Continuous Selection of Optimized Traffic Light Schedules: A Machine Learning Approach

Shumeet Baluja
Google Research
shumeet@google.com

***Abstract*—Machine learning-based optimization of traffic light programs has been successfully employed to reduce emissions and traffic delays. Due to the variability of traffic flows, it is common practice to optimize multiple traffic light programs tailored for specific conditions and deploy them at predetermined times of the day or days of the week. We explore an alternative to this manual set-interval methodology. We create a system to automatically select the appropriate light controller program in response to continuously changing conditions. We analyze the current traffic density and close-time traffic patterns and instantiate the correct pre-optimized light program based on current conditions. Rather than creating a small set of programs tailored for specific periods of the day, we automatically create, and select from, an over-complete set of light controllers. Based on historic observations, a combination of machine learning approaches are used to find the best representative set of traffic flows to model the system. From these, multiple traffic-light controllers are created to address each flow individually. Using the automated matching system, we achieved reductions in both emissions and travel time over previously optimized lights. We examine the robustness of the system by ensuring that the system operates under large amounts of variability in traffic.**

***Index Terms*—Traffic Signal Timing, Stochastic Optimization, Clustering, Case-Based Reasoning**

1. Background and Motivation

Inefficient configuration of traffic lights remains a common problem in many urban areas – one of the largest complaints of commuters in the Mountain View, California area is the amount of traffic they face during the morning and evening rush hours. The principal goal of this project was to address one of the problem areas shown in Figure 1. In our previous studies [1] [2], we focused on improving the timing of seven lights.We used a variant of genetic algorithms with a novel controller paradigm, based on the concept of signal-micro-auctions, to demonstrate the potential for significant reduction in wait times over the deployed lights schedules. This same approach was recently extended to a system of lights in the River North area of Chicago, Illinois [3].

In both the Chicago and Mountain View experiments, as well as in many approaches found in the research literature,

Figure 1: Area to be optimized in Mountain View, California. Rush hour traffic flow (Google Maps with Traffic.)

it is assumed that the controller designer can *a priori* determine and specify the number of unique optimized light-controller programs required. Commonly, a separate program is created for different periods of the day – each of which has unique traffic flow characteristics (*e.g.* rush hours, middle of day, night, weekends, etc.). For each period, traffic flows are recorded and the light controllers optimized, potentially independently, to handle the traffic flows most associated with that time. Although there will naturally be variation observed over the multiple days of recording traffic, it is expected that the variation seen over multiple recordings of the same period will be smaller than the variation recorded across time periods. Although theoretically possible to create a single controller to handle the traffic over the entire period, this often results in a larger and far more complex controller. Further, some of the global, "full-day", controllers also receive as an input signal the time-of-day, thereby *implicitly* creating different control schemes for different times.

One of the observed weaknesses in the standard approach was the tacit assumption that we knew the best divisions of times for which to model traffic. We noticed that even among the seven lights, changing the divisions of traffic times we chose to model resulted in very different controllers. To address this pragmatic problem, in this pa-

per, we explore a novel alternative to the pre-determined set-interval methodology. Instead, we create a system to automatically adapt a set of traffic lights to continuously changing conditions. Succinctly, the goal is to examine the current traffic density and flow patterns, and then, based on the results of that examination, to select the appropriate pre-optimized light program. Unlike previous techniques which divided the traffic patterns into a set number of categories based on the time-of-day, we create an over-complete set of traffic patterns to match from. We select a pre-optimized program from the entire-set dynamically. This allows us to automatically handle special conditions such as large-scale events, accidents, and weather-related traffic changes. A combination of machine learning approaches are used to create a diverse set of traffic-light programs that can be instantiated when traffic flow patterns are recognized. We term this system "Match-Based Program Selection" [1].

