# Parallel Implementations of Cellular Automata-Based Traffic Models Using the AGILA Cluster Computer

Eden Delight B. Provido, Mark Adrian Ramires, Winfer C. Tabares, and Rafael P. Saldaña
Computational Science and Scientific Computing Group
Mathematics Deparment, Ateneo de Manila University
E-mail for correspondence: rsaldana@ateneo.edu

## ABSTRACT

Previous studies have shown that the simulation using cellular automata models traffic flow based on theoretical assumptions and data from the classical theory of traffic dynamics. In this study, the parallel implementation of the one-lane and two-lane freeway models of vehicular traffic dynamics on the AGILA cluster computer are presented. The implementation technique used is the Master-Slave technique. Results gathered using the parallel implementations show that the present model benefits from parallelization.

## Keywords

Traffic Modeling, Parallel Computing, Cellular Automata, AGILA Cluster Computer

## 1.INTRODUCTION

Cognizant of the importance of studying vehicular traffic dynamics, the Computational Science and Scientific Computing Group of the Mathematics Deparment of Ateneo de Manila University has embarked on a series of computational work involving computer simulations and cellular automata-based algorithms [2, 3, 4, 5, 7].

In the present study, parallel implementations of 1D and 2D using a modified version of the Nagel and Schreckenberg's [1] cellular automata model of traffic dynamics are done on the AGILA cluster computer.

## 2. THE CA MODEL

The simulations are based on the model developed by Nagel and Schreckenberg [1 ]. The following assumptions are made:

- Each time step is equivalent to one second.
- Cars can accelerate from 0 kph to a maximum of 130 kph or, equivalently, from 0 cells to 6 cells forward.
- The movement of the vehicles is from left to right with periodic boundary.
- Initial arrangement of cars for the program testing is randomly generated. Hence, a normalization scheme is employed; that is, data gathering will start at the $11^{th}$ time step.
- When a car changes lane, it is placed on the cell in front of it on the other lane.

The simulations for one-lane and two-lane models are programmed in parallel using C, and implemented on the AGILA cluster built by the Computational Science and Scientific Computing Group of the Ateneo de Manila University [ 6].

## 3.THE AGILA BEOWULF CLUSTER

In 2000 a team of researchers belonging to the School of Science and Engineering of the Ateneo de Manila University built a Beowulf-class parallel computer called the AGILA High Performance Computing System. Details about the features of the AGILA cluster computer are reported in [6 ].

## 4.PARALLEL IMPLEMENTATION

Parallel programming uses multiple computers, or computers with multiple internal processors. It is usually employed to solve a problem that demands greater computational speed. It gives more opportunity to tackle bigger problems. Parallel programming is used to solve problems with more computational steps or memory requirements; the latter is because more total memory is often possessed by multiple computers and multiprocessor systems than a single computer. In this study, multiple computers that communicate between themselves by sending messages will be used. This kind of implementation is called *message-passing parallel programming*. The environment used to run all the parallel implementation in this paper is the AGILA Cluster Computer.

There are various techniques in designing a parallel program. The techniques apt to be used largely depend on the nature of the problem. In parallel programming, problems are classified into three groups based on their nature—a problem can be inherently parallelizable (coarse-grained), moderately parallelizable, and fine-grained. Inherently parallelizable problems are problems that can be subdivided into smaller problems or tasks. They are the easiest to parallelize and are most likely to benefit from parallelization. The most common technique used in parallelizing an inherently parallelizable problem is Partitioning or Divide-and-Conquer. The most difficult to parallelize are the fine-grained problems. The steps in solving

these problems are very connected and it seems impossible to divide them into smaller problems. Hence, fine-grained problems call for careful planning of its implementation [9].

Cellular Automata is classified as a fine-grained problem. This is largely because the state of each cell entirely depends on the state of the other cells in its neighborhood. However, it ceases to be a fine-grained problem if a subdivision of the problem according to neighborhoods, given that the neighborhood is not very large, is made.

