Scheduling Automated Traffic on a Network of Roads

Arvind Giridhar and P. R. Kumar, *Fellow, IEEE*

***Abstract*—The authors consider the problem of scheduling automated traffic in a city. Such automatic scheduling can improve efficiency of the system by decreasing delays, increasing capacity, and easing congestion. The algorithms described in this paper can be thought of as which belongs to a layer in the corresponding control–communication–computational system, which is responsible for generating timed trajectories for individual vehicles [S. Graham, G. Baliga, and P. Kumar, "Issues in the convergence of control with communication and computing: Proliferation, architecture, design, services, and middleware," in Proceedings of CDC '04, Nassau, Bahamas, December 2004]. Each vehicle has a specified route from its origin to its destination, and the task of the scheduler is to provide timed trajectories for all vehicles, which follow the respective vehicles' routes and further ensure that no collisions or deadlock will result. The authors' approach reduces the problem to a discrete-time graph-scheduling problem, by defining an appropriate graph to model the road network. Their main result is a sufficient condition on the graph of the road network and on the initial distribution of vehicles, under which there exists a scheduling algorithm that is guaranteed to clear the system in finite time. The nature of this result allows the design of provably correct scheduling algorithms that require only a small portion of future routes of all vehicles to be known, and are consequently able to work in real time. The authors also address the optimization of performance with respect to delay, by focusing on the "one-step move" problem. Finding an optimal solution for the one-step problem would provide a greedy solution of the original network-wide scheduling problem, but is itself NP-hard. They present a polynomial time heuristic algorithm and evaluate its performance through simulations.**

***Index Terms*—Deadlocks, graph scheduling, smart vehicles, traffic networks, traffic scheduling.**

I. Introduction

FUTURE TRAFFIC systems could include vehicles that are automatically driven along smart highways and/or city roads [2], [3]. Such automation of traffic movement, along with a control infrastructure for the overall coordination and scheduling of traffic, could improve efficiency, reduce travel time, and prevent traffic deadlocks.

Manuscript received April 17, 2003; revised March 19, 2004, June 22, 2005, and September 22, 2005. This work was supported in part by Air Force Office of Scientific Research (AFOSR) under Contract No. F49620-02-1-0217, Defense Advanced Research Projects Agency (DARPA) under Contract Nos. F33615-01-C-1905 and N00014-01-1-0576, United States Army Research Office (USARO) under Contract Nos. DAAD19-00-1-0466 and DAAD19-01010-465, DARPA/AFOSR under Contract No. F49620-02-1-0325, and National Science Foundation (NSF) under Contract Nos. NSF ANI 02-21357 and NSF CCR-0325716. The review of this paper was coordinated by Prof. D. Lovell.

The authors are with the Department of Electrical and Computer Engineering, and Coordinated Science Laboratory University of Illinois at Urbana–Champaign, Urbana, IL 61801 USA (e-mail: giridhar@uiuc.edu; prkumar@uiuc.edu).

Digital Object Identifier 10.1109/TVT.2006.877472

A miniature and highly simplified version of such a futuristic scenario is the vehicular testbed recently constructed in the Information Technology Convergence Lab at the University of Illinois [4]. This testbed consists of remote controlled vehicles controlled individually by dedicated laptop computers, cameras providing vision feedback, and additional computers, which process this feedback and communicate instructions to the individual laptops. The laptops and the servers are connected together to form a wireless network. One of the main tasks required of this distributed system is to automatically schedule movements of the vehicles in such a way that no collision occurs, and no vehicle is ever required to deviate away from its prespecified route.

Such an automated system features control at several layers [5], [6]. At the lowest layer, closed loop control based on the sensor inputs ensures that each vehicle follows a timed trajectory given by a higher layer for a short duration of time. At a higher layer, this timed trajectory is itself generated, based on the layout of the roads and the intersections, the desired routes of all the vehicles, and the positions of all the vehicles. The main component of this layer is a scheduling algorithm, which must be provably correct in that it must ensure that at no time in the future will the system ever reach a deadlock, and must also yield low delays for the vehicles in reaching their destinations. It is this scheduling problem that is the focus of this paper.

The contribution of this paper can be summarized as follows. We construct a simple discrete-space and discrete-time model for the movements of vehicles, which ensures a minimum safe distance between adjacent vehicles. We then derive a class of scheduling algorithms that are provably deadlock free given a certain sufficient condition. Our main result is a sufficient condition on the configuration of vehicles on the road network and the next "step" in the route of each vehicle, which guarantees that deadlock can be avoided and the network cleared, irrespective of future (beyond the one step) route information. This allows the design of "online" algorithms, which only require current and immediate future information to produce schedules that are guaranteed to be loop free. We further address the important issue of optimization with respect to delay, and describe a delay efficient deadlock free algorithm that has been implemented in the testbed described above.

Traffic flow modeling has been extensively studied in the transportation literature. There are macroscopic approaches (e.g., [7], [8]) which model traffic flow on the basis of "macro" parameters such as throughput, mean velocity, global density, etc. In contrast, microscopic modeling approaches, such as vehicle following models [9], cell transmission models [10], and cellular automaton models [11]–[13], propose simple rules for individual vehicle behavior and proceed to analyze the

dynamic behavior of aggregates of such vehicles. Our modeling approach is microscopic in nature, and can be regarded as a simplification of the cellular automaton model. However, our model does not seek to capture the macroscopic behavior of current human driven traffic, but rather to provide an abstraction for the control algorithm.

Research into traffic control has included study of route guidance strategies as well as traffic-signal-control strategies. Route guidance has been extensively studied under the context of dynamic traffic assignment, which is a component of Advanced Traveler Information Systems and Advanced Traffic Management Systems (ATIS/ATMS). Methodologies used include simulation-based approaches [14], [15], optimal control-based approaches [16], and optimization-based approaches [17]. Traffic-signal-control strategies would fall in the class of what is referred to here as scheduling policies or algorithms. There has been research into traffic responsive signal control (see [18] for an overview), involving design of strategies to optimize the various signal control parameters such as cycle time, phase split, and offset in order to maintain a high degree of saturation in the road network.

