Comparison of Optimization Methods for Assisted Calibration of Traffic Micro-Simulation

David K. Hale¹, Constantinos Antoniou², Mark Brackstone³, Dimitra Michalaka⁴, Ana T. Moreno⁵, and Kavita Parikh⁶

¹Leidos, Inc.
11251 Roger Bacon Drive
Reston, VA 20190
david.k.hale@leidos.com

²National Technical University of Athens
School of Rural and Surveying Engineering
Laboratory of Transportation Engineering
antoniou@central.ntua.gr

³IOMI Consulting
Southampton
Hants, UK, SO40 7FF
mark.brackstone@iomi.eu

⁴The Citadel
171 Moultrie Street
Charleston, SC 29409
Dimitra.Michalaka@citadel.edu

⁵Universitat Politecnica de Valencia
Camino de Vera, S/N, Edificio 4ª
46071 Valencia, Spain
anmoch@cam.upv.es

⁶RS&H, Inc.
10748 Deerwood Park Blvd South
Jacksonville, FL 32256
Kavita.parikh@rsandh.com

October 2014

Word Counts:

Abstract and Manuscript Text: 5,250
Number of Tables and Figures: 9 (= 2,250 words)
Total: 7,500
ABSTRACT

Usage of traffic simulation has increased significantly in recent decades; and this high-fidelity modelling, along with moving vehicle animation, has allowed important transportation decisions to be made with better confidence. During this time, traffic engineers have typically been encouraged to embrace the process of calibration, in which steps are taken to reconcile simulated and field-observed traffic performance. According to international surveys, top experts, and conventional wisdom, existing (non-automated) methods of calibration have been difficult and/or inadequate. There has been a significant amount of research on techniques to improve calibration, but many of these projects and papers have not provided the level of flexibility and practicality typically required by real-world engineers. With this in mind, a patent-pending (US 61/859,819) architecture for software-assisted calibration was developed to maximize practicality, flexibility, and ease-of-use. This architecture is called SASCO (i.e. Sensitivity Analysis, Self-Calibration, and Optimization). The original optimization method within SASCO was based on “directed brute force” (DBF) searching; performing exhaustive evaluation of alternatives in a discrete, user-defined search space. Simultaneous Perturbation Stochastic Approximation (SPSA) has also gained favor as an efficient method for optimizing computationally expensive, “black-box” traffic simulations, and was also implemented within SASCO. This paper assesses the qualities of DBF and SPSA, so they can be applied in the right situations. Test results imply the two optimization methods have different advantages, and in some cases should be applied in tandem. Regardless of which optimization method is selected, the SASCO architecture appears to offer a new and practice-ready level of calibration efficiency.

Key words: Microscopic Simulation, SASCO, SPSA, Assisted Calibration, Calibration, Simulation-Based Optimization.
1. INTRODUCTION

Computer programs for traffic simulation have become more advanced in recent decades. Use of simulation has increased significantly; and this high-fidelity modeling, along with moving vehicle animation, has allowed important transportation decisions to be made with better confidence. During this time, traffic engineers have been encouraged to embrace the process of calibration, where steps are taken to reconcile simulated and field-observed performance. One example of such encouragement can be found in the Federal Highway Administration guidelines for applying micro-simulation modeling software [1]. They state “the importance of calibration cannot be overemphasized”; and refer to a study [2] by Bloomberg et al., which makes the following statement: “Recent tests of six different software programs found that calibration differences of 13 percent in the predicted freeway speeds for existing conditions increased to differences of 69 percent in the forecasted freeway speeds for future conditions.” Therefore, these studies and guidelines demonstrate some of the dangers of neglecting calibration.

Despite the importance of calibration, practical application has been difficult. According to international surveys, top experts, and conventional wisdom, existing (non-automated) methods of calibration have been difficult and/or inadequate. Consulting engineers and DOT personnel have expressed strong interest in making calibration faster, cheaper, easier, and requiring less expertise. Comprehensive surveys [3] revealed that 19% of simulation users do not perform any amount of calibration, and that only 55% of calibration efforts are based on guidelines that exist in the literature [1, 4, 5]. Finally, some simulation users believe that they have somewhat mastered the process of calibration; but that many years of experience are required to master this process, or that high-quality calibration is overly time-consuming.

