Int. J. Appl. Math. Comput. Sci., 2013, Vol. 23, No. 1, 183–200
DOI: 10.2478/amcs-2013-0015

AN AUTONOMOUS VEHICLE SEQUENCING PROBLEM AT INTERSECTIONS: A GENETIC ALGORITHM APPROACH

FEI YAN *, MAHJOUB DRIDI **, ABDELLAH EL MOUDNI **

* School of Information Science and Technology
Southwest Jiaotong University, No. 111, Section 1, Northern 2nd Ring Road, Jinniu District, ChengDu, China
e-mail: yf0413@gmail.com

**Systems and Transport Laboratory
University of Technology of Belfort–Montbeliard, Rue Thierry Mieg, Belfort, 90010, France

This paper addresses a vehicle sequencing problem for adjacent intersections under the framework of Autonomous Intersection Management (AIM). In the context of AIM, autonomous vehicles are considered to be independent individuals and the traffic control aims at deciding on an efficient vehicle passing sequence. Since there are considerable vehicle passing combinations, how to find an efficient vehicle passing sequence in a short time becomes a big challenge, especially for more than one intersection. In this paper, we present a technique for combining certain vehicles into some basic groups with reference to some properties discussed in our earlier works. A genetic algorithm based on these basic groups is designed to find an optimal or a near-optimal vehicle passing sequence for each intersection. Computational experiments verify that the proposed genetic algorithms can response quickly for several intersections. Simulations with continuous vehicles are carried out with application of the proposed algorithm or existing traffic control methods. The results show that the traffic condition can be significantly improved by our algorithm.

Keywords: autonomous vehicles, autonomous intersection management, genetic algorithm, dynamic programming, heuristics.

1. Introduction

Transportation has always been a crucial aspect of human civilization, but it is only since the second half of the 20th century that the phenomenon of traffic congestion has become predominant due to the rapid increase in the number of vehicles and in the transportation demand (Hall and Papageorgiou, 1999). Especially over the last decade, traffic congestion attracted extensive attention because of the worldwide energy crisis and environmental concerns.

The conventional method of preventing or reducing congestions in modern cities is based on the traffic signal at intersections. The traffic signal assigns the right-of-way to a stream of vehicles and pedestrians in order to guarantee their safe crossings. The specification of the signal setting (traffic control strategy) has a major impact on intersection capacity and efficiency. Traffic control strategies at intersections generally fall into two basic categories: pre-timed control, which is also called fixed-time control, and semi/fully traffic actuated control. The first category uses the historical data of traffic demand to calculate the signal setting, e.g., the TRANSYT (traffic network study tool) system (Robertson, 1969). The second strategy, the traffic actuated control, which is also known as the traffic-responsive strategy, has been attracting more and more attention since the 1980s. This control strategy makes use of real-time measurement like the traffic flow rate, which is detected by loop detectors, to specify the signal setting in real time. The most notable system of this kind is SCOOT (Split Cycle Offset Optimization Technique) (Hunt, 1982). However, both control strategies are based on the estimation of traffic flow rates.

Since the flow rate is a continuous variable that needs a period of time to be estimated, there are always big deviations between the last computed value and the actual one. Besides, the traffic signal is assigned to a stream of vehicles for a period of time. The traffic control command cannot adapt to the fluctuation of the traffic state quickly. Hence, traffic control at intersections based on traffic signals limits the potential of improving

intersection capacity.

Since the early 1990s, the development in information systems has provided us with an opportunity to overcome these drawbacks. The technology of wireless communication like WiFi, WiMax and 3G enabled Vehicle to Vehicle (V2V) communication and Vehicle Infrastructure Integration (VII), which enforce the link between vehicles, infrastructure and the driving environment (Kato *et al.*, 2002; Gradinescu *et al.*, 2007; Chisalita and Shahmehri, 2002). Besides, advances in the fields of computation and sensor technologies lead to the emergence of fully autonomous vehicles, which take complete control of vehicle operations and eliminate the driver from the control loop, at least in a specific area (Dresner and Stone, 2004; 2006). Various applications of autonomous vehicles have been demonstrated in Europe (Bertolazzi *et al.*, 2010), Japan (Aotani *et al.*, 2002) and the United States (Shladover *et al.*, 1991; Shladover, 2007). Under this background, the concept of Autonomous Intersection Management (AIM) has attracted great interests over the last decade.