Recently, [9] used a clustering approach to automatically find related time periods at both small and large intervals, from time-of-day to seasons. The results of their work could be used in conjunction with our results to help narrow the set of timing schedules that are considered, based on similar periods. A large body of related previous work includes the SCAT and SCOOT systems [10]–[12]. Both of these systems adapt and modify schedules in response to the current traffic flows. Though similar in intent to our system, our approach differs in key aspects. In our approach, we do not attempt to modify the current schedule to adapt to the current traffic flow; instead, we select from pre-optimized schedules that are tailored to work well in similar traffic flows. Once set during training, the light programs *are not* changed dynamically. Second, we rely on a large, over-complete set of light programs to find one that matches well to the current traffic flow. By following this procedure, we hope to simultaneously address both needs of (1) selecting from a set of appropriate programs for specific days/times/seasons as well as (2) adapting those programs to changing and/or extraordinary conditions within those times.

Next, we will describe the data used for these experiments. For this exploration, we created synthetic data, modeled after the type of data that we gathered for the Mountain View and Chicago studies. In Section 3, we briefly review how, with a set of observed traffic flows, the light timings are optimized using a stochastic search procedure; this establishes a real baseline of performance. Section 4 extends the procedure to optimize the lights for multiple traffic conditions. Once the traffic lights have been optimized for a large number of scenarios, a combination of machine learning tools, such as clustering, feature reduction and nearest-neighbor look-ups are used as the basis for the Match-Based Program Selection. Details of how the final system could be deployed and the experimental findings are given in Section 5. We have observed significant reduction in expected emissions and delays, while being agnostic to the number of underlying distinct patterns in traffic.

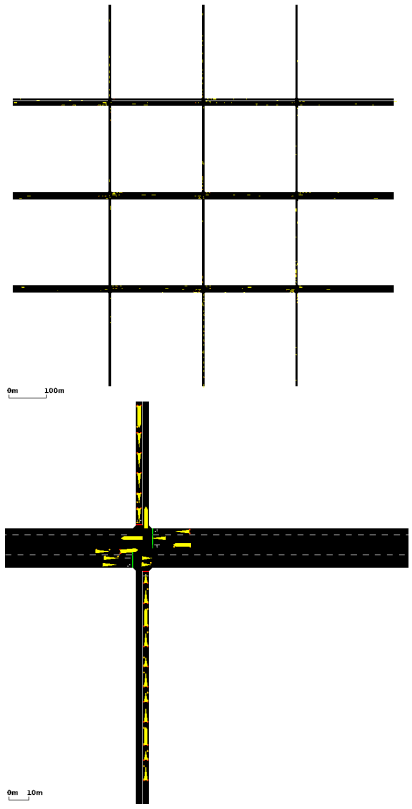

Figure 2: Top: Simulated roadway visualized in SUMO. 1-3 lanes in each direction. Traffic-lights at each of the 9 intersections. Cars can enter and exit in any of the 12 outer edges. Bottom: Expanded versions of the simulated roadways.

2. Data and Simulations

In this study, we utilize a 9-light grid-based world within the SUMO[2] environment [13], see Figure 2. In the simulations, cars enter and exit on any of the 12 edge segments. The speed limit for each road segment was chosen independently and randomly. Five types of car were instantiated with differing settings for base acceleration, deceleration, following distance, length, etc. The 9 light controllers employ fixed-time controllers that are independently optimized; we will discuss the applicability to actuated controllers in the context of future work.

In our previous experiments with real-user data, we determined that modeling traffic density through anonymized location tracks collected from opted-in Google/Android cell phone users [14] provided a robust and reliable source of flow information — an alternative to historic induction loop sensor data. For this study, we simulated the collection of

1. To avoid potential confusion, it should be pointed out that *reinforcement learning* [4]–[8] encompasses a largely separate set of machine learning techniques that have been used as alternatives to stochastic search algorithms for creating the light controllers.

2. SUMO was chosen because it is available to all researchers (open-source), extensible, and allows for massively parallel scenario testing, which was crucial for the $O(10^5)$ simulations required for the experimental results.

similar data: traffic density measurements of all the road segments to be controlled were recorded during the entire length of the simulations.