There has been significant research to improve calibration for traffic simulation. Some of this research focuses on traffic assignment and origin-destination flows. Other research focuses on pattern matching of simulated versus field-measured vehicle trajectories and/or speed-flow relationships. A third set of research focuses on simulation-based optimization, where the numeric discrepancy between simulated and field-measured results becomes an objective function to be minimized. Many of these projects and papers have not provided the flexibility and practicality typically required by real-world engineers. In the papers by Lee and Ozbay [6], Lee et al. [7], and Menneni et al. [8], the authors present substantial literature reviews for both manual and automated calibration techniques. The authors then emphasize that, despite the extensive efforts, existing calibration procedures continue to require excessive time and expertise.

With this in mind, a patent-pending (US 61/859,819) architecture for software-assisted calibration was developed, in the simulation-based optimization (SO) family of methods. The new architecture uses a database of input parameters, to pre-define a narrow set of trial values to be used during optimization. The architecture also allows engineers to prioritize input and output parameters, and specify a tolerable computer run time, prior to initiating the SO-based calibration process. The “directed brute force” (DBF) search process is believed to be a key element in making the architecture flexible and practical. These same features were later extended to provide an enhanced platform for sensitivity analysis, and optimization. The acronym “SASCO” (Sensitivity Analysis, Self-Calibration, and Optimization) was adopted. Thus, SASCO provides a database-centric framework to support all three of these analysis types.

In recent years, Simultaneous Perturbation Stochastic Approximation (SPSA) has gained favor as an efficient method for optimizing computationally expensive, “black-box” traffic simulations. For example in 2007, Balakrishna et al. [9] selected SPSA “for its proven
performance and computational properties in large-scale problems”. In 2007-2008, Ma et al. [10] and Lee’s dissertation [11] both demonstrated the effectiveness of calibrating PARAMICS traffic simulations with SPSA. Also in 2008, the Transportation Research Board posted a research needs statement [12] for calibrating simulation models, saying “recent research indicates that the SPSA algorithm can solve very large noisy problems in a computationally attractive fashion.” More recently in 2013, Paz et al. [13] demonstrated the effectiveness of calibrating CORSIM simulations with SPSA. Given this track record, SPSA appears to be an attractive option for assisted calibration of computationally expensive simulations, and a possible alternative to the DBF optimization originally implemented within SASCO.

As such, this paper will assess the qualities of DBF and SPSA, so they can be applied in the right situations. The scope is limited to calibration (without validation) of average output values (not distributions), based on data collection from one real-world corridor, using one underlying simulator (FRESIM, which is part of CORSIM). Follow-up studies will hopefully perform similar assessments with more optimization methods, validation following calibration, calibration of output distributions, on a variety of real-world corridors, using additional simulators.

2. SASCO ARCHITECTURE

As stated earlier, SASCO can be classified as belonging to the simulation-based optimization (SO) [14] family of methods. Under SO, numerous input combinations are simulated for the purpose of minimizing or maximizing an objective function. Several of the papers cited by [6, 7, 8] investigated SO-based calibration via genetic algorithms (GA) [15]. GA is a well-known, intelligent heuristic searching method. The heuristic methods are able to continuously adapt their search in response to intermediate trial results. Although GA appears to have gained commercial success in multiple industries, it has not gained commercial popularity for calibration of traffic simulations. This is likely due to the fact that GA frequently requires thousands of trials to locate an acceptable solution, and the traffic simulations cannot process thousands of trial runs in a reasonable time frame. Other heuristic methods requiring a relatively large [16, 17] number of trials, such as downhill simplex or simulated annealing, also do not seem suitable for computationally expensive simulation models. On the other hand, faster heuristics such as hill-climbing and the greedy algorithm are known to produce unsatisfactory [18, 19] solutions.

