Knowledge Based Hybrid Systems (B-11)
I. Mezgár and P. Bertók (Editors)
Elsevier Science Publishers B.V. (North-Holland)

Exploiting Domain Knowledge, Neural Networks and Genetic Algorithms to Harvest Traffic Simulation Results

Mark Foy and Carl Uhrik

Laboratorio di Ingegneria Informatica, Dipartimento di Informatica e Studi Aziendali,
Università degli Studi di Trento, 38050 Mesiano-TN ITALY
e-mail: mfoy@itnsun4.cineca.it and carl@itnsun4.cineca.it

Abstract

This paper discusses the development of an incremental hybrid simulation-learning system that can be applied to problems with (1) many features, (2) high cardinality features, and (3) many classification categories. First, this hybrid system integrates a simulation module and a previously developed learning algorithm called a genetic algorithm (GA). Secondly, this hybrid system is extended to include another previously developed learning algorithm call a neural network (NN). Lastly, the initial NN is constructed with domain knowledge, to facilitate learning in large space environments. The theory behind adding domain knowledge is that by *starting* the NN closer to what is believed to be the optimal/converged configuration, the final system will more accurately generalize unseen examples and be generally more robust.

Specifically, this investigation has its testing grounded in the domain of automobile traffic signal control because this domain offers many features (road network conditions), many classification categories (traffic signal timing strategies), and available domain knowledge that can be integrated into the construction of a NN.

This paper first presents the positive results of combining a GA and a simulation, and applying this to the problem of determining traffic signal timings. In this system, the GA has been able to converge on what a traffic engineer would regard as optimal or near-optimal solutions. Next, this framework is expanded to integrate a NN, and preliminary results of this integration are given. These early results indicate difficulties in training the currently described NN. We observed that the present domain knowledge, used in our system to help build the initial NN, did not have a large positive effect on the training of the NN under the conditions discussed in this paper.

Keyword Codes: I.2.1; I.2.6; I.6.3
Keywords: Artificial Intelligence, Applications and Expert Systems; Artificial Intelligence, Learning; Simulation and Modeling, Applications

1. INTRODUCTION

Automobile traffic signal control is a large, non-Boolean, erratic, multi-modal (i.e., two or more optimal solutions may exist for a specific traffic situation), non-linear, gray, and hard-to-predict domain. Due to these characteristics, many traditional control strategies are not sufficient to handle the task of configuring traffic signals to allow for the efficient flow of cars. Specifically, since closed-form analytical solutions for calculating signal timing configurations are not available (i.e., they are infeasible), simulation is often used in traffic engineering systems. Additionally, because the traffic signal control domain is large (i.e., there are many possible ways in which to configure a network of traffic signals), it is desirable to have an intelligent system specify the timings for traffic signals.

This paper first presents a traffic simulation implemented within a genetic algorithm (GA), an intelligent search algorithm that searches by manipulating populations of structures (i.e.,

binary strings representing data structures which symbolize possible solutions to a problem) into new populations using operators patterned after natural genetic operators [Holland, 1975]. A GA offers a good way of searching the space of possible traffic signal timing strategies because it minimizes the solution space points it needs to evaluate by intelligently searching this space. The GA implemented in this paper uses the simulation module repeatedly to evaluate different signal timing strategies, eventually choosing a near-optimal signal timing strategy for the given traffic conditions.

Even though the GA searches intelligently, it still needs to evaluate many points in the solution space. The time to perform one complete GA run with multiple simulations can be prohibitive, therefore, although it is not necessary, it is desirable to implement a system which could learn how the GA "acts". This system should take the same input as the GA, and predict how the GA would respond to this input. A neural network (NN) has this ability because it learns by iterating over a given set of "examples" (i.e., input-output or condition-action pairs). A NN can also predict output for examples it has not seen by extrapolating and interpolating over the examples which it has "seen" (i.e., the examples that were used to train the NN).

Furthermore, by integrating a NN into this hybrid GA/simulation system, traffic engineers could add partial or incomplete domain knowledge. Domain knowledge is desirable because it could make the NN easier to train. Moreover, it would seem that the integration of domain knowledge could not hinder that NN because the NN can use approximate domain knowledge and then modify the constraints of this domain knowledge as it sees fit. Therefore, this strategy is mostly designed to give the NN a "head start" on learning. The initial idea of employing domain knowledge was taken from concepts presented in [Towell et al., 1990], [Shavlik and Towell, 1989], and [Katz, 1989].

