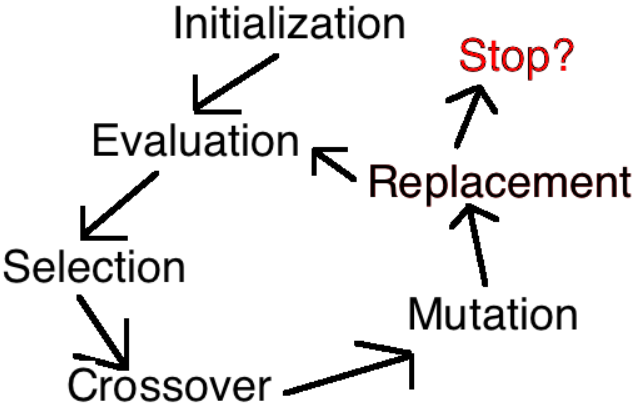
Step 6: Replacement

- When appropriate, replace parent with offspring for next generation



Step 7: Stopping Conditions

- Stop after a certain amount of time/number of generations/result is achieved...



Problems with Serial GA?

- Xiong's GA converges very quickly to local optima (not global)

Problems with Serial GA?

- Xiong's GA converges very quickly to local optima (not global)
- Modify selection, crossover, mutation, replacement to increase diversity

GA Improvement 1: Coin toss?

- Currently - when deciding which genes to give to offspring, follow pre-determined order

GA Improvement 1: Coin toss?

- Currently - when deciding which genes to give to offspring, follow pre-determined order
- Xiong 2006: Modified Genetic Algorithm (MGA)- do neighborhood search in crossover operator

GA Improvement 1: Coin toss?

- Currently - when deciding which genes to give to offspring, follow pre-determined order
- Xiong 2006: Modified Genetic Algorithm (MGA)- do neighborhood search in crossover operator
- Make crossover/mutation stochastic. Mix up ordering

GA Improvement 2: Keep equal?

- Currently - replace offspring with parent if offspring better

GA Improvement 2: Keep equal?

- Currently - replace offspring with parent if offspring better
- Replace parent with offspring if offpring better or equal

GA Improvement 3: Favor mutation?

- Currently - in mutation introduce a random gene, treat as any other

GA Improvement 3: Favor mutation?

- Currently - in mutation introduce a random gene, treat as any other
- Treat mutation gene specially (make inclusion more likely)

Test instances

- Tested against 36 sets of instances (10 instances per set) from Cerulli et al. [2005]

Test instances

- Tested against 36 sets of instances (10 instances per set) from Cerulli et al. [2005]
- Compared with global optimum when known, best known solution (BKS) in literature otherwise

Serial GA changes - results

- GA + variants:

Algorithm	% Above BKS	% Above Optimum (if known)	Time (s)
Original GA	6.97	3.79	129.3
Crossover Coin toss	4.7	2.3	139.8
Mutation Coin Toss	5.1	3.0	130.5
Keep Equal	8.3	5.1	123.1
Favor Mutation	5.1	2.4	130.11
Everything	3.9	2.4	139.7

Serial GA changes - results

- GA + variants:

Algorithm	% Above BKS	% Above Optimum (if known)	Time (s)
Original GA	6.97	3.79	129.3
Crossover Coin toss	4.7	2.3	139.8
Mutation Coin Toss	5.1	3.0	130.5
Keep Equal	8.3	5.1	123.1
Favor Mutation	5.1	2.4	130.11
Everything	3.9	2.4	139.7

- Xiong's Modified GA (MGA) + variants

Algorithm	% Above BKS	% Above Optimum (if known)	Time (s)
MGA	4.1	2.2	474.8
MGA with Everything	2.9	1.5	421.1

Dividing up the Population

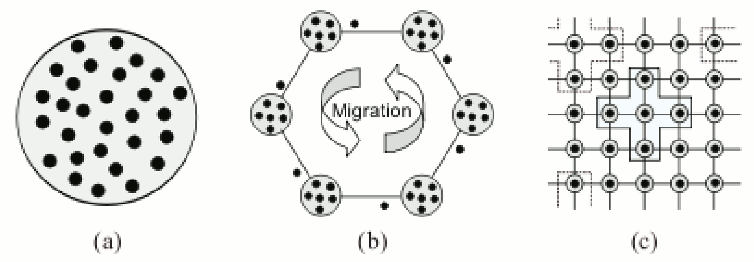


Figure: Three different types of GAs showing interaction between individuals (black dots) in the population. a)Panmictic b) Distributed c) Cellular [Alba:2008]

