Does LUCE outperform OBA? A Comparison Study of Two Bush-based Algorithms for the Traffic Assignment Problem

Jun Xie
School of Traffic and Transportation Engineering
Tongji University
4800 Cao’an Road, Shanghai, 201804
Phone: 86-13621859041
Email: jonasxie@gmail.com

Yu (Marco) Nie *
Department of Civil and Environmental Engineering
Northwestern University
2145 Sheridan Road
Evanston, Illinois, 60208
Phone: 1-847-467-0502
Email: y-nie@northwestern.edu

Xiaoguang Yang
School of Traffic and Transportation Engineering
Tongji University
4800 Cao’an Road, Shanghai, 201804
Phone: 86-21-69588994
Email: yangxg@tongji.edu.cn

November 13, 2012

Word counts based on Translator’s Abacus on the PDF File
5650 words + 1 tables + 6 figures = 7400

*Corresponding author
Abstract

This paper compares two bush-based traffic assignment algorithms, the origin-based algorithm (OBA) and the local user cost equilibrium algorithm (LUCE). The two algorithms are closely related with one major difference: they solve the decomposed elementary node-based subproblem using different methods. Specifically, LUCE employs a greedy algorithm that is able to solve the subproblem exactly, whereas OBA uses a one-step quasi-Newton method known as gradient projection to solve the subproblem approximately. Therefore, LUCE seems to hold promises to improve OBA because its subproblem solver is presumably faster and more precise. We implemented these two algorithms in the same programming platform, where the codes of them are shared as many as possible. Numerical experiments reported in this paper, however, indicate that LUCE not only provide no obvious computational advantages over OBA, it often fails to converge beyond certain point. The focus of this paper is to find an answer to this counter-intuitive phenomenon. Our analysis suggests that the greedy method used by LUCE require highly accurate estimation of second-order derivatives. When second-order derivatives are subject to large errors, the greedy method can provide consistently sub-optimal descent direction, which seems to be unable to fix.

Keywords: Bush-based algorithm, traffic assignment problem, greedy method, gradient projection method, local user cost equilibrium
1 Introduction

The traffic assignment problem is used to predict the route choices for travelers on urban road networks under the user equilibrium (UE) principle [1], which was first formulated as a convex optimization problem with linear constraints by Beckmann [2]. Since then, finding efficient algorithms to solve this problem has attracted much attention, especially following the initial success of the Frank-Wolfe algorithm [3] and the later recognition of its painfully slow convergence behavior [4, 5, 6, 7, 8, 9, 10]. The past decade has witnessed the development of a class of bush-based algorithms [11, 12, 13, 14, 15] that is able to converge to highly precise solutions with reasonable computational overhead. A recent research [16] compares the convergence performance of all these algorithms by implementing them in the same platform. The efficiency of these algorithms makes them a new focus in traffic assignment research.

All bush-based algorithms (BA) known so far share a few important features. They construct and maintain a bush for each origin (or destination) and restrict the assignment operation only to these bushes. The concept of bush was originally coined by Dial in his celebrated work on logit traffic assignment [17]. Simply speaking, a bush is an acyclic subnetwork that intends to encompass all UE shortest paths. Acyclicity promise great efficiency for many network operations. Moreover, when all bushes are equilibrated, the entire network is equilibrated. That is, the use of bush not only offers efficiency, but also assures optimality, in the sense that UE flows can be represented by equilibrated bushes [13]. Bush-based algorithms construct optimal bushes at UE by iterating between two sub-problems: bush construction and bush equilibration.

