Simulation of a Vehicle Traffic Control Network Using a Fuzzy Classifier System

John R. Clymer
California State University Fullerton
Jclymer@fullerton.edu

Abstract

A Complex Adaptive System (CAS) is a network of communicating, intelligent agents where each agent adapts its behavior in order to collaborate with other agents to achieve overall system goals. Further, the overall system often exhibits emergent behavior that cannot be achieved by any proper subset of agents alone. A graphical simulation library called Operational Evaluation Modeling for Context-Sensitive Systems (OpEMCSS) has been developed to simulate complex systems, including CAS. This simulation library includes a Classifier Event Action block that is a forward chaining, expert system controller. The Classifier Event Action block can implement both crisp and fuzzy rules. A network of traffic light controller agents, one at each intersection, is simulated for a city traffic grid. Each traffic controller agent uses a fuzzy classifier block to make decisions about traffic light timing in order to minimize local vehicle wait time. Out of the co-evolutionary interaction of these agents, emerges the global minimization of vehicle wait time in the network.

1. Introduction

A Complex Adaptive System (CAS) is a network of communicating, intelligent agents where each agent adapts its behavior in order to collaborate with other agents to achieve overall system goals. Further, the overall system often exhibits emergent behavior that cannot be achieved by any proper subset of agents alone. CAS includes satellite communications networks, vehicle traffic control networks, multi-national corporations, the global economic system, ecological systems, flexible manufacturing systems, and military command and control C4ISR networks [1].

A graphical simulation library called Operational Evaluation Modeling for Context-Sensitive Systems (OpEMCSS) has been developed to simulate CAS [5-9]. OpEMCSS works with EXTEND, a powerful yet relatively inexpensive simulation tool. OpEMCSS implements the Operational Evaluation Modeling (OpEM) graphical Discrete Event Simulation (DES) language. The OpEM DES language, based on interacting concurrent processes, includes primitives for process flow and concurrent process interactions such as resource contention, process synchronization, and process communication and adaptation [2,5,7,9].

An agent is a CAS component capable of perceiving and acting on its own behalf, and it decides for itself what needs to be done to satisfy its own design objectives [12]. Agent operation is modeled using the OpEM language as a collection of communicating process instances that perform all agent functions. Each agent decision-making function is implemented using a classifier system block.

Holland [10] has promoted the idea of a classifier system for many years. In this paper, the OpEMCSS Classifier Event Action block, that can use fuzzy rules to make decisions, is applied. This block is comparable to Holland's classifier system [8].

A distributed, vehicle traffic control network located in a large city is discussed in this paper. Each major intersection has a vehicle traffic light control agent that incorporates a fuzzy classifier system to determine traffic light timing. In this system, each traffic light agent uses its perceptions about incoming traffic flow to optimize light timing, minimizing local vehicle waiting time. The result of each traffic light agent adapting light timing to accommodate local traffic flow coming from other intersections is to minimize the average waiting time in the entire network. Global minimization of traffic waiting time results as a consequence of the emergent behavior of this system, which is the self-synchronization of light timing by each traffic control agent.

2. Fuzzy Classifier Event Action Block

The Classifier Event Action block contains a forward chaining, inference engine that uses condition-action rules to transform condition attributes into action attributes. The condition attributes are obtained from a process instance item passing through the block [9]. After inference is complete, action attributes are added to the process instance item passing through the block before the item is sent to the output connector. When the classifier system block is used as a fuzzy controller, several rules may be eligible in a decision context. A weighted average is calculated over all eligible rules to produce action attribute values used to control traffic light timing.

Process attributes of the form "AttributeName = RealValue," that represent state variables in EXTEND, must be transformed into a symbolic form "AttributeName = ValueName" suitable for the inference engine. Low and High values are specified by a command in the rule file that defines the range of numerical values allowed for each symbolic value name. The result of each transformation is a set of one or more facts of the form "AttributeName=ValueName" that can be used during the inference process.

A similar command must be placed at the beginning of the rule file to accomplish the required reverse transformation when a rule fires. The Low and High values for action attributes define the range of numerical values allowed for each symbolic value name. The result of each transformation is one or more process attributes of the form "AttributeName = RealValue" added to the process instance item passing through the block.

