Link and Path-Based Traffic Assignment Algorithms: Computational and Statistical Study

Article *in* Transportation Research Record Journal of the Transportation Research Board · January 2002

DOI: 10.3141/1783-11

4 authors, including:

Der-Horng Lee
National University of Singapore
117 PUBLICATIONS **2,402** CITATIONS

SEE PROFILE

Yu Nie
Northwestern University
81 PUBLICATIONS **1,410** CITATIONS

SEE PROFILE

Anthony Chen
Utah State University
181 PUBLICATIONS **3,829** CITATIONS

SEE PROFILE

Paper No. 02-2900

Link- and Path-Based Traffic Assignment Algorithms

Computational and Statistical Study

Der-Horng Lee, Yu Nie, Anthony Chen, and Yuen Chau Leow

The computational performance of five algorithms for the traffic assignment problem (TAP) is compared with that of mid- to large-scale randomly generated grid networks. The applied procedures include the Frank–Wolfe, PARTAN, gradient projection, restricted simplicial decomposition, and disaggregate simplicial decomposition algorithms. A statistical analysis is performed to determine the relative importance of various properties (network size, congestion level, solution accuracy, zone–node ratio) of the traffic assignment problem for the five selected algorithms. Regression models, which measure central processing unit time and number of iterations consumed by each algorithm using various factors and their combinations, are derived to provide a quantitative evaluation. Ultimately, the findings of this research will be useful in guiding transportation professionals to choose suitable solution algorithms and to predict the resulting algorithm performance in TAPs.

The traffic assignment problem (TAP) has been of interest to transportation professionals. This interest is based mainly on the need by traffic assignment models and associated algorithms in network design, network capacity analysis, road network operation improvement, optimal routing application, and so on. It is known that the traffic pattern satisfying the Wardrop principles (*1*) in a transportation network can be obtained by solving an optimization problem, which involves the travel cost of each link dependent only on the flow on that link itself. In past decades, the development of faster and more efficient algorithms has been one of the most often investigated research topics in transportation network modeling.

A number of algorithms have been proposed to solve the TAP, but little is known about their relative computational efficiency and the relative importance of the various factors such as network size, congestion level, number of zones, and required solution accuracy to the efficiency of the TAP because comprehensive comparison among the algorithms remains limited. Previous research has attempted to compare the computational and convergence performance of different algorithms and to explore their sensitivity with respect to the aforementioned factors in real and hypothetical problems (*2*–*7*). Although the relation between the computational efficiency of the algorithm and properties of the problem (network size, congestion level, solution accuracy, number of zones) plays a vital role in the algorithm evaluation, clarifying such a relation in a theoretical frame is fairly difficult. This relation is elaborated on here through statistical analysis of regression equations constructed by abundant computational results obtained from state-of-the-art and state-of-the-practice link- and path-based algorithms.

For this research a spectrum of computational runs was performed on a collection of link- and path-based algorithms for the TAP. Two path-based algorithms, gradient projection (GP) and disaggregate simplicial decomposition (DSD), and three link-based algorithms, Frank–Wolfe (F-W), PARTAN (PT), and restricted simplicial decomposition (RSD), are studied and tested using randomly generated grid networks to investigate the algorithm performance, which is measured in this research by central processing unit (CPU) time consumed and number of algorithm iterations while a prespecified accuracy is achieved. The primary consideration in this part of the study is whether the algorithm will solve the problem and how long it will take to reach a satisfactory solution within the desired accuracy. Note that computational performance is dependent on programming technique, computing environment, and algorithm efficiency. Thus, the code development for the algorithms used in this study employs the same data structures for storing network topology and paths. Whenever possible, the same subroutines, such as the shortest path and function evaluation codes, were used in all algorithms. All algorithms were implemented in FORTRAN, and all results were tested under the same computing environment.

A random grid network generator is developed to create the test networks. In this research, any computational run is characterized by the combination of network size, zone–node ratio, and congestion level, and 10 grid networks generated by different random seeds are employed and average output measurements are obtained in order to reduce the impact of randomness. Moreover, five accuracy levels are used in each run to examine the sensitivity of the algorithms.

In addition, to study the algorithm performance computationally, another notable endeavor and theme of this research is to elaborate quantitatively the connection that resides between the algorithm efficiency and properties of the TAP. To this end, a series of statistical analyses are conducted using the results generated from abundant computational runs. Correlation coefficients of different independent variables (network size, congestion level, zone–node ratio, accuracy level, and their combinations) are used to interpret their contribution to the required CPU time and the number of algorithm iterations. The findings are useful in guiding transportation professionals to choose suitable solution algorithms and to predict the resulting algorithm performance in the application of TAPs.

The next section describes the TAP and algorithms considered in this paper, followed by the presentation of computational experiments and results. The statistical analyses of the algorithms are then reported. Concluding remarks and needs for future research are summarized last.

D.-H. Lee, Department of Civil Engineering, National University of Singapore, BLK E1A, 07-16, 1 Engineering Drive 2, Singapore 117576, Republic of Singapore. Y. Nie, Department of Civil and Environmental Engineering, University of California, One Shields Avenue, Davis, CA 95616-5294. A. Chen, Department of Civil and Environmental Engineering, Utah State University, Logan, UT 84322-4110. Y. C. Leow, Land Transport Authority, 460 Alexandra Road, #19-00 PSA Building, Singapore 3757213, Republic of Singapore.