When developing a new methodology for SO-based calibration, it was believed the end-user needed complete control over the run time, number of trials, and the trial values themselves. Given the wide variety of computer run times for various software packages and traffic networks, a “one size fits all” searching method seemed impractical. Moreover, different jurisdictions have different standards and tolerances for calibration. Finally, different analysis types (small corridor, large network, academic research) require substantially different amounts of calibration. Although the run time for more powerful heuristics can be arbitrarily controlled, for example with small annealing schedules or numbers of generations, this was likely to produce unacceptable calibration results. Giving the end-user total control over the run time, number of trials, and the trial values themselves seemed the best solution. This led to the development of a database-centric architecture; designed with directed brute force optimization in mind, and described in this section.

2.1 Input Data User-Interface

Figure 1 illustrates an example user-interface (UI) software screen design, for selecting inputs to be calibrated. This UI screen loads an input parameter database at runtime; thus databases
can be customized or updated at any time, without needing new software. The end-user need only put a check mark next to each input parameter they'd like to calibrate, and then select a calibration thoroughness level (Quick, Medium, or Thorough) for each input selected.

The input data UI can provide valuable run time estimates. In the concept of fast food menus discussed earlier, if a customer decided the total price was too high, they could change their order prior to purchase. In the case of directed brute force optimization, if an end-user chose Quick searching (3 trials) on one input parameter, Medium searching (5 trials) on a second, and Thorough searching (10 trials) on a third, run time from an initial simulation could be multiplied by 150 (i.e., $3 \times 5 \times 10$), to produce a reasonable estimate. If the end-user could afford a longer run time, they might choose more thorough searching on some parameters, or add to the list of parameters to be calibrated. If the estimated run time were too high, they might reduce the amount of searching on some inputs, or perhaps omit certain inputs. In the case of Simultaneous Perturbation Stochastic Approximation (SPSA), run time estimates could reflect the max number of iterations, and the number of simulations per iteration. Moreover, by allowing the end-user to adjust range limits (continuous SPSA) or trial values (discrete SPSA) for any input, SASCO’s input data UI could augment the efficiency of SPSA.

**FIGURE 1:** Example user-interface design for selecting calibration input parameters

### 2.2 Output Data User-Interface
According to comprehensive literature reviews [6, 7, 8], prior methods offer a limited set of output parameters for calibration. By contrast, SASCO architectures allow any output value, or distribution [20] of output values, to be used for calibration. The software must also accept data entry for “ground truth” output values; which can be measured in the field, estimated from the office, or obtained from a separate traffic analysis tool. Figure 2 illustrates an example screen design for selecting outputs. Similar to the input data UI, the end-user can simply put a check mark next to each output they’d like calibration to be based on. Priority weighting defaults to 100 for each output, and can be left alone if all outputs are deemed equally important. The total percent difference (between simulated and field-measured outputs) appears at the bottom; responding to data entry in real-time, and reflecting priority weightings.

Because the full set of outputs is not simultaneously viewable, software controls must allow browsing amongst all outputs. The database-centric, fast-food ordering concept allows calibration to be based on cumulative outputs, time period-specific outputs, global outputs, link-specific outputs, surveillance detector outputs, surface street outputs, freeway outputs, or any combination of the above. The end-user can provide ground truth values for any number of output parameters, depending on their needs. Input and output calibration settings entered by the user are continuously saved into a separate file (e.g., “Filename.self”); for future reference, and to avoid re-typing. The need for transparency in calibration has been cited by top experts; the “self” file provides this transparency. The UI design shows it is not necessary to load all outputs simultaneously. Although there may be millions of values, it is only necessary to load outputs consistent with active choices on screen. Most traffic analysis tools support map-based data entry via right-clicking on links and nodes; so for large networks with many links, the software should support jumping to the UI in Figure 2, and switching to the chosen link.