For fuzzy rules, the inference engine requires a Confidence Factor (CF) for each condition in order to compute the rule support that is needed to calculate the weighted average for the action attribute real value. The format of the ConditionFuzzySet command is:

ConditionFuzzySet(AttributeName) =
ValueName(A, B, C)

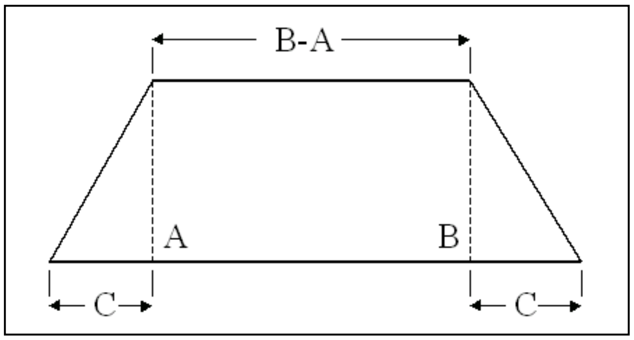


Figure 1. Fuzzy set membership function format.

The fuzzy set function defined by A, B, and C has a trapezoidal shape shown in figure 1. The top of the trapezoid (length [B-A]) is smaller than or equal to the base (length [[B-A]+2C]). At the top of the trapezoid, the CF is 100. The A and B values define where the function begins a linear descent to zero. The C value defines the slope, ABS(100/C), of the descent. Thus, if C is zero, the trapezoid becomes a rectangle. If A equals B and C is not zero, the function has a triangular shape.

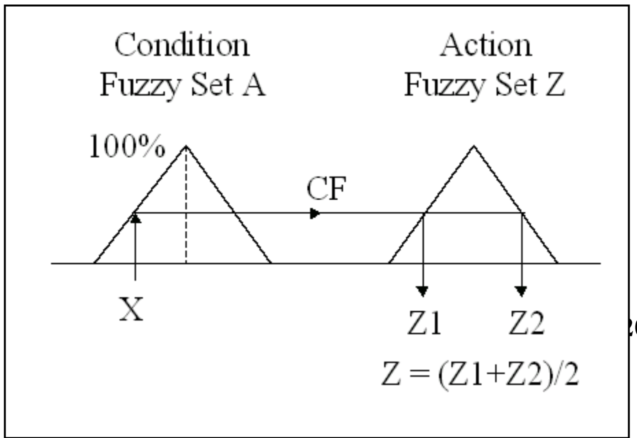


Figure 2. Weighted average calculations.

Fuzzy sets can be defined for action facts as well as condition facts. The fuzzy set function takes an action fact confidence factor (CF) and transforms it into an output, real attribute value “AttributeName = RealValue” that is added to the process instance item passing through the block. The real attribute value is computed as the weighted average over all occurrences of each AttributeName, obtained from the set of eligible rules, using the fuzzy set definition. The ActionFuzzySet command that defines the conversion of Cfs into output real attribute values is as follows:

ActionFuzzySet(AttributeName) =
ValueName(A, B, C)

The value X shown in figure 2 is obtained for a rule condition fact. The condition fuzzy set definition for the fact is used to obtain the degree of fuzzy set membership CF as discussed above. If several fuzzy conditions are combined by an AND operation, the minimum CF for each fact is used for value X, and it is then used to obtain values Z1 and Z2 as shown in figure 2. The average value Z=(Z1+ Z2)/2 is computed as shown. A weighted average Z’ is computed for all eligible rules that have the same action name but different value names. This calculation is done for each action name specified by the eligible rules. Therefore, a rule that has several different fuzzy action attributes, has a fuzzy output value computed for each fuzzy action attribute. Each fuzzy action attribute, specified for the selected rule, is added to the process instance item passing through the block using the fuzzy action real value calculated for it.

An example rule is as follows:

```
RuleName:IF
  ConditionName1 = ValueName1 AND
  ConditionName1 = ValueName2 AND
  ConditionName2 = ValueName3,
THEN
  ActionName1=ValueName4 AND
  ActionName2=ValueName5, CF=100.0%
```

3. Distributed, Vehicle Traffic Control Network

A distributed, vehicle traffic control network located in a large city is considered next. Each major intersection has a vehicle traffic light control agent to determine traffic light timing. In this system, each traffic light agent uses its perceptions about incoming traffic flow to optimize light timing, minimizing local vehicle waiting time. The result of each traffic light agent adapting light timing to accommodate local traffic flow coming from other intersections is to minimize the average waiting time in the entire network. Global minimization of traffic waiting time results as a consequence of the emergent behavior of this system, which is the self-synchronization of light timing by each traffic control agent.

