Computational study of state-of-the-art path-based traffic assignment algorithms

Anthony Chen a,*, Der-Horng Lee b, R. Jayakrishnan c

a Department of Civil and Environmental Engineering, Utah State University, Logan, UT 84322-4110, USA
b Department of Civil Engineering, National University of Singapore, BLK E1A #07-16, 1 Engineering Drive 2, Singapore 117576, Singapore
c Department of Civil and Environmental Engineering, University of California, Irvine, CA 92697-3600, USA

Accepted 1 November 2001

Abstract

Recent research has demonstrated and established the viability of applying path-based algorithms to the traffic equilibrium problem in reasonably large networks. Much of the attention has been focused on two particular algorithms: the disaggregate simplicial decomposition (DSD) algorithm and the gradient projection (GP) algorithm. The purpose of this paper is to evaluate the performance of these two path-based algorithms using networks of realistic size. Sensitivity analysis is performed on randomly generated networks to examine the performance of the algorithms with respect to network sizes, congestion levels, number of origin-destination (OD) pairs, and accuracy levels. In order to be empirically convincing, a realistic large-scale network, known as the ADVANCE network, is also used to show that path-based algorithms are a viable alternative in practice. © 2002 IMACS. Published by Elsevier Science B.V. All rights reserved.

Keywords: Traffic assignment; User equilibrium; Gradient projection; Simplicial decomposition

1. Introduction

A significant rediscovery in the traffic assignment problem is the viability of path-based algorithms that are now made possible due to the dramatic advances in computing power in recent years. In the past, path-based algorithms, even those not enumerating all possible paths, were not considered a viable alternative for solving large-scale network problems because of intensive memory requirements and the difficulties in manipulating and storing paths. Recent research on path-based algorithms has demonstrated and established that it is a viable approach for traffic assignment problems with reasonably large network sizes [6–9]. Much of the attention has been focused on two particular algorithms: the disaggregate simplicial decomposition (DSD) algorithm and the gradient projection (GP) algorithm. DSD is considered

* Corresponding author. Tel.: +1-435-797-7109; fax: +1-435-797-1185.
E-mail address: achen@cc.usu.edu (A. Chen).

0378-4754/02 – see front matter © 2002 IMACS. Published by Elsevier Science B.V. All rights reserved.
PII: S0378-4754(01)00437-2
as one of the state-of-the-art algorithms for traffic assignment. Application of GP to solve the traffic assignment problem is relatively new [6], but as shown in this paper GP is as good as or better than DSD in direct comparisons.

Both DSD and GP have been extensively compared with the most commonly used Frank–Wolfe algorithm and have shown excellent results [8,9]. However, there has not been any comparison between DSD and GP on realistic networks. The purpose of this paper is to compare and evaluate the performance of these two path-based algorithms using networks of realistic size. Randomly generated networks are used to evaluate the relative performance of the algorithms with respect to network sizes, congestion levels, number of origin-destination (OD) pairs, and solution accuracy. In addition, a realistic large-scale network, known as the ADVANCE network, is used to show that path-based algorithms are a viable alternative in practice.

Following the introduction, we briefly review the path-formulated traffic assignment problem in the next section. An overview of the algorithmic concepts of DSD and GP will be provided. Numerical results are presented and analyzed, and conclusions are addressed last.

2. The path-formulated traffic assignment problem

Consider an urban traffic network represented as a graph \( G = (N, A) \), where \( N \) and \( A \) are the sets of nodes and links, respectively. Let \( R \) and \( S \) be subsets of \( N \) and \( q_{rs} \) be the steady-state demand generated from origin \( r \in R \) to destination \( s \in S \). The independent variables are a set of path flows \( f_{rs}^k \) that satisfy the travel demand:

\[
\sum_{k \in K_{rs}} f_{rs}^k = q_{rs}, \quad \forall r \in R, s \in S
\]

(1)

where \( K_{rs} \) is a set of cycle-free paths connecting \( r \) and \( s \). Let \( x_a \) be the traffic flow on link \( a \in A \), then the total traffic flow on each link can be written as:

\[
x_a = \sum_{r \in R} \sum_{s \in S} \sum_{k \in K_{rs}} f_{rs}^k \delta_{rs}^a, \quad \forall a \in A
\]

