

Dynamic Traffic Assignment Genetic Algorithms Approach

ADEL W. SADEK, BRIAN L. SMITH, AND MICHAEL J. DEMETSKY

Real-time route guidance is a promising approach to alleviating congestion on the nation's highways. A dynamic traffic assignment model is central to the development of guidance strategies. The artificial intelligence technique of genetic algorithms (GAs) is used to solve a dynamic traffic assignment model developed for a real-world routing scenario in Hampton Roads, Virginia. The results of the GA approach are presented and discussed, and the performance of the GA program is compared with an example of commercially available nonlinear programming (NLP) software. Among the main conclusions is that GAs offer tangible advantages when used to solve the dynamic traffic assignment problem. First, GAs allow the relaxation of many of the assumptions that were needed to solve the problem analytically by traditional techniques. GAs can also handle larger problems than some of the commercially available NLP software packages.

Transportation departments throughout the United States are making significant investments in computer software as they deploy Intelligent Transportation Systems (ITSs). Much of this software supports such traffic management activities as incident detection, traffic monitoring, and control of variable message signs (VMSs). Very little software is available to help traffic managers provide route guidance information to travelers.

Considerable research is being conducted to develop route guidance tools. An efficient dynamic traffic assignment model is central to the development of route guidance strategies. Given the travel demand, a dynamic traffic assignment model is used to estimate the time-varying traffic volume on each road segment (link) of the network, which will result in an optimal use of the region's transportation resources. The model can then be used to develop effective routing strategies. These can be relayed to the public by devices such as VMS, highway advisory radio (HAR), or personal computers. The solution of dynamic traffic assignment models, however, has proven to be difficult, and previous attempts have introduced a number of simplifying assumptions to allow the model to be solved by conventional algorithmic techniques (1-5).

In this paper, the artificial intelligence (AI) technique known as genetic algorithms (GAs) is used to solve a dynamic traffic assignment model developed for a real-world routing scenario in the Hampton Roads region of Virginia. The paper starts by describing the dynamic assignment model developed by Merchant and Nemhauser (1), which was selected to illustrate the feasibility of the GA approach, and then formulates a dynamic model for the Hampton Roads area. The development of the GA program, and the major issues associated with the application of GAs to dynamic traffic assignment problems, are addressed next, followed by the results of the GA program.

MERCHANT AND NEMHAUSER DYNAMIC TRAFFIC ASSIGNMENT MODEL

Merchant and Nemhauser (M-N) were among the first researchers to consider the dynamic traffic assignment problem (1). The M-N model addressed the case of single-destination networks and a system-optimal assignment formulation. The planning horizon was divided into equal time intervals of suitably small length $\{i | i = 0, 1, \dots, I\}$. Each arc of the network $\{j | j = 1, 2, \dots, J\}$ was assigned a cost function (h_{ij}) and an exit function g_j . When x is the amount of traffic on arc j at the beginning of time period i , it was assumed that a cost $h_{ij}(x)$ is incurred and an amount of traffic $g_j(x)$ exits from the arc. The two functions $h_{ij}(x)$ and $g_j(x)$, however, must satisfy certain requirements. First, to represent traffic flow accurately, the function $g_j(x)$ must be nondecreasing, continuous, and concave. The function $h_{ij}(x)$, on the other hand, should be nonnegative, nondecreasing, continuous, and convex, to represent the disutility associated with congestion.

Denoting the number of vehicles that are admitted onto the arc j during the i th period by d_{ij} (the decision variables), and assuming that the external inputs are known for each time period, $F_i(q)$, and that the volume admitted onto a link cannot leave that link in the same time interval, the fundamental state equations can be written as

$$x_{i+1,j} = x_{ij} - g_j(x_{ij}) + d_{ij} \quad i = 0, 1, \dots, I-1 \text{ and } j = 1, 2, \dots, J \quad (1)$$

The flow conservation equations at each node are given as

$$\sum_{j \in A(q)} d_{ij} = F_i(q) + \sum_{j \in B(q)} g_j(x_{ij}) \quad i = 0, \dots, I-1 \quad (2)$$

where $A(q)$ is the set of arcs pointing out of node q , and $B(q)$ is the set of arcs pointing into the node.

Therefore, the full model can be written as

minimize

$$\sum_{i=1}^I \sum_{j=1}^a h_{ij}(x_{ij}) \quad (3)$$

subject to

- The state equations (Equation 1),
- The conservation constraints (Equation 2),
- The initial condition

$$x_{0j} = R_j \geq 0 \quad j = 1, 2, \dots, J \quad (4)$$

- The nonnegativity constraints

$$d_{ij} \geq 0 \quad i = 0, 1, \dots, I-1 \text{ and } j = 1, 2, \dots, J \quad (5)$$

$$x_{ij} \geq 0 \quad i = 0, 1, \dots, I-1 \text{ and } j = 1, 2, \dots, J \quad (6)$$

The model is a discrete time, nonlinear, and nonconvex model. It is interesting that the model does not explicitly impose capacity constraints on the arc flows.

MODEL FORMULATION

As previously mentioned, the model was developed for a specific routing challenge near Hampton Roads, Virginia. This routing scenario (Figure 1) involves westbound traffic originating from Virginia Route 44 and destined for Interstate 64 in Newport News, Virginia. Essentially, the problem is how to allocate drivers between the Hampton Roads Tunnel and the Monitor-Merrimac Bridge Tunnel.

