Optimization Models for Assessing the Peak Capacity Utilization of Intelligent Transportation Systems

Nirav Shah
Department of Computer Science
University of Texas at Dallas
800 W. Campbell Road,
Richardson, TX 75080-3021
E-mail: nirav@utdallas.edu
Phone: 214-293-6322

Subodha Kumar†
Michael G. Foster School of Business
University of Washington
Box 353200, Seattle, WA 98195
E-mail: subodha@u.washington.edu
Phone: 206-543-4777
Fax: 206-543-3968

Farokh Bastani
Department of Computer Science
University of Texas at Dallas
800 W. Campbell Road,
Richardson, TX 75080-3021
E-mail: bastani@utdallas.edu
Phone: 972-883-2299

I-Ling Yen
Department of Computer Science
University of Texas at Dallas
800 W. Campbell Road,
Richardson, TX 75080-3021
E-mail: ilyen@utdallas.edu
Phone: 972-883-6446

† Corresponding Author
Optimization Models for Assessing the Peak Capacity Utilization of Intelligent Transportation Systems

Abstract

Due to economic and physical resource constraints, it is not feasible to continually expand transportation infrastructures to adequately support the rapid growth in the usages of these infrastructures. This is especially true for traffic coordination systems where the expansion of the road infrastructure has not been able to keep pace with the increasing number of vehicles, thereby resulting in congestion and delays on the roads as well as in substantial increase in pollution. Hence, in addition to striving for the construction of new roads, it is imperative to develop new intelligent transportation management and coordination systems that can effectively enable the existing infrastructure to be used more efficiently. However, for a deployed solution to be practical and cost-effective, it needs to be provably superior to current methods. The effectiveness of a given technique can be evaluated by comparing it with the optimal capacity utilization. If this comparison indicates that substantial improvements are possible, then the cost of developing and deploying an intelligent traffic system can be justified. Moreover, developing an optimization model can also help in capacity planning. For instance, at a given level of demand, if the optimal solution worsens significantly, this implies that no amount of intelligent strategies can handle this demand, and expanding the infrastructure would be the only alternative. In this paper, we demonstrate these concepts through a case study of scheduling vehicles on a grid of intersecting roads. We develop two optimization models namely, the mixed integer programming model and the space-time network flow model for this problem, and show that the latter model is substantially more effective. Moreover, we prove the strong NP-hard property of this problem and develop two polynomial-time heuristic solutions, which are evaluated and compared with the optimal capacity utilization obtained using the space-time network model. In addition, we also present important implications for the managers.

Keywords: Traffic, Transportation, Integer Programming, Space-Time Network

1. Introduction

Urban road traffic has grown substantially over the past few decades. The number of vehicles being added to the roads is increasing rapidly, especially in the developing countries. However, the capacity of the road infrastructure is not keeping pace with the increase in traffic due to a variety of factors, including the lack of space to build new roads, cost
Thus, traffic congestion is occurring much more frequently in urban areas. Traffic congestion is undesirable because it has several adverse effects. Increased congestion significantly increases the travel time of commuters, thereby deteriorating the quality of human life and contributing to psychological stress. Moreover, due to traffic congestion, since the vehicles travel at lower speeds and stop more frequently, fuel consumption greatly increases, which has adverse economic impacts. This also increases the emission of greenhouse gases that cause pollution and contribute to global warming, thus impacting the environment adversely. In addition, the wear and tear of the vehicles and the roadway infrastructure also increases.

Thus, it is imperative to eliminate or, at least, minimize traffic congestion. A traditional approach used by city planners is to simply build new roads to meet the increased demand on the infrastructure. However, the roadway infrastructure cannot be expanded at the same rate as the increase in the number of vehicles due to lack of free space, or infeasibility due to cost constraints. One other conventional approach is to reduce the number of vehicles on the roads by promoting vehicle sharing through carpool lanes. However, this approach benefits only a small percentage of commuters and, hence, it has not been very effective as a general strategy to reduce congestion.

Hence, it is essential to investigate new approaches to reduce traffic congestion. With recent advances in computer and networking technologies, new intelligent solutions can be developed that make efficient use of the existing road infrastructure and, perhaps, reduce the travel times and reduce the chances of congestion. Several researchers have applied intelligent techniques in various transportation problems such as controlling traffic on highways, traffic management at intersections, railroad traffic control etc. For example, work by Varaiya (1993) demonstrates a method that increases the flow of vehicles on highways by reducing inter-vehicle distance. Porche and Lafontune (1997) describe a method for adaptive traffic control at intersections. In the railroad domain, Matsumoto et al. (2002) explain how to use intelligent techniques to reduce the distance between successive trains and increase the frequency of trains. All these examples demonstrate that it is indeed possible to deploy intelligent solutions that use the existing infrastructure more efficiently and reduce the probability of congestion.
When city planners decide to invest in intelligent solutions, for the intelligent solutions to be cost-effective, it is essential that they be efficient. Obviously, deploying the optimal solution is the most efficient method. However, given the scale and the complexity of transportation systems, optimal solutions cannot be feasibly deployed. Therefore, heuristic solutions are most widely used in these cases. Since efficiency is a key requirement, it is necessary to evaluate the heuristic solutions. This can be achieved by comparing a deployed heuristic with the optimal solution. More importantly, finding the optimal solution can also help in capacity planning and in justifying the cost of deploying intelligent solutions. If the optimal solution shows that significant improvement is possible by promoting more efficient use of the existing infrastructure, then it would be feasible to invest resources in deploying intelligent solutions.

In this paper, we demonstrate these ideas through a case study of scheduling vehicles on a grid of intersecting roads. Much of the existing work in road traffic control at intersections is focused on reducing wait times at single intersection. We first develop an integer programming (IP) formulation for the problem of scheduling vehicles on a grid of intersecting roads rather than a single intersection. Then, we develop the optimization model using a special type of multi-commodity flow network, called the “space-time” network and show that this formulation is more efficient than the integer programming formulation. We also show that this problem is strongly NP-hard and, hence, we develop two heuristic algorithms. We evaluate these algorithms by comparing their results with the value of the optimal solution of the multi-commodity flow network. The results show that these heuristics are very effective and efficient for a wide variety of problem instances.

The rest of the paper is organized as follows. Section 2 presents an overview of related literature. In Section 3, we describe the problem and present mixed integer programming formulation. In addition, we prove that this problem is strongly NP-hard, and also develop another optimization model using space-time network. Two polynomial-time heuristics are proposed in Section 4. In Section 5, we present the experimental study, where we evaluate two optimization approaches and compare the heuristic algorithms with the optimal solution. In this section, we also discuss the managerial implications. Finally, Section 6 concludes the paper with summary and some of the possible future research directions.
2. Related Work

In this section, we present a brief overview of previous work in the related areas. The first stream of research that we survey is the design of intelligent systems for traffic management. Since we use the multi-commodity network flow to model the vehicle scheduling problem, its applications in various transportation domains is also reviewed in this section. We also highlight our contribution with respect to the past literature.