(2)

where

\[
\delta_{rs}^a = \begin{cases} 
1 & \text{if link } a \text{ is on path } k \text{ connecting } r \text{ and } s \\
0 & \text{otherwise}
\end{cases}
\]

To ensure meaningful solutions, all path flows must be non-negative:

\[
f_{rs}^k \geq 0, \quad \forall k \in K_{rs}, r \in R, s \in S
\]

(3)

To model congestion effects, a link cost function \( t_a(x_a) \) is often used to express the travel time along the link as a function of the total link flow. In this paper, we use the standard Bureau of Public Road (BPR) link cost function for our numerical study. The functional form is given by

\[
t_a(x_a) = t_l \left( 1 + 0.15 \left( \frac{x_a}{C_a} \right)^b \right)
\]

(4)
where $t_f$ and $C_a$ are the free-flow travel time and link capacity, respectively. Depending on the purpose of the assignment, the objective can be minimizing either the travel time of the individual users or the system as a whole. The objective functions for these two assignments can be mathematically stated as:

$$Z_{UE} = \sum_{a \in A} \int_{0}^{x_a} t_a(w) \, dw$$  \hspace{1cm} (5)

$$Z_{SO} = \sum_{a \in A} t_a(x_a) x_a$$  \hspace{1cm} (6)

where $Z_{UE}$ and $Z_{SO}$ are the objective functions of the user equilibrium (UE) and system optimal (SO) formulations corresponding to the two assignments stated above. The UE assignment finds the traffic flow pattern by allocating the OD demands to the network such that all used paths between a given OD pair have equal travel time, and no unused path has a lower travel time. Such an equilibrium state is what results if each and every driver simultaneously attempts to minimize individual travel times. As for the SO assignment, the objective is to minimize the total system travel time. Both assignments are very important in the efficient planning and applications of optimal routing, signal control, and traffic prediction in urban traffic networks. The path-formulated traffic assignment problem is to minimize (5) or (6) subject to constraints (1)-(3).

3. Path-based traffic assignment algorithms

Two path-based algorithms are considered in this paper for solving the path-formulated traffic assignment problem. Specifically, we compare the DSD algorithm with the GP algorithm. Only a brief description of the algorithmic concept is provided here. The readers are referred to [4] for the details of the two algorithms and the implementation steps.

3.1. Disaggregate simplicial decomposition (DSD) algorithm

The DSD algorithm belongs to the general class of simplicial decomposition algorithm, which is based on Caratheodory’s theorem [1]. This theorem states that any point in a bounded polyhedral set can be described by a convex combination of the extreme points. The general solution approach of this class of algorithms iterates between a linear subproblem and a master problem to determine the next solution point. The subproblem solves a linear approximation of the objective function to obtain extreme points. For the traffic assignment problem, this step corresponds to finding the shortest path for each OD pair and performing an all-or-nothing assignment to that path. The master problem solves a reduced dimension of the problem over a convex hull of extreme points generated thus far.

The constraint set of the path-formulated traffic assignment problem is highly structured that allows it to be expressed as a Cartesian product (i.e. $F = \prod_{r,s \in RS} F_{rs}$, where $F_{rs}$ is the simplex associated with the path-flow variables of OD pair $(r,s)$ and $|RS|$ is the number of OD pairs). Its simple structure makes it possible to decompose the constraint set to individual OD pairs and represents each simplex $F_{rs}$ using one constraint per OD pair. Exploiting this simple structure, the master problem can be reformulated by disaggregating into individual OD pairs since each simplex is represented separately. Considering each
simplex as a non-empty set containing a subset of paths denoted by $\hat{K}_{rs}$ and letting $\lambda_{rs}^{\mu}$ be the portion of demand $q_{rs}$ assigned to path $k$, the disaggregate master problem becomes:

$$\min Z(\lambda) = \sum_{a \in A} \int \sum_{r \in R} \sum_{s \in S} q_{rs} \sum_{k \in \hat{K}_{rs}} \lambda_{rs}^{\mu} \delta_{rs}^a \, dt$$

subject to

$$\sum_{k \in \hat{K}_{rs}} \lambda_{rs}^{\mu} = 1, \quad \forall r \in R, s \in S$$

$$\lambda_{rs}^{\mu} \geq 0, \quad \forall k \in \hat{K}_{rs}, r \in R, s \in S$$