This paper first outlines the reasons for implementing this system in the traffic domain. Next, a simple example that specifically addresses traffic signal control is described. This example will be used throughout this paper. This will lead to a discussion of the overall framework and a specific description of the systems contained in this framework. First, the utilization of a GA with a traffic simulation will be discussed, followed by the results of applying this GA/simulation hybrid system to the simple traffic control situation. Second, the motivation for applying a NN will be presented, followed by a discussion of how domain knowledge was included in this framework. This will lead to an analysis of this integration, examining some of the difficulties of using NNs and domain knowledge together. Specifically, explanations to the difficulties encountered during testing will be suggested, as well as proposing fixes to these difficulties .

2. WHY A TRAFFIC DOMAIN?

The state of traffic control today is in need of an intuitive, robust system to continually optimize traffic signal timings. Intelligent computer traffic control systems are needed to dynamically handle changing traffic conditions.

In standard pre-timed controllers, traffic signal timings are fixed at what is determined to be the most effective timing strategy. Timing determination involves either extensive analysis of traffic data or after short observations of traffic trends, making it fairly time consuming task. Due to the time constraints, timing determinations are done infrequently, making the pre-timed control method a static model. Therefore, with this method, signal cycle times and offset times are calculated once, for current conditions, and are then set into the individual traffic signals for an extended period of time (i.e., months). The signal cycle timings do not change with demand. This static characteristic is a clear disadvantage, motivating the development of more dynamic methods.

The hybrid framework described here is a more dynamic method and its development is encouraged by the lack of traffic control systems that do intelligent, dynamic control of traffic signals.

3. THE TRAFFIC SIGNAL CONTROL DOMAIN

The problem addressed in this study entails finding a near-optimal traffic signal timing configuration at all intersections of a street network given the current intersection

characteristics. The current characteristics consist of: (1) the current number of cars at each lane of each intersection and (2) the external arrival volumes. The system should produce a near-optimal timing configuration for north/south green phase and east/west green phase for the current conditions for all the intersections in the network. First, the inputs and outputs of the hybrid system described in this paper will be defined.

There are many ways in which traffic control can be viewed. To facilitate understanding, a typical traffic situation is constructed that is both manageable and comprehensible. The street network has 4 intersections shaped in a square/circle configuration, each intersection adjacently connected to two other intersections by perpendicular roadways (see Figure 1). This lets us define traffic control as a function of two input vectors:

(input.1) $= [n_{111} \cdots n_{ijk} \cdots n_{443}]$ (1)

where n_{ijk} = number of cars on lane k of approach j of intersection i.

For the example in this paper, i=1 to 4, j=1 to 4, k=1 to 3, and total length is 48.

(input.2) $= [v_{11} \cdots v_{ij} \cdots v_{44}]$ (2)

where v_{ij} = external arrival volume on approach j of intersection i.

For the example in this paper, i=1 to 4 , j=1 to 4 , and total length is 16 (note, in this example, there are only 8 locations where cars can enter the network, therefore the other 8 approaches have an external arrival volume of zero and were removed).

where the result of evaluating these two vectors through our system is three output vectors:

(output.1) = [tgt] (3)

where tgt = total green time given to each intersection for one full cycle.

The same total green time is used for all intersections in the current system, but their is no reason this can not be changed.

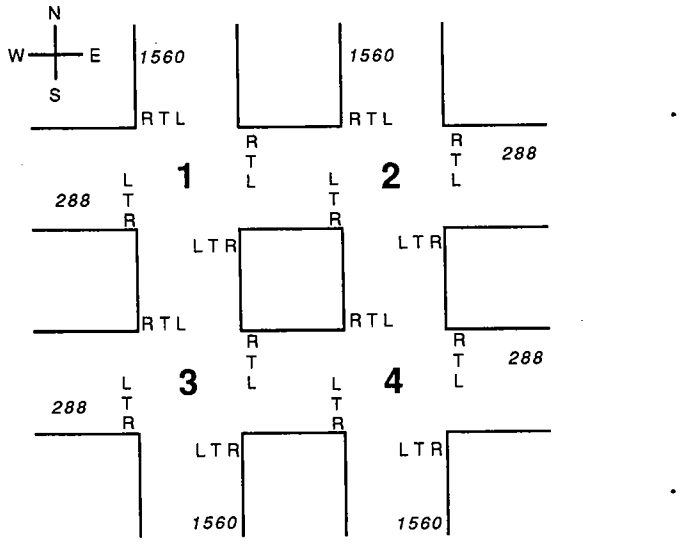

Figure 1 - Street Network Configuration - intersections numbered from 1 to 4 - L=turning Left R=turning Right T=going straight Through - numbers appearing at external components are the external arrival volumes in cars per hour used for the example GA run described later

(output.2) $= [d_1 \ . \ . \ . \ d_i \ . \ . \ . \ d_4]$ (4)

where d_i = direction in which the first green phase will allow traffic to flow at intersection i, either north and south or east and west.
For the example in this paper, i=1 to 4.

