CS 651: Island Genetic Algorithm in the context of Vehicle Routing Problems

SONAL BIHANI, PEDRO FERRAZ, and CALLUM TAKASAKA

Authors' address: Sonal Bihani; Pedro Ferraz; Callum Takasaka.

In this project, we implement an approach for minimizing the maximum path distance within the Vehicle Routing Problem (VRP). Our approach is based on the island model, a type of parallel Genetic Algorithm (GA) that divides the population of solutions into separate islands that evolve independently. We conduct experiments with several VRP instances from both benchmark and synthetic datasets to evaluate the performance of our approach. Our results show that our island model GA is able to consistently find high-quality solutions to the VRP with a monotonic decrease in path cost, converging to some minimum. This makes it a promising approach for solving the VRP in real-world applications.

Additional Key Words and Phrases: Vehicle Routing Problems, Genetic Algorithms, Parallel Processing, Metaheuristics

ACM Reference Format:
Sonal Bihani, Pedro Ferraz, and Callum Takasaka. 2022. CS 651: Island Genetic Algorithm in the context of Vehicle Routing Problems. 1, 1 (September 2022), 6 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The Vehicle Routing Problem (VRP) is a complex optimization problem that involves finding the shortest or most cost-effective routes for a fleet of vehicles to visit a set of destinations while meeting various constraints such as vehicle capacities and time windows. The routes are designed such that they start and end at the same depot, ensuring that each customer is visited exactly once by one of the vehicles.

The VRP is an NP-hard problem with many possible solutions, making it difficult to solve using traditional optimization techniques. Using metaheuristics such as Genetic Algorithms, it is possible to quickly find solutions that are reasonably close to optimality. GAs are based on the principles of natural selection and genetics, and use a population of candidate solutions that evolve over time through a process of selection, crossover, and mutation. This allows them to find near-optimal solutions to complex optimization problems in a relatively short amount of time.

Additionally, parallel genetic algorithms (PGAs), with their ability to search a large space of potential solutions in parallel, can help find good solutions more quickly and efficiently than traditional sequential GAs. PGAs are flexible and can easily handle complex constraints and objectives, such as vehicle capacity and time windows, that are often present in the VRP. This can make it possible to find good solutions to large, complex VRP instances in a reasonable amount of time. We have implemented the Island Model approach where a population is partitioned and assigned to multiple islands. Each island contains a subpopulation of candidate solutions, and independent GAs are run on each island in parallel. Occasionally, individuals are migrated between islands, allowing the populations to evolve and improve. In theory, islands can evolve sequentially, but running GAs in parallel on each island improves computational efficiency.

2 METHODOLOGY

To apply the GA methodology to the VRP, we encoded the problem as a string of genes that represents a potential solution to the problem.

2.1 Encoding

Each chromosome (possible solution) is divided into two parts - the number of customers each vehicle will be serving and the order of customers followed by every vehicle [1]. Figure 1 illustrates a simple example with 2 vehicles and 12 customers.

Once the problem is encoded, an initial population of solutions is generated by randomly generating a set of strings of genes (chromosomes) that represent potential routes for the vehicles.

2.2 Fitness Evaluation

It's a method of assessing the quality of a potential solution which is typically determined by how well it satisfies a set of constraints and how well it optimizes a set of objectives. In the context of a VRP with a limited number of available vehicles and no capacity constraints, an objective that may be considered is to minimize the longest route among the vehicles. This can be achieved through a min-max approach, where the fitness value of a solution is calculated as the negative of the maximum path cost among the vehicles. Consequently, maximizing the fitness value of a solution corresponds to minimizing the maximum path cost among the vehicles.

2.3 Selection

It is the process of choosing which individual solutions will be used to produce the next generation of solutions. This is typically done by evaluating each individual's fitness, which measures how well it solves the problem at hand. The fittest individuals are then selected to serve as parents and produce offspring, which will inherit some of the characteristics of their parents. *Tournament selection* is one effective method of selection, where a small sample of individuals are selected at random from the population, and the fittest individual from this group is chosen as a parent. This can help prevent the genetic algorithm from getting stuck in local optima, where the population becomes too similar and is unable to make further progress. This is because tournament selection introduces some randomness into the selection process, and can allow less fit individuals to be chosen as parents if they are part of a particularly fit subgroup.

