Revisiting the Application of Simultaneous Perturbation Stochastic Approximation towards Signal Timing Optimization

David K. Hale, Ph.D.
Transportation Project Manager
Leidos, Inc.
11251 Roger Bacon Drive
Reston, VA 20190
david.k.hale@leidos.com

Constantinos Antoniou, Ph.D.
Full Professor
Technical University of Munich
Department of Civil, Geo and Environmental Engineering
Chair of Transportation Systems Engineering
c.antoniou@tum.de

Byungkyu Brian Park, Ph.D.
Associate Professor
University of Virginia
351 McCormick Rd
Charlottesville, VA 22904
bpark@virginia.edu

Jiaqi Ma, Ph.D.
Research Scientist / Project Manager
Leidos, Inc.
11251 Roger Bacon Drive
Reston, VA 20190
jiaqi.ma@leidos.com

Lei Zhang
Graduate Research Assistant
Mississippi State University
Mississippi State, MS 39762
lz218@msstate.edu

Alexander Paz, Ph.D., P.E.
Associate Professor
University of Nevada, Las Vegas
4505 Maryland Parkway, PO Box 454015
Las Vegas, NV 89154-4015
apaz@unlv.edu

Corresponding Author: David K. Hale

Revisiting the Application of Simultaneous Perturbation Stochastic Approximation towards Signal Timing Optimization

ABSTRACT

Simultaneous Perturbation Stochastic Approximation (SPSA) has gained favor as an efficient optimization method for calibrating computationally intensive, "black box" traffic flow simulations. Few recent studies have investigated the efficiency of SPSA for traffic signal timing optimization. It is important for this to be investigated, because significant room for improvement exists in the area of signal optimization. Some signal timing methods and products perform optimization very quickly, but deliver mediocre solutions. Other methods and products deliver high-quality solutions, but at a very slow rate. When using commercialized desktop signal timing products, engineers are often forced to choose between speed and solution quality. Real-time adaptive control products, which must optimize timings within seconds on a cycle-by-cycle basis, have limited time to reach a high-quality solution. The existing literature indicates that SPSA provides the potential for upgrading both off-line and on-line solutions alike, by delivering high-quality solutions within seconds. This paper describes an extensive set of optimization tests involving SPSA and genetic algorithms (GA). The final results suggest that GA was slightly more efficient than SPSA. Moreover, the results suggest today's signal timing solutions could be improved significantly by incorporating GA, SPSA, and 'playbooks' of pre-optimized starting points. However, it may take another 5-10 years before our computers become fast enough to simultaneously optimize coordination settings (i.e., cycle length, phasing sequence, and offsets) at numerous intersections, using the most powerful heuristic methods, at speeds that are compatible with real-time adaptive solutions.

INTRODUCTION

The practice of signal timing optimization has seen a number of significant advancements in recent years. Some improvements have been implemented within off-line desktop software products, and others are now present within the real-time adaptive control solutions. Examples of desktop optimization advances include (1) broader use of simulation-based optimization, (2) improved modeling of oversaturated conditions, and (3) ability to export timings directly into signal controllers. Examples of adaptive control advances include (1) web-based controller interfaces, (2) improved video detection of vehicles, and (3) conversion of signal phasing into a digitized state. Moreover, automated traffic signal performance measures are now being used to constantly monitor system performance (*Bullock et al., 2014*).

The optimization algorithms currently used by most of the popular signal timing products are extremely efficient, in terms of their computer run times. The algorithms appear to have been chosen, in part, to accommodate the speed of personal computers in the 1990s and early 2000s. These 'maximum speed' methods include hill-climbing (*Robertson, 1969*), Equisat (*Cipriani and Gori, 2000*), Webster's method (*Webster, 1958*), and the greedy algorithm (*He et al., 2011, Zhao et al., 2011*). These methods were able to optimize signal timings within seconds, even on personal computers built 15 years ago. However, computer processing speeds have increased substantially in recent years. The improved processing speed is making it possible to rely more heavily on macroscopic or microscopic simulation-based and optimizations, as opposed to simplistic analytical methods (e.g. Equisat, Webster's). Simulation-based optimization is presumably a more favorable paradigm than analytical optimization, due to simulation's ability to capture advanced effects (e.g. permissive left turns, queue spillback, actuated signals) more accurately.

In addition to allowing more widespread use of simulation-based optimization, improved computer speeds are allowing consideration of heuristic (self-adapting) optimization algorithms that are more powerful than hill-climbing or the greedy algorithm. In fact, the same computer run times once needed for 'maximum speed' methods can now accommodate efficient heuristics and meta-heuristics, including genetic algorithms (*Park, 1998*) and particle swarm optimization (*Poli et al., 2007*). Genetic algorithms (GA) in particular have been rated favorably for signal optimization (*Oda et al., 1996; Park, 1998; Agbolosu-Amison et al., 2009*; *Ghanim and Abu-Lebdeh, 2015*) and have been commercialized within multiple desktop products. For this reason, this study used GA as a benchmark. More recently, simulated annealing (*Kirkpatrick et al., 1983*) was shown to optimize signal timings approximately as well as GA (*Hale et al., 2015*), at least for isolated intersections. In addition, GA have provided excellent results in various relevant domains including traffic data mining (Basyoni and Talaat, 2015), traffic safety (Schorr et al., 2016), and traffic operations (Zhu et al., 2017)

PROBLEM STATEMENT

Despite the significant advancements in recent years, many desktop products and adaptive solutions continue to be dependent on outdated optimization methods. The result is that so-called state-of-the-art products are likely generating highly suboptimal solutions. This is creating an excessive amount of field observation, corrective re-timing, and subjective judgments, from

traffic engineers in the field. In other cases suboptimal designs are not being corrected, causing unnecessary traffic congestion.

One possible solution is the Simultaneous Perturbation Stochastic Approximation (SPSA) method of simulation-based optimization. Despite the SPSA-based traffic signal timing patent filed by Spall (*Spall, 1995*), SPSA is receiving little if any attention by signal timing practitioners and product vendors. SPSA is rated highly by researchers in the fields of traffic demand modeling (*Cipriani et al., 2011*), traffic assignment (*Ozguven and Ozbay, 2008*), and traffic model calibration (*Hale et al., 2015; Paz et al., 2015*). SPSA has been lauded by transportation researchers "for its proven performance and computational properties in large-scale problems" (*Balakrishna et al., 2007*), and for its ability to "solve very large noisy problems in a computationally attractive fashion" (*TRB Research Needs Statements, accessed in 2014*). If SPSA can truly deliver high-quality solutions within seconds, it would become a top candidate to replace outdated signal timing methods still heavily in use.