Bush-based algorithms differ from each other mainly due to different bush equilibration methods. According to the flow aggregation level in bush equilibration, bush-based algorithms can be classified as route flow based or origin flow based. Dial’s Algorithm B [12] is in the former category; it equilibrates bushes by swapping flows between the longest and shortest routes. In contrast, the origin-based algorithm, or OBA [11], and the local user cost equilibrium algorithm, or LUCE [14], operate in the space of origin flows, which are represented by proportions of traffic arriving at each node from its predecessor links. OBA is now widely used as a benchmark to gauge the performance of newer traffic assignment algorithms such as LUCE. In many aspects, these two algorithms are closely related. First of all, they solve the same restricted master problem (RMP) in the same space of origin flow. Second, they employ the same method to approximate the first- and second-order derivatives of origin flows for the RMP. Thirdly, in bush equilibration stage, both OBA and LUCE transform the RMP into a sequence of small quadratic programs associated with each node in the bush, referred to as node-based subproblems [14]. The solution to these quadratic programs, constructed using the first-order and second-order derivatives mentioned above, provide a descent direction. Then, a line search has to be used to find a proper step size for both algorithms.

The only major difference, from the algorithmic perspective, between OBA and LUCE lies in the method to solve the node-based subproblem. LUCE applies a greedy method, inspired by the hyper-path algorithm for transit assignment. In contrast, OBA employs a gradient projection (GP) method, which is similar to those used in [8, 12]. The research question we ask here is how much this difference could contribute to the relative convergence performance of the two algorithms. Intuitively, the algorithm that can solve this subproblem more quickly and precisely is likely to be the winner. If this intuition holds true, then one should definitely prefer the greedy method to the GP method, because the former can get exact solutions by directly solving the KKT system, whereas GP is designed to achieve just an approximation (see Section 4 for more details). Indeed, limited computational evidence in the literature seems to support such a theory.

---

1LUCE was originally presented as destination-based. Yet this difference is trivial from the algorithmic point of view.
As observed by [14], LUCE “compares favourably with the most advanced methods recently proposed in the literature.” Yet, all experiments of which we are aware compare executive codes, which, in our opinion, makes it almost impossible to determine if the observed performance differences come from implementation or the algorithm itself.

In this study, we first implement both LUCE and OBA on the same programming platform that allows us to use the same code wherever possible. Effectively, our LUCE and OBA codes share almost all components except for those related to solving the node-based subproblem. To our surprise, we found that the computational performance of the two algorithms are strikingly similar for the convergence range that is of practical interest (up to $10^{-6} - 10^{-8}$). Perhaps more important for our interest, LUCE seems to be unable to achieve very high levels of precision - it consistently gets stuck around $10^{-8} - 10^{-10}$ in almost all cases. This mysterious barrier for LUCE appears counter-intuitive at first glance, considering that its greedy method is able to solve the quadratic subproblems exactly. Then, several variants of LUCE that aim to amending LUCE are presented, and two of them show better performance than the original LUCE, but the standstill still persists. Consequently, the main focus of this paper is to offer an explanation to this mystery.

The rest of this paper is organized as follows. Section 2 provides the formulation of the single-origin traffic assignment problem and the overall algorithmic framework shared by both LUCE and OBA. Results of numerical experiments are presented in Section 3, followed by discussions. In Section 4, we provide a theoretical investigation that aims to explain the observed behaviour of OBA and LUCE in Section 3. The last section concludes the paper with a summary of main findings.

2 Formulation and solution framework

It is well known that the traffic assignment problem (TAP) can be formulated as a convex optimization problem with linear constraints [9]. Often, this problem is not solved directly. Instead, existing algorithms usually decompose TAP into a series of sub-problems and solve them iteratively. In bush-based algorithms, TAP is decomposed into a sequence of single-origin sub-problems. In each sub-problem, only demands departing from the origin can be assigned and equilibrated on the bush, with flows on other bushes viewed as constant background traffic. The single-origin sub-problem defined on a bush is often called a Restricted Master Problem (RMP) because the assignment is restricted to the bush. Using routing variables $\phi_{ij}$ to represent the flow proportion on bushes rooted at origin $r$, LUCE and OBA solve exactly the same RMP as follows:

$$\min z(\phi) = \sum_{ij} \int_{0}^{x_{ij}} t_{ij}(w)dw$$  \hspace{1cm} (1)

subject to:

$$x_{ij}^{r} = \eta_{j}^{r} \phi_{ij}^{r}, \quad \forall j,$$ \hspace{1cm} (2)

$$\eta_{j}^{r} = q_{j}^{r} + \sum_{l \in O(j,r)} \eta_{l}^{r} \phi_{jl}^{r}, \quad \forall j,$$ \hspace{1cm} (3)

$$\sum_{i \in I(j,r)} \phi_{ij}^{r} = 1, \quad \forall j; \phi_{ij}^{r} \geq 0, \quad \forall ij$$ \hspace{1cm} (4)

Note that the link cost function $t_{ij}(x_{ij})$ is function of the total link flow on link $ij$, $x_{ij} = x_{ij}^{r} + x_{ij}^{0}$, and $x_{ij}^{0}$ is the background traffic the other bushes. Other notations used in this paper are listed as follow:
\( G \) network
\( N, A \) network’s node set and link set
\( R, S \) the set of origins and destinations
\( r, s \) origin and destination, \( r \in R \) and \( s \in S \)
\( I(j, r) \) the set of backward star of node \( j \) on bush rooted at \( r \)
\( O(j, r) \) the set of forward star of node \( j \) on bush rooted at \( r \)
\( x_{r ij}^r \) the current origin-based link flow on link \( ij \)
\( t_{ij} \) time of link \( ij \)
\( q_j^r \) demand from origin \( r \) to destination \( j \).
\( \eta_j^r \) the total auxiliary flow arriving at node \( j \) on the bush rooted at \( r \)
\( \phi_{ij}^r \) the auxiliary routing variables denoting the proportion of \( \eta_j^r \) that use link \( ij \) on bush rooted at \( r \)
\( g_{ij} \) cost derivative of link \( ij \)
\( \eta_{ij}^r \) the auxiliary origin-based flow on link \( ij \)
\( x_j^r \) the current total flow arriving at node \( j \) on the bush rooted at \( r \), defined as \( x_j^r = \sum_{i \in I(j, r)} x_{ij}^r \)
\( \varphi_{ij}^r \) the current flow proportion of link \( ij \), defined as \( \varphi_{ij}^r = x_{ij}^r / x_j^r \)
\( u_j^r \) the shortest route cost of node \( j \) on bush rooted at \( r \)
\( U_j^r \) the longest route cost of node \( j \) on bush rooted at \( r \)
\( v_{ij}^r \) the first order derivatives of RMP with respect to \( \phi_{ij}^r \)
\( s_{ij}^r \) the second order derivatives of RMP with respect to \( \phi_{ij}^r \)

To solve the traffic assignment problem, both OBA and LUCE employs the same two-loop framework. For simplicity we shall focus on the single-origin case. The outer loop is used to initialize, expand and trim bushes. When no bushes can be changed, the solution to the RMP is considered as the solution to the original problem. Hence, the outer loop also controls the algorithm’s overall convergence. The inner loop attempts to equilibrate flows on bushes. Let \( B \) denotes a bush; the outer loop can be summarized as follows:

**Outer loop:**

- **Step 0** Initialize \( B \) as a shortest path tree rooted at the origin. Assign all flows to links on the tree.
- **Step 1** Expand \( B \) by adding new links that have the potential to reduce travel times in the bush.
- **Step 2** **Inner loop:** Equilibrating flows on the current bush
  - **Step 3** Reduce \( B \) by removing unused links.
  - **Step 4** If the convergence requirement is satisfied, stop; otherwise, goto **Step 1**.