2.4 Crossover

In genetic algorithms, a crossover is a genetic operator used to combine the genetic material of two parents to produce a child. The child will inherit some of the characteristics of each parent, and the resulting genetic diversity can help the population evolve and adapt to changing environments. We have used two different types of crossover operators. The uniform crossover operator is applied to the first part of the chromosomes, while the order crossover operator is applied to the second part (Figure 2). A crossover probability is used to determine which chromosomes are selected for crossover. This approach allows for more flexibility in the genetic operations and can improve the performance of the GA on the VRP by allowing it to explore a wider range of potential solutions.

2.5 Mutation

Mutation is used in genetic algorithms to introduce new genetic material into the population. This can help maintain genetic diversity and can prevent the population from becoming too similar and getting stuck in local optima. It also breaks up clusters of similar individuals and allows the population to explore new solutions and adapt to changing environments. A gene's mutation probability decides whether it gets mutated or not.

For the first part of each chromosome, with the vehicle-specific encoding, we've used uniform mutation. This operator replaces the value of the chosen gene with a uniform random value selected between the upper and lower bounds for that gene. After performing the mutation operation, the gene must be repaired in order to correctly represent a partition of the vehicles. This is performed via incrementing or decrementing the values of random vehicle genes until the sum of the vehicle-part genes is equal to the total number of orders.

As for the latter part of the chromosome, representing the routes taken the vehicles, we've used random access swaps and adjacent swaps. Random access swap operators select two random genes and swap their values, while the adjacent swap operator selects one gene and swaps its value with one of its adjacent genes.

Other mutation operators that have not been implemented but could provide further variety to the population are the inversion and insertion operators. The inversion operation switches the order of the genes between the two random positions, and for the insertion operator - two random positions are created and genes from one of the positions is inserted with the other. The genes in between these two positions get shifted. Figure 3 below illustrates all mentioned mutation operators.

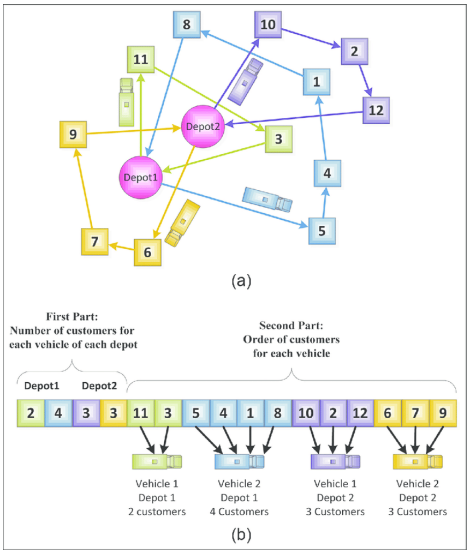

Fig. 1. Encoding Representation

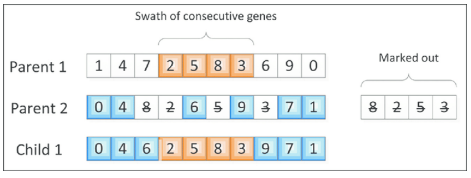

Fig. 2. Order Crossover

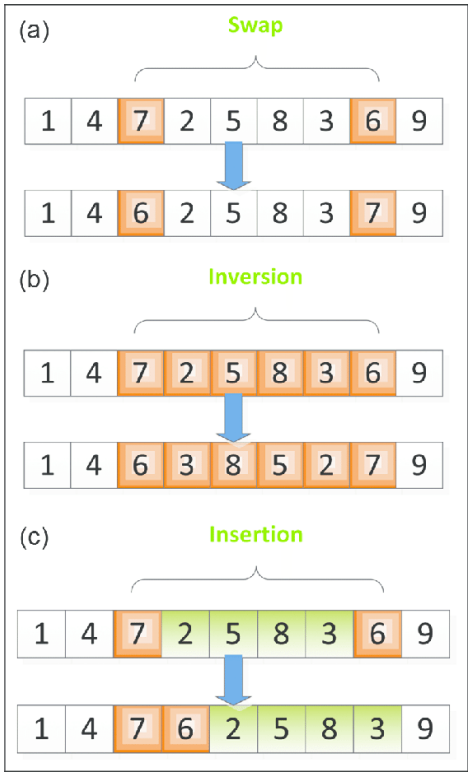

Fig. 3. Mutation Operators

3 DISTRIBUTED ISLAND-MODEL

Genetic algorithms can be distributed in such a way that multiple populations are run in isolated parallel, with individuals periodically "migrating" between populations. These distinct populations can be run in various ways, either distributed

within a single machine, or, in our case, distributed across various machines, providing a way to scale horizontally. Genetic algorithms can tend towards local optima as a result of genetic drift [2], which is essentially the change in the frequency of some pattern within a solution, due to random chance. This is undesirable when global optima are desired, and the increased diversity afforded by the island model can help to mitigate this issue. Following a number of generations within each distinct population, all workers will return some portion of their population back to the driving machine, who will then randomly replace a small amount of each population with the fittest individuals from two other populations. In this way, each population will be enriched with fit individuals containing gene patterns that may no longer be likely to occur in the original population due to other dominating patterns already being more frequent.

