2008

An Evolutionary Method for the Minimum Toll Booth Problem: the Methodology

Lihui Bai
Valparaiso University

Matthew T. Stamps
University of California, Davis

R. Corban Harwood
George Fox University, rharwood@georgefox.edu

Christopher J. Kollmann
Concordia Seminary

AN EVOLUTIONARY METHOD FOR THE MINIMUM TOLL BOOTH PROBLEM: THE METHODOLOGY

Lihui Bai, College of Business Administration, Valparaiso University
Matthew T. Stamps, Department of Mathematics, University of California, Davis
R. Corban Harwood, Department of Mathematics, Washington State University
Christopher J. Kollmann, Concordia Seminary

ABSTRACT

This paper considers the minimum toll booth problem (MINTB) for determining a tolling strategy in a transportation network that requires the least number of toll locations, and simultaneously causes the most efficient use of the network. The paper develops a methodology for using the genetic algorithm to solve MINTB and presents the algorithm GAMINTB. The proposed method is tested and validated through a computational study with six example networks. Additional numerical test discovers some interesting properties for the proposed method, and provides guidelines for further application of the GAMINTB.

INTRODUCTION

Today, traffic congestion is a pressing issue for society and a major concern for urban planners. The 2007 Urban Mobility Report (Schrank & Lomax, 2007) states that congestion in the U.S. caused 4.2 billion hours of travel delay and 2.9 billion gallons of wasted fuel for a total cost of $78 billion in 2005. Although congestion pricing has been proposed by transportation economists as a means of reducing congestion for over 80 years (see e.g., Pigou (1920) and Beckmann, McGuire & Winston (1956)), it is only recently that this idea has been implemented in practice. Examples include the "Electronic Road Pricing" (formerly "Area Licensing Scheme") program in Singapore implemented in 1975, a toll ring in Bergen, Norway implemented in 1986, and subsequently two toll rings in Oslo and Trondheim, respectively. More recently, London introduced a $5 (now raised to $8) daily fee on cars entering the city center in February 2003. In the United States, the Federal Highway Administration (DeCorla-Souza, 2003) has been funding toll pricing projects in cities such as San Diego, Houston, and Seattle under the Congestion Pricing Pilot Program established by Congress since 1991.

This paper focuses on determining tolls with the primary objective of inducing travelers, who are encouraged only by their own travel costs, to choose routes that would collectively benefit all travelers and use the transportation system resources more efficiently. The secondary objective is to find tolls that require the least number of collection facilities or toll booths. The problem was first introduced in Hearn and Ramana (1998) and was referred to as the minimum toll booth problem (or MINTB). When compared to the marginal social cost pricing (MSCP) advocated by transportation economists, the MINTB solution often requires substantially fewer number of toll booths. For example, the well documented Sioux Falls network (Leblanc, Morlok, & Pierskalla,

1975) with 24 nodes and 76 arcs requires to toll on all 76 arcs under MSCP, but only 32 under the MINTB solution. For the Hull network (Florian & Guélat, 1987) with 501 nodes and 798 arcs, the MINTB solution requires 39 toll booths and this saves 224 compared to MSCP. Considering that operating a manned toll booth costs $180,000 per year (Todd, 2005), MINTB solutions would save approximately eight and forty million dollars per year for Sioux Falls and Hull, respectively. While today's electronic tolling through pre-registered cards, cameras and other systems can be available at a reasonable cost, which may affect the popularity of manned toll booths, safety concerns still exist when drivers have to merge to electronic tolling lanes frequently. Indeed, the latter may pose a major obstacle to the public and the transportation agency's acceptance of the tolling decisions when many toll booths are required in the proposal.

As stated, the MINTB problem was first introduced as a mixed integer linear programming in Hearn and Ramana (1998), where toll pricing problems are defined as finding the optimal tolling strategies that will make the transportation system run most efficiently and will simultaneously optimize a secondary tolling objective. In MINTB, the tolling objective is to minimize the total number of required toll facilities, i.e., toll booths. Other examples of tolling objectives include minimizing the total toll revenues (MINREV) and minimizing the maximum toll on a network (MINMAX). While toll pricing problems such as MINREV and MINMAX are linear and easier to solve for practical networks, efficient solution for MINTB has remained relatively unexplored in the transportation and management science literature.

An investigation of MINTB in Bai's dissertation (Bai, 2004) shows that MINTB is NP-complete. This, from a theoretical perspective, implies that solving MINTB optimally is very difficult. In fact, numerical experiments in Bai (2004) notes that general purpose solvers for mixed-integer programs, such as CPLEX, have been unable to produce optimal solutions for MINTB for a moderate network with 501 nodes and 798 arcs. This confirms the challenge of solving MINTB from an empirical perspective. Thus, solutions to MINTB resort to heuristic methods. For example, Bai (2004) and Bai, Hearn, & Lawphongpanich (2008) solved MINTB using a dynamic slope scaling procedure (DSSP), which is shown to be effective on larger networks such as Sioux Falls, Hull, and Stockholm. She also proposed a DSSP-based neighborhood function and used a simulated annealing method to solve MINTB (Bai, 2004). However, these local search methods may not always reach a global optimal solution.

