**Conventional GA vs Island Model Parallel GA for TSP**

Jacob Hopkins  
 Computer Science  
St Cloud State University  
 St Cloud MN USA  
jrhopkins@stcloudstate.edu

ABSTRACT

A CPU based parallel genetic algorithm was proposed, which takes the conventional genetic algorithm and parallelizes it with a traditional coarse-grained architecture called island model found in a 2020 paper released by Hailong Zhang and their team. [1] The idea is to simulate different environments/islands for species richness. This is done by splitting the larger population into smaller subpopulations which on their own perform selection, cross-over, mutation, and now, as a group of sub-populations, a newly introduced genetic operation: migration: where which a specified number of high fitness individuals are cyclically moved between subpopulations. Using modern CPU multicore and multi-threaded capabilities these subpopulations where run simultaneously across threads to improve computing efficiency. It provided a new solution to the TSP problem.

INTRODUCTION

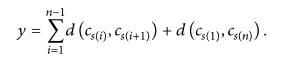
Traveling Salesmen Problem (TSP) is a historical problem in computer science, one closely tied to the success of genetic algorithms in their capabilities to find good solutions in relatively reasonable time in with NP hard problems like traditional TSP. The formal description is as follows. [1] Given a set of cities {C1, C2, ..., Cn }, with the Euclidean distance between these cities denoted d{ Ci, Cj } this problem involves searching for the sequence *s* that minimizes the overall route distance *y* defined below, Fig 1.

Figure 1  
Genetic Algorithms (GA) are heuristic algorithms based on evolution of chromosomes through selection-based processes in which higher fitness individuals in populations have a higher chance of reproducing in the iterative process of various genetic operations. Populations are generated initially through random encodings, and each generation includes evaluating individual’s fitness and selecting individuals based on their fitness and then editing them with operations that simulate mutation and crossover to form new generations. Terminating when the specified number of generations is reached.

ENCODING

Populations consist of chromosomes, which are encoding of candidate solutions. For TSP candidate solutions or tours include sequences of cities. Cities are represented as points in Euclidean plane. Sequences of cities means lists of these points. Suppose we had three cities:  
*City A (3, 4) City B (5, 3) City C (1, 2)*  
A candidate solution for TSP, a chromosome, would be a list of those cities { (1,2), (3,4), (5,3) }, meaning start at City C, visit City A, visit City B, and to finish the tour return to city C, but that is encoded by the first and last city being represented at the first and last index of the sequence.

FITNESS

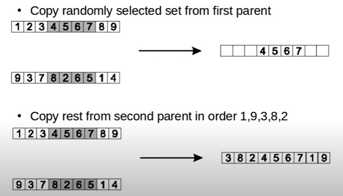
The fitness of the candidate solutions is evaluated as the inverse of the Euclidean distance of the tour. This provides a smoother landscape for the GA to traverse, and makes the search problem a maximization of fitness, which results in shorter distance tours/candidates. Suppose we evaluated the distance of the tour specified with the respective cities above: { (1,2), (3,4), (5,3) }. The distance between (1,2) and (3,4) is 2√2. The distance between (3,4) and (5,3) is √5. The distance between (5,3) and (1,2) is √17. The sum of those distances is 9.18760... Giving a fitness of 0.00001088423… by taking the inverse.

SELECTION

Selection is done proportionate to fitness, sometimes called roulette wheel selection, with elitism. Elitism means those individuals with the highest fitness are preserved for the next generation. Proportionate selection means that the selection of a chromosome is based on its fitness. This was implemented through the cumulative sum of the finesses within the population to the total sum of all fitness’.

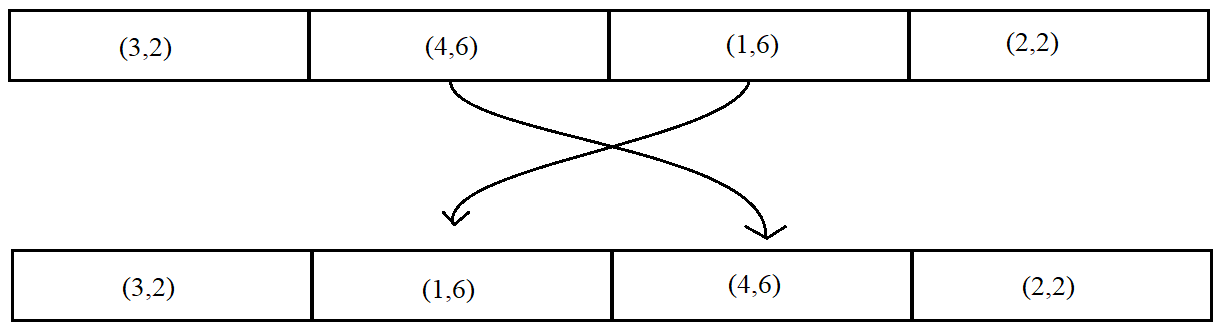
ORDERED CROSSOVER

Crossover is an operation that resembles the exchange of genetic information of between chromosomes. Ordered crossover involves taking a subsection of one parent, and placing those genes in the child, and then adding in those genes not in the child that are still in the other parent in order, more is seen below in Fig 2.