To test the limits of this approach, traffic flows were constructed with variability beyond what would normally be expected. This variability would render any single light program (fixed or with induction-loops or other sensors) sub-optimal. 100 independently generated probability distributions, specifying the probability of each of the 12 external edges being the origin and destination of the simulated vehicles, were stochastically created.

To provide intuition to what these probability distributions represent, recall that commonly the day is divided into semantically meaningful time intervals: rush-hour, night, weekend, etc. Though overly simplified, each of these can be thought of as having their own underlying probability distribution of entrance and exit edges. In the morning rush hour, the probability of entering on edge A and exiting on edge B may be much higher than entering on edge A in the evening rush hour. In fact, the entrance and exit probabilities of edge A and edge B may be reversed between the morning and evening rush-hours. This does *not* suggest that all morning rush hours will look the same – only that in aggregate the probability of entry and exits will be similar for scenarios drawn from the same underlying probability distribution. However, even small variations in sampling the probability distribution can create drastic changes in the traffic flows [15] [16] and where the traffic problems occur. Therefore, each time a probability distribution is chosen for sampling, it yields a unique traffic flow. Interestingly, when scenarios are created by sampling a probability distribution, they may yield a traffic flows that, when examined in isolation, are difficult to ascertain which probability distribution was used to generate it.

By using 100 independent probability distributions, we are implying that for the full-system we are attempting to model, there are 100 such underlying distributions (or "unique time-periods") that represent the expected traffic flows. In reality, we expect fewer (e.g. rush-hours, weekends, nights, midday, etc), but if we can tackle the problem of 100 scenarios, a smaller number can easily be handled. Handling a large number of scenarios provides the ability to model even uncommon situations that may have been observed in the data collection phase. This is beneficial as it allows us to expand the applicability of the system simply by using more historic data, encompassing more periods of the day, without sacrificing the performance at peak times. Remember that in real-world deployment we would not know the *true* number of the unique underlying distributions that best represent the system. However, as we will demonstrate, the exact number is not needed.

We simulated collecting data from 500 unique recording sessions each of length 45 minutes. For each of the 500 recording sessions, one of the 100 distributions was uniformly randomly selected to be the underlying generating distribution. No attempt was made to equalize the sampling between the 100 probability distributions. For each session, 4,000 cars and paths were instantiated by randomly drawing from the associated origin-destination distribution. As expected, even among sessions drawn from the same underlying distribution, tremendous variability in the backups and delays was created just by the randomized sampling.

With the data gathered, we now turn to establishing a viable, realistic, baseline performance. The optimization procedure used to find the baseline is described next.

3. Establishing a Realistic Baseline

No matter which algorithm is used for traffic light control, each approach has numerous parameters that must be specified to complete the program. With a simple fixed-schedule controller, the length of the phase and offsets of each light have a large impact on the performance of the overall system. As mentioned earlier, many machine learning approaches have been used in setting these parameters. Perhaps the most common approach seen in traffic light optimization literature is the use of genetic algorithms (GA) [17] to set the numeric and/or enumerable values associated with traffic lights [18]–[21].

We also started our exploration of optimization techniques with GAs and other evolutionary variants following published research. Despite the prevalence of genetic algorithms in this domain, we found that a much simpler mechanism, *next-ascent stochastic hillclimbing (NASH)* works as effectively as GAs and is simpler to implement and faster in practice. This counter-intuitive result has also been observed by other researchers in exploring the trade-offs between genetic algorithms and stochastic hillclimbing techniques [22]. This trend is especially pronounced in problems in which mutation (as opposed to the genetic algorithm's *crossover* operator) is the main driver for improvement in the solution — as we have found to be the case in these search-spaces. The *NASH* algorithm is described below.

With any controller, we start by specifying the set of parameters that can be modified. For the studies presented here, the set of parameters is the concatenation of all the parameters of all the lights considered (if there are 5 modifiable parameters in each of the controllers of 10 lights, there are a total of $5 \times 10 = 50$ parameters in the set). At the conclusion of the optimization process, the set of lights are *not* constrained to have the same logic as each other since each lights' parameters are fully specified within the parameter set. Once the set of parameters to be modified is specified, NASH operates as follows.

