*Proceedings of the 2012 International Conference on Industrial Engineering and Operations Management*
*Istanbul, Turkey, July 3 – 6, 2012*

# A Viral Systems Algorithm for the Traveling Salesman Problem

**Dedy Suryadi, and Yoana Valois Maryska Kandi**
**Department of Industrial Engineering**
**Parahyangan Catholic University**
**Bandung 40141, Indonesia**

## Abstract

The Traveling Salesman Problem (TSP) is a complex combinatorial problem, This research proposes a Viral Systems algorithm to solve the TSP. In Viral Systems, a solution is encoded into the genome of a cell. The virus that infects a cell may replicate. There are two types of replications, i.e. lytic and lysogenic. The end of lytic replication leads to the infection of neighboring cells, while the end of lysogenic replication causes genome mutation. Based on parameter testing, nearly all parameters influence the algorithm's performance. The performance of the algorithm is as good as the previous result using Genetic Algorithm.

## Keywords
Traveling Salesman Problem (TSP), Viral Systems, metaheuristic, operations research.

## 1. Introduction
The Traveling Salesman Problem (TSP) has drawn much interest from researchers, mainly because the problem is very easy to describe, yet it is difficult to solve [1]. TSP is a decision-making problem, which may be analogized with the problem faced by a salesman who needs to visit a number of cities. It is given the matrix distance between the cities, and it is required to find a tour which visits each city exactly once with minimum total distance [2]. There are quite some problems that may be considered as a TSP. The examples of those problems are the cutting stock problem, crystal structure analysis, job scheduling on a machine, etc. [1].

TSP is a complex combinatorial problem. The number of its feasible solutions grows significantly when the problem involves more cities to visit. For example, a TSP involves 5 cities has 120 possible solutions. However, merely adding five more cities to the problem raises the number of possible solutions to no less than 3,628,800. In general, the number of feasible solutions for a TSP is n!, in which n is the number of cities involved in the problem.

Optimization methods, such as Dynamic Programming and Branch-and-Bound, may be applied to solve the TSP. These methods, obviously, guarantee the final solution to be the optimal solution. However, the TSP which involves a great number of cities surely needs a great amount of computational time to solve with those methods [3]. In general, many real-world problems from Operations Research are very complex and therefore relatively difficult to solve by conventional optimization techniques [4].

In order to overcome the problem related to computational time, heuristics may be applied to solve the TSP. In particular, there is a group of heuristics that draws the analogy from the nature. The methods belong to the group is usually named metaheuristics. There have been quite a number of metaheuristics being applied to solve the TSP, for example Genetic Algorithm [5], Simulated Annealing and Tabu Search [6], Ant Colony System [7], Particle Swarm Optimization [8].

This research applies Viral Systems in order to design an algorithm to solve the TSP. Viral Systems is an algorithm which draws the analogy from the virus infection to an organism's cell. It is a relatively new metaheuristic and it seems quite promising. For Steiner problem, Viral Systems performs better than the other metaheuristics in the experimental cases [9].

The following sections are organized as follows. The next section describes the formulation of TSP. Viral Systems is presented in the third section, including the particular genome encoding and decoding, neighborhood structure, and mutation procedure used in this research, as well as the general algorithm to solve the TSP. The fourth section shows the result of algorithm implementation in the experimental cases, along with the discussions of it. In the last section, conclusions of this research are drawn.

## 2. The Traveling Salesman Problem (TSP)

The Traveling Salesman Problem (TSP) which involves N cities may be formally formulated as an Integer Programming [3]. First, let the decision variable of the problem be defined as follows:

$x_{ij}$ = 1, if the solution to the TSP goes from city i to city j (i = 1, 2, …, N; j = 1, 2, …, N; i≠j)
= 0, otherwise

The objective function may be formulated as follows:

$$\min z = \sum_i \sum_j c_{ij} x_{ij} \tag{1}$$

in which z is the total distance of the tour, and $c_{ij}$ is the distance between city i and city j (i≠j).

There are three constraints for the problem, they are:

$$\text{s.t.} \quad \sum_{i=1}^{i=N} x_{ij} = 1 \quad (\text{for } j = 1, 2, \ldots, N) \tag{2}$$

$$\text{s.t.} \quad \sum_{j=1}^{j=N} x_{ij} = 1 \quad (\text{for } i = 1, 2, \ldots, N) \tag{3}$$

$$u_i - u_j + N x_{ij} \leq N - 1 \quad (\text{for } i \neq j;\ i = 2, 3, \ldots, N;\ j = 2, 3, \ldots, N) \tag{4}$$

$$\text{All } x_{ij} = 0 \text{ or } 1, \text{ All } u_j \geq 0$$

The constraint (2) ensures that a city is visited exactly once from any of other cities. The constraint (3) ensures that a city visits exactly one and only one city among all cities. The constraint (4) is used to ensure the solution to become a complete tour. Any solution which contains a subtour violates the inequality, and thus is regarded as an infeasible solution to the TSP problem.

