# Virus Transmission Genetic Algorithm

Weixin Ling and Walter D. Potter

**Abstract.** In this paper we propose a novel Virus Transmission Genetic Algorithm, which is inspired by the evolution of immune defense and the infection transmission model. Containing one virus population and one host population, the VTGA simulates biological infections by using new operators such as virus infection and virus spread. To study the effectiveness, we apply the algorithm to several function optimization problems, several travelling salesman problems and a forest planning problem. Results of the experiments show that the VTGA performs well at searching for optimal solutions and preserving diversity of population.

**Keywords:** Virus Transmission Genetic Algorithm, Function Optimization, Travelling Salesman Problem, Forest Planning Problem.

## 1 Introduction

Premature convergence is a well-known problem of the Genetic Algorithm [1] [2] [3]. One solution is storing patterns during a search. Hence, diversity won't be eliminated when the algorithm converges. The Virus-Evolutionary Genetic Algorithm [4] [5] is an algorithm implementing this method. The VEGA has a virus population and a host population. The virus population is for saving effective patterns. When a new generation of hosts is created, these effective patterns offer information to make the new hosts better than their parents. Reasonable as it sounds, there are problems with the VEGA. The first one is that every virus represents a continuous chromosome region, not a complete solution. If the effective patterns don't exist as continuous regions, then the viruses can't offer much help. The second problem is how to determine the fitness of the viruses. Since the virus population has a limited size. Hence, only the most effective patterns will be stored and we need to evaluate patterns to decide which ones will stay. In the

Weixin Ling · Walter D. Potter
Institute for Artificial Intelligence, University of Georgia, United States

M. Ali et al. (Eds.): *Contemporary Challenges & Solutions in Applied AI*, SCI 489, pp. 41–46.
DOI: 10.1007/978-3-319-00651-2_6 

VEGA, the fitness value of a virus measures how much the virus improves the hosts infected by it. So, the evaluation of a virus involves many evaluations of the hosts related to the virus, which could be computationally expensive for some complex problems.

Consequently, considering the reasonableness of storing patterns and the problems of the VEGA, we propose a new algorithm called the Virus Transmission Genetic Algorithm. The VTGA uses two populations as well, one virus population and one host population. However, they work differently. The virus population is used for search and the host population is used for storing good schemes. In the VTGA, we simulate the evolution of viruses inside their hosts. Therefore, we have crossover and mutation working similarly to that of the simple GA. To mimic the behaviors of viruses, we implement infection, spread and recovery operators following the Susceptible-Infectious-Recovered (SIR) model in epidemiology [6].

In the VTGA, we involve discussion about the biological immune system. However, the VTGA is an algorithm very different from the well-known Artificial Immune System (AIS), which abstracts the structure and function of the natural immune system through pattern recognition techniques [7].

In the remainder of the paper, we provide a description of the VTGA. Then we examine the efficiency of our algorithm by using it to solve some function optimization problems, some travelling salesman problems and a 73-stand forest planning problem. Finally, we discuss the features of the algorithm and future work.

## 2 Virus Transmission Genetic Algorithm

To model the evolution of immune defenses against infectious diseases, one host population and one virus population are defined in the VTGA. Although named differently, host individuals and virus individuals both represent a complete solution. The VTGA initializes hosts and viruses by assigning random solutions to them, which are evaluated by the same objective function. The host population is mainly used for recording good solutions found during computation. No operation is defined for the host population and there is no interaction among hosts. Virus individuals, on the other hand, perform many operations, the outline of which is this: There are attacks from virus individuals to hosts. Weak hosts are killed by viruses and replaced by stronger hosts. As a result the overall fitness of the host population is gradually improved. To simulate this process, we need to implement virus infection, virus evolution (crossover and mutation) and virus spread.

For infection, a virus selects a weak host adopts information from the host. We use a tournament selection to randomly pick up a few candidate hosts. The candidate host with the worst fitness value will be selected as the target, following the fact that people with weak immune systems are likely to get sick. When solving an optimization problem, we find that it's usually harder to improve better solutions than worse ones. Hence, with the "selecting the worst scheme", we improve poor solutions first. After selection, the virus will adapt itself to the host. In biology, adaptation is a complex process. A virus replicates and mutates to generate lots of strains to beat the host's immune system. But in this study, we simplify this

process by copying several genes from the host, which could be seen as a process of bringing effective patterns from the host to the virus. The parameter INFECTION_RATE is used to determine the percentage of genes being copied.

After infection, a virus starts developing itself by crossover and mutation inside its host. Only viruses infecting the same host could be matched together for crossover, so the viruses infecting the same host form a dynamic subpopulation, which is isolated from other subpopulations of other hosts and whose size is changing because some of the viruses will leave and infect another host from time to time. For this reason, although the size of the whole virus population remains the same, the virus subpopulation size inside a host is changing all the time. In this case, it is difficult to use traditional crossover operators. Because they usually require parent selection and replacement and implementing selection and replacement on dynamic virus subpopulations in different hosts is complicated.