Defining Network To Be Modeled

The first step in formulating the model was to define the highway network considered for modeling. This network had to include the major facilities in the area as well as the location of those access/exit points where it was believed that a significant change in the traffic volume occurs. The network selected is shown in Figure 2.

As can be seen, the network comprises the Interstate system in the region, I-64, I-264, I-464, and I-664. The network, besides the intersection of Route 44 with I-64 (Point O1) where traffic is originating, has two major access points: (a) Point O2 where I-564 intersects the network (this point connects the network to a major naval base); and (b) Point O3, where Terminal Avenue intersects I-664 (this point connects the network to an international port). In total, there are nine links, the lengths of which, as obtained from the Virginia Department of Transportation’s mileage tables, are given in Figure 2. For simplicity, all links were assumed to consist of two lanes per direction.

Traffic Volumes

Dynamic traffic assignment models assume that the originating traffic demand as well as all external inputs are known for the different

time intervals throughout the planning horizon. Therefore, they need to be linked to real-time traffic data and integrated with a tool for short-term traffic prediction. Researchers at the Virginia Transportation Research Council (VTRC) and the University of Virginia have recently developed a nonparametric regression model well-suited for predicting traffic based on current and historical volumes (6). In this case, the VTRC model will be linked to the Suffolk traffic management system (TMS) traffic sensors to provide the input data for the dynamic traffic assignment model.

Unfortunately, the TMS was not yet on-line at the time this study was conducted. The only traffic information available was in the form of 15-min counts for the two tunnels. These counts were used as guides in assuming the traffic demand for the different time intervals.

Exit Function

The functional form for the exit function selected is

$$Y = A[1 - \exp(-Bx)]$$

where *A* and *B* are regression parameters. This form, which closely resembles the free-flow regime of Edie’s two-regime traffic model (7), satisfies the different requirements of an exit function since it is nonnegative, nondecreasing, and concave. Moreover, it satisfies the assumption that the gradient is 0 for large *x*’s.

Ideally, to calibrate this function, traffic volume data from the Hampton Roads network should have been used. However, these data were not available and data collected from Santa Monica Freeway in California were used instead (7). Using nonlinear regression analysis, the following relationship between the flow, *Q*, in vehicles per hour per lane, and the density, *d*, in vehicles per mile per lane, was developed:

$$Q = 2200 * [1 - \exp(-d/25)] \tag{7}$$

Since two-lane links have been assumed, Equation 7 can be written as

$$Q' = 4400 * \{1 - \exp[-(x/l)/50]\} \tag{8}$$

where

- Q'* = number of vehicles exiting from a particular link (per hour),
- x* = number of vehicles on that link, and
- l* = length of link (mi).

As previously mentioned, the M-N model assumes that the volume admitted onto a link cannot leave that link in the same time interval. This means that the selected length of the time interval, *T*, has to be short enough for such a condition to be satisfied. According to Papageorgiou (4), this translates into

$$T < \text{minimum} (l_j * 50 / 4400)$$

that is,

$$T < (4.91 * 50 / 4400) < 0.056 \text{ hr, or } 3.36 \text{ min} \tag{9}$$

A time interval length of 3 min was assumed. The exit function was then derived by appropriately scaling Equation 8 to give

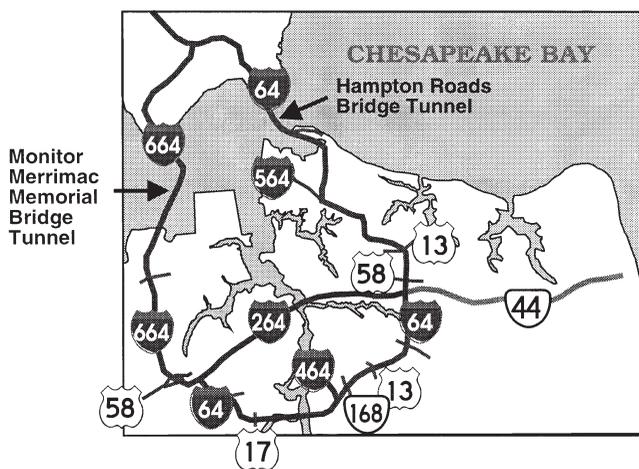


FIGURE 1 Hampton Roads area network.