## 3. Viral Systems

In Viral Systems, a solution is encoded into the genome of an organism's cell. The genome itself consists of genes. In the TSP, a solution is any feasible tour which is visited by the salesman. Thus, a genome is the code which represents a particular feasible tour.

In this research, for a TSP which involves N cities, the genome's length is (N+1) genes. The values of the first N genes are integers, from 1 to N. The value of the (N+1)-th gene is the same with the first gene, because in the TSP the salesman must go back to the first visited city to complete the tour. Let us consider a TSP which involves 5 cities. An example of a genome for the problem is shown in Figure 1.

| 1 | 3 | 4 | 5 | 2 | 1 |
|---|---|---|---|---|---|

Figure 1: A genome example for a 5-city TSP

The decoding of the genome in Figure 1 gives a solution for the problem. It is a tour that starts at City 1, then visits City 3, City 4, City 5, and City 2, respectively. The tour is completed by going back to the first visited city, i.e. City 1.

In Viral Systems, all infected cells are stored in a clinical picture. The clinical picture is updated at each iteration. In computational term, infected cells are the solution obtained and examined at a particular iteration.

At each iteration, each infected cell has the chance to free itself from infection by producing antigene. For those unable to produce antigene, the virus inside the cell undergoes replication. There are two replication types, they are

lytic replication and lysogenic replication. After a particular limit has been passed, lytic replication results in the breaking of the cell's border and the replicated viruses infected its neighboring cells. Meanwhile, after a certain condition as well, lysogenic replication results in the mutation of the cell's genome.

### 3.1 Lytic Replication

Lytic replication occurs with the probability plt. During this type of replication, the virus lodging inside the cell replicates itself. The number of replicated viruses in an infected cell is recorded in the variable NR. Each time lytic replication occurs, the NR is added by a random variable Z, which is governed by the probability:

$$P(Z = z) = \binom{LNR}{z} p_r^z (1 - p_r)^{LNR - z} \tag{5}$$

in which LNR is the limit for lytic replication and pr is the probability of replicating a virus. Once the number of replicated viruses inside a cell (NR) exceeds the limit, LNR, the replicated viruses break the cell's border and infect the neighboring cells. A cell's LNR is computed using the equation:

$$LNR = LNR^0 \left( \frac{f(x) - f(\bar{x})}{f(\bar{x})} \right) \tag{6}$$

in which $LNR^0$ is a determined constant, f(x) is the cell's objective function value, and $f(\bar{x})$ is the best objective function value which has been found so far.

Since the consequence of this replication might lead to the infection of neighboring cells, it is required to define the neighborhood of an infected cell. In this research, the neighboring cells are created by swapping the genes of the infected cell. The rule is swapping two genes at a time, starting from the 2nd and the 3rd genes until the (N-1)-th and the N-th genes. Each swap results in a neighboring cell. Thus, each cell has (N-2) neighbors. For example, a cell whose genome is shown in Figure 1 has 3 neighboring cells. They are shown in Figure 2.


Figure 2: The neighboring cells of a genome shown in Figure 1

The first and the last cells are not swapped in order to maintain the starting (and the ending) city of the tour. It is done so because neighboring cells should have characteristics that relatively resemble the infected cell.

### 3.2 Lysogenic Replication

Lysogenic replication occurs with the probability (1-plt). During the lysogenic replication, the virus does not replicate itself. It simply lodges inside the cell until an external cause activates it, which results in the mutation of the cell's DNA. In the algorithm, the mutation occurs after lysogenic replication happens for a particular number of iterations. Therefore, a variable IT is used to record the number of iterations of which lysogenic replication occurs.

Each time the virus inside a cell undergoes lysogenic replication, its IT is added by 1. Once the number of iterations (IT) has passed its limit, LIT, the mutation happens. The LIT itself is computed using the equation:

$$LIT = LIT^0 \left( \frac{f(x) - f(\bar{x})}{f(\bar{x})} \right) \tag{7}$$

in which $LIT^0$ is a determined constant, f(x) is the cell's objective function value, and $f(\bar{x})$ is the best objective function value which has been found so far.

In the contrast to the procedure of defining neighboring cells, the mutation procedure should be able to produce a cell which has a related but relatively different genome with the original cell. Based on the idea, the mutation procedure in this research is performed by swapping the first gene and another gene which is picked arbitrarily. It should be noted, however, that the other gene must not be either the first or the last gene. In other word, the other gene must be picked among the second up to the N-th gene. After the swap takes place, the procedure is completed by conforming the last gene to the first one.