Considering that, we replace the traditional selection process with enumerating every pair of two virus individuals inside the same host and exchanging information between them. The crossover process is one-way. The better virus copies some random genes to replace the corresponding genes of the worse virus but not the other way around. How many positions are copied is controlled by the parameter CROSSOVER_RATE. After the weaker virus is rewritten, the newly generated virus will not be evaluated and its fitness value remains the same. The crossover is finished when every pair of viruses of the same host is processed.

We gain two advantages from this method. Inside the same host, the best virus has influence on every other virus. The second best virus has influence on every other virus except the best one and so on. On the other hand, the worst virus will be affected by all other viruses. And the second worst virus will be affected by all other viruses except the worst. That means better solutions could have an influence on more solutions and worse ones cause impacts on fewer solutions.

After crossover, the VTGA mutates viruses by reassigning random values to genes with a low probability, which is controlled by a parameter named MUTATION_RATE. When mutation is finished, all viruses will be evaluated.

In an appropriate situation, a virus escapes from its current host and selects another one. Virus spread as an operator simulates this process of escaping. After crossover and mutation, the VTGA goes through all viruses to check their fitness values. If a virus is good enough to defeat its host by having a better fitness, then there is a chance that the host's solution will be replaced completely by the solution of the virus. Whether a host will be replaced or not is controlled by a parameter called SPREAD_RATE. In another case when a virus is not better than its host, the VTGA allows the virus to escape with a probability controlled by a parameter called the RECOVERY_RATE. Once a virus escapes from its host, we would consider the host's immune system to have recovered from the infection of this virus so that the virus won't perform crossover or mutation until it infects next host.

In the VTGA, the infection operator randomly selects a few candidates from the host population, and a virus will infect the weakest one of the candidates. So the worse a host is the more opportunities it has to get infected. But there is one problem. Suppose the VTGA selects N candidates from the host population in every infection. If N is smaller than the host size, some hosts will never be selected

because they are the best in the population. So the VTGA spends all its effort on improving relatively bad solutions. This mechanism gradually raises the lower bound of all host fitnesses. However, in optimization problems, this is not efficient because the goal is to find the best solution. So we use a method called fitness masking on the best host. At the end of every generation, the VTGA finds the best individual in the population. After the VTGA records the best individual, the fitness masking operator randomly selects another fitness value from the host population to replace the fitness value of the best individual in the population.

## 3 Experiment Results

**Function Optimization.** At the first stage of experiments we have six quality tests using several famous test functions [8]. The six functions are: De Jong's function, Axis parallel hyper-ellipsoid function, Rotated hyper-ellipsoid function, Rosenbrock's valley, Rastrigin's function and Schwefel's function.

**Travelling Salesman Problems.** The travelling salesman problem (TSP) is a well-known NP-hard problem in combinatorial optimization. In this study, we test the VTGA with several TSP problems (Bays29, Swiss42, Berlin52, Eil76 and Ch150) obtained from the TSPLIB[1]. We describe a solution of the TSP problems as a permutation of integers. And we use the order based crossover operator proposed by Syswerda [9] [10] and the exchange mutation operator.

**The Forest Planning Problem.** The goal of solving a forest planning problem [11] is to find the best valid harvest schedule maximizing the even-flow of harvest volume. We use the VTGA to solve the 73-stand Daniel Pickett Forest[2] [11].

The forest planning problem is a constrained optimization problem. A valid solution of the problem has to obey all rules listed in the problem. To acquire valid solutions using a search algorithm, we repair invalid solutions after mutation so that they become violation-free before they are evaluated.

**Results.** We run 50 trials for every function optimization problem. From the results (Table 1), we learn that the VTGA is able to obtain good results in most tests.

We use different configurations when solving the TSP problems (Table2). From the results (Table 2), we believe the VTGA is an effective algorithm for solving the TSP problems generally.

For comparison, we implement one generational GA (Table 3) with the same repair function. We find that after 1 million evaluations, it is difficult for the GA to improve the solutions it finds. The best solution found by the GA remains the same for many generations before any improvement. But the VTGA is able to improve its best solution more or less after every 1,000,000 evaluations. It outperforms the GA on the average fitness and the best fitness. And the VTGA achieves 69.2301% accuracy of finding the global optimum in 30 trials.

---

[1] `http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`

[2] Data could be found at: `http://www.warnell.forestry.uga.edu/Warnell/Bettinger/planning/index.htm`

**Table 1** Results for the Function Optimization Problems[3]

| Function | N | Global Best | 20000 Evaluations | | 100000 Evaluations | |
|---|---|---|---|---|---|---|
| | | | Best | Average | Best | Average |
| De Jong's function | 5 | 0 | 3.52E-17 | 1.6E-05 | 0 | 2.8E-22 |
| Axis parallel hyper-ellipsoid | 5 | 0 | 1.90E-18 | 9.8E-06 | 0 | 1.2E-22 |
| Rotated hyper-ellipsoid | 5 | 0 | 6.45E-14 | 0.00167 | 0 | 1.1E-20 |
| Rosenbrock's valley | 5 | 0 | 0.13 | 2.66363 | 0.12 | 2.54 |
| Rastrigin's function | 5 | 0 | 0 | 3.21E-07 | 0 | 0 |
| Schwefel's function | 5 | -2094.91 | -1787.40 | -1451.82 | -2094.71 | -2010.84 |