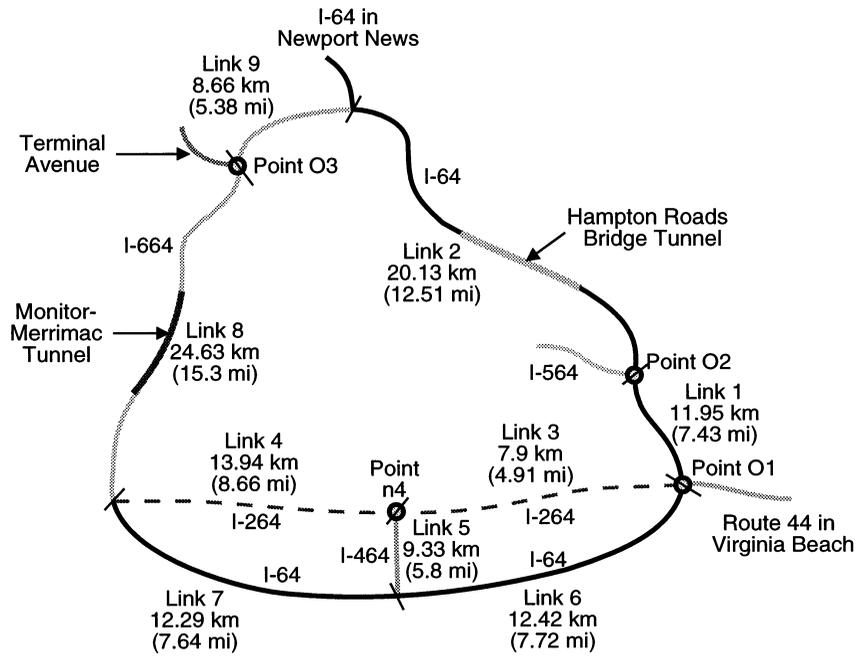


FIGURE 2 Network selected for modeling.

$$g = 219 * \{1 - \exp[(-x/l)/50]\} \tag{10}$$

where g is the number of vehicles exiting in 3 min. Equation 10 is the exit function used in this paper.

Model

To solve the model, a period of 45 min was used, corresponding to 15 intervals of 3 min each. The objective function minimizes the total number of vehicle periods spent on the network and can be expressed as

Minimize

$$\sum_i \sum_j x_{ij} \tag{11}$$

It should be mentioned, however, that since we are adopting a GA approach, any functional form of the objective function can be used.

Minimizing the objective function (Equation 11) is subject to five groups of constraints; state equations, conservation equations, initial conditions, upper bounds, and nonnegativity.

State Equations Constraints

For nine links and 15 time intervals, there are 135 state equations constraints. A sample of these constraints for time interval 0 is given here

$$x_{11} = x_{01} - 219. [1 - \exp (-x_{01}/50.l_1)] + d_{01} \tag{12}$$

$$x_{12} = x_{02} - 219. [1 - \exp (-x_{02}/50.l_2)] + d_{02} \tag{13}$$

$$x_{13} = x_{03} - 219. [1 - \exp (-x_{03}/50.l_3)] + d_{03} \tag{14}$$

$$x_{14} = x_{04} - 219. [1 - \exp (-x_{04}/50.l_4)] + d_{04} \tag{15}$$

$$x_{15} = x_{05} - 219. [1 - \exp (-x_{05}/50.l_5)] + d_{05} \tag{16}$$

$$x_{16} = x_{06} - 219. [1 - \exp (-x_{06}/50.l_6)] + d_{06} \tag{17}$$

$$x_{17} = x_{07} - 219. [1 - \exp (-x_{07}/50.l_7)] + d_{07} \tag{18}$$

$$x_{18} = x_{08} - 219. [1 - \exp (-x_{08}/50.l_8)] + d_{08} \tag{19}$$

$$x_{19} = x_{09} - 219. [1 - \exp (-x_{09}/50.l_9)] + d_{09} \tag{20}$$

Conservation Equations Constraints

For each time interval, there are six conservation equations—that is, there are a total of 90 conservation constraints. A sample of these equations for time interval 0 is shown here.

$$d_{01} + d_{03} + d_{06} = O1_0 \tag{21}$$

$$d_{02} = O2_0 + g(x_{01}) \tag{22}$$

$$d_{04} + d_{05} = g(x_{03}) \tag{23}$$

$$d_{07} = g(x_{05}) + g(x_{06}) \tag{24}$$

$$d_{08} = g(x_{04}) + g(x_{07}) \tag{25}$$

$$d_{09} = O3_0 + g(x_{08}) \tag{26}$$

where $O1_0$, $O2_0$, and $O3_0$ are the external traffic volume inputs at access points O1, O2, and O3; and the g 's are the exit functions.

Initial Conditions

The Hampton Roads Tunnel typically carries more traffic than the Monitor-Merrimac Tunnel. To reflect this Links 1 and 2 (Figure 2) were assumed to be initially loaded to a uniform traffic density of 31.1 veh/km (50 veh/mi), whereas an 18.6-veh/km (30-veh/mi) traffic density was assumed for the other seven links. The initial number of vehicles on each link (the x_{0j} 's) was then computed by multiplying the traffic density by the link length, resulting in the following set of constraints:

$$x_{0j} = 50 \cdot l_j \quad \text{for } j = 1 \text{ and } 2 \quad (27)$$

$$x_{0j} = 30 \cdot l_j \quad \text{for } j = 3, \dots, 9 \quad (28)$$

Upper Bounds on Control Variables, d_{ij} , and State Variables x_{ij}

Merchant and Nemhauser did not impose explicit bounds on the link capacities in order to facilitate the solution of their model. In the current study, to more realistically capture the traffic dynamics, upper bounds were explicitly imposed on the control and state variables.