**Step 0, Step 1 and Step 3** are bush management procedures aimed at optimizing the bush and maintaining its acyclicity. In our implementation, the bush management strategies proposed by Nie [13] are employed for both OBA and LUCE. The inner loop mainly focuses on equilibrating flow on bushes, which can be summarized as follows:

**Inner loop :**

- **Step 2.0** : Forward pass: update link travel time \( t_{ij} \) and its derivative \( g_{ij} \).
- **Step 2.1** : Backward pass: compute a descent direction by decomposing the RMP into a series of node-based subproblem, and solving their quadratic approximation.
Table 1: Detail of test networks

<table>
<thead>
<tr>
<th>Scale</th>
<th>Network</th>
<th>Node</th>
<th>Link</th>
<th>Zone</th>
<th>Trip</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>Anaheim</td>
<td>416</td>
<td>914</td>
<td>38</td>
<td>104,694</td>
</tr>
<tr>
<td></td>
<td>Barcelona</td>
<td>1020</td>
<td>2522</td>
<td>97</td>
<td>184,680</td>
</tr>
<tr>
<td></td>
<td>Winnipeg</td>
<td>1052</td>
<td>2836</td>
<td>135</td>
<td>64,784</td>
</tr>
<tr>
<td></td>
<td>Chicago Sketch</td>
<td>933</td>
<td>2950</td>
<td>386</td>
<td>1,260,910</td>
</tr>
<tr>
<td>Large</td>
<td>PRISM</td>
<td>14,639</td>
<td>33,937</td>
<td>898</td>
<td>609,670</td>
</tr>
<tr>
<td></td>
<td>Chicago Regional</td>
<td>12,982</td>
<td>39,018</td>
<td>1771</td>
<td>1,360,430</td>
</tr>
</tbody>
</table>

- **Step 2.2**: Perform a line search to find a proper step size associated with the descent direction obtained in Step 2.1.
- **Step 2.3**: Update bush flows based on the step size and descent direction obtained from previous steps. If the convergence requirement is satisfied for the current bush or the maximum number of iteration is reached, stop; otherwise, go to Step 2.0.

Details of OBA and LUCE are omitted here for brevity. The readers are referred to [11, 13, 14]. However, in Section 4, we will explain Step 2.1 in the inner loop, which has to do with solving the node-based subproblem using the greedy or GP methods. Step 2.1 is not only the key step to solve the RMP, but also represents the most critical difference between LUCE and OBA. We now turn to numerical experiments and discuss their results.

## 3 Numerical experiments

In this section, we will compare the convergence performance of OBA and LUCE using numerical experiments. To provide a performance benchmark, the Frank-Wolfe (FW) algorithm is also included in the experiments. Note that the FW algorithm is well known for its inability to achieve a precise equilibrium solution. All three algorithms are coded using TNM, a C++ class library specialized in modelling transportation networks [18]. All numerical results reported in this section were produced on a Windows XP-64 Workstation with two Xeno 3.0 GHz CPUs and 8 GB RAM.

The algorithms are applied to perform traffic assignment on six networks, of which two are regional scale. The main convergence indicator used in this study is the so-called relative gap, which measures how close the current solution is to the true user equilibrium. In our notation, the relative gap is calculated by

\[ G_{gap} = 1 - \sum_{rs} u_{rs} q_{rs} / \sum_{ij \in A} x_{ij} t_{ij} \]  

where \( u_{rs} \) is the minimum travel time between the O-D pair \( rs \) based on the current link travel time \( t_{ij} \), and \( q_{rs} \) is the travel demand between the O-D pair \( rs \). Finally, the BPR-function is used to calculate link travel times.