Scheduling problems similar to those in this paper have been considered in the context of flexible manufacturing systems (FMS). In such systems, processes concurrently competing for exclusive use of resources, combined with a circular chain of processes in which each process is waiting for the next process in the chain, can lead to deadlock. In this paper, we refer to such an occurrence as an occupied cycle. The problem of deadlock avoidance, or scheduling to avoid deadlocks (as opposed to other approaches, which seek to prevent deadlocks by restraining request structure of resources, or approaches that seek to detect and recover from deadlock) has been studied in the FMS context [19]–[22]. A traffic network can be regarded as an instance of a single-unit resource allocation system (SU-RAS) [22], with vehicles being the jobs and discrete sections of streets being the resources. There are however some important distinctions between our model and the FMS model. First, in our traffic model, simultaneous transitions of jobs (vehicles) from one resource to another available resource may be forbidden due to "edge conflicts" in the corresponding graph. This feature does not exist in FMSs. Second, there is a difference in the notion of a deadlock in an FMS and that used here. In a traffic network with automated vehicles and centralized control, we contend that an occupied cycle does not imply deadlock; if every vehicle in the circuit moves in synchronized fashion, no collisions will result and each vehicle can move forward. Our notion of deadlock requires an even stronger condition involving conflicting edges. Interestingly, our main result, independently derived here, turns out to be an extension of a similar result derived for SU-RASs [21], [22].

The remainder of the paper is organized as follows. Section II describes the model of the system and elaborates on the problem statement. Section III provides conditions for the existence of a feasible scheduling algorithm. Section IV presents a greedy algorithm for one step in the above scheme and assesses its performance through simulations. Section V addresses the applicability of this approach to actual traffic networks, followed by some possible directions for future work.

II. System Model and Problem Statement

We construct a discrete-time discrete-space model of the system. Let each directed lane be divided into cells, and discretize time into slots so that at the beginning of each slot, at most one vehicle occupies a single cell. This essentially imposes a minimum safe distance between vehicles in the same lane (assuming each vehicle is situated in the middle of its cell at the start of every time slot), equal to the length of the cell less the length of the vehicle, assuming that the lengths of each cell and vehicle are fixed. Assume further that all vehicles move at a constant speed, or are stationary, and the length of a time slot is chosen such that a vehicle can only traverse from one cell to an adjacent one in a single time slot. For instance, if the vehicle length is c, minimum safe distance d, and the common speed v, the length of a cell is $d + c$ and the duration of a single slot is $(d + c)/v$.

Additionally, the intersection of two streets is not modeled as a cell, but rather consider the cells immediately preceding and succeeding the intersection to be adjacent.

Further, if there are adjacent lanes in the same direction, the lanes can be sectioned so that a lane change operation from a cell in one lane to the "next" cell in the adjacent lane (with next denoting the cell adjoining the upstream neighbor) is permitted. This defines adjacency between cells in parallel lanes going in the same direction. It is important to note that the scheduling requirement of single time-slot cell-to-cell shifting does not require all cells to have exactly the same dimension, or for all vehicles to have the same speed all the time, or indeed for speed to jump from 0 to v instantaneously. For instance, cells adjoining an intersection can be of shorter length, since they are not physically adjacent. Further, this physical nonadjacency means that a vehicle must travel at a higher speed to cross an intersection in a single time slot. The salient fact is that this class of schedules ensures that vehicles in the same lane will be separated by a minimum safe distance.

The road network can be modeled as a directed graph $\mathcal{G}$ by representing each cell by a vertex $v \in \mathcal{V}$, and by representing cell adjacencies by directed edges in a set $\mathcal{E}$. Edge uv oriented from vertex u to vertex v indicates the adjacency of the corresponding cells, and orientation of the lane joining them. The notation "$u \to v$" will be used to mean that such an edge exists. Two edges are said to conflict if the paths between the corresponding pairs of cells intersect. The graph can thus be represented by a $|\mathcal{V}| \times |\mathcal{E}|$ incidence matrix

$$I(i,j) = \begin{cases} 1, & \text{if } i \text{ is the tail vertex of edge } j \\ -1, & \text{if } i \text{ is the head vertex of edge } j \\ 0, & \text{otherwise} \end{cases}$$

and a symmetric $|\mathcal{E}| \times |\mathcal{E}|$ conflict matrix

$$C(i,j) = \begin{cases} 1, & \text{if edges } i \text{ and } j \text{ conflict} \\ 0, & \text{otherwise.} \end{cases}$$

Next, let $[n] := \{1, 2, \ldots, n\}$ denote the set of vehicles. The state of the system is described by the $X(t) = [x_1(t), x_2(t), \ldots, x_n(t)]$, $t = 1, 2, 3, \ldots$ where $x_i(t) \in \mathcal{V}$ is the vertex corresponding to the cell containing vehicle i at time t. Equivalently,