(output.3) $= [nsgt_1 \ . \ . \ . \ nsgt_i \ . \ . \ . \ nsgt_4]$ (5)

where $nsgt_i$ = proportion of total green time (output.1) that will be allocated for the north/south green phase at intersection i.
For the example in this paper, i=1 to 4.

This view, considered in a typical setting, results in many high cardinality feature variables and many classification categories. For this example, the feature space size (i.e., the number of possible distinct inputs) is approximately 10^{92}. Likewise, the classification space size (i.e., the number of possible distinct outputs) is approximately 10^7. This demonstrates that this domain, and certainly many more domains, could benefit from frameworks that operate efficiently in large space problems.

Due to these large spaces, many algorithms have been ruled out because they do not have the capability to handle this type of problem. It is difficult to envision ID3 [Quinlan, 1983] (and many other induction algorithms) classifying 10^{92} different instances into more than 10^7 categories, though it may be possible. In our framework a GA is implemented first so that our system could take advantage of GAs' global, parallel search characteristics and their black box evaluation strategy. Secondly, NN are considered because they may have the right types of characteristics to attack large space problems and because they could be used to supplement the GA.

4. OVERALL FRAMEWORK

4.1. Genetic Algorithm (GA)

GAs are useful because they can operate in environments with no domain knowledge and no care about the size of the instance space. They also can be very effective at finding optimal or near-optimal solutions to dynamic real-world problems [Goldberg, 1989] [PICGA, 1985] [PICGA, 1987] [PICGA, 1989].

Our initial framework first employs a GA to find "near-optimal" signal timing control strategies for "typical" traffic situations. The GA bases its selection of a "near-optimal" solution on the "total average wait time per car", trying to minimize the amount of time cars have to wait. The GA is given the two input vectors defined above. These inputs entirely describe one traffic situation. The GA is initiated, and by using the given traffic simulation module to evaluate points in the solution space, it converges on a "near-optimal" solution and outputs this solution. This "near-optimal" solution consists of timings for north/south green phase and east/west green phase for all intersections in the traffic network. Further details regarding the structure of this "Traffic GA" can be found in [Foy et al., 1992].

4.2. GA Results

The results of applying a GA to this traffic control problem have been positive. The Traffic GA produced very reasonable traffic signal timing plans for many different traffic conditions. One typical Traffic GA run is described below.

4.2.1. Traffic Environment

To illustrate the performance of the GA in this framework, a simple traffic situation was constructed. First, the (input.1) vector (i.e., the number of cars at all locations), was initialized with typical numbers. Next, the (input.2) vector (i.e., the external arrival volumes) was set to the values corresponding to the volumes shown in Table 1 (and also in Figure 1). The north and south approaches were given 5 to 6 times as much traffic as east and west directions.

4.2.2. GA Performance

To obtain a stable, unbiased, average result, five GA runs were executed. Each run was executed for 60 GA generations with a GA population of 50. This means that for each run, the traffic simulation was executed 3000 times. At each generation, the average fitness of the

Table 1 - Network Configuration

	Arrival Volume (cars/hour)
from [illegible] to Intersection 1	1560
from [illegible] to Intersection 1	288
from [illegible] to Intersection 2	1560
from [illegible] to Intersection 2	288
from [illegible] to Intersection 3	1560
from [illegible] to Intersection 3	288
from [illegible] to Intersection 4	1560
from East to Intersection 4	288

generation was calculated and the member in the population with the best fitness value (i.e., the shortest wait time) was identified. Refer to Figure 2 for a graphical illustration of how the GA starts with bad solutions (i.e., solutions that produce on-average high wait times) and locates good solutions (i.e., solutions that produce on-average short wait times). To produce Figure 2, the average fitness of each GA generation (from generation 0 to 60) from each of the 5 Traffic GA runs were averaged together. This produced the "Average Wait Time of Population" curve. The "Member With Minimum Wait Time" curve was produced in the same manner by averaging the fitness of the best-of-generation members for each of the 5 Traffic GA runs.

Figure 2 shows that the population seems to converge to the optimum or near-optimum member by the 20th or 30th generation. Therefore, it is possible to terminate the GA after 20 generations, instead of after 60 generations, and still obtain a near-optimal solution. This reduced generation scenario would reduce the number of simulations from 3000 to 1000. The graph also shows that typical minimum wait time values were around 40 seconds for these traffic conditions. After the last generation, which in this case was the 60th generation, the member with the maximum fitness (minimum wait time) can be selected as the best signal timing configuration and called the solution from the Traffic GA.