For the four small networks, the convergence criterion is set to \( 10^{-14} \) in order to examine what levels of precision the two bush-based algorithms are capable of achieving. The convergence curves of all three algorithms are reported in Figure 1. In these plots, \( x \) axis represents CPU times, and \( y \) axis represents the relative gap. As expected, the convergence of the FW algorithm almost becomes a standstill after it reaches
a relative gap of about $10^{-5}$. Figure 1 also depicts a much sharper convergence behaviour for both OBA and LUCE up to certain relative gap. In fact, the convergence curves of OBA and LUCE almost overlap completely, before they diverge after reaching certain critical relative gap. It is clear that LUCE stops converging after it attains a critical relative gap, which seems problem specific. For the Anaheim network, that gap is about $10^{-8}$, for Barcelona and Chicago sketch network, it’s $10^{-10}$, whereas for Winnipeg network, the gap can reach $10^{-12}$.

We now turn to the larger regional scale networks, PRISM and Chicago regional networks. We set the convergence criterion to $10^{-10}$ to avoid excessive computation time. The results presented in Figure 2-a basically tells the same story as in Figure 1. That is, OBA and LUCE both reached a relative gap of $10^{-8}$ comfortably along a very similar convergence trajectory. Yet, after that point, LUCE’s convergence made a sharp turn for worse and never recovered from it. The situation is slightly different for the Chicago regional network (see Figure 2-b), where LUCE had not yet reached its critical gap observed for all the other networks within the committed computation time. Interestingly, the performance of LUCE seems to always lag behind that of OBA after they passed the relative gap of $10^{-4}$.

That the two bush-based algorithms would perform similarly is not a big surprise given how alike they are in the overall structure. Notwithstanding, such a lack of improvements may be disappointing to those who enthusiastically hypothesize the greedy algorithm would significantly improve OBA because of its
ability to achieve exact subproblem solution with a closed form computation. What is more surprising, however, is that LUCE seems to get into a similar standstill as FW for all networks, even though it usually occurs at a much smaller relative gap. In contrast, OBA was able to converge well into the vicinity of the floating number limit ($10^{-15}$) for the small networks. The trend revealed from Figure 2 suggests that it likely does the same for the larger networks should enough computation time be allowed. It is true that the level of precision beyond $10^{-10}$ is of little practical significance. Yet, the point is that LUCE’s worse-than-expectation performance seems difficult to explain. We set out to search the answer to this question in the next section.

4 Discussions

The performance of bush-based algorithm are affected by many factors. It is usually difficult to pinpoint which factor is to blame for unexpected performance such as observed above. However, in our implementation, LUCE and OBA share most code except for Step 2.1 in the inner loop. It is thus reasonable to attribute the discrepancy in their performance to that step, which involve solving a sequence of node-based assignment problems. Before we thoroughly inspect that part, however, let us first examine LUCE’s convergence behaviour more closely.

4.1 Why LUCE stops converging?

To understand why LUCE stops converging after certain point, we track the changes in the bush structure in the outer loop. Intuitively, as the solution approaches the true equilibrium, the number of links that can be added into the current bush after equilibration should diminish. When no links can be added, the solution reaches the equilibrium. In light of this observation, we propose to measure the bush change using the following index (which is called the average optimality gap in [14]):

$$\tau = \frac{\sum_{r \in R} |P^r|}{|A||R|}$$  

(6)

where $P^r$ is the number of links added into the bush rooted at $r$ in Step 1 of the outer loop, $A$ is link set and $R$ is origin set. Figure 3 plots the convergence pattern of $\tau$ for the Anaheim and Chicago sketch networks.
Comparing Figure 3 to Figure 1, an immediate observation is that the time when LUCE stops converging is exactly the same time when $\tau$ begins the significant oscillation around a constant value. Why did $\tau$ oscillate so much in LUCE? A reasonable explanation is that LUCE’s RMP solver fails to further improve bush equilibration beyond certain point. Unable to distinguish which are actually on the optimal bush among a set of competitive links, LUCE’s outer loop begins to randomly pick links from iteration to iteration. Now, why does LUCE fail to equilibrate bush to very high precision, equipped with a seemingly better solver - the greedy algorithm? To this question we now turn.