1) A parameter is randomly chosen from the set and the modification operator for that parameter is applied. In the simplest case, if the parameter is a real number, it is perturbed by a small amount (for example $\pm 5\%$). If the parameter can take on a set of distinct values, a value other than the one currently selected is chosen. Alternatively, instead of a single modification, multiple parameters can be chosen and modified.
2) Once the parameter modifications are completed, the schedule is then "repaired", if needed. The repair process ensures that the parameters are consistent with each other and are set within the appropriate ranges. For

example, in the case of fixed-schedule light settings, we may want to ensure that the overall cycle of the light remains constant to keep all lights synchronized, but the individual phase lengths can change. In this case, once a phase length perturbation has been made, the repair process ensures that the other phase lengths are enlarged/reduced appropriately to compensate and keep the overall cycle length constant.

3) Once any repairs are made, the new schedule is evaluated with the desired *objective function*. The standard objective functions that have been used in the literature are minimizing the overall/average wait time, maximum wait time, emissions, stop time, or maximizing throughput, speed, etc. For our studies, we set the objective function to minimize total travel time.
4) If the perturbations improved the performance on the objective function over the previous settings without the perturbations, the perturbation is accepted, and the schedule with the perturbation becomes the new baseline.
5) If the perturbation has not performed as well on the objective function, the set of perturbations is recorded (so that they are not explored again together), the perturbations are discarded from the schedule, and the previous baseline remains unchanged.

The exact number of perturbations made in each iteration is chosen stochastically. This process is iterated until either a satisfactory solution is found or time expires.

NASH is used to create the light schedules based upon a set of gathered data. NASH was allowed to modify each light's phase and offset timings independently. Recall that with real data, it would be impossible to ascertain the exact number of underlying distributions that generated the scenarios (*if* we had this information, we would ideally select scenarios from each distribution for training). Instead, to reflect real usage, the training scenarios for NASH were randomly selected from the set of 500. This is typical of the majority of real-world light optimization work where the data is gathered over a limited set of days/scenarios and is then used to optimize the timings. It mirrors what was done in our studies of Mountain View and Chicago [1].

Once the light programs were optimized, all 500 scenarios were again simulated and the density of flows on each road segment were recorded (as specified previously). Optimization of the programs yielded significant reduction in emissions and travel times over the SUMO default. These optimized lights provide a strong baseline of performance to which improvements will be measured in the rest of this study. The results, after optimization, are shown in Figure 3.

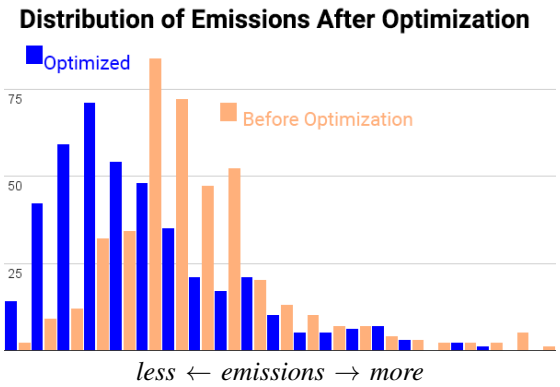

Figure 3: Histogram of relative emissions for NASH-optimized lights (blue/darker) vs. SUMO default (orange/lighter) on all 500 scenarios. Notice (1) There is significant variability across the overall emissions across the different scenarios (220%). (2) The overall left shift of the histogram for the optimized timings (NASH optimized): this reflects the substantial *decrease in emissions*. Similar graphs were obtained for time (not shown).

4. Clustering for Program Selection

In the last section, we created a traffic light program that works well, on average, over many of the scenarios encountered. Next, we will create specific controller rules to work well on many individual scenarios. First, we describe how to select the set of scenarios to use for training the individual controllers. Next, we explain the procedure to recognize which scenario the currently observed flow of traffic most closely resembles – and thereby which set of light controllers should be instantiated.