This process of evolution over several generations, and then migration, can be repeated at an arbitrary frequency, to improve diversity across the entire set of populations. Additional parameters are introduced in the island model, from generations between migration, to number of individuals who migrate between populations. The number of islands can also be changed according to hardware limitations. It is suggested by [3] that these parameters can be set in a semi-random fashion, and perform similarly to tuned genetic algorithms not benefiting from distributed islands.

3.1 Island-Model Implementation

Our island model takes advantage of the Spark engine, to run multiple populations in parallel, and distribute large numbers of individuals between them. We run a classical GA for 50 generations, before beginning the process of migration. Following the first 50 generations, and every 50 generations after that, we trigger a migration event. This migration event involves each island returning a portion of its population to the driver, who then replaces the bottom 20% of each population with 10% of the most fit individuals from 2 other islands at random. These percentages, and of course the number of islands to migrate, can be tuned, however this level of granularity requires some amount of experience within the field of the problem and genetic algorithms themselves.

Following each migration event, these modified populations are then sent back to their respective islands to repopulate and continue evolving. Re-population involves a modified breeding phase, where a population larger than the initial population, is generated. This is achieved by maintaining the initial population as elites, and randomly drawing upon them for parents until the required amount of children are produced to create the desired size.

We decided upon 5 migration events, with a final evolutionary period of 500 generations to ensure that competing individuals from the final migration have time to share genes, and "burn-in".

4 EVALUATION

In order to evaluate the algorithm, we have used both a benchmark dataset and synthetic instances. For the dataset, we've adapted the instances for the "multi-vehicle pickup-and-delivery problem with split deliveries (MPDPSL)", available at https://w1.cirrelt.ca/~vidalt/en/VRP-resources.html.

As for the synthetic instances, we have generated two types of instances: organized in clusters and uniformly spread. The clustered instances were generated using a Gaussian Mixture Model (GMM), while the uniform instances were generated by sampling points uniformly in $[0, L]^2$, with $L \in \mathbb{R}$.

Each of the following figures demonstrates the best routing achieved by the algorithm and the curves of maximum path cost for the best solution in each iteration. Note that each curve corresponds to the results of one island.

Figure 4 and Figure 5 represent the results on two adapted instances from MPDPSL. As we can see, the algorithm effectively improves upon the path cost and finds very reasonable routes.

In addition, note that the when presented with clustered data, the algorithm quickly finds a solution that adapts to this structure and dispatches one vehicle per cluster with no travels in between clusters. This behavior can bee seen in the sythetic GMM instance in Figure 6.

Also note that when presented with non-clustered data, the algorithm may find a solution that is better than what a "cluster-first route-second" solution might find, due to having more flexibility in routing to minimize the maximum path. Figure 7 demonstrates this behavior on a uniformly generated instance.

The figures below involving fitness over time for all islands show some interesting trends related to migration. Given that during migration, islands will receive the most fit individuals from other populations, there is a chance that an island which is performing poorly (individuals have on average, lower fitness than other populations) will see a large increase in fitness. This is simply due to the way in which population fitness is calculated, i.e. the most fit individual in a population will define that population's fitness. A very apparent case of this phenomenon can be seen in Figure 7, at approximately generation 100. Beyond immediately improving the fitness of lesser fit populations, migration allows for diversity in a population which may otherwise have reached local convergence due to unfortunate genetic drift. Fit individuals who have diverged to other solution spaces in a different population, will provide gene sequences which may be highly improbable in the population which it migrated to.