As light timing control in the overall traffic grid evolves in the way discussed above, a complex but definite pattern in network operation, north-south red to green transition times, emerges out of an initial random light pattern [4]. The emergent behavior of the traffic grid cannot be explained through an understanding of the behavior of each agent alone. Understanding only comes when we study the interactions of the traffic control agents as they adapt their behaviors in response to perceived information about incoming traffic flow, achieving self-synchronization of all traffic light timing in the network.

The desired emergent behavior for the traffic control system was achieved by experimenting with the control system rules and gains that change the light timing of an agent in relation to his neighbors, reducing local wait time. With no control, the traffic light agents operate independently and average vehicle waiting time is high. With very strong control, the traffic control network operation appears chaotic (no self synchronization occurs) and the average vehicle waiting time is high. When the control is just right, the emergent behavior discussed above is observed and the average vehicle waiting time in the network is minimized. It is also interesting to note that the overall system never reaches steady state operation; indeed, the system seems to be in a constant state of flux as observed by the random appearance of the light timing control signals, even when the average vehicle waiting time is being minimized. Operating far from equilibrium seems to be a common feature of CAS [1,9,10].

3.1 Traffic Control Agent

Figure 3. Traffic control agent context diagram.

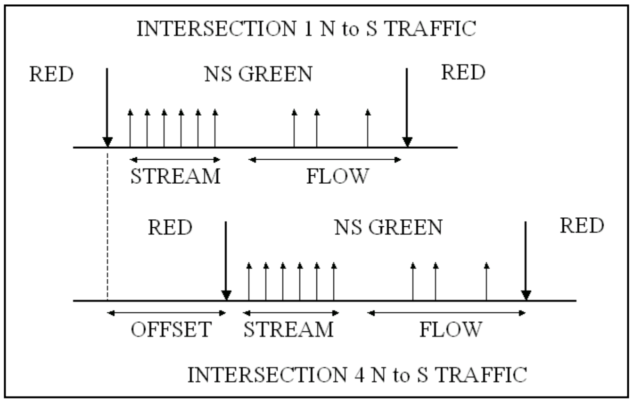


Figure 4. Perception of traffic flow through sensors.

Each traffic control agent in the network has a context diagram as shown in figure 3. Traffic flows in four directions: north to south (NS), south to north (SN), east to west (EW), and west to east (WE). A traffic control agent perceives the nature of the flow in each direction and translates these perceptions into condition attributes that characterize the flow relative to the goals of the agent.

Each direction of traffic flow through an intersection has a sensor that emits a signal when a vehicle passes over it. In figure 4, when the traffic light signal goes from red to green, the vehicles begin to move into the intersection. The pattern of signals that indicates that vehicles were waiting for the light to change is called a "Stream." One can see that a stream consists of a sequence of evenly spaced signals that are close together and follow immediately after a light change. The pattern of signals that indicate that vehicles moved through the intersection without waiting is called a "Flow." A flow consists of signals that are not part of a stream.

The Goal of a traffic control agent is to minimize the number of vehicles waiting for the light to change. The degree of goal satisfaction is directly proportional to the number of vehicles in the flow minus the number of vehicles in the stream for each direction of traffic flow through the intersection. A hierarchical block "Stream" located in the vehicle motion process of the traffic grid simulation diagram, shown in figure 7, computes the stream and flow condition attributes used for fuzzy control. For the NS direction these attributes are:

StrmCntNS,
StrmCntDiffNS,
FlwCntNS, and

FlwCntDiffNS.

All 12 such attributes are placed in a global memory location indexed by intersection number, and the traffic agent uses them to make light timing decisions each cycle. Global memory is only one way that components of the traffic controller agent can share information. An agent component can also send messages to the other components using an OpEMCSS block.