DESCRIPTION OF ALGORITHMS

A transportation network $G = (N, A)$ is given, where N and A are the sets of nodes and links, and each directed link $a \in A$ is associated with a positive travel time $t_a(x_a)$ as a function of link flow x_a. R is the set of origins and S is the set of destinations. For origin–destination (O-D) pair (rs), there is a given positive flow demand q_{rs}. Then the TAP in Wardrop's first principle (*1*) can be stated as follows:

$$\min z(x) = \sum_a \int_0^{x_a} t_a(w)dw \tag{1}$$

subject to

$$\sum_k f_k^{rs} = q_{rs} \qquad \forall k \in K_{rs}, r \in R, s \in S \tag{2}$$

$$f_k^{rs} \geq 0 \qquad \forall k \in K_{rs}, r \in R, s \in S \tag{3}$$

$$x_a = \sum_r \sum_s \sum_k f_k^{rs} \delta_{a,k}^{rs} \qquad \forall a \in A, k \in K_{rs}, r \in R, s \in S \tag{4}$$

where f_k^{rs} denotes the flow on path k between O-D pair rs, and K_{rs} is the path set between O-D pair rs. $\delta_{a,k}^{rs} = 1$ if link a belongs to path k; otherwise, $\delta_{a,k}^{rs} = 0$.

The F-W algorithm (*8*), also known as the convex combination method, is a conventional choice for solving the TAP (Equations 1–4). Frank and Wolfe designed this algorithm to solve the convex quadratic problem (*8*), and LeBlanc et al. first adopted it for solution of the TAP (*9*). The algorithm is easy to understand and simple to implement. Furthermore, the core storage needed is rather small. Both properties are advantageous considering the problem size that the F-W can handle. From the viewpoint of convergence speed, however, the efficiency of the F-W algorithm is not satisfactory because search directions generated by solving the linear programming subproblems tend to be perpendicular to the steepest descent direction as the optimal solution is approached.

The PT algorithm is a variation of F-W that tries to speed up the convergence (*10, 11*). It attempts to alleviate the zigzagging effect in F-W by performing another line search to adjust the search direction of F-W such that it is not orthogonal to the gradient of the objective function at the current flow pattern. After each normal F-W step, PT combines the current solution with the one before the last solution such that the objective function is minimized along the line joining the two. In this way, PT tends to prevent falling to constraint corners.

The RSD algorithm is a version of simplicial decomposition in which the number of retained extreme points is restricted to v (*12–14*). These extreme points are generated by a series of linearized subproblems and used to solve the master problem with a projected Newton method. The master problem is a reduced dimension of the original optimization problem. Hearn et al. have shown that the rapid convergence can be ensured if v is properly chosen and the decomposition is implemented with a second-order method (*12*).

The GP algorithm operates directly on the space of the path flows and can be viewed as a constrained version of the steepest descent or Newton method (*4*). The algorithm does not find auxiliary solutions that are at corner points of the linear constraint space. Instead, it makes successive moves toward the direction of the minimum of a Newton approximation of a transformed objective function that includes the demand conservation constraints. When the move to the minimum in the negative gradient direction results in an infeasible solution point, a projection is made to the boundary of the constraint set, which can be easily accomplished by making the associated path variable zero.

The DSD algorithm (*5*) is a simplicial decomposition approach based on Cartheodory's theorem (*15*). The method is also known as a path-based algorithm. Because the entire feasible set is actually a simple Cartesian product with respect to separate O-D pairs, it is possible to use the disaggregate simplicial decomposition by treating one convexity constraint in a disaggregate master problem for each O-D pair. When solving the master problem, DSD uses either a reduced gradient method (low accuracy level) or an approximate Newton's method (high accuracy level) with a set of paths produced by a column generation scheme.

PRIMARY EXPERIMENTS

Experiment Design

The code development for the algorithms adopts common data structures for storing the network topology and paths. All algorithms are coded in double precision FORTRAN-90, and the same subroutines, such as the shortest-path tree search and function evaluation, are used wherever possible through all codes of the algorithms. Results reported in this study were conducted on the SGI Origin2000 at the Computer Center of the National University of Singapore.

Two indexes are used in this research to evaluate the algorithm. They are CPU time t_c and number of iterations k_c required for an algorithm to reach the desired accuracy. CPU time is typically regarded as the primary measure for the algorithm's computational efficiency even if it is dependent on programming technique and the computer being used. The required number of iterations is used to represent the convergence performance of the algorithms.

Specifically, the performance of the algorithms is evaluated with respect to the following experiment factors: (*a*) network size n, (*b*) congestion levels c_e, (*c*) zone–node ratio r_z, and (*d*) solution accuracy levels ζ, in which c_e is calculated as follows:

$$c_e = \frac{\sum_r \sum_s q_{rs}}{\sum_a C_a} \times \sqrt{n} \qquad \forall r \in R, s \in S, a \in A \tag{5}$$

where n denotes the number of nodes and C_a denotes the capacity of link a.

The experiments are divided into three parts according to different combinations of experiment factors. In Part 1 of the study, the generated grid networks have 900, 1,600, 2,500, 3,600, and 4,900 nodes. The congestion level and zone–node ratio are 0.5 and 0.125, respectively. In Part 2, the congestion levels are 0.1, 0.3, 0.5, 0.7, and 0.9, with 2,500 nodes and a zone–node ratio of 0.125. In Part 3, the zone–node ratios are 0.03125, 0.0625, 0.125, and 0.25, with 2,500 nodes and a congestion level of 0.5. In all cases, 10 networks generated with different random seeds are used to reduce potential random errors. For each part of the experiments, algorithms are examined with five accuracy levels (0.00001, 0.0001, 0.001, 0.01, and 0.1).