We re-examine the 500 scenarios to cluster them into similar groups. In this study, we know which underlying probability distribution generated each of the scenarios; however, with real data, neither the underlying probability distributions nor even their number are known. Therefore, to realistically design this system, we need to perform this clustering with no knowledge other than the observed traffic flows. The clustering approach groups like-scenarios so that they can be tractably handled. The stochastic nature of the sampling, coupled with the chaotic nature of traffic flows, may yield clusters that are different than the underlying generating distributions. Nonetheless, as long as the clusters are grouped with respect to similarity, matching to the underlying generators is not necessary. The clustering steps are given in detail:

1) Using the optimized lights described in Section 3, the density of traffic for each of the 500 simulated scenarios is measured. Specifically, the number of vehicles on each directional-road-segment is recorded for 9 non-overlapping 100-second periods (15 min.). This characterizes the traffic conditions in any 15-minute interval by a time-series of traffic densities on each road segment. To be specific, for our experiments, with 9 entries for each of the 48 road segments in the map, the result is a 432 dimensional vector. Intuitively, when two traffic conditions are similar in their traffic distributions, they should be close in this simplified feature space as well. From the 432 dimensions, we further sub-select only those that exhibit high variance across the 500 scenarios and discard the rest. This is a simple form of *feature subset selection* to find discriminative features [23], [24].

2) Using the selected measurements in Step (1), the full 500×500 matrix of correlations between all scenarios is calculated. This reveals, for each pair of scenarios, how similar the density of traffic on the road segments is, when measured over the 15 minute intervals.
3) The scenarios are clustered into C clusters; the similarity between clusters is specified by the correlation matrix. Numerous clustering procedures were explored, a procedure similar to Learning Vector Quantization (LVQ) was finally employed [25]. Briefly, in LVQ, C points are randomly placed in the high-dimensional feature space (of the selected features). Each scenario is assigned its closest point, c (from C). Then, simultaneously, all of the C points are moved closer to the centroid of their matched scenarios. The procedure is repeated until the points no longer move. Note that with real data, we will not know the correct number of clusters. Overestimating $|C|$ will not degrade performance since information will be be present, but duplicated. As the effects of underestimating the number are less certain, we study the performance with $|C| = 10, 50, 100$.

At the completion of Step 3, the clusters have been created. Given these clusters, we now optimize the lights settings to the traffic flows for each cluster. For each cluster, c, where $c \in C$, we find the scenario, s, where $s \in c$, that is closest to the c's centroid. With s, the light programs for all 9 lights are trained from scratch to reduce emissions (or reduce delays, etc) just on the particular scenario, s. This yields a specialized light setting, L_c. Exactly as before, the system of lights is trained using *NASH*. More scenarios that were assigned to cluster c can be used in training L_c, at the expense of extra computation. At the completion of this step, $|C|$ specialized lights settings (each for 9 lights) are created: one for each cluster.

We now have $|C|$ light programs (L_c) — each optimized to a particular cluster/scenario. Each L_c is tested on *all* 500 scenarios, not just those in it own cluster c. Why test each against all 500 scenarios? It is tempting to assume that scenarios in cluster c will perform best with light setting L_c; however, due to the stochastic nature of training, it is possible that local minima may have been encountered in training and that other light settings have superior performance. Usually these are from clusters that share many similar attributes. Further, some scenarios may have traffic densities that are not well represented by their assigned cluster. At the completion of these $500 \times C$ simulations, each scenario from the 500 is now assigned the single light setting (L_s) that best reduced emissions. Next, we describe how these are used in practice.