Several studies describe the importance of using the existing infrastructure more efficiently through the use of intelligent methods. Matsumoto et al. (2002) describe a decentralized automatic train control system (ATC) that reduces headway (minimum safe distance) between trains, and hence increases the frequency of trains by enabling the onboard computers of locomotives to autonomously compute their braking patterns. Similarly, Varaiya (1993) and Sotelo et al. (2000) present methodologies to improve the utilization of roadway infrastructures. Varaiya (1993) describes the concept of a “platoon,” where vehicles traveling along the same direction, in the same lane are grouped into a logical unit. All the vehicles in the platoon except the leading vehicle are automatically driven and possess sufficient intelligence to maintain a minimum safe distance with the preceding vehicle. The advantage of this approach is that the inter-vehicle distance between the vehicles belonging to a platoon can be substantially reduced, and higher average speeds are possible, which in turn improves the roadway utilization. Sotelo et al. (2000) perform a similar study by forming platoons of transit buses and demonstrate that the higher average speeds and reduced intra-bus spacing is achievable by grouping the buses into platoons. Halle et al. (2004) describe different approaches to organize vehicles into platoons.

Now we focus on related work in urban traffic control. The research in urban traffic control systems is focused to alleviate traffic congestion in three major areas, namely, improving the flow on highways (Varaiya 1993), controlling traffic lights at intersections (e.g., Porche et al. 1996, Porche and Lafontune 1997, Febbraro et al. 2003), and providing route guidance (Lei and Ozguner 2000, Mamei et al. 2003). Varaiya (1993) describes a hierarchical control architecture that seeks to balance traffic along all the lanes of highway. Based on announced destinations of different vehicles, the system suitably groups them into platoons, and assigns appropriate lanes to the platoons such that all the lanes are equally
utilized. Usually, the vehicles tend to take the shortest route to their respective destinations which could result in congestion on major arterial roads. Thus, several researchers have studied route guidance as an alternative approach to distribute traffic evenly on the road infrastructure. Mamei et al. (2003) describe a dynamic route guidance algorithm using an agent based approach. The agents (vehicles) autonomously compute the least congested paths to their respective destinations by relying on the traffic information (in form of force-fields) provided by the road network. In the similar direction, Lei and Ozguner (2000) propose an integrated route guidance and intersection control scheme. This system computes routes for the vehicles by taking into consideration the traffic conditions on the road network and feedback from intersection control system (i.e., the expected wait time at the intersections). Jozefowiez et al. (2009) address a different vehicle routing problem in which the total length of the routes is minimized. These authors propose a meta-heuristic method based on an evolutionary algorithm involving classical multi-objective operators.

Controlling the traffic lights to reduce traffic congestion has been studied by numerous researchers. Porche et al. (1996) and Porche and Lafortune (1997) focus on reducing the average waiting time at the intersections, whereas Robertson and Bretherton (1991) study the method of reducing the average intersection queue lengths. In addition, different types of techniques have been used to find innovative solutions for scheduling vehicles on intersections. For example, Oliveira and Bazzan (2005) and Ferreira et al. (2001) use an agent-based approach, Heung et al. (2005) propose fuzzy logic approach, and Bazzan (2005) resorts to game-theoretic approach. Febbraro et al. (2003) address the problem of minimizing the transit time for emergency vehicles using a hybrid approach with local controllers and a central controller. On the other hand, Porche et al. (1996) and Porche and Lafortune (1997) present a decentralized adaptive scheme for traffic signal optimization, called the ALLONS-D, which is not restricted to generate cyclic paths like the SCOOT method explained by Robertson and Bretherton (1991), and it can generate any arbitrary phase sequencing. In ALLONS-D, each intersection must be equipped with a signal optimizer that requires information about the current queue lengths and the future arrivals. Since each intersection independently generates phase timings, this may not result in global optimization of travel times. Hence, Porche and Lafortune (1998) propose a multi-layer coordination scheme where
global higher layers influence the decision of local signal controllers. Since it is essential to have coordinating intersection controllers to achieve global optimization of travel times on the road network, this is the focus of agent-oriented techniques to control intersection traffic lights. In this direction, Ferreira et al. (2001) propose a decentralized traffic control scheme where each intersection agent considers the feedback from neighboring intersection and the current traffic queue lengths to compute a phase switching sequence. Oliveira and Bazzan (2005) describe another agent-based approach for coordinated intersection control where the intersection agents engage in a coordination protocol to collectively decide on one traffic direction that should be given the priority. Some more work on urban traffic management involves traffic prediction and modeling of road conditions under different conditions. D’Acierno et al. (2009) propose a method that predicts road traffic conditions based on the location of bus fleets. Since much of existing traffic prediction work has been undertaken for arterial or freeway-like roads, as explained by Lee et al. (2009), the authors propose a knowledge-based real-time travel prediction model for urban roads. Since, a road network can encounter traffic incidents, Baykal-Gursoy et al. (2009) develop a model to simulate road conditions having incidents using a steady state M/M/C queueing system and determine the number of stationary vehicles. They present simulation results that validate the queueing models for computing average travel times. During traffic incidents, the approaching vehicles need to be re-routed in real-time. This is studied in the paper by J. Li (2009) in context of delivery or pick-up service vehicles.

Most of the current works involving urban traffic control systems focus on some form of experimental study to evaluate the proposed traffic control algorithms. However, not much work has been undertaken to develop optimization models for transportation problems and evaluate the proposed methods against the optimal solution. In this research, we fill this gap by presenting optimal solution methodologies and the heuristics for traffic control. In this section, we present a brief overview of related optimization methods. Transportation systems can be easily modeled as multicommodity flow networks, as explained by Ahuja et al. (1993). Furthermore, Ahuja et al. (2005) use a special type of multicommodity flow network, called the “space-time” network for assigning locomotives to freight trains such that each train obtains the required power and the railroad’s operating costs are minimized. Dessouky et al.
(2006) develop a branch and bound procedure to determine the optimal dispatching times for trains traveling in complex rail networks. In the similar direction, a study of crew assignment for railroads has been undertaken by Vaidyanathan *et al.* (2007); this study also models the problem as an optimization problem based on the multicommodity flow network. Multicommodity flow networks find applications in other domains of transportation problems as well, such as air traffic control systems (Helme 1992), and mass-evacuation of a city to safe destinations using transportation infrastructure (Chiu *et al.* 2007).

In this paper, we study the problem of scheduling vehicles on a grid of intersecting roads. We develop heuristic solutions for this problem based on the clock-driven scheduling approach used in real-time scheduling systems. To augment the existing work, we develop an optimization model for this problem using a special type of multicommodity network, called the “space-time” network. We demonstrate the superiority of space-time network to solve this problem over another popular optimization model, namely, the integer programming method. Finally, we evaluate our heuristic solutions against the optimal solution and show that they are very efficient.

### 3. Model Description and Complexity

We study the problem of scheduling vehicles on a grid of intersecting roads. We consider a grid formed by the intersections of $V$ north-south roads and $H$ east-west roads, which intersect in $V \times H$ intersections. The objective of this study is to compute a schedule at the intersections such that the average travel time of vehicles when they travel through the grid is minimized. We refer to this problem as the “Vehicle Scheduling Problem.”