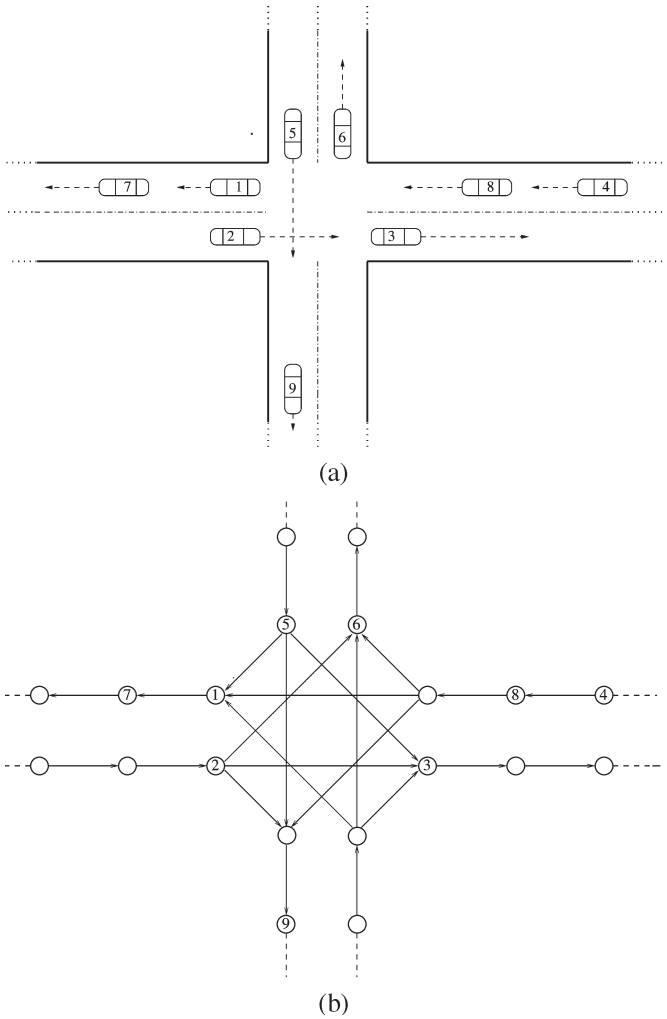


Fig. 1. Graph representation of a four-way intersection. Crossing edges represent edge conflicts. (a) Road network and (b) graph.

vehicle i is said to occupy vertex $x_i(t)$ at time t. Fig. 1 shows the graph obtained from a four-way intersection.

The evolution of the system over time is thus described by the sequence $\{X(t) : t = 1, 2, 3, \ldots\}$. The constraints on the movements of vehicles are as follows.

1) Given any time t and vehicle i, either $x_i(t) = x_i(t+1)$, or, if not, $x_i(t) \to x_i(t+1)$. In other words, a vehicle can only shift to an adjacent vertex in accordance with the corresponding edge orientation.
2) If $i \neq j$, $x_i(t) \neq x_j(t)$. A vertex can be occupied by only one vehicle.
3) Vehicles cannot simultaneously shift across conflicting edges. That is, if $x_i(t) \neq x_i(t+1)$ and $x_j(t) \neq x_j(t+1)$, then edges $x_i(t)x_i(t+1)$ and $x_j(t)x_j(t+1)$ cannot conflict.

Each vehicle i has an associated route R_i, which is a directed walk on the graph $\mathcal{G}$, along which it is constrained to travel. Each vehicle begins at time 0 at its source and exits the system on reaching its destination. If vehicle i reaches its destination at time t', $x_i(t) = \phi$ for all $t > t'$, where ϕ is a dummy vertex. Note that the above conditions describing state transitions are valid only for vehicles that have not exited the system.

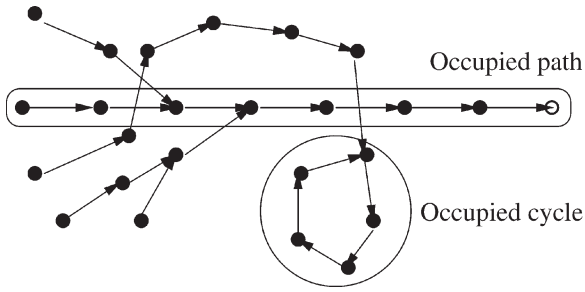


Fig. 2. Occupied path and cycle in a configuration.

Thus, the next state $X(t+1)$ is fully specified by the current state $X(t)$, the remaining parts of the routes of all vehicles, and the set of vehicles $S(t) \subseteq [n]$ that shift during $[t, t+1]$. Such a subset is called feasible (with feasibility implicitly referencing a particular time instant) if states $X(t)$ and $X(t+1)$ satisfy the feasibility conditions described above, together with the route following constraint.

A feasible system clearing schedule S is a sequence of feasible subsets $S_0, S_1, \ldots S_T$ of vehicles to be shifted at each time $0 \leq t \leq T$, such that for all $t' > T$ and each vehicle i, $x_i(t') = \phi$. Thus, a schedule is simply a sequence of sets of vehicles to be moved in each time slot, terminating when all vehicles reach their destination.

The objective of the scheduling problem can now be stated:

Given an initial state X_0, to determine a feasible schedule $S_0, S_1, \ldots, S_T$ for some $T \leq \infty$ that clears the system, or to show that no such schedule exists.

It must be emphasized that this definition of a feasible schedule, and indeed the discrete model of the system, is a restriction of the most general class of feasible schedules, the latter consisting of any set of continuous time trajectories that, if followed by the corresponding vehicles, ensures that each vehicle will travel from the corresponding source to destination along the given route.

III. Existence of Feasible Schedules

We now address the problem of finding conditions under which the system can be feasibly cleared. Naturally, we would like these conditions to be as weak as possible, with minimal assumptions on the underlying graph, as well as on the initial configuration of vehicles.

We define some terms for convenience.

A. Definitions

Given a state $X(t)$.

1) An occupied cycle (defined implicitly at time t) is a directed cycle that satisfies the following conditions.
   a) Each vertex in the cycle is occupied by a vehicle.
   b) If u, v are successive vertices in the cycle with $u \to v$, then v follows u in the remaining part of the route R_i of the vehicle i occupying u.
2) An occupied path is a directed path that satisfies condition b) above, in which the terminating vertex is either unoccupied or belongs to an occupied cycle, and in which every other vertex is occupied. Fig. 2 shows instances of

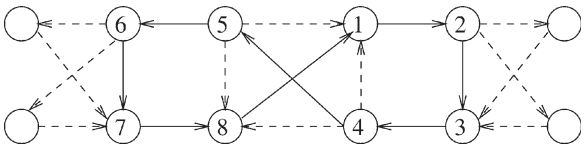


Fig. 3. Deadlock configuration.