Since travel time minimization is the objective of the TAP being considered, a lower value of the objective function implies that the model is closer to the optimal solution. Although a lower value of the objective function could be achieved by increasing the number of iterations, it is practically prohibitive to continue unlimited execution of the algorithm. In this research, F-W, PT, and RSD are

assumed to achieve a satisfying converged solution after 1,000 iterations, whereas the maximum iterations for the DSD and GP algorithms were set at 30. There are two reasons to limit the permitted iterations of DSD and GP to this comparatively small number. First, the storage requirement of a path-based algorithm is highly pertinent to the number of iterations. The more iterations that are performed, the more memory is required. Second, previous researchers have indicated that DSD and GP need much fewer iterations (around 10 to 20) to reach satisfied accuracy levels than do link-based algorithms (*2*, *4*, *5*).

The comparison among the algorithms is based on the best converged objective function value z^* ever obtained by all tested algorithms within a given maximum number of iterations, which is used as an anchor to locate five objective function values by a given accuracy level ζ according to

$$z_\zeta = z^* \times (1 + \zeta) \tag{6}$$

where z_ζ is the reference to locate the CPU time and number of iterations of all algorithms that give the same or approximately the same objective value. The speed of convergence for each algorithm can thereby be investigated with different ζ's.

Experiments were conducted through Parts 1 to 3. To show the tracks of convergence of the different algorithms, Figure 1 plots the relationship between the objective function value and iteration number, using a network of 2,500 nodes, with a zone–node ratio of 0.125, congestion level of 0.5, and generated with a random seed value of 50.

For the F-W and PT algorithms, the objective function value is greatly improved after the first few iterations. However, in many cases, the F-W and PT procedures failed to obtain the accuracy level specified by the anchor algorithm when the maximum iteration was reached. This result is consistent with the theoretical sublinear convergence rate of the F-W algorithm.

By performing another line search, PT improves F-W when the network size remains small and a low accuracy level is required. Unfortunately, its improvement is largely wiped out by the growth of network size, congestion level, and accuracy level. The long zigzagging tail still persists in PT.

Similar to the F-W and PT algorithms, the RSD procedure produces a significant decrease in the objective function value after the first few iterations. However, the convergence speed of RSD is much faster than those of F-W and PT because the size of the restricted master problem is kept small and solved with an efficient second-order method. In all results reported in this research, RSD obtained the lowest objective function value after 1,000 iterations, which is unmatchable by the other algorithms being considered.

Although DSD in general requires fewer iterations to achieve the similar objective function value obtained by the anchor algorithm, it consumes a substantially high amount of CPU time since its computational overhead required per iteration is much more than those of the other algorithms according to the computational runs exercised. For a network of 2,500 nodes, with a zone–node ratio of 0.125 and a congestion level of 0.5, DSD takes about 240 s of CPU time to complete one iteration, whereas GP only spends 16 s. This observation is not surprising since the line search in DSD has to be performed for each O-D pair in each iteration, which is computationally expensive. Moreover, solving the restricted master problem by the approximate Newton method is another time-consuming task in DSD.

GP outperforms other algorithms studied in this research in terms of the convergence speed when the accuracy level is relatively low, for example, no smaller than 0.001. For the network in Figure 1, the CPU time required by F-W and PT to satisfy the 0.01 accuracy level is 7 times more than that of GP. For RSD and DSD, the CPU time needed is about 3.5 times and 13 times more, respectively, than that for GP to obtain the same convergence level. This finding shows that GP can quickly arrive at the neighborhood of the optimal solution. However, it slows down significantly when further improvement is demanded. Note that GP takes 217 s to reach the accuracy level of 0.001, whereas the time consumed by RSD for the same problem is only 184.6 s. This characteristic of GP is mainly due to its linear convergence rate.

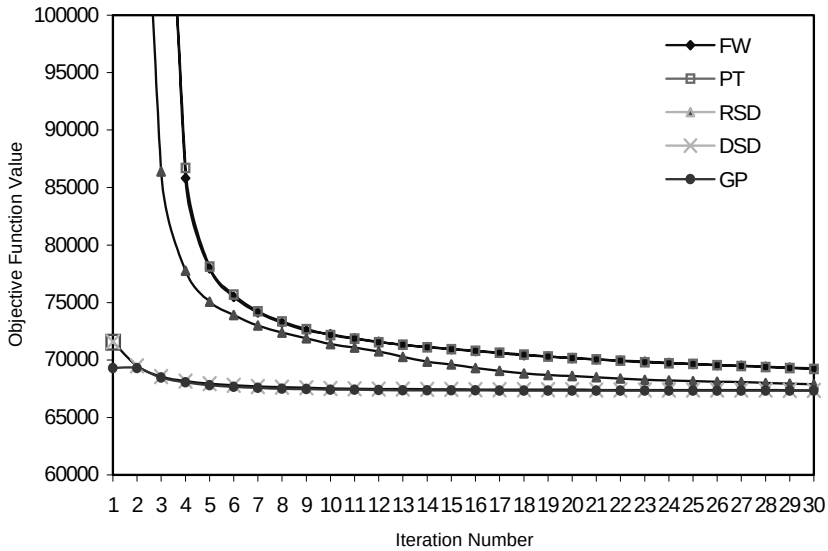

FIGURE 1 Objective function value versus iteration number for 2,500-node network: r_z = 0.125, c_e = 0.5.

Sensitivity Analysis

Network Size

The five algorithms are tested under various network sizes of 900, 1,600, 2,500, 3,600, and 4,900 nodes, and their effects upon number of algorithm iterations for an accuracy level of 0.001 are shown in Figures 2 and 3.