In this paper, the technique used in parallelizing the CA traffic models is the Master-Slave technique. It is somewhat like the boss-worker relationship in human setting. One node will function as the maste while all the other nodes serve as slaves. The general framework of a Master-slave technique is as follows:

- Master notifies all the slaves that it is ready to distribute work.
- Slaves reply by sending a request for work.
- When master receives a request message, it sends work if there is still work available. Otherwise, it sends a message that there is no work available.
- When the slave receives the work, it instantly does the work and sends the result back to the master, together with a request for more work. If the slave receives the no work message, it exits the work environment.
- When the master receives the results, it saves them or prints them out.
- When all of its slaves have exited the work environment, the master does the same.

The work in the given framework can be translated to data or, specifically in the case of the CA implementation, the neighborhood and all the attributes of each cell with a car. For the one-lane model, work consists of the position of the car in the lane, its velocity, and the state of (up to its maximum velocity) cells in front of it; that is, if the maximum velocity of the car is 6, the state of the 6 cars in front of it is considered. In the two-lane model, work consists of the lane where the car presently belongs, its position on that lane, its velocity, and its neighborhood which consists of its forward neighborhood on its current lane, and its backward and forward neighborhood on the other lane.

The one-lane and two-lane traffic dynamics CA models are implemented, using the update rules of Nagel-Schreckenberg, in C and are run using AGILA. The input values are the following:

| | | |
|---|---|---|
| *NODES* | - | number of nodes |
| *lane_len* | - | length of the road (number of cells) |
| *time* | - | number of iterations |
| *cars* | - | number of cars |
| *percent* | - | percentage of jeepneys in the system. |

In accelerating and decelerating, cars exhibit certain randomness. To accommodate this randomness, a another variable, *p_noise*, is introduced. The variable *p_noise* is the probability that the driver will opt to retain his present velocity instead of accelerating when it is possible to accelerate.

To run the simulation, the user must type the following

**lamrun –np** *NODES* [simulation name] *lane_len time cars percent p_noise.*

# 5.RESULTS AND DISCUSSION

Previous studies have shown that the simulation using cellular automata models the traffic flow based on theoretical assumptions and data from the classical theory of traffic dynamics [6]. With that already shown, further investigations only need to check if the system benefits from parallelization. A system is said to benefit from parallelization if using more nodes (or processors) would mean shorter processing time. Since there are several variables involved in the study, each of them is considered one at a time. That is, one of variables except the one being considered is fixed while the number of nodes is varied (one to seven nodes are used). Then, the time elapsed, or the number of seconds it takes the program to process the data, is recorded. The default noise value used is 80%, probability that the drivers will accelerate, unless specified.

Figures 5.1-5.4 illustrate some of the Time vs Space plots of the system.

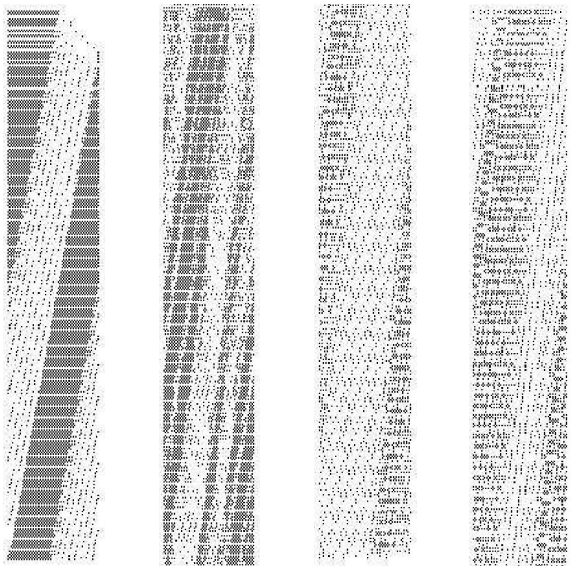

**Figure 5.1 One-lane without noise**

**Figure 5.2 One-lane with noise**

**Figure 5.3 Two-lane without noise**

**Figure 5.4 Two-lane with noise**

The figures above show the behavior of traffic as simulated in 1800 timesteps (roughly 30 minutes). Each row show in Figures 5.1 and 5.2 shows the behaviour of the lane in every second. In figure 5.3 and 5.4, every pair of rows shows the behavior of traffic in both lanes. It is obvious that when the randomness of acceleration is not considered (without noise), the behavior of the simulation follows a pattern. Once the randomness is introduced, the behavior of the traffic seems to be somewhat erratic which is what we usually observe in reality.