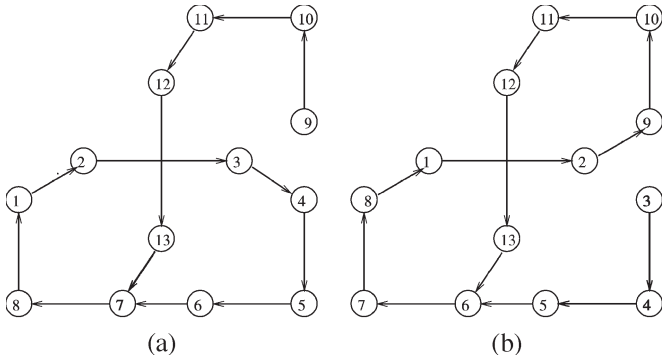


Fig. 4. Occupied cycle leading to deadlock. (a) Before shift and (b) after shift.

an occupied path and an occupied cycle (the edges shown are shift edges).

It is clear from the definitions that any occupied vertex must belong either to an occupied path or to an occupied cycle, and further that the occupied path beginning from that vertex, or the occupied cycle, is uniquely determined. This follows from the fact that the route of each vehicle uniquely specifies the next vertex to be shifted to.

B. Sufficient Conditions for Feasible Scheduling

The system is said to be in deadlock at time t if, given the state $X(t)$, there is a nonempty subset of vehicles, which cannot be shifted by any future feasible schedule. Fig. 3 gives an example of deadlock. The shift edges are solid, and the other edges are shown as dashed lines. The vehicles numbered 1 to 8 occupy a cycle. Clearly, if any one of them is to be shifted, every one must be shifted as well, which is, however, not feasible since two of the shift edges conflict.

It is thus clear that the presence of a cycle with mutually conflicting edges causes the system to be in deadlock. It may seem plausible that the absence of such a configuration at the initial time may be a sufficient condition for the existence of a feasible schedule. However, Fig. 4 shows that is not the case. Note that only the shift edges are shown. In Fig. 4(a), the vertices occupied by vehicles 1, 2, 3, 4, 5, 6, 7, and 8 form an occupied cycle. Shifting any one of these vehicles forces every one of them to be shifted, resulting (due to the routes of each of these vehicles) in Fig. 4(b). Here, we have a self-conflicting occupied cycle consisting of the vertices occupied by 1, 2, 9, 10, 11, 12, 13, 6, 7, and 8. Therefore, in fact, the configuration in Fig. 4(a) is already in deadlock, since vehicles 9, 10, 11, 12, and 13 cannot be shifted.

However, a slightly stronger condition is sufficient to guarantee absence of deadlock, as well as existence of a feasible schedule. The following result is an extension in [21, Th. 2] (rederived in [22]) to our model.

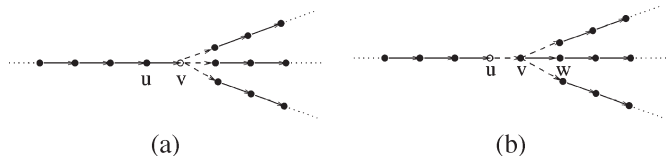


Fig. 5. Case 1. (a) Time t and (b) time $t+1$.

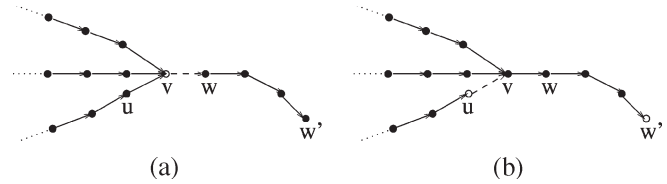


Fig. 6. Case 2. (a) Time t and (b) time $t+1$.

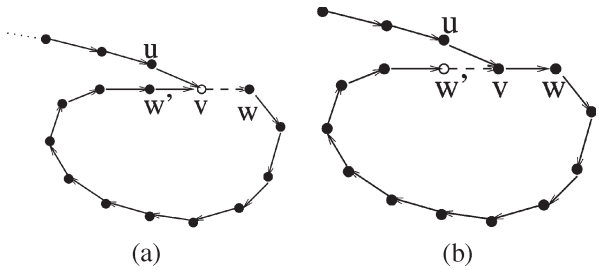


Fig. 7. Case 3. (a) Time t and (b) time $t+1$.

Theorem 1: If the directed graph $\mathcal{G}$ is such that each vertex has either in-degree 1 or out-degree 1, and the system has no occupied cycle at time 0, then there exists a schedule that clears the system in finite time.

Proof: It is enough to prove the following: If the system has no occupied cycle at time t, there is a feasible set S_t of size 1, which results in a state $X(t+1)$ having no occupied cycle. If this is proved, it follows that at each step it is possible to move at least one vehicle by one step along its route. Since the number of vehicles and the route lengths are finite, it follows that the system will eventually be cleared.

Consider any vertex, and the maximal occupied path beginning at that vertex. Such a path must end at a vertex u, and, since there is no occupied cycle, the vertex v following u in the route of the vehicle occupying u must be unoccupied.

The vertex v must either have in-degree or out-degree equal to 1, by assumption. Consider first the case in which the in-degree of v is 1 (Fig. 5).

Let $S_t = \{i : x_i(t) = u\}$. We now claim that there is no occupied cycle at time $t+1$. Since there is no occupied cycle at time t, and the only vertex that goes from unoccupied to occupied at time $t+1$ is v, any occupied cycle must include v. Since the in-degree of v is one, the vertex preceding v, i.e., u, must also belong to this occupied cycle. However, u is unoccupied at time $t+1$, so this is not possible.

