CoSIGN: A Parallel Algorithm for Coordinated Traffic Signal Control

Shih-Fen Cheng, Marina A. Epelman, and Robert L. Smith

***Abstract*— The problem of finding optimal coordinated signal timing plans for a large number of traffic signals is a challenging problem because of the exponential growth in the number of joint timing plans that need to be explored as the network size grows. In this paper, we employ the game-theoretic paradigm of fictitious play to iteratively search for a coordinated signal timing plan that improves a system-wide performance criterion for a traffic network. The algorithm is robustly scalable to realistic-size networks modelled with high fidelity simulations. We report results of a case study for the the city of Troy, Michigan, where there are 75 signalized intersections. Under normal traffic conditions, savings in average travel time of more than 20 percent are experienced against a static timing plan, and even against an aggressively tuned automatic signal re-timing algorithm, savings of more than 10 percent are achieved. The efficiency of the algorithm stems from its parallel nature. With a thousand parallel CPUs available, our algorithm finds the plan above in under 10 minutes, while a version of a hill-climbing algorithm makes virtually no progress in the same amount of wall-clock computational time.**

***Index Terms*— Coordinated traffic signal control, optimization, area traffic control**

I. Introduction

Since Webster and Cobbe [1] first published their research on pre-timed isolated traffic signal control, significant progress in traffic signal control has been made. With the introduction of advanced computer, control, and communication technologies in traffic networks, signal control systems are now able to receive more network-related information and respond in a more congestion-adaptive manner. From past research, we can see that, in general, the more information a signal controller uses, the better performance it can achieve. However, the complexity of algorithms for designing signal timing plans correspondingly grows as more information is being utilized. Another factor that complicates the problem is the number of signalized intersections considered. In the general case, with non-periodic signal timing plans allowed, the size of the problem grows exponentially as the number of considered signals increases. Therefore in practice, the tradeoff between the accuracy of the algorithm, the amount of traffic-related information used, and the size of the network remains an issue.

Based upon amount of information used in the control schemes, we can classify related research into the following categories:

1) **Offline**: Pre-timed signal control schemes for both isolated and coordinated signal control belong to this category. Since pre-timed signal timing plans are computed in an offline manner, they can only use information related to historical flow statistics and network configuration. Webster's method [1] and its extensions, SIGSET [2], and SIGCAP [3] are examples of isolated control methods (only a single signalized intersection is considered). MAXBAND [4], [5] and its extensions, and TRANSYT [6] are notable examples of coordinated control methods (a group of signalized intersections is considered simultaneously).
2) **Online**: The use of sophisticated surveillance technologies, including inductive loop detectors and surveillance cameras at signalized intersections, enables traffic signal controllers to make use of real-time traffic information. This information, including, but not limited to, vehicle counts, link volume and link occupancy, proved to be very useful in computing real-time signal timing plans for both isolated and coordinated signal control. Most modern traffic signal control technologies belong to this category. For the isolated control case, it was Miller [7] who first proposed a control strategy based on online traffic information. Other more recent methods include SCATS [8], PRODYN [9], [10], OPAC [11], [12], UTOPIA [13], SPPORT [14], COP [15]. It should be noted that although many of the above control strategies (e.g., OPAC, PRODYN and SCATS) are also used in coordinated control, the coordinations are mostly done heuristically due to the combinatorial complexity of the problem. Other notable research that focuses on the coordinated control problem includes SCOOT [16], CRONOS [17], REALBAND [18], Lin and Wang [19], and Heung et al. [20].
3) **Predictive**: Based on offline and online information, the next promising extension is to come up with predictions of future network congestion, and compute the signal timing plans in anticipation of predicted future traffic conditions. An example of such an approach is RHODES [21], [22]. It uses a combination of current real-time information and planned timing plans from upstream signals to predict future arrivals.

Among these three categories, the control schemes with offline and online information are well-studied and are widely implemented. In comparison, control schemes that are capable of using predictive information are still mostly experimental and researchers are just beginning to explore the benefits of

Manuscript received.

S.-F. Cheng, M. A. Epelman and R. L. Smith are with Department of Industrial and Operations Engineering, the University of Michigan, Ann Arbor, Michigan 48109-2117, USA (e-mail: chengsf@umich.edu, mepelman@umich.edu, rlsmith@umich.edu)

using such information.

The method we propose in this paper does make use of such predictive information. We rely on information on time-dependent origin-destination flows, which can be used to predict link congestion in the future. We believe that high quality predictive information will become more and more accessible due to the following two important technological advances. The first important advance is high quality estimation of dynamic origin-destination trip flows [23], [24]. The second is the use of vehicle-based GPS systems and other vehicle tracking technology in vehicle routing. With such equipment, we can precisely collect the origin-destination information for the "smart" vehicles (i.e., vehicles outfitted with such equipment). Also, by using these vehicles as traffic probes, we can get better estimates of current link congestions. By combining the above two branches of research, high quality predictive information required by our method should become available. The first goal of the paper is thus to introduce an algorithm that is capable of incorporating this predictive information in computing adaptive traffic signal timing plans.

Another goal of this paper is to address the difficulty of finding solutions to the combinatorial problem that arises in general coordinated traffic signal control. The size of the set of solutions that need to be considered grows exponentially as the number of intersections and/or the length of the time horizon considered increases. Moreover, functions typically used to measure performance of the network, such as, for example, average trip time experienced by the drivers, have to be evaluated via computationally intensive traffic simulators. These functions also lack structural properties that traditional optimization algorithms rely upon, calling for novel methods for searching the solution space. Our algorithm allows for parallel execution, which makes real-time signal control possible even in a large network. The applicability of our approach (called **CoSIGN**, for "Coordinated SIGNals") is demonstrated by a test case study based on the real traffic network of Troy, Michigan.

The paper is organized as follows. In section 2, the problem formulation is stated. In section 3, we motivate the use of a game-theoretic approach to this problem. In section 4, the necessary technical background is provided and the algorithm is stated. In section 5, we restate the coordinated traffic signal control problem in game theoretic terms, and explain the details of the algorithm's implementation. In section 6, the test case and results of experiments are discussed. Future work is proposed in section 7.

II. Traffic signal control problem formulation

We consider the problem of finding an optimal coordinated traffic signal plan for a group of signalized intersections over a given time horizon. A problem instance is defined by specifying the topology of the traffic network, the time horizon, as well as the time-dependent origin-destination flows over this time horizon. In particular, for every origin-destination pair in the network, the timing of vehicles' departures from the origin for the destination and the route they take are presumed to be known. The goal is to minimize the average travel time experienced by all drivers in the network during the given time horizon (we use the terms "driver" and "vehicle" interchangeably).

We formulate this coordinated traffic signal control problem as a discrete optimization problem, where the planning horizon is divided into N time periods of equal length of δ seconds, and the decision variables are the *signal phases*[1] prevailing during each of the N time periods, at each of the I signalized intersections[2]. The following notation will be used in describing the coordinated traffic signal control problem:

- $\mathbf{I} = \{1, 2, \ldots, I\}$: set of signalized intersections;
- $\mathbf{N} = \{1, 2, \ldots, N\}$: set of time periods (each time period is δ seconds long);
- $\mathbf{S}_i = \{1, 2, ..., S_i\}$: set of permissible signal phases for intersection i, $i \in \mathbf{I}$;
- $s_{i,n} \in \mathbf{S}_i$: a decision variable representing the signal phase at intersection i during time period n.

The problem can be formally written as:

$$\begin{aligned} \min \quad & \textsc{AverageTravelTime}(\{s_{i,n}, i \in \mathbf{I}, n \in \mathbf{N}\}) \\ \text{s.t.} \quad & \\ & s_{i,n} \in \mathbf{S}_i, \forall i \in \mathbf{I}, \forall n \in \mathbf{N} \end{aligned} \tag{1}$$

where the mapping from the vector of decision variables, $\{s_{i,n}\}$, to the objective value is represented by the function $\textsc{AverageTravelTime}(\cdot)$, which reflects the performance measure we discussed above. The dependence of this function on the decisions made in the problem, i.e., the signal timing plans over the planning horizon, is inherently complex and possesses neither analytical representation nor known structural properties (such as monotonicity or subadditivity). In effect, we are faced with a problem of optimizing a "black-box" function. In particular, in our research, all function evaluations are provided by a traffic simulation program, as described in subsection V-B.

One immediate concern resulting from this formulation is the exponential explosion of possible joint decisions as N and I get larger. In the worst case, all joint decisions, with number bounded by $(\max_i\{S_i\})^{N \cdot I}$, have to be enumerated and evaluated in order to find an optimal solution to assure global optimality. For a practical size problem, this is impossible. Therefore, we take the approach of searching for a high-quality locally optimal solution instead. Still, considering the complexity and scale of the problem, it is not obvious how even this can be achieved within reasonable time.