Upper bounds on the control variables were determined from roadway capacity considerations, which state that the maximum flow rate is 2,200 veh/hr/lane. For two lanes and 3-min intervals, this corresponded to an upper bound of 219 on the control variables, resulting in the following set of constraints:

$$d_{ij} \leq 219 \quad \text{for } i = 0, 1, \dots, I - 1 \text{ and } j = 1, 2, \dots, 9 \quad (29)$$

The upper bounds on the state variables were calculated assuming a jam density of 80.8 veh/km/lane (130 veh/mi/lane), giving rise to the following constraints:

$$x_{ij} \leq (2 \cdot 130 \cdot l_j) \quad \text{for } i = 0, 1, \dots, I - 1 \text{ and } j = 1, 2, \dots, 9 \quad (30)$$

It is noted that in the absence of these constraints, the volume that the model admits onto a link may exceed the actual capacity of that link.

Nonnegativity Constraints

Finally, we have the nonnegativity constraints

$$d_{ij} \geq 0 \quad \text{for } i = 0, 1, \dots, I - 1 \text{ and } j = 1, 2, \dots, 9 \quad (31)$$

$$x_{ij} \geq 0 \quad \text{for } i = 0, 1, \dots, I \text{ and } j = 1, 2, \dots, 9 \quad (32)$$

The next section describes the GA program that was designed to solve this model.

GENETIC ALGORITHM PROGRAM

Overview

GAs are stochastic algorithms whose search methods are based on the principle of evolution and survival of the fittest. GAs use a vocabulary borrowed from natural genetics. One would speak about

individuals (sometimes called *strings*, or *chromosomes*) in a population. Chromosomes are made of genes arranged in linear succession. The basic idea of the GA is quite simple. During each iteration, t , the procedure maintains a population of individuals, $P(t)$. Each individual or chromosome represents a potential solution to the problem under consideration. The procedure starts with a randomly generated initial population of chromosomes (a set of potential solutions). Each solution, x_t^i , is evaluated to give some measure of its fitness (the *evaluate* step). Then, a new population (iteration $t + 1$) is formed by selecting the more fit individuals (the *select* step). Some members of this new population undergo alterations by means of genetic operations (typically referred to as *crossover* and *mutation* operations) to form new solutions (the *alter* step). After some number of generations (iterations of the select, alter, and evaluate steps), it is expected that the algorithm converges to a near-optimum solution (8).

Traditional Versus Modern Approaches to GA Design

Traditionally, GAs have used a binary representation scheme, in which the solution was represented in the form of a binary string (e.g., 10010001111000). The crossover operator was designed to combine the features of two parent chromosomes to form two offspring by swapping corresponding segments of the parents. For example, if the parents are represented by five-dimensional vectors $(a_1, b_1, c_1, d_1, e_1)$ and $(a_2, b_2, c_2, d_2, e_2)$, then crossing the chromosomes after the second gene would produce the offspring $(a_1, b_1, c_2, d_2, e_2)$ and $(a_2, b_2, c_1, d_1, e_1)$. The intuition behind the applicability of the crossover operator is information exchange between different potential solutions. *Mutation*, on the other hand, arbitrarily selected one or more genes of a chromosome. It then flipped the gene into a 1 if it were a 0 and vice versa. The intuition behind the mutation operator is the introduction of some extra variability into the population.

Recently, however, it has become more and more apparent that real-world problems cannot be handled with binary representations and binary operators. In addition, every real-world domain has associated domain knowledge that is of use when considering a transformation of a solution. Consequently, the modern approach to GA design is characterized by a departure from classical, bit-string GAs toward the use of appropriate data structures, such as floating point representations, and special genetic operators (8). In the current study, the authors have adopted this modern view of GA design.

Constraints Handling

A major problem in applying GAs is that of constraints. Currently, there are three main approaches for handling constraints in connection with GAs. The first approach uses appropriate data structures and specially designed operators. The idea is to start with a feasible initial population and design genetic operators that maintain such feasibility. The second adopts a penalty function approach. In this approach, potential solutions are generated without considering the constraints. Solutions that violate the constraints are then penalized by decreasing the goodness of the fitness function. Several penalty functions have been proposed in recent years. A paper by Michalewicz and Janikow provides an excellent overview of these functions (9). Finally, the third approach uses "decoders" or repair algorithms that either avoid building or repair an illegal individual.

In this paper a hybrid approach for constraint handling was designed, combining features from all three approaches. The following section describes the details of the GA implementation to solve the dynamic traffic assignment model formulated in the previous section.

GA Implementation

Representation

Since every state $x_{i+1,j}$ is a function of the previous state x_{ij} and a control variable d_{ij} , and since, the initial state x_{0j} is given, the objective function will only depend on the control variables, the d_{ij} 's. It should also be noted that although there are nine d_{ij} 's for each time interval, in fact only three control variables need to be considered. This is because once the d_{i1} and the d_{i3} variables are selected, the d_{i6} is determined automatically from the fact that the sum of the d_{i1} , d_{i3} , and d_{i6} must be equal to $O1_i$ (Figure 3a). Similarly, selecting d_{i4} determines the value for d_{i5} (Figure 3b). Finally, the values of d_{i2} , d_{i7} , d_{i8} , and d_{i9} are a function of the amount of traffic volume exiting from the preceding links, and the external inputs $O2_i$ and $O3_i$ (Figure 3c). The exit volumes are also decided once the values for the d_{i1} , d_{i3} , and d_{i4} variables are selected.