First, let us investigate on the general trend when the number of nodes is varied. Figures 5.5-5.8 illustrate the result of making all inputs except the number of nodes constant. The system is set

with 400 cells, 3600 iterations or roughly an hour, 300 cars, and 50% jeepneys.

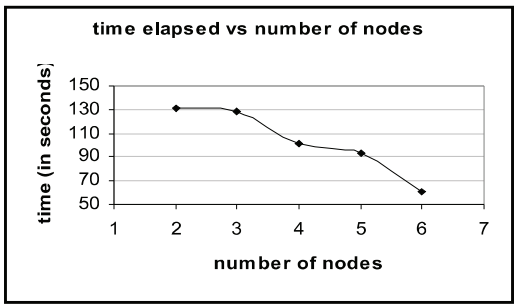


**Figure 5.5: One-lane Model without Noise**

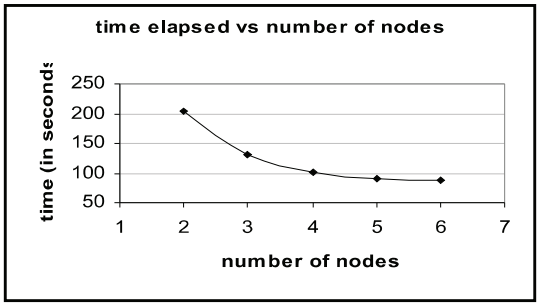


**Figure 5.6 One-lane Model with Noise**

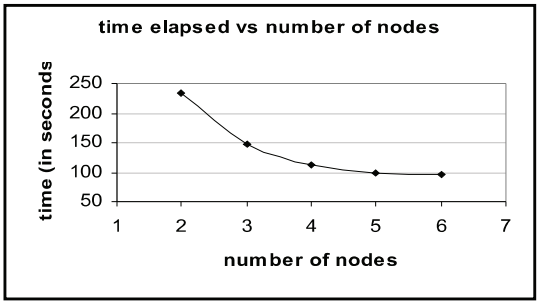


**Figure 5.7: Two-lane Model without Noise**

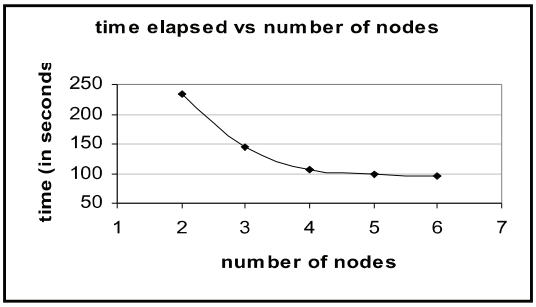


**Figure 5.8: Two-lane Model with Noise**

The general pattern shows that when the number of nodes is increased, the elapsed time decreases. In this case, it can be concluded that the system benefits from parallelization. One can also observe that as the number of nodes increases, the rate by which the elapsed time decreases is slower. This is because the efficiency of parallelization is dependent on the number of data fed into the system; that is, parallelization is useless if the number of data is very small. If the amount of data fed into the system is somewhat sufficient, the time taken by the nodes to communicate to each other is significantly higher than the time taken by the system to process all the data. Hence, a significant number of data is needed to perform efficient parallelization.

Having shown the general pattern, the efficiency of the system when the number of cars is varied is examined. For this purpose, a system with 400 cells, 3600 iterations or roughly an hour, and 50% jeepneys will be used. This set-up is executed using 2, 3, 4, and 5 nodes. In the figures below and for the rest of the figures, the line with square dots result using 2 nodes, triangle dots for 3 nodes, cross dots for 4 nodes and the diamond dots for 5 nodes.