Genetic algorithms (GA) are well known for their ability to sort through a variety of local optimal solutions until they converge to a global optimal solution (Shepherd & Sumalee, 2004). In the literature, genetic algorithms have been used to solve a variety of toll pricing problems. Zhang (2003) uses a GA to determine the optimal toll locations for a second-best link-based congestion pricing problem, where a simulated annealing method was used to solve for the optimal toll levels. Sumalee (2004) uses a genetic algorithm in conjunction with a branch-tree framework to create a toll set in a cordon-pricing scheme.

Thus far, a genetic algorithm has not been used to solve MINTB and this is the main purpose of the present paper. We propose a genetic algorithm for MINTB (GAMINTB), focusing on the methodology development and validation through small examples. The algorithm first randomly generates binary vectors, each of which represents whether or not a toll booth is used on any arc for a given network. After using

a linear program solver to determine feasibility and calculate optimal toll levels, GAMINTB divides all randomly generated toll vectors into two groups: feasible and infeasible groups; and then ranks them according to their feasibility and total number of toll booths. Based on this ranking, the algorithm randomly selects toll vectors to become "parents" where higher ranked toll vectors have higher probability of being selected. The parent toll vectors then reproduce the new generation of toll vectors following an alteration process consisting of "crossover" and "mutation." The new population is again evaluated and ranked, and the process continues until the algorithm reaches the specified number of generations. Finally, the solution with the minimal number of toll booths is returned as the final optimal solution.

For the remainder, the next section provides mathematical formulations for the traffic assignment and the MINTB problems. Then, the genetic algorithm is introduced, followed by preliminary computational results. Finally, we discuss an alternative implementation for future study before offering concluding remarks.

PROBLEM DESCRIPTION

This section formulates the minimum toll booth problem using network notation. The transportation network is represented by sets of arcs and nodes corresponding to roads and intersections, respectively. An origin-destination (O-D) pair (also known as a commodity) is a pair of nodes on which a user (traveler) in the system must begin and end. There is usually more than one path available for each user to travel along for a given commodity. Mathematically, let *G=(N,A)* denote a network with the set of nodes *N* and set of arcs *A*. The set of origin destination pairs is denoted as *K* where *o(k)* and *d(k)* are the origin and destination nodes for the O-D pair *k*. The demand for an O-D pair *k*, denoted as D_k, is simply the number of travelers going from the origin to the destination. Let $x^k \in R_+^{|A|}$ represent the flow vector for commodity *k*. The vector x^k is feasible if it satisfies $U x^k = b_k$ where *U* is the node-arc incidence matrix for *G=(N,A)* and $b_k \in R^{|N|}$ is the demand vector defined by

$$(b_k)_i = \begin{cases} D_k & \text{if } i = o(k) \\ -D_k & \text{if } i = d(k) \\ 0 & \text{otherwise} \end{cases}.$$

The sum of all flows x^k over all O-D pairs that pass over a given arc *a* is denoted $v_a = \sum_k x_a^k$. It follows, then, that an aggregate flow vector *v* is feasible only if $v = \sum_k x^k$, $U x^k = b_k$, and $x^k \geq 0, \ \forall k \in K$. Finally, the cost of each path can be measured by the amount of time it takes to travel from the origin to the destination for each user. Let *s(v)* be the travel cost vector for a given flow *v* in the network.

The User Equilibrium Model

The *User Equilibrium* (UE) model is used to describe the behavior of users on a given traffic network when every user chooses the shortest path available (See e.g.,

Florian and Hearn (1995)). Equivalently, a network is at UE only if the cost of every utilized path for each O-D pair is less than or equal to the cost of every non-utilized path for the respective O-D pair. In mathematical terms, a network $G(N,A)$ is at UE with $\tilde{v}$ being the equilibrium flow if it satisfies the following condition:

$$s(\tilde{v})^T (v - \tilde{v}) \geq 0, \quad \forall v \in V,$$

where V is the set of all feasible aggregate flow vectors for $G(N,A)$.

$$V = \{v \mid v = \sum_k x^k, Ux^k = b_k, x^k \geq 0, \forall k \in K\}.$$

Under UE, each user chooses the path for their own interests. Collectively, UE may not be in the best interest for the system as a whole. If too many travelers select the same path, congestion develops. Typically, the per capita cost of traveling along an arc increases as flow increases, slowing down the network as a whole, and hence, the original, low cost path becomes high cost. An example network is offered later in this section to illustrate the user equilibrium model.

The System Optimal Model

The system optimal model describes the network when it is working as efficiently as possible. If the total travel cost of a given network is minimal, then the network flow, $\bar{v}$, is at *System Optimal* (SO). Mathematically, a network $G(N,A)$ is at SO with $\bar{v}$ being the system optimal flow if $\bar{v} \in V$ satisfies

$$s(\bar{v})^T \bar{v} = \min\{s(v)^T v \mid v \in V\}.$$

In the SO model, the network, as a whole, is operating the most efficiently. A network at SO is rarely at UE as well. This is because some travelers may pay higher costs than they would in UE, as SO minimizes the average cost per user.