In the framework of AIM, autonomous vehicles communicate with each other or with roadside infrastructures to exchange the information of their states for ensuring driver safety and improving travel efficiency. More specifically, based on the VII technology, the roadside infrastructure at intersections, which can be seen as a controller, can communicate with approaching vehicles continuously. Vital vehicle data such as vehicle speed, position and destination are collected by advanced sensors and sent to the controller in real time. Hence, it is possible to elaborate a traffic control strategy with considering vehicles to be independent individuals. In other words, the right-of-way is assigned to each vehicle according to its state and the state of the whole intersection. Only vehicles that have received the right-of-way can get through the intersection. Traffic control aims at determining the vehicle passing sequence, which is a sequence of distributing the right-of-way.

However, to implement this new control method, we mainly face two difficulties:

- First, how to exchange the information among vehicles or between vehicles and roadside infrastructure.
- Second, less studied, how to find an efficient vehicle passing sequence to maximize the traffic throughput at intersections while maintaining driver safety.

For the first difficulty, research on wireless protocols and advanced devices used in autonomous vehicles has been well studied (see, e.g., Dresner and Stone, 2004; Huang and Miller, 2003; Nadeem *et al.*, 2004). The second one deserves more attention. Most research determines the vehicle passing sequence based on a simple control policy "First In First Out" (FIFO) (Dresner and Stone, 2004; 2006; Lachner, 1997). Although this policy requires very low computational cost, it limits the potential of maximizing the capacity of intersection. Recently, researchers have been paying more attention to optimize vehicle passing sequences. Li and Wang (2006) enumerated all feasible vehicle passing sequences and used trajectory planning algorithms to find the most efficient one. However, they admitted that the algorithms were not efficient enough because the complexity increases exponentially with the number of vehicles and lanes. Wu *et al.* (2009) presented a dynamic programming algorithm to get an optimal solution based on AIM, but they only considered a simple isolated intersection with two lanes. Indeed, an intersection consists of a number of approaches and each approach may have one or more lanes, which will render considerable combinations of the passing sequence. Thus, how to reach a satisfactory passing sequence at a relative low computational cost becomes a big challenge.

Besides, another common concern about the strategy based on VII is that it increases the cost of building and maintaining infrastructures at each isolated intersection (Li and Wang, 2006). In fact, since congestion in modern cities is usually caused by several adjacent intersections located in dense street networks, it is natural to extend the control strategy to several adjacent intersections with only one controller, but this change will increase the computational cost.

To solve these difficulties, in our earlier works (Yan *et al.*, 2009; 2010) we presented some useful structural properties of the proposed new control strategy at an isolated intersection. A branch-and-bound algorithm was designed to find an optimal passing sequence for all detected vehicles for the isolated intersection. However, if we extend the control strategy to several adjacent intersections with only one controller, an exact algorithm like branch and bound cannot be fast enough for all intersections in the sense of computational times. Thus, in this paper we propose a genetic algorithm to solve the sequencing problem efficiently. We also present a technique for combining certain vehicles into some basic groups taking into consideration the proposed properties. This will help us to reduce the search space.