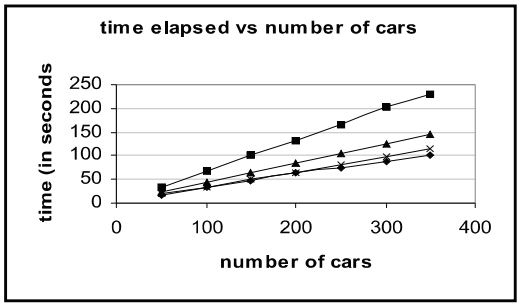


**Figure 5.9: One-lane Model without Noise**

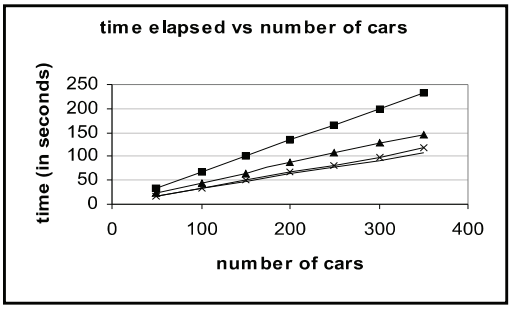


**Figure 5.10: One-lane Model with Noise**

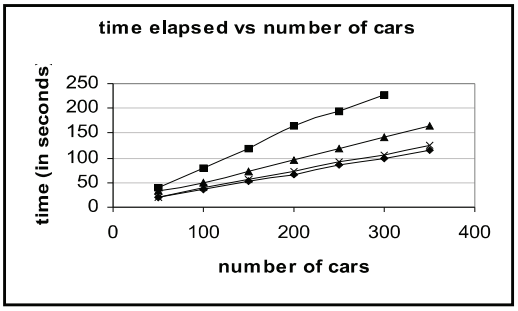


**Figure 5.11: Two-lane Model without Noise**

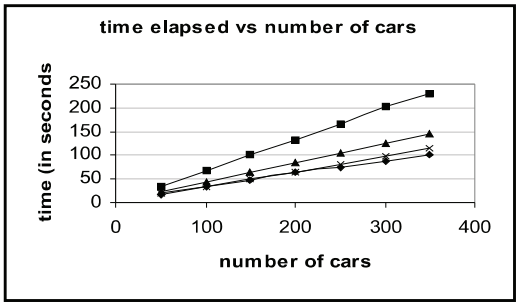


**Figure 5.9: One-lane Model without Noise**

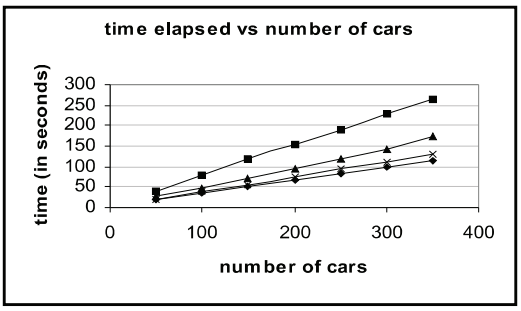


**Figure 5.12: Two-lane Model with Noise**

The figures show two things. It shows that as the number of cars in the system increases, the time elapsed increases as well. This result is obvious since it takes time to compute the properties of each car. Moreover, it also shows that more nodes working together makes the computation time faster. Again, in this case, the system benefits from parallelization. It can also be seen that the rate of change of the time elapsed is decreasing, as in the previous step.

For the next set-up, all inputs except the number of cells or the length of the road is fixed. The system is set with cars consisting 50% of the number of cells, 3600 iterations, and 50% jeepneys. The first notable thing about the graphs in Figures 5.13-5.16 is that they have the same trend as those in the previous set-up. The topmost line is for the 2-node set-up, and the line at the bottom is for the 5-node set-up.