The developed GA program uses a real-value vector representation. Since 15 time intervals were considered and each had three control variables, a potential solution to the problem would be represented as a 45-element vector as follows:

$\mathbf{u} = (u_1, u_2, u_3, u_4, u_5, u_6, \dots, u_{45})$, corresponding to the control variables, $(d_{0,1}, d_{0,3}, d_{0,4}, d_{1,1}, d_{1,3}, d_{1,4}, \dots, d_{14,4})$

Initial Population and Constraint Handling

The basic idea in creating the initial population was first to determine the upper and lower bounds for each control variable, and then to select a random number between these bounds for this variable. As mentioned, the lower bound for these control variables (the d_{ij} 's) is 0, since we cannot have a negative flow, while the upper bound is 219.

Therefore,

$$0 \leq d_{i1} \leq 219 \tag{33}$$

$$0 \leq d_{i3} \leq 219 \tag{34}$$

$$0 \leq d_{i4} \leq 219 \tag{35}$$

Also, as previously discussed,

$$d_{i1} + d_{i3} + d_{i6} = O1_i$$

that is,

$$d_{i6} = O1_i - d_{i1} - d_{i3} \tag{36}$$

Imposing the bounds on the d_{i6} means that

$$O1_i - d_{i1} - d_{i3} \geq 0$$

that is,

$$d_{i1} + d_{i3} \leq O1_i \tag{37}$$

and

$$O1_i - d_{i1} - d_{i3} \leq 219$$

that is,

$$d_{i1} + d_{i3} \geq O1_i - 219 \tag{38}$$

Similarly,

$$d_{i4} + d_{i5} = g(x_{i,3}) = f(d_{i-1,3})$$

that is,

$$d_{i5} = f(d_{i-1,3}) - d_{i4} \tag{39}$$

Imposing the bounds on the d_{i5} means that

$$f(d_{i-1,3}) - d_{i4} \geq 0$$

that is,

$$d_{i4} \leq f(d_{i-1,3}) \tag{40}$$

and

$$f(d_{i-1,3}) - d_{i4} \leq 219$$

that is,

$$d_{i4} \geq f(d_{i-1,3}) - 219 \tag{41}$$

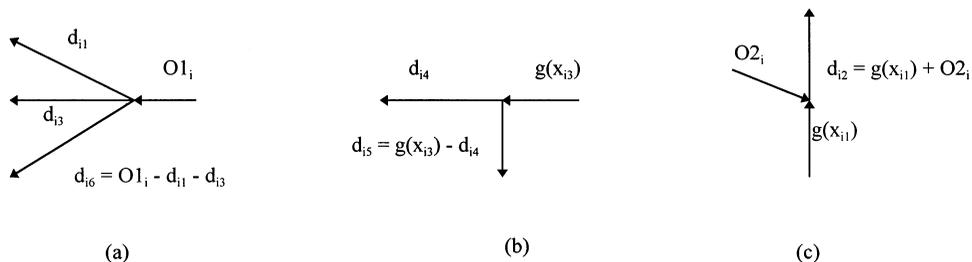


FIGURE 3 Interdependency among control variables.

The procedure for population initialization thus consists of the following steps:

1. Randomly select the d_{i1} from between 0 and 219.
2. Calculate the lower and upper bounds for the d_{i3} using inequalities 34, 37, and 38; randomly select the d_{i3} from between these bounds.
3. Calculate the lower and upper bounds for the d_{i4} using inequalities 35, 40, and 41; randomly select the d_{i4} from between these bounds.

This approach for initializing the population will guarantee that the control variables (namely, d_{i1} , d_{i3} , and d_{i4}) are within the bounds, as well as the d_{ij} 's for links that share a node with a control variable, namely, d_{i6} and d_{i5} . There is no guarantee, however, that this will hold for the other d_{ij} 's that do not share a node with a control variable (i.e., d_{i2} , d_{i7} , d_{i8} , and d_{i9}). In fact, calculating the appropriate bounds for the control variables in such a case is rather complicated, since the bound is a function of a combination of the values assigned to the control variables in previous time intervals. The same problem applies to the constraints on the state variables.

To avoid the need for such complex calculations, a hybrid approach for constraint handling was adopted. The basic idea was to divide the upper-bound constraints on the variables into two groups:

- *Group 1*, consisting of the constraints on the d_{ij} 's that are part of the control vector (d_{i1} , d_{i3} , and d_{i4}), as well as those that share a node with a control variable (d_{i6} and d_{i5}); and
- *Group 2*, consisting of the constraints on the other d_{ij} 's, namely, d_{i2} , d_{i7} , d_{i8} , and d_{i9} , as well as the constraints on the state variables.

The Group 1 constraints are treated as before by creating an initial "feasible" population with respect to these constraints and attempting to maintain this partial feasibility. For Group 2, the constraints were handled using a penalty function approach. The penalty function used was a simplified version of the function used by Michalewicz and Attia (10). This function is given as

$$eval(X, r) = f(X) + 1/2r \sum_{k \in CA} f_k^2(X) \quad (42)$$

where

$f(X)$ = evaluation function;

CA = set of active constraints;

$$f_k = \begin{cases} \max\{0, a_k(X)\} & \text{for inequalities of form } a_k(X) \leq 0 \\ |b_k(X)| & \text{for equalities of form } b_k(X) = 0 \end{cases}$$

r = parameter with a value > 0 .