IMPLEMENTATION

Overview of SPSA

The following step-by-step summary displays how SPSA iteratively produces a sequence of estimates (*Markou and Antoniou, 2015; Spall, 2012*):

1) The process is initialized (i = 0) so that $\theta_i = \theta_0$, a K-dimensional vector of a-priori values. SPSA coefficients α, A, c, a, and γ are picked according to characteristics of the problem.
2) The number of gradient replications **gradrep** is chosen.
3) The iteration counter is increased by one unit, and step sizes a_i and c_i are calculated as:

$$a_i = a / (A + i)^{\alpha} \quad (1)$$
$$c_i = c / (i)^{\gamma} \quad (2)$$

4) A K–dimensional vector Δ_i of independent random perturbations is generated by Monte-Carlo simulation. Elements Δ_{ik},k = 1,2,...,K are drawn from a probability distribution symmetrically distributed around zero, satisfying that both $|\Delta_{ik}|$ and $E|\Delta_{ik}^{-1}|$ are bounded above by constants. A simple choice for each Δ_{ik} is the Bernoulli ±1 with equal probability.
5) The loss function is evaluated at two points, by obtaining measurements of simultaneous perturbation on 'either side' of θ_i, corresponding to $\theta_{i+} = \theta_i + c_i\Delta_i$ and $\theta_{i-} = \theta_i - c_i\Delta_i$. These points are also known as x+ and x-. Each point is confirmed to be between the lower and upper bound before function evaluation.
6) The K-dimensional gradient vector is approximated as

$$\hat{g}(\theta_i) = \frac{z(\theta_{i+}) - z(\theta_{i-})}{2c_i} \quad (3)$$

7) Steps 4 to 6 are repeated **gradrep** times using *independent* Δ_i draws. An average gradient vector for θ_i is also computed.
8) An updated solution point θ_{i+1} is obtained:

$$\theta_{i+1} = \theta_i - a_i \hat{g}_i(\theta_i)[\Delta_{i1}^{-1}, \Delta_{i2}^{-1}, \ldots, \Delta_{iK}^{-1}] \qquad (4)$$

9) Termination is declared when θ_i and function value $z(\theta_i)$ stabilize across several iterations.

Signal Timing Implementation of SPSA

Literature sources are available to demonstrate the implementation of traffic signal timing methods such as hill-climbing, simulated annealing, and genetic algorithms. But for implementing SPSA in the context of signal optimization, a brief overview is now needed. Due to the lack of literature on applying SPSA towards signal optimization, it is helpful to start with a simple case (e.g., isolated intersections) before considering more complex cases (e.g., arterials and grids).

Modern controllers typically have eight signal phases, whose green durations can be optimized. In a simulation-based optimization framework, many traffic model simulations are executed with different timing plans. The simulation producing the 'best' performance measures according to pre-defined specification(s) is set aside. Its timing plan is classified as 'optimal'. Many choices influence which timing plan is ultimately rated as optimal; such as the chosen traffic model, green time constraints, cycle length constraints, number of simulation runs, and the algorithm (e.g., SPSA, GA, etc.) for choosing which candidate timing plans will be simulated.

For signal timing at isolated intersections, typically the eight maximum green times are optimized. There is no need to explicitly optimize the cycle length, because this will be automatically determined by the maximum green times. There is no need to optimize the phasing sequence, because it usually has little effect on results (*Koonce et al., 2008*). There is no need to optimize offsets, because they are only relevant at coordinated signals. Other settings – such as minimum greens and gap settings (also known as unit extensions) – are fixed by local policies on safety and driver expectancy. One example of a common optimization objective function is the average delay per vehicle, which needs to be as small as possible.

Isolated intersections are thus helpful for a preliminary investigation of SPSA because it is only necessary to optimize eight maximum green values, one for each signal phase. At some intersections, the number of maximum greens to be optimized is even smaller. When left-turn demand volumes are light on a specific intersection approach, there is no need for an exclusive left-turn signal phase on that approach. This means that at some intersections, some of the maximum green times are irrelevant, and will not need to be optimized.

Perturbation of One Maximum Green Value

The first step is perturbing (i.e., modifying) one of the eight maximum green values. The process must always begin with an initial signal timing plan. Later in this paper, it will be shown that the initial timing plan is important to the process, and that a good choice of initial timing can significantly improve optimization outcomes. For isolated intersections, the initial timing plan consists of up to eight maximum green values. The initial value is called theta (θ). Therefore, the overall initial timing plan can be expressed as θ_1 through θ_8:

$$\{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7, \theta_8\} \qquad (5)$$

SPSA literature (*Markou and Antoniou, 2015*) states that convergence to a solution can be expected within 100 iterations, in which each iteration is comprised of two simulations. To perform two simulations in a given iteration, it is necessary to generate and evaluate two candidate timing plans. To perturb the maximum green value of Phase #1, it would be necessary to know the θ value of Phase #1, and the available solution space. For example if θ_1 were 100 seconds, and the available solution space were 180 seconds, then the perturbed values (x+ and x-) in the first iteration would default to 190 s and 10 s, respectively.

The second step of SPSA implementation involves performing the two simulations in a given iteration, obtaining the objective function values for these two simulations, and then adjusting θ based on those objective function values. For example if Simulation 1 reported 50 s of delay per vehicle (s/veh), and Simulation 2 reported 100 s/veh, then θ should move in the direction of x+, which was 190 s. The length of this jump, in the direction of x+, can be called the 'theta step size'. Based on the value of this step size, the value of θ_1 in the next iteration could be anywhere between 101 and 190 s.

The challenge with SPSA is to determine values of the available solution space, and theta step size, which will allow for efficient optimization. In addition, most SPSA literature suggests that the solution space should decrease with each iteration, making it necessary to determine a 'reduction rate' of this space. Moreover, the literature suggests that theta step size should be tied to the 'gradient', which is the difference between the two objective function values (e.g., 50 and 100 s/veh from the earlier example). The concept is that large gradients call for large jumps, whereas small gradients only deserve small jumps.

Fig. 1 illustrates an initially large solution space that is gradually reduced, thus causing x+ and x- to eventually converge on an optimum value of θ. In this figure, solution space is reduced at a constant rate, and theta step size is also constant (i.e., not tied to the gradient). Again, the challenge with SPSA is to determine internal values (e.g. solution space, theta step size, reduction rate) that will facilitate efficient optimization.