There are three aspects of intersection light timing that can be controlled by a traffic control agent. These are cycle, split, and offset. The cycle time is a complete green-red period for the NS and SN directions of traffic flow, and it is also a complete red-green period for the EW and WE directions of traffic flow. A cycle begins when the traffic light goes from red to green for the NS and SN directions of traffic flow. The split is the percent of NS/SN green time during a cycle. The offset is the difference-in-begin-cycle times between one intersection and any other intersection in the network. The offset between intersections 1 and 4 in the traffic grid simulation is shown in figure 4.

3.2 Fuzzy Control

The strategy used to reduce vehicle-waiting time for a direction of traffic flow is to decrease the cycle time by the amount of the offset in that direction. As shown in figure 4, if the stream is large during the current cycle then a reduction of the cycle time during next cycle will cause more vehicles to enter the flow. This is because the start of the cycle is moved back in time, allowing more vehicles to go through the intersection without waiting.

For each direction of traffic flow, the following calculation is performed, using the NS direction as an example:

AvgStrmCntNS = (StrmCntNS + StrmCntDiffNS) / 2

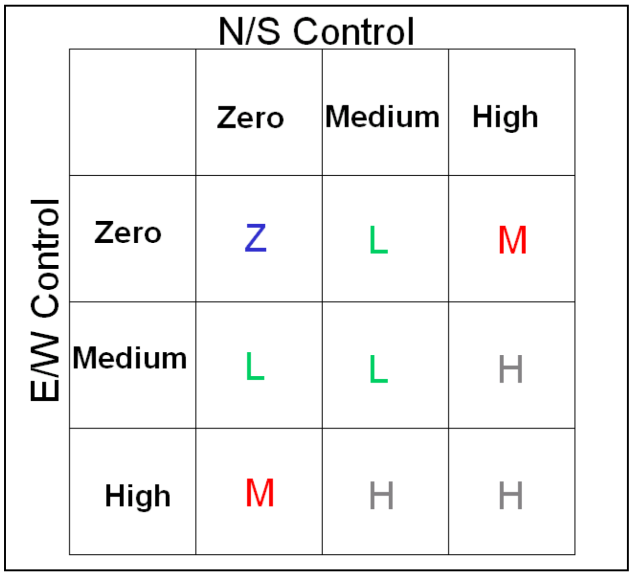
It is common knowledge among control engineers that using differentials in a control equation makes the system more responsive. The StrmCntDiffNS is the first difference, based on StrmCntNS, that provides some information about the rate of change of StrmCntNS. Indeed, the reduction of wait time responds faster to control signals when the first difference is present.

Figure 5. Condition and action fuzzy sets.

A sum of the average NS plus SN directions is used for one rule condition and a sum of the average EW plus WE directions is used for the other rule condition. The fuzzy sets defined for the N/S control condition and the E/W control condition are shown in figure 5.

Figure 6. Fuzzy control rules used to compute offset.

The fuzzy control rules used to compute offset are shown in figure 6. Suppose that N/S control condition attribute has a value of 1.5 and the E/W control condition