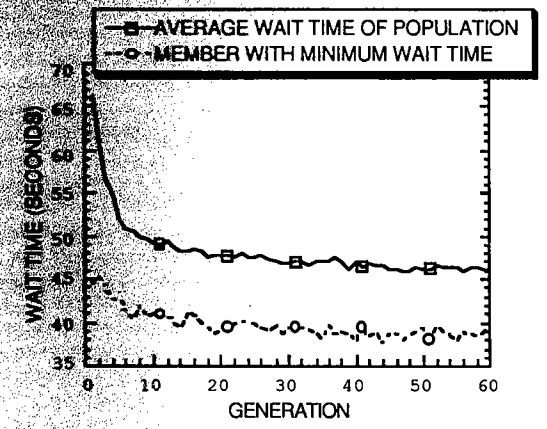

Figure 2 - The Average of 5 Traffic GA Runs - showing generation average (average wait time) and best-of-generation (minimum wait time) results

Table 2 - The GA's Maximum Fitness (Minimum Wait Time) Timing Strategy

Intersection Number	First Direction	Green Time (sec)	Second Direction	Green Time (sec)	Total Cycle Time (sec)[a]
1	E/W	12	N/S	42	60
2	N/S	36	E/W	18	60
3	N/S	42	E/W	12	60
4	E/W	9	N/S	45	60

[a] Total Cycle Times have two yellow phases - of 3 seconds each - added, in addition to the two green times.

4.2.3. GA Output

A typical maximal fitness member, actually found by one of the GA runs executed on the traffic environment described in Table 1, is shown in Table 2. Note that for all intersections, the green phase time for the north/south (N/S) directions was considerably longer than for the east/west (E/W) directions. Observe that total cycle times are equal because this constraint was put on the GA. Additionally, it should be pointed out that the Traffic GA selected a total cycle time of 60 by itself, this number is not programmed into the GA. Also recognize that the GA found a strategy that utilized very similar green phase times for the north/south directions and also for the east/west directions. We expect this behavior because similar green phase times often allow the best flow of traffic because cars can move through the network with fewer stops if there is some type of synchronized cycle time [Reljic, 1988]. Next, notice that the GA could have given green phase times up to 114 seconds, but only went as high as 45 seconds. This is because the GA was searching for a strategy that would *minimize* wait time, and if it were to allocate more green phase time to north and south directions, the wait times for cars coming from the east and west would increase too dramatically to make this a beneficial action. Lastly, we were impressed with this solution because it provides two 33 second green bands, one for northbound traffic through intersections 1 and 3, and another for southbound traffic through intersections 2 and 4. These long north/south green bands are desirable because they allow smoother traffic flow on these major north/south arterials. Overall, the hybrid system with the GA and a simulation module has performed very well. Next, we will expand this framework to integrate a NN which will be designed to generalize GA results.

4.3. Neural Network (NN)

4.3.1. Motivation for Integrating a NN

There is one primary motivation for using a NN in addition to a GA. In problems with large hypothesis (or classification) spaces like traffic control, even tough GA search is more efficient than traditional search methods, the run time of the GA can become prohibitively long due to the simulation module. If every time an optimization of a specific traffic situation was desired, the GA had to be rerun, then execution times would become too long and the GA would not be practical. A NN has the potential to addresses this issue.

To reduce repetitive computations, a NN could be used to "learn" or generalize the GA's behavior (i.e., how it reacts to input). If a NN knew how the GA reacted to input, the GA would not have to be executed every time a new near-optimal signal timing strategy was needed for a new traffic situation. To "teach" a NN how to "predict" what output the GA will produce given certain input, a training set of condition-action pairs (i.e., input-output pairs) is needed. These condition-action pairs can be collected from multiple GA runs, where for each GA run, different inputs (i.e., traffic conditions) are given. A set of GA condition-action pairs will be used as the training set for a NN. Figure 3 illustrates the framework for augmenting the GA with a NN.