The rest of this paper is arranged as follows. In Section 2, detailed assumptions of the studied problem are described, along with some basic notions of traffic control at intersections. Section 3 presents the technique for partitioning certain vehicles to fundamental mini-groups. A genetic algorithm is presented in Section 4 to solve the problem efficiently. A dynamic programming algorithm is also devised to evaluate the solution quality of the genetic algorithm. In Section 5, computational experiments show that the proposed genetic algorithm can decide on a vehicle passing sequence in less than 0.5 seconds for 100

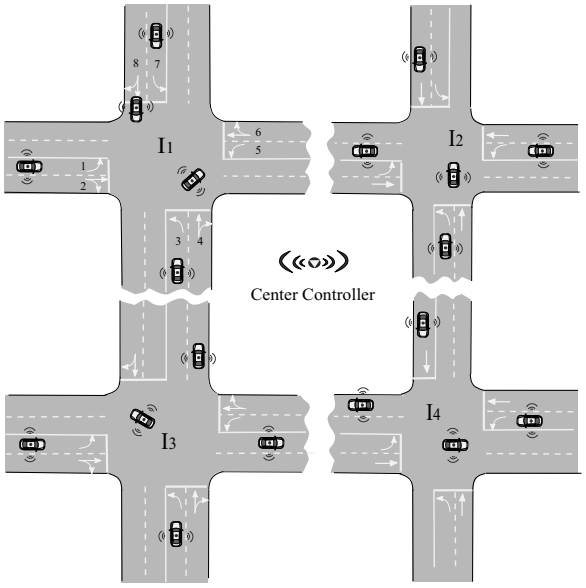


Fig. 1. Scheme of the vehicles sequencing problem at adjacent intersections.

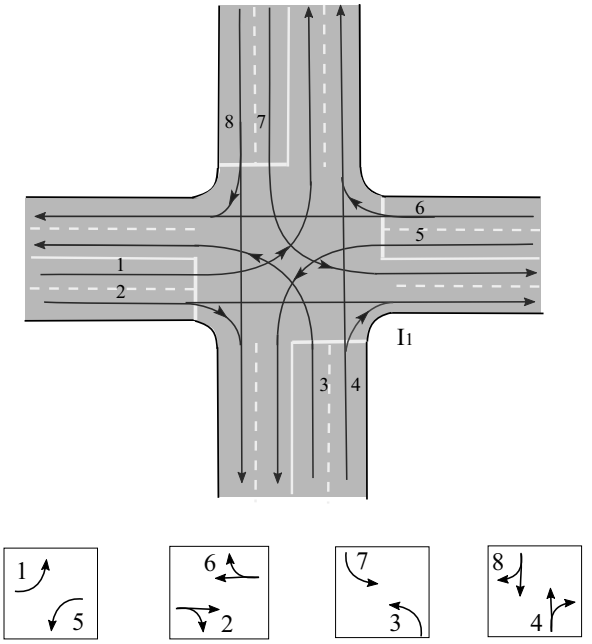


Fig. 2. Illustration of compatible and incompatible vehicle streams.

vehicles with small deviations from optimal solutions. Simulations with a continuous traffic flow are carried out at four adjacent intersections. The results indicate that the proposed algorithm can efficiently improve the traffic condition. Conclusions are drawn in Section 6.

2. Problem description and formulation

2.1. Problem description. The studied intersection network consists of several adjacent intersections that may have different layouts. An illustration is given in Fig. 1. There are four isolated intersections covered by a center controller. The information of each vehicle is gathered by the center controller in real time and vehicle passing sequences are decided on by the controller. Since the layout between two adjacent intersections is fixed (the number of lanes, the distance, etc.), we can easily decentralize the adjacent intersections control problem to several vehicle sequencing problems at isolated intersections. This will reduce the complexity of the decision making process.

In the sequel we focus on the vehicle sequencing problem at an isolated intersection. First, some basic notions should be introduced. Typically, an intersection consists of a number of approaches and a crossing area. Each approach may be used by several traffic streams. For example, for intersection I_1 in Fig. 2, the approach from west to east consists of two traffic streams (stream 1 and 2). Each stream has its own lane and an independent vehicle queue. We suppose that overtaking is not allowed, which indicates that vehicles on each lane need to pass an intersection in the first-in first-out way. The path used by a traffic stream to traverse the intersection is called the trajectory. A trajectory connects an approach on which vehicles enter the intersection to the intersection leg on which these vehicles leave the intersection. Vehicles belonging to some streams may have more than one trajectory (e.g., streams 2 and 8). The objective of traffic control at an intersection is to transform input traffic flows into output ones while preventing traffic conflicts and satisfying a specific criterion.