For example, let us consider a cell with the genome as shown in Figure 1. Suppose the virus inside it undergoes lytic replication such that mutation occurs. The first gene is 1. Suppose the third gene is arbitrarily picked, it is 4. They are swapped, such that the first gene becomes 4 and the third gene becomes 1. The last gene becomes 4 as well. It is illustrated in the figure in Figure 3.


Figure 3: An illustration of mutation procedure

**3.3 The Algorithm for the TSP**

The algorithm for solving the TSP is generally listed as follows.

1. Determine the required inputs for the TSP, i.e. the number of cities and the distance between cities.
2. Determine the value of Viral Systems parameter, i.e. the maximum number of iterations (ITER), the size of clinical picture (POB), the probability of occurring lytic replication (plt), the probability of infecting a neighboring cell (pi), the probability of replicating a virus (pr), probability of producing antigene (pan), the constant multiplier for computing LNR ($LNR^0$), and the constant multiplier for computing LIT ($LIT^0$).
3. Create infected cells as many as POB and put them in the initial clinical picture. Each infected cell has a unique genome, has NR equals 0 and IT equals 0.
4. Start the first iteration; t = 1.
5. Start checking the objective function value and antigene production for the first infected cell, m = 1.
6. Compute the objective function value for cell m. Update the best objective function value found so far $f(\bar{x})$, if the objective function value of cell m is better (lower) than the previously recorded $f(\bar{x})$.
7. Check the ability of cell m to produce antigene, based on the probability of producing antigene is pan. If cell m is able to produce antigene, the cell is deleted from the clinical picture.
8. If m ≠ POB, then m = (m+1) and go back to step 6. Otherwise, go to the next step.
9. Start the replication for the first cell in the clinical picture, m = 1.
10. Determine the replication type for the virus inside cell m, based on the probability of occurring lytic replication is plt. If the replication type is lytic, then go to the next step. Otherwise, go to Step 15 for lysogenic replication.
11. Update the NR of cell m; NR = NR + Z.
12. Compute the LNR for cell m. If its NR is larger than or equal to LNR, then go to the next step. It means, the replicated viruses break the cell's border. Otherwise, go to Step 19.
13. Delete cell m from clinical picture. Start infecting neighboring cells, based on the probability of infecting a neighboring cell is pi.
14. For each infected neighboring cell, check whether it is able to produce antigene. Keep the neighboring cells who are unable to produce antigene. Go to Step 18.
15. Update the IT of cell m; IT = IT + 1.
16. Compute the LIT for cell m. If LIT is larger than or equal to LIT, then go to the next step. It means, the virus is activated and thus it will mutate the cell's genome. Otherwise, go to Step 19.
17. Delete cell m from clinical picture. Perform the mutation procedure by selecting the first gene and another gene randomly. Swap those genes and conform the last gene to the new first gene.
18. Store either neighboring cells or mutated cell in a temporary storage.
19. If m ≠ POB, then m = (m+1) and go back to step 10. Otherwise, go to the next step.
20. If there is any duplication between the genome of cells in the temporary storage, remove the duplicates. If there is any duplication with the cells in the clinical picture, delete the duplicates as well.
21. Insert the cells in the temporary storage into the clinical picture. If the number of cells in the temporary storage is larger than POB, then delete all cells in the clinical picture and select cells from the temporary storage starting from the best to enter the new clinical picture.
22. If t ≠ ITER, then t = (t+1) and go back to step 5. Otherwise, the algorithm stops with the best solution found is the one whose objective function value is recorded as $f(\bar{x})$.

## 4. Experimental Cases and Discussions

The first experimental case is taken from a previous research which applies Genetic Algorithms to solve the TSP [10]. The case involves 10 cities. The distance between cities is shown in Table 1.

Table 1: Distance between cities for the 10-city TSP experimental case

| From\To | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 14 | 23 | 25 | 36 | 42 | 50 | 10 | 25 | 32 |
| 2 | 14 | 0 | 17 | 23 | 30 | 36 | 90 | 22 | 67 | 84 |
| 3 | 23 | 17 | 0 | 29 | 35 | 28 | 40 | 33 | 89 | 12 |
| 4 | 25 | 23 | 29 | 0 | 17 | 11 | 30 | 44 | 32 | 90 |
| 5 | 36 | 30 | 35 | 17 | 0 | 6 | 20 | 55 | 65 | 45 |
| 6 | 42 | 36 | 28 | 11 | 6 | 0 | 10 | 66 | 12 | 31 |
| 7 | 50 | 90 | 40 | 30 | 20 | 10 | 0 | 77 | 86 | 74 |
| 8 | 10 | 22 | 33 | 44 | 55 | 66 | 77 | 0 | 55 | 23 |
| 9 | 25 | 67 | 89 | 32 | 65 | 12 | 86 | 55 | 0 | 12 |
| 10 | 32 | 84 | 12 | 90 | 45 | 31 | 74 | 23 | 12 | 0 |

The best result obtained from Genetic Algorithm is 157. In this research, the value of each parameter for testing the case is shown below:
ITER = 50,000
POB = 100
plt = 0.6
pi = 0.4
pr = 0.2
$LNR^0$ = 10
$LIT^0$ = 10
pan = 0.5

From 10 replications, the algorithm reaches 157 twice. The average objective function value obtained is 165.30, with standard deviation 7.57. Therefore, for this case, Viral Systems is able to reach the solution as good as the one obtained by Genetic Algorithms.