Distributed GA - results

Genetic Algorithm for the MLST

Oliver Rourke

The MLST

Genetic Algorithms

Serial GA+

Parallel GA

Future

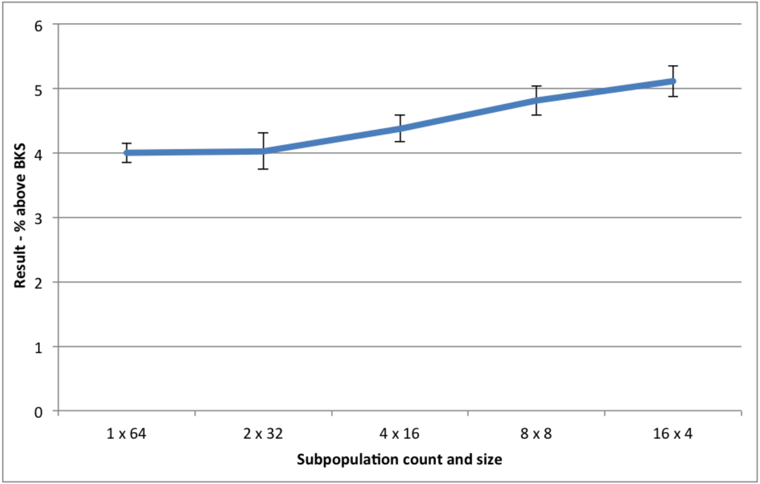


Figure: % above BKS for a variety of island sizes - no migration

Serial Genetic Algorithm $->$ Parallel Genetic Algorithm

- Why?
  - Speedup
  - Larger Problems

Serial Genetic Algorithm $->$ Parallel Genetic Algorithm

- Why?
  - Speedup
  - Larger Problems
- Allocate different subpopulations to different processors

Distributed GA - results

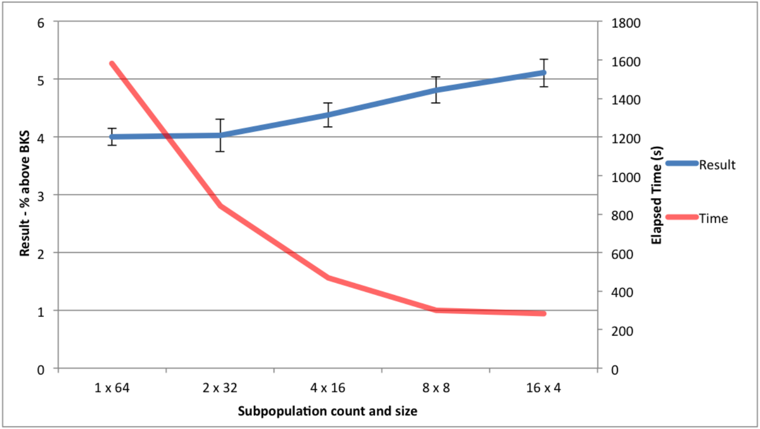


Figure: % above BKS and computational time for a variety of island sizes - no migration (run on 2.2GHz quad-core Intel Core i7)

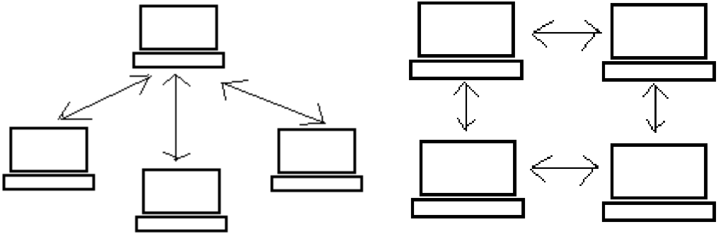
Serial Algorithm $->$ Parallel Algorithm

- Why?
  - Speedup
  - Larger Problems
- Allocate different subpopulations to different processors
- Communication between subpopulations - better results?

Serial Algorithm $->$ Parallel Algorithm

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

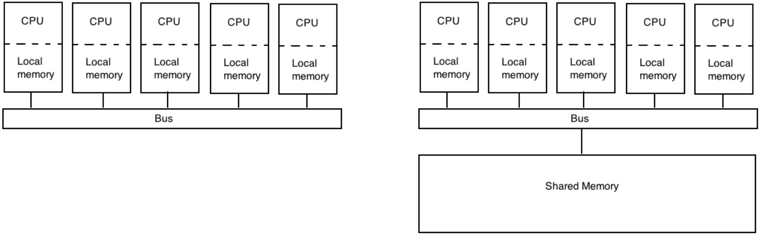
Parallel Structures (Who?)