In order to prevent the conflicts of vehicle streams, frequently used traffic conventions provide the notion of compatible streams and incompatible streams. Obviously, when trajectories of two traffic streams do not cross, these streams can simultaneously get the right-of-way, and we call these two streams compatible streams. The lanes on which the two streams are moving are called compatible lanes. For example, streams 1 and 5 are compatible streams. On the other hand, when trajectories of two traffic streams do cross, the streams are in conflict (e.g., streams 1 and 7), and their simultaneous movement through the intersection should not be permitted. When several streams are compatible with each other, we call the set of these streams a Compatible Stream Group (CSG). In this example, we can partition the eight streams into four compatible stream groups: CSG 1 (streams 1 and 5), CSG 2 (streams 2 and 6), CSG 3 (streams 3 and 7) and CSG 4 (streams 4 and 8). One should note that the division of these groups is not constant when the traffic flow of a specific stream is much greater during peak time (e.g., morning peak time or evening peak time). However,

this division usually remains the same during a specific period.

Besides, for security reasons, there is always lost time (similar as the integral red time in traffic signal control) at the beginning of a vehicle passing sequence and when we switch the right-of-way between two vehicles of different $CSGs$ to avoid interference of incompatible streams. During lost time, no vehicle waiting for traverse the intersection is allowed to pass. The traffic control process is to decide a sequence of assigning the right-of-way to specific vehicles of each CSG to make these vehicles get through the intersection while satisfying a specific objective.

Suppose that at a start time $t_0 = 0$ there are n vehicles in the control range approaching an intersection from different approaches; the vital data of all vehicles in this range can be gathered by the controller immediately. At a basic level, the information from each vehicle contains the following parts:

1. Vehicle Identification (ID): used to identify an individual vehicle.
2. Stream number: which stream the vehicle belongs to, i.e., which lane it is moving on.
3. Precise vehicle arrival time: the precise time a vehicle arrives at the stop line from t_0 without interference.
4. Vehicle passing time: the time interval a vehicle needs to get through the intersection.

Remark that the vehicle passing time is actually the time interval in which a vehicle can accelerate from the waiting position until it reaches a safe distance with its follower on the same lane, i.e., in the same stream. We assume this time duration is a constant for each vehicle and only depends on the type of vehicle. For example, trucks are slower than small vehicles, they need more time to change the speed and therefore more time to accelerate until they can reach a safe distance for the following vehicle to move.

Typically, the most often used measure for evaluating the performance of a traffic control algorithm at an isolated intersection is the throughput. Under the environment of VII, maximizing the throughput can be viewed as minimizing the evacuation time of all approaching vehicles in each decision. Therefore, our objective is to minimize the Overall Evacuation Time (OET) of vehicles that are approaching an isolated intersection, i.e., the time that the last vehicle has passed the intersection.

2.2. Useful notation and formulation. Suppose there are overall n vehicles approaching the intersection and all vehicles are partitioned into m $CSGs$ based on compatible streams. In each CSG, vehicles are separated on different lanes. Some notation that will be used is as follows:

- CSG_i, the i-th CSG, where $1 \le i \le m$.
- n_i, the number of vehicles in CSG_i.
- s_i, the lost time when changing the right-of-way from other CSGs to vehicles in CSG_i.
- l_i, the number of lanes in CSG_i.
- $l_{(i,l)}$, the l-th lane in CSG_i, where $1 \le l \le l_i$.
- $n_{(i,l)}$, the number of vehicles on lane $l_{(i,l)}$.
- $v_{(i,l,j)}$, the j-th vehicle on lane $l_{(i,l)}$; j is indexed according to the arrival order of vehicles on lane $l_{(i,l)}$.
- $a_{(i,l,j)}$, the arrival time of $v_{(i,l,j)}$, i.e., the time $v_{(i,l,j)}$ needs to arrive at the stop lane (or the waiting queue) from time t_0.
- $s_{(i,l,j)}$, the time vehicle $v_{(i,l,j)}$ starts to pass the intersection.
- $p_{(i,l,j)}$, the passing time of $v_{(i,l,j)}$, i.e., the time $v_{(i,l,j)}$ needs to accelerate from the waiting position until it reaches a safe distance with its follower on the same lane.
- $C_{(i,l,j)}$, the completion time of $v_{(i,l,j)}$, i.e., the time that vehicle $v_{(i,l,j)}$ is already a safe distance from the vehicle following it on the same lane.

Thus, for n vehicles detected at time t_0, the objective can be described as

$$\min\{\max\{C_{(i,l,j)}\}\}, \quad 1 \le i \le m, \quad 1 \le l \le l_i, \quad 1 \le j \le n_{(i,l)} \tag{1}$$

subject to

$$s_{(i,l,j)} \ge a_{(i,l,j)}, \tag{2}$$

$$C_{(i,l,j)} = s_{(i,l,j)} + p_{(i,l,j)}, \tag{3}$$

$$s_{(i,l,j+1)} \ge C_{(i,l,j)}, \tag{4}$$

$$C_{(i,l,j)} - C_{(i',l',j')} \ge s_i + p_{(i,l,j)}, \quad \text{if } i \ne i' \text{ and } C_{(i,l,j)} \ge C_{(i',l',j')}. \tag{5}$$

The inequality (2) ensures that each vehicle starts to cross an intersection after its arrival. The equality (3) describes the vehicle completion time equals its real start time plus its passing time. The inequality (4) assures that a vehicle cannot start to traverse the intersection before its preceding vehicle on the same lane has finished the traversing process. The inequality (5) indicates that vehicles do not have the right-of-way to cross an intersection during lost time even if they have arrived.

3. Structural properties and fundamental mini-groups

Since we assume that each vehicle is an independent individual, there may be a great number of combinations of the vehicle passing sequence, especially when there are many vehicles approaching an intersection and the layout of the intersection is complex (e.g., many parallel lanes, many CSGs). In our earlier work, some useful structural properties were proposed based on the analysis of the optimal solutions of this problem (Yan *et al.*, 2009; 2010). In order to use these properties to simplify the search procedure, we briefly recall them and provide some useful notations.

3.1. Structural properties. In fact, from the description above, we can view the studied vehicle sequencing problem as a passing group sequencing problem. A vehicle Passing Group (PG) can be defined as a set of vehicles from the same *CSG* that pass intersection without the interruption of vehicles in other *CSG*s. Specifically, in a vehicle passing group sequence, a PG has the following properties:

1. Each *CSG* contains at least one PG.
2. Each vehicle should be in one and only one PG.
3. During the passing process of vehicles in one PG, there may be idle time waiting for some vehicles to arrive, but there is no lost time of any other *CSG* s.

The completion time of a vehicle passing group is defined as the maximum vehicle completion time in this passing group, i.e.,

$$C_{PG} = \max\{C_{(i,l,j)}\}, \text{ where } v_{(i,l,j)} \in PG.$$

Thus, the studied issue changes to a problem containing two main tasks: (i) partitioning vehicles into different vehicle passing groups and (ii) finding an efficient vehicle passing group sequence to optimize the overall vehicle evacuation time. This vehicle Passing Group Sequence (PGS) can be described as

$$PGS = (PG_1, PG_2, \ldots, PG_b), \quad b \geq m.$$

Since there may be several lanes in each *CSG* and vehicles on each lane should pass the intersection in an FIFO way, the vehicle arrival order on each lane should be taken into account. We can then have the following property easily.