Path flows are readily available once $\lambda_{rs}^{\mu}$ are solved. Thus, the disaggregated formulation is equivalent to the path formulation. Note that the disaggregate master problem is a convex program with simple linear constraints. Many efficient algorithms can be used to solve it. In Larsson and Patriksson [7], a hybrid approach using reduced gradients and Newton diagonalization is employed to solve the disaggregate master problem.

3.2. Gradient projection (GP) algorithm

The GP algorithm described here is based on the Goldstein–Levitin–Polyak GP method formulated by Bertsekas [2] for general nonlinear multi-commodity problems and applied with appropriate data structures to the traffic assignment problem by Jayakrishnan et al. [6]. The algorithm operates directly in the space of path flows. In every iteration, the moves are in the direction of the negative gradient, scaled by the second derivative Hessian. A projection is made back to the non-negativity constraints whenever the move results in an infeasible solution.

The main idea is to eliminate the travel demand constraints (1) by reformulating the path-flow variables in terms of the non-shortest path flows to make the projection operation simpler. This is accomplished by partitioning the path set $K_{rs}$ of each OD pair $(r, s)$ into the shortest path $f_{rs}^{\mu}$ and the non-shortest path $f_{rs}^{\nu}$ and expressing constraints (1) in terms of $f_{rs}^{\nu}$ as follows:

$$f_{rs}^{\nu} = q_{rs} - \sum_{k \in K_{rs}, r \neq \tilde{k}_{rs}} f_{rs}^{\mu}, \quad \forall r \in R, s \in S$$

Embedding constraints (1) into the objective function, we obtain a new formulation with just the non-negativity constraints on the non-shortest path flows as the decision variables.

$$\min \tilde{Z}(f')$$

subject to

$$f_{rs}^{\nu} \geq 0, \quad \forall k \in K_{rs}, k \neq \tilde{k}_{rs}, r \in R, s \in S$$

where $\tilde{Z}$ is the transformed objective function expressed in terms of the non-shortest path flows $f'$ for all OD pairs. This reformulation is a convex program with only non-negativity constraints. It can be easily
shown that in each iteration, the scaled GP algorithm updates the path flows according to the following iteration equations:

\[ f_{rs,k}(n+1) = \left[ f_{rs,k}(n) - \alpha(n) \right] s_{rs,k}(n)(d_{rs,k}(n) - d_{rs,k}(n)) + , \forall k \in K_{rs}, k \neq \overline{k}, r \in R, s \in S \quad (12) \]

\[ f_{rs,\overline{k}}(n+1) = q_{rs} - \sum_{k \in K_{rs}, k \neq \overline{k}} f_{rs,k}(n+1), \forall r \in R, s \in S \quad (13) \]

where \( n \) is the iteration number, \( \alpha(n) \) the stepsize, \( s_{rs,k}(n) \) a diagonal, positive definite scaling based on the second derivative Hessian, \( d_{rs,k}(n) \) and \( d_{rs,\overline{k}}(n) \) are the first derivatives of the objective function \( Z \) along path \( k \) and the shortest path \( \overline{k} \) between \( r \) and \( s \) (i.e. the travel times on path \( k \) and \( \overline{k} \)), and \( [ \cdot ]^+ \) denotes the orthogonal projection to the positive orthant (i.e. if the argument is negative, the projection makes it zero).

4. Numerical experiments

Several sensitivity tests are performed to examine the performance of the two algorithms. Specifically, we evaluate the performance of the algorithms with respect to the following factors: (a) network sizes, (b) congestion levels, (c) number of OD pairs, and (d) solution accuracy levels. Five randomly generated networks of various sizes are used in the sensitivity analysis. In order to be empirically convincing that path-based algorithms are a viable alternative in practice, a real and large-scale network, known as the ADVANCE network in the Chicago area, is also used for generating computational results from DSD and GP algorithms. Table 1 provides the statistics of these networks.