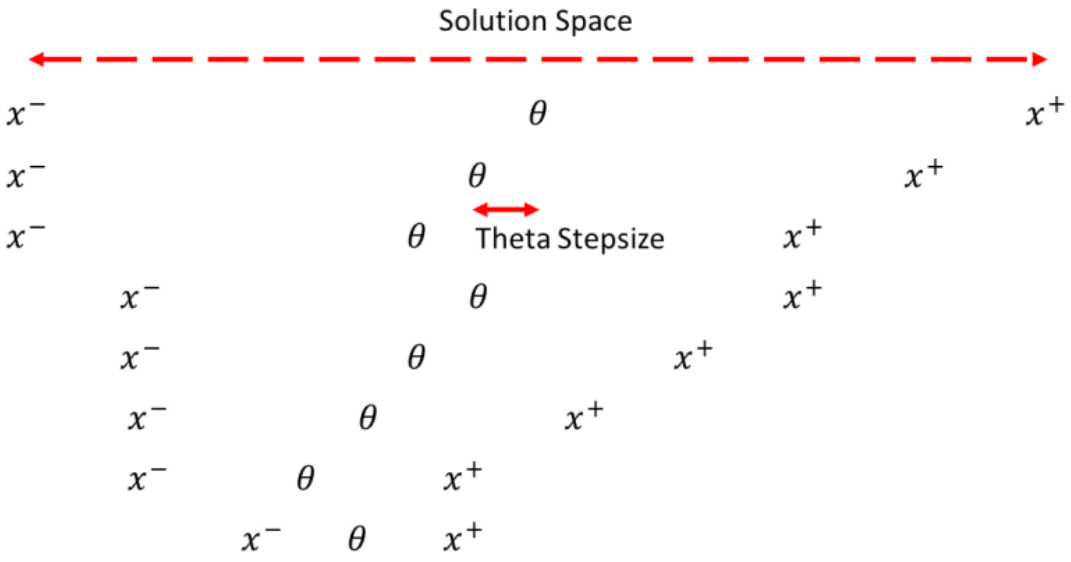

FIGURE 1: Theory behind perturbation of one maximum green value

The SPSA framework is consistent with operations research theory that calls for a balance between exploration and exploitation (*Trelea, 2003*). Exploration considers a wide variety of values throughout the solution space, whereas exploitation performs fine-tuning of a local area. When the exploration-exploitation ratio is not properly balanced, optimization algorithms tend to produce weaker final solutions. Optimization algorithms that seek this balance by combining exploration and exploitation methods are named Memetic (*Paz et al., 2015*).

Simultaneous Perturbation of All Maximum Greens

Earlier, it was stated that in order to perform two simulations in a given iteration, it is necessary to generate two candidate timing plans. Each candidate timing plan consists of eight maximum green values, and each value must be simultaneously perturbed. However, they are usually not all perturbed in the same direction. SPSA is also stochastic, because the choice of whether to simulate x+ or x- is a random one.

Fig. 2 illustrates one candidate timing plan in which θ_4 and θ_6 were randomly chosen for simulating x-, and the other thetas were chosen for simulating x+. The second candidate timing plan in this iteration would need to be the opposite; θ_4 and θ_6 would be evaluated at x+, with the rest being evaluated at x-. In subsequent iterations, the choice between x+ and x- would again be random.

This randomness is a helpful mechanism for avoiding premature convergence, and for exploring fresh alternatives. It often causes certain values of θ to step in a suboptimal direction, while most values of θ step in an optimal direction. This apparent imperfection is actually necessary for efficient overall optimization; similar to the genetic algorithm's mutation rate, which randomly forces consideration of fresh solutions. When the mutation rate is zero, GA consistently converges too quickly on local, suboptimal solutions.

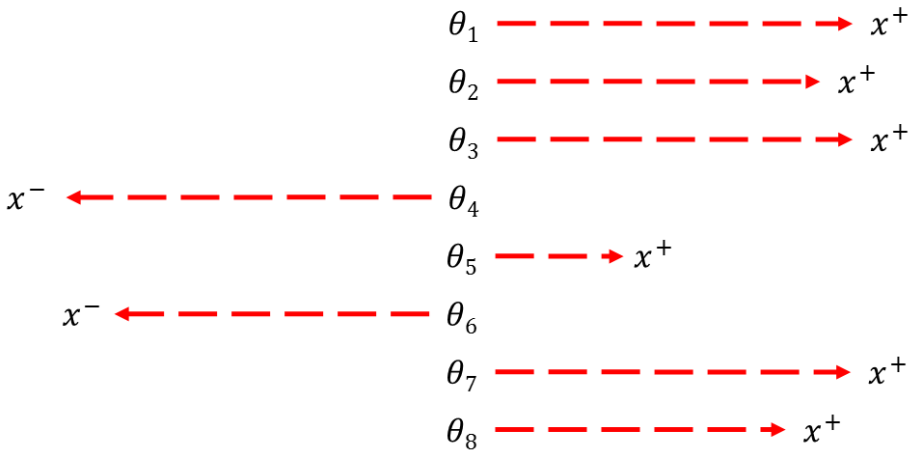

FIGURE 2: Theory behind simultaneous perturbation of all maximum greens

EXPERIMENTAL DESIGN

This study used the deterministic *Highway Capacity Manual* (HCM) analysis procedure for signalized intersections (*Transportation Research Board, 2010*) to test the effectiveness of SPSA for signal optimization. It contains sufficient complexity to be classified as a macroscopic simulation, whose outcomes cannot be predicted easily. However it is not computationally intensive by the standards of today's computers, and relative to microscopic simulations. In fact, the procedure can analyze 15 minutes of traffic flow in less than one second, on a typical personal computer. This rapid function evaluation made it possible to perform thousands of signal timing optimizations under a wide variety of conditions, to carefully determine the efficiency of SPSA.

This experiment extended a recent HCM-based optimization study (*Hale et al., 2015*) comparing efficiencies of several heuristic methods including a genetic algorithm (GA), simulated annealing, Tabu search, and traditional hill-climbing. The study tested 72 intersection scenarios representing the widest possible variety of topographies and signal phasing, as documented in Hale et al. (*2015*). It used delay as the main performance indicator, because the HCM stipulates that delay should be used to determine signalized intersection level of service. Each scenario modeled 15 minutes of traffic as recommended by the HCM (*Transportation Research Board, 2010*). These 72 original datasets can now be re-used, to evaluate the relative efficiency of SPSA. Fig. 3 summarizes the results of 1152 optimization runs, from the prior research. The figure shows that simulated annealing (SA) and GA with tournament (T) or roulette wheel (R) selection were capable of 32-33% improvements. Hill-climbing (HC) only achieved 22-23% improvement.