Property 1. Consider there are two vehicles $v_{(i,l,j)}$ and $v_{(i,l,j')}$ on lane $l_{(i,l)}$ with $v_{(i,l,j')}$ arriving at the stop line after $v_{(i,l,j)}$, i.e., $j < j'$. Then we have

1. $a_{(i,l,j)} < a_{(i,l,j')}$, and $C_{(i,l,j)} < C_{(i,l,j')}$.
2. In a final passing group sequence, $v_{(i,l,j)}$ and $v_{(i,l,j')}$ have the following properties:
   (a) If $v_{(i,l,j)}$ and $v_{(i,l,j')}$ are contained in the same vehicle passing group PG, there is $\{v_{(i,l,k)}, j < k < j'\} \in PG$.
   (b) If $v_{(i,l,j)}$ and $v_{(i,l,j')}$ are contained in two different passing groups PG and PG', respectively, there must be a PG traversing the intersection before PG' in the passing group sequence.

Suppose there is a partial vehicle passing group sequence, in which some passing groups are already partitioned, but no decision has been taken yet on how to partition the remaining 'un-partitioned' vehicles. Some extra notation is given:

- C_{PG_r}, the completion time of the partial vehicle passing group sequence, i.e., the completion time of the last passing group PG_r in the partial sequence, where $1 \leq r < b$.
- $v_{(i,l,y)}$, the first 'un-partitioned' vehicle on lane $l_{(i,l)}$ of CSG_i after C_{PG_r}.
- $\mathcal{P}_{(i,l)}$, the sum of passing time of all 'un-partitioned' vehicles on lane $l_{(i,l)}$ of CSG_i, i.e.,

$$\mathcal{P}_{(i,l)} = \sum_{j=y}^{n_{(i,l)}} p_{(i,l,j)}. \tag{6}$$

Each time after a passing group is formed, we re-index the lane that has the maximum $\mathcal{P}_{(i,l)}$ of all lanes in CSG_i as lane 1, i.e., $l_{(i,1)}$, and $\mathcal{P}_{(i,1)}$ is the sum of the passing time of all 'un-partitioned' vehicles on this lane after C_{PG_r}.

Then we can have the following property.

Property 2. There exists an optimal vehicle passing group sequence, in which any vehicle passing group PG_x from CSG_i ($1 \leq x \leq b$) has at least one vehicle belonging to lane $l_{(i,1)}$.

Assume we are going to partition vehicles in CSG_i after a partial passing group sequence, and the completion time of this partial sequence is C_{PG_r}. We can have the following properties.

Property 3. Suppose that PG_x ($1 \leq x \leq b$) is a vehicle passing group containing vehicles in CSG_i. If

1. $C_{(i,l,y')} \leq C_{(i,1,y)}$ or
2. $0 < C_{(i,l,y')} - C_{(i,1,y)} \leq \mathcal{P}_{(i,l)} - (\mathcal{P}_{(i,1)} - p_{(i,1,y)})$,

there exists an optimal group sequence in which vehicles $v_{(i,1,y)}$ and $v_{(i,l,y')}$ are both contained in PG_x.

Property 4. There exists an optimal sequence, in which a vehicle passing group PG_x contains all 'un-partitioned' vehicles in CSG_i after time C_{PG_r} if $2s_i \geq C_{PG_x} - C_{PG_r} - \mathcal{P}_{(i,1)}$.

The proofs of these properties can be found in our earlier works, which focused on the exact algorithm for the vehicle sequencing problem at an isolated intersection (Yan *et al.*, 2009; 2012).

3.2. Fundamental mini-groups. Based on the properties above, we can combine some vehicles together and let them pass the intersection without the interference of vehicles in any other *CSG*s. Since the properties are obtained from optimal solutions, the combination will not jeopardize the optimality of the studied problem. Thus, by using the proposed properties, we can produce some basic groups by combining specific vehicles of the same *CSG* together. We define these basic groups as fundamental mini-groups (*FG*s). These *FG*s can be used as the minimum units in each *CSG* instead of the individual vehicles. A step by step procedure is described in Algorithm *FG*.