In the next section, we will propose the use of a game-theoretic approach to resolve our dilemma.

III. Motivation for a game-theoretic approach

In this section we briefly describe the motivation and the intuition behind using *game theory* in solving coordinated traffic signal control problems (see [25] for an early application to dynamic route guidance). Although some game

[1] A signal phase is a collection of traffic movements that receive right-of-way simultaneously. Therefore, all movements within a phase must be non-conflicting.

[2] By defining decision variables this way, we allow acyclic signal timing plans. In the absence of cyclic parameters, we assume that a fixed amount of yellow time is incurred if two consecutive decisions at a signal is different.

theory-related terms are mentioned throughout this section, their formal definitions are deferred to the next section. The intuition behind our approach is emphasized here.

Recall that the decision variables in our problem are the signal phases prevailing during each of the N time periods at each of the I signalized intersections. The number of joint decisions is thus bounded by $(\max_i\{S_i\})^{N \cdot I}$. The problem quickly becomes intractable as we increase N and/or I. However, if we decompose the problem into smaller subproblems, we may be able to find a sufficiently good solution in a reasonable amount of time.

The decomposition of the problem can be accomplished by assuming that each signal in each period is an independent decision maker. By adopting this decomposition, the centralized decision problem, with $(\max_i\{S_i\})^{N \cdot I}$ possible decisions, can then be transformed into $(N \cdot I)$ subproblems, each with at most $\max_i\{S_i\}$ possible decision alternatives. The effect is to reduce an exponential to a linear number of alternatives to consider. However, if we decompose the problem without considering the interactions among these independent decision makers, we are just solving $(N \cdot I)$ isolated signal control problems over very short time horizons, and there is no coordination among traffic signals.

In order to effectively incorporate coordination of a large number of decision makers, we turn to game theory, which originates from economics. Modern game theory was created after von Neumann and Morgenstern [26] in 1944 and quickly became a popular tool in explaining and predicting behavior of groups of rational decision makers (*players* in game theory terminology) when their well-beings are associated with the joint actions of all decision makers (players). If each decision maker who controls a time period for a signal is viewed as a player in the game, and the average travel time of all vehicles in the traffic network is viewed as a *common payoff* for every player, the coordinated traffic signal control problem can then be represented as a *game of identical interests*. The notion of a solution to a game is that of a *Nash equilibrium* (a similar, but more transportation-specific result is Wardrop's principle [27]), which for a game of identical interests can be viewed as a coordinate-wise local optimum. Intuitively, a joint decision is a Nash equilibrium if no individual player can improve its payoff by unilaterally deviating from the original joint decision. Note that in a game of identical interests, Nash equilibrium is not necessarily a global optimum.

It is well-known that finding Nash equilibria is a hard problem [28]. One of the earliest algorithms used to find Nash equilibria is an iterative process called *fictitious play* [29], [30]. The primary pitfall of fictitious play (FP) is that in general it does not converge to an equilibrium. However, Monderer and Shapley [31] showed that for a special class of games, namely games of identical interests, FP will converge to equilibrium. Since virually all unconstrained discrete optimization problems can be represented as games of identical interests, this result has recently inspired researchers in optimization to introduce FP as an optimization tool [32], [25]. In this paper, after we model the traffic signal control problem as a game of identical interests, we will apply a variation of the FP algorithm to find a solution.

IV. Game theory and the FP algorithm

In this section, we formally define a game and the solution concept of a Nash equilibrium, and discuss how one can use FP to find a Nash equilibrium of a game.

A. Game theory fundamentals

Game theory studies how independent decision makers would act under the assumption that an individual's payoff will be determined by actions of all participants. We now define the components of a game.

- **Players:** Each independent decision maker in the game is defined as a player. Every player has a finite set of decisions called **strategies** (or **pure strategies**) that it can choose from (or "play," in game theory terminology). A **mixed strategy** is a probability distribution over the set of the player's strategies. A **joint strategy** is a specification of (mixed) strategies for all players.
- **Payoff function:** For every player, its associated payoff function is defined as a mapping from joint strategies to the corresponding payoff this player will get were these joint strategies played (or expected payoff, if mixed strategies were played). In general, players may have different payoff functions. However, in this paper, all players will be assumed to have identical payoff functions.
- **Best reply function:** Given an arbitrary joint strategy, a player's best reply function will return a strategy that gives this player its highest payoff value, assuming that all other players use the strategies specified in this joint strategy. As we will see later, this is the critical operation in our approach.
- **Nash equilibrium:** A joint strategy is a Nash equilibrium if no individual player can improve its payoff by unilaterally deviating from the play of the original joint strategy. More precisely, a joint decision is a Nash equilibrium if for every player, its current decision is its best reply against this joint strategy. In other words, Nash equilibrium is a fixed point of the best reply function.

The first important existence theorem, proved by Nash [33], stated that every finite game in strategic form[3] has a mixed strategy equilibrium.

For a complete treatment of these introductory terms and concepts, we refer to Fudenburg and Tirole [34].

B. FP and SFP algorithms

Computing Nash equilibria can be a difficult task. McKelvey and Mclennan's work on GAMBIT [28] is an excellent reference for various computational methods for finding Nash equilibria. In this research, we will use a simple-to-implement iterative algorithm which is a variation of Fictitious Play (FP). Convergence results for the FP algorithm and its variants are stated in [31], [32]. Since in this paper we are mainly interested in solving the traffic signal control problem, most technical

[3]A game is said to be in strategic form if it has a finite set of players, each player has a nonempty strategy set, and each player's payoff functions are properly defined for all joint strategies. A strategic game is finite if the number of players and all players' strategy sets are finite.

details are omitted here. We refer interested readers to [32] for a complete treatment.

The intuition behind FP lies in the theory of learning in games. In a classical FP process (see, for example, [29]), every player assumes that other players are playing unknown stationary mixed strategies, and tries to learn them iteratively. The estimates of the unknown stationary mixed strategies are represented as *belief distributions*, or *beliefs*, and are shared among all players. The belief distribution for player i is a mixed strategy calculated by finding the relative frequency of all strategies from the history of its past plays. During each iteration, each player finds its *best reply* against the belief distribution of other players (i.e., its belief of how they will play). These best replies are then included in the history of past plays and the beliefs are updated accordingly. To start the FP process, an arbitrary joint strategy is used. The FP algorithm doesn't converge to equilibrium in general. However, for games of identical interests, as in our case, the sequence of beliefs generated by the FP algorithm are guaranteed to converge to equilibrium [31].

The best reply operation of the classical FP algorithm outlined above is too computationally expensive to implement in practice. Lambert et al. [32] thus suggested a variant they called *sampled fictitious play* (SFP) that is computationally practical. SFP is very similar to FP except the best reply evaluation in each iteration is done against samples randomly drawn from the belief distribution instead of the belief distribution itself. A convergence result for SFP with gradually increasing sample sizes is proved in [32]. In practice, however, samples of size one are often used at each iteration.

The SFP algorithm, with sample size one, is described below:

1) **Initialization:** An initial joint strategy is chosen arbitrarily. It is then stored in the history.
2) **Sample:** A strategy is independently drawn from the history of each player (i.e., for each player, each past play is selected with equal probability).
3) **Best Reply:** For every player, the best reply is computed by assuming that all other players play the strategies drawn in step 2.
4) **Update:** The best replies obtained in step 3 are stored in the history.
5) **Stop?** Check if the stopping criterion is met; if not, go to step 2, otherwise stop.

The pseudo-code for the SFP algorithm and the sampling subroutine is listed in Algorithm 1 below. This pseudo-code is specified for a game with P players. Here, **D** and **B** are P-dimensional vectors whose components contain individual strategies of the players, and $(\cdot)^T$ denotes the transpose operation. **H** is a "history" matrix, where $\mathbf{H}(k, j)$ represents player j's best reply in the k^{th} iteration. Notation $\mathbf{H}(k, :)$ represents the k^{th} row of matrix **H**, while $\mathbf{H}(:, j)$ is the column containing the history of past plays of player j. This representation of the history allows convenient access to relevant information for sampling in step 2.

Algorithm 1 implements the SFP algorithm in a straightforward way. Line 1 generates an initial solution (joint strategy) by calling function INITIALSOLUTION, thus populating the 0^{th} row of history matrix **H**. Line 4 performs uniform sampling from each player's history independently. Line 5 computes a best reply **B** to the sampled decision **D**. Line 6 appends **B** at the end of the history matrix **H**. Note that except for $k = 0$, each row k of matrix **H** stores best replies computed in iteration k. The above three lines are then repeated until STOPCRITERION returns **true**. Since BESTREPLY subroutine simply solves a collection of P one-dimensional optimization problems whose input is the sampled decision **D**, it can be executed in parallel. As we will see later, the parallelization of the best reply computation is the most important feature that makes SFP algorithm efficient.