As expected, the computational time and number of iterations for all algorithms increase when the network size grows. F-W and PT perform similarly. As the network size increases, CPU time and number of iterations required for PT are increasingly more than those of F-W, which means that PT is more sensitive than F-W to the properties of network size. Another observation from Figure 2 is that the advantage of GP over link-based algorithms in terms of CPU time required for convergence actually decreases with increasing network size. As illustrated in Figure 2, CPU time spent by GP increases from 10.7 to 1,689 s (or 167.2 times) when the network size varies from 900 nodes to 4,900 nodes. The corresponding increases in CPU time for F-W, PT, and RSD are 115.6, 134.5, and 109.8 times, respectively. DSD especially seems to be rather sensitive to network size compared with the other algorithms being investigated. Path-based algorithms are more sensitive to network size because path operations like storage and comparison typically consume a considerable amount of time for larger networks. According to Figure 3, the number of iterations of F-W and PT grows significantly along with network size, followed by DSD. In order to reach a prespecified accuracy level, path-based algorithms such as GP and DSD are not sensitive to number of iterations consumed as the network size grows, as illustrated in Figure 3.

Congestion Level

To examine the effect of congestion, the demand level is incrementally increased from light (0.1) to heavily congested (0.9) traffic conditions. The resulting mean CPU time and number of iterations are summarized in Table 1. Figures 4 and 5 illustrate the performance of the algorithms with increasing congestion level at a convergence level of 0.001. Obviously, the computational overhead grows significantly for all algorithms when the network becomes more and more congested. For uncongested networks, most link flows may be in the range where link cost functions remain flat, which means that the shortest paths generated in consecutive iterations are likely to be identical. As congestion builds up, a little change in link flows may lead to a significant deviation of the link cost. Consequently, the network is unstable and needs more computation to equilibrate.

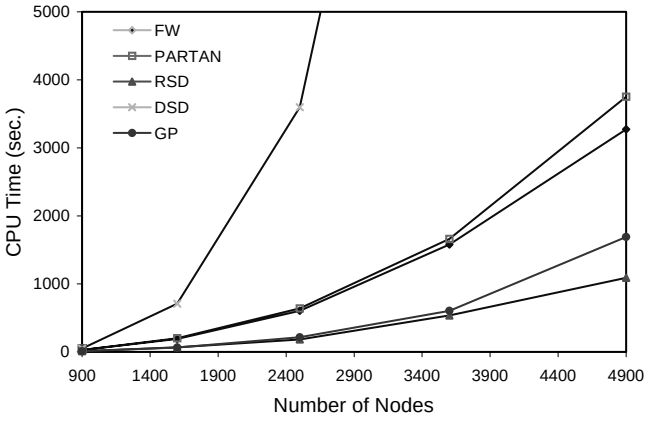

FIGURE 2 CPU time versus network size: $\zeta = 0.001$, $r_z = 0.125$, $c_e = 0.5$.

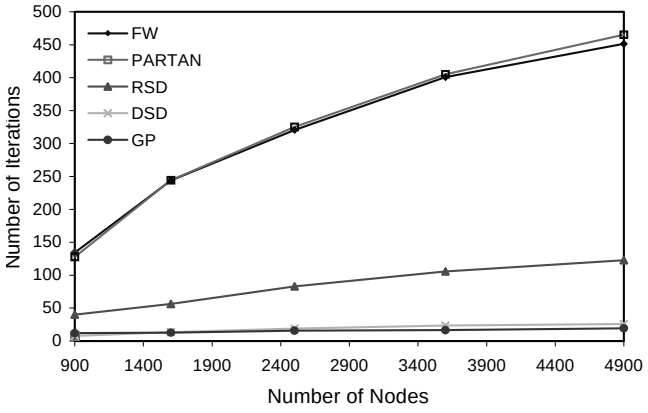

FIGURE 3 Number of iterations versus network size: $\zeta = 0.001$, $r_z = 0.125$, $c_e = 0.5$.

PT was observed in these results to be more sensitive to congestion level than F-W. For the network in Figures 4 and 5, PT drops behind F-W to reach the 0.001 accuracy level once the congestion level is larger than 0.5.

When the congestion level is very light, the performance of GP is better than that of the other algorithms being considered. However, RSD always outperforms GP once the congestion level becomes heavy (larger than 0.3). It seems that RSD is not as sensitive to congestion level as GP is.

The number of iterations of the path-based algorithms seems to be a little inertial to congestion level compared with those of their link-based counterparts. However, this finding does not mean that the path-based algorithms are advantageous for heavily congested networks in terms of computational efficiency considering the trade-off between the number of iterations and the computation overhead spent in each iteration. Unlike link-based algorithms, whose computational costs consumed per iteration keep nearly constant, path-based algorithms in general require much more effort to complete an iteration if the congestion in the network becomes heavier. These additional costs are mainly due to the expanded path set resulting from the increase in congestion level.

Zone–Node Ratio

The effect of density of O-D nodes is tested by varying the zone–node ratio, that is, $\frac{1}{32}$ (0.03125), $\frac{1}{16}$ (0.0625), $\frac{1}{8}$ (0.125), and $\frac{1}{4}$ (0.25). For a 50×50 grid network, these zone–node ratios correspond to 78, 156, 312, and 625 zones.