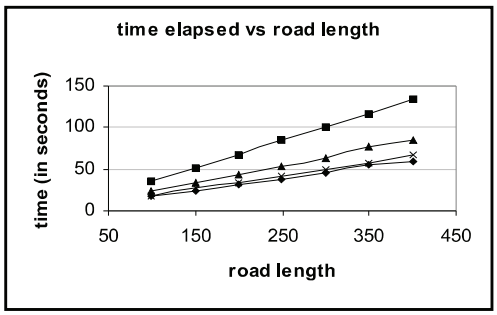


**Figure 5.13: One-lane Model without Noise**

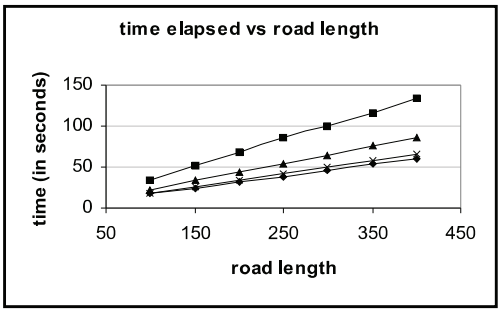


**Figure 5.14: One-lane Model with Noise**

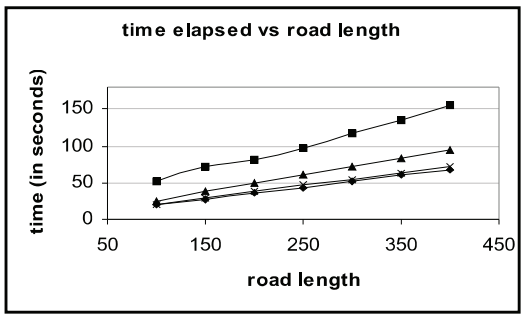


**Figure 5.15: Two-lane Model without Noise**

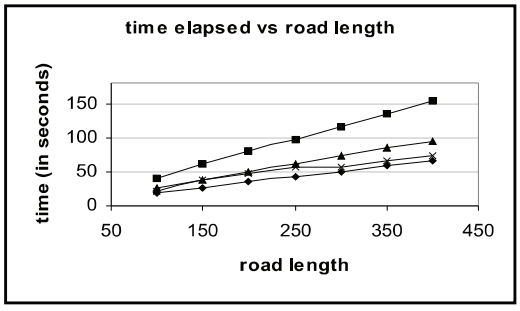


**Figure 5.16: Two-lane Model with Noise**

Each graph in Figures 5.13-5.16 shows that increasing the number of cells in the system will make the computation time longer. However, the fact that increasing the length of the road leads to an increase in the number of cars (set as 50% of the number of cells) should also be considered. For the table, the system is set with 100 cars, 100 iterations and 50 jeepneys. It shows that if the number of cells or the length of the road is increased without increasing the number of cars, the elapsed time will almost be the same.

| road length | time elapsed |
|---|---|
| 200 | 2.39 |
| 250 | 2.39 |
| 300 | 2.40 |
| 350 | 2.4 |
| 400 | 2.39 |
| 450 | 2.4 |

**Table 5.1**

This result explains why the graphs are very similar those in the previous set-up, when the number of cars in the system is varied. This, as in other cases, leads to the conclusion that the system benefits from parallelization.

In the next system, all the inputs except the number of iterations is held constant. In the following graphs, the highest line is the data for set-up with 400 cells and 200 cars, and the lowest line is for the set-up with 100 cells and 50 cars. The system is set with 5 nodes and 50% jeepneys, and a variable number of iteration that ranges roughly from 30 minutes (1800 iterations) to 3 hours (10800 iterations). It can be seen that, as the number of iterations increases, the time elapsed also increases. However, this does not give any proof of parallelization efficiency.