![Fig. 3: Convergence of average optimality gap](image)

**4.2 Greedy method and GP method**

In Step 2.1 of the inner loop, both LUCE and OBA perform a backward pass to solve a series of node-based assignment problems. In essence, each node problem is an elementary assignment problem allocating the total flows into a node to each of its approaching link such that the cost is equilibrated among all those links. Solving this node problem is very similar to solving a traffic assignment problem on a network described in Figure 4. Certainly, the actual number of parallel links would depend on the bush topology. Yet, for most transportation networks, that number rarely exceeds 2 or 3.

![Fig. 4: Simple network](image)

We now present the greedy and GP method in the context of solving the traffic assignment problem described in Figure 4. The performance function of three links are given by

$$
t_1 = 10[1 + 0.15\left(\frac{x_1}{2}\right)^4],
t_2 = 20[1 + 0.15\left(\frac{x_2}{4}\right)^4],
t_3 = 25[1 + 0.15\left(\frac{x_3}{3}\right)^4]
$$

(7)
and the total demand between \( o \) to \( d \) is \( q = 10 \). The formulation of the assignment problem on this network can be stated as:

\[
\min z(\phi) = \sum_n \int_0^{x_n} t_n(w) dw, \quad n = \{1, 2, 3\} \tag{8}
\]

Subject to:

\[
x_n = q\phi_n, \quad \forall n \in \{1, 2, 3\} \tag{9}
\]

\[
\phi_1 + \phi_2 + \phi_3 = 1 \tag{10}
\]

\[
\phi_n \geq 0, \quad \forall n \in \{1, 3, 4\} \tag{11}
\]

To solve this problem, the following iterative procedure may be used:

**A general assignment algorithm:**

- **Step 0** Assign \( q \) on three links arbitrarily to get \( x_n^0 \), and calculate \( \varphi_n = x_n^0/q, \forall n \), initialize convergence indicator \( \epsilon \). Set iteration \( k = 0 \).

- **Step 1** Update link travel time \( t_n^k \) and its derivative \( g_n^k \), \( \forall n \in A \)

- **Step 2** Convergence test: Let \( u \) denotes the shortest cost, if \( 1 - \frac{uq}{\sum_n t_n^k x_n^k} < \epsilon \), stop; else go to Step 3.

- **Step 3** Approximate the objective function (8) by a quadratic function

\[
z_k(\phi) = \sum_n \int_0^{x_n} (t_n^k + g_n^k x_n^k) dw, \tag{12}
\]

then solve the assignment problem with the quadratic objective function either by the greedy method (in LUCE) or GP (in OBA) method.

- **Step 4** Set \( k = k + 1 \), update \( x_n^k = q\phi_n^k, \quad \forall n \in A \), go to Step 1.

The greedy method for solving the quadratic approximation problem is described below [14]. Note that the iteration index is dropped for simplicity.

**Greedy method:**

1. **3.0** Set \( H = \{ n \in A : \varphi_n > 0 \} \)
2. **3.1** Estimate the UE cost by \( w = [q(1 - \sum_{n \in H} \varphi_n) + \sum_{n \in H} \frac{t_n}{g_n}]/(\sum_{n \in H} \frac{1}{g_n}) \)
3. **3.2** Let \( \Delta H = \{ n \in H : t_n - (\varphi_n g_n q) > w \} \)
4. **3.3** If \( \Delta H \neq \emptyset \) then \( H = H \setminus \Delta H \), go to 3.1; else goto 3.4.
5. **3.4** Calculate \( \phi_n = [w - (t_n - \varphi_n g_n q)]/(g_n q), \forall n \in H ; \phi_n = 0, \forall n \in A \setminus H \). Go to Step 4.

Gentile [14] shows that the above algorithm actually finds the exact solution to the approximation problem. In contrast, the following gradient projection attempts to solve the approximation problem using the steepest direction method, but it only carries out one iteration.