Since the two algorithms are normally implemented with different appropriate criteria for termination, we use the objective value as a common criterion to measure the performance of the algorithms. That is, we run DSD first and use the objective function value at termination as the reference to locate the CPU time of GP that gives approximately the same objective function value. In our computational experiments, DSD terminates when the relative error between the upper and lower bounds of the objective value satisfies the pre-specified threshold of 0.005 or the algorithm reaches 20 iterations (depends on which meets first).

4.1. Sensitivity analysis

It is well known that computational effort of the algorithms increases with respect to the number of network nodes. Here, we examine the effect of (a) using different network sizes, ranging from 900

<table>
<thead>
<tr>
<th>Network</th>
<th>No. of nodes</th>
<th>No. of links</th>
<th>No. of zones</th>
<th>No. of OD pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>30 x 30</td>
<td>900</td>
<td>3480</td>
<td>112</td>
<td>12432</td>
</tr>
<tr>
<td>40 x 40</td>
<td>1600</td>
<td>6240</td>
<td>200</td>
<td>39800</td>
</tr>
<tr>
<td>50 x 50</td>
<td>2500</td>
<td>9800</td>
<td>312</td>
<td>97032</td>
</tr>
<tr>
<td>60 x 60</td>
<td>3600</td>
<td>14160</td>
<td>450</td>
<td>202050</td>
</tr>
<tr>
<td>70 x 70</td>
<td>4900</td>
<td>19320</td>
<td>612</td>
<td>373932</td>
</tr>
<tr>
<td>ADVANCE</td>
<td>2552</td>
<td>7850</td>
<td>447</td>
<td>137417</td>
</tr>
</tbody>
</table>
Table 2
Sensitivity analysis with respect to network sizes

<table>
<thead>
<tr>
<th>Network</th>
<th>CPU time (s)</th>
<th>No. of iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GP</td>
<td>DSD1</td>
</tr>
<tr>
<td>30 × 30</td>
<td>7.47</td>
<td>60.28</td>
</tr>
<tr>
<td>40 × 40</td>
<td>30.04</td>
<td>316.11</td>
</tr>
<tr>
<td>50 × 50</td>
<td>70.32</td>
<td>2170.26</td>
</tr>
<tr>
<td>60 × 60</td>
<td>206.48</td>
<td>4919.85</td>
</tr>
<tr>
<td>70 × 70</td>
<td>512.06</td>
<td>21446.20</td>
</tr>
</tbody>
</table>

As expected, computational time for both versions of DSD and GP increases as the size of the network grows. In all cases, GP performs better than DSD2 and significantly better than DSD1. Averaging over the five networks, GP is about seven times quicker than DSD2 and 25 times faster than DSD1, while DSD2 is at least two times better than DSD1. The number of iterations required for GP to converge to the same objective value as DSD2 is, in general, less than DSD1, but more than DSD2. The reason is that GP is more efficient than DSD1 in finding accurate solution, because it incorporates second derivative information for scaling. However, since GP does not re-equilibrate the master problem, it generally takes more iterations than DSD2 to reach an equivalent objective value. This result agrees with a recent study done by Chen and Jayakrishnan [3] showing the effects of equilibration with a restricted path set under two flow update strategies. That study found that by repeatedly equilibrating the master problem, it could reduce the number of iterations, however, at the expense of adding more computational time in the equilibration procedure. The overall performance would be better if the master problem is only solved with sufficient accuracy. Similar results have also been reported by Damberg [5] who compares the two versions of DSD using different termination strategies.