Algorithm 1 Sampled Fictitious Play (sample size 1)

```
SFP()
 1: H(0,:) ← INITIALSOLUTION()
 2: k ← 0
 3: while STOPCRITERION() is false do
 4:     D ← SAMPLE(H, k)
 5:     B ← BESTREPLY(D)
 6:     H(k+1,:) ← B^T
 7:     k ← k+1
 8: end while

D = SAMPLE(H, k)
 1: for j = 1 to P do
 2:     u ← DISCRETEUNIFORM(0, k−1)
 3:     D(j) ← H(u, j)
 4: end for
 5: return D
```

Although this is not explicitly specified in the general pseudo-code, we will keep track of the "incumbent" solution, i.e., the pure strategy with best performance observed so far, throughout the algorithm. At termination, the SFP algorithm returns the current and therefore best incumbent solution.

The SFP algorithm was first implemented and used as an optimization scheme by Garcia et al. [25], who applied it to a dynamic traffic assignment problem. When compared to previously established methods, the SFP algorithm was able to obtain solutions of the same quality significantly faster. Lambert and Wang [35] further demonstrated the effectiveness of the SFP algorithm as compared to simulated annealing for a communication protocol design problem.

V. CoSIGN: SFP ALGORITHM FOR THE TRAFFIC SIGNAL CONTROL PROBLEM

As mentioned above, traffic signal control problems are usually solved by either restricting the space of solutions by searching for parameters of predetermined cyclic patterns, or by limiting the number of signals considerably. Instead, our approach will be to search for solutions to the full-scale coordinated signal planning problem by using the SFP algorithm.

To solve a problem with the SFP algorithm, we must first formulate it as a game. In the following sections, we will describe how to construct a game-theoretic model for the traffic signal optimization problem. Based on this formulation, we can then specify the performance measure used to evaluate signal timing plans and describe the best reply subroutine using this performance measure.

A. Formulating coordinated traffic signal control problem as a game

With the same notation as defined in section II, we can formulate the problem as a game:

- **Player:** each tuple (i, n), $i \in \mathbf{I}$, $n \in \mathbf{N}$, is a player. Let $\mathbf{P}$ be the set of all players, and $P = I \cdot N$, be the number of players.
- **Strategy Space:** for each player $(i, n) \in \mathbf{P}$, its strategy space is the set $\mathbf{S}_i$. Player (i, n)'s decision is denoted by $\mathbf{D}(i, n)$.
- **Payoff function:** by collecting decisions $\mathbf{D}(i, n)$ from all players, a signal timing plan for the planning horizon is formed. By sending this plan to the traffic simulator, we can find the average travel time experienced by all drivers, which is the payoff function value for all players.

B. Simulation by INTEGRATION-UM

Accurate evaluation of the average travel time can be accomplished by invoking a computer traffic simulator. In our experiment, the simulation is done by INTEGRATION-UM, developed by Van Aerde [36] and modified by researchers at the Intelligent Transportation Systems Research Center of Excellence at the University of Michigan. INTEGRATION-UM is an event-based, mesoscopic deterministic traffic simulator. In order to perform a simulation, we need to provide INTEGRATION-UM with following inputs:

- **Network topology definitions**: the transportation network is modelled as a directed graph in INTEGRATION-UM. To fully specify the network topology, we first define intersections and connection points as the nodes in the graph. There are two types of nodes in INTEGRATION-UM: zone centroids, which can be used as origins and destinations for the vehicle trips, and normal nodes, which can be used as intersections or connecting points. The roads are then defined as directed links connecting these nodes. Important physical properties of each link, including length, capacity, free-flow travelling speed[4], and the signal timing plan and the phase controlling this link (if any), must also be provided.
- **Traffic signal settings**: signal timing plans in the original version of INTEGRATION-UM were assumed to be cyclic. Cyclic plans were specified by parameters that define cyclic patterns, i.e., cycle length, green split, offset, and lost (yellow) time. We modified INTEGRATION-UM in order to take players' joint strategy as input. Note that with a short enough time period δ, the player model can emulate any cyclic pattern. Unlike cyclic plans, the signal timing plans specified by players' joint decisions incur lost time at intersection i only when players (i, n) and $(i, n + 1)$ in two consecutive periods n and $n + 1$ have different decisions.
- **Traffic flows**: INTEGRATION-UM assumes that the network is empty at the start of the simulation and all the traffic entering the network is generated by multiple "flows." Each flow, implicitly assumed to consist of only homogeneous motorized vehicles, is defined by specifying origin, destination, flow rate (in number of vehicles per hour), and flow starting and ending times. As mentioned in section I, this information is usually not directly available, therefore we must combine data from several sources, including survey, real time adjustments, and predictions, in order to come up with reasonable estimates. This is where accurate predictive information can really help us. With better predictive information, the simulation will better describe real traffic congestion, and this implies that CoSIGN will be optimizing a more realistic traffic simulation. As a result, for the signal timing plan generated by CoSIGN, the gap between its performance in the simulation and in the real traffic network should also become smaller.

[4]Free-flow travelling speed of certain link is the speed driver experiences when he/she is the only user of that link.

A detailed description of specifications of INTEGRATION-UM can be found in Wunderlich's PhD dissertation [37].

We selected INTEGRATION-UM as our traffic simulator purely on the basis of convenience of implementation, since its source code was readily available to us. We would like to emphasize that since our system architecture is flexible with regard to the type of simulator used, any traffic simulator could have been used here. The only requirement is that it must be able to accept the signal timing plan generated by our algorithm as input, and output necessary information to our solver, as described below.

C. SFP with simulation-based best reply computation

A crucial step in implementing SFP is the computation of best replies in line 5 of Algorithm 1. Since for the coordinated signal control problem the objective function can only be evaluated through the execution of the traffic simulator, the only way to accurately compute each player's best reply is by pure enumeration of all player's strategies. In a problem with I intersections and N time periods, best reply computations for all players would generally require $(N \sum_{i=1}^{I} S_i)$ simulations.

In practice the number of simulations can be decreased somewhat by observing the following facts:

1) In line 4 of Algorithm 1, a joint strategy $\mathbf{D}$ is sampled. One can evaluate this strategy (using the simulator) and pass the resulting objective function value as a parameter to the best reply function. Recall that, for each player, best reply is obtained by comparing the objective function values of the sampled joint strategy and the joint strategies obtained by substituting this player's strategy with other elements of its strategy set. Since the value of the former is provided to the best reply subroutine, $(N \cdot I)$ simulations can be saved.
2) Given a sampled joint strategy $\mathbf{D}$, there may exist some intersections/time periods when there is only light traffic waiting to pass through. Since the performances of all strategies of the corresponding players are likely to be very close, best reply computations (and hence calls to the simulator) can be skipped for those players. We can define a threshold α, and calculate a best reply for a player by invoking the simulator only if combined traffic volume in the time period is greater than α.

(In our experiments, we used $\alpha = 0$, skipping best reply computations only when no traffic was traveling through the intersection in a time period.) When the traffic volume is less than or equal to α, the best reply of this player can be essentially selected arbitrarily. To increase the exploration of the joint strategy space, we drew a random strategy uniformly from the player's strategy set in this case.

To take advantage of the second observation, in addition to the objective function value (i.e., average travel time), we need information on the traffic volume at each intersection during each time period, obtained from time-dependent traffic statistics for the sampled strategy. Since this information only needs to be obtained in the beginning of each iteration, we distinguish between executing INTEGRATION-UM in two different modes: mode MAX, where both average travel time and the time-dependent traffic statistics are outputted, and mode MIN, where only average travel time is outputted. (The latter mode is much less time consuming than the former.)

SFP algorithm for the coordinated signal control problem with simulation-based best reply computation scheme described as above will be called **CoSIGN** and used throughout the paper. The stopping criterion used in **CoSIGN** is the number of SFP iterations.

Algorithm 2 Simulation-based best reply function

B=BESTREPLY(**D**)
1: $(v, \mathbf{F}) \leftarrow$ INTEGRATION-UM$_{\text{MAX}}$(**D**)
2: **for all** $i \in$ **I do**
3: **for all** $n \in$ **N do**
4: **if** $\mathbf{F}(i,n) \geq \alpha$ **then**
5: $v_{\min} \leftarrow v$
6: $\mathbf{B}(i,n) \leftarrow \mathbf{D}(i,n)$
7: $\mathbf{D}' \leftarrow \mathbf{D}$
8: **for all** $s \in \mathbf{S}_i$, $s \neq \mathbf{D}(i,n)$ **do**
9: $\mathbf{D}'(i,n) \leftarrow s$
10: $v_s \leftarrow$ INTEGRATION-UM$_{\text{MIN}}$(**D**$'$)
11: **if** $v_s < v_{\min}$ **then**
12: $v_{\min} \leftarrow v_s$
13: $\mathbf{B}(i,n) \leftarrow s$
14: **end if**
15: **end for**
16: **else**
17: $\mathbf{B}(i,n) \leftarrow$RANDOM(**S**$_i$)
18: **end if**
19: **end for**
20: **end for**
21: **return B**

The pseudo-code for the simulation-based best reply function is listed in Algorithm 2. Below is the list of functions used in Algorithm 2 (here **D** denotes a joint strategy):

- INTEGRATION-UM$_{\text{MIN}}$(**D**): the function runs the simulation and returns the objective function value.
- INTEGRATION-UM$_{\text{MAX}}$(**D**): the function runs the simulation and returns the objective function value and time-dependent traffic statistics. The objective function value is stored in v, while the time-dependent traffic statistics data are stored in **F**, a matrix where $\mathbf{F}(i,n)$ represents traffic volume at intersection i during time period n.
- RANDOM(**S**$_i$): the function uniformly picks an element from **S**$_i$ and returns it.