N/S Control

E/W Control	Zero	Medium	High
Zero	Z	L	M
Medium	L	L	H
High	M	H	H

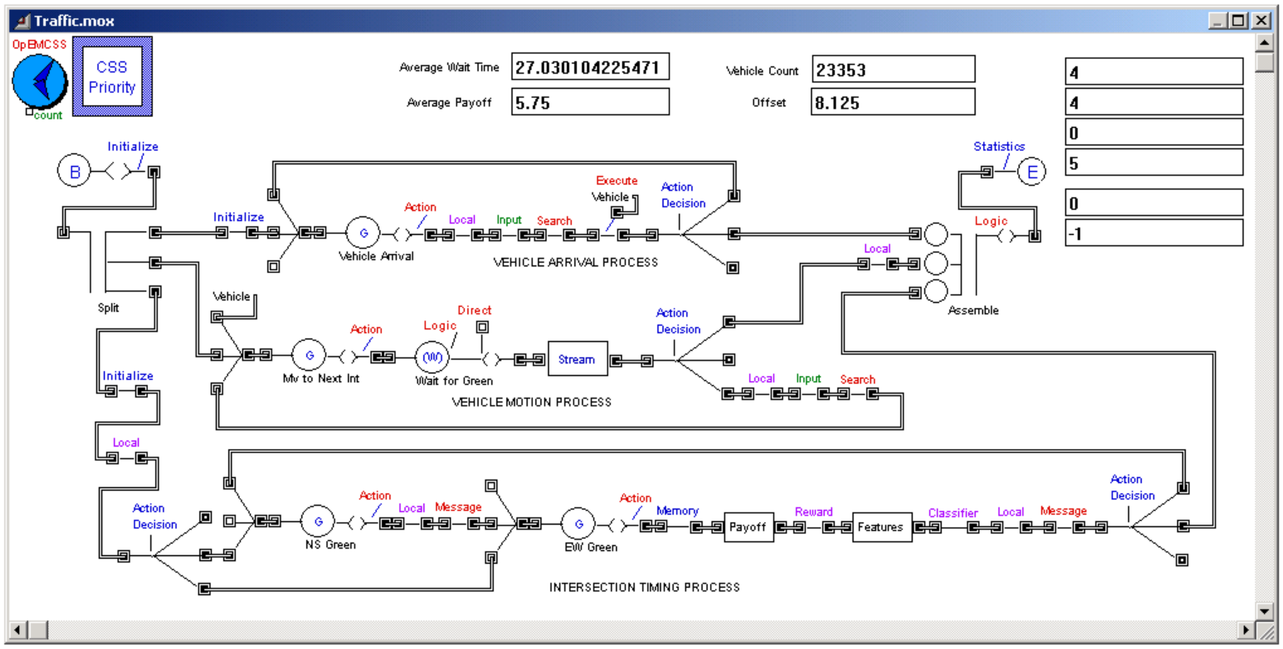
attribute has the same value. Given these values, fuzzy condition sets Zero and Medium have equal confidence, and four fuzzy rules are eligible. These rules are: zero & zero => zero, medium & zero => low, zero & medium => low, and medium & medium => low.

Each of these four eligible rules implies a separate offset value; therefore, in order to produce a single value, these values are combined using a weighted average as discussed above. In summary, the fuzzy control transforms continuous condition attributes into a continuous action attribute (offset) using discrete rules.

3.3 Simulation of a Vehicle Traffic Control Network

The vehicle traffic control network scenario consists of nine intersections, each intersection having a traffic control agent as discussed above. There are twelve streets where vehicles enter and leave the network system. Vehicles that enter the network via one of these streets have an exponentially distributed inter-arrival time. Therefore, changing the offset to reduce vehicle-waiting time for vehicles that enter the network has no effect. Thus, feature facts describing traffic flows from outside of the system are excluded from the fuzzy rule conditions.

The OpEM directed graph model shown in figure 7 consists of three concurrent, sub-processes. The top sub-process is a generator process that models vehicles arriving from outside the system, and there is one vehicle arrival process instance for each of the 12 streets discussed above. The middle sub-process models the motion of vehicles through the network, and there is one vehicle motion process instance for each vehicle moving through the network. The bottom sub-process models the light timing of the nine intersections, and there is one intersection timing process instance for each of the nine intersections in the network.



Average Wait Time 27.030104225471

Average Payoff 5.75

Vehicle Count 23353

Offset 8.125

4

4

0

5

0

-1

Figure 7. Directed graph model of vehicle traffic network operation.

In the vehicle arrival sub-process and vehicle motion sub-process there are two Input Event Action blocks that define the path that each vehicle takes through network. When a simulation run is begun, this block inputs "Network.txt" that is a text file that contains path information for vehicles traversing each of the 12 streets.

As a vehicle moves through the network, it must determine the red or green state of the traffic light that it is approaching. A Context-Sensitive Event Action block uses attribute "IntNumber," that specifies the next intersection in the path, to obtain attribute "Light" from the appropriate intersection timing process instance. This block is used in the vehicle arrival sub-process to obtain the light timing state of the first intersection and in the vehicle motion sub-process to determine the light timing state for subsequent intersections along the vehicle path.

When an intersection traffic light changes state, a Message Event Action block sends the attribute "Light" to any vehicle approaching the intersection. There are two of these blocks, one for each light change. Only red and green light timing states are modeled to simplify the simulation. This simulation does not require detailed vehicle motion in order to study the emergent behavior of the traffic control network.