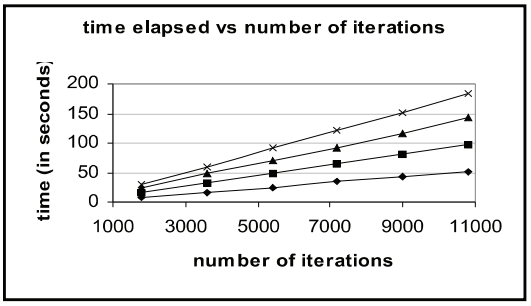


**Figure 5.17: One-lane Model without Noise**

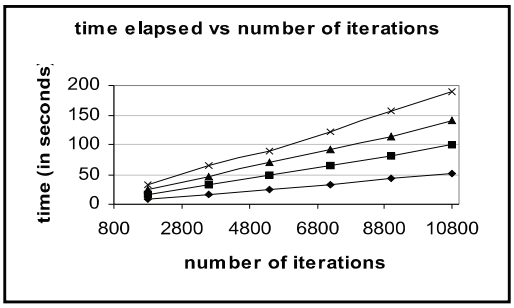


**Figure 5.18: One-lane Model with Noise**

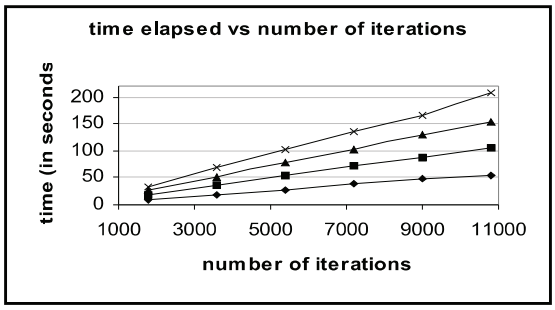


**Figure 5.19: Two-lane Model without Noise**

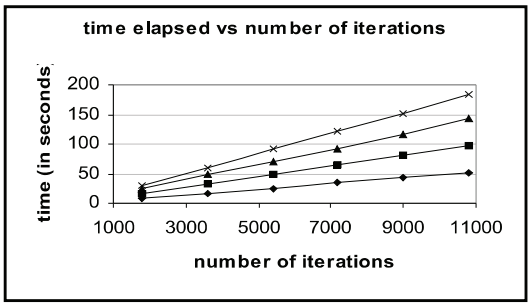


**Figure 5.17: One-lane Model without Noise**

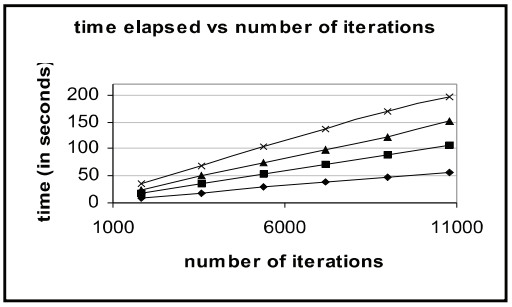


**Figure 5.20: Two-lane Model with Noise**

The only input left to be examined is the percentage of cars in the system.