Now consider the case in which the out-degree of vertex v is 1; i.e., there is a unique vertex w such that $v \to w$. If w is occupied (at time t), consider the occupied path consisting of w and all its downstream occupied vertices. Let w' be the terminating vertex of this path (see Fig. 6). If w' is an "in-neighbor" of v, i.e., if $w' \to v$ (see Fig. 7), then let $S_t = \{i :$

$x_i(t) = w'\}$. Otherwise, let $S_t = \{i : x_i(t) = u\}$ (see Fig. 6). Once again, any occupied cycle at time $t+1$ must include v. Since the out-degree of v is one, it must also include w, and indeed the occupied path (in case w is occupied) starting from w, including w'. If w itself is unoccupied, this is impossible. If not, either w' is empty, or the next vertex in the route of the car occupying w' at time t remains unoccupied at time $t+1$. In either case, an occupied cycle including w' is not possible. ■

The condition that every vertex must have either in-degree or out-degree not greater than 1 (i.e., either in-degree or out-degree can be greater than 1, but not both) may seem rather restrictive. It would be violated if there are multiple adjacent lanes in the same direction, and lane changes are allowed. One way to preserve the desired graph structure in such situations would be to limit lane changes to alternate sections (and thereby limit the graph adjacency to preserve the degree condition).

IV. One-Step Scheduling: Choosing a Move

The next question that arises is: How do we choose a sequence of moves that minimizes the clearing time? In the general setting of arbitrary routes of vehicles on an arbitrary graph, this problem is NP-hard.

Theorem 2: Finding a schedule that clears the system in minimum time is an NP-hard problem.

Proof: Consider an arbitrary graph $\mathcal{G}$. Finding a proper coloring of minimum size is equivalent to partitioning the vertex set into the minimum number of independent sets. This problem is known to be NP-hard [23]. Just as in Theorem 2 above, transform $\mathcal{G}$ into a directed graph $\mathcal{G}'$. Now, let each vehicle occupying a vertex in $\mathcal{G}'$ have a route of length 1 (i.e., along the corresponding directed edges to the adjacent vertices). At each time step, the subset of vehicles that can be shifted occupy a set of vertices that correspond to an independent set in $\mathcal{G}$. The shifted vehicles reach their respective destinations and exit the system. The minimum time to clear the system is therefore the minimum number of such subsets that together include all the vehicles, which corresponds to the minimum number of disjoint independent sets that cover all the vertices in $\mathcal{G}$. ■

In addition, finding the global minimum requires complete route information of all vehicles, which may not be available in practice.

However, Theorem 1 shows that the sufficient condition of absence of any occupied cycle depends only on current state and the next step in each route. This allows the design of real time deadlock free algorithms, which only require current state information and very limited (specifically, two steps) future route information, and output schedules for only a limited time horizon. This is a more practical solution, since continuous replanning may be necessary due to factors such as inexact following of schedule instructions, changes in routes, changes in traffic conditions, etc.

We are thus motivated to study the following one-step-move problem: Given a current state $X(t)$ of the system at time t, which is free of occupied cycles, find the largest subset of vehicles $S_t \in [n]$ (recall that $[n]$ is the set of vehicles) that can be simultaneously shifted such that the resulting configuration $X(t+1)$ is also free of occupied cycles. We will henceforth refer to this largest subset as the largest feasible subset (i.e., we expand the notion of feasibility of a subset as used in previous chapters to include the constraint that shifting the subset does not create any occupied cycles). However, the one-step-move problem is also NP-hard for graphs with arbitrary edge conflicts.

Theorem 3: The largest feasible subset problem described above is NP-hard.

Proof: We prove NP-hardness by reduction of an instance of this problem to the maximum independent set problem, which is known to be NP-hard [23]. Consider an arbitrary instance of a maximum independent set problem, i.e., finding the largest induced subgraph of a graph $\mathcal{G}$ with the restriction that the subgraph has no edges. Transform $\mathcal{G}$ into a directed graph $\mathcal{G}'$ by substituting each vertex of $\mathcal{G}$ with a directed edge from an occupied vertex to an unoccupied vertex, with any two edges in $\mathcal{G}'$ conflicting if and only if the corresponding vertices in $\mathcal{G}$ are adjacent. We now seek a largest subset of edges in $\mathcal{G}'$ that have no mutual conflicts, which is equivalent to finding the largest subset of vehicles that can be feasibly shifted. ■

As an alternative solution, we propose the following polynomial time suboptimal algorithm, which is guaranteed to be deadlock free, and as will be seen in the simulation results of Section V, appears to perform quite well.

The algorithm uses a notion of neighborhood, defined on the collection of feasible subsets of $[n]$. A feasible subset S_2 (note that we have dropped the time index, since we are dealing with a one-step problem) is said to be a neighbor of feasible subset S_1, if S_2 can be obtained from S_1 by adding to S_1 a single occupied path[1] and deleting whichever of the remaining vehicles/occupied vertices must be deleted to obtain a feasible subset. The deletion step is a linear time operation, which consists simply of visiting the occupied vertices with shift edges that conflict with the added path, and deleting the occupied paths starting with these vertices.

S_1^m is said to be an m-neighbor of S_2 if it is a maximal feasible subset (i.e., it is not contained in any feasible subset), and it contains a neighbor of S_2. Observe that the notions of neighbor and the m-neighbor are asymmetric; if S_2 is the m-neighbor of S_1, it need not be that S_1 is the m-neighbor of S_2.

In light of the above definitions, we slightly modify the above description of the algorithm by replacing feasible subsets by maximal feasible subsets, and neighbors by the m-neighbors. In other words, we begin with a maximal feasible subset, and successively replace the current subset by a larger m-neighbor, stopping when we reach a local optimum.

The algorithm works as follows: Construct an initial set S_{init} by setting up a list of nonself-conflicting occupied paths in decreasing order of length, and sequentially picking out the path at the top of the list, adding it to S_{init} if the addition preserves feasibility, or else discarding it. The procedure ends when the list is empty. This gives a maximal feasible set S_{init}.

Given the current set S, pick any path in $\bar{S}$ (the complement of S) and add it to S. Delete the elements of the original set that conflict with the chosen path. Using the list of paths in $\bar{S}$, repeat

[1] "Paths being added to feasible subset" is shorthand for vehicles occupying paths being added.