Using the 50 × 50 network, we test the sensitivity of factors (b)–(d) listed above and the results are provided in Table 3. As the congestion level grows, it is intuitive to expect the computational time will also increase. To examine this effect, the demand level is incrementally and uniformly increased from light (0.1) to heavily congested (0.9) traffic conditions. The medium congestion level of 0.5 is the same one reported in Table 2. As shown in Table 3, computational time generally increases when congestion

---

1. Note that this is known not to be unusual for DSD. Appropriate parameters are often difficult to find to cause the switch. Our implementation caused switching as expected for the small networks presented in [7], but not for larger networks presented in this paper.
2. Though both DSD1 and DSD2 terminate at the specified relative error of 0.005, DSD2 always terminates at a lower objective value than DSD1.


### Table 3
Sensitivity analyses using 50 × 50 network

<table>
<thead>
<tr>
<th>Factor</th>
<th>CPU time (s)</th>
<th>No. of iterations</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GP DSD1 GP DSD2</td>
<td>GP DSD1 GP DSD2</td>
<td></td>
</tr>
<tr>
<td>Congestion level(a)</td>
<td>0.1 8.87 137.01 23.07 95.64</td>
<td>3 6 4 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.3 35.44 441.37 81.40 361.27</td>
<td>5 9 10 6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.5 70.32 2710.26 107.08 827.69</td>
<td>7 12 10 8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.7 87.22 4525.94 128.13 1324.45</td>
<td>8 20 11 10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.9 106.02 5166.59 150.86 2044.92</td>
<td>9 18 12 12</td>
<td></td>
</tr>
<tr>
<td>OD/nodes (b)</td>
<td>1/16 25.11 615.74 36.43 222.57</td>
<td>8 16 11 8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1/8 70.32 2710.26 107.08 827.69</td>
<td>7 12 10 8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3/16 146.24 4066.40 222.31 1894.81</td>
<td>7 10 10 8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1/4 246.46 9143.70 419.03 3567.81</td>
<td>7 12 11 9</td>
<td></td>
</tr>
<tr>
<td>Relative error</td>
<td>0.01 58.04 1462.52 94.80 690.96</td>
<td>6 9 9 7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.005 70.32 2710.26 107.08 827.69</td>
<td>7 12 10 8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.001 131.81 7600.36 232.91 4280.01</td>
<td>12 7(c) 20 19</td>
<td></td>
</tr>
</tbody>
</table>

\(a\) The OD demand matrix is uniformly increased from 0.1 to 0.9 (the base case is 0.5).

\(b\) Ratio of origin-destination zones to total number of nodes.

\(c\) The algorithm reaches the pre-specified maximum number of iterations.

level is high. However, the computational time of both DSD1 and DSD2 increases at a much faster rate compared to GP. In all cases, GP shows an order of magnitude better performance.

Next, we examine the effect of the density of origin/destination (O/D) nodes by varying the fractions of nodes, which are the O/D nodes, from 1/16 to 1/4. For the 50 × 50 network, these percentages correspond to 156–625 O/D nodes or 24,180–390,000 OD pairs, respectively. It can be seen from Table 3 that all algorithms require more computational effort for higher number of OD pairs. This is to be expected since more paths need to be processed. In terms of computational time, GP, on the average of four O/D density levels, requires about 1/30 of DSD1’s time and 1/8 of DSD2’s time to meet the accuracy level of 0.005. However, the increase in OD pairs does not seem to affect the convergence in terms of the number of iterations.

The last factor that we examine is accuracy level. We test DSD1, DSD2, and GP with a relative error of 0.01, 0.005, and 0.001, as shown in Table 3. All three algorithms naturally require more iterations and higher computational times as the level of accuracy increases. However, in all cases, GP once again outperforms both versions of DSD. Specifically, the speed-up ratios are 1.0–43.2 for GP and DSD1, 1.0–13.3 for GP and DSD2, and 1.0–1.9 for DSD2 and DSD1. It should also be noted that DSD1 fails to meet the accuracy level of 0.001 even 75 iterations.

#### 4.2. ADVANCE network

To demonstrate the applicability of path-based algorithms, we apply the algorithms to a realistic and large-scale network with an accuracy level of 0.001. The ADVANCE network is located in the
northwestern suburbs of Chicago and covers about 800 km² (330 miles²). The topology of this large-scale network is basically a regular grid with a few diagonal major arterials directed toward the Chicago central business district. The network consists of 2552 nodes, 7850 links, 447 zones, and 137,417 OD pairs. The OD demand data are based on Chicago area transportation studies (CATS) estimates for 1990 [10].