Michalewicz and Attia propose iteratively reducing the value of r , with the best solution from one iteration serving as a starting point for the next. In the current implementation, however, the authors used just a single value for r of 0.10.

Evaluation Function and Selection Scheme

The objective function for the dynamic traffic assignment model served as the evaluation function for the GA program. The selection scheme used to select the more fit individuals from a population was

the commonly used roulette wheel procedure. Michalewicz (8) has described the details of this stochastic selection procedure.

Genetic Operators

The operators used in the GA program were specially designed to maintain the partial feasibility of the solution with respect to the Group 1 constraints, and therefore they are quite different from the classical operators.

Mutation Operator When designing the mutation operator, special attention was given to the fact that the domain of the problem was dynamic. That is, the value of the i th component of the solution vector was always in some dynamic range, where the bounds depended on the values of the other elements of the vector and the set of inequalities. To accommodate this factor, the mutation operator was designed to proceed in the following fashion:

1. Randomly select a gene replace it by a random number selected from between that gene's bounds.
2. Update the bounds for the genes that follow the gene selected in Step 1.
3. Check the values of the genes following the mutated gene to see if each is within its new range as determined from Step 2. If any variable is outside such a range, reset its value to that of the boundary.

Crossover Operator The crossover operator is of the whole arithmetical crossover type (8). It is defined as a linear combination of two vectors. That is, if the two vectors u_1 and u_2 are to be crossed, the resulting offspring is $\rho \cdot u_1 + (1 - \rho) \cdot u_2$. The value ρ is a randomly generated number in the range $[0 \dots 1]$.

A basic characteristic of a convex space, S , is that for any two points in S , the linear combination is also a point in S . Therefore, had the solution space been a convex set, the whole arithmetical crossover operator would have been guaranteed to always generate a legitimate offspring. But since the solution space for this problem was not necessarily convex, the crossover operator was designed to check for any violation and to repair such a violation by resetting the gene's value to that of the boundary. It should be noted, however, that for all the runs performed in the study, the crossover operator was never seen to generate any violation.

DISCUSSION OF RESULTS

The GA model, illustrated in Figure 4 as a flow chart, was coded in C++ and used to solve the formulated dynamic assignment model for the demand volumes given in Table 1. The program required 210 sec to run on an IBM-PC80486 for the following set of values for the control parameters:

- Population size = 30,
- Probability of crossover = 0.25,
- Probability of mutation = 0.03, and
- Number of generations = 1,000.

A value of 45,009 vehicle periods spent on the network was obtained for the objective function. Figure 5 shows the resulting tra-

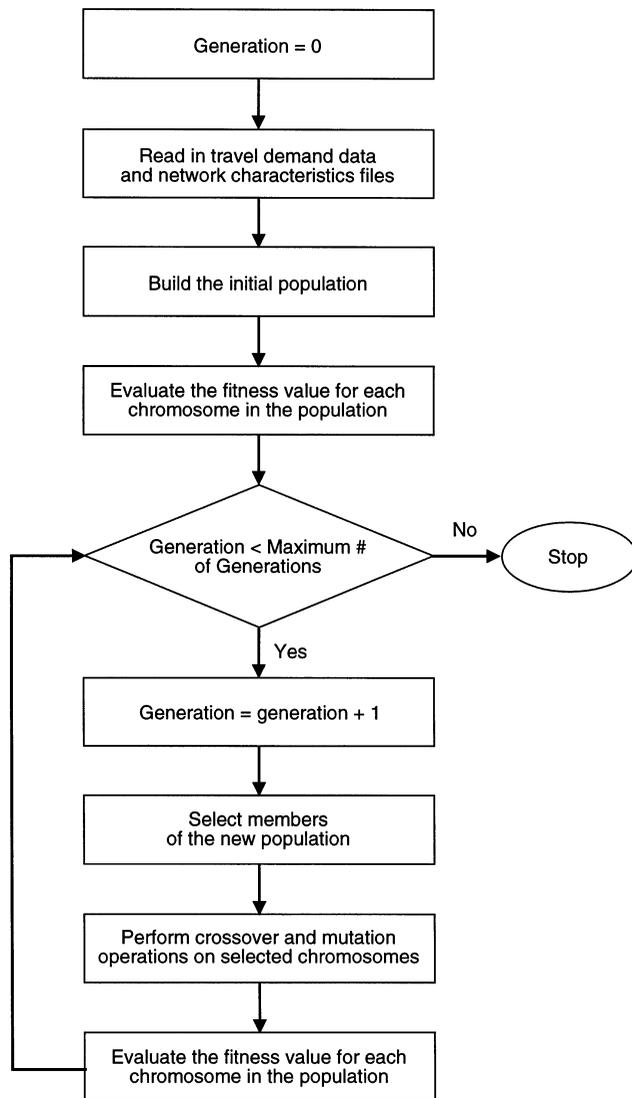


FIGURE 4 Basic structure of GA program.

jectories of the three control variables d_{i1} , d_{i3} , and d_{i4} . As can be seen, in general the variable d_{i1} was greater than d_{i3} , which in turn was greater than d_{i4} . This appears quite reasonable if the following facts are considered.