Figures 6 and 7 show the performance of the algorithms by varying the number of zones from 0.03125 to 0.25 of the nodes in the network. Obviously, all algorithms require more calculations for a larger number of O-D pairs. This finding is expected since each O-D pair is associated with a sequence of unavoidable calculations

TABLE 1 Correlation Coefficients for Number of Iterations (k_c) in Regression Model 2

Explanatory Variables	Algorithm					Avg. Correlation Coefficient
	FW	PARTAN	RSD	DSD	GP	
n	0.07	0.08	0.16	0.10	0.06	0.09
c_e	0.24	0.24	0.30	0.21	0.25	0.25
r_z	-0.02	-0.02	-0.05	0.00	-0.03	-0.02
ζ	0.89	0.89	0.77	0.90	0.90	0.87
n^2	0.07	0.08	0.15	0.09	0.06	0.09
$c_e^{\,2}$	0.20	0.20	0.28	0.18	0.18	0.21
$r_z^{\,2}$	0.00	-0.01	-0.03	0.01	0.00	-0.01
ζ^2	0.89	0.89	0.83	0.87	0.88	0.87
$n\zeta$	0.85	0.85	0.79	0.86	0.85	0.84
$c_e\zeta$	0.85	0.85	0.85	0.83	0.86	0.85
$r_z\,\zeta$	0.64	0.64	0.52	0.66	0.64	0.62

such as the shortest-path search in all algorithms. As shown in Figure 6, CPU time increases almost linearly with the number of zones for the link-based algorithms. However, the CPU time of path-based algorithms exhibits an exponentially increasing rate when the zone–node ratio grows.

It is interesting to note that in most cases tested in Part 3, the number of iterations for all algorithms decreases slightly when the number of zones increases. The interpretation of this finding could be that the tested algorithms experience some difficulty in convergence with a decrease in number of zones. Given the same trip demand, the traffic between each O-D pair will increase as the zone number is reduced. Therefore, in the vicinity of origins and destinations, the congestion level of links may grow extensively, and more computational efforts are triggered for the algorithm to converge.

STATISTICAL ANALYSIS

A statistical analysis is performed to quantify the relative importance of various factors in addressing the efficiency of TAP algorithms, in which a polynomial function is employed as an approximation because the shape of the true curvilinear response function is unknown. It may be viewed as a nonparametric approach to obtain information about the shape of the response function. The factors include

1. Network size n: 900, 1,600, 2,500, 3,600, and 4,900 nodes;
2. Congestion level c_e: 0.1, 0.3, 0.5, 0.7, and 0.9;
3. Zone–node ratio r_z: 0.03125, 0.0625, 0.125, and 0.25; and
4. Accuracy level ζ: 0.1, 0.01, 0.001, 0.0001, and 0.00001.

The polynomial models are described first, followed by a magnitude analysis of the effect of a given factor on the response variables (i.e., CPU time t_c and number of iterations k_c), which is in fact a quantitative version of the sensitivity analysis. The regression results are given last.

Description of Polynomial Models

Polynomial regression models are used to study the relation of response variables to several possible explanatory variables (the properties of the problem and their combinations) based on a sample of 120 randomly generated networks. These networks represent 12 typical combinations of the four basic explanatory variables, including network size (n), congestion level (c_e), zone–node ratio (r_z),

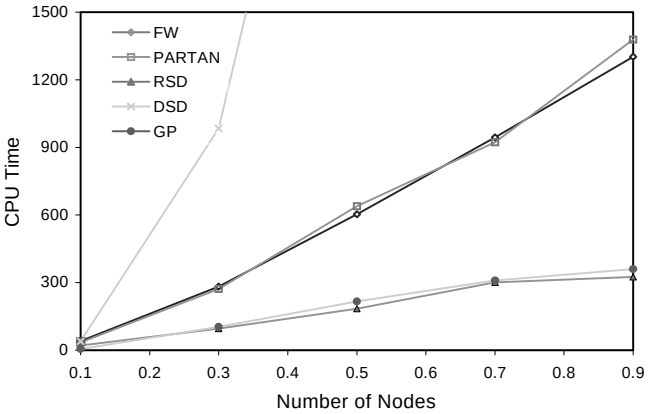

FIGURE 4 CPU time versus congestion level: ζ = 0.001, r_z = 0.125, n = 2,500.

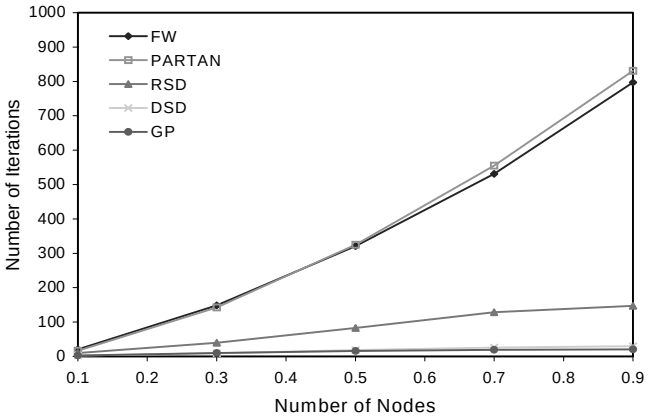

FIGURE 5 Number of iterations versus congestion level: ζ = 0.001, r_z = 0.125, n = 2,500.

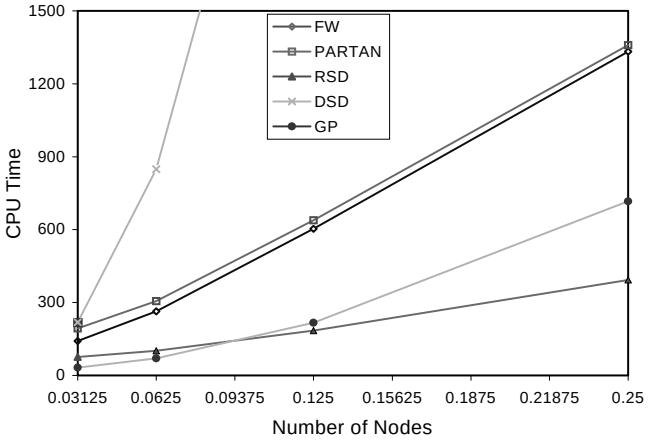

FIGURE 6 CPU time versus zone–node ratio: $\zeta = 0.001$, $c_e = 0.5$, $n = 2{,}500$.