The Tolled User Equilibrium Model

Imposing tolls on a network adds a monetary component to the cost function in terms of time. The new cost for each arc, composed of time and money, is called the *generalized* cost. Let β_a denote the toll on arc a, then the *generalized* cost for arc a is expressed as $(s_a(v_a) + \beta_a)$. If a traveler has the choice between two paths where Path 1 takes less time than Path 2, but Path 1 also costs money, the traveler may be more likely to take Path 2 depending on how much the toll on Path 1 is. By adding a toll vector β to a network, the user equilibrium, $\tilde{v}$ goes from

$$s(\tilde{v})^T (v - \tilde{v}) \geq 0, \quad \forall v \in V$$

to

$$(s(\tilde{v}_\beta) + \beta)^T (v - \tilde{v}_\beta) \geq 0, \quad \forall v \in V,$$

where the new equilibrium $\tilde{v}_\beta$ is called the *Tolled User Equilibrium* (TUE).

The TUE model describes the network behavior where tolls can be added such that $\tilde{v}_\beta = \bar{v}$, i.e., the tolled user equilibrium reproduces the system optimal flow. In other words, there exist tolls that allow the network to operate so that when every user does what is best for oneself, he/she is also doing what is best for the system.

Toll Pricing Problems

A toll vector is said to be valid if it causes TUE to equal SO. To find the set of all valid toll vectors for a given network, refer to Hearn and Ramana (1998). It states that a toll vector is valid if there exists a ρ vector for each O-D pair *k* satisfying the following:

$$s(\bar{v}) + \beta \geq U^T \rho^k, \ \forall k \in K$$
$$(s(\bar{v}) + \beta)^T \bar{v} = \sum_k b_k^T \rho^k.$$

These are essentially the Karush-Kuhn Tucker (KKT) conditions that ensure $\tilde{v}_\beta = \bar{v}$. Moreover, from this set of valid toll vectors, a traffic planner can select an optimal toll for any specific objective such as minimizing total revenue or in the case of this paper, minimizing the total number of toll booths.

An Illustrative Example

Figure 1: An Example Network

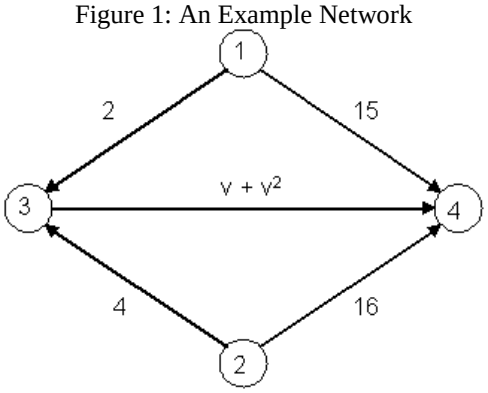
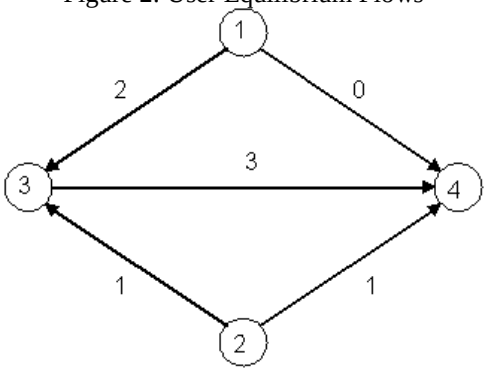


Figure 1 is a 4-node, 5-arc network with the given costs. Note that the cost on the arc connecting nodes 3 and 4 is dependent on the flow. The cost vector for this network is $s(v) =< 2, 4, 16, 15, v_5 + v_5^2 >$. There are two O-D pairs: 1 to 4 and 2 to 4. The demand on each is 2 cars.

Figure 2: User Equilibrium Flows



The network is at UE when *v* =< 2, 1, 1, 0, 3 >, as illustrated in Figure 2. To verify the flow is in UE, first calculate the new cost vector *s*(*v*) =< 2, 4, 16, 15, 12 >. Because, for the first O-D pair, the cost for path 1-4 is 15 and the cost for path 1-3-4 is 14, path 1-4 is not utilized. For the other O-D pair, the cost for both paths is 16, so they are both utilized. Note that the total travel cost for the system is 60 units of time.

Figure 3: System Optimal Flows

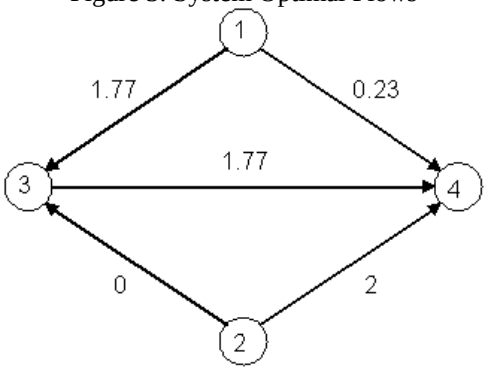
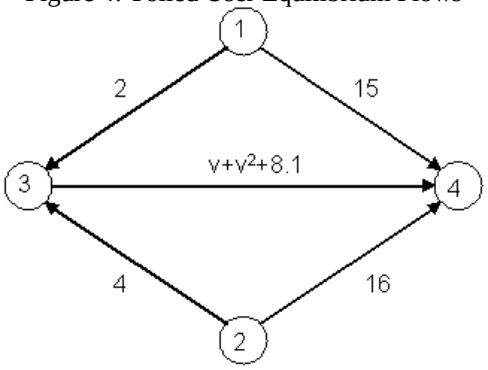


Figure 3 illustrates the network flow at SO. The total travel cost for this flow on the network is approximately 47.67 units of time. This is the minimum travel time for the whole system.