Figure 2

SWAP MUTATION

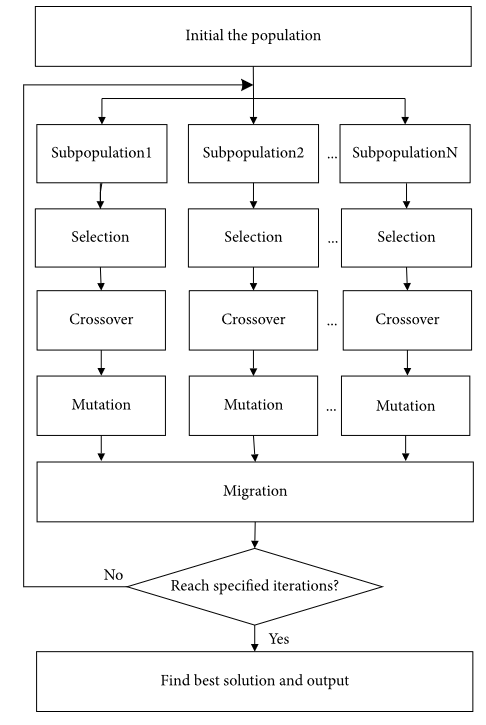
A chromosome has each gene looked at and a specified mutation rate is tested to a uniformly random selected value, both between 0 inclusive and 1 exclusive. If that gene is then ‘selected’ for mutation it is then swapped to a new randomly selected location in the chromosome, with the randomly selected index taking the prior. See fig 3.

  
Figure 3

CONVENTIONAL GA STRUCTURE

This algorithm comes with the parameter of number of generations. After initializing the initial population, the algorithm runs the number of generations given. Then returns the best solution found. Each generation includes the evaluation of fitness, selection, crossover, mutation, and then a return.

PARRALLEL GA STRUCTURES  
  
Single population and multi-population and Global breeding and local breeding are the different groupings of the parallel genetic algorithms. The proposed GA uses the island model, shown in figures 4 and 5. This rises to 3 main types of parallel GA’s described as such and shown in Figure 5:  
Master-Slave (a) – Single population, global mating, and fitness evaluation is distributed. Strong for collective computing devices. Mating is free to be done with anyone in the population.   
Island Model (b) - Multi population, local mating, and fitness is also evaluated locally. Mating and mutation are local, but a new operator migration is introduced to spread genetic material. See Figure 4 for details of its structure.

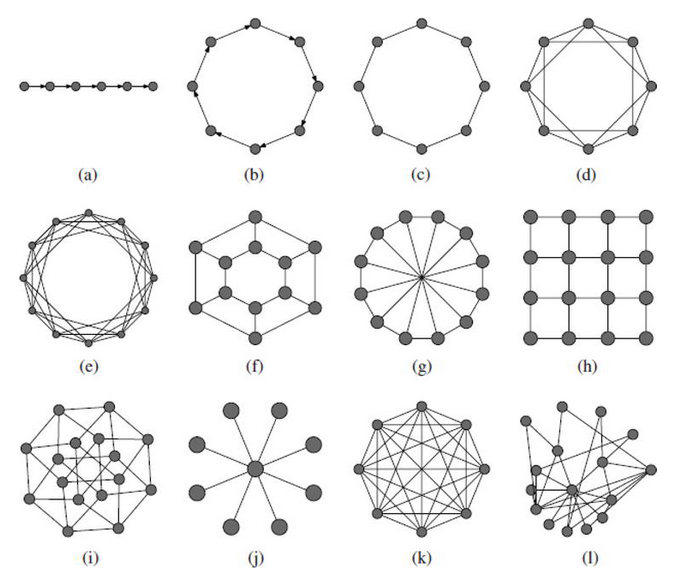
Figure 4

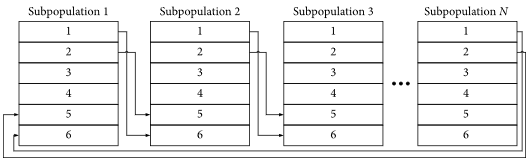
Cellular Model (c) – Detailed by Cheng and Gen as a spatially structured population, where selection and mating are restricted to small neighborhoods, but neighborhoods overlap, permitting some interaction among all the individuals. This model is suited for massively parallel computers.’ [2]