The pseudo-code in Algorithm 2 implements the ideas discussed earlier. A common evaluation of the simulator in MAX mode is performed in line 1. For each player, if the traffic volume is below the threshold α (as checked in line 4), a phase of the corresponding signal is randomly selected in line 17. Otherwise, the algorithm loops through and evaluates all phases of the signal (except the phase used in **D**, which is already evaluated), starting in line 8.

Notice that whenever the simulator is executed in either MIN or MAX modes, we will be able to read the performance measures and therefore update the incumbent pure strategy. This best pure strategy will be delivered as the solution at the end of the algorithm execution, as described in section IV-B.

VI. CASE STUDY: TROY, MICHIGAN, NETWORK

In order to test performance of the CoSIGN algorithm, we used a realistic traffic network model built by Wunderlich [38], [39], [37]. This case study model has been constructed based on the real traffic network of Troy, Michigan, and, to ensure fidelity, carefully calibrated against empirical measurements. To maintain this fidelity, we did not modify the model in any way except to insert the signal timing plans we generated. A map snapshot of the Troy network is shown in Fig. 1. The corresponding model of the network topology is shown in Fig. 2. Here are the parameters used in our experiments:

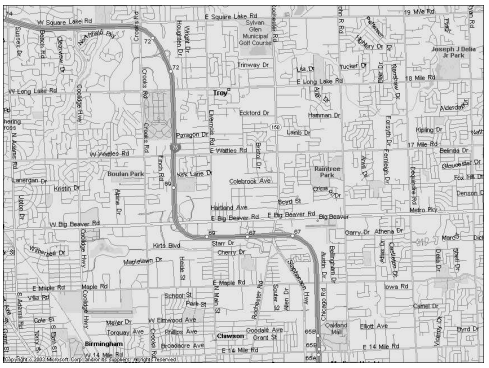

Fig. 1. The snapshot of Troy's area map.

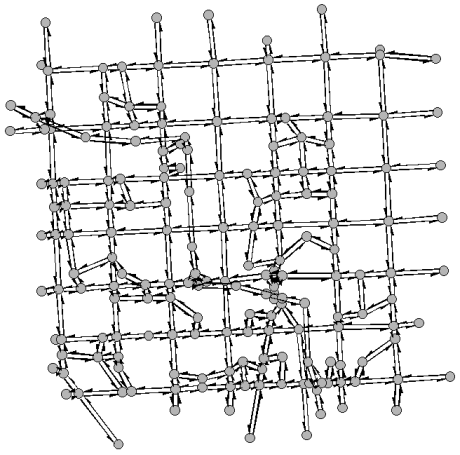

Fig. 2. The Troy network topology model, composed of 529 links, 200 nodes and 72 zone centroids that can serve as origins or destinations.

- Length of the time period: $\delta = 10$ seconds
- Number of time periods: $N = 720$
- Number of signalized intersections: $I = 75$
- Number of players: $P = N \cdot I = 54{,}000$
- Stopping criterion: 20 iterations of CoSIGN are executed

The original cyclic pattern of traffic signals embedded in the model was used as the initial solution. We assumed that all vehicles will follow fastest free-flow paths[5] from their origins to destinations.

A. Competing Timing Plans and Algorithms

The goals of this section are twofold: to demonstrate the potential benefits of coordinated traffic signal control using predictive traffic information (as discussed in the Introduction), as well as evaluate the effectiveness of our algorithmic approach, the CoSIGN algorithm, for this task. Towards these goals, we compared CoSIGN to the following alternatives:

- **Static**: fixed cyclic signal timing plans were supplied by the city of Troy and embedded in the original model. When implemented, these signal timing plans were defined by cycle time, offsets, and phase splits. Since real-time signal plan optimization was not available in Troy at the time the model was built, these plans are kept constant throughout the planning horizon.
- **Automatic Signal Re-timing (ASR)**: although real-time signal timing plan optimization was not available in Troy when the model was constructed, the INTEGRATION-UM simulator provides an automatic cycle and phase split optimization tool, which can be used to evaluate the potential impact of such schemes. When the tool is turned on, cycle lengths and green splits at all signals are recalculated at user-specified intervals, using current traffic volume information. For detailed description of this algorithm, refer to Appendix.

Since static and ASR timing plans control each signal in isolation, the benefits of coordinated signal control can be demonstrated by comparing CoSIGN to static and ASR control schemes. This comparison is conducted in section VI-B.

- **Coordinate Descent (CD)**: a straightforward way to solve a discrete optimization problem of the form (1) is to start with some initial solution, loop through all variables (i.e., coordinates) one by one, and solve each single-variable problem while keeping the values of all other variables fixed. The result from the single-coordinate optimization is used to update the current solution. The process stops when a solution cannot be further improved after looping through all variables. In our setting, CD can be formally implemented as follows (here $\mathbf{D}^k$ denotes the joint strategy at iteration k, $(s_p, \mathbf{D}^k_{-p})$ denotes the same joint strategy with the strategy of player p replaced by s_p, and the subroutine BESTREPLY_p evaluates the best reply strategy for player p only):

  The stopping criterion in line 3 of CD is based on the number of consecutive non-improving iterations, u. If $u = P$ (recall that P is the number of variables in this problem), the objective function value cannot be improved after looping through all P variables, and thus we stop.

[5]The fastest free-flow paths are computed with the assumption that free-flow speeds prevail on all links over the planning horizon.

Algorithm 3 Coordinate Decent (CD) algorithm.

```
CD()
 1: D^0 ← INITIALSOLUTION()
 2: k ← 0, p ← 1, u ← 1
 3: while u < P do
 4:     ŝ_p ← BESTREPLY_p(D^k)
 5:     D^{k+1} ← (ŝ_p, D^k_{-p})
 6:     if D^{k+1} = D^k then
 7:         u = u + 1
 8:     else
 9:         u = 1
10:     end if
11:     k ← k + 1, p ← (p mod P) + 1
12: end while
```

The CD algorithm by construction considers coordinated signal timing plans, thus we also expect it to enjoy the benefits of coordination, as CoSIGN does. However, CD is a "serial" algorithm in that it considers the variables sequentially, with the output of one single-variable optimization serving as an input into the next one. In a real traffic network (like the Troy network), where the number of variables is large and the time required to invoke a single simulation is non-negligible, the time required to obtain any significant improvement through running CD algorithm may be prohibitively long. To demonstrate the benefits of parallelization, we will explore the possibility of parallel execution of CoSIGN and compare it to CD in subsections VI-C and VI-D.

B. Benefits of signal coordination and predictive information

Results of experiments comparing CoSIGN to the static and ASR signal timing plans can be seen in Table I. The performance measure is the average travel time experienced by all drivers in the traffic network, evaluated by INTEGRATION-UM. For the *normal-flow case* taken from Wunderlich's model, around 26,000 vehicles were allowed to flow into the network from the beginning of the simulation to the 24^{th} minute mark. This traffic volume, as well as the flow patters used in our experiments, are consistent with the traffic patterns observed in Troy at the time the model was constructed. After the inflow was stopped, the simulator was allowed to run an additional 96 minutes in order to clear all traffic. To evaluate performance under different traffic conditions, we created two similar scenarios, *light-flow case* and *heavy-flow case*, where the same traffic flow pattern and time horizon were used, but the flow rate was decreased (increased) by 50%, so that approximately 13,000 (39,000) vehicles were allowed to flow into the network.