5. Experiments

Upon completion of the procedure detailed in the previous section, we have a repository of 500 scenarios and the traffic-light program (from the set of 10,50 or 100 of them depending on the experiment) that work best with each. In deployment, every 15 minutes, 432 traffic-density measurements across all the road segments are recorded. These are obtained by examining patterns in Android cell phone usage (or alternatively, any mechanical or camera-based "standard" sensors can be used). From these measurements, data extraction proceeds in a straightforward adaptation of the method described earlier:

- Data Collection: 432 traffic density measurements are recorded.
- Data Reduction: The same high-variance dimensions found in Step (1) are extracted and the rest discarded.
- Find Best Match: Based on the extracted features, the the most similar k scenarios out of the the 500 are found. This is done via a *k-nearest neighbor* search on the high-variance-dimensions of the 432-dimensional vector using an L_2-Euclidean distance metric.
- Deployment of the new Program: As in standard k-nearest-neighbor lookup, from these k matches, the associated best light settings are tabulated. The setting with the most votes (weighted by the scenario's closeness) are deployed to the 9-lights. (In our experiments, $1 \leq k \leq 10$, revealed similar, good, performance).

TABLE 1: Relative reduction in emissions and travel-time on 1000 scenarios using Match-Based Program-Selection. Consistent 10-12% improvement. This is measured over previously *optimized* timings. The lower portion of the table provides the number of scenarios that were improved, made worse, or stayed the same using the Match-Based-Program Selection.

	Optimized Baseline	C=100	C=50	C=10
Emissions (CO_2, mg)	100%	90.6%	91.9%	91.7%
Travel Time (sec.)	100%	87.7%	89.4%	89.2%
# Scenarios Improved		900	884	925
Tied		18	23	31
Worse		82	93	44

To fairly test our system, we can no longer use the 500 scenarios from which we developed the model. Instead, 1,000 *new* scenarios were created. As before, the scenarios were generated by uniformly sampling one of the 100 distributions of origin/destination pairs. Then, the 1,000 scenarios were simulated with the above procedure instantiated to control the lights. No knowledge of the probability distributions used to generate each scenario was given to the system; all similarity measurements were directly based on observation of traffic flows — reflecting how the system would be deployed.

Table 1 compares the performance of the baseline

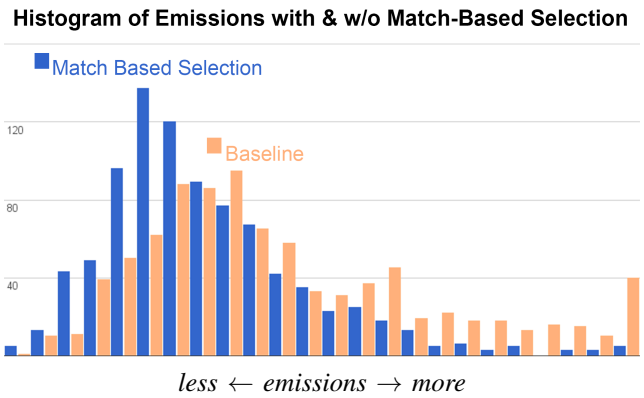

Figure 4: Histogram of emissions of trials with and without Match-Based Selection. The baseline (orange/lighter) is a previously optimized light schedule. Note histogram's shift to the left (lower emissions) with matching (blue/darker), indicating significant reduction emissions. Similar graphs were seen with trip-completion times.

NASH-optimized lights with those that used the match-based light switching to instantiate a new program after 15 minutes of observation. The reduction in CO_2 emissions and travel times, compared to the single NASH-optimized baseline are given (reductions in CO are similar).

Table 1 also shows the number of scenarios that had an overall reduction in emissions and a lower average travel time. Approximately 90% of the scenarios were improved. This is also reflected in Figure 4, where there is a large shift to the left (lower emissions) when the clustering approach was used.

Despite the large variability in the distributions of flows, and the number of unique underlying probability distributions (100) that were employed to generate the traffic flows, the results were very encouraging across all settings of $|C|$. For example, though there was a large mismatch between the number of clusters hypothesized when we set $|C| = 10$ and the actual number of underlying probability distributions, the system modeled a sufficient amount of the underlying variability to yield improvement in the majority of test scenarios (Table 1, last column). This resulted in substantial average savings across all scenarios.