A picture containing strainer, game

Description automatically generatedFigure 5

MIGRATION

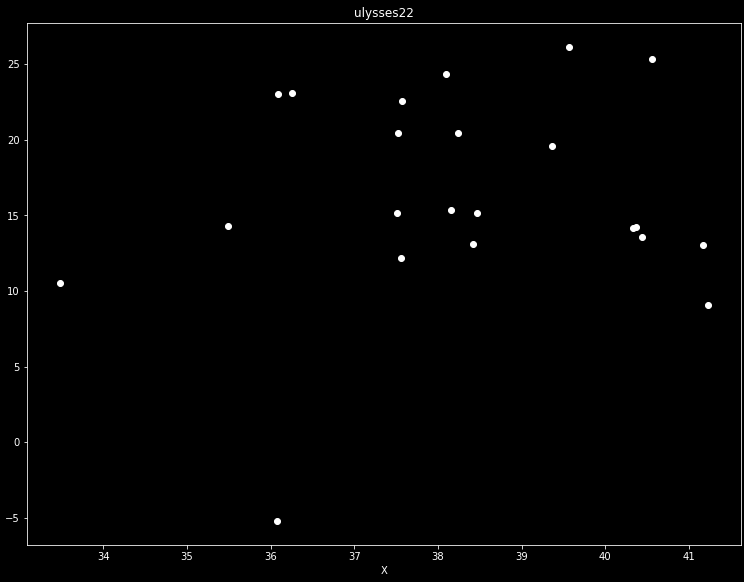
Migration involves rotating individuals amongst the subpopulations. This can be done in many ways, see below Figure 6 for some of the many topologies. Topology B would be the one implemented in the proposed GA, shown in Figure 7. It was also implemented to involve the transfer of the high fitness individuals amongst the sub populations. This migration is done via a migration rate, by which it is found to work best with a moderate rate of transfer.   
Figure 6

Figure 7

TSPLIB DATASETS

The TSPLIB Symmetric Traveling Salesman Problem Instances. This is a compendium of many various sized datasets for TSP problem practice, many of which have the optimal tour with the dataset. Among the used datasets the ones described will be such: Ulysses 22

ULYSSES 22



Created by Groetschel and Padberg it is designed to resemble the tour taken by Ulysses or Odysseus in the Iliad. It contains 22 data points.

TESTING PROCEDURE

Testing of the specified genetic algorithms were done through python 3 with the anaconda tool kit. This limited the potential testing due to the performance hit of the language compared to hard typed languages. Tests were done with small, medium, and large population sizes. With the generation count slightly reducing when testing larger population sizes. These tests were run for each dataset used.

SMALLER POPULATION PARAMETERS

The conventional GA and parallel GA was run with a mutation rate of 1% always. The conventional GA first ran with a population size of 50 and elite count of 20 across 500 generations. Respectively the parallel GA was run with a population size of 50, and two sub populations. Each with 8 elites. Migration was ran moving 2 people 15% of the time. Also, across 500 generations.

RESULTS ULYSSES22

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Length | Time | Length | | Time |
| Tour | CGA | CGA | PGA | | PGA |
| 1 | 77.146 | 5.216 | 75.535 | | 5.576 |
| 2 | 75.309 | 5.188 | 76.371 | | 5.477 |
| 3 | 77.136 | 5.107 | 76.372 | | 5.556 |
| 4 | 75.88 | 5.101 | 77.824 | | 5.521 |
| 5 | 77.806 | 5.102 | 77.186 | | 5.497 |
| 6 | 80.312 | 5.061 | 76.856 | | 5.538 |
| 7 | 76.314 | 5.138 | 76.922 | | 5.528 |
| 8 | 77.814 | 5.127 | 78.395 | | 5.52 |
| 9 | 76.378 | 5.009 | 77.58 | | 5.495 |
| 10 | 77.386 | 5.015 | 76.4 | | 5.3777 |
|  |  |  |  | |  |
| Length | Average | Best | Worst |
| CGA | 77.1481 | 75.309 | 80.312 |
| PGA | 76.9441 | 75.535 | 78.395 |
|  |  |  |  |
| Time | Average | Best | Worst |
| CGA | 5.1064 | 5.009 | 5.216 |
| PGA | 5.50857 | 5.3777 | 5.576 |

LARGER POPULATION PARAMETERS

Population count has been moved to 120 with elite count at 20 for the conventional GA. The generation count is also reduced to 350. For the parallel GA, this time were using 3 sub populations, a migration rate of 15% and a migration count of 2.