Figure 4: Tolled User Equilibrium Flows



Finally, in Figure 4, a toll of 8.1 has been added to Arc 5. This toll raises the *generalized* costs for paths 1-3-4 and 2-3-4 such that when the network is at SO, the travel cost for path 1-4 is equal to the cost on path 1-3-4, so this O-D pair is at UE. Furthermore, the travel cost on 2-3-4 becomes larger than the travel cost on path 2-4 so this O-D pair is at UE as well. Since both O-D pairs satisfy UE, the new TUE is the same as the SO.

The Minimum Toll Booth Problem (MINTB)

As stated previously, the MINTB objective is to place the fewest number of toll booths on a network so that when the system is at TUE, it is also at SO. Mathematically, let *y* be a binary vector which corresponds to β so that

$$y_a = \begin{cases} 0 & \text{if } \beta_a = 0 \\ 1 & \text{if } \beta_a > 0 \end{cases}.$$

Then, the MINTB problem can be formulated as:

$$
\begin{aligned}
&\min \sum_{a \in \mathrm{A}} y_a \\
&\text{s.t. } s(\bar{v}) + \beta \geq U^T \rho^k, \ \forall k \in K \\
&\quad (s(\bar{v}) + \beta)^T \bar{v} = \sum_k b_k^T \rho^k \qquad (1) \\
&\quad 0 \leq \beta_a \leq M y_a, \ \forall a \in A \\
&\quad y_a \in \{0,1\}, \quad \forall a \in A,
\end{aligned}
$$

where M is an arbitrarily large constant. Note that the first two constraints ensure β is a feasible (valid) toll vector and the third condition states that y_a=1 when β_a>0 and y_a=0 when β_a=0.

The MINTB problem is a mixed integer linear programming problem. In Bai (2004), the problem is shown to be NP-complete by reducing the partitioning problem, an NP-complete problem, to MINTB in polynomial time. Consequently, our focus in this paper is to develop heuristic algorithms to solve MINTB. In particular, we choose to study the genetic algorithm because it is well known for converging to global optimal solutions.

THE GENETIC ALGORITHM FOR MINTB (GAMINTB)

In the past few decades, genetic algorithms have become an increasingly popular heuristic method for problem solving. As one of several evolutionary optimization methods, the genetic algorithm (GA) uses ideas from natural selection to evaluate, breed, and filter through random solutions to obtain an optimal solution after many generations. Most genetic algorithms follow the basic steps described below.

1. *Initialization*: the algorithm begins with randomly generating an initial population of strands, often called chromosomes. Each strand is designed based on certain parameters which can be modified by the user, including the population size and the length of each strand.

2. *Evaluation*: this portion of the algorithm rates each chromosome by assigning it a fitness score, which reflects how well the decoded strand satisfies the constraints and optimizes the objective function. The strands are then ranked based on their respective fitness scores. Note that the means to deal with an infeasible solution include merely accepting it, penalizing it by lowering its fitness score, or completely eliminating it.

3. *Selection*: this process determines which strands from a population will be allowed to reproduce and create the next generation. There are a variety of ways to determine which strands move on, but most involve assigning each strand a probability. This selection probability can be uniform, but it is generally weighted in order to increase the likelihood that the fittest solutions will reproduce.

4. *Alteration*: the most popular methods of alteration are crossover and mutation. The crossover process is designed to take the traits from a set of parents, and pass them on to a new set of (ideally stronger) offspring. Although there are many ways to implement it, crossover typically involves taking a set of alleles from one parent, and switching them with the alleles of the other parent strand. Some forms of crossover produce two offspring, while others elect to produce one child, and allow one of the

parent strands to continue on to the next generation. The mutation process, which ensures diversity among the population, usually involves randomly changing some of the numerical values in the offspring.

5. *Termination*: after reproducing, most genetic algorithms are designed to cycle through the evaluation, selection, and alteration using the new and altered strands as the new parent population. Genetic algorithms stop after a specified number of generations.

Next we present the customized genetic algorithm for the MINTB problem (GAMINTB). The algorithm begins by generating a population of size N chromosomes, which are used to represent the presence of a set of toll locations. The strands are of length $|A|$, where $|A|$ is equal to the number of arcs, and they are indexed so that the ith gene corresponds to the ith arc on the network. Each strand, y, is binary, where a 0 represents an arc that does not have a toll booth, and a 1 represents an arc that has a toll booth. For example, $y = (0, 1, 0, 0, 1)$ would represent a 5-arc network with tolls on arcs 2 and 5. The number of chromosomes can be changed in our program by modifying the value of N. The algorithm uses an initialization method comparable to Zhang's (Zhang, 2003) in order to prevent bias among the strands. A random number generator creates an $|A|$ length strand of random numbers between 0 and 1. These numbers are then rounded to the nearest integer (0 or 1). The process is repeated until the number of strands is equal to the specified population size N.