4.3.2. Format of NN Input and Output

The format of the input and the output from the GA is simple and is easily manipulated to work as training instances for the NN. Combining the input to the GA with the corresponding

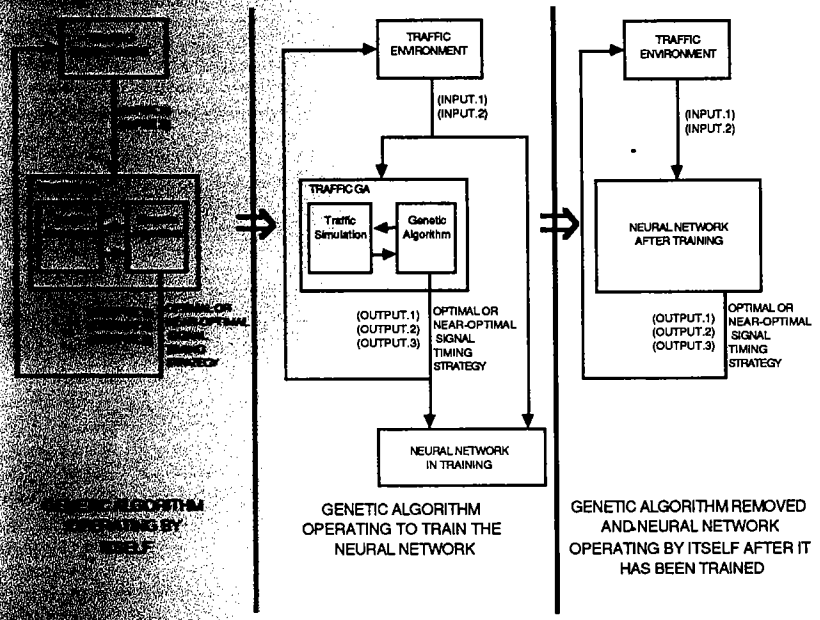

Figure 3 - Framework for Combining the GA with a NN

condition-action pair is formed with the following format:
- the inputs, or the traffic situation (taken directly from (input.1) and (input.2))
  - x1=cars turning *left* from the *north* side of intersection *1*
  - x2=cars going *straight* from the *north* side of intersection *1*
  - x3=cars going *straight* from the *west* side of intersection *4*
  - x4=cars turning *right* from the *west* side of intersection *4*
  - x5=external arrival volume from the *north* to intersection *1*
  - x6=external arrival volume from the *west* to intersection *1*
  - x7=external arrival volume from the *north* to intersection *2*
  - x8=external arrival volume from the *east* to intersection *4*
- the outputs, or the optimal signal timing strategy (taken directly from (output.1), (output.2), and (output.3))
  - y1=the total green time given to each intersection for one full cycle
  - y2=the direction in which the first green phase will allow traffic to flow at intersection *1*
  - y3=the direction in which the first green phase will allow traffic to flow at intersection *2*
  - y4=the direction in which the first green phase will allow traffic to flow at intersection *3*
  - y5=the direction in which the first green phase will allow traffic to flow at intersection *4*
  - y6=the proportion of total green time that will be allocated for the north/south green phase at intersection *1*
  - y8=the proportion of total green time that will be allocated for the north/south green phase at intersection *4*.

The GA input features were actually combined or reduced for input to the NN. The NN only receives 12 combined features that represent the 56 features described above. We have made an assumption (which may not be correct) that this reduction would not significantly affect training because the combined features are believed to have the same effects on the output. This was done to reduce the number of distinct inputs into the NN in the hopes of making it simpler to train.

4.3.3. Establishing the Structure of the NN Using Domain Knowledge

To help the NN more easily learn how to predict GA output, the idea of including traffic engineering domain knowledge into the construction of the NN was considered. This idea, which was first discussed in [Towell et al., 1990], integrates domain knowledge into the NN by converting if-then clauses into structural network characteristics. This may include which nodes should be connected as well as what weights should be initially established on some of those connections. After domain knowledge is integrated into the NN, the NN is trained as usual, adjusting connection weights as needed. The only exception to this is if some of the connection weights are constrained such that they are fixed and can not be changed during training. Two other similar approaches exist in the open literature. First, a paper by [Shavlik and Towell, 1989] that presents an approach very similar to the one adopted here, and second, a paper by [Katz, 1989] that focuses on improving learning time rather than improving classification.

In this paper, traffic engineering domain knowledge is used to construct the NN shown in Figure 4. This configuration was formed by using the guidelines given in [Towell et al. 1990] regarding the connecting nodes, the initializing connection weights, and the setting of nodal biases. Refer to the original paper for more information on determining exactly how the domain knowledge is used.

For this paper, the domain knowledge rules come in a form similar to the following:

1) IF [[the average number of cars coming from the north is large] AND [the average number of cars coming from the south is large]] THEN [there are many cars coming from the north/south]

2) IF [there are many cars coming from the north/south] THEN [[the first direction to travel at inter 1 should be north/south] AND [the proportion of total green phase time given to the north/south green phase at inter 1 should be long]]

These vague, general rules then construct the following connections in the NN (see Figure 4):

from rule 1) [node 0 to node 12 with weight near 3.0; node 2 to node 12 with weight near 3.0]

from rule 2) [node 12 to node 23 with weight near 3.0; node 12 to node 24 with weight near 3.0]

Additionally, after considering these rules, the biases are established based on information in [Towell et al. 1990]. The rest of the weights and biases are established similarly, and the network in Figure 4 is produced.