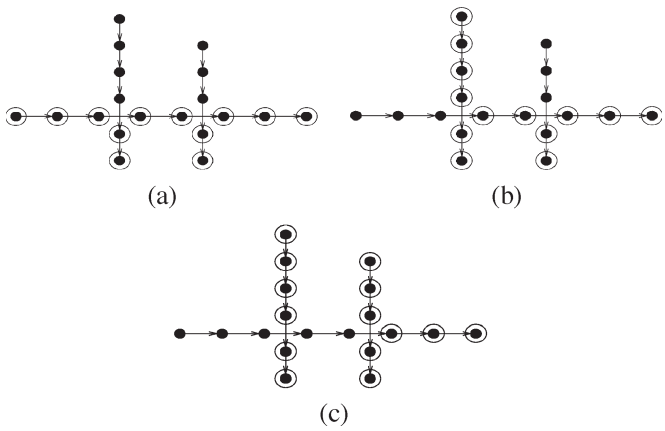


Fig. 8. Application of heuristic algorithm to example configuration. (a) First iterate, (b) second iterate, and (c) third iterate.

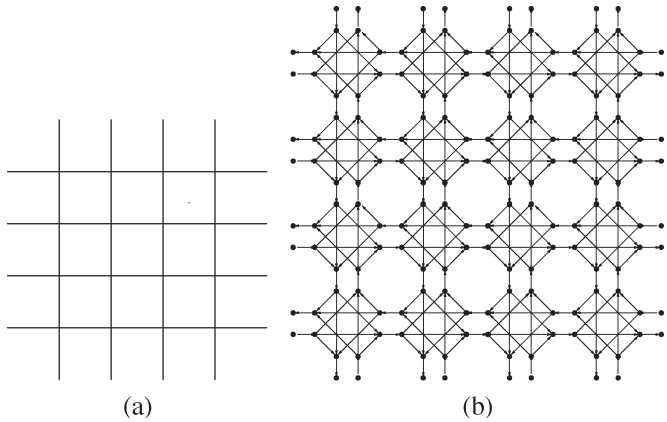


Fig. 9. Test graph for simulation. (a) 4×4 grid and (b) graph representation.

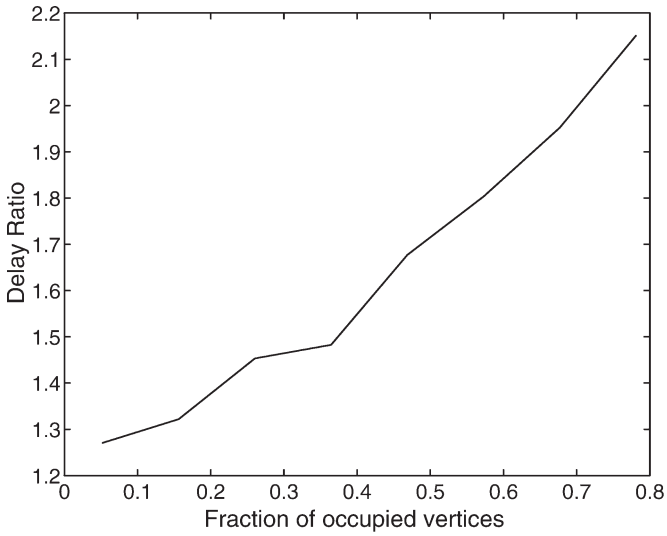


Fig. 10. Performance of the heuristic algorithm.

the procedure described in the previous paragraph to obtain the m-neighbor of S, say S'. If $|S'| > |S|$, then set $S = S'$ and begin a new iteration. If not, repeat the same operation with another path in S.

Each step in the algorithm results in a strictly larger feasible subset. The algorithm terminates when all the paths in S are checked without obtaining a larger m-neighbor subset S'.

We illustrate the working of the algorithm by applying it to an example configuration (see Fig. 8). The vertices shown are occupied vertices, and the edges are the shift edges. Fig. 8(a) shows the result of the initial greedy covering (the selected occupied vertices are circled). After two steps [Fig. 8(b) and (c)], the algorithm achieves the local optimum, which in this case is also the global optimum.

It is not hard to obtain a crude polynomial bound (polynomial in the number of vehicles) on the running time of the algorithm. Let n be the number of vehicles. Checking whether a subset is feasible has complexity $O(n^2)$. The number of occupied paths is $O(n)$. To obtain the initial set S_{init} consequently takes $O(n^3)$ time. There are $O(n)$ neighbors of a feasible subset. Comparing with the set of neighbors takes $O(n^3)$ time, and this process is done $O(n)$ times (since the set S must grow strictly larger each time). The algorithm's running time is therefore bounded by $O(n^4)$. This is the time complexity per step. There are at most $O(R_{\max}n)$ steps in all, where $R_{\max}$ is the length of the longest route.

V. Simulation Results

Given an arbitrary configuration, in the worst case, only a single vehicle can be shifted in each move. Since the algorithm does not exercise any control on future configurations, apart from ensuring that they will contain no occupied cycle, we cannot provide a theoretical lower bound on the performance, except for the worst case bound $O(R_{\max}n)$ of moving one vehicle section per time slot.

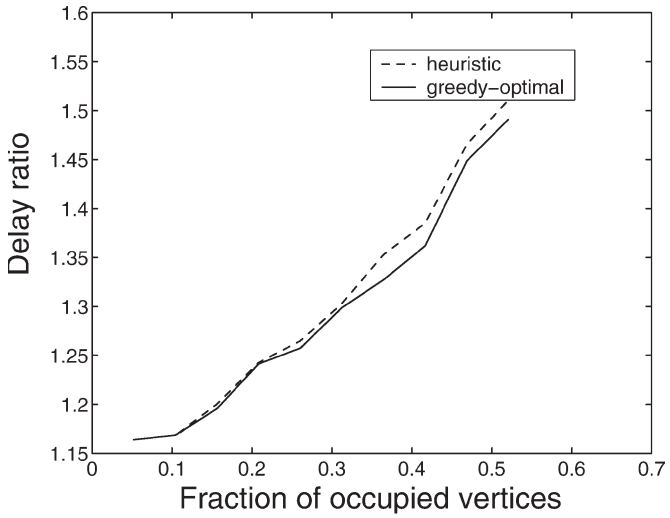
We have therefore tested through simulations the performance of the above heuristic algorithm and a greedy-optimal algorithm that chooses at each step the largest feasible subset. The greedy-optimal algorithm uses essentially an exponential-time brute-force approach (with some slight refinements that reduce the exponent by a factor) at each time step, going through all possible subsets to find the largest feasible subset. This algorithm has exponential complexity.