**Table 2** Results for the Travelling Salesman Problems

| TSP Instance | HS | VS | CR | MR | IR | SR | RR | Global Optimum | Best | Average Fitness | STD | Avg. Number of Evaluations (Million) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bays29 | 200 | 1000 | 0.3 | 0.01 | 0.7 | 1.0 | 0.8 | 2020 | 2020 | 2020.2 | 1.08 | 0.34 |
| Swiss42 | 200 | 1000 | 0.3 | 0.01 | 0.7 | 1.0 | 0.8 | 1273 | 1273 | 1274.7 | 5.92 | 2.5 |
| Berlin52 | 200 | 1000 | 0.3 | 0.01 | 0.7 | 1.0 | 0.8 | 7542 | 7542 | 7570.3 | 56.89 | 7.1 |
| Eil76 | 200 | 1000 | 0.3 | 0.01 | 0.9 | 1.0 | 0.5 | 538 | 538 | 548.9 | 5.62 | 10.8 |
| Ch150 | 200 | 1000 | 0.1 | 0.01 | 1.0 | 1.0 | 0.8 | 6528 | 7107 | 7803.7 | 367.56 | 14.7 |

**Table 3** Comparison between the VTGA[4] and the GA[5] on Forest Planning

| Algorithm | Average Fitness | The Best Fitness | The Number of Trials Finding the Global Optimum |
|---|---|---|---|
| VTGA | **5794786.28011043** | **5500330.279304971** | **21** |
| GA | 8107453.58754281 | 5502420.092225004 | 0 |

## 4 Conclusion and Future Work

**Favoring Bad Solutions.** Traditional GA favors good solutions by giving them more opportunities for recombination. Considering that improving good solutions is harder and harder while their fitness values increase, we implement a mechanism draws more attention to improving bad solutions.

[3] Host Size = 10, Virus Size = 1000, Crossover Rate = 0.7, Mutation Rate = 0.01, Infection Rate = 0.2, Spread Rate = 0.2 and Recovery Rate = 0.8.

[4] Host Size = 10, Virus Size = 1000, Crossover Rate = 0.3, Mutation Rate = 0.01, Infection Rate = 0.9, Spread Rate = 0.8 and Recovery Rate = 0.8.

[5] Population = 1000, Selection = Tournament, Crossover = 2-point, Crossover Probability = 0.8, Mutation = Random Resetting, Mutation Probability = 0.4, Mutation Rate = 0.01, Elitism = Yes and Repair = Yes.

**Isolated Evolution.** In the VTGA, viruses are isolated within different hosts. Evolution, therefore, involves only viruses in the same host. Through experiments, we find that this mechanism helps with maintaining diversity in the population.

**Different Implementation of Operators.** We implement a one-way crossover sending genetic information only from strong viruses to weak viruses. And the way this operator delivers information is similar as the way uniform crossover performs in a traditional GA. We think there are other ways to implement operators.

**The Role of the Algorithm.** Because the VTGA uses the host population to record good solutions of different areas of the search space, it preserves diversity well. Besides, the GA can only return one solution after it converges. So people generally don't have many choices. However, the VTGA returns many diverse solutions with relative good fitness values when it finishes one computation, which helps us understand the search space of a problem better by giving us representative solutions from different areas and gives us flexibility to adjust our decisions.

## References

[1] Holland, J.H.: Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor (1975)
[2] Davis, L.D.: Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York (1991)
[3] Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989)
[4] Kubota, N., Fukuda, T., Shimojima, K.: Virus-evolutionary gentic algorithm for a self-organizaing manufacturing system. Computers & Industrial Engineering 30(4), 1015–1026 (1996)
[5] Kubota, N., Shimojima, K., Fukuda, T.: The role of virus infection in virus-evolutionary genetic algorithm. In: Proceedings of IEEE International Conference on Evolutionary Computation, pp. 182–187 (1996)
[6] Anderson, R.M., May, R.M.: Population biology of infectious diseases: Part 1. Nature 280, 361–367 (1997)
[7] de Castro, L.N., Timmis, J.: Artificial Immune Systems: A New Computational Intelligence Approach. Springer, London (1996)
[8] Molga, M., Smutnicki, C.: Test functions for optimization needs (2005), http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf
[9] Larrañaga, P., Kuijpers, C.M., Murga, R.H., Inza, I., Dizdarevic, S.: Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators. Artificial Intelligence Review 13(2), 129–170 (1999)
[10] Syswerda, G.: Schedulel Optimization Using Genetic Algorithms. In: Handbook of Genetic Algorithms, pp. 332–349. Van Nostrand Reinhold, New York (1991)
[11] Bettinger, P., Zhu, J.: A new heuristic for solving spatially constrained forest planning problems based on mitigation of infeasibilities radiating outward from a forced choice. Silva Fennica 40(2), 315–333 (2006)