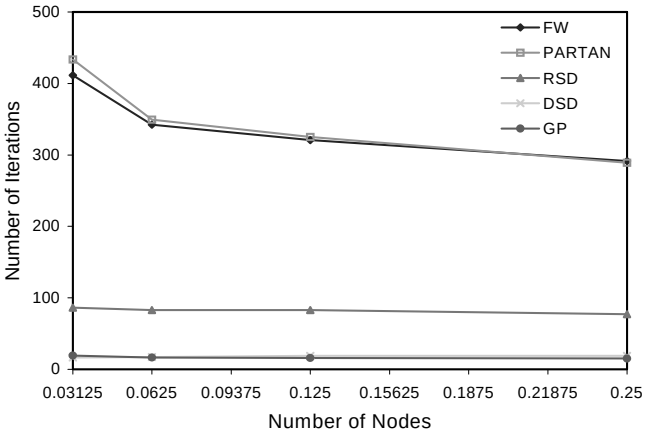

FIGURE 7 Number of iterations versus zone–node ratio: $\zeta = 0.001$, $c_e = 0.5$, $n = 2{,}500$.

and accuracy level (ζ). Note that a full factorial model ($5 \times 5 \times 4 \times 5$ combinations in this case) is not necessary because of the expensive computational cost. As will be shown later in this research, experiments with partial combinations have been able to produce satisfactory regression results.

Whenever necessary, simple reformulations were introduced to adjust the value of basic variables. This procedure was employed to avoid the possible singularity when matrix inverse operations were performed and to obtain better regression results. For instance, the square root of n was used instead of n to represent the network size.

Each of the basic explanatory variables was raised to the second power and presented as n^2, c_e^2, r_z^2, and ζ^2. Although the effects of the explanatory variables on the response variable are not addable, the effect of one explanatory variable depends on the levels of the other explanatory variables. Considering these effects, the regression model for the four explanatory variables, with the first and second order with linear effects on t_c and k_c as well as their interacting effects represented by cross-product terms, is as follows:

$$t_c = \beta_0 + \beta_1 n + \beta_2 c_e + \beta_3 r_z + \beta_4 \zeta + \beta_5 n^2 + \beta_6 c_e^2 + \beta_7 r_z^2 + \beta_8 \zeta^2 + \beta_9 n\zeta + \beta_{10} c_e \zeta + \beta_{11} r_z \zeta \quad (7)$$

$$k_c = \beta_0 + \beta_1 n + \beta_2 c_e + \beta_3 r_z + \beta_4 \zeta + \beta_5 n^2 + \beta_6 c_e^2 + \beta_7 r_z^2 + \beta_8 \zeta^2 + \beta_9 n\zeta + \beta_{10} c_e \zeta + \beta_{11} r_e \zeta \quad (8)$$

where $\beta_0, \ldots, \beta_{11}$ are regression coefficients.

Interpretation of Correlation Coefficients

A correlation coefficient shows the relationship of the response variable to one of the explanatory variables, holding constant for all other variables in the model. That is, the correlation coefficients measure the strength of the linear relationship between two variables after adjustment for the relationship involving all the other variables.

A correlation coefficient takes a value from –1 to +1. Values of +1 and –1 signify an exact positive and negative relationship between the variables, with a value of 0 indicating no relationship and values of –1 and +1 indicating perfect linear relationship. The condition with a value of 0, however, does not imply that no relationship exists since correlation does not measure the strength of curvilinear relationships. In Table 2, the correlation coefficients n and n^2 and ζ and ζ^2 are large in the regression model. This result implies that network size and accuracy level are highly correlated with CPU time, which

TABLE 2 Correlation Coefficients for CPU Time (t_c) in Regression Model 1

Explanatory Variables	Algorithm					Avg. Correlation Coefficient
	FW	PT	RSD	DSD	GP	
n	0.49	0.51	0.46	0.53	0.52	0.50
c_e	0.10	0.09	0.14	0.12	0.11	0.11
r_z	0.29	0.26	0.16	0.29	0.28	0.25
ζ	0.55	0.53	0.52	0.42	0.45	0.49
n^2	0.52	0.54	0.50	0.57	0.56	0.54
$c_e^{\,2}$	0.05	0.04	0.11	0.08	0.06	0.07
$r_z^{\,2}$	0.27	0.24	0.14	0.27	0.26	0.23
ζ^2	0.55	0.52	0.55	0.41	0.44	0.49
$n\,\zeta$	0.75	0.73	0.71	0.64	0.67	0.70
$c_e\,\zeta$	0.49	0.46	0.52	0.42	0.43	0.46
$r_z\,\zeta$	0.66	0.61	0.53	0.55	0.57	0.58

is consistent with the observations obtained earlier. In this model, the average correlation coefficient of c_e is about 0.11, which means that congestion level has a relatively weak effect on t_c. Note that a strong correlation between two variables may be caused by their dependence on a third variable. For instance, the correlation of n and ζ with t_c is medium, around 0.5, but the correlation coefficient of the interaction terms $n\zeta$ is about 0.70. This finding means that the interaction between t_c and n is highly related to ζ. If ζ is stringent, more CPU time is required as the network size is increased. Otherwise, as the network size is increased, the marginal increase in CPU time may not be high.

The correlation coefficients of DSD and GP to the four basic variables are close to each other because both of them operate on path space and thus are similar in their data storage strategy and algorithmic features. Network size is more significant for the CPU time of DSD and GP than for those of F-W, PT, and RSD. However, accuracy level is more important for the CPU time of F-W, PT, and RSD than for those of DSD and GP. Apparently, RSD is less affected by the zone–node ratio compared with the other algorithms.