In the example simulated, the graph is obtained from a road network grid (Fig. 9). Each line in the grid is a two-way street. Random pairs of vertices are picked as sources and destinations, and the routes are randomly selected from among the minimum length paths between each source and destination pair. Fig. 9 shows a 4×4 grid with two vertices per square edge.

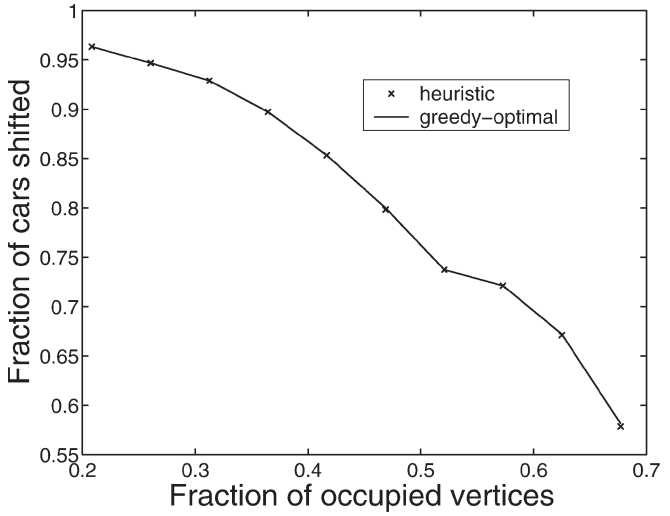
The algorithm's performance is measured by the ratio

$$\text{delay\_ratio}(n) = \frac{\text{schedule\_sum}}{\text{route\_sum}}$$

where schedule_sum is the total time for each vehicle to reach its destination (according to the schedule obtained by the algorithm) summed over all vehicles, route_sum is the sum over all vehicles of the route lengths, and n is the number of vehicles in the system. Clearly, $(\text{route\_sum})/n$ is a lower bound on the minimum average time per vehicle to clear the system. For example, a ratio of 2 implies that on average a vehicle is shifted in as many time slots as it is held stationary. These values are averaged over multiple random placements.



(a)



(b)

Fig. 11. Comparisons of greedy-optimal and heuristic algorithms. (a) Delay ratio comparison and (b) one-step comparison.

Fig. 10 shows the plot of delay_ratio(n) versus fraction of occupied vertices, for a 3×3 grid with two vertices per square edge (each way). The graph has 96 vertices. We are not able to show a comparison with the greedy optimal, since the exponential complexity of the latter makes it infeasible to calculate for a large number of vehicles. It shows that even when the density of vehicles per vertex approaches 1, delay_ratio(n) does not increase significantly.

Fig. 11(a) shows the comparison of the two algorithms for a smaller graph, a 2×2 grid with four vertices per square edge (a 96 vertex graph). Fig. 11(b) shows the comparison in terms of number of vehicles moved in a single step. The results indicate that the heuristic algorithm closely, or exactly in the case of the one-step simulation, matches the performance of the greedy-optimal algorithm.

Note that these results do not tell the story for large dense configurations, in which the exponential complexity of the greedy-optimal algorithm makes it infeasible to run, and thus does not allow such comparisons. Also note that none of the plots reflect algorithm performance relative to the global optimum. We have no approximation or bound for the latter, and exhaustively searching over all possibilities is computationally infeasible even for small configurations.

VI. Generalizations, Real World Considerations, and Future Work

Some extensions and relaxations of assumptions in our model can be made without much difficulty. For example, the assumption that no new vehicles can enter the system can be relaxed to allow vehicles to enter the system at arbitrary times as long as the occupancy constraints are not violated, and no occupied cycles are introduced. The assumption that the routes are fixed and known beforehand can also be relaxed due to the online nature of Theorem 1: Routes can be changed at any time as long as occupied cycles are not introduced.

Some important elements, however, are absent in this model. These include multiple vehicle speeds, the cost attached to stopping and starting, and the possible requirement of different minimum distances depending on speed.

The class of allowable schedules can be expanded to include some of these elements. For instance, the requirement that the speeds of all vehicles are the same could be relaxed to allow speeds to be multiples of a minimum speed. The system could then be discretized according to this minimum speed, and multiple vertices would be allowed to be traversed in a single time slot. Also, a cost can be attached to stopping and starting, and the resulting optimization problem of picking, which vehicles are allowed to shift would include the costs as weights. To guarantee different safe distances depending on speed, a rule enforcing multiple empty slots in between vehicles could be introduced. The mapping of the road into sections would then be according to the minimum safe distance at the minimum nonzero speed. Analyzing such a model could be a possible future direction to investigate.

With the likelihood of graphs of very large size, the computational complexity of any algorithm becomes crucial. A greedy approach in this class of algorithms results in an NP-complete problem. We present instead a polynomial time heuristic algorithm. The simulations show that in most of the randomly generated instances on the graph obtained from a grid road network, the heuristic performs almost as well as the greedy optimal solution. Finding lower order polynomial algorithms is definitely worth looking into. Another issue concerning scalability which arises is that providing scheduling instructions individually to all vehicles could be a hard problem from a communication point of view. It would be worth investigating whether such algorithms could be implemented in decentralized fashion.