**FIGURE 2**: Example user-interface design for selecting calibration output parameters

### 2.3 Optimization Algorithm
After the end-user chooses inputs and outputs, any optimization method can be applied. Powerful heuristics (e.g., GA, simulated annealing) require an excessive and unpredictable number of trials, and are not suited to computationally expensive simulations. Simpler heuristics (e.g., hill-climbing, greedy algorithms) tend to produce poor [18, 19] solutions. By contrast, SASCO was designed to calibrate expensive simulations. End-users can customize trial values and run times for directed brute force (DBF) searching. The only difference between “directed” and “regular” brute force is that DBF is restricted to fewer trial values. In the earlier example, Quick searching (3 trials) on one input, Medium (5 trials) on a second, and Thorough (10 trials) on a third leads to 3*5*10=150 possible solutions. The number of trials for Quick, Medium, and Thorough is flexible. The database offers intelligent defaults, but also allows customization. SASCO was developed with DBF in mind, but Simultaneous Perturbation Stochastic Approximation (SPSA) has gained favor for efficient optimization of complex simulations. For the discrete and continuous forms of SPSA, the database can specify trial values and range limits for each input, respectively. Given that no optimization method outperforms all others under all conditions [21], it is important for SASCO to support multiple methods.

Figure 3 illustrates the overall SASCO architecture. When the run is launched, the optimization algorithm (e.g., DBF or SPSA) proceeds to minimize the objective function, by testing acceptable input values from the input data UI. Throughout the run objective function values become lower and lower, thus indicating a better-calibrated model.

FIGURE 3: Overall SASCO architecture

3. REAL-WORLD NETWORK CASE STUDY
To assess DBF and SPSA, tests were performed using FRESIM as the simulator. Figure 4 illustrates I-95 near Jacksonville FL, having three mainline lanes. Mainline and ramp free-flow speeds were 113 and 40 km/h. The model contained twelve 15-minute time periods, with 7% trucks. Vehicle speeds were obtained in the peak hour on twelve days (periods 5 through 8). Locations “C”, “D”, and “E” exhibited the largest discrepancy between real speeds (averaged over twelve days), and simulated speeds, in km/h. Table 1 illustrates peak-hour flow rates.

![I-95 network roadway geometry](image)

**FIGURE 4: I-95 network roadway geometry**

<table>
<thead>
<tr>
<th></th>
<th>TP5</th>
<th>TP6</th>
<th>TP7</th>
<th>TP8</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB mainline</td>
<td>2644</td>
<td>3240</td>
<td>2780</td>
<td>3032</td>
</tr>
<tr>
<td>SB off-ramp at &quot;C&quot;</td>
<td>1144</td>
<td>1456</td>
<td>1024</td>
<td>1084</td>
</tr>
<tr>
<td>NB on-ramp at &quot;C&quot;</td>
<td>1208</td>
<td>992</td>
<td>1124</td>
<td>1252</td>
</tr>
<tr>
<td>NB mainline</td>
<td>3800</td>
<td>3488</td>
<td>3468</td>
<td>3484</td>
</tr>
</tbody>
</table>

**TABLE 1: I-95 network flow rates (in vehicles per hour) across four time periods**

At location C, the discrepancy in speeds was large in both northbound and southbound (NB and SB) freeway directions. Location D represents a SB segment, whereas E is a NB segment. After entering field-measured speeds for the four locations and time periods into the output data UI, SASCO showed an overall difference of 4.3% versus simulated results. Differences were highest (10-20%) in location C, and 0-5% elsewhere. Preliminary stochastic analysis, involving 8 simulations with 8 sets of random number seeds, indicated this percentage difference was closer to 4.5% than 4.3%. Thus, the original objective function value was 4.5%.
3.1 Preliminary Univariate Calibration

A nearby interchange exists just north of location C, but was not included in the simulation dataset. Because of this, and because speeds were known to be inaccurate in location C, there was reason to believe drivers exhibited more aggressive car-following behavior between those interchanges. Thus a preliminary calibration of car-following sensitivity multipliers was performed on the seven SB segments upstream of location C. SPSA optimization was able to reduce the objective function value from 4.5 to 3.5% within 13 iterations (26 simulations). DBF reduced the objective function value from 4.5 to 3.0% within a similar number of simulations. The optimum car-following multiplier recommended by SPSA was 72.7%, whereas 74.0% was produced by DBF. In both cases, “post-stochastic” analysis (same as preliminary stochastic analysis, but with optimum car-following multipliers) increased the percent difference to 3.6%. Thus the end result of preliminary calibration was a car-following sensitivity multiplier of 74.0% on seven segments, and an objective function value of 3.6%. Following these runs it was decided no further localized (i.e., link-specific) calibrations were justified, and that all subsequent calibrations would involve “global” (i.e., network-wide) input parameters.