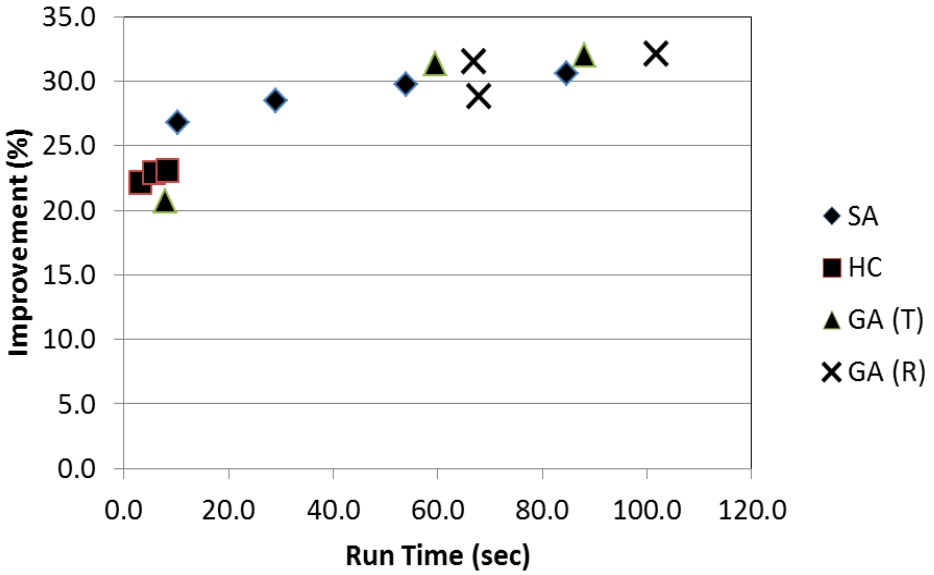

FIGURE 3: Improvement percentage versus computer run time (*Hale et al., 2015*)

Fig. 4 illustrates the objective of this study, which was to reach the 'new target' of high-quality solutions within seconds. Prior literature and data collection have shown that simple methods could complete optimization within seconds, delivering mediocre solutions; and that powerful heuristics could complete within minutes, reaching global optima. If SPSA could reach global optima within seconds, it would become a prime candidate for replacing current methods

used in most commercial products. To determine whether SPSA could reach the new target, SPSA was thoroughly tested in a new round of data collection along with GA, because GA has earned a strong reputation by researchers of signal timing. In addition, GA recently outperformed SPSA in a study on car-following model calibration (*Li et al., 2016*). To perform these tests, it was necessary to examine a wide variety of internal parameters and starting points.

In both SPSA and GA, one internal parameter that must be tested thoroughly is the number of iterations. In the GA literature, this setting is commonly called the number of generations, and its effect is similar to the number of SPSA iterations. Number of iterations is directly tied to the run time on the x-axis of Figs. 3 and 4. It is not a mystery that increasing the number of iterations should produce better optimization outcomes, provided that excessive run times and diminishing returns are not a concern. So although it was necessary to test various numbers of iterations, to examine SPSA's efficiency through the lens of optimality versus run time (i.e., as in Figs. 3 and 4), it was more important to learn the impact of other internal parameters.

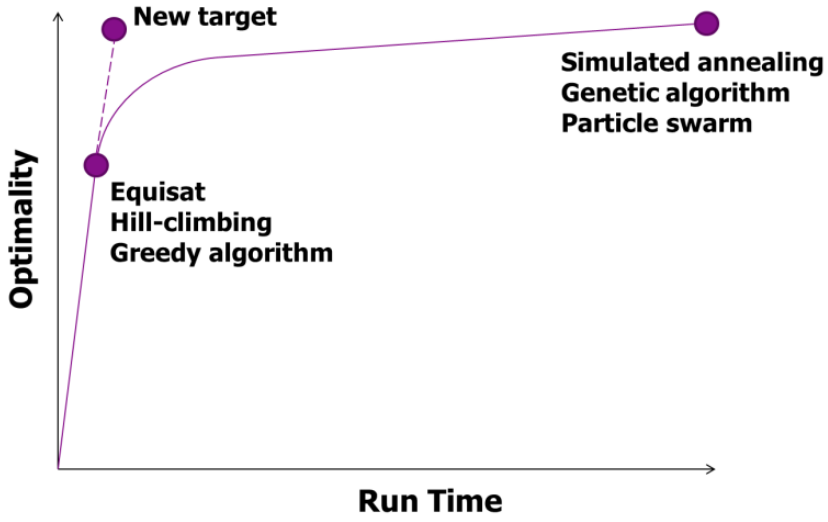

FIGURE 4: Theory of optimality versus run time

SPSA Internal Parameters

A sensitivity analysis was conducted on four SPSA parameters: initial solution space, theta step size, solution space reduction rate, and minimum solution space. Spall (*1998*) recommends customizing step sizes and reduction rates by using parameters *a* and *c*. In addition to testing various *a* and *c* parameter values, extensive testing was performed with fixed step sizes, fixed reduction rates, and random step sizes. For such tests, it was necessary to develop SPSA in a customized fashion, so that it would implement equations (1) through (4) only when desired.

As discussed earlier, optimization processes are said to have a mixture of solution space exploration and exploitation. Large solution spaces and theta step sizes, combined with small reduction rates for both, would prioritize exploration. Large reduction rates would quickly prioritize exploitation after a brief initial exploration. Medium solution spaces and theta step sizes, combined with medium reduction rates for both, would balance exploration and exploitation. Small solution spaces and theta step sizes prioritize exploitation, regardless of reduction rate.

When testing SPSA according to its implementation in the literature, *a* and *c* values between 0.1 and 4.0 were tested. Most test optimizations were performed with 25 combinations of the following 5 typical values of *a* and *c*: {0.5, 1.0, 2.0, 3.0, 4.0}. When testing SPSA according to customized implementations in this study, various initial and minimum solution spaces between 2 s and 190 s were tested. Many theta step sizes between 1 s and 190 s were tested. A variety of large and small reduction rates were tested; including zero reduction rates, which produced fixed solution spaces and theta step sizes. Finally, certain custom implementations involved random theta step size values. Some random step size values were applied equally to all eight signal phases. In other cases, unique random values were applied separately to each phase.

The goal of these sensitivity analyses was to identify the best internal settings, causing optimum ratios of exploration-to-exploitation, to produce the best signal timings for isolated intersections. In the event that different internal settings would work best for various intersections, it was hoped that certain patterns might emerge. If so, then the best internal settings might easily be predictable based on elements of lane geometry, signal phasing, or volume demand.

GA Internal Parameters

Similar to SPSA, the genetic algorithm has internal parameters affecting its efficiency. These parameters include mutation rate, crossover rate, population size, elitist method, selection method, and convergence threshold. However unlike SPSA, a wealth of GA-based signal timing research has tested various parameter values. For isolated signalized intersections, at least two sensitivity analysis efforts (*Park, 1998; Hale et al., 2015*) yielded a very specific set of values:

- Mutation rate: 4%
- Crossover rate: 30%
- Population size: 10
- Elitist method: True
- Selection method : Tournament
- Convergence threshold: 0.01%

For example, small (non-zero) mutation rates have been shown to work well in preserving solution improvement, while simultaneously preventing convergence on local optima. Similarly, ‘medium’ crossover rates worked well by ensuring that a significant portion of signal timings are obtained from both ‘parents’. In the optimization tests conducted for this study, mutation rate was the only internal GA parameter – apart from the ubiquitous number of generations – that was varied extensively. This was because in the authors’ experience with isolated intersections, mutation rate had the most effective and predictable impact on the exploration-exploitation ratio.