6. Discussion and Future Work

Unlike many previous studies, we do not change light programs based on set times. Additionally, we do not tune programs on-the-fly to adapt to changing conditions. Instead, we continuously monitor the traffic flow to select a light program, from a large set of pre-optimized light programs, that works best for the observed traffic conditions. The approach is fully data-driven and, as such, requires an extensive number of simulations to not only optimize a large set of light programs, but also to assign each program to the traffic flows that is best suited to handle.

In the first step of this study, we set the light controllers such that the program instantiated worked well across a wide variety of traffic flows. Though this, itself, provided a substantial benefit over default settings, we then tried to optimize the controllers to more specific traffic flows. To find out which traffic flows were best representative of those seen, we used a clustering approach, with the appropriate feature selection mechanisms, to group the observed traffic flows and optimized the parameters for each cluster. In deployment, the current traffic flows is matched to the previously seen clusters, and the associated light program instantiated.

Improved performance in both emissions and travel time is consistently obtained over the previously optimized lights by matching observed traffic densities to similar scenarios and then instantiating the known good light setting for that scenario. Intuitively, each scenario serves as a fingerprint to which observed densities are matched. The most similar related approaches comes from the field of Case-Based Reasoning, see [26]–[28].

This paper incorporated machine learning into the procedure of light-program *selection*. This works in conjunction with related machine learning approaches that are used for the individual light program *optimization*. By using the clustering approach to match current flows to those previously seen, we can expand the applicability of the optimized system by simply collecting more historic data that encompasses more periods of the day, without sacrificing the performance at peak times.

There are five immediate next steps. The most important is moving beyond the synthetic data – scaling and testing this system on city traffic data is paramount. Using the synthetic data, we were able to more thoroughly examine the effects of number of clusters, the clustering procedures and the optimization schemes. The experimental results were encouraging: the large number of underlying distributions that were successfully modeled is promising for the variations expected in real data. Second, we presented a new approach to adapting traffic light schedules - simply selecting the best one from a set of pre-optimized programs. This should be quantitatively compared to previous approaches, such as those that rely on adaptation of existing schedules. Third, employing this system with lights triggered by external (*e.g.* induction-loop/camera) sensors is readily possible, though we will need to address the potential challenges in measuring *similar* traffic flows, as the variations in expected flows within any single program may be substantially different with actuated controllers than with non-actuated controllers. Fourth, though the computational requirements for this study were immense, with well over 20,000 simulations performed, there is room for further experiments. Ablative studies and determining the effects of non-uniform distribution sampling should be conducted. This can be done in simulation prior to collecting real data. The fifth avenue for future work is the most speculative. In this work, we used a system to discretely select between light controllers. The method for determining similarity was easily implemented: k-nearest neighbors with an L_2 distance. An alternative is

to use tools such as deep neural networks to not only model and match the traffic flows to previously seen examples, but also to potentially weight and mix multiple controller settings to best address the current traffic flows.

References

[1] M. Covell, S. Baluja, and R. Sukthankar, "Micro-auction based traffic light control," in *ITSC-2015*, 2015.

[2] S. Baluja, M. Covell, and R. Sukthankar, "Approximating the effects of installed traffic lights:a behaviorist approach based on travel tracks," in *IEEE Intelligent Transportation Systems Conference-2015*, 2015.

[3] ——, "Traffic lights with auction-based controllers: Algorithms and real-world data," *CoRR*, vol. abs/1702.01205, 2017. [Online]. Available: http://arxiv.org/abs/1702.01205

[4] M. Wiering, "Multi-agent reinforcement learning for traffic light control," in *ICML*, 2000, pp. 1151–1158.

[5] I. Arel, C. Liu, T. Urbanik, and A. Kohls, "Reinforcement learning-based multi-agent system for network traffic signal control," *Intelligent Transport Systems, IET*, vol. 4, no. 2, pp. 128–135, 2010.