The intersection timing sub-process includes two hierarchical blocks that contain detailed models. The "Payoff" hierarchical block computes the flow count minus the stream count for each direction of traffic flow and adds these up to produce a single measure of performance or measure of fitness [1] for an intersection. This block also calculates an average computed over all intersections, and this average is reset every 10 minutes of simulated time. The "Feature" hierarchical block computes the NS and EW control condition attributes as discussed above. The Classifier Event Action block uses these condition attributes in fuzzy rules to compute offset.

The Memory Event Action block in the intersection timing sub-process obtains the traffic flow feature facts for an intersection. Global memory is used for the agent sensor component to communicate with the light timing control component.

3.4 Results of Simulation Runs

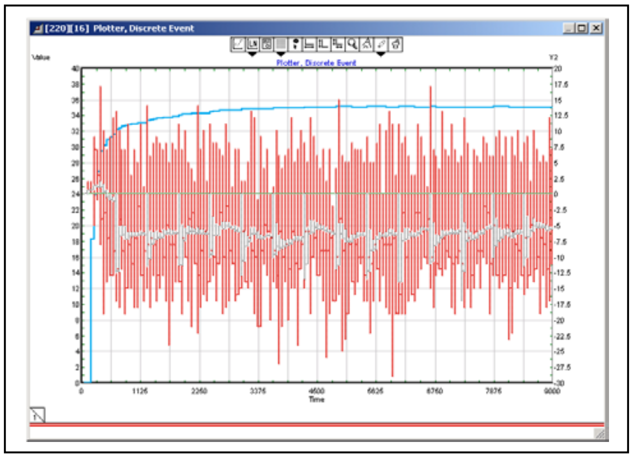


Figure 8. Plot of several simulation output attributes as a function of simulated time, control gain is zero.

Figure 8 shows a plot of several simulation output attributes as a function of simulated time. The control gain was set to zero so that offset was always zero. The top trace that slowly converges is the average vehicle waiting time that is computed by the Wait Until Event block located in the vehicle motion sub-process. The random looking trace is the output of the Payoff hierarchical block, and it shows the uncontrolled variation in the performance of the traffic control agents to reduce vehicle-waiting time. The light trace in the center is the average intersection performance reset every 10 minutes of simulated time. The spikes in average performance, that equal zero, show where each reset occurs. When the intersections are operating independently, the average intersection performance stays around minus 6 as shown in figure 8

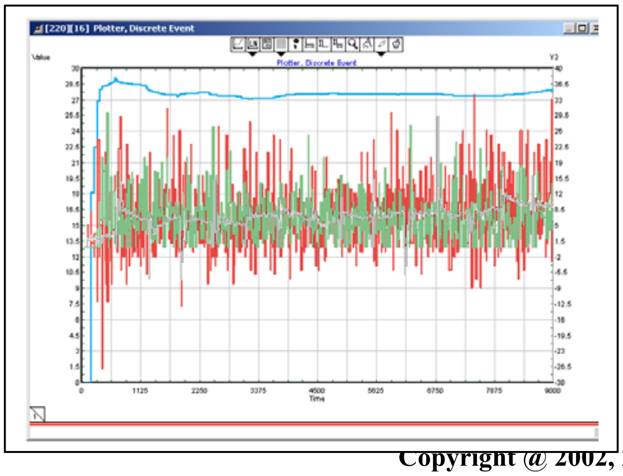


Figure 9. Plot of several simulation output attributes as a function of simulated time, control gain is two.

Figure 9 shows a plot of the simulation output attributes as a function of simulated time. The control gain was set to two. There are now two random looking traces. The smaller random trace trace is the offset for each intersection when a cycle begins. The average intersection performance (light trace in center) starts at around plus 6, after about ten minutes of adjustment time, and slowly increases to plus 7.5 at the end of the run. The light trace in figure 8 is flat for no control, but in figure 9 the light trace shows that many more vehicles are flowing through the intersection without waiting.

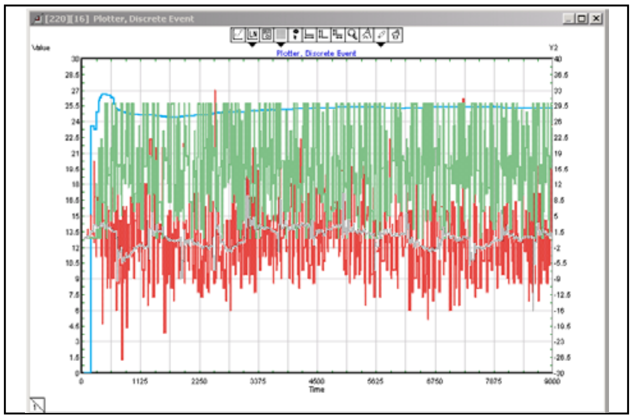


Figure 10. Plot of several simulation output attributes as a function of simulated time, control gain is four.