These rules are of course only approximately correct, so it is important that the system be allowed to and able to adjust most NN weights to modify incorrect rules/theories.

Once the structure of the NN has been established using the domain knowledge, the NN is ready to be trained using the condition-action pairs obtained from multiple GA runs. The next section will discuss the initial results observed from using a NN in this hybrid system.

4.4. NN Testing

Here we present the NN training and testing scenario. Some initial results are given, followed by a discussion of why some of the results are counter intuitive.

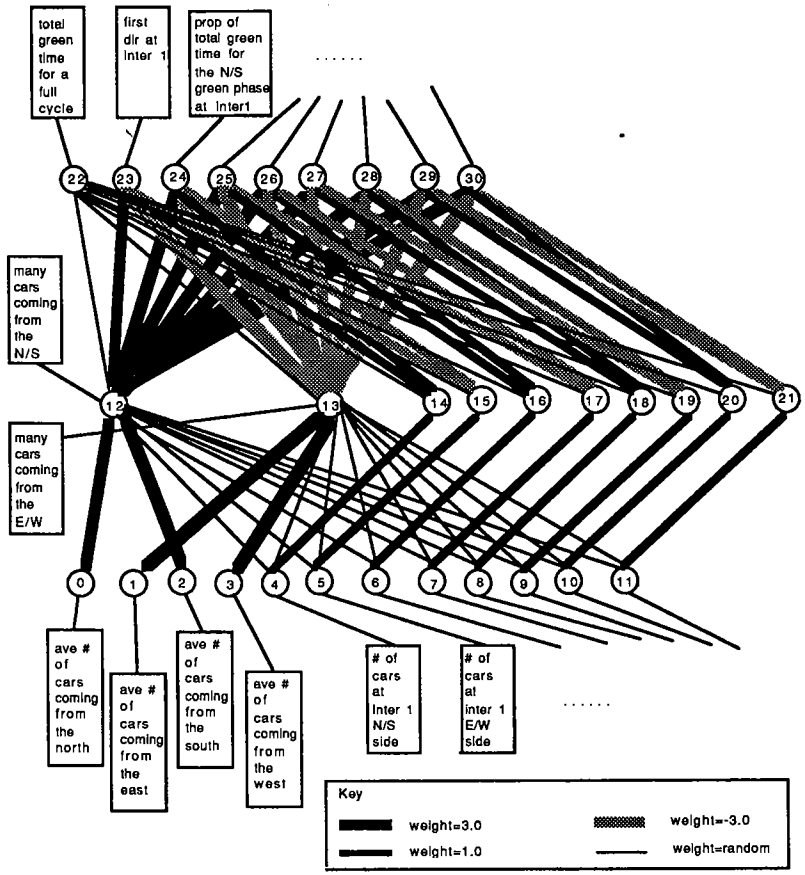

Figure 4 - The NN Constructed with Domain Knowledge - input nodes (at the bottom) represent the traffic situation and output nodes (at the top) represent the optimal signal timing strategy

4.4.1. NN Training Results

Four different implementations (or NN configurations) were constructed and used to test the usefulness of blending domain knowledge into a NN. The commonalties between configurations include: (1) all have 31 nodes, (2) all have 12 inputs and 9 outputs, and (3) all use back propagation for learning after initialization (whether the initialization is random or controlled by domain knowledge). The four network configurations follow.

(Net Config 1)=(No domain knowledge used) A network with complete interconnection between the input and hidden layers and between the hidden and output layers. Initialized with all random connection weights. No nodal biases.

(Net Config 2)=(Domain knowledge used to establish structure of NN) A network with reduced interconnection as shown in Figure 4. Initialized with all random connection weights except the weights on the connections between nodes 4

through 11 to nodes 14 through 21. These special connections were fixed at 1.0. Random biases on nodes 12 and 13 and all output nodes.

(Net Config 3)=(Domain knowledge used to establish structure of NN and weights) A network identical to Figure 4. Connection weights initialized as shown in Figure 4. All weights could change during back propagation except 0 to 12, 1 to 13, 2 to 12, and 3 to 13, which were fixed to values within 0.01 of 3.0 based on the information gained from [Towell et al. 1990]. The other weights that could not change were between nodes 4 through 11 connected to 14 through 21, which were fixed at 1.0. All weights between the hidden and output layers were preset to within 0.01 of 3.0 (or -3.0 for negated antecedents), but were allowed to change during training. Biases were fixed (without being allowed to change) at values determined by the scheme discussed in [Towell et al. 1990].