We begin with the basic assumptions of our model. First, we assume that each road of the grid is bi-directional with only one lane per direction. This is not a restriction because a multi-lane road can be mapped to an equivalent single lane road, as long as we do not consider lane changes. It is a reasonable assumption because intelligent transportation systems strive to minimize lane changes by suitably assigning lanes to vehicles (Varaiya 1993). Another assumption of our model is that we do not consider turning movements at the intersections. In other words, once a vehicle enters the grid on any road, it continues to travel on the same road and exit from the grid. This model could be viewed as a good first-step
approximation when the number of vehicles turning at the intersections on the grid is not high.

In this section, we first present a mixed integer programming (MIP) formulation of the problem as we explained in Shah et al. (2007). Then, we prove that the vehicle scheduling problem is strongly NP-hard. Hence, it is unlikely to obtain an optimal solution within a reasonable amount of time (Garey and Johnson 1979). In this section, we also present a more efficient formulation based on the Space-Time Network.

3.1. Mixed Integer Programming Formulation

We assume that a total of $N$ vehicles enter the system. Each vehicle $i$ arrives at an arbitrary time $S_i$. Each vehicle $i$ requires time to travel a distance equal to its length on intersection $j$; we use $e_{ij}$ to represent this time. We represent the travel time of each vehicle $i$ while traveling between consecutive intersections $m$ and $(m+1)$ as $t_{im(m+1)}$. To prevent collisions between vehicles traveling in the same direction, we define a parameter $g$ that corresponds to delay required while scheduling these vehicles such that they maintain a minimum safe distance with each other. Similarly, whenever a phase change occurs at an intersection, there is usually some delay before the vehicles from another direction start to move. We define a parameter $q$ to represent this context switch delay. The difference between $q$ and $e_{ij}$ is illustrated in Fig. 1.

As shown in Fig. 1(a), at time $T = t$, vehicle $v_i$ reaches the edge of the intersection. At time $T = t + e_{ij}$, the vehicle $v_i$ travels the distance equal to its length on the intersection. The next vehicle on the same direction can be allowed to begin crossing the intersection at time $T = t + e_{ij} + g$, so that these vehicles maintain the necessary safe distance with each other. Finally, at time $T = t + q$, vehicle $v_i$ leaves the intersection. Thus, $q$ represents the time a vehicle would take to cross the intersection. Vehicles from an orthogonal direction can be allowed to cross only after this time interval.
We use the term “conflicting directions” whenever vehicles from two different directions cannot be permitted to cross the intersection concurrently (e.g., vehicles traveling in east-west direction cannot be permitted to cross when vehicles from north-south direction are traveling across the intersection). Similarly, we use the term “safe directions” whenever vehicles from two different directions can be permitted to cross the intersection concurrently (e.g., vehicles traveling in the north-south direction can cross an intersection concurrently with the vehicles traveling in the south-north direction).

We use the following variables in this model. Variable $P_{ij}$ represents the time when vehicle $i$ is scheduled to cross intersection $j$. Binary variable $X_{ijl}$ determines if vehicle $i$ is the $l^{th}$ vehicle on intersection $j$. We use another binary variable $\gamma_{j(i_{(l+1)})k}$ to determine if the $l^{th}$ and the $(l+1)^{st}$ vehicles scheduled on intersection $j$ are both traveling in the same direction $k$. Finally, binary variable $\beta_{j(i_{(l+1)})k}$ determines if the $l^{th}$ vehicle on intersection $j$ is from direction $k$ and the $(l+1)^{st}$ vehicle is from a “safe direction” with respect to $k$ (i.e., direction $A_k$). Now, we list the notations and explain our formulation.

**Notations:**

**Input Parameters:**

$N$ Total number of vehicles

$W$ Total number of intersections

$D$ Set of all directions, i.e., $D = \{NS, SN, EW, WE\}$
Variables:

- \( A_k \): Safe direction with respect to direction \( k \)
- \( \eta_i \): Set of intersections visited by vehicle \( i \)
- \( \phi_i \): Final intersection visited by vehicle \( i \) on its path
- \( S_j \): Arrival time of \( i^{th} \) vehicle in the system
- \( e_{ij} \): Time required by vehicle \( i \) to travel a distance equivalent to its length
- \( t_{im(m+1)} \): Travel time of vehicle \( i \) between \( m^{th} \) and \((m+1)^{st}\) intersections on its path
- \( g \): Time corresponding to inter-vehicle gap when consecutive vehicles are scheduled from the same direction
- \( q \): Time interval corresponding to context-switch delay at intersection. This time interval also represents the time a vehicle takes to completely cross the intersection
- \( \psi_{jNS} \): Set of vehicles approaching intersection \( j \) from north-south direction
- \( \psi_{jSN} \): Set of vehicles approaching intersection \( j \) from south-north direction
- \( \psi_{jEW} \): Set of vehicles approaching intersection \( j \) from east-west direction
- \( \psi_{jWE} \): Set of vehicles approaching intersection \( j \) from west-east direction
- \( \psi_j = \psi_{jNS} \cup \psi_{jEW} \cup \psi_{jSN} \cup \psi_{jWE} \)
- \( \delta_{ijm} \): 1 if intersection \( j \) is the \( m^{th} \) intersection on path of vehicle \( i \), 0 otherwise

The objective function can be expressed as:
Minimize \( \sum_{i=1}^{N} (P_{ij} - S_i) \)

Subject to:

\[ \sum_{l \in \psi_i} X_{ijl} = 1; i = 1, 2, \ldots, N; \forall j \in \eta_i \]

(1)

\[ \sum_{i \in \psi_j} X_{ijl} = 1; j = 1, 2, \ldots, W; \forall l \in \psi_j \]

(2)

\[ \sum_{i \in \psi_j} P_j X_{ij(l+1)} \geq \sum_{i \in \psi_j} P_j X_{ijl} + e_{ij} + g \sum_{k \in D} \gamma_{ijl} - q(1 - \sum_{k \in D} \beta_{ijl}) \]

(3)

\[ j = 1, 2, \ldots, W; l = 1, 2, \ldots, \psi_j \]

\[ \sum_{k \in D} \gamma_{ijl} \leq 1; j = 1, 2, \ldots, W; l = 1, 2, \ldots, \psi_j \]

(4)

\[ 2 \sum_{i \in \psi_j} X_{ijl} + X_{ij(l+1)} \geq 4 \gamma_{ijl} - 1; j = 1, 2, \ldots, W; l = 1, 2, \ldots, \psi_j; \forall k \in D \]

(5)

\[ \sum_{i \in \psi_j} X_{ijl} + X_{ij(l+1)} \leq \gamma_{ijl} + 1; j = 1, 2, \ldots, W; l = 1, 2, \ldots, \psi_j; \forall k \in D \]

(6)

\[ 2 \left( \sum_{i \in \psi_j} X_{ijl} + \sum_{i \in \psi_j} X_{ij(l+1)} \right) \geq 4 \beta_{ijl} - 1; j = 1, 2, \ldots, W; l = 1, 2, \ldots, \psi_j; \forall k \in D \]

(7)

\[ \sum_{i \in \psi_j} X_{ijl} + \sum_{i \in \psi_j} X_{ij(l+1)} \leq \beta_{ijl} + 1; j = 1, 2, \ldots, W; l = 1, 2, \ldots, \psi_j; \forall k \in D \]