Starting Points

For most simulation-based optimization heuristics, the starting point often has a significant impact on final outcomes (*Cipriani et al., 2011; Lo and Chow, 2004*). In the prior study (*Hale et al., 2015*) of signal timing at isolated intersections, signal timings and traffic demands for 72 test cases were pre-adjusted, so that overall intersection delays would always fall between 80 and 85 s/veh. This was a logical starting point for optimization because (1)

intersections with delays exceeding 80 s/veh are considered to operate at a failing level of service (*Transportation Research Board, 2010*), (2) congestion levels much higher or lower than this could reduce the amount of optimization improvement that can take place, and (3) severely oversaturated conditions are only analyzed accurately by time-consuming, multi-period models.

The signal timings producing 80-85 s/veh could be described as 'bad' starting points, because the under-saturated intersections were capable of operating at much smaller delays than this. In the new study, whose results are shown in the upcoming Experimental Results section, a baseline test of the 72 datasets with bad starting points produced an average starting delay of 82.2 s/veh. The average GA-optimized delay was 55.6 s/veh after 250 generations, representing an average improvement of 32.4%. The new study also examined a wide variety of starting points, to see how they might affect the efficiency of SPSA and GA.

a) Bad Starting Points (0% improvement). The first set of optimization tests was performed using the bad timing plans as a starting point, as described above.

b) Good Starting Points (17% improvement). Another set of optimization tests was performed with 'good' timing plans as a starting point. Signal timings of the 72 datasets were modified until their delays fell between 65 and 70 s/veh. The average delay was 68.4 s/veh; representing an average improvement of 16.8%, compared to the bad starting points.

c) Equisat Starting Points (7% improvement). A third set of tests was performed with Equisat (*Cipriani and Gori, 2000*) timing plans as a starting point. For some datasets Equisat produced excellent starting points, reducing delays by 50-75%. However in other cases, Equisat timings actually increased overall intersection delay to greater than 100 s/veh. This discrepancy was apparently due to Equisat's inability to handle permissive left-turn operations, and highly-unequal lane utilizations. The average delay was 76.8 s/veh; representing an average improvement of 6.5%, compared to the bad starting points.

d) Pre-Optimized Starting Points (32% improvement). A fourth set of tests was performed under timing plans pre-optimized by GA. The average delay was 56.2 s/veh; representing an average improvement of 31.6%, compared to the bad starting points. This average was slightly different than the prior baseline test (32.4% improvement), most likely due to running fewer generations of optimization.

EXPERIMENTAL RESULTS

Standard SPSA Optimizations

The initial set of optimization tests involved the recommended implementation of SPSA found in the literature. Most test optimizations were performed with 25 combinations of the following 5 typical values of a and c: {0.5, 1.0, 2.0, 3.0, 4.0}. A given combination of a and c was consistently applied within consecutive optimizations of the 72 core datasets. This process was repeated for 25 combinations of a and c. As shown in Table 1, improvement percentages varied between 18.3% and 23.3% when running 100 iterations of SPSA. Increasing to 250 iterations did not significantly improve the outcomes. These initial results were disappointing, considering the

baseline 32.4% improvement from GA. Table 2 shows that the optimum combination of values varied significantly among the 72 core datasets. This implies that the best *a* and *c* values might be difficult to predict when optimizing timings at new, real-world intersections.

At this stage, it was hypothesized that if the optimum values of *a* and *c* could be known for each intersection in advance, then SPSA might consistently outperform GA. Although this hypothesis held true for some core datasets, it clearly failed for others. Further, it was hypothesized that the optimum values of *a* and *c* could be predicted as a function of signal phasing, or traffic congestion levels. Unfortunately, a series of test runs found that the optimum values of *a* and *c* were seemingly random. Moreover, adjusting congestion levels at a given intersection would change the optimum combination of *a* and *c*.

The standard SPSA optimization experiments were all performed with bad starting points (i.e., intersection delays between 80 and 85 s/veh). These experimental results implied that SPSA might simply be a 'lucky shot' algorithm, similar to traditional hill-climbing, and that SPSA efficiency might be dependent on lucky selections of *a* and *c*. However, further investigation was performed to determine whether SPSA's exploration-exploitation balance could be improved.

TABLE 1: Average improvement under standard SPSA implementation

	a (0.05)	**a (0.10)**	**a (0.20)**	**a (0.30)**	**a (0.40)**
c (0.05)	22.2%	21.8%	22.9%	22.8%	21.5%
c (0.10)	**23.3%**	23.1%	22.3%	22.8%	22.1%
c (0.20)	22.4%	21.9%	22.7%	22.1%	19.5%
c (0.30)	21.1%	21.6%	20.4%	20.5%	20.1%
c (0.40)	18.3%	19.9%	20.8%	19.9%	20.6%

TABLE 2: Best *a* and *c* combinations for 72 core datasets

	a (0.05)	**a (0.10)**	**a (0.20)**	**a (0.30)**	**a (0.40)**
c (0.05)	4	**8**	4	2	**8**
c (0.10)	6	3	2	2	3
c (0.20)	1	3	4	2	0
c (0.30)	3	5	3	1	0
c (0.40)	1	3	3	0	1

Customized SPSA Optimizations

After observing the initial set of disappointing results, the SPSA calculations were studied in a step-by-step manner. It was believed that the exploration-exploitation balance was too unequal; and that by customizing the SPSA algorithm, a more equal balance would produce better optimization outcomes. The initial step-by-step analysis revealed excessively small theta step sizes, often just a few tenths of a second.

These miniscule step sizes were being caused by the gradient, as shown previously in equations (3) and (4). For example, SPSA performs two function evaluations per iteration. If the first evaluation returns an intersection delay of 81 s/veh, and the second evaluation returns 83 s/veh, the gradient difference of 2.0 s would cause maximum green times to step by 1.0 s in the direction of the timing plan that produced 81 s/veh. Subsequent iteration step sizes would decrease below 1.0 s, due to reduction rates caused by equations (1) and (2). This reflected an imbalance with too much exploitation, and too little solution space exploration.