(Net Config 4)=(Domain knowledge used to establish structure of NN and weights, but connectivity reduced even more) Same as Net Config 3 (similar to Figure 4) except connectivity is reduced even further by eliminating all connections between nodes 4 through 11 and nodes 12 and 13.

The training set consisted of 22 examples (i.e., condition-action pairs), and all networks were trained 10 separate times with different random seeds (thereby changing all of the initial connection weights). Training targeted a total sum of squared error (TSS) (a common error measuring value for NNs) of 16.0. Both Net Config 1 and Net Config 2 achieved a TSS value of 16.0, but Net Config 3 and Net Config 4 could not achieve this error level. The average number of training epochs required by the four network configurations were respectively: 46, 158, >20,000, and >20,000. Both Net Config 3 and Net Config 4 could only realize average TSS values of 19.88 and 25.41, respectively, after 20,000 training epochs. These results seem to show that networks prefer total connectivity to restricted connectivity.

Three test sets were then used to check the generalization abilities of the four network configurations. Each test set contained 10 examples. Test Set 1 consisted of traffic configurations that were very similar to those used in the training set. Test Set 2 encompassed fairly random traffic situations, most of which are uncommon, and Test Set 3 contained typical, more conventional traffic situations. All examples within the test sets, as well as the training set, were generated by the GA. The average results obtained from these test sets are shown in Table 3.

This table shows that as the test sets become further removed from the training set, errors increase. Likewise, as the networks become more restricted by domain knowledge, errors generally increase, except for the case "Test Set 3 with Net Config 4" in which general traffic domain knowledge is applied to typical traffic situations. Some of these results are unexpected, and in the next section an analysis of some of these counter intuitive results, as well possible explanations for them, will be given.

4.4.2. Analysis of NN Results

The results of applying a NN, trained with sparse data, to generalize how a GA reacts to input has not been as successful (i.e., not as accurate) as using a GA by itself. Although the results above are partially disappointing, some information can be gained from examining them. First, Net Config 4 seems to handle typical traffic situations (like the type included in Test Set 3) better than the others. This is one of the few cases in which a domain knowledge structured

Table 3 - NN Testing Results
Total Sum of Squared Error (TSS) Values

Test Set	Net Config 1	Net Config 2	Net Config 3	Net Config 4
1	5.91	6.40	9.50	11.20
2	10.39	9.32	12.31	12.65
3	18.09	21.95	18.03	13.33
All Test Sets	11.46	12.56	13.28	12.39

network was able to out perform the random network with random weights (Net Config 1). The other instance where a knowledge derived network outperformed Net Config 1 was with Net Config 2 processing Test Set 2. Net Config 2 was the best at classifying situations similar to the cases in the training set (examples in Test Set 2). All other comparisons between Net Config 1 and the domain knowledge constructed networks indicate that Net Config 1 was equivalent, or often times better than the other networks. The question is: why did domain knowledge help in only a few cases?

4.4.2.1. Was the Instance Space Too Large and Erratic?

Generally, NNs can learn mappings regardless if they are linearly separable or not, but this space may be too large and erratic to be reasonably *learned*. Expecting a NN to learn a mapping of 10^{92} instances to 10^{7} classification with only 31 nodes may be asking too much. Additionally, most instances do not have just one optimal classification to which they can be associated. When the GA is given a traffic situation for which an optimal control strategy is need, it will produce one, of maybe many, control strategy that would work well on the situation. Therefore solutions are not distinctly right or wrong, giving this domain many gray areas. This whole question of training instances leads to the next question.

4.4.2.2. Were Too Few Training Instances Used?

There is good reason to believe that too few training examples were used. First, the fraction of all possible instances that appeared in the training set was very small (only 22 instances in the training set). This very small set is generally not considered enough to train a 31 node NN; the network is not likely to develop a good set of weights with these limited examples. Even if the network is trained for thousands of epochs, this would not help because the training set is not diverse enough and all the network can do is overfit the examples. Overfited networks cannot generalize to produce reasonable results on unseen instances. The most desirable action is to increase the number (and diversity) of examples in the training set. Some experiments have been run with a training set consisting of about 50 examples, and improved performance has been observed. This should certainly be one of the first directions for future work, to insure that the hybrid system is properly tested. Overall, it may be asking too much for a network to develop a robust set of weights when so few instances are available for training.