(8)

\[ \sum_{j \in \eta_i} P_j \delta_{ij(m+1)} \geq \sum_{j \in \eta_i} P_j \theta_{ijm} + \delta_{ijm} + t_{ijm} \]

(9)

\[ X_{ijl} \in \{0,1\}; P_j \geq 0; \beta_{ijl} \in \{0,1\}; \gamma_{ijl} \in \{0,1\} \]

(10)

The objective function minimizes the total transit time (and, hence the average transit time) of vehicles. Constraint set (1) ensures that each vehicle is scheduled exactly once on every intersection on its path. Constraint set (2) enforces the condition that only one vehicle is the lth vehicle on a given intersection. If two vehicles from “safe directions” are scheduled concurrently, we consider them as distinct vehicles, i.e., we consider them as lth and (l+1)st vehicles on that intersection.
Constraint set (3) computes the time when the \((l+1)\)st vehicle can be scheduled at an intersection after the \(l\)th vehicle has been scheduled. Considering the direction of the \((l+1)\)st vehicle, there are three possibilities, namely, the \((l+1)\)st vehicle is from the same direction as the \(l\)th vehicle, it is from the “safe direction” with respect to the direction of the \(l\)th vehicle, or it is from the “conflicting direction” set for the \(l\)th vehicle. When both the \(l\)th and \((l+1)\)st vehicles are from the same direction, it is necessary to add a small delay \(g\) while scheduling the next vehicle. \(g\) corresponds to the inter-vehicle gap. When the \((l+1)\)st vehicle belongs to the “safe direction” (e.g. north-south and south-north; or east-west and west-east) with respect to the direction of the \(l\)th vehicle, then both these vehicles can be simultaneously scheduled. However, if these vehicles have “conflicting directions” (e.g. north-south and east-west, north-south and west-east, etc.) with each other, then the \((l+1)\)st vehicle can be scheduled only after a pre-selected context switch loss time \(q\). This delay \(q\) represents the time the \(l\)th vehicle would take to completely travel through the intersection. Constraint set (3), along with constraint sets (4) through (8), computes the scheduling time for the \((l+1)\)st vehicle. Constraint set (4) ensures that two consecutive vehicles scheduled on a given intersection can have either same direction, “safe” direction, or have “conflicting” directions with each other. Constraint sets (5) and (6) specify that the binary variable \(\gamma_{jl(l+1)k}\) is set to one, if and only if the \(l\)th and \((l+1)\)st vehicles at intersection \(j\) have the same direction \(k\). Constraint sets (7) and (8) specify that the binary vehicle \(\beta_{jl(l+1)k}\) is one, if and only if the \(l\)th and \((l+1)\)st vehicles belong to the “safe” direction set at intersection \(j\).

Constraint set (9) enforces the requirement that once a vehicle is scheduled on a given intersection, it cannot be scheduled on its next intersection at least for the time duration equal to the time required to cross the intersection plus the travel time on the road segment between these intersections. Constraint set (10) specifies the bounds on the values for the variables.

Clearly, constraint (3) cannot be solved by linear solvers since it involves non-linear terms. Hence, we linearize this constraint by introducing variable \(Z_{ijl}\), where \(Z_{ijl} = P_{ij}X_{ijl}\) and another term \(M = \sum_{j=1}^{N} \sum_{j\in j} \left[ e_{ij} + t_{ij(j+1)} \right] + (N-1)q\). Here, \(M\) is an upper bound on the value of \(P_{ij}\). Now, the constraints (3B) through (3D) shown below guarantee that \(Z_{ijl} = P_{ij}\), if and only...
if vehicle $i$ is the $l^{th}$ vehicle on intersection $j$ (i.e. $X_{ijl} = 1$); otherwise $Z_{ijl} = 0$. Therefore, we replace constraint (3) by constraints (3A) through (3D).

$$\sum_{i \in \psi_j} Z_{ij(l+1)} \geq \sum_{i \in \psi_j} Z_{ij} + e_j + g \sum_{k \in D} \gamma_{j(k+1)} + q(1 - \sum_{k \in D} \beta_{j(k+1)} - \sum_{k \in D} \gamma_{j(k+1)});$$

$$j = 1, 2, \ldots, W; \forall l \in \psi_j$$

(3A)

$$Z_{ijl} \leq P_{ij}; i = 1, 2, \ldots, N; \forall j \in \eta_i; l = 1, 2, \ldots, \psi_j$$

(3B)

$$Z_{ijl} \leq M X_{ijl}; i = 1, 2, \ldots, N; \forall j \in \eta_i; l = 1, 2, \ldots, \psi_j$$

(3C)

$$Z_{ijl} \geq P_{ij} - (1 - X_{ijl}) M; i = 1, 2, \ldots, N; \forall j \in \eta_i; l = 1, 2, \ldots, \psi_j$$

(3D)

### 3.2. Complexity of the Problem

In this section, we show that the vehicle scheduling problem is strongly NP-hard by presenting a polynomial transformation of a known strongly NP-hard problem to an instance of this problem. Here, the known NP-hard problem is the problem of scheduling non-preemptive jobs with arbitrary release times on a processor (Pinedo 1995).

We consider an instance of vehicle scheduling problem with only one intersection. In this transformation, each vehicle corresponds to a non-preemptive job, and the intersection corresponds to the processor. In terms of job scheduling, the arrival time of a vehicle on an intersection corresponds to the release time of a job. Moreover, each vehicle has a crossing time, which represents the time a vehicle would take to travel through the intersection. Correspondingly, each job has an execution time on the processor. Hence, the problem of scheduling non-preemptive jobs (having arbitrary release times) on a processor can be solved by solving this instance of vehicle scheduling problem. Now, we present the theorem and its proof.

**Theorem 1**: The vehicle scheduling problem is strongly NP-hard.

**Proof**: Consider the following decision problem corresponding to the vehicle scheduling problem:
**Decision Problem (VS):** Consider a set $\mathcal{R}_{NS} = \{1, 2, \ldots, V\}$ of north-south roads, a set $\mathcal{R}_{EW} = \{1, 2, \ldots, H\}$ of east-west roads that intersect in $I = \{1, 2, \ldots, V \times H\}$ intersections, where $N$ vehicles arrive. Consider a set $\mathcal{V}_j = \{1, 2, \ldots, n_j\}$ of vehicles arriving at intersection $j$. Each vehicle $i$ has an arbitrary arrival time $S_i$ in the system, and a crossing time $q$ for each intersection $j$ on its path. Does there exist a schedule of vehicles such that $\sum_{i \in \mathcal{N}} F_i \leq X$, where $\sum_{i \in \mathcal{N}} F_i$ is the sum of exit times (time when vehicle $i$ leaves the grid) of all $N$ vehicles, and $X$ is a positive real number? Here, a vehicle $i$ on a given road can cross an intersection $j$ only when all the vehicles reaching intersection $j$ on the same road before vehicle $i$ have crossed the intersection. Hence, there are chain precedence constraints among the vehicles.

Now consider the decision problem corresponding to the non-preemptive single-machine scheduling problem with chain precedence constraints (where number of chains = 2) and arbitrary release times. This problem is strongly NP-hard, because a special case of this problem, i.e., the non-preemptive single-machine scheduling problem with arbitrary release times, has been shown to be strongly NP-hard (Kan 1976), (Pin1995).