After the initial population of toll locations is produced, it must be evaluated in some way. The y vectors are ranked based on two criteria: feasibility and the number of toll booths. A y vector is feasible if there exist corresponding β and ρ satisfying the constraints in problem (1). For evaluation, the y's are divided into feasible and infeasible groups and then each group is ranked according to the fewest number of toll booths required. For the final rankings, they are regrouped so that feasible solutions are ranked higher than all of the infeasible solutions. To decide on feasibility, the GAMINTB keeps track of βs and ρs for each y. It is important to consider infeasible solutions because some of them may be very similar to the optimal solution, and might only require a slight alteration in order to become feasible. Because of the elaborate alteration process in the GAMINTB, the good qualities of an infeasible solution may be passed down. On the other hand, the ranking process makes the probability very low for infeasible solutions to continue on from one generation to the next.

After the chromosomes are ranked, a set of parents is chosen to produce the next generation. In nature, 'survival of the fittest' typically applies to reproduction. Although the strongest individuals have the best chance of survival, they are not the only ones that reproduce. Rather than settling on a uniform selection probability, GAMINTB bases a strand's selection probability on its ranking (fitness score). It uses a weighted selection probability comparable to the one used by Sumalee (Sumalee, 2004). The weighted probability of selection for a strand of rank i is

$$P(i) = \frac{2((N+1)-i)}{N(N+1)}, \qquad (2)$$

where N is the population size. Thus, the likelihood of choosing the strongest strand (ranking index $i = 1$) for the breeding process is $2/(N+1)$, while the probability of choosing the weakest individual (ranking index $i = N$) is $2/(N(N+1))$. After the selection probability has been determined for each strand, the genetic algorithm uses a 'roulette

wheel' selection process to determine the parents (See e.g., Sumalee, 2004). Each slot on the roulette wheel is assigned an interval between 0 and 1, and the upper and lower bounds are assigned by cumulatively stacking the selection probabilities in (2). A random number generator is then used to select the strands that will produce the subsequent generation. If the random number is between the bounds of a specified slot, that slot's respective chromosome will be chosen as a parent. The algorithm picks the nearest even integer to ($R \cdot N$) parents, where *R* is the reproduction rate defined by GAMINTB.

Once the parents are selected, they are randomly paired to create offspring. Since the GA relies on the passing down of good traits from generation to generation, each pair of parents should pass on each shared allele to one child. Because each *y* chromosome is a binary strand, each child can only receive a 0 or 1 for each unshared parent allele. The GAMINTB objective is to minimize the sum across the strand, so giving each child a 0 for each unshared parent allele seems logical. However, when implementing this crossover method, the GAMINTB converges to a strand of 0's, which is most likely infeasible, and decreases the variety of solutions. Giving the child a 1 in place of the 0 results in equally bad solutions. Since a critical feature of the GAMINTB is randomness, each child is given a random binary number wherever the parents disagree.

For mutation, the GAMINTB selects the nearest integer to ($M \cdot N$) chromosomes randomly from the entire population pool, where *M* is the rate of mutation. Being that each *y* chromosome is a binary strand, mutation is merely changing one randomly selected allele from 0 to 1 or 1 to 0 for a chromosome that is selected.

Research done by Ahuja, Orlin, & Tiwari (2000) proposes immigration as a useful tool to adding diversity among the strands. The idea of immigration is to only breed a percentage of the next generation, and fill the remaining slots with randomly generated strings. The GAMINTB incorporates this idea by replacing those strands not selected for reproduction with entirely new strands, following the method used in the initialization phase.

COMPUTATIONAL RESULTS

The purpose of our computational study is to validate GAMINTB through small example networks, as well as to investigate how parameter settings affect the performance of GAMINTB. In this section we present the results of GAMINTB tested for performance and implementational comparisons over six small example networks. The six examples are built upon two basic network structures, the 4-node diamond structure and the 5-node split structure shown in Figures 5 and 6, respectively.

The first three example networks were derivations of the 4-node diamond and the last three were derivations of the 5-node split. The demand for each network was randomly selected, while the cost function vector for each was decided by the Bureau of Public Roads (BPR) cost function $s_a(v_a) = (T_a + 0.15(\frac{v_a}{C_a})^{P_a})$. Here $s_a(v_a)$ is the cost function, v_a is the variable traffic flow on arc *a*, T_a is the free travel time constant, C_a is the capacity, and P_a is the delay time factor for arc *a*. For each network, the constants

were randomly selected from uniform distributions as follows: $T_a \in [1, 10]$, $C_a \in [10, 25]$, $P_a \in \{0, 1, 2, 3, 4\}$.

Figure 5: A 4-node 5-arc network

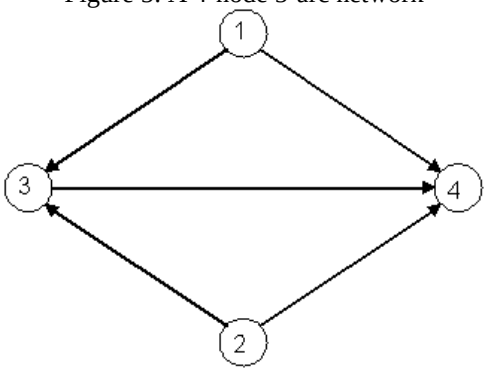
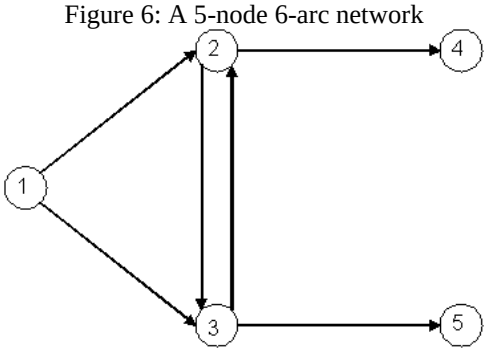