The first SPSA customization was to normalize theta step sizes against solution space *S*, shown in equations (6) and (7), so that large gradients could produce step sizes spanning the full solution space. This customization helped SPSA to produce much better outcomes than before, due to the increased exploration. However the overall outcomes were still well short of GA, because now the amount of exploitation was too low. For example, if the first evaluation returned an intersection delay of 176 s/veh, and the second evaluation returned 93 s/veh, the gradient difference of 83 s could cause maximum green times to step by more than 200 s. This was causing oscillations between extremely high and low maximum greens. Although some subsequent iterations yielded more reasonable step sizes, there was still too much instability.

$$\hat{g}_i(\theta_i) = \frac{z(\theta_i + c_i\Delta_i S) - z(\theta_i - c_i\Delta_i S)}{2c_k} \qquad (6)$$

$$\theta_{i+1} = \theta_i - a_i\hat{g}_i(\theta_i)S \qquad (7)$$

The third SPSA customization was to stabilize theta step size, by uncoupling it from the gradient. The fourth customization was to specify a constant reduction rate *R* for solution space *S*. Fortunately these two customizations, shown in equations (8) and (9), allowed SPSA to produce final outcomes that were very similar to GA. More promising was the fact that SPSA outcomes were no longer dependent on *a* and *c*, and produced GA-like solutions across all 72 core datasets. In fact when using the bad starting points, final outcomes of SPSA and GA differed by less than 1%. Still, the original study objective was for SPSA to consistently outperform GA. Because of this, a new set of experiments was performed that had a much wider variety of starting points. The hope was that SPSA could reach global optima much more quickly than GA, potentially making it a prime candidate for real-time adaptive solutions.

$$S_{i+1} = S - ((i-1) \times R) \qquad (8)$$

$$\theta_{i+1} = \theta_i - S_{i+1} \qquad (9)$$

The third and fourth customizations of SPSA resulted in four internal parameters that could be adjusted: initial solution space, theta step size, solution space reduction rate, and

minimum solution space. Therefore, in the new set of experiments with a wide variety of starting points, the four internal parameters were also varied widely.

Regarding initial solution space, it was found that 190 s worked quite well; given that reduction rates would gradually reduce this space, and transition the process from exploration to exploitation. Smaller solution spaces below 100 s did not work well; causing SPSA outcomes to become more than 1% inferior to GA, likely due to insufficient exploration.

Regarding theta step size, several methods and constant values worked well, while several others did not. Constant values between 2 s and 3 s worked well when coupled with wide initial solution spaces, and 2-s solution space reduction rates. Randomly-generated values between 1 s and 70 s worked well, but only when solution space was continuously set equal to ($2 * \theta$). Constant theta step sizes outside the range of 2-3 s did not work well. Random theta step sizes with maximums far away from 70 s did not work well. These ranges imply that only medium step sizes could achieve the right balance between exploration and exploitation.

Regarding solution space reduction rate, the best rate was dependent on the initial solution space. Under an initial solution space of 190 s, a reduction rate of 2 s per iteration worked quite well. For example, the solution space would reduce from 190 s to 10 s after 90 SPSA iterations, allowing a good amount of both exploration and exploitation. In fact, a minimum solution space of 10 s was found to work well. In the Final Results section, the best SPSA results will be compared against the best GA results, at numerous starting points.

GA Optimizations

In addition to the customized SPSA optimization runs, numerous GA optimizations were performed at numerous starting points, using various values of the mutation rate. When using the bad starting points, mutation rates of 8% and 10% produced the best improvement percentages (32%). All other mutation rates between 3% and 20% produced 31% improvement. Mutation rates outside the range of 3% to 20% produced improvement percentages below 31%. Thus the medium rates (8% and 10%) produced the best balance of exploration and exploitation.

When using the good starting points, most mutation rates between 3% and 20% produced 32% improvement. When using the pre-optimized starting points, all mutation rates between 3% and 15% produced 33% improvement. The narrowing of effective mutation rates implies that pre-optimized starting points reduced the amount of exploration that was necessary.

Final Results

Fig. 5 illustrates the final comparison between SPSA and GA, across three sets of starting points. Although earlier tests showed both methods achieving improvement rates of 32-33% after 120 s of run time, Fig. 5 illustrates divergent behaviors during the first 60 s of run time. Under bad starting points, SPSA reached the 29% level more quickly, but GA reached the 32-33% level more quickly. Under good starting points, GA reached 32% in less than 20 s, while SPSA needed more than 60 s to reach the same level.

Finally, under pre-optimized starting points, GA reached 33% after 15 s, while SPSA reached 33% after 38 s. Except for the Equisat starting point results, these summary results represented the best performances of SPSA and GA. Inferior results, generated by inferior internal parameter values, were not included in Fig. 5. When using Equisat starting points, both methods reached 33% improvement after 120 s of run time. However their performance during the first 60 s was not recorded.