**Decision Problem (NPS):** Consider a single processor $Q$ and a set $\mathcal{J}$ of non-preemptive jobs, where $\mathcal{J} = \{1, 2, \ldots, m\}$. Each job $i$ has an arbitrary release time $r_i$ and a processing time $p_i$, and there exists a set of chain precedence constraints with number of chains equal to 2. Does there exist a schedule of jobs such that $\sum_{i \in \mathcal{J}} C_i \leq K$, where $\sum_{i \in \mathcal{J}} C_i$ is the sum of completion times of all $m$ jobs and $K$ is a positive real number?

From an arbitrary instance of the NPS problem, we construct an instance of the VS problem using the following transformation:

Let $\mathcal{R}_{NS} = \{1\}$ and $\mathcal{R}_{EW} = \{1\}$. Let these roads intersect once. Hence, $I = \{1\}$. Moreover, $n_i = m$, where $n_i$ is the number of vehicles arriving at the intersection, and $m$ represents the number of non-preemptive jobs. Also, $S_i = r_i$ and $q = p_i$; $i = 1, 2, \ldots, m$; $j = 1$. The vehicles on the north-south road arrive at the intersection following the precedence
constraint of the first chain. Similarly, the vehicles on the east-west road arrive at the intersection following the precedence constraint of the second chain.

Therefore, when $X = K$, an instance of the NPS problem corresponds to the given instance of the VS problem. Since these problems are equivalent, the solution of this instance of VS problem provides the solution for NPS problem and vice-versa. Hence, the vehicle scheduling problem is strongly NP-hard.

3.3. The Space-Time Network Formulation

The optimization model developed using mixed integer programming (MIP) formulation provides an optimal solution for the vehicle scheduling problem. However, given the strongly NP-hard nature of this problem, we could solve the MIP formulation of this problem optimally for very small sized problem instances, as we show in Section 5.

Hence, we develop another optimization model as explained in Shah et al. (2008a) for this problem using a type of multicommodity network flow called the “space-time network,” which has been used to model different scheduling problems in the past studies as discussed earlier. We show in Section 5, that it is possible to optimally solve instances of the vehicle scheduling problem with larger data sizes (i.e. having larger number of vehicles) using this method. Now, we present the “space-time network” model of the vehicle scheduling problem.

We model this problem as a flow of vehicles (commodities) on different arcs of the space-time network. In this model, each node represents both time and location. Each arc connects two nodes. The length of an arc on the distance axis represents the distance the vehicles travel while flowing over this arc. The length of an arc on the time axis corresponds to the delay experienced by vehicles while traveling on that arc. Hence, each arc represents the distance between two nodes that vehicles may travel, and the delay incurred by the vehicles in traveling between these nodes. There are two types of nodes, namely, the arrival nodes and the departure nodes. Similarly, there are three types of arcs: (i) road arcs, (ii) wait arcs, and (iii) crossing arcs. Each intersection contains a set of arrival and departure nodes (connected by arcs) per direction. A space-time network model for one road for one direction of flow is shown in Fig. 2.
We now describe how these nodes and the arcs enable us to model the vehicle scheduling problem. An arrival node represents the event when one or more vehicles are available to cross the intersection. A departure node represents the event of a vehicle leaving the intersection. Road arcs model the movement of vehicles between adjacent intersections. Crossing arcs model the movement of vehicles when they are crossing the intersection. The wait arcs model the fact that one or more vehicles might be waiting to cross the intersection.

When one or more vehicles reach an intersection (by traveling over a road arc), one of the following happens:

1) One of these vehicles crosses the intersection by traveling on the crossing arc. The remaining vehicles wait at the intersection by joining the next wait arc.

2) All these vehicles wait at the intersection. In this case, all these vehicles join the next wait arc. This can occur if vehicles from a conflicting direction are crossing the intersection at that time.

![Diagram of a space-time network model for one road in one direction](image)

**Fig. 2. The space-time network model for one road in one direction**
The waiting vehicles keep joining subsequent wait arcs until they are eventually scheduled on the crossing arc. When a vehicle reaches the departure node, it travels to the next intersection by traveling on the corresponding road arc.

In the mixed integer programming formulation of the vehicle scheduling problem, we explained that if vehicles from the same direction are scheduled consecutively, it is essential to add a small delay $g$ corresponding to the minimum safe distance between vehicles. By suitably selecting the length of wait arcs, we can model this behavior. To ensure safety between conflicting flows of vehicles at any intersection, we introduce additional constraints, which we explain in the formulation below. We start with the notations used to formulate this problem, and then describe the formulation.

**Notations:**

- **WaitArcs** The set of wait arcs for the entire network
- **CrossingArcs** The set of crossing arcs for the entire network
- **RoadArcs** The set of road arcs for the entire network
- **AllArcs** The set of all the arcs of the network
- **Intersections** The set of all the intersections in the network
- **Directions** The set of all the directions in the network
- **ArrivalNodes** $[k, l]$ The set of arrival nodes at intersection $k$ for the flow in direction $l$
- **Conf** $[l]$ The set of conflicting directions for direction $l$
- **Next** $k$ The intersection after intersection $k$ in the given direction of flow

- $x_i$ The flow (total number of vehicles) on arc $i$
- $t_i$ Propagation delay (travel time) on arc $i$
- $r_{jkl}$ $j$th road arc terminating at the $j$th arrival node of intersection $k$ and originating from the $j$th departure node of the previous intersection for the flow in direction $l$
- $w_{jkl}$ $j$th wait arc connecting the $j$th arrival node to the $(j + 1)$th arrival node on intersection $k$ for the flow in direction $l$
- $c_{jkl}$ $j$th crossing arc connecting the $j$th arrival node to the $j$th departure node at intersection $k$ for the flow in direction $l$
Loss of time during context switch while permitting vehicles from a conflicting flow to start crossing the intersection.

The formulation for the space-time network model is presented next.

Minimize $Z = \sum_{i \in \text{AllArcs}} t_i x_i$

Subject to:

$x_i \leq 1; \forall i \in \text{CrossingArcs}$ \hspace{2cm} (11)

$x_{r_jl} + x_{w_{j,l,j+1}} = x_{c_{j,l}} + x_{w_{j,l+1}}; \forall k \in \text{Intersections}; \forall l \in \text{Directions}; \forall j \in \text{ArrivalNodes}[k,l]$ \hspace{2cm} (12)

$x_{c_{j,l}} = x_{r_{j,\text{Next}[k,l]}}; \forall k \in \text{Intersections}; \forall l \in \text{Directions}; \forall j \in \text{ArrivalNodes}[k,l]$ \hspace{2cm} (13)

$x_{c_{j,l}} + x_{w_{j,k}} \leq 1; \forall k \in \text{Intersections}; \forall l \in \text{Directions}; \forall j \in \text{ArrivalNodes}[k,l]; \forall s \in \text{Conf}[l]; \forall p = j, j+1, j+2, \ldots, j+(q/t_{w_{j,l}})$ \hspace{2cm} (14)