Note that as depicted in line 4 of Algorithm 1, a random sample is drawn from the history during the beginning of each iteration. This randomness makes CoSIGN a stochastic algorithm. Therefore, to assess performance of CoSIGN, we report summary statistics (mean, best and worst values) of solutions found by 15 independent runs of CoSIGN on each problem instance. Although there is some variability in quality

TABLE I

PERFORMANCE OF THREE COMPETING ALGORITHMS [a]

		Avg. travel time (min.)		
		Light flow	Normal flow	Heavy flow
Static		10.1 (+13%)	19.4 (+29%) [c]	43.8 (+58%)
Best ASR		9.4 (+5%)	17.2 (+14%)	38.2 (+38%)
CoSIGN [b]	Best	8.8	14.9	25.9
	Mean	8.9	15.1	27.6
	Worst	9.0	15.3	29.8

[a] Average travel times are used for performance comparison purpose.
[b] Fifteen independent CoSIGN runs are executed in all flow scenarios, and best, mean and worst times are obtained accordingly.
[c] The number in each cell is corresponding average travel time (in minutes) for that case. The percentages listed in row "Static" and "Best ASR" are margins computed with "CoSIGN — Mean" as base. For example, +29% in Static-Normal flow cell means that the average travel time of static timing plan, under normal flow, is 29% more than that of CoSIGN on average.

of obtained solutions, stemming from the stochastic nature of the algorithm, CoSIGN finds a signal plan that significantly improves on the starting solution in each instance.

Table I compares average travel times of signal plans found by multiple CoSIGN executions to that of a static signal plan and the one found by ASR. From Table I we can see that the plans found by CoSIGN (both on average and even in the worst case) perform better than the other two, under all flow conditions, and the margin of advantage increases as flow gets heavier. Since the static signal timing plan is not adaptive to traffic conditions, this result is to be expected. As for the ASR algorithm, although it is responsive to the real-time traffic condition, its underlying assumption is that the network is undersaturated, and this condition is more likely to be violated in the heavy-flow case than in the light-flow and normal-flow case. This leads to relative deterioration of performance of the ASR approach in the heavy-flow case.

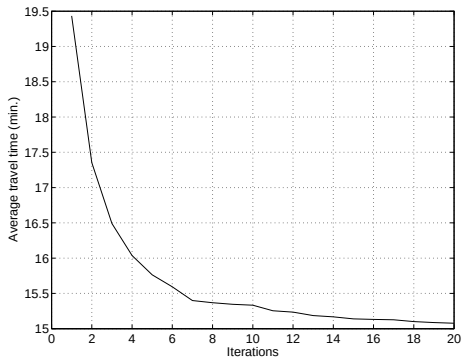

Fig. 3. The evolution of best values as a function of iteration count for the normal-flow case.

It should also be noted that in the ASR implementation within INTEGRATION-UM, the interval between signal re-timings is a user-specified parameter. Our experiments with various settings of this parameter demonstrated its critical importance to the performance of ASR. Results reported in Table I reflect the performance of ASR with the re-timing interval that was empirically found to be the best for each experiment. (These "best" intervals had different lengths under different traffic conditions, and we found no discernible pattern of dependence of the method's performance on the interval length; e.g., more frequent re-timings did not necessarily lead to improvements.) In other words, the reported margin of CoSIGN over ASR is a conservative bound, and in practice, with re-timing intervals determined mostly ad hoc, this margin will be much larger. In Fig. 3, we plot the evolutions of mean best value (average travel time of current incumbent solution) versus iteration number for the normal-flow case. Similar evolutions are drawn for the light-flow and heavy-flow cases in Fig. 4 and Fig. 5 respectively. Fig. 3, 4 and 5 motivate our choice of terminating CoSIGN after 20 iterations: most of the improvements were achieved within the first 10 iterations, and improvements around 20^{th} iteration were small.

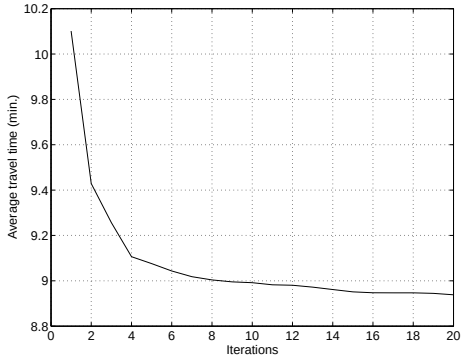

Fig. 4. The evolution of best values as a function of iteration count for the light-flow case.

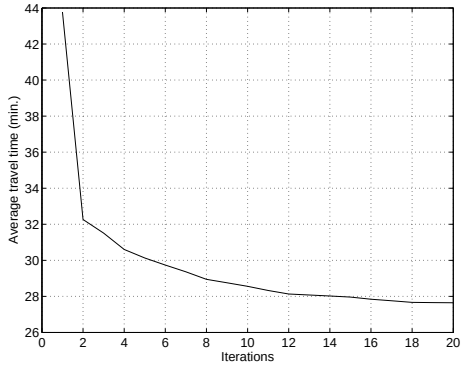

Fig. 5. The evolution of best values as a function of iteration count for the heavy-flow case.

Another interesting statistic we observe in these computational experiments is the average travel time experienced by drivers leaving their origins at different times. For all three flow scenarios, we consider 24 groups of vehicles, grouped according to their departure times, where the i^{th} group contains vehicles departing within the i^{th} minute. For each such group, the average travel time of all vehicles in the group is then plotted as a data point. In Figs. 6, 7, and 8, average travel times of each group for each control scheme are plotted against all possible departure minutes $(1, 2, \ldots, 24)$. From these figures we can conclude that as flow grows heavier, CoSIGN performs relatively better than the two alternatives.

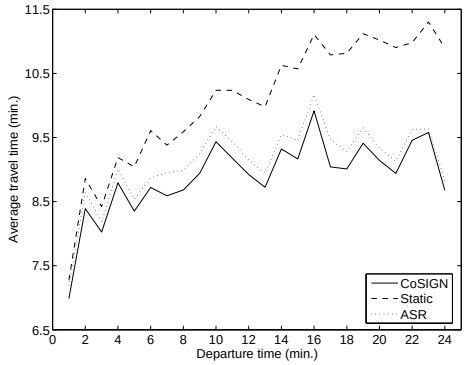

Fig. 6. Average travel time as a function of vehicles' departure times, for the light-flow case.

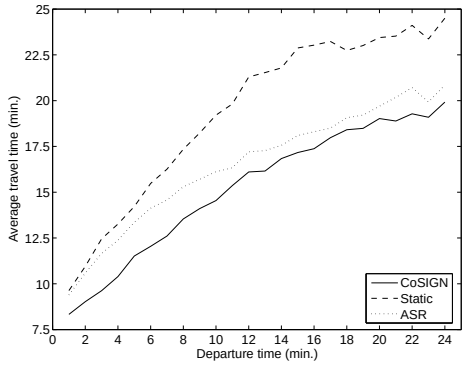

Fig. 7. Average travel time as a function of vehicles' departure times, for the normal-flow case.

C. Parallelized implementation of CoSIGN

We have demonstrated the benefits of a coordinated signal control algorithm that takes into account predictive traffic information in the previous subsection. However, another important consideration is the time required to execute such an algorithm. In a straightforward serial implementation on a Pentium-4 2.8GHz PC with 1GB RAM, running RedHat Linux, 20 iterations of CoSIGN took 169.04 hours for the normal-flow case, and 397.6 hours for the heavy-flow case.

Since CoSIGN is expected to be responsive to current traffic conditions and forecasts, its execution time should be short enough to fit into the desired update interval. One way to significantly reduce the "wall-clock" running time without sacrificing the precision or scope of the solution is through parallelization. In this subsection we will describe how to parallelize CoSIGN and discuss the impact that degree of parallelization has on the running time of the algorithm.

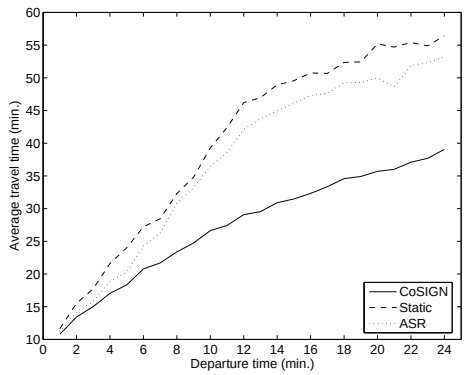

Fig. 8. Average travel time as a function of vehicles' departure times, for the heavy-flow case.

As mentioned earlier, computation between line 2 and line 17 in Algorithm 2 can be parallelized. With K identical CPUs available, we can divide the best reply evaluations for all players into K tasks, and assign each task to a CPU. Each task will take the sampled joint strategy, $\mathbf{D}$, its associated objective value, v, and the set of players, $\mathbf{P}_j$, as input parameters. The output of each task will be the best replies, $\mathbf{B}_j$, for players in $\mathbf{P}_j$. Note that since $\bigcup_{j=1}^{K} \mathbf{P}_j = \mathbf{P}$, we have $\bigcup_{j=1}^{K} \mathbf{B}_j = \mathbf{B}$. Regardless of the degree of parallelization, as long as samples drawn in line 4 of Algorithm 1 and in line 17 of Algorithm 2 remain the same, CoSIGN will evaluate the same set of solutions and return the same output.