On the theoretical side, given the assumption of arbitrary fixed routes, we have not been able to find useful lower bounds on the clearing time of the algorithm (i.e., an algorithm with a useful lower bound or approximation factor). It might be possible to obtain stronger results if another degree of freedom is added, i.e., if the routes are allowed to be variable.

Indeed, there are a number of ways to generalize the problem definition, and some of them may be worth investigating for possible savings achievable through more flexible constraints.

REFERENCES

[1] S. Graham, G. Baliga, and P. Kumar, "Issues in the convergence of control with communication and computing: Proliferation, architecture, design, services, and middleware," in *Proc. CDC*, Nassau, Bahamas, Dec. 2004, pp. 1466–1471.

[2] P. Varaiya, "Smart cars on smart roads: Problems of control," *IEEE Trans. Autom. Control*, vol. 38, no. 2, pp. 195–207, Feb. 1993.

[3] S. Shladover, C. Desoer, J. Hedrick, M. Tomizuka, J. Walrand, W. Zhang, D. McMahon, H. Peng, S. Sheikholeslam, and N. McKeown, "Automated vehicle control developments in the PATH program," *IEEE Trans. Veh. Technol.*, vol. 40, no. 1, pp. 114–130, Feb. 1991.

[4] The I.T. Convergence Lab Univ. Illinois, [Online]. Available: http://black1.csl.uiuc.edu/~prkumar

[5] S. Graham and P. Kumar, "The convergence of control, communication and computation," in *Proc. PWC*, Venice, Italy, 2003, pp. 458–475.

[6] G. Baliga, S. Graham, L. Sha, and P. Kumar, "EtherWare: Domainware for wireless control networks," in *Proc. 7th IEEE Symp. Object-Oriented Real-time Distrib. Comput.*, Vienna, Austria, 2004, pp. 155–162.

[7] G. Newell, "A simplified theory of kinematic waves in highway traffic, Part 1: General theory," *Transp. Res. B*, vol. 27, no. 4, pp. 281–287, Aug. 1993.

[8] B. Kerner and P. Kornhauser, "Structure and parameters of cluster in traffic flow," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 50, no. 1, pp. 54–83, Jul. 1994.

[9] G. Newell, "A simplified car-following theory: A lower order model," *Transp. Res. B*, vol. 36, no. 3, pp. 195–205, Mar. 2002.

[10] C. Daganzo, "The cell transmission model, Part 2: Network traffic," *Transp. Res. B*, vol. 29B, no. 2, pp. 79–93, Apr. 1995.

[11] K. Nagel and M. Schreckenberg, "A cellular automaton model for freeway traffic," *J. Phys. I*, vol. 2, pp. 2221–2229, 1992.

[12] K. Nagel, "Particle hopping models and traffic flow theory," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 53, no. 5, pp. 4655–4672, May 1996.

[13] A. Schadschneider and M. Schreckenberg, "Cellular automaton models and traffic flow," *J. Phys. A, Math. Gen.*, vol. 26, no. 15, pp. 679–683, Aug. 1993.

[14] H. Mahamassani, T. Hu, S. Peeta, and A. Ziliaskopoulos, "Development and testing of dynamic traffic assignment and simulation procedures for ATIS/ATMS applications," Center for Transportation Research, Univ. Texas, Austin, TX, Tech. Rep. DTFH61-90-R-00074-FG, 1994.

[15] M. Ben-Akiva, M. Bierlaire, H. Koutsopoulos, and R. Mishalani, "DynaMIT: A simulation based system for traffic prediction," in *Proc. DACCORD Short Term Forecasting Workshop*, 1998.

[16] T. Friesz, F. Luque, R. Tobin, and B. Wie, "Dynamic network traffic assignment considered as a continuous time optimal control problem," *Oper. Res.*, vol. 37, no. 6, pp. 893–901, Nov./Dec. 1989.

[17] D. Merchant and G. Nemhauser, "A model and an algorithm for the dynamic traffic assignment problem," *Transp. Sci.*, vol. 12, no. 3, pp. 183–199, 1978.

[18] M. Bell, "Future directions in traffic signal control," *Transp. Res. A*, vol. 26, no. 4, pp. 303–313, Jul. 1992.

[19] Z. Banaszak and B. Krogh, "Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows," *IEEE Trans. Robot. Autom.*, vol. 6, no. 6, pp. 724–734, Dec. 1990.

[20] R. Wysk, N. Yang, and S. Joshi, "Detection of deadlocks in flexible manufacturing cells," *IEEE Trans. Robot. Autom.*, vol. 7, no. 6, pp. 853–859, Dec. 1991.

[21] M. Fanti, B. Maione, and A. Turchiano, "Event-based feedback control for deadlock avoidance in flexible production systems," *IEEE Trans. Robot. Autom.*, vol. 13, no. 3, pp. 347–363, Jun. 1997.

[22] M. Lawley and S. Reveliotis, "Deadlock avoidance for sequential resource allocation systems: Hard and easy cases," *Int. J. Flexible Manuf. Syst.*, vol. 13, no. 4, pp. 385–404, Oct. 2001.

[23] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Murray Hill, NJ: Freeman, 1979.

Arvind Giridhar received the B.Tech. degree from Indian Institute of Technology (IIT) Bombay, India, and the M.S. degree from University of Illinois at Urbana–Champaign, where he is currently working toward the Ph.D. degree, in 2000 and 2002, respectively.

P. R. Kumar (S'77–SM'86–F'88) received the B.Tech degree from Indian Institute of Technology (IIT), Madras, India, in 1973 and the D.Sc. degree from Washington University, St. Louis, MO, in 1977.

From 1977–1984, he was with the Department of Mathematics, University of Maryland Baltimore County, and since 1985, he has been with the University of Illinois at Urbana-Champaign, where he is currently Franklin Woeltge Professor of Electrical and Computer Engineering and Research Professor in the Coordinated Science Laboratory.

Prof. Kumar was the recipient of the Donald P. Eckman Award of the American Automatic Control Council in 1985.