Figure 6: A 5-node 6-arc network



The GAMINTB was programmed using Fortran, and the tests were run on a computer platform with a 2.6 GHz Pentium 4 processor and 512 Mb of RAM. To ensure the GAMINTB converged to the optimal solution, LINGO 7, an NLP solver was used to solve MINTB. For each example network, the GAMINTB converged to the same solution produced by LINGO. In addition, LINGO was also used as the LP solver for GAMINTB to determine the feasibility for binary vector *y* with respect to constraints in problem (1).

The *quality* of the GAMINTB is defined as the percentage of the final population that is the optimal solution. It can be used as a measure of both the convergence rate of the GAMINTB and the diversity created within the GAMINTB. The algorithm performed well if the quality fell between 40% and 60%. A solution quality in this range indicates that a clear winner was discovered, yet a sufficient amount of diversity was introduced throughout the algorithm. Note that 100% is not considered to be the best quality because it could indicate that the algorithm only terminates at a local optimum.

We first conduct the "implementational comparison test" to compare the quality of various levels of immigration and mutation, and quality differences between implementing uniform and weighted selection. This test demonstrated the effectiveness of each implementational change according to the GA's quality and aided in the selection of the rate of reproduction, which determined the immigration rate, the percentage of chromosomes to mutate, and whether to implement uniform or weighted probabilities in the selection of parents.

The test was run as follows: for each example network, the GAMINTB was run with mutation alone, immigration alone, and with both; the tested rates of alteration for

each scheme were 20%, 30%, and 40%. The initial population size was 100 and the generation number was 20. The selection probability was weighted. To show the effects of mutation and immigration on the convergence rate, the qualities for all GAMINTB runs were recorded and summarized in Table 1.

Table 1: Comparing the Effects of Mutation and Immigration

Network	The Percentage of Optimal Solutions in The Final Generation									
	Mutation Rate			Immigration Rate			Rate of Both			Neither
	20%	30%	40%	20%	30%	40%	20%	30%	40%	0%
1	82%	70%	62%	61%	30%	31%	56%	11%	3%	0%
2	79%	74%	73%	75%	50%	32%	59%	37%	15%	100%
3	82%	74%	64%	66%	43%	27%	62%	30%	13%	100%
4	80%	65%	64%	61%	37%	10%	43%	22%	8%	0%
5	82%	76%	67%	70%	38%	35%	45%	12%	14%	100%
6	82%	69%	63%	64%	41%	8%	42%	16%	10%	100%

Table 1 makes it clear that immigration is much more effective in slowing the convergence rate of the GAMINTB. From another perspective, immigration allowed for more variety among optimal solutions into the final population to minimize any fears that the algorithm may prematurely converge to some local optimal solution. The results looked very promising with the standard immigration rate of 30%. On the other hand, mutation was not nearly as effective. It did slow the convergence rate, but only after the mutation rate was raised to an absurdly high level. (A typical mutation rate usually hovers between 2% and 5%). In most cases, the effect of mutation on the genetic algorithm is minimal, thus it is not essential to the success of our genetic algorithm. Finally, utilizing crossover alone led to an extremely high convergence rate for the GAMINTB, which implied that the solution may have converged prematurely.

Strong arguments have been made in favor of determining the parents based on a uniform probability or on a weighted probability. Ahuja et al. (1993) argue that biasing selection in favor of the fitter individuals leads to a faster convergence, and that the use of uniform probability provided them with better results (Ahuja et al., 1993). Conversely, Sumaleee (2004) used a weighted probability with a "selection bias," which could focus on choosing the better strands. We tested both selection probability schemes for GAMINTB with the population size and generation number being 100 and 20, respectively. The quality results for GAMINTB (in percentages) are listed in Table 2.

Table 2: Comparing Uniform and Weighted Probability

Network	Uniform	Weighted
1	7%	30%
2	3%	50%
3	11%	43%
4	1%	37%
5	7%	38%
6	5%	41%

Comparing the uniform and weighted probability methods, Table 2 suggests that weighted probability converges at a much higher rate. However, the convergence rate is not high enough to generate much alarm for premature termination. On the other hand, some of the uniform cases experienced rather low convergence rates and were dangerously close to not converging at all. Thus, it can be concluded that GAMINTB is much more effective when using a weighted selection probability. The benefits of uniform probability may have been limited by the fact that the algorithm was only tested on smaller networks. This will be investigated in our future work testing larger networks.

Our second computational test is the "Performance Test." The goal is to demonstrate the effect of population size and generation number on run time and quality. The experiment was conducted as follows: the mutation rate (percentage of population to be mutated with each generation) and immigration rate (percentage of population to be filled by immigration with each generation) were set at zero and thirty percent, respectively and the selection probability was weighted. The first part of the experiment fixed the number of generations at 100 and ran the GAMINTB with population sizes of 500, 1000, 1500, and 2000; the second part of the experiment fixed the population size at 100 and ran the GAMINTB with generation numbers of 500, 1000, 1500, 2000. For each trial, the CPU time and quality of the GAMINTB were recorded. Table 3 displays the CPU time information when varying the population and generation sizes. Additionally, Table 4 provides the quality information when varying the population and generation sizes.