The objective function minimizes the total transit time (hence, the average transit time) of the vehicles. Constraint set (11) assigns the capacity of a crossing arc to at most one vehicle. Constraint sets (12) and (13) are the flow balance equations. Constraint set (12) specifies that all the vehicles arriving at an arrival node, either through a road arc or a wait arc from the previous arrival node, must either travel on the crossing arc or join the next wait arc. Constraint set (13) ensures that the vehicles that leave the intersection (by traveling on the crossing arc), continue on the road arc to arrive at the next intersection. As we explained before, whenever a context switch (i.e. phase change) occurs at an intersection, there is usually some context switching delay $q$ before the vehicles from another conflicting direction are allowed to travel through the intersection. This delay is necessary to ensure that the last vehicle from the previous direction completely crosses the intersection before the next flow starts. In the space-time network model, we model this delay by ensuring that a new flow waits for $q$ time units (or $q/t_{w_{j,l}}$ wait arcs) intervals once a context switch occurs. This behavior is modeled by constraint set (14).
4. Heuristic Algorithms

Since the vehicle scheduling problem is strongly NP-hard (as shown in Section 3.2), now we propose two heuristics to solve this problem. In order to develop these heuristics, we use the clock-driven scheduling approach for cyclic jobs explained by Liu (2000) and the concept of platoons developed by Varaiya (1993).

Clock-driven scheduling method is a static time-driven scheduling method. The jobs are scheduled at pre-determined time intervals. The jobs have pre-determined period and execution times. The period represents the time interval between the release times of consecutive instances of a job. The execution time is the time interval for which a job runs on the processor. Based on these two parameters, a schedule can be constructed offline. The clock-driven approach is deterministic, and hence it is easy to guarantee hard real-time constraints. In clock-driven scheduling, the scheduling sequence repeats periodically. This period is called the hyperperiod. More details about clock-driven scheduling method are explained by Liu (2000).

In the proposed heuristics, we generate virtual platoons of specified execution times and specified periods for each road for each direction in the grid as we explained in Sha (2006). Whenever, a vehicle enters the system, it must join the next available virtual platoon on that road (for the specified direction), and travel through the grid as a part of that virtual platoon. Now, the execution time of a virtual platoon determines the length of the platoon which, in turn, determines the maximum number of vehicles a platoon can accommodate. By varying the period and the execution time of these virtual platoons, we develop two algorithms, namely, the “Fixed platoon length, variable period” algorithm and the “Fixed period, variable platoon length” algorithm. Both these algorithms require that the estimated mean arrival rate on each road is known apriori. Hence, we define a set \( R_{NS} = \{1, 2, \ldots, v\} \) consisting of north-south and south-north direction roads with each road having an average arrival rate \( \{a_1, a_2, \ldots, a_v\} \) respectively, and a set \( R_{EW} = \{1, 2, \ldots, h\} \) of east-west and west-east direction roads with average arrival rate \( \{b_1, b_2, \ldots, b_h\} \) respectively. We also define Intersections\((r)\) as the set of intersections for road \( r \), \( \forall r \in R_{NS} \cup R_{EW} \). Now, we describe both these algorithms below.
4.1. Fixed Platoon Length, Variable Period Algorithm

In this algorithm, we generate virtual platoons of the same length, and vary their periods. The length of the platoon determines its execution time (i.e., the time a platoon takes to travel through the intersection). Hence, we set \( E_i = t, \forall i \in R_{NS} ; E_l = t, \forall l \in R_{EW} ; t \in \mathbb{R}^+ \), where \( E_i \) and \( E_l \) are the execution times of the platoon on roads \( i \) and \( l \), respectively; and \( t \) is a positive real number. Let \( m \) be a road in set \( R_{NS} \) for which the arrival rate is maximum across all the roads in that set. Let \( n \) be a road in set \( R_{EW} \) for which the arrival rate is maximum across all roads in that set. Moreover, let \( j \) be the intersection formed by the roads \( m \) and \( n \). When the arrival rate \( a_m \) on road \( m \) is higher than the arrival rate \( b_n \) on road \( n \) such that \( a_m = k \cdot b_n \), we select period of a platoon on road \( n \) as \( P_n = k \cdot P_m ; m \in R_{NS} ; n \in R_{EW} \), where \( P_m \) is the period of a platoon on road \( m \). Similarly, when \( b_n \) is higher than \( a_m \) such that \( b_n = k \cdot a_m \), we select \( P_m = k \cdot P_n ; m \in R_{NS} ; n \in R_{EW} \).

As per the clock-driven scheduling principle (Liu 2000), a feasible solution exists if and only if the schedulable utilization of an intersection is not more than 1, i.e., \( t/P_m + t/P_n \leq 1 \). On solving this, we obtain \( P_m \geq t(1+k)/k \) and \( P_n = k \cdot P_m \) when \( a_m \) is higher than \( b_n \). Similarly, when \( b_n \) is higher than \( a_m \), we obtain \( P_n \geq t(1+k)/k \) and \( P_m = k \cdot P_n \). Hence, we select appropriate values for \( P_m \) and \( P_n \) that satisfy these equations. Moreover, we set \( P_i = P_m ; \forall i \in R_{NS} \), \( P_i = P_n ; \forall l \in R_{EW} \), and hyperperiod \( H = \text{LCM}(P_m, P_n) \) for the system. Now, we compute a clock-driven schedule as explained by Liu (2000); for brevity, we omit the details of computing the clock-driven schedule. The complete algorithm is described below.

**Step 1:** Set the length of platoon on each road \( r \), \( \text{len}_r = L, \forall r \in R_{NS} \cup R_{EW} \).

**Step 2:** Set the execution time of platoon on each road \( r \), \( E_r = t, \forall r \in R_{NS} \cup R_{EW} \), where \( t = (W + L)/\text{Speed} \). Here, \( W \) represents the width of an intersection and \( \text{Speed} \) represents the speed limit (maximum allowed speed).
**Step 3:** Set $m$ to the index of a road $i$ in set $R_{NS}$ having the maximum arrival rate.

**Step 4:** Set $n$ to the index of a road $l$ in set $R_{EW}$ having the maximum arrival rate.

**Step 5:** Compare the arrival rate $a_m$ on road $m$ with the arrival rate $b_n$ on road $n$.

**Step 6:** If $a_m > b_n$, then set $k = a_m / b_n$. Set the period of platoons on road $m$, $P_m = \lceil t(1+k) / k \rceil$. Set the period of platoons on road $n$, $P_n = \lceil k \cdot P_m \rceil$. Go to step 8.

**Step 7:** If $b_n \geq a_m$, then set $k = b_n / a_m$. Set the period of platoons on road $n$, $P_n = \lceil t(1+k) / k \rceil$. Set the period of platoons on road $m$, $P_m = \lceil k \cdot P_n \rceil$.

**Step 8:** For each road $i \in R_{NS}$, set the period of the platoon $P_i = P_m$. Similarly, for each road $l \in R_{EW}$, set the period of the platoon $P_i = P_n$.

**Step 9:** Set the hyperperiod $H = \text{LCM}(P_m, P_n)$.

**Step 10:** Compute a clock-driven schedule using the hyperperiod $H$, the period for platoon on each road, and the execution time of platoons on each road (Liu 2000).

Clearly, the complexity of this algorithm is $O(v+h)$. Therefore, it is a polynomial-time algorithm.