The convergence characteristics of the algorithms are shown in Fig. 1. Again, two DSD versions were tried: one with just the first-order reduced gradient method and another where the first-order method was inactive and only the second-order method was used. The second-order method achieved great improvement in terms of number of iterations and computational times over DSD1, but was still inferior to GP. DSD2 terminated at the 12th iteration with a relative error of 0.0008, while DSD1 was not able to reach the pre-set accuracy of 0.001 at the maximum number of iterations. Its relative error at the 20th iteration was 0.0019. Computational time was reduced more than half. This result is consistent with the results reported in [5], which shows an average of 70% improvement with using second-order method to solve the restricted master problem. As for GP, it takes 18 iterations to yield approximately the same objective value as DSD2.

At termination, DSD1 generated 499,717 paths and used 499,446 paths (about 99.9%), DSD2 generated 498,379 paths and used 493,972 (about 99.1%), while GP generated 520,772 paths and used 145,907 paths (about 28.0%). The averages per OD pair of generated and used paths are 3.64 and 3.63 for DSD1, 3.63 and 3.59 for DSD2, and 3.79 and 1.06 for GP. About 95% of the OD pairs in GP contain only one path compared to 22.1% in DSD1 and 22.4% in DSD2. It should be noted that these single-path OD pairs do not need any flow update computations, which is one reason why GP is faster.

3 This number includes paths that were dropped and re-entered during the iterations.
5. Conclusions

We have implemented and compared two path-based algorithms for the traffic assignment problem. Based on our numerical experiments and results found by other researchers, the following conclusions are drawn:

GP outperforms both versions of DSD in all cases tested in the sensitivity analysis as well as the ADVANCE network. GP is the most efficient, followed by DSD2, and DSD1. We identify two reasons why GP is faster in converging to a neighborhood of an optimal solution than both versions of DSD: (1) GP maintains fewer used paths in each iteration and (2) GP avoids performing expensive line searches. As discussed in Chen et al. [4], GP identifies zero-flow paths much quicker than DSD, which helps to maintain a much smaller path set by employing column dropping to remove the unused paths. A one-at-a-time flow update strategy helps GP to avoid expensive line searches by keeping the second derivative information accurate.

As for the DSD algorithm, the strategy of using the first-order reduced gradient method to quickly obtain a near-optimal solution and switching to the second-order diagonalized Newton to achieve a highly accurate solution is intuitively very appealing. However, it is very difficult to determine the “optimal” switching point in practice and the performance of the algorithm depends strongly on a good switching rule. Recent implementations of DSD have abandoned the first-order method and exclusively opted for the second-order method [5,9]. This eliminates the difficulty of identifying different switching rules for different network topologies, as they often have influence on the performance of the algorithm. By using exclusively the second-order method along with a bounding terminating strategy, significant improvement in CPU time was achieved corroborating earlier research [5]. Similar results have been found in this study. We believe DSD can further improve if the parameters are chosen properly, as we have not attempted to find good values to fit each individual network. In addition, if a column dropping scheme can be incorporated into DSD, the performance could be further enhanced.

Overall, this paper has shown that path-based algorithms are a viable approach to solving large-scale networks with reasonable computational times. The main drawback in the past of using a path-based algorithm was the memory requirement. This restriction has been relaxed considerably in recent years due to rapid advances in computing environment. It is likely to become less of a concern as computing power progresses in the future. On the other hand, path solutions are increasingly important in many applications that cannot be adequately modeled without explicitly using path variables. For future research, we plan to explore ways to apply the path-based algorithms discussed in this paper to different applications.

Acknowledgements

The authors are pleased to acknowledge the support from the Institute of Transportation Studies at the University of California at Irvine in the preparation of this paper. We also would like to thank Professor David Boyle at the University of Illinois at Chicago for providing us with the ADVANCE network data sets for this study.

References