3.2 Intermediate Sensitivity Analysis

At this stage 16 global inputs were available for possible calibration, but simultaneous DBF calibration of 16 inputs would require huge computer run times. Moreover, it was believed simultaneous SPSA calibration of 16 inputs might also cause excessive run times and/or highly suboptimal solutions. As such, intermediate sensitivity analysis (SA) runs were performed to evaluate the remaining 16 inputs. By exhaustively simulating every trial value, every DBF optimization produces SA results. Thus the 16 inputs were examined one by one, with 16 runs. This series of DBF SA runs implied that 9 out of 16 inputs reduced the objective function value (i.e., percent difference between simulated and field-measured results), known as “diff”, by more than 0.1%. However one of these inputs was Desired Ramp Free-Flow Speed, whose optimum value was higher than 97 km/h. Although the extremely high ramp speeds would allow simulated mainline speeds to match field-measured mainline speeds more closely, these ramp speeds would be unrealistic. After discarding this input there were 8 remaining, but post-stochastic analysis implied only 5 of them reduced diff by more than 0.1%:

- Vehicle Entry Headway
- Time to Complete a Lane Change
- Minimum Entry Headway
- Percentage of Cooperative Drivers
- Off-Ramp Reaction Distance

Optimizing Vehicle Entry Headway with SPSA would be difficult. These headways consist of discrete words (i.e. Normal, Erlang, Uniform), and numeric values associated with each word. The combination of DBF searching and the input parameter database is ideal for optimizing unusual inputs like this, but SPSA seems designed to optimize numbers only. Therefore the Vehicle Entry Headway was optimized by DBF SA, producing a diff value of 3.1% within 7 simulations. Post-stochastic analysis then decreased diff from 3.1 to 3.0%. Thus the end result of intermediate SA was a new Vehicle Entry Headway distribution, an objective function value of 3.0%, and four remaining input parameters to be optimized.
3.3 Final Multivariate Calibration

At this stage there were 4 global inputs remaining. Following the calibration of Vehicle Entry Headway, DBF SA runs implied that Minimum Entry Headway and Percentage of Cooperative Drivers would no longer reduce diff by more than 0.1%. Time to Complete a Lane Change (TTCLC) and Off-Ramp Reaction Distance (ORRD) were simultaneously calibrated by DBF; with 7*8=56 simulations, in less than one hour. This optimization located a diff value 1.7% at TTCLC of 4.5 seconds, with ORRD between 1050 and 1650 meters. Post-stochastic analysis then increased diff from 1.7 to 1.9%. Thus the end result of a typical DBF calibration would be an objective function value of 1.9%, with several inputs optimized.

However to set the stage for detailed comparisons between DBF and SPSA, TTCLC and ORRD were analyzed with 17*18=306 simulations (four hours). The resulting Table 2 shows optimal ranges for TTCLC (5.5-7.5) and ORRD (1200-1800), but reveals inconsistent diff values caused by stochastic noise. According to Table 2, the quick DBF calibration based on only 56 runs did a fairly good job of locating a near-optimum solution in a reasonable time frame.

**TABLE 2: Objective function (“diff”) values for the Jacksonville case study**

<table>
<thead>
<tr>
<th>Time to Complete a Lane Change (sec)</th>
<th>avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off-Ramp Reaction Distance (m)</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>2.6</td>
</tr>
<tr>
<td>300</td>
<td>2.6</td>
</tr>
</tbody>
</table>

Figure 5 illustrates optimized “theta” values over 100 iterations (200 simulations) of SPSA. The top two graphs were seeded with good starting points near the optimal range. Each graph compares four optimization runs. In “2P” and “2N”, TTCLC and ORRD were simultaneously optimized with wide and narrow range limits, respectively. In “4P” and “4N”, Minimum Entry Headway and Percentage of Cooperative Drivers were simultaneously optimized along with TTCLC and ORRD, using wide and narrow range limits. These tests were performed because it was believed SPSA would be more efficient under narrow ranges (e.g. 4.5-7.5 instead of 2.0-10.0), and when optimizing fewer inputs (e.g. TTCLC and ORRD by themselves, instead of TTCLC and ORRD with two additional inputs).
Based on Table 2 results showing optimal ranges for TTCLC (5.5-7.5) and ORRD (1200-1800), Figure 5 implies the “bad” starting points indeed cause less efficient optimization. Regarding the comparison between optimizing 2 or 4 inputs, it appears SPSA handled them with similar efficiency, although the two “extra” parameters were known to have little impact on results. Finally, the narrow ranges were usually more effective than wide ranges, implying that SASCO’s range-setting features would likely augment the efficiency of SPSA.