The second experimental case is used to examine the Viral Systems parameters which affect its performance significantly. The performance itself is defined in terms of the best objective function value found at each replication. The second case involves 48 cities, taken from TSPLIB [11]. The optimal solution for the case is known to have objective function value equal to 5046.

There are 5 parameters tested and there are 2 levels for each parameters. Thus, there are 32 ($2^5$) combinations of parameters tested. The values of the parameters tested are shown below:
plt = 0.3 and 0.7
pi = 0.1 and 0.9
pr = 0.2 and 0.8
$LNR^0$ = 10 and 20
$LIT^0$ = 10 and 20
The fixed parameters are ITER = 10,000 and POB = 100. Each combination is run for 15 replications.

The best solution found is 46-28-43-24-20-38-19-11-14-48-18-13-5-15-6-21-23-47-2-17-22-26-39-40-35-30-25-44-41-34-9-36-29-16-8-10-12-31-45-32-27-1-7-4-42-3-33-37-46, with total distance equal to 14,931.

Based on the ANOVA ($\alpha$ = 5%), the parameters and their interactions found to be statistically significant are: $LIT^0$, pr, pi, the interaction between $LIT^0$ and pr, the interaction between pr and pi, and the interaction between $LNR^0$, pr, pi.

The only parameter tested which does not affect the objective function value significantly is plt, which means there is no significant difference whether lytic replication is allowed to occur less or more. It may happen because the values for plt are not too extreme, such that the parameter's effect is not clearly seen. The parameter $LNR^0$, which is also related to lytic replication, only occurs in the interaction. In order this parameter to take effect, it seems that the value should be quite small, i.e. smaller than 10. The reason is that using the updating clinical picture strategy as in this research allows the cells to be replaced before it finishes lytic replication, due to the relatively high $LNR^0$.

## 5. Conclusions

The algorithm for solving the TSP based on Viral Systems has been proposed in this paper. The performance matches that of the previous research using Genetic Algorithm. However, for another case from the TSPLIB, the performance is indeed still lower than the known optimal. The parameter testing suggests that nearly all parameters give significant effect to the objective function value obtained. For further research, it is suggested that the level of parameters tested is added in order to obtain better result as well as obtaining a more complete image of the effect of each parameter.

## References


1. Hoffman, K., and Padberg, M., [Online]. "The Traveling Salesman Problem" http://iris.gmu.edu/~khoffman/papers/trav_salesman.html
2. Lin, S., and Kernighan, B.W., 1973. "An Effective Heuristic Algorithm for the Traveling-Salesman Problem", Operations Research, Vol. 21, No. 2, 498-516.
3. Winston, W. L., 2004. "Operations Research: Applications and Algorithms", 4th Edition, Duxbury Press, Belmont.
4. Gen, M., Cheng, R., and Lin, L., 2008. "Network Models and Optimization: Multiobjective Genetic Algorithm Approach," Springer-Verlag London Limited.
5. Freisleben, B., and Merz, P., 1996. "New Genetic Local Search Operators for the Traveling Salesman Problem," Parallel Problem Solving from Nature – PPSN IV: Lecture Notes in Computer Science, Volume 1141/1996, 890-899.
6. Malek, M., Guruswamy, M., Pandya M., and Owens, H., 1989. "Serial and Parallel Simulated Annealing and Tabu Search Algorithms for the Traveling Salesman Prbolem," Annals of Operations Research, 21, 59-84.
7. Dorigo, M., and Gambardella, L. M., 1997. "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem," IEEE Transactions on Evolutionary Computation, Vol. 1, No. 1.
8. Zhong, W. L., Zhang, J., and Chen, W. N., 2007. "A Novel Discrete Particle Swarm Optimization to Solve Traveling Salesman Problem," 2007 IEEE Congress on Evolutionary Computation (CEC), Singapore.
9. Cortés, P., Garcia, J. M., Munuzuri, J., Onieva, L., 2008. "Viral Systems: A New Bio-Inspired Optimation Approach," Computers & Operations Research 35, 2840 – 2860.
10. Wijaya, S., 2006. "Performance Comparison Between Genetic Algorithm and Branch-and-Bound Algorithm for the Traveling Salesman Problem," (in Indonesian), Thesis, Parahyangan Catholic University, Bandung.
11. Reinelt, G., [Online]. TSPLIB http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/