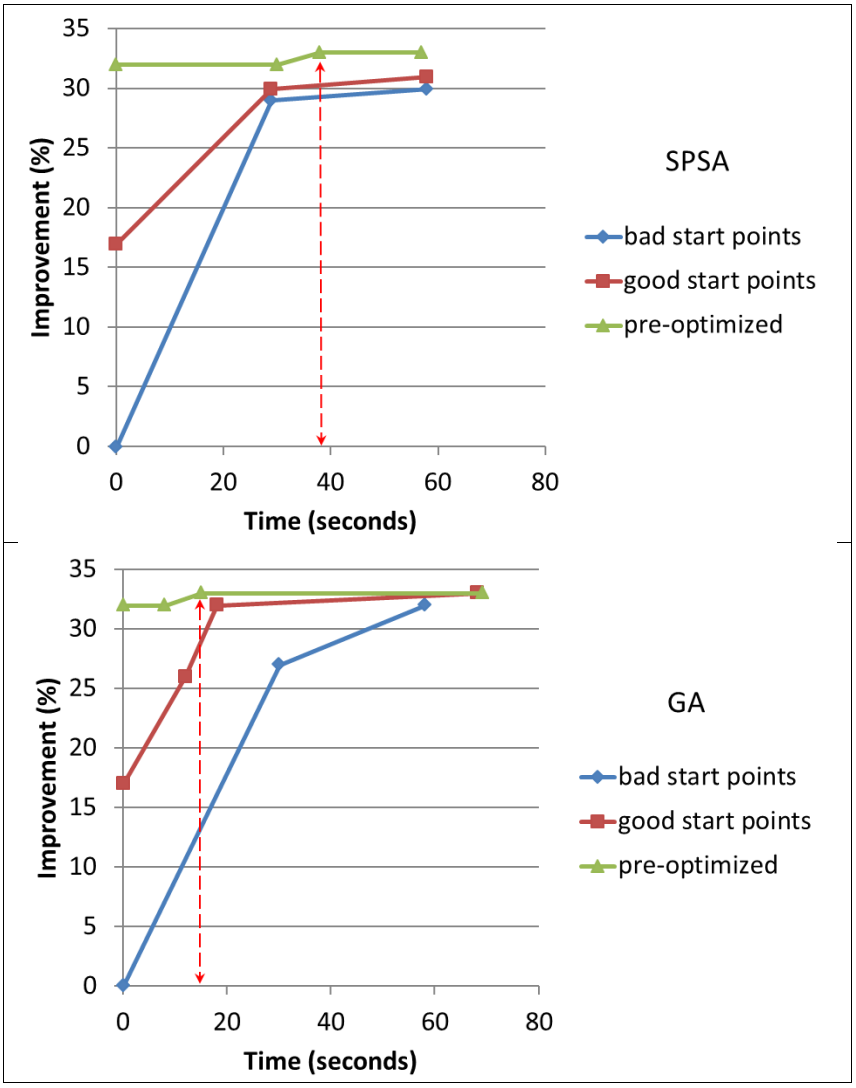

FIGURE 5: Comparison of SPSA (top) and GA (bottom) improvement rates

CONCLUSIONS

For traffic signal timing optimization, improved computer speeds enable the use of algorithms that provide superior solutions compared to legacy methods. Too many desktop products and adaptive solutions are dependent on outdated methods such as hill-climbing, Equisat, the greedy algorithm, and Webster's method; leading to unnecessary traffic congestion, and excessive fine-tuning efforts in the field. The SPSA method of simulation-based optimization

was viewed as a possible solution to this problem, after being highly rated by researchers of traffic assignment and assisted calibration. If SPSA could truly deliver high-quality solutions within seconds, it would become a top candidate to replace outdated signal timing methods developed long ago, but still heavily used in practice.

SPSA was tested extensively against a genetic algorithm for its ability to optimize timings at isolated intersections. These experiments involved 72 test intersections containing the widest possible variety of known conditions. Experimental results revealed that SPSA and GA were both capable of reaching global optimum solutions when applied with proper internal parameter values. However SPSA required custom modifications to achieve these solutions, and GA reached global optimum levels more quickly than SPSA. This was unexpected based on SPSA commentary in the literature, and on SPSA requiring only two function evaluations per iteration.

The practical implications of these results are deceptively promising. GA is considered a slow optimization method. This is likely because long after other methods have quit, GA continues to find minor improvements. Because of this perceived slowness, GA is considered less attractive for desktop products, and not viable for real-time adaptive solutions. However this study demonstrated that on today's typical laptop computers, GA consistently reached global optima within 60 s. Moreover, when seeded with pre-optimized starting points, GA consistently reached global optima within 15 s. At these speeds, GA now appears quite practical to use for both desktop products and adaptive solutions.

To ensure the fastest (e.g., 15 s) run times in adaptive signals, it should theoretically be possible to employ a 'playbook' of pre-optimized starting points. For example, if an hour could be devoted to off-line optimization of PM peak, AM peak, and off-peak directional weightings at low, medium, and high congestion levels, the resulting nine optimized starting plans could be stored in a controller's memory. Then from that point forwards, upon detecting any set of traffic conditions, an adaptive signal could load the corresponding plan as a starting point, and nominally perform 15 s of HCM-based GA optimization. According to the study results here, this would achieve an average of 33% improvement for the 72 core test cases. According to the prior study results (*Hale et al., 2015*), the outdated heuristics in today's products would likely achieve an average of 22-23% improvement in these same cases.

Traffic engineers are frustrated by the fact that popular adaptive algorithms are secret, unpublished, and proprietary. Simulation-based optimizations based on transparent methods – such as HCM, GA, and SPSA – could end this uncertainty, and perhaps achieve superior outcomes. Even in desktop products without playbooks of starting points, heuristics such as SPSA, GA, simulated annealing, and particle swarm optimization are becoming more and more appropriate for signal timing optimization, with each passing year. These methods can now generate demonstrably superior results within minutes. Accordingly, the desktop signal timing software vendors should consider updating their algorithms.

Although SPSA did not achieve the performance that the authors expected, this research contributes to the body of knowledge by finding that 1) GA had slightly better performance than SPSA for signal timing at isolated intersections, 2) SPSA performance followed the improvement versus runtime curve observed in prior studies, 3) computers are now fast enough to support

implementation of improved heuristic methods within adaptive signals, and 4) playbooks of optimized starting points should be used in real-world product solutions. Future research should include micro-simulation analysis, field-deployed adaptive controller testing, and coordinated grid system implementation of these powerful and transparent optimization methods. In addition, there are other alike optimization methods that merit investigation for the research problem addressed in this paper. For example, Cobos et al. (2016a, 2016b) illustrated the advantages of the MA-SW-Chains and NSGA-II algorithms for the calibration of microscopic traffic flow simulation models relative to GA and SPSA implementations. These studies include analysis or running time as well as the time that was required to fine-tune the corresponding optimization parameters. Memetic and multi-objective evolutionary algorithms (*Deb et al., 2002*) could also be considered.

Some of the top-selling adaptive control solutions only perform optimization of green durations, without explicitly considering coordination settings such as cycle lengths, phasing sequences, and offsets. Thus the findings of this research could have immediate benefits for such products. Mathematically, there is nothing to prevent the implementation from being scaled up, to optimize the coordination settings for signalized arterials and grids. However it may be another 5-10 years before our computers become fast enough to simultaneously optimize coordination settings at numerous intersections, using the most powerful heuristic methods, at speeds that are compatible with real-time adaptive solutions.

REFERENCES

S. Agbolosu-Amison, B. Park, and I. Yun (2009, October). *Comparative evaluation of heuristic optimization methods in urban arterial network optimization*. Paper presented at the IEEE ITSC, St. Louis, MO.

R. Balakrishna, C. Antoniou, M. Ben-Akiva, H. Koutsopoulos, and Y. Wen (2007). Calibration of microscopic traffic simulation models: methods and application, *Transportation Research Record: Journal of the Transportation Research Board* 1999, 198-207.

Y. Basyoni, and H. Talaat (2015). A Bilevel Traffic Data Extraction Procedure via Cellular Phone Network for Intercity Travel, Journal of Intelligent Transportation Systems. 19(3), 289-303.

D. Bullock et al. (2014). Automated traffic signal performance measures, *ITE Journal* March 2014, 33-39.

C. Cobos, C. Daza, C. Martínez, M. Mendoza, C. Gaviria, C. Arteaga, and A. Paz (2016). Calibration of CORSIM Vehicular Traffic Flow Models using a Memetic Algorithm with Solis and Wets Local Search Chaining. Advances in Artificial Intelligence: Lecture Notes in Computer Science. 10022, 365-375.

C. Cobos, C. Erazo, J. Luna, M. Mendoza, C. Gaviria, C. Arteaga, and A. Paz (2016). Multi-Objective Memetic Algorithm based on NSGA-II and Simulated Annealing for Calibrating CORSIM Micro-Simulation Models of Vehicular Traffic Flow. Advances in Artificial Intelligence: Lecture Notes in Computer Science. 9868, 468-476.