[6] B. Abdulhai, R. Pringle, and G. J. Karakoulas, "Reinforcement learning for true adaptive traffic signal control," *Journal of Transportation Engineering*, vol. 129, no. 3, pp. 278–285, 2003.

[7] K. Dresner and P. Stone, "Traffic intersections of the future," in *AAAI-2006*, 2006, pp. 1593–1596.

[8] D. E. Moriarty and P. Langley, "Learning cooperative lane selection strategies for highways," in *AAAI/IAAI,1998*, 1998, pp. 684–691.

[9] E. M. Almannaa M. and R. H.A., "A novel clustering algorithm for traffic operational analysis," in *97th Transportation Research Board Annual Meeting*, 2018.

[10] J. Luk, "Two traffic-responsive area traffic control methods: Scat and scoot," *Traffic engineering & control*, vol. 25, no. 1, 1984.

[11] C. Kergaye, A. Stevanovic, and P. T. Martin, "An evaluation of scoot and scats through microsimulation," in *International Conference on Application of Advanced Technologies in Transportation, Transportation and Development Institute, Athens, Greece*, 2008.

[12] A. Stevanovic, C. Kergaye, and P. T. Martin, "Scoot and scats: A closer look into their operations," in *88th Annual Meeting of the Transportation Research Board. Washington DC*, 2009.

[13] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO - simulation of urban mobility," *Int. J. On Advances in Systems and Measurements*, vol. 5, 2012.

[14] D. Barth, "The bright side of sitting in traffic: Crowdsourcing road congestion data," http://googleblog.blogspot.com/2009/08/-bright-side-of-sitting-in-traffic.html, 2009.

[15] T. Nagatani, "The physics of traffic jams," *Reports on progress in physics*, vol. 65, no. 9, p. 1331, 2002.

[16] T. Li, "Nonlinear dynamics of traffic jams," *Physica D: Nonlinear Phenomena*, vol. 207, no. 1-2, pp. 41–51, 2005.

[17] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[18] J. J. Sanchez, M. Galan, and E. Rubio, "Genetic algorithms and cellular automata: a new architecture for traffic light cycles optimization," in *Congress on Evolutionary Comp.*, vol. 2. IEEE, 2004, pp. 1668–1674.

[19] B. Park, C. J. Messer, and T. Urbanik II, "Enhanced genetic algorithm for signal-timing optimization of oversaturated intersections," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1727, no. 1, pp. 32–41, 2000.

[20] A. M. Turky, M. S. Ahmad, and M. Z. M. Yusoff, "The use of genetic algorithm for traffic light and pedestrian crossing control," *Int. J. Comput. Sci. Netw. Security*, vol. 9, no. 2, pp. 88–96, 2009.

[21] T. Kalganova, G. Russell, and A. Cumming, "Multiple traffic signal control using a genetic algorithm," in *Artificial Neural Nets and Genetic Algorithms*. Springer, 1999, pp. 220–228.

[22] A. Juels and M. Wattenberg, "Stochastic hillclimbing as a baseline method for evaluating genetic algorithms," in *NIPS*, 1995.

[23] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial intelligence*, vol. 97, no. 1-2, pp. 273–324, 1997.

[24] P. Mitra, C. Murthy, and S. K. Pal, "Unsupervised feature selection using feature similarity," *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 3, pp. 301–312, 2002.

[25] T. Kohonen, "Learning vector quantization," in *Handbook of Brain Theory and Neural Networks*, 1995.

[26] A. Kofod-Petersen, O. J. Andersen, and A. Aamodt, "Case-based reasoning for improving traffic flow in urban intersections," in *Case-Based Reasoning Res. and Dev.* Springer, 2014, pp. 215–229.

[27] D. B. Leake, *Case-Based Reasoning: Experiences, lessons and future directions*. MIT press, 1996.

[28] J. Kolodner, *Case-based reasoning*. Morgan Kaufmann, 2014.