In order to asses the impact of parallelization without resorting to repeatedly re-running CoSIGN on clusters of CPUs of various sizes, we instead analytically relate the running time of CoSIGN to the degree of parallelization, and rely on a single run of CoSIGN to make performances estimates.

We will use the following notation:

S_{MAX}:	time required to execute INTEGRATION-UM$_{\text{MAX}}(\cdot)$
S_{MIN}:	time required to execute INTEGRATION-UM$_{\text{MIN}}(\cdot)$
P:	number of players
N_{CoSIGN}:	number of CoSIGN iterations executed ($N_{\text{CoSIGN}} = 20$ in our implementation)
K:	number of available CPUs

In our calculations we neglect time spent on communications between CPUs and samplings in the implementation of CoSIGN since the time spent on simulations dominates total execution time. Also, we assume that at every iteration, K tasks for best reply evaluation are created in a balanced manner, i.e., they require approximately equal time for execution.

In BESTREPLY function, one call to INTEGRATION-UM$_{\text{MAX}}(\cdot)$ and at most $(N\sum_{i=1}^{I}(S_i - 1))$ calls to INTEGRATION-UM$_{\text{MIN}}(\cdot)$ will be made. Let P_T be the number of calls made to INTEGRATION-UM$_{\text{MIN}}(\cdot)$ in one iteration. The wall-clock running time of BESTREPLY function with K CPUs utilized as described above is bounded above by

$$T_{BR} \leq S_{\text{MAX}} + \left\lceil \frac{P_T}{K} \right\rceil S_{\text{MIN}} \tag{2}$$

(this is an upper bound since, as discussed in section V-C, best reply computations are skipped for some of the players). Therefore, the total wall-clock running time of N_{CoSIGN} iterations of CoSIGN will be

$$\begin{aligned} T(K) &= N_{\text{CoSIGN}} \cdot T_{BR} \\ &\leq N_{\text{CoSIGN}} \left(S_{\text{MAX}} + \left\lceil \frac{P_T}{K} \right\rceil S_{\text{MIN}} \right). \end{aligned} \tag{3}$$

To obtain a tighter bound, let P_s be the average number of simulations actually used per iteration, after we consider the savings described in subsection V-C; we can then replace (3)

with

$$T(K) = N_{\text{CoSIGN}} \left(S_{\text{MAX}} + \left\lceil \frac{P_s}{K} \right\rceil S_{\text{MIN}} \right) \approx N_{\text{CoSIGN}} \left\lceil \frac{P_s}{K} \right\rceil S_{\text{MIN}}. \quad (4)$$

In the Troy test case with normal traffic flows, we observed during a typical run of CoSIGN (with $N_{\text{CoSIGN}} = 20$) $S_{\text{MIN}} = 1.3$ seconds and $P_s = 21{,}582$ (note that this is about a 60% reduction in the number of simulations). Hence (4) becomes:

$$T(K) \leq 20 \left\lceil \frac{21{,}582}{K} \right\rceil 1.3 \text{ sec.} = 20 \left\lceil \frac{21{,}582}{K} \right\rceil \frac{1.3}{60} \text{ min.} \quad (5)$$

For instance, for $K = 134$, 70 minutes of wall-clock

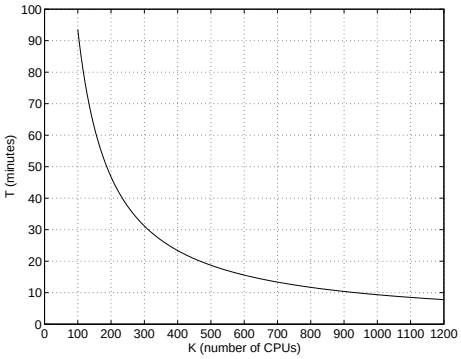

Fig. 9. Running time of CoSIGN versus degree of parallelization K.

computation time will be needed to execute CoSIGN. For $K = 256$, the required time is 37 minutes, and for $K = 1024$ — just 9 minutes. We chose these illustrative values of K since such computational facilities are readily available at educational institutions such as the University of Michigan and University of Texas. To give the reader a broader sense of the impact that different degrees of parallelization have on the wall-clock time required by CoSIGN, we plotted (5) in Fig. 9.

To demonstrate that parallelization is indeed feasible, we implemented a parallel version of CoSIGN on cluster systems managed by the Center for Advanced Computing[6] at the University of Michigan. The specifications of the cluster systems are as follows:

- morpheus: the 208 processor Athlon cluster is composed of 17 nodes of dual Athlon 1600MP CPUs, 29 nodes of dual Athlon 2400MP CPUs, and 58 nodes of dual Athlon 2600MP CPUs.
- nyx: the 450 processor Opteron cluster is composed of 225 nodes of dual Opterons, ranging from Opteron 240s (@ 1400 MHz) to Opteron 244s (@ 1800 MHz).

In our experiments, the typical number of processors used was either 8, 16, or 32, due to the job scheduling policy at the Center.

Note that these systems are equipped with CPUs slower than the one we have run our serial experiment on, therefore the curve in Fig. 9 is not directly applicable. However, a corresponding plot for running time versus degree of parallelization can be easily reconstructed by measuring S_{MIN} on each system.

[6]http://cac.engin.umich.edu

One of the main assumptions in our derivation is that the time spent on communication can be neglected. We verified this assumption by looking at the timing analysis from our parallel experiments. We observed that in all cases, the percentage of time spent on communication is less than 0.005%. Therefore, at least in our current experiments, the communication time is indeed negligible.

D. Relative performance of parallelized CoSIGN vs. Coordinate Descent

As noted in prior sections, CoSIGN is a heuristic that searches for an optimal solution to the coordinated traffic signal control problem. Although we have empirically shown the algorithm's benefits based on a realistic test case, the solution found in 20 iterations is not guaranteed to be an optimal solution to the problem, even in the local sense. In fact, while the average vehicle travel time in the normal flow case was 15.60 minutes under the signal plan found by CoSIGN, the Coordinate Descent (CD) algorithm described in subsection VI-A, given sufficient time, found a plan with average time of 13.13 minutes. It should be noted, however, that it took CD 362,500 iterations over several days of running time to identify this solution.

A meaningful way to compare practical performance of any two heuristic algorithms, such as CoSIGN and CD, on a problem is to compare the objective values of solutions they find given the same amount of wall-clock time. As we demonstrate in this section, as the number of processors made available to CoSIGN increases, its wall-clock running time decreases, and the quality of solutions found by CD in the same time deteriorates dramatically.

As in the previous subsection, we do not resort to multiple algorithm runs, but rather use analytical estimates of running times of CD and CoSIGN to perform the comparison.

Recall that the CD algorithm is initialized with some initial solution, and in each step afterwards, uses a simulation to evaluate the current player's alternative decision. In each of these steps, the solution will be modified if the current player's alternative decision improves the solution. As this process suggests, the CD algorithm cannot be parallelized and must be executed serially. Therefore, the wall-clock time required to execute N_{CD} iterations of CD is

$$(N_{\text{CD}} + 1) S_{\text{MIN}}. \quad (6)$$

(We did not invoke the threshold test to bypass potentially unnecessary simulations in CD since that would require running INTEGRATION-UM$_{\text{MAX}}$ at every iteration. Since S_{MAX} exceeds S_{MIN} by 50% to 150%, depending on the number of vehicles in the network, the added computational effort would outweigh potential savings.)

Let $N_{\text{CD}}(K)$ denote the number of iterations CD would be able to perform if it were allowed the same amount of wall-clock time as it takes to execute N_{CoSIGN} iterations of the parallelized CoSIGN algorithm running on a cluster of K processors, i.e., $T(K)$. Setting $(N_{\text{CD}}(K) + 1) S_{\text{MIN}} = T(K)$

and using the formulas above, we obtain:

$$\begin{aligned} N_{\text{CD}}(K) &\leq \frac{N_{\text{CoSIGN}}\left(S_{\text{MAX}} + \lceil P_T/K \rceil \cdot S_{\text{MIN}}\right)}{S_{\text{MIN}}} - 1 \\ &= N_{\text{CoSIGN}}\left(\frac{S_{\text{MAX}}}{S_{\text{MIN}}} + \left\lceil \frac{P_T}{K} \right\rceil\right) - 1. \end{aligned} \quad (7)$$

(Recall that $P_T = N\sum_{i=1}^{I}(S_i - 1)$.) Once again, if P_s is the actual average number of simulations used per iteration by CoSIGN, we can obtain a tighter bound:

$$N_{\text{CD}}(K) \leq N_{\text{CoSIGN}}\left(\frac{S_{\text{MAX}}}{S_{\text{MIN}}} + \left\lceil \frac{P_s}{K} \right\rceil\right) - 1. \quad (8)$$

In the Troy test case with normal traffic flows, $N_{\text{CoSIGN}} = 20$, $P_s = 21{,}582$, and the numeric form of (8) becomes:

$$\begin{aligned} N_{\text{CD}}(K) &\leq 20\left(\frac{S_{\text{MAX}}}{S_{\text{MIN}}} + \left\lceil \frac{21{,}582}{K} \right\rceil\right) - 1 \\ &\approx 20\left\lceil \frac{21{,}582}{K} \right\rceil. \end{aligned} \quad (9)$$

The number of iterations CD will be able to complete in the same amount of wall-clock time as CoSIGN is inversely proportional to the number of processors available to CoSIGN.