### 4.2. Fixed Period, Variable Platoon Length Algorithm

In this algorithm, we use a fixed period for all the platoons and vary their lengths (execution times). To compute a period for all the platoons, we first compute the hyperperiod $H$ as $H = \text{LCM}\{a_1, a_2, \ldots, a_v, b_1, b_2, \ldots, b_h\}$. We select $H$ as the period of all the platoons on all the roads, i.e., $P_i = H, \forall i \in R_{NS}; P_i = H, \forall l \in R_{EW}$, where $P_i$ and $P_l$ are the periods of the platoon on the roads $i$ and $l$ respectively.

To determine the execution time of each platoon, we first compute the execution time of virtual platoons on the north-south/south-north and the east-west/west-east direction for each intersection $j$ in the grid as shown below:

\[
e_{j_{NS}} = H \cdot (a_i / (a_i + b_i))
\]

\[
e_{j_{EW}} = H \cdot (b_i / (a_i + b_i))
\]
where $e_{jNS}$ is the execution time for a virtual platoon on a road $i$ in set $R_{NS}$ at intersection $j$, $e_{jEW}$ is the execution time for a virtual platoon on a road $l$ in set $R_{EW}$ at intersection $j$, and $j$ is the intersection of roads $i$ and $l$.

Since different roads have different arrival rates, we need to ensure that when we select the execution time of a platoon for a given road, it must be schedulable on all the intersections on that road. In other words, the execution time of a virtual platoon on a given road must be equal to the least execution time for that direction over all the intersections on that road. Therefore,

$$E_i = \min\{e_{jNS}\}; \forall j \in \text{Intersections}(i); \forall i \in R_{NS}$$

$$E_l = \min\{e_{jEW}\}; \forall c \in \text{Intersections}(l); \forall l \in R_{EW},$$

where $E_i$ is the execution time of a virtual platoon on road $i \in R_{NS}$ and $E_l$ is the execution time of a virtual platoon on road $l \in R_{EW}$.

Given the period and execution time for each platoon for each road and the hyperperiod, a clock driven schedule can be easily computed (Liu 2000). From the execution time of each platoon, the capacity of the platoon (i.e., the maximum number of vehicles platoon can contain) can also be computed. The complete algorithm is described below.

**Step 1:** Set hyperperiod $H = \text{LCM}(a_1, a_2, \ldots, a_r, b_1, b_2, \ldots, b_h)$.

**Step 2:** For each road $r \in R_{NS} \cup R_{EW}$, set Period $P_r = H$.

**Step 3:** For each intersection $j$ in the grid formed of road $i \in R_{NS}$ and road $l \in R_{EW}$, set execution time for north-south/south-north platoons at intersection $j$,

$$e_{jNS} = H \cdot (a_i/(a_i + b_j)).$$

Set the execution time for the east-west/west-east platoons at intersection $j$,

$$e_{jEW} = H \cdot (b_j/(a_i + b_j)).$$

**Step 4:** For each road $i \in R_{NS}$, set execution time for platoon $E_i$ as

$$E_i = \min\{e_{jNS}, \forall j \in \text{Intersections}(i)\}.$$

**Step 5:** For each road $l \in R_{EW}$, set execution time for platoon $E_l$ as

$$E_l = \min\{e_{jEW}, \forall c \in \text{Intersections}(l)\}.$$
Step 6: Set platoon length for each road \( r \in R_{NS} \cup R_{EW} \), \( len_r = Speed \cdot E_r - W \), where \( W \) is the width of the intersection and \( Speed \) represents the speed limit (maximum allowed speed).

Step 7: Using the period of virtual platoons for each road, the execution times of the platoons, and the hyperperiod, compute a clock-driven schedule as explained by Liu (2000).

It is easy to see that the complexity of this heuristic is \( O(h \cdot v) \). Hence, it is a polynomial-time algorithm.

5. Experimental Studies and Implications

In this section, we first compare the effectiveness of two optimization methods, namely, the MIP formulation and the space-time network formulation. Thereafter, we evaluate the performance of the heuristics proposed in Section 4 with the optimal solution.

5.1. Evaluation of Optimization Methods

We used CPLEX 8.1 (2002) to solve the MIP formulation and the space-time network formulation. Given the strongly NP-hard nature of this problem, we could not find optimal solutions for larger road network grids. Therefore, we restricted the road network to only a 2x2 grid, having a total of 4 intersections. Moreover, we considered only one lane per each road, where the flow of traffic was either flew the north to south, or east to west directions. For the mixed integer programming formulation, CPLEX could find a solution for only 14 vehicles. However, for the space-time network formulation, CPLEX could find an optimal solution for 200 vehicles. The evaluation of these optimization methods is presented in Fig. 3.
To solve a mixed integer programming (MIP) formulation, CPLEX initially computes a lower bound on the optimal solution and a feasible solution. The optimal solution value is always higher than or equal to the lower bound value in the minimization problem. The difference between the lower bound and the feasible solution is called the “gap.” CPLEX reduces this gap over time by increasing the value of lower bound and reducing the feasible solution value. When the gap becomes zero, then the feasible solution is guaranteed to be the optimal solution, as we explained in Sha (2007).

For the MIP formulation, CPLEX could not compute the optimal solution. Hence, in Fig. 3, we present the values of lower bounds where the gap was less than or equal to 1%. The computation time was about 24 hours to find the lower bound having a gap of 1% or less. As we discussed earlier, we could find such accurate lower bounds for very small problem sizes, i.e., problem instances having not more than 14 vehicles in the grid and arrival rates not greater than 2 vehicles/sec. When arrival rate exceeded 2 vehicles/sec, even after 24 hours, CPLEX could not find the value of lower bound that where gap was less than 5%. Hence, we
did not report these results in the graph. These limitations of MIP formulation are overcome by the space-time network formulation. Since the MIP formulation increases rapidly in size with the increase in the number of vehicles in the grid, computation time increases exponentially with the increase in the number of vehicles. In contrast, the space-time formulation is independent of the number of vehicles in the grid, and therefore it does not increase in size if the number of vehicles increases. Since we could optimally solve the problem for 200 vehicles, even for higher arrival rates with the space-time network formulation, we use the space-time formulation for the rest of the study.

5.2. Evaluation of Heuristic Algorithms

Although the space-time network formulation can solve larger-size problems, it cannot solve arbitrarily large problem instances because this problem is strongly NP-hard. Hence, we need to deploy heuristic solutions. Now, we evaluate the performance of the proposed heuristics by comparing them with the optimal solution obtained using the space-time network formulation, and a base case. For the base case, we select a fixed-interval intersection algorithm implemented by Berger (1980), where, the traffic light phase switching is performed at predetermined intervals, independent of the expected traffic on the roads. Since we are evaluating the heuristic algorithms against the optimal solution, we consider 2x2 grid of intersecting roads. We computed the average transit time of the heuristics over a period of 2 hours. The simulation results are presented in Fig. 4.
From Fig. 4, it is evident that the heuristics perform significantly better than the base case. Moreover, these heuristics are very close in performance with the optimal solution for the low to moderate arrival rates. In the “fixed period, variable platoon length” algorithm, there are less context switches within a hyperperiod as compared to the “fixed platoon length, variable period” algorithm. Thus, less time is lost during phase switching in the former algorithm, which results in virtual platoons having higher capacity. Hence, at higher arrival rates, the “fixed period, variable platoon length” algorithm outperforms the other heuristic. However, at lower arrival rates, the “fixed platoon length, variable period” algorithm performs marginally better because the waiting time of vehicles to join the next available platoon is reduced as the virtual platoons have smaller period.