**One-step GP method:**
• 3.0 Set $\lambda = 1.0$

• 3.1 Find the link with smallest $t$ marked as $t_n$, calculate the new flow $\phi_n$ for each link $n \neq \bar{n}$ as follows

$$\phi_n = \varphi_n - \min \left\{ \varphi_n, \frac{t_n - t_{\bar{n}}}{q(g_n + g_{\bar{n}})} \right\}$$

$$\phi_{\bar{n}} = 1 - \sum_{n \neq \bar{n}} \phi_n$$

• 3.2 Compute the social pressure $\rho = \sum_{n \in \{1, 2, 3\}} t_n (\phi_n)(\phi_n - \varphi_n)$ [11]. If $\rho > 0$, let $\lambda = 0.9\lambda$ and go to 3.1; else go to Step 4.

Step 3.2 in the above algorithm is necessary because the one-step approximation may not necessarily guarantee descending. As shown in [11], the so-called social pressure can be used to correct a non-descending solution resulted from the one-step approximation. A seemingly obvious method to improve the quality of solutions to the quadratic approximation is to perform more GP iterations. A multi-iteration GP method is described below.

**GP method:**

• 3.0 Set $\hat{t}_n = t_n$, $\hat{\varphi}_n = \varphi_n$, $\hat{\phi}_n = 0$, $\forall n$, set $\sigma = 1e - 15$

• 3.1 Find the smallest $\hat{t}$ marked as $\hat{t}_{\bar{n}}$, calculate the new flow $\hat{\phi}_n$ for each link $n \neq \bar{n}$ as follows:

$$\hat{\phi}_n = \hat{\varphi}_n - \min \left\{ \hat{\varphi}_n, \frac{\hat{t}_n - \hat{t}_{\bar{n}}}{q(g_n + g_{\bar{n}})} \right\}$$

$$\hat{\phi}_{\bar{n}} = 1 - \sum_{n \neq \bar{n}} \hat{\phi}_n$$

• 3.2 Calculate the $\hat{t}_n$ by the following cost function:

$$\hat{t}_n = t_n + g_n q(\hat{\phi}_n - \varphi_n), \forall n$$

• 3.3 Check the UE condition: estimate the UE cost by $w = \sum_n \hat{t}_n / N$, $N$ is the links number. Let

$$\lambda = \max \{\text{abs}(\hat{t}_n - w)\}$$

if $\lambda < \sigma$, go to Step 4; otherwise, let $\hat{\varphi}_n = \hat{\phi}_n$, $\forall n$ and go to 3.1.

Having explained all three methods, we now apply them to solve our toy problem. We set the initial solution arbitrarily as (2,2,6). Figure 5-a reports the convergence curves of the three methods. As expected, the convergence pattern of GP is exactly same as that of the greedy method in this example because they both solve the quadratic program exactly in each main iteration and use that solution to generate the search direction. In contrast, one-step GP only solves the quadratic problem approximately, so it takes much more main iterations to reach the same level of convergence. Hence, the greedy method indeed outperforms one-step GP in this example. Then why does LUCE in the real tests fail to demonstrate such superiority? We note that the quadratic problem itself is an approximation to the original problem. When it is a good approximation, solving it precisely undoubtedly helps the overall convergence. Yet, how well the quadratic
problem approximates the original problem depends on many factors: the form of link performance function, network topology, congestion level, to name a few. If the quadratic approximation is poor due to, for example, the inaccurate estimation of the second order derivatives, it may be counter-productive to solve it precisely.

To show this point clearly, let us use a two-link example (see [14]) to illustrate the difference between greedy and GP method. Let $\varphi$ be the current solution, the tangent lines $\hat{t}_1(\varphi_1)$ and $\hat{t}_2(\varphi_2)$ are the estimated link travel time calculated based on $\varphi$, and the slopes of them are derivatives of link travel time. $\varphi^{gr}$ is the next solution computed by greedy method, which is the crossing point of two tangent lines. The solution of one-step GP falls into the range between $\varphi$ and $\varphi^{gr}$, but exactly where it will be depends on the step size. Most likely, one-step GP will not reach the point $\varphi^{gr}$.