4.4.2.3. Was the Domain Knowledge Incorrect?

The domain knowledge used to construct the NN was taken from [Foy et al., 1992], from the results of the GA run presented above, and from intuitive notions about traffic characteristics. Although it is very likely that this domain knowledge was only approximately correct, we believed the NN could modify it as needed.

One may think that the correctness of this knowledge could be checked by exploring the NN weights after training, but this idea should be approached with caution. Some trained networks were examined as to their final weights, and it was found that sometimes very little change had taken place in the preset weights. On the other hand, other similar cases showed large preset weight changes. This is a tricky area because weight changes depend on the weights as a set, not individually. Different random starting weights (exclusive of the preset weights) affect all weight changes. Therefore, changes are hard to generalize across all networks because the random weights are generally so different. Interpreting connection weights is mentioned as a future research area in [Towell et al., 1990].

Overall, we believe the domain knowledge was approximate, but not incorrect.

4.4.2.4. Was Knowledge Used in the Wrong Way?

It is possible that knowledge was used in the wrong way to establish NN structure. Maybe it would have been better to use the knowledge to develop a more elaborate *class* hierarchy of nodes with additional intermediate nodes associated with intermediate concepts (e.g., "north/south flow is greater than east/west for entire traffic network" or "intersection 1 is at oversaturation"). The current structure may have been incongruent with the best learning conditions, thereby resulting in sub-optimal learning.

5. CONCLUSIONS

This paper presented a hybrid system that was designed to utilize a simulation module and handle large space problems. A need for systems that approach larger more real-world problems was identified, and this research is a step in that direction. The genetic algorithm (GA) proved that it was able to identify "near-optimal" signal timing strategies, but the idea of integrating a neural network (NN) and domain knowledge needs to be further explored.

It is important to realize the benefits of using a GA and the potential benefits of integrating a NN. The GA especially has proved to be useful for attacking the problem of determining traffic signal timings even though it does require significant computational effort. It can be postulated that because some success has been reported, other problems, which involve large feature spaces and large classification spaces, could especially gain from using a GA, and also may gain from implementing a similar hybrid system.

On the other hand, our preliminary tests which used domain knowledge to structure NNs did not seem to consistently improve NN learning or classification. The work done in this area is only preliminary, and further study is needed to determine the absolute value of using a NN in this framework.

We have learned that integrating knowledge may not be as easy as it appears, but we have outlined different ideas which could help improve the performance of the NN in the future.

6. REFERENCES

[Foy et al., 1992] Foy, M., Benekohal, R.F., and Goldberg, D.E. "Signal Timing Determination Using Genetic Algorithms" to be published in The Transportation Research Record, in 1993.

[Goldberg, 1989] Goldberg, D.E. Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, MA: Addison-Wesley, 1989.

[Holland, 1975] Holland, J.H. Adaptation in Natural and Artificial Systems. Ann Arbor, MI: University of Michigan Press, 1975.

[PICGA, 1985] Proceedings of an International Conference on Genetic Algorithms and Their Applications. Carnegie-Mellon University, July 24-26, 1985. Edited by John J. Grefenstette. U.S. Navy Center for Applied Research in Artificial Intelligence, 1985.

[PICGA, 1987] Proceedings of the Second International Conference on Genetic Algorithms. Massachusetts Instituted of Technology, July 28-31, 1987. Edited by John J. Grefenstette. L. Erlbaum Associates, 1987.

[PICGA, 1989] Proceedings of the Third International Conference on Genetic Algorithms. George Mason University, June 4-7, 1989. Edited by J. David Schaffer. Morgan Kaufmann Publishers, Inc., 1989.

[Katz, 1989] Katz, B. "EBL and SBL: A Neural Network Synthesis" in Proceeding of the Eleventh Conference of Cognitive Science School, 683-689, 1989.

[Quinlan, 1983] Quinlan, J. R. "Learning Efficient Classification Procedures and their Application to Chess End Games" in Ryszard Michalski (Ed.) Machine Learning: An Artificial Approach, Tioga, 1983.

[Reljic, 1988] Reljic, S. "TRAFSIG: A Computer Program for Signal Settings at an Isolated, Under- or Oversaturated, Fixed-Time Controlled Intersection" in Traffic Engineering and Control, Vol. 29, No. 11, Nov. 1988, 562-566, 1988.

[Shavlik and Towell, 1989] Shavlik, J.W. and Towell, G.G. "An Approach to Combining Explanation-based and Neural Learning Algorithms" in Connection Science, Vol. 1, No. 3, 231-251, 1989.

[Towell et al., 1990] Towell, G.G., Shavlik, J.W., and Noordewier, M.O. "Refinement of Approximate Domain Theories by Knowledge-Based Neural Networks" in AAAI-90, 861-866, 1990.