As mentioned in the beginning of the section, we did perform one multi-day run of CD for the normal flow scenario in the Troy network. We can now compare the performance of the algorithms as follows: for a particular value of K, we estimate $N_{\text{CD}}(K)$ based on (9) and consult the output of the CD run to obtain the average travel time for the signal plan found by CD in $N_{\text{CD}}(K)$ iterations. The resulting comparison is presented in Fig. 10, where we plot the average travel time of solutions found by CD in $N_{\text{CD}}(K)$ iterations versus K for the normal-flow case. A similar graph for the heavy-flow case is plotted in Fig. 11. (These graphs may appear a bit counterintuitive at first, as the increase in the number of CPUs results in worse objective function values found. To interpret these graphs, recall that addition of CPUs decreases the amount of wall-clock time allotted to CD, allowing for fewer iterations and less progress.) For comparison, the average travel times of 15.08 minutes (for the normal flow case) and 27.62 minutes (for the heavy flow case) obtained by CoSIGN are also plotted on the same graph. (Recall that these are the mean performance measures of solutions found by several runs of CoSIGN on each problem instance.)

As Fig. 10 indicates, CD underperforms CoSIGN in this comparison if the latter is allowed 26 CPUs or more. Moreover, if CPUs number in the hundreds, CD makes almost no progress from the initial solution in the time it takes CoSIGN to complete its run. Similar result can be observed in Fig. 11, where CD underperforms CoSIGN in this comparison if the latter is allowed 16 CPUs or more.

Even though in the long (very long!) run CD found a better solution than CoSIGN, since wall-clock times available in practice are limited, the parallelized CoSIGN algorithm will always be superior to CD in practice. Since CD is an inherently sequential algorithm, multiple available CPUs can be utilized by running CD for the specified number of iterations starting at different initial solutions on each CPU and reporting the best solution found. However, based on our empirical experience,

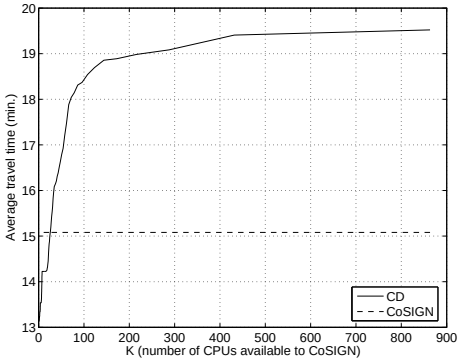

Fig. 10. Average travel time of solution found by CD when given the same wall-clock time as the parallel execution of CoSIGN with K processors, vs. K. for the normal-flow case

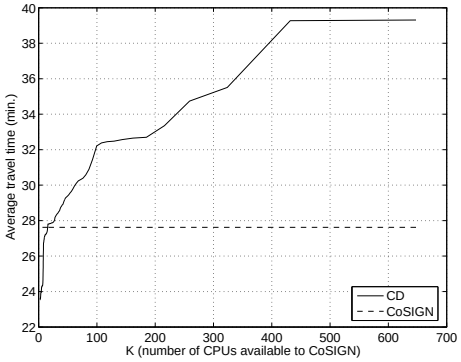

Fig. 11. Average travel time of solution found by CD when given the same wall-clock time as the parallel execution of CoSIGN with K processors, vs. K. for the heavy-flow case

CD makes very slow progress in each iteration. Therefore, it will not in fact achieve significant improvement over the starting points it is provided.

VII. Future Work

A natural extension of the paper is to test CoSIGN on other even larger and more detailed traffic networks. The use of more advanced traffic simulators may also be desirable in modeling more complicated traffic characteristics. Also, in some cases, we may want to reduce the length of control intervals in order to better emulate real-world scenarios. All these factors, when combined together, will make an already challenging problem even more so. To reduce computational requirement, we can replace the full-blown simulations with simplified ones (e.g., see [40]) in best reply evaluations. Since, in evaluating best replies, what we really care about is the relative superiority of a single player's strategy selections; a simplified simulation that can accurately provide this relative performance comparison will be good enough. Of course, in order to design good approximated best replies, a deep understanding of the problem structure is required (as demonstrated in [25]). This issue is critical, since being able to complete best reply evaluations quickly is key to the implementation of real-time control.

Another interesting extension is to evaluate the robustness of the signal timing plans obtained by CoSIGN in the presence

of stochastic traffic flows. For a small traffic network, the reliability of traffic signal timing plans can be derived analytically, e.g., see [41]. In our case, analytical derivation is not possible due to the size of the problem, therefore we should seek other indicators, e.g., the variance of vehicle travel times. This type of analysis will be very useful in determining the effectiveness of our approach in the face of stochasticity.

Another way to deal with the stochasticity in traffic flows is to adopt a rolling-horizon type of implementation (e.g., see [21], [42]). Each time we observe a change in the traffic flow pattern (either measured directly, or inferred indirectly), we can use the latest information to update the model and rerun the CoSIGN algorithm.

Finally, to more accurately capture the operating conditions of real traffic networks, we should introduce feasibility constraints to our model (e.g., minimal or maximal green time continuously given to a phase). However, if constraints are introduced to our model, some sampled joint decisions may become infeasible, and this requires special treatment. Our current conjecture is that such difficulty can be handled by defining proper repair rules.

Our ultimate goal is to design an algorithm that is capable of finding a robust, scalable, and responsive coordinated traffic signal timing plan in a large-scale traffic network. The framework introduced in this paper provides the foundations for developing such a system. However, to move closer to our ultimate goal, we should incrementally incorporate the improvements discussed here.

VIII. Acknowledgements

This work was partially supported by the National Science Foundation under Grants DMI-0217283 and DMI-0422752. The authors gratefully acknowledge Karl Wunderlich's valuable suggestions and advice. Also, the authors would like to thank Michael Wellman and anonymous referees for their detailed comments and suggestions.

Appendix
Automatic Signal Re-timing

Automatic signal re-timing in INTEGRATION-UM is an online cycle time and phase-split optimization heuristic, as described in Wunderlich [37]. The underlying theory for this approach is based on Webster and Cobbe's model [1]. Underlying analysis will not be explained in detail here; instead, the implementation of the algorithm as embedded in INTEGRATION-UM is presented.

The automatic signal re-timing algorithm determines signal timing plans based on current flows on the approaches[7] leading to the signalized intersections. (In this appendix we use the term "flow" to represent the volume of traffic on a link or approach.) The re-timing algorithm in INTEGRATION-UM is invoked repeatedly at user-specified intervals, and proceeds in three steps:

[7]If a signal timing plan is used at more than one intersection within the traffic network, the approach is defined as the set of links coming into these controlled intersections during the same phase.

1) **Estimating link flows**: for each signalized intersection, the equivalent flow for each link is estimated by combing average incoming flow and average size of the standing queue. The following formula is used for this purpose:

$$v^a = f^a + 4q^a, \tag{10}$$

where v^a is the estimated flow on link a, f^a is the exponentially smoothed average flow on link a, and q^a is the exponentially smoothed average size of the standing queue on link a.
Both average incoming flow (f^a) and average size of the standing queue (q^a) of link a are obtained by periodically performing the following exponential smoothing updates:

$$f^a := 0.75f^a + 0.25f^a_{\text{in}} \tag{11}$$
$$q^a := 0.9q^a + 0.1\hat{q}^a, \tag{12}$$

where f^a_{in} is the number of vehicles flowing into link a during the interval between smoothing updates, and $\hat{q}^a$ is the size of standing queue on link a during the same interval.

2) **Computing critical values**: based on the above flow data, the procedure will compute a measure (i.e., critical value) that represents the relative congestion of each link. By using this measure, the procedure then computes cycle length and the allocation of green times.
For each link a leading to the intersections controlled by the signal timing plan, a critical value (measure of congestion) y^a is computed as the ratio between estimated link flow and link's saturation flow:

$$y^a = \frac{v^a}{s^a}, \tag{13}$$

where s^a is link a's saturation flow rate (as defined in the network topology definition).
Let the set A_p consist of all the links that have the right of way during phase p of the signal under consideration. The critical value for phase p is then the maximal y^a of all links in A_p:

$$y_p = \max\left\{\max_{a\in A_p}\{y^a\}, y_{\text{min}}\right\}, \tag{14}$$

where y_{min} is a predefined minimal critical value.
The combined critical value for the signal timing plan, denoted by Y, is then the sum of values of y_p over all its phases:

$$Y = \sum_p y_p. \tag{15}$$

3) **Computing cycle time and green time for each phase**: the new cycle time for each signal timing plan, C_o, is computed from its corresponding critical value, Y, and the sum of lost time (i.e., yellow time) for all phases, L. For $Y \leq 0.95$,

$$C_o = \max\{\min\{\frac{(1.5L+5)}{(1-Y)}, C_{\max}\}, C_{\min}\} \tag{16}$$

Otherwise, $C_o = C_{\text{max}}$. $C_{\min}$ and $C_{\max}$ are the specified minimal and maximal cycle times, respectively.