Table 3: CPU time Comparison (Population Size vs. Generation Size)*

Network	Population Size				Number of Generations			
	500	1000	1500	2000	500	1000	1500	2000
1	1.25	4.16	8.83	15.04	0.56	1.16	1.68	2.26
2	1.34	4.37	8.97	15.33	0.64	1.29	1.92	2.55
3	1.38	4.41	8.9	15.05	0.72	1.43	2.15	2.84
4	1.04	3.82	8.06	14.06	0.35	0.7	1.05	1.38
5	1.47	4.71	9.59	16.27	0.74	1.47	2.21	2.92
6	1.45	4.67	9.4	16.06	0.71	1.42	2.14	2.84

*All CPU times are in seconds.

Table 4: Quality Comparison (Population Size vs. Generation Size)

Network	Population Size				Number of Generations			
	500	1000	1500	2000	500	1000	1500	2000
1	38%	43%	38%	39%	24%	38%	38%	37%
2	41%	43%	41%	43%	35%	34%	35%	50%
3	38%	46%	44%	41%	18%	30%	46%	43%
4	34%	41%	38%	39%	35%	35%	40%	37%
5	21%	24%	15%	18%	14%	10%	19%	15%
6	36%	38%	36%	32%	39%	28%	36%	39%

Examining Table 3, we find that the CPU time grows much more rapidly when increasing the population size than increasing the generation size. Thus, it may be better to increase the number of generations before increasing the population size.

On the other hand, Table 4 shows the quality gains from increasing population

and generation sizes. Note that in Table 4 after raising the population size above 1000, the quality began to decrease, while increasing the number of generations the quality showed an upward trend for the six example networks. Because of these trends, it is better to increase the number of generations than it is to increase the population size as the numbers get very large.

Overall, as illustrated in Figures 7, 8, 9 and 10, increasing the number of generations increases the total CPU time less rapidly and still produces similar quality results when compared to increasing the population size. Each of these figures demonstrates the following performance data calculated on the six networks. Networks 1 through 6 are marked by $\Diamond, \square, \Delta, \times$, and $*$, respectively.

Figure 7

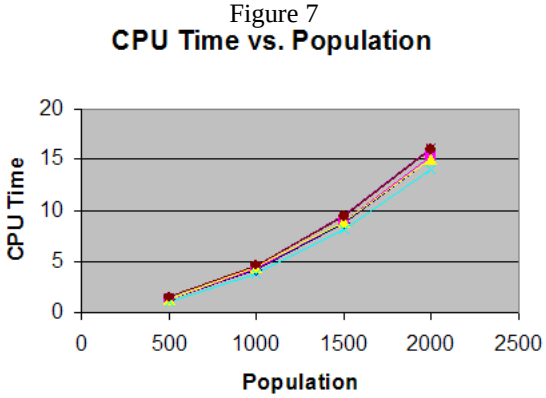


Figure 8

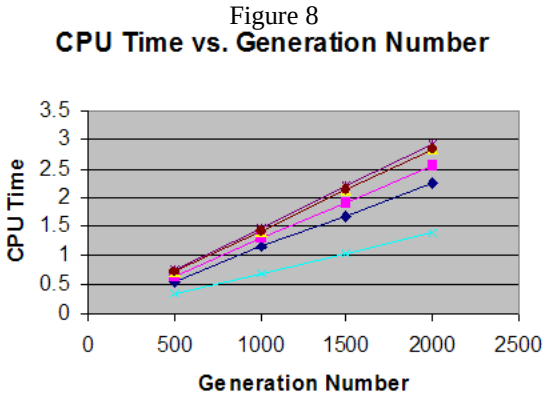


Figure 9

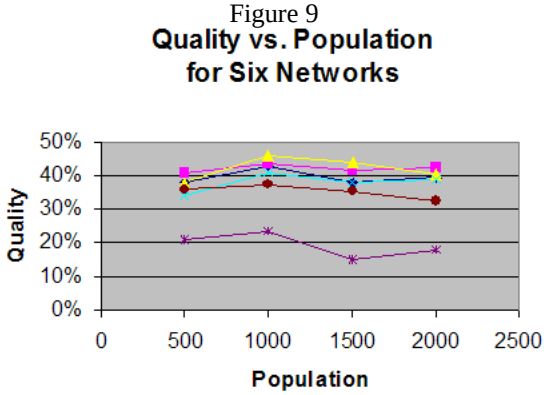
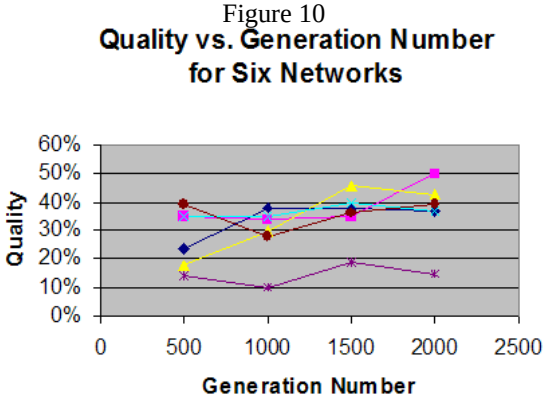