The network modeled essentially had four alternative paths leading from the origin (Route 44) to the destination:

1. (Link 1 → Link 2) with a total length of 19.94 mi (32.09 km),
2. (Link 3 → Link 4 → Link 8 → Link 9) with a total length of 34.25 mi (55.1 km),

3. (Link 3 → Link 5 → Link 7 → Link 8 → Link 9) with a total length of 39.03 mi (62.8 km), and
4. (Link 6 → Link 7 → Link 8 → Link 9) with a total length of 36.08 mi (58.06 km).

In this solution, the variable d_{i1} corresponds to the traffic volume following Path 1 to the destination, while the d_{i3} variable represents the volume taking either Path 2 or 3. Now, since Path 1 is shorter than either Paths 2 or 3, one should expect the model would attempt to route as many vehicles as possible through Path 1, therefore d_{i1} should be greater than d_{i3} .

To further check the plausibility of the model, it was assumed that an incident took place on Link 2, and that this incident has reduced the link’s exit capacity to 25 percent of its original value. Under such circumstances, one should expect that the model would attempt to divert traffic from Path 1 onto alternative paths. Figure 6 shows that this is exactly what happened. The total traffic volume admitted onto Path 1 over the 45-min period in this case is less than the volume for the case of no incidents. It is noted, however, that because of the oscillations in the value of the control variable, the number of vehicles for some time intervals during an incident could be at the same or higher level as during normal conditions, despite the fact that the total volume over the planning horizon is less. If desired, these dynamic oscillations could be smoothed or even eliminated by adding a term in the objective function that penalizes such time variations (4).

Varying GA Control Parameters

To study the effect of the GA control parameters on the algorithm performance, a series of experiments were performed with the mutation and crossover probabilities varied as follows:

1. The mutation probability was set at 0.01, 0.03, and 0.05; and
2. The crossover probability was set at 0.25, 0.50, and 0.80.

The number of generations was fixed at 500 generations and a population size of 30 was used. Table 2 gives the fitness value for the solution obtained in each case. As can be seen, the best result was obtained when the mutation rate was set at 0.03 and the crossover at 0.25.

Execution Time Characteristics of GA Model

As previously mentioned, the execution time for the previous example on an IBM-PC80486 was 210 sec. A study of the effect of the number of generations on the solution quality, however, revealed that no significant improvement was achieved beyond 300 generations. Since the execution time is a linear function of the generation

TABLE 1 Demand Volumes at Three Access Points During 15 Time Intervals (vehicles/3-min intervals)

int. #	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)
Point O1	240	200	260	280	220	330	180	270	240	210	250	270	210	290	230
Point O2	70	80	90	60	40	75	80	60	70	40	60	30	65	75	65
Point O3	50	45	60	30	70	40	60	45	55	50	50	40	80	60	30

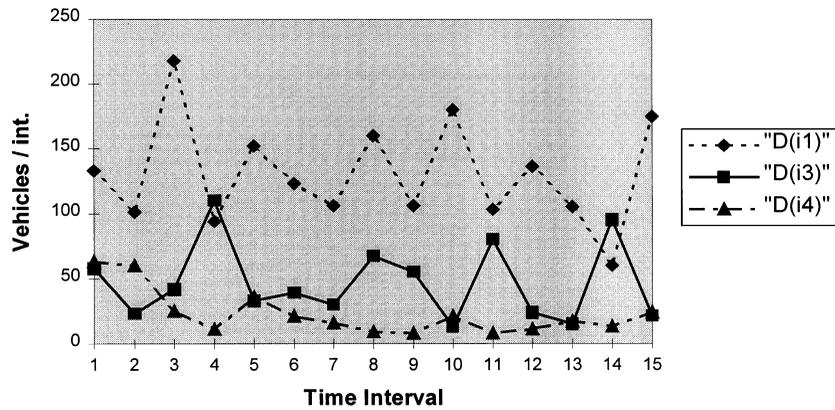


FIGURE 5 Control variables trajectories.

number, running the program for 250 generations would require only 52.5 sec on a 486 PC. This is reasonable.

To better assess the execution time characteristics of the GA program, four demand patterns were considered, and the number of generations after which there was no significant improvement in the solution quality was recorded for each case. The results are shown here along with the time required to run the program for the corresponding number of generations.

Case	Number of Generations	Execution Time (sec)
1	330	69.3
2	30	6.3
3	285	59.9
4	182	38.2

As can be seen, the maximum execution time required was 69.3 sec, which is again adequate.

GA Program Versus Traditional Nonlinear Programming Software

To better assess the performance of the GA program, it was compared with a commercially available nonlinear programming (NLP) software, Microsoft Excel Solver, which uses a gradient descent approach to solve NLP problems (11).

The first observation was the fact that a dynamic traffic assignment problem could be too large for some of the commercially

available software. The authors were forced to reduce the size of the original 15 time interval problem to a 6 time interval problem to be able to use Solver. This simplified problem was then solved using Solver as well as by running the GA program for a 1,000 generations. The same travel demand volumes and the same model formulations were used in both cases. The results obtained are summarized here:

	Excel Solver	GA Program (1,000 generations)
Objective function value	17,691	17,697
Execution time (sec)	370	105

As can be seen, the GA program yielded similar results in less than a third of the time required by Solver. It should be noted that the 17,697 solution was reached by the GA program after only 120 generations. Running the GA for 120 generations would require less than 13 sec on a 486 PC.