After C_o is obtained, the length of green time for all phases can be computed accordingly. g_p, the length of green time assigned to phase p, is determined by

$$g_p = \frac{y_p}{Y}(C_o - L). \tag{17}$$

REFERENCES

[1] F. V. Webster and B. M. Cobbe, *Traffic Signals*, Road Research Technical Report 39, Her Majesty's Stationery Office, London, 1958.

[2] R. Allsop, "SIGSET: A computer program for calculating traffic capacity of signal-controlled road junctions," *Traffic Eng. & Control*, vol. 13, pp. 58–60, 1971.

[3] ——, "SIGCAP: A computer program for assessing the traffic capacity of signal-controlled road junctions," *Traffic Eng. & Control*, vol. 17, pp. 338–341, 1976.

[4] J. D. C. Little, "The synchronization of traffic signals by mixed-integer linear programming," *Oper. Res.*, vol. 14, no. 4, pp. 568–594, 1966.

[5] J. D. C. Little, M. D. Kelson, and N. H. Gartner, "MAXBAND: A program for setting signals on arteries and triangular networks," *Transportation Research Record*, vol. 795, 1981.

[6] D. I. Robertson, "TRANSYT method for area traffic control," *Traffic Eng. & Control*, vol. 10, pp. 276–281, 1969.

[7] A. J. Miller, "A computer control system for traffic networks," in *2nd Int. Symp. on the Theory of Road Traffic Flow*, London, 1965, pp. 200–220.

[8] A. G. Sims, "The Sydney coordinated adaptive traffic system," in *Urban Transport Division of ASCE Proc.*, New York, NY, 1979, pp. 12–27.

[9] J. J. Henry, J. L. Farges, and J. Tuffal, "The PRODYN real time traffic algorithm," in *4th IFAC/IFIP/IFORS Conf. on Control in Transp. Systems*, 1983, pp. 305–310.

[10] J. J. Henry and J. L. Farges, "PRODYN," in *6th IFAC/IFIP/IFORS Symp. on Control, Computers and Comm. in Transp.*, 1989, pp. 253–255.

[11] N. H. Gartner, "OPAC: A demand-responsive strategy for traffic signal control," *Transportation Research Record*, vol. 906, pp. 75–81, 1983.

[12] N. H. Gartner, F. J. Pooran, and C. M. Andrews, "Implementation of the opac adaptive control strategy in a traffic signal network," in *Proc. IEEE Intell. Transport. Syst. Conf.*, 2001, pp. 195–200.

[13] V. Mauro and D. DiTaranto, "UTOPIA," in *6th IFAC/IFIP/IFORS Symp. on Control, Computers and Comm. in Transp.*, 1989, pp. 245–252.

[14] S. Yagar and B. Han, "A procedure for real-time signal control that considers transit interference and priority," *Transp. Res. B*, vol. 28, no. 4, pp. 315–331, 1994.

[15] S. Sen and K. L. Head, "Controlled optimization of phases at an intersection," *Transp. Science*, vol. 31, pp. 5–17, 1997.

[16] P. B. Hunt, D. I. Robertson, R. D. Bretherton, and R. I. Winton, "SCOOT - a traffic responsive method for coordinating signals," in *Laboratory Report no. LP 1014*. Crowthorne, Berkshire, England: Transportation and Road Research, 1981.

[17] F. Boillot, J. Blosseville, J. Lesort, V. Motyka, M. Papageorgiou, and S. Sellam, "Optimal signal control of urban traffic networks," in *6th Int. Conf. on Road Traffic Monitoring and Control*. London, England: IEE, 1992, pp. 75–79.

[18] P. Dell'Olmo and P. B. Mirchandani, "REALBAND: An approach for real-time coordination of traffic flows on a network," *Transportation Research Record*, vol. 1494, pp. 106–116, 1995.

[19] W.-H. Lin and C. Wang, "An enhanced 0-1 mixed-integer LP formulation for traffic signal control," *IEEE Trans. Intell. Transport. Syst.*, vol. 5, no. 4, pp. 238–245, December 2004.

[20] T. H. Heung, T. K. Ho, and Y. F. Fung, "Coordinated road-junction traffic control by dynamic programming," *IEEE Trans. Intell. Transport. Syst.*, vol. 6, no. 3, pp. 341–350, September 2005.

[21] P. B. Mirchandani and L. Head, "A real-time traffic signal control system: Architecture, algorithms, and analysis," *Transp. Res. C*, vol. 9, no. 6, pp. 415–432, 2001.

[22] P. B. Mirchandani and F.-Y. Wang, "RHODES to intelligent transportation systems," *IEEE Intell. Syst.*, vol. 20, no. 1, pp. 10–15, 2005.

[23] K. Ashok and M. E. Ben-Akiva, "Alternative approaches for real-time estimation and prediction of time-dependent origin-destination flows," *Transp. Science*, vol. 34, no. 1, pp. 21–36, 2000.

[24] ——, "Estimation and prediction of time-dependent origin-destination flows a stockhastic mapping to path flows and link flows," *Transp. Science*, vol. 36, no. 2, pp. 184–198, 2002.

[25] A. Garcia, D. Reaume, and R. L. Smith, "Fictitious play for finding system optimal routings in dynamic traffic networks," *Transp. Res. B*, vol. 34, no. 2, pp. 146–157, February 2000.

[26] J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*, 2nd ed. Princeton University Press, 1947.

[27] J. G. Wardrop, "Some theoretical aspects of road traffic research," in *Proc. of Institute of Civil Engineers, Part II*, vol. 1, 1952, pp. 325–378.

[28] R. D. McKelvey and A. McLennan, "Computation of equilibria in finite games," in *Handbook of Computational Economics*. Elsevier, 1996, vol. 1.

[29] G. W. Brown, "Iterative solution of games by fictitious play," in *Activity Analysis of Production and Allocation*. John Wiley, New York, 1951, pp. 374–376.

[30] J. Robinson, "An iterative method of solving a game," *Annals of Mathematics*, vol. 54, pp. 296–301, 1951.

[31] D. Monderer and L. S. Shapley, "Fictitious play property for games with identical interests," *J. Econom. Theory*, vol. 68, no. 1, pp. 258–265, 1996.

[32] T. J. Lambert, M. A. Epelman, and R. L. Smith, "A fictitious play approach to large-scale optimization," *Oper. Res.*, vol. 53, no. 3, pp. 477–489, May-June 2005.

[33] J. F. Nash, "Equilibrium points in n-person games," in *Proc. National Academy of Sciences*, vol. 36, 1950, pp. 48–49.

[34] D. Fudenberg and J. Tirole, *Game Theory*. MIT Press, 1991.

[35] T. J. Lambert and H. Wang, "Fictitious play approach to a mobile unit situation awareness problem," Univ. Michigan, Tech. Rep., 2003.

[36] M. Van Aerde, J. Voss, and G. McKinnon, *INTEGRATION Simulation Model User's Guide*, Queen's Univ., 1989.

[37] K. E. Wunderlich, "Link travel time prediction for dynamic route guidance in vehicular traffic networks," Ph.D. dissertation, Univ. Michigan, 1994.

[38] K. E. Wunderlich, D. E. Kaufman, and R. L. Smith, "Link travel time prediction for decentralized route guidance architectures," *IEEE Trans. Intell. Transport. Syst.*, vol. 1, no. 1, pp. 4–14, March 2000.

[39] K. E. Wunderlich and R. L. Smith., "Large scale traffic modeling for route guidance evaluation: A case study," Univ. Michigan, IVHS Program Technical Report 92-08, 1992.

[40] P. Dell'Olmo and P. B. Mirchandani, "A model for real-time traffic coordination using simulation based optimization," in *Advanced Methods in Transportation Analysis*, L. Bianco and P. Toth, Eds. Springer, 1996, pp. 525–546.

[41] H. K. Lo, "A reliability framework for traffic signal control," *IEEE Trans. Intell. Transport. Syst.*, vol. 7, no. 2, pp. 250–260, June 2006.

[42] F. Busch and G. Kruse, "MOTION for SITRAFFIC - a modern approach to urban traffic control," in *Proc. IEEE Intell. Transport. Syst. Conf.*, 2001, pp. 61–64.