Figure 10



AN ALTERNATIVE IMPLEMENTATION

An alternative approach for GAMINTB without having to check the feasibility for each binary vector *y* is to relax the first two conditions in problem (1) and add them into the objective function as a penalty. This change will resolve the issue of dividing the *y*'s into feasible and infeasible groups. Instead, they will all be rated by the number of toll booths, but they will also be penalized for not satisfying the toll constraints. To implement this method, the objective function will be composed of three parts. The first is the total number of toll booths, which is our original main objective. The second is adding on a scalar of $\sum_k (U^T \rho^k - (s(\bar{v}) + \beta))^2$ if $U^T \rho^k > (s(\bar{v}) + \beta)$. This is the penalty for not satisfying the first constraint in (1). The last component of the objective function is adding on a scalar of $((s(\bar{v}) + \beta)^T \bar{v} - \sum_k b_k^T \rho^k)^2$, which penalizes *y* for not satisfying the second constraint in (1).

There are issues associated with this method. For example, the penalty method still requires accurate values for β's and ρ's, which are difficult for the GAMINTB. Secondly, the determination of appropriate penalty factors may require solving additional nonlinear programs iteratively, which can be very computationally expensive. Many other challenges with this penalty method need adequate exploration and we leave them for future research.

CONCLUSIONS

The purpose of this paper is twofold. One is to apply the genetic algorithm to the MINTB problem and develop a general framework for the methodology. The other is to validate the so developed method through small numerical examples. In response, the paper presents a customized genetic algorithm, GAMINTB, as a heuristic method for solving the minimum toll booth problem; and the algorithm is shown to be effective at solving MINTB for six small networks, all matching the optimal solutions generated by an NLP solver LINGO 7. Furthermore, computational tests suggest the following

guideline for using GAMINTB. First, when searching for a better solution, the number of generations should be increased before increasing the population size, due to a lower CPU time and a higher projected convergence rate. Second, the use of immigration is much more effective in promoting diversity than the traditional mutation process. Third, the use of weighed probability in the selection process is better than the uniform probability for solving MINTB. However, whether or not the uniform probability would still be ineffective for larger networks needs further investigation. Finally, other streams of future research include testing and applying GAMINTB for larger networks, and considering the penalty approach for a variation of the GAMINTB.

REFERENCES

Ahuja R.K., Orlin J.B., & Tiwari A. (1993). A Greedy Genetic Algorithm for the Quadratic Assignment Problem. *Computers and Operations Research,* 27, 917-934.

Bai L. (2004). Computational Methods for Toll Pricing Models. Unpublished Ph.D. Dissertation, Department of Industrial and Systems Engineering, University of Florida.

Bai L., Hearn D.W., & Lawphongpanich S. (2008). A Heuristic Method for the Minimum Toll Booth Problem. Working Paper.

Beckmann M., McGuire C., & Winston C. (1956). *Studies in the Economics of Transportation.* Yale University Press, New Haven, CT.

DeCorla-Souza P. (2003). Report to the Transportation Research Board Joint Subcommittee on Pricing. Presented to *82nd Annual Transportation Research Board Meeting 2003.* Washington, D.C.

Florian M., Guélat J., & Spiess H. (1987). An Efficient Implementation of the PARTAN Variant of the Linear Approximation Method for the Network Equilibrium Problem. *Networks,* 17, 319-339.

Florian M., & Hearn D.W. (1995) Network Equilibrium Models and Algorithms. In Ball M.O., Magnanti T.L., Monma C.L., & Nemhauser G.L. (Eds.). *Chapter 6 of Handbooks in Operations Research and Management Science (8): Network Routing.* North-Holland, New York.

Hearn D.W., & Ramana M. (1998). Solving Congestion Toll Pricing Models. In Marcotte P, & Nguyen S. (Eds.), *Equilibrium and Advanced Transportation Modeling,* (pp. 109-124). Kluwer Academic Publishers.

LeBlanc, Morlok E.K., & Pierskalla W.P. (1975). An Efficient Approach to Solving the Road Network Equilibrium Traffic Assignment Problem. *Transportation Research,* 9, 309-318.

Pigou A.C. (1920). *The Economics of Welfare. MacMillan.* New York.

Schrank D., & Lomax T. (September 2007). Urban Mobility Report 2007. Texas Transportation Institute. from http://mobility.tamu.edu.

Shepherd S., & Sumalee A. (2004). A Genetic Algorithm Based Approach to Optimal Toll Level and Location Problems. *Networks and Spatial Economics*, 4, 161-179.

Sumalee A. (2004). Optimal Road User Charging Cordon Design: A Heuristic Optimization Approach. *Computer-Aided Civil and Infrastructure Engineering*, 19, 377-392.

Todd J. (2005). Duke Student Math Aims to Alleviate Tollbooth Lines. *Duke University News and Communications*, June 2005. from http://www.dukenews.duke.edu/2005/06/tollbooths.html.

Zhang X. (2003). Optimal Road Pricing in Transportation Networks. Unpublished Ph.D. Dissertation. Department of Civil Engineering, Hong Kong University of Science and Technology, Hong Kong.