In Table 1, the correlation coefficients of n, c_e, and r_z are about 0.09, 0.25, and -0.02 among the algorithms. The negative correlation of –0.02 may be attributed to the fact that there is a negative or almost zero relationship between zone–node ratio r_z and iteration number k_c; k_c is partially dependent on congestion level c_e but not much on network size n. Intuitively, all algorithms need more iterations for heavily congested networks since the equilibrium is harder to achieve in this case. The correlation coefficient of ζ is about 0.87, which is very high compared with those of the other variables. The number of iterations required is much larger when the accuracy level is tighter.

According to Table 1, the number of iterations of RSD is more dependent on network size, congestion level, and zone–node ratio but less dependent on accuracy level than in the other algorithms. The number of iterations of GP is less affected by the network size, whereas the number of iterations of DSD has no direct relationship with zone–node ratio (the correlation coefficient is zero).

Regression Results

The final coefficients of the polynomial regression equations for CPU time and number of iterations are given in Tables 3 and 4.

The polynomial model consists of explanatory variables presented at higher powers. Also, the interacting variables affect the shape of the response function. As a result, the regression coefficients β_1, β_2, β_3, and β_4 no longer indicate a change in mean response with a unit increase of the explanatory variable, with the other explanatory variables held constant at any given level. Hence, in this polynomial regression model, the effect of n, c_e, r_z, and ζ depends on the level of the other explanatory variables.

Take RSD as an example. In Table 3, the coefficients for n, c_e, r_z, and ζ are −206.6, −31.8, −42.2, and −2588.7, respectively. Their signs are negative, but that does not mean that increases in n, c_e, r_z, and ζ produce decreases in CPU time. It is important to note that the main-effect coefficients have a particular meaning in a model with interactions. In general, whenever there is a product term in a regression model, the statistical significance of the main effect of the four variables to the CPU time (t_c) becomes different.

The last row in Tables 3 and 4 shows the coefficient of multiple determinations R^2 for each regression analysis, which is used to illustrate the adequacy of the fitted regression model. It is a quantity to indicate the proportion of the total variation in CPU time and iteration number being explained by the fitted model. In the first polynomial regression model, for CPU time, the average value of R^2 is 0.89, or 89% for each algorithm. This result demonstrates that using n, c_e, r_z, and ζ to explain CPU time yields a reduction in the sum of squared explanatory errors compared with using only the mean. In the second polynomial regression model, for number of iterations, the average value of R^2 is 0.91, or 91% for each algorithm. This result means that network size, congestion level, zone–node ratio, and accuracy level explain 91% of the variation in iteration number for running the TAPs. The R^2 reported in Tables 3 and 4 indicate that the regressions explained by the models are significant. However, this finding did not rule

TABLE 3 Regression Coefficients for CPU Time (t_c) in Regression Model 1

Regression Coefficient	Corresponding Explanatory Variable	Algorithm				
		FW	PARTAN	RSD	DSD	GP
β_0	-	7723.9	10765.1	7419.5	82580.1	4800.4
β_1	n	-259.0	-365.0	-206.6	-2826.0	-158.2
β_2	c_e	211.5	192.4	-31.8	-146.7	15.6
β_3	r_z	-97.9	-121.0	-42.2	-1085.6	-61.2
β_4	ζ	-2702.4	-3139.3	-2588.7	-17836.3	-1176.6
β_5	n^2	2.1	3.0	1.6	24.6	1.3
β_6	c_e^2	-20.9	-18.3	-4.6	-9.9	-3.6
β_7	r_z^2	0.7	1.4	0.2	20.8	1.1
β_8	ζ^2	33.2	36.7	124.6	-278.3	1.4
β_9	$n\zeta$	45.5	54.9	33.1	331.4	20.1
β_{10}	$c_e\zeta$	27.1	24.3	51.1	329.8	19.0
β_{11}	$r_z\zeta$	56.3	57.4	22.7	371.2	21.8
	R^2	0.91	0.90	0.88	0.87	0.89

TABLE 4 Regression Coefficients for Number of Iterations (k_c) in Regression Model 2

Regression Coefficient	Corresponding Explanatory Variable	Algorithm				
		FW	PARTAN	RSD	DSD	GP
β_0	-	-707.1	-677.7	418.4	-31.7	-14.7
β_1	n	137.5	123.8	-3.0	4.4	1.5
β_2	c_e	73.6	74.2	-31.2	2.7	2.8
β_3	r_z	-1.4	-1.8	-0.2	0.1	-0.3
β_4	ζ	4.6	4.8	-409.3	7.8	3.5
β_5	n^2	-12.1	-10.8	-3.9	-0.3	-0.1
β_6	$c_e^{\ 2}$	-7.8	-7.9	-1.0	-0.2	-0.3
β_7	$r_z^{\ 2}$	0.0	0.0	0.1	0.0	0.0
β_8	ζ^2	23.0	22.0	48.3	-0.4	-0.1
β_9	$n\,\zeta$	5.8	6.8	27.1	0.1	0.2
β_{10}	$c_e\,\zeta$	20.0	20.3	26.5	0.4	0.7
β_{11}	$r_z\,\zeta$	-0.4	-0.3	-1.9	0.0	0.0
	R^2	0.89	0.89	0.96	0.88	0.92

out the possibility that the models might be more effective with the inclusion of other variables in addition to the 11 variables used or perhaps with the deletion of one or more of the variables in the model.

CONCLUSIONS

The contributions of this research are twofold. First, an extensive computational study was performed on a sequence of link- and path-based TAP algorithms using mid- to large-scale hypothetical networks, on the basis of which the relative performances of these typical algorithms were explored from the viewpoint of computational efficiency. Second, a statistical analysis was conducted to elaborate quantitatively the connection between the algorithm efficiency and the characteristics or properties of the TAP.