CONCLUSIONS AND FUTURE DIRECTIONS

This study has demonstrated the feasibility of using GAs to solve dynamic network flow modeling problems. The conclusions are as follows:

1. The GA approach offers tangible advantages when used to solve dynamic traffic assignment problems. First, it permits the

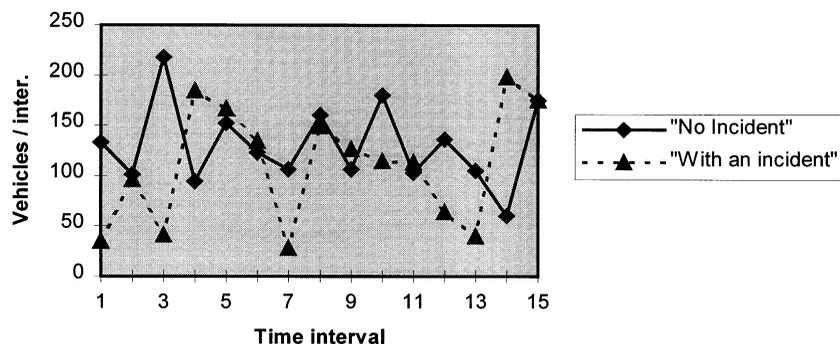


FIGURE 6 d_{i1} under incident and no-incident trajectories.

TABLE 2 Effect of Changing the Mutation and Crossover Rates

Mut./ Xover Prob.	0.25	0.50	0.80
0.01	45131	45132	45189
0.03	45009	45056	45046
0.05	45024	45050	45069

relaxation of many of the assumptions necessary to solve the problem analytically by traditional techniques. For example, the GA approach allows for using any mathematical form for the cost and exit function. Moreover, the approach allows for explicitly imposing capacity constraints on the arc flows.

2. The hybrid approach for constraint handling adopted by the current study appears promising in terms of both solution quality and coding complexity.

3. The execution time of the GA program for the range of demand patterns considered in the study was reasonable.

4. The GA program can handle larger problems than some of the commercially available NLP software.

5. When compared with Microsoft Excel Solver, an NLP tool, the GA program was much faster and yielded comparable results.

6. Since GA performance is affected by the values selected for the control parameters, running a set of experiments at first is recommended to arrive at the most appropriate values for these parameters.

There are several opportunities for further research in this topic. Some of the most important future directions that the VTRC research team plans to pursue given here:

1. The present study has considered only models that represent a system-optimal assignment; the authors also intend to formulate user-equilibrium models. Such an extension should be rather easy, since the GAs approach is capable of dealing with any functional form of the objective function.

2. The dynamic traffic assignment model formulated in this study addressed the case of a single-destination network. The

authors plan to develop a GA program for solving the more general case of a multiorigin, multidestination network.

3. Once the Suffolk TMS is on-line, the authors intend to refine the exit functions on the basis of the real-world data collected from the TMS's sensors. It is also intended to link the program to the short-term prediction module developed by the VTRC. This will result in a complete system for real-time traffic routing in the Hampton Roads area.

REFERENCES

1. Merchant, D. K., and G. L. Nemhauser. A Model and an Algorithm for the Dynamic Traffic Assignment Problem. *Transportation Science*, Vol. 12, 1978, pp. 183–199.
2. Ho, J. K. A Successive Linear Optimization Approach to the Dynamic Traffic Assignment Problem. *Transportation Science*, Vol. 14, 1980, pp. 295–305.
3. Carey, M. Optimal Time-Varying Flows on Congested Networks. *Operations Research*, Vol. 35, 1987, pp. 58–69.
4. Papageorgiou, M. Dynamic Modeling, Assignment and Route Guidance in Traffic Networks. *Transportation Research*, Vol. 24B, 1990, pp. 471–495.
5. Ran, B., D. E. Boyce, and L. J. LeBlanc. A New Class of Instantaneous Dynamic User-Optimal Traffic Assignment Models. *Operations Research*, Vol. 41, 1993, pp. 192–202.
6. Smith, B. L., and M. J. Demetsky. *Traffic Flow Forecasting for Intelligent Transportation Systems*. VTRC-95-R24. Virginia Transportation Research Council, Charlottesville, 1995.
7. May, A. D. *Traffic Flow Fundamentals*. Prentice-Hall, Englewood Cliffs, N.J., 1990.
8. Michalewicz, Z. *Genetic Algorithms + Data Structures = Evolutionary Programs*. Springer-Verlag, 1994.
9. Michalewicz, Z., and C. Janikow. Handling Constraints in Genetic Algorithms. In *Proc., 4th International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, Inc., pp. 151–157.
10. Michalewicz, Z., and N. Attia. Evolutionary Optimization of Constrained Problems. In *Proc., 3rd Annual Conference on Evolutionary Programming*. A. V. Sebald and L. J. Fogel, eds., World Scientific Publishing, River Edge, N.J., pp. 98–108, 1994.
11. *Microsoft Excel—User's Guide*. Microsoft, Redmond, Wash., 1994.