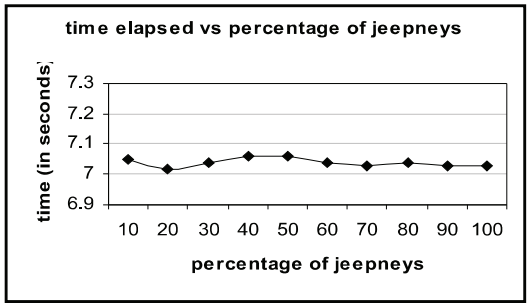


**Figure 5.21: One-lane Model without Noise**

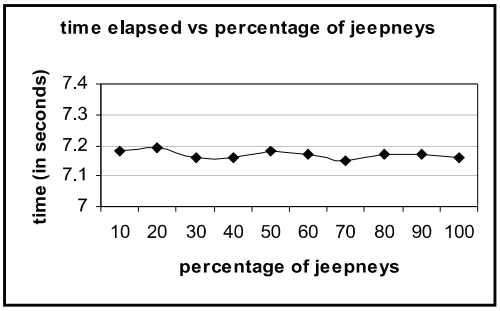


**Figure 5.22: One-lane Model with Noise**

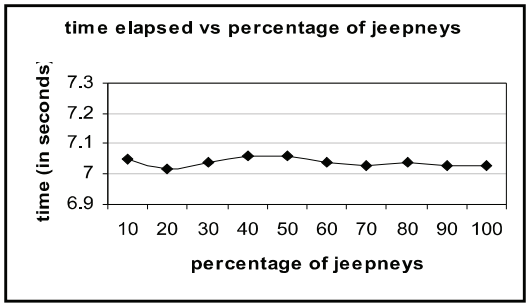


**Figure 5.21: One-lane Model without Noise**

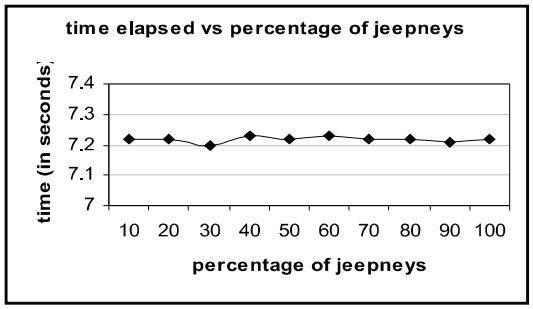


**Figure 5.23: Two-lane Model without Noise**

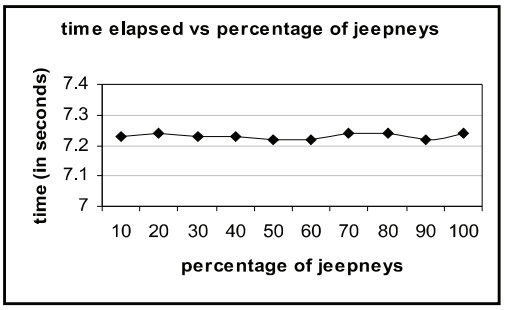


**Figure 5.24: Two-lane Model with Noise**

However, as the results in Figures 5.21-5.24 show, varying the percentage of cars in the system barely affects the computational time. Hence, using it to test the parallelization efficiency of the system will be redundant since it will just give results similar to the first set-up where the number of nodes is varied.

In summary, the variable that affects the efficiency of parallelization are the number of cars, the number of cells, and the number of iterations.

# 6.CONCLUSION

A parallel version of Nagel and Schreckenberg's one-dimensional and two-dimensional, multi-state, deterministic and stochastic cellular automata was programmed in C and implemented using AGILA Cluster Computer. The following observations have been noted:

- Taking all the input values into consideration, the system benefits from parallel implementation.
- When the number of nodes is increased, the time it takes for the processors to execute the work (elapsed time) decreases.
- As the number of cars in the system increases, the time elapsed increases as well. Moreover, more nodes working together makes the computation time faster.
- If the number of cells or the length of the road is increased without increasing the number of cars, the elapsed time will almost be the same.
- Even if the number of iterations is varied from 30 minutes to 3 hours, the system still benefits from parallelization.
- Varying the percentage of cars in the system barely affects the computational time. Hence, using it to test the parallelization efficiency of the system will be redundant.

# 7. ACKNOWLEDGEMENT

We would like to acknowledge funding support from the Commission on Higher Eduation (CHED) Center of Excellence Fund of the Mathematics Department of Ateneo de Manila University. We also thank Mr. William Yu for his assistance in parallel programming and the development of the AGILA cluster computer.

# 8.REFERENCES

[1] Nagel, K. and M. Schreckenberg (1992). "A Cellular Automaton Model for Freeway Traffic". J. Physique, Vol. 2, pp. 2221-2229.

[2] Saldaña, R. and Tabares, W. (2000). "Mathematical and Computatonal Aspects of Modeling and Simulation of Traffic Flow Dynamics". 19th Annual PAASE Meeting and Symposium. Makati City, Philippines.

[3] Saldaña, R. and Tabares, W. (2000). "Traffic Modeling on High Performance Computing Systems." Proceedings of the First Philippine Computing Science Congress (PCSCS 2000). Manila, Philippines.

[4] Saldaña, R. and Tabares, W. (2001). "A Study on Modeling Vehicular Traffic Dynamics: Comparison Between Macrosimulation and Microsimulation Methods." MODEL 2001. Mindanao State University-Iligan Institute of Technology, Iligan City, Philipppines.

[5] Saldaña, R. and Tabares, W. (2001). "A Cellular Automata-Based Study of Vehicular Traffic Dynamics." 2nd Philippine Computing Science Congress (PCSC 2001). Mindanao Stuate University-Iligan Institue of Technology, Iligan City, Philippines.

[6] Saldaña, R., Garcia, J., Muga, F., and Yu, W. (2001). Development of a Beowulf-Class High Performance Computing System for Compuational Science Applications. Science Diliman, vol. 13, no. 2, pp. 97-99.

[7] Tabares, Winfer C. (2002). "A Cellular Automata-Based Study of Vehicular Traffic Dynamics." M.S. Thesis, Ateneo de Manila University, Quezon City, Philippines.