Suppose now that the derivative of the second link $g_2$ cannot be estimated correctly \(^2\), then the estimated link cost is not the tangent line of $t_2(\varphi_2q)$ any more. Assume it is $\hat{t}_3(\varphi_2q)$ for example in Figure 5-b. In this case, the solution of greedy method would be $\hat{\varphi}^{gr}$, which is even worse than the current solution $\varphi$. In comparison, the solution of one-step GP would still be located between $\varphi$ and the new crossing point $\hat{\varphi}^{gr}$. Importantly, the use of social pressure in Step 3.2 can make sure that the next solution is always better than $\varphi$.

To verify the above analysis, we artificially introduce errors into the second-order derivative $g_a$ in our toy example. Specifically, we multiply $\{g_n, n \in \{1, 2, 3\}\}$ with a coefficient $\theta > 0$ in every iteration. As reported in Figure 6, when $\theta = 0.70$, the greedy method is comparable to GP; when $\theta = 0.58$, the efficiency of greedy algorithm is further reduced. However, with the help of social pressure, one-step GP can maintain the convergence performance even with the faulty second-order information. As it turns out, the guidance provided by the social pressure is very important in large networks. Unfortunately, an algorithm like LUCE or GP cannot take advantage of this guidance because they are designed to reach that crossing point, somewhat blindly.

In a nutshell, the above analysis offers a plausible explanation for the mysterious convergence behavior of LUCE. That is, to properly solve the node-based subproblem, the greedy method used by LUCE requires highly accurate estimation of second-order derivatives. When second-order derivatives are subject to large errors, the greedy method can provide consistently sub-optimal descent direction.

\(^2\)For this example, the derivative can always be computed correctly. Yet, in the context of solving the traffic assignment using bush-based algorithms, the second-order derivatives are very difficult to estimate with high accuracy, see [15] for discussions.
5 Conclusions

As a recent addition to the family of bush-based algorithms, LUCE is closely related to Bar-Gera’s OBA. LUCE’s promise comes mainly from its use of the greedy method for solving the decomposed node-based problems. Ironically, this new feature turns out to be a two-edged sword. On one hand, the greedy algorithm can accelerate the solution of the subproblems. On the other hand, it could disrupt the smooth convergence behaviour shared by other bush-based algorithm such as OBA and Algorithm B, as analysed in Section 6. Our numerical experiments clearly showed how bad the disruptions could become in real networks. However, the benefits that LUCE is supposed to bring to the table are not clear in these experiments.

One can argue that LUCE’s convergence performance is good enough because for most practical purposes a relative gap of $10^{-6}$ likely suffices. LUCE does seem to be able to achieve such levels of convergence comfortably, although it is not necessarily faster than OBA based on our experiments. It should be pointed out, however, that the best relative gap that LUCE can achieve seems to depend on how large the errors are in the estimation of second-order derivatives.

The convergence performance of LUCE may be improved in various ways. Possibilities include (1) applying the upper bound estimation to the second order derivatives, as suggested in [15], and (2) updating upstream approach costs and derivatives after each node sub-problem is solved, as suggested in [13]. While investigating how LUCE would respond to these refinements certainly constitutes an worthwhile effort, it is unlikely that they will dramatically change the relative performance between OBA and LUCE.

Acknowledgements

We are grateful to Professor David Boyce at Northwestern University, who have read an earlier draft of this paper and offered many valuable comments. We would also like to thank Klaus Noekel at PTV Group for sharing the PRISM network through Professor Boyce. The first author would like to thank China Scholarship Council for providing a scholarship that allows him to visit Northwestern University between 2010 and 2012, during which this work was conducted. The remaining errors and mistakes are the authors’ alone.
References