We also observe in Fig. 4 that when the vehicle arrival rate exceeds 2.3 vehicles/sec, both the algorithms perform poorly compared to the optimal solution value. This implies that it is possible to further reduce the travel time of the vehicles by deploying more intelligent algorithms to control the intersections. In addition, this study helps the city planners in
understanding the saturation point of the system, where deploying an intelligent solution will not improve the travel time. It is important for them to identify the saturation point, because the only alternative beyond this point is to increase the infrastructure capacity, possibly by building more roads. This information is captured in Fig. 3. At the arrival rate of 5 vehicles/sec, the optimal solution value computed by the space-time network formulation rapidly degenerates. It implies that at this arrival rate, the only possible way to improve the travel time is to increase the infrastructure capacity.

Thereafter, we study the performance of the proposed heuristics with increasing grid sizes. The results are presented in Figs. 5 and 6. From these figures, it is evident that the performance of these heuristics does not degenerate appreciably with the increase in grid size. Therefore, these heuristics are scalable, and hence suitable for real-world scenarios having larger grid sizes. Moreover, the saturation point (where the performance rapidly degenerates) is almost the same for all the grid sizes. This is an important result for the managers, which indicates that the effectiveness of proposed heuristics does not worsen with an increase in the grid size. Also, both of the heuristics obtain the solution in less than a second even for large grid sizes and high arrival rates. Hence, we do not report the exact computation time.

![Effect of Increasing Grid Size on Fixed Platoon Length, Variable Period Algorithm](image)

**Fig. 5.** Effect of increasing grid size on fixed platoon length, variable period algorithm
To determine the effectiveness of the heuristics for larger sized problems, we also evaluate the heuristics against the lower bound of the solution for larger grid sizes. To compute the lower bound for this problem, we compute the minimum transit time when only one vehicle is traveling through the grid. To compute this value of lower bound, we assume that the vehicle moves at the maximum permissible speed, namely, the speed limit. In addition, computing the minimum transit time implies that this vehicle does not incur any delay at the intersections. In other words, all the traffic lights have the “green” phase on the road that the vehicle is traveling on. We refer to this lower bound as 1-vehicle lower bound.

Using the 1-vehicle lower bound value, we compare the average transit time calculated by the proposed heuristics. Table 1 reports the percentage difference between 1-vehicle lower bound of the solution and the average transit time reported by the “fixed platoon length, variable period” heuristic. Similarly, Table 2 reports the percentage difference between the lower bound of the solution and the average transit time reported by the “fixed period, variable platoon length” algorithm.

**Fig. 6. Effect of increasing grid size on fixed period, variable platoon length algorithm**

To determine the effectiveness of the heuristics for larger sized problems, we also evaluate the heuristics against the lower bound of the solution for larger grid sizes. To compute the lower bound for this problem, we compute the minimum transit time when only one vehicle is traveling through the grid. To compute this value of lower bound, we assume that the vehicle moves at the maximum permissible speed, namely, the speed limit. In addition, computing the minimum transit time implies that this vehicle does not incur any delay at the intersections. In other words, all the traffic lights have the “green” phase on the road that the vehicle is traveling on. We refer to this lower bound as 1-vehicle lower bound.

Using the 1-vehicle lower bound value, we compare the average transit time calculated by the proposed heuristics. Table 1 reports the percentage difference between 1-vehicle lower bound of the solution and the average transit time reported by the “fixed platoon length, variable period” heuristic. Similarly, Table 2 reports the percentage difference between the lower bound of the solution and the average transit time reported by the “fixed period, variable platoon length” algorithm.
<table>
<thead>
<tr>
<th>Arrival Rate</th>
<th>2x2 Grid</th>
<th>4x4 Grid</th>
<th>8x8 Grid</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>4.53</td>
<td>2.8</td>
<td>1.94</td>
</tr>
<tr>
<td>0.5</td>
<td>4.61</td>
<td>2.78</td>
<td>1.93</td>
</tr>
<tr>
<td>0.9</td>
<td>4.93</td>
<td>2.79</td>
<td>1.99</td>
</tr>
<tr>
<td>1.5</td>
<td>6.51</td>
<td>3.54</td>
<td>2.62</td>
</tr>
</tbody>
</table>

Table 1. Performance of fixed platoon length, variable period algorithm

<table>
<thead>
<tr>
<th>Arrival Rate</th>
<th>2x2 Grid</th>
<th>4x4 Grid</th>
<th>8x8 Grid</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>13.645</td>
<td>8.26</td>
<td>4.54</td>
</tr>
<tr>
<td>0.5</td>
<td>8.56</td>
<td>6.59</td>
<td>6.27</td>
</tr>
<tr>
<td>0.9</td>
<td>7.64</td>
<td>5.6</td>
<td>5.73</td>
</tr>
<tr>
<td>1.5</td>
<td>9.2</td>
<td>6.67</td>
<td>6.29</td>
</tr>
</tbody>
</table>

Table 2. Performance of fixed period, variable platoon length algorithm

Table 1. and Table 2. show that the percentage difference between 1-vehicle lower bound with the heuristic algorithms is small. Therefore, these results show that the performance of the heuristics do not deteriorate with the size of the problem. Please note that the deviation of these heuristics from the optimal solution will be smaller than the values reported in these tables. This is because the gap between the 1-vehicle lower bound and the optimal solution may be significant. We plan to compute more accurate lower bound for the solution in future work.
6. Summary and Future Research Directions

In this paper, we have analyzed the problem of scheduling vehicles on a grid of intersecting roads. We have formulated a mixed integer programming model for this problem and proved that this problem is strongly NP-hard. We have developed another optimization model using the space-time network flow technique, and shown that the space-time network model is a more effective approach for computing an optimal solution as compared to the mixed integer programming model. In addition, we have presented two polynomial-time heuristic algorithms and evaluated them against the optimal solution. Through this process, we have motivated the need to develop optimization models for assessing the best possible utilization of transportation systems. Moreover, these optimization models can be used to motivate the research for better intelligent solutions, and they can also help managers in capacity planning.

Our results indicate that the proposed heuristics provide near-optimal solution for small grid sizes. We could not compare the results for larger grid sizes because it is not possible to obtain the optimal solution for large problems due to strong NP-hard nature of the problem. However, we find that these heuristics are scalable over larger grids. Moreover, these heuristics obtain the result instantly, even for larger problem instances, thereby making them suitable for real-time implementation.

Some future research directions include the development of more effective heuristic algorithms for this problem and extensions to account for more complex traffic scenarios, including turning movements in the model at intersections, supporting emergency vehicles, and ensuring that the system remains deadlock free. Another interesting research direction is to compute an accurate lower bound on the solution, and use that as another benchmark to evaluate the heuristics. Investigation of intelligent coordination strategies and associated real-time communication and networking support as explained in Shah et al. (2008b) for proactively preventing accidents and achieving high levels of capacity utilization are also potential areas of future research.
References