When the required accuracy level is not stringent, GP is consistently faster in approaching the neighborhood of the optimal solution than the other algorithms being compared. There are several reasons to account for GP's outstanding performance at a low accuracy level. GP avoids performing an extensive line search and reduces the number of stored paths significantly by the projection procedure. Moreover, its one-at-a-time link flow update strategy guarantees adequate convergence. Nevertheless, GP is inappropriate for the application in which a high-accuracy solution is demanded because of its linear convergence rate. DSD demonstrates good convergence performance by a modest number of iterations to obtain solutions with low to middle accuracy. DSD, however, lacks computational efficiency because of its onerous calculations to solve the disaggregate restricted master problem for each O-D pair. Note that the implementation of DSD in this research may not deliver its best performance; that is, improvement is possible by trying more combinations of algorithm parameters. In general, path-based algorithms excel in convergence speed (i.e., the number of iterations consumed to reach the prespecified accuracy level) compared with link-based algorithms. However, path-based algorithms are more sensitive to network size, zone–node ratio, and congestion level because of costly path operations.

For link-based algorithms, RSD is the one that exhibits favorable computational efficiency as compared with GP. The convergence speed as well as the overall competence of RSD is notable since it even outperforms GP when strict accuracy levels are required. Besides, the increases in zone–node ratio, network size, and congestion level do not seem to affect RSD's performance much in a negative way. F-W and PT perform similarly and suffer from slow convergence speed. PT improves F-W slightly only when the network size remains small and low accuracy level is demanded.

A newly developed origin-based algorithm (*16*) was not included in this research. To the authors' knowledge, the origin-based algorithm is also capable of getting a highly accurate solution with reasonable computing resources. It will be interesting to include the origin-based algorithm with the other algorithms investigated in this paper for computational comparisons and statistical analyses in the future.

Finally, the findings gained from this research are useful in guiding transportation professionals to choose suitable solution algorithms and to predict the resulting algorithm performance in TAPs.

REFERENCES

1. Wardrop, J. G. Some Theoretical Aspects of Road Traffic Research. *Proceedings of the Institute of Civil Engineers, Part II*, Vol. 1, 1952, pp. 325–378.
2. Chen, A., and D.-H. Lee. Path-Based Algorithms for Large-Scale Traffic Equilibrium Problems: Comparison Between DSD and GP. Presented at the 78th Annual Meeting of the Transportation Research Board, Washington, D.C., 1999.
3. Chen, A., D.-H. Lee, and Y. Nie. Path and Link Based Traffic Assignment Algorithms: A Comprehensive Computational Study. *Proc., 6th*

International Conference on Application of Advanced Technologies in Transportation Engineering (CD-ROM), Singapore, American Society of Civil Engineers, New York, 2000.
4. Jayakrishnan, R., W. K. Tsai, J. N. Prashker, and S. Rajadhyaksha. Faster Path-Based Algorithm for Traffic Assignment. In *Transportation Research Record 1443*, TRB, National Research Council, Washington, D.C., 1994, pp. 75–83.
5. Larsson, T., and M. Patriksson. Simplicial Decomposition with Disaggregated Representation for the Traffic Assignment Problem. *Transportation Science*, Vol. 26, 1992, pp. 4–17.
6. Sun, C., R. Jayakrishnan, and W. K. Tsai. Computational Study of a Path-Based Algorithm and Its Variants for Static Traffic Assignment. In *Transportation Research Record 1537*, TRB, National Research Council, Washington, D.C., 1996, pp. 106–115.
7. Tatineni, M., H. Edwards, and D. Boyce. Comparison of Disaggregate Simplicial Decomposition and Frank-Wolfe Algorithms for User-Optimal Route Choice. In *Transportation Research Record 1617*, TRB, National Research Council, Washington, D.C., 1998, pp. 157–162.
8. Frank, M., and P. Wolfe. An Algorithm for Quadratic Programming. *Naval Research Logistics Quarterly*, Vol. 3, 1956, pp. 95–110.
9. LeBlanc, L. J., E. K. Morlok, and W. P. Pierskalla. An Efficient Approach to Solving the Road Network Equilibrium Traffic Assignment Problem. *Transportation Research*, Vol. 9, 1975, pp. 309–318.
10. Florian, M., J. Guelat, and H. Spiess. An Efficient Implementation of the "PT" Variant of the Linear Approximation Method for the Network Equilibrium Problem. *Networks*, Vol. 17, 1987, pp. 319–339.
11. LeBlanc, L. J., R. V. Helgason, and D. E. Boyce. Improved Efficiency of the Frank-Wolfe Algorithm for Convex Network Problems. *Transportation Science*, Vol. 19, 1985, pp. 445–462.
12. Hearn, D. W., S. Lawphongranich, and J. A. Ventura. Finiteness in Restricted Simplicial Decomposition. *Operations Research Letters*, Vol. 4, 1985, pp. 125–130.
13. Hearn, D. W., S. Lawphongranich, and J. A. Ventura. Restricted Simplicial Decomposition: Computation and Extensions. *Mathematical Programming Study*, Vol. 31, 1987, pp. 99–118.
14. Hearn, D., and D. Kim. *RSDTA User Manual*. Center for Applied Optimization, Department of Industrial and Systems Engineering, University of Florida, 1995.
15. Barzaraa, M. S., and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. John Wiley & Sons, New York, 1979.
16. Bar-Gera, H. *Origin-Based Algorithms for Transportation Network Modeling*. Technical Report 103. National Institute of Statistical Sciences, Research Triangle Park, N.C., 1999.

Publication of this paper sponsored by Committee on Transportation Network Modeling, Subcommittee on Network Equilibrium Modeling.