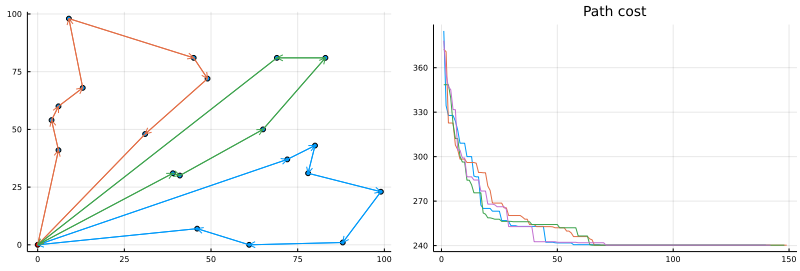

Fig. 4. Instance with 20 nodes and 3 vehicles

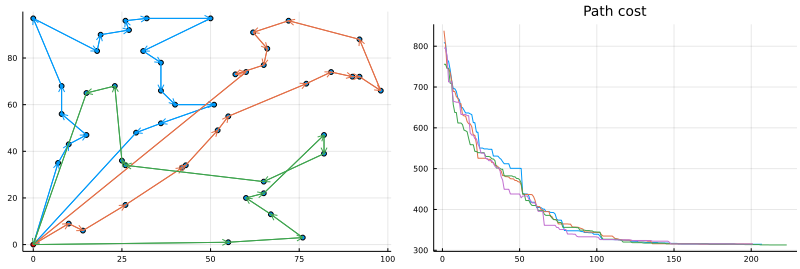

Fig. 5. Instance with 50 nodes and 3 vehicles

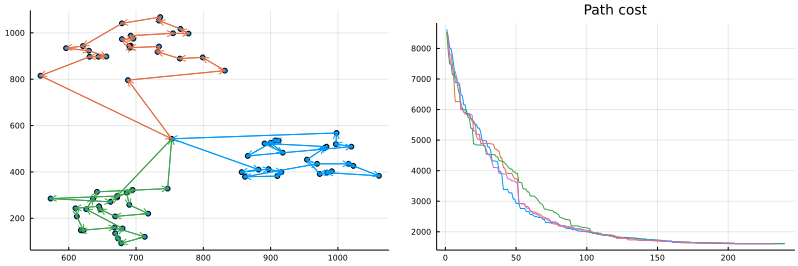

Fig. 6. Synthetic GMM instance with 75 nodes and 3 vehicles

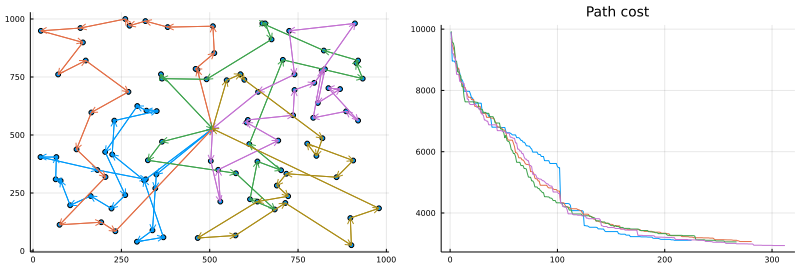

Fig. 7. Synthetic uniform instance with 100 nodes and 5 vehicles

5 ALTERNATE APPROACHES

Known as the Master-Slave approach, another way of parallelizing GAs is to simply perform the different operations in parallel on multiple workers. Distributed data processing frameworks allow a GA to be run on a cluster of machines, with each machine executing a portion of the GA in parallel. This can potentially speed up the solution process and improve the runtime taken for each generation until convergence. The population and other data structures used by the GA must be distributed across the cluster, and the GA operations, such as fitness evaluation, selection, crossover, and mutation are implemented using a distributed programming model like Spark or Hadoop.

In order to achieve even better results, we could also have combined the standard GA with VRP-specific heuristic approaches to improve and diversify solutions, leading to a Hybrid-GA model such as presented by Vidal et al. [4].

In addition, the controlled execution environment in the available cluster didn't allow the Spark program to request more workers. In an environment with more flexibility, it would be possible to instantiate even more islands with larger populations, leading to a greater diversity of solutions and, therefore, a greater likelihood of not being trapped in local optima and thus converging to better solutions.

REFERENCES

[1] Shuihua Wang, Zeyuan Lu, Ling Wei, Genlin Ji, and Jiquan Yang. Fitness-scaling adaptive genetic algorithm with local search for solving the multiple depot vehicle routing problem. *SIMULATION*, 92(7):601–616, 2016. doi: 10.1177/0037549715603481. URL https://doi.org/10.1177/0037549715603481.

[2] Alfian Gozali and Shigeru Fujimura. Localized island model genetic algorithm in population diversity preservation. In *Atlantis Highlights in Engineering (AHE), volume 2*, 11 2017. doi: 10.2991/icoiese-18.2019.22.

[3] Yiyuan Gong and Alex Fukunaga. Distributed island-model genetic algorithms using heterogeneous parameter settings. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pages 820–827, 2011. doi: 10.1109/CEC.2011.5949703.

[4] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, Nadia Lahrichi, and Walter Rei. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60:611–624, 06 2012. doi: 10.1287/opre.1120.1048.