Figure 5 results imply that bad starting points can make SPSA highly ineffective, but this is not the only way SPSA can be evaluated or applied. Trial values (“xplus”, “xminus”) are quite different than “theta”, especially in early iterations, and often produce very good results. Figure 6 shows that all SPSA optimization runs produced trial values resulting in diff values below 2.0%, in 30 simulations or less. This was true even when simultaneously optimizing 4 inputs, having wide ranges and bad starting points. Thus the end result of DBF or SPSA calibration was a diff reduction from 4.5% to below 2.0%. Moreover, location C diff values dropped from 10-20% to below 5%.

One caveat is that several optimizations terminated prematurely, and had to be re-run with better internal parameter values. The effectiveness of SPSA is also known to be dependent on its starting point, and its internal optimization parameters [22]. Moreover, SPSA is known to sometimes converge on local optimum solutions instead of global optimum solutions [23]. Some research [6, 24] indicates there are special and/or customized ways SPSA can be implemented, to facilitate escaping local optima. In the end, it appears SPSA is capable of better efficiency than DBF if 1) proper internal parameters are chosen, 2) trial values are used instead of theta values, and 3) convergence is not required. However SPSA can be inefficient when not applied properly,
is less suitable for sensitivity analysis, and cannot explicitly optimize interdependent or non-numeric inputs.

**FIGURE 6a: Trial (xplus, xminus) objective function (diff) values (good starting points)**

**FIGURE 6b: Trial (xplus, xminus) objective function (diff) values (bad starting points)**

3.4 Simultaneously Optimizing all Inputs with SPSA

In the I-95 case study, inputs were calibrated systematically. First, preliminary univariate calibration was used to reflect driver aggression between interchanges. Then, sensitivity analysis was used to determine high-impact inputs. Finally, multivariate calibration was used to optimize high-impact inputs. This procedure is necessary for DBF, which cannot simultaneously optimize
numerous inputs in a reasonable time frame. However, SPSA can do it. The question is how well can it do it, and are such optimizations effective?

The simulator used in the case study was FRESIM, having approximately 16 input parameters relevant to the calibration process. Three of these inputs were interdependent distributions of numbers and/or discrete words; which could not be optimized by SPSA without “encoding algorithms”, which were not developed during this research. One of these inputs was Desired Ramp Free-Flow Speed, whose optimum value (113 km/h) was judged as unrealistic during the DBF calibration. Therefore, a new set of SPSA optimization experiments was performed; to calibrate the remaining 12 inputs simultaneously, instead of sequentially.

For this experiment a discrete form of SPSA was applied, as there was insufficient time to update the software for continuous SPSA optimization of 12 inputs. It is noted that discrete SPSA might be less efficient than continuous. Secondly, local calibration was not performed near location C, as car-following sensitivity was instead calibrated globally. Third, the number of simulations was doubled compared to the sequential optimizations. Finally, the internal SPSA random number (SRN) for generating “delta” (+1 or -1) values was also varied, to assess variability within the optimization process.

Results of the prior experiments (sequential optimization) indicated that trial values (e.g., from Figure 6) provide the fastest way to obtain good solutions. Previous results also showed that a “good solution” would mean reducing the original diff value (4.5%) to somewhere below 2.0%. Table 3 illustrates the number of trial values (simulations) needed before locating any solution below 2.0%. A total of 6 optimization runs (2400 simulations) were performed, based on 3*2 = 6 combinations of internal (“a” and “c”) parameter values and SRN values.