(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

MPI versus Pthreads

- Initially I planned on using MPI, switched to pthreads

MPI versus Pthreads

- Initially I planned on using MPI, switched to pthreads
- MPI advantages:
  - Simpler to code (less chance of race conditions/data corruption)
  - Can run on wider variety of grids (no need for shared memory)

MPI versus Pthreads

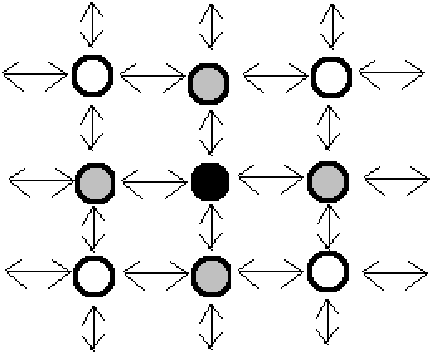
- Initially I planned on using MPI, switched to pthreads
- MPI advantages:
  - Simpler to code (less chance of race conditions/data corruption)
  - Can run on wider variety of grids (no need for shared memory)
- Pthreads advantages:
  - Less overhead
  - GENOME cluster has shared memory
  - More flexible - share objects, access shared memory whenever

MPI versus Pthreads

- Initially I planned on using MPI, switched to pthreads
- MPI advantages:
  - Simpler to code (less chance of race conditions/data corruption)
  - Can run on wider variety of grids (no need for shared memory)
- Pthreads advantages:
  - Less overhead
  - GENOME cluster has shared memory
  - More flexible - share objects, access shared memory whenever
- Pthreads allows asynchronous code (not simultaneous), more flexible communication BUT must deal with mutexes, condition variables etc.

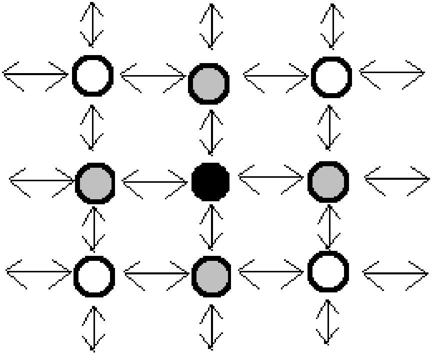
Communication scheme 1

- Scharrenbroich: 'CGA-inspired' distributed GA
- Migration between 'neighboring' subpopulations if no improvement for 10 generations, replacing weakest individual in subpopulation



Communication scheme 1

- Scharrenbroich: 'CGA-inspired' distributed GA
- Migration between 'neighboring' subpopulations if no improvement for 10 generations, replacing weakest individual in subpopulation
- Tested for 2, 4, 8 islands - no significant difference in results



Communication Scheme 2

- Best solution from each subpopulation is copied to shared memory
- Individuals in subpopulations occasionally breed with any individuals saved in shared memory (although individuals in shared memory do not directly replace individuals in subpopulations)

Communication Scheme 2

- Best solution from each subpopulation is copied to shared memory
- Individuals in subpopulations occasionally breed with any individuals saved in shared memory (although individuals in shared memory do not directly replace individuals in subpopulations)
- Results:
  - 2 islands - no significant difference in results
  - 4, 8 islands - minor improvement

Plans: Spring 2012

- Investigate more communication/migration schemes
- Test/run on GENOME cluster at UMD

Plans: Spring 2012

- Investigate more communication/migration schemes
- Test/run on GENOME cluster at UMD
- Investigate other GAs (canonical CGA?) which are suitable for parallelization
- Final presentation and report

Summary to date

- Significant improvement on serial code (code running on each individual processor)

Summary to date

- Significant improvement on serial code (code running on each individual processor)
- Can run in parallel, significant speed up achieved

Summary to date

- Significant improvement on serial code (code running on each individual processor)
- Can run in parallel, significant speed up achieved
- Shared memory allows for effective inter-processor communication (details to be finalized...)

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
- Cerulli, R. et al., Metaheuristics comparison for the minimum labelling spanning tree problem in The Next Wave in Computing, Optimization, and Decision Technologies, vol. 29 of Operations Research/Computer Science Interfaces Series, NY, NY, USA, 2005
- Consoli, S. et al., Greedy Randomized Adaptive Search and Variable Neighbourhood Search for the minimum labelling spanning tree problem, European Journal of Operational Research, vol. 196 pp. 440-449, 2009
- 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

Bibliography (cont.)

- 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.
- 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