RESULTS ULYSSES22

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | CGA | | CGA | | | PGA | | PGA |
| Tour | Length | | Time | | | Length | | Time |
| 1 | 78.63 | | 15.697 | | | 75.968 | | 8.82 |
| 2 | 75.309 | | 15.65 | | | 76.841 | | 8.832 |
| 3 | 76.161 | | 15.612 | | | 75.794 | | 8.819 |
| 4 | 75.934 | | 15.667 | | | 76.694 | | 8.851 |
| 5 | 77.073 | | 15.655 | | | 76.007 | | 8.802 |
| 6 | 76.993 | | 15.6044 | | | 76.664 | | 8.761 |
| 7 | 76.102 | | 15.657 | | | 76.022 | | 8.849 |
| 8 | 76.102 | | 15.657 | | | 76.094 | | 9.126 |
| 9 | 76.811 | | 15.713 | | | 75.309 | | 8.974 |
| 10 | 76.278 | | 15.723 | | | 75.894 | | 8.957 |
|  |  | |  | | |  | |  |
| Length | | Average | | Best | Worst | |
| CGA | | 76.5393 | | 75.309 | 78.63 | |
| PGA | | 76.1287 | | 75.309 | 76.841 | |
|  | |  | |  |  | |
| Time | | Average | | Best | Worst | |
| CGA | | 15.66354 | | 15.6044 | 15.723 | |
| PGA | | 8.8791 | | 8.761 | 9.126 | |

LARGEST POPULATION PARAMETERS

Population count has been moved to 200 with elite count at 50 for the conventional GA. The generation count is also reduced to 250. For the parallel GA, this time were using 4 sub populations, a migration rate of 15% and a migration count of 2.

RESULTS ULYSSES22

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | CGA | CGA | PGA | PGA |
| Tour | Length | Time | Length | Time |
| 1 | 76.775 | 24.588 | 77.096 | 10.595 |
| 2 | 76.585 | 24.862 | 77.061 | 10.834 |
| 3 | 76.881 | 25.625 | 75.597 | 10.843 |
| 4 | 76.087 | 25.322 | 77.15 | 10.815 |
| 5 | 76.016 | 25.244 | 76.161 | 10.774 |
| 6 | 76.503 | 25.265 | 77.899 | 10.81 |
| 7 | 77.77 | 25.458 | 76.191 | 10.879 |
| 8 | 75.88 | 25.41 | 76.627 | 10.822 |
| 9 | 76.154 | 25.109 | 76.519 | 10.927 |
| 10 | 77.481 | 25.172 | 76.605 | 10.834 |

|  |  |  |  |
| --- | --- | --- | --- |
| Length | Average | Best | Worst |
| CGA | 76.6132 | 75.88 | 77.77 |
| PGA | 76.6906 | 75.597 | 77.899 |

|  |  |  |  |
| --- | --- | --- | --- |
| Time | Average | Best | Worst |
| CGA | 25.2055 | 24.588 | 25.625 |
| PGA | 10.8133 | 10.595 | 10.927 |

REVISIONS

GPU architecture is much more powerful for parallel workloads. The next step after this would be to convert this algorithm to take advantage of cuda coding in order to gain more performance. Migration also has many different implementations, many of which are more unexplored. Parallel GA systems also give rise to meta-GA’s and the capacity to run many different parameters at once. This is also unexplored. There is also the wonder of clustering these cities and feeding in those cities into the sub populations in order to seed with a heuristic.

CONCLUSIONS

As generation increases this GA seems to scale similar, however when the population is larger, and there are many subpopulations this algorithm can achieve similar work/performance significantly quicker.

REFERENCES

[1] “Multi-population Genetic Algorithm Based on GPU for Solving TSP Problem” By Boqun Wang, Hailong Zhang, Jun Nie, Jie Wang, Xinchen Ye, Toktonur Ergesh, Meng Zhang, Jia Li, and Wanqiong Wang. Purushothaman Damodaran 29th, Feb 2020 <https://www.hindawi.com/journals/mpe/2020/1398595/>

[2] “Parallel Genetic Algorithms with GPU Computing” By John Runwei Cheng and Mitsuo Gen, May 20th, 2019

<https://www.intechopen.com/books/industry-4-0-impact-on-intelligent-logistics-and-manufacturing/parallel-genetic-algorithms-with-gpu-computing>