For a gain of four, shown in figure 10, the offset trace is near the limit of 30 much of the time. The average intersection performance varies around one indicating that the controller is still working but degraded from the performance achieved when the control gain was equal to two. In general, the range of the payoff trace is decreased and the light trace of the average intersection performance increases when the gain is two. This indicates that the fuzzy controller is having a positive affect on the system.

4. Summary

Each intersection controls the offset in order to bring more vehicles into the flow. Changing the offset changes the intersection cycle in relation to all other intersections. Such operation ought to create a conflict with neighboring intersections by changing when vehicles arrive there. However, as seen in figure 9, after about 10 minutes the intersections self-synchronize and begin to cooperate with each other, reducing the overall waiting time in the network.

There are no communications links in this system so how do the agents communicate? An intersection changing its light timing cycle is perceived in the traffic flow patterns of another agent. The second agent adjusts its offset, and a third agent perceives this. It seems that perturbing the environment is transmitting information. There are no communication links among the agents yet they communicate and adapt. Each agent perceives intersection traffic flow patterns and adapts its light timing to reduce local vehicle waiting time.

The next step in this research is to have the fuzzy controller learn the best control rules. In this paper the rules were discovered manually, but are these the best rules? The classifier block currently has the ability to learn crisp rules based on graded-learning where each rule decision receives a payoff based on the effectiveness of the decision. In supervisory learning, used by most neural networks, the correct decision is used to train the network. Graded learning is a much more difficult rule-learning problem.

5. References

[1] Casti, J. Would-Be Worlds; Wiley: New York, 1997.

[2] Clymer, J. R., Systems Analysis Using Simulation and Markov Models, Englewood Cliffs, NJ: Prentice-Hall Inc, 1990.

[3] Clymer, J. R., Corey, P. D., and J. Gardner, "Discrete Event Fuzzy Airport Control", IN IEEE Transactions on Systems, Man, and Cybernetics, Volume 22, Number 2, March-April 1992, pages 343-351.

[4] Clymer, J. R., "Expansionist/Context-Sensitive Methodology: Engineering of Complex Adaptive Systems", IN IEEE Transactions on Aerospace and Electronic Systems, April 1997 issue, Volume 33, Number 2, pages 686-695.

[5] Clymer, J.R., "Simulation-Based Engineering of Complex Adaptive Systems," In Simulation, Journal of the Society of Computer Simulation, San Diego, CA, Volume 72, Number 4, April 1999 issue, pages 250-260.

[6] Clymer, J.R., "Optimization of Simulated Systems Effectiveness using Evolutionary Algorithms," IN Simulation, Journal of the Society of Computer Simulation, San Diego, CA, Volume 73, Number 6, December 1999, pages 334-340.

[7] Clymer, J.R., "Optimizing Production Work Flow Using OpEMCSS," IN Proceedings of the 2000 Winter Simulation Conference, J.A. Joines, R.R. Barton, K. Kang, and P.A. Fishwich, Editors, Orlando, FL, December 10-13, 2000, pages 1305-1314.

[8] Clymer, J.R., and D.J. Cheng, "Simulation-Based Engineering of Complex Adaptive Systems Using a Classifier Block," The 34th Annual Simulation Symposium, Seattle, WA, April 22-26, 2001, pages 243-250.

[9] Clymer, J.R., Simulation-Based Engineering of Complex Systems, John R. Clymer & Associates: Placentia, CA, 2001.

[10] Holland, J.H., Hidden Order, Addison-Wesley, 1995.

[11] Solow, D., "On the Challenge of Developing a Formal Mathematical Theory for Establishing Emergence in Complex Systems," In Complexity, New York, NY: John Wiley & Sons, Inc., Volume 6, Number 1, September-October 2000, Pages 49-52.

[12] Weiss, G., editor, MultiAgent Systems: A Modern Approach to Distributed Artificial Intelligence, Cambridge, MA : The MIT Press, 1999.