E. Cipriani, M. Florian, M. Mahut, and M. Nigro (2011). A gradient approximation approach for adjusting temporal origin-destination matrices, *Transportation Research Part C: Emerging Technologies* 19(2), 270-282.

E. Cipriani, and S. Gori (2000). Signal control and assignment joint models, *Traffic and Transportation Studies*, 614–621.

D. Kalyanmoy et al. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation* 6()2, 182-197.

D.K. Hale, C. Antoniou, M. Brackstone, D. Michalaka, A. Moreno, and K. Parikh (2015). Optimization-based assisted calibration of traffic simulation models, *Transportation Research Part C: Emerging Technologies*, 100-115.

D.K. Hale, B. Park, A. Stevanovic, P. Su, and J. Ma (2015). Optimality versus run time for isolated signalized intersections, *Transportation Research Part C: Emerging Technologies*, 191-202.

Q. He, K.L. Head, and J. Ding (2011). Heuristic algorithm for priority traffic signal control, *Transportation Research Record: Journal of the Transportation Research Board* 2259, 1-7.

Highway Capacity Manual 2010, Transportation Research Board, 2010.

S. Kirkpatrick, G.C. Gelatt, and Vecchi (1983). M. P. Optimization by Simulated Annealing, *Science*, 220(4598).

P. Koonce et al. (2008). *Traffic signal timing manual*. No. FHWA-HOP-08-024.

L. Li, X. Chen, and L. Zhang (2016). A global optimization algorithm for trajectory data based car-following model calibration. *Transportation Research Part C*, 311-332.

H. K. Lo, and A. H. Chow (2004), *Control strategies for oversaturated traffic*, ASCE Journal of Transportation Engineering, 130, (4), 466-478.

M. S. Ghanim and G. Abu-Lebdeh (2015), Real-Time Dynamic Transit Signal Priority Optimization for Coordinated Traffic Networks Using Genetic Algorithms and Artificial Neural Networks. Journal of Intelligent Transportation Systems, 19(4), 327-338.

Markou, and C. Antoniou (2015, June). *Developing insight into effective SPSA parameters through sensitivity analysis*. Paper presented at the 4th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS 2015), Budapest Hungary.

T. Oda, T. Otokita, T. Tsugui, M. Kohno, and Y. Mashiyama (1996). *Optimization of signal control parameters using a genetic algorithm*. Paper presented at the 3rd World Congress on ITS, Orlando, Florida.

E.E. Ozguven, and K. Ozbay (2008). Simultaneous perturbation stochastic approximation algorithm for solving stochastic problems of transportation network analysis: performance evaluation, *Transportation Research Record: Journal of the Transportation Research Board* 2085, 12-20.

B. Park (1998). Development of genetic algorithm-based signal optimization program for oversaturated intersections, Ph.D. Dissertation, Texas A&M University.

A, Paz, V, Molano, and M. Sanchez (2015). Holistic Calibration of Microscopic Traffic Flow Models: Methodology and Real World Application Studies. Engineering and Applied Sciences Optimization: Dedicated to the memory of Professor M.G. Karlaftis. Volume 38, Edition 1. Springer International Publishing. ISBN 9783319183206.

A, Paz, V,Molano, E. Martinez, C., Gaviria, and C. Arteaga (2015). Calibration of Traffic Flow Models Using a Memetic Algorithm. *Transportation Research Part C: Emerging Technologies*. 55, 432–443.

R. Poli, J. Kennedy, and T. Blackwell (2007). Particle swarm optimization, *Swarm Intelligence*, 33-57.

D. Robertson (1969). TRANSYT: a traffic network study tool, *Road Research Laboratory Report* LR 253.

J. Schorr, a S. H. Hamdar, and C. Silverstein (2016). Measuring the safety impact of road infrastructure systems on driver behavior: Vehicle instrumentation and real world driving experiment, Journal of Intelligent Transportation Systems, 0(0), 1-11.

J.C. Spall (1998). Implementation of the simultaneous perturbation algorithm for stochastic optimization, *IEEE Transactions on Aerospace and Electronic Systems*. 34(3), 817-823.

J.C. Spall, Method and apparatus for model-free optimal signal timing for system-wide traffic control, US Patent 5668717 A.

J. C. Spall (2012). Stochastic optimization, *Handbook of computational statistics*, 173-201.

TRB Research Needs Statements accessed on February 28th, 2014: http://rns.trb.org/dproject.asp?n=15590

I.C. Trelea (2003). The particle swarm optimization algorithm: convergence analysis and parameter selection, *Information Processing Letters* 85(6), 317-325.

F.V. Webster (1958). Traffic signal settings, *Road Research Technical Report* 39.

L. Zhao, X. Peng, L. Li, and Z. Li (2011), *A fast signal timing algorithm for individual oversaturated intersection*, IEEE Transactions on Intelligent Transportation Systems, 12(1), 280-283.

N. Zhu, S. Ma, and L. Zheng (2017). Travel time estimation oriented freeway sensor placement problem considering sensor failure. Journal of Intelligent Transportation Systems, 21(1), 26-40.

ACKNOWLEDGMENT

This study was supported by the Saxton Traffic Operations Laboratory, at the Turner-Fairbank Highway Research Center in McLean Virginia. Many thanks to our Technical Writer at UNLV's Howard R. Hughes College of Engineering, Mrs. Julie Longo, for her help reviewing this manuscript.

AUTHOR INFORMATION

David K. Hale is a Transportation Project Manager with Leidos Inc., at the Turner-Fairbank Highway Research Center. His research interests include traffic operations modeling, simulation, optimization, and calibration.

Constantinos Antoniou is a Full Professor of Transportation Systems Engineering at the Department of Civil, Geo and Environmental Engineering in the Technical University of Munich (TUM), Germany. His research focuses on modelling and simulation of transportation systems, Intelligent Transport Systems (ITS), calibration and optimization algorithms and applications, road safety and sustainable transport systems.

Byungkyu Brian Park is an Associate Professor with the Department of Civil and Environmental Engineering, University of Virginia, Charlottesville, VA.

Jiaqi Ma is a Research Scientist / Project Manager with Leidos Inc., at the Turner-Fairbank Highway Research Center. His research interests include dynamic traffic management, automated/connected vehicles, human factors in transportation engineering, highway safety analysis, and intelligent transportation systems.

Lei Zhang is a graduate research assistant and Ph.D. candidate in Civil Engineering at Mississippi State University.

Alexander Paz is an Associate Professor of Civil Engineering at the University of Nevada Las Vegas (UNLV). His research interests include transportation systems analysis, traffic operations and control, demand modeling and forecasting, intelligent transportation systems, human behavior and learning, traffic flow theory, infrastructure management, and statistical econometric methods.