<table>
<thead>
<tr>
<th>SRN</th>
<th>7783</th>
<th>7783</th>
<th>7783</th>
<th>7781</th>
<th>7781</th>
<th>7781</th>
</tr>
</thead>
<tbody>
<tr>
<td>param_a</td>
<td>0.1</td>
<td>0.2</td>
<td>0.3</td>
<td>0.1</td>
<td>0.2</td>
<td>0.3</td>
</tr>
<tr>
<td>param_c</td>
<td>0.1</td>
<td>0.2</td>
<td>0.3</td>
<td>0.1</td>
<td>0.2</td>
<td>0.3</td>
</tr>
<tr>
<td># of runs</td>
<td>146</td>
<td>106</td>
<td>189</td>
<td>30</td>
<td>28</td>
<td>n/a*</td>
</tr>
</tbody>
</table>

**TABLE 3: Number of simulations before SPSA located diff below 2.0%**

Table 3 illustrates the extent to which SPSA might be affected by its internal parameter settings. In the best-case scenario, a solution below 2.0% was located after 28 simulations, which is drastically more efficient than DBF. However under a different set of SRN, changing delta values caused the number of required simulations to increase from 28 to 106. Furthermore, a slightly different set of “a” and “c” values caused the number of simulations to increase from 106 to 189. Finally, one combination of internal values prevented any solution below 2.0% from being located within 400 simulations, although a reasonably* good solution (2.1%) was found within 28 simulations. Similar to the sequential results, these simultaneous optimization results suggest that SPSA has the potential for extremely fast calibration. Unfortunately this efficiency depends on internal parameter settings; and the impact of random delta values, revealed by changing SRN, was disconcerting.

4 CONCLUSIONS

Because the existing (non-automated) methods of calibration have been difficult and/or inadequate, significant research has been performed in software-assisted calibration techniques. However much of this research has not provided the level of flexibility and practicality typically
required by real-world engineers. With this in mind, a patent-pending (US 61/859,819) architecture for assisted calibration was developed to maximize practicality, flexibility, and ease-of-use. This architecture was extended to support applications of sensitivity analysis and optimization, leading to the “SASCO” acronym.

Some calibration techniques in the literature are ideal for origin-destination flow modeling, while others employ pattern matching of vehicle trajectories and/or speed-flow relationships. A third family of calibration method employs simulation-based optimizations (SO), which have usually relied on heuristic searching methods. However the SO-based calibration methods have not gained commercial popularity for calibration of traffic simulations, most likely because their heuristic methods tend to require an excessive and unpredictable number of trial simulations. To achieve the necessary reduction in trial simulations, the SASCO architecture allows for quick and easy reduction of the search space, by defining trial values and range limits for SO-based calibration. Once the search space has been reduced through SASCO, the directed brute force (DBF) and Simultaneous Perturbation Stochastic Approximation (SPSA) methods appear to be good candidates for optimizing the input parameters.

Testing was done to assess DBF and SPSA qualities, so they could be applied in the right situations. While SPSA performed true to its reputation by optimizing rapidly, its effectiveness was dependent on internal (“a” and “c”) parameter values, randomly-generated delta (+1 and -1) values, and starting points. Moreover, trial (xplus, xminus) values seemed much more helpful than the so-called optimum (theta) values, and SPSA usually found local instead of global optimum solutions. By contrast, DBF guarantees locating the best solution within a user-defined search space, although these search spaces sometimes fail to include global optimum solutions. DBF can also optimize advanced (interdependent numeric distributions and discrete words) inputs. Finally, DBF is ideal for sensitivity analysis, and can help determine which inputs to calibrate.

Given these complementary attributes, a calibration process could conceivably be more efficient with the two methods applied in tandem. For example, SPSA might be ideal for locating a local optimum solution more quickly than DBF would. Using this local optimum solution as a starting point, DBF could then perform a more detailed and reliable scan of the surrounding area, to locate a global optimum solution.

Follow-up studies should examine more optimization methods, validation following calibration, calibration of output distributions, more real-world corridors, and more simulators. Moreover it would help to systematically determine the best SPSA internal parameter values for various network conditions. Regardless of which optimization method is selected, the SASCO architecture appears to offer a new and practice-ready level of efficiency, for software-assisted calibration.

REFERENCES