Algorithm 1. FG.

Step 1. Find the lane that has maximum $\mathcal{P}_{(i,l)}$ of all lanes from time $t_0 = 0$, re-index it as $l_{(i,1)}$.

Step 2. Partition the first 'un-partitioned' vehicle $v_{(i,1,y)}$ of the first lane $l_{(i,1)}$ into a new empty *FG*. If the other vehicles in CSG_i fit in Property 4, put these vehicles into the new *FG* and the partition of this *CSG* is finished; otherwise continue Step 3.

Step 3. If the first 'un-partitioned' vehicle on each lane $l_{(i,l)}$ $(1 < l \leq l_i)$ can be partitioned into the same *FG* according to Property 3, add this vehicle to the *FG* (i.e., the *FG* that contains $v_{(i,1,y)}$), continue Step 4.

Step 4. Recount $\mathcal{P}_{(i,l)}$ of all lanes with eliminating the vehicles already partitioned, repeat Steps 2–3 until all vehicles are partitioned into fundamental mini-groups.

An example is given in Fig. 3. There are three lanes in a *CSG* and six vehicles approaching an intersection at time t_0. The data of vehicles are given in Table 1.

Table 1. Vehicle arrival time and passing time in the FG partition example.

Vehicles	Arrival time	Passing time
$v_{(i,1,1)}$	1	3
$v_{(i,1,2)}$	5	3
$v_{(i,1,3)}$	23	4
$v_{(i,2,1)}$	2	4
$v_{(i,2,2)}$	15	2
$v_{(i,3,1)}$	7	3

On the lane $l_{(i,1)}$, $v_{(i,1,1)}$ is firstly partitioned into a new FG, then $v_{(i,2,1)}$ and $v_{(i,3,1)}$ should be examined as to whether they should be contained in same FG. The result is that $v_{(i,1,1)}$ will form an FG by itself. For the second FG, the lane that has maximum $\mathcal{P}_{(i,l)}$ should be recounted. Finally, all vehicles are partitioned into three *FG*s.

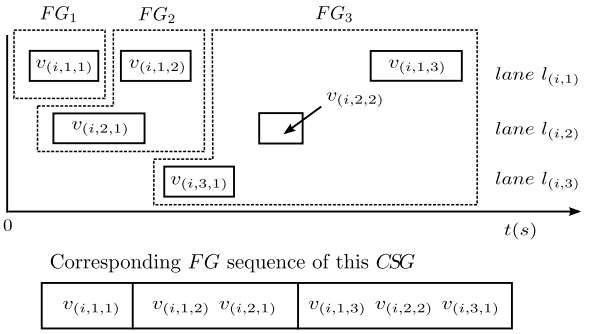


Fig. 3. Example of the fundamental mini-group partition procedure.

Clearly, this procedure can be done in polynomial time. Note that, after the *FG* partition, *FG*s in the same *CSG* also have a sequence because of the vehicle arrival order. Moreover, because of the fundamental mini-group partition procedure, the vehicle arrival order on different lanes of the same *CSG* are reduced to only one sequence, i.e., the sequence of *FG*s. We can observe that the sequencing of vehicles based on fundamental mini-groups is much easier than the original vehicle sequencing problem.

4. Proposed genetic algorithm and a dynamic programming algorithm

Genetic Algorithms (GAs), which mimic the evolutionary process in nature, have shown many successful applications to many fields, for example, to solve optimization problems (Akpinar and Bayhan, 2010; Xing *et al.*, 2008), control problems (Witkowska *et al.*, 2007; Belter and Skrzypczyński, 2010), operational problems (Aytug *et al.*, 2003; Hart *et al.*, 2005; Kashan *et al.*, 2008; Wang *et al.*, 1999) and transportation problems (Dridi and Kacem, 2004), etc.

Like creatures in nature evolve to adapt to the environment, solutions in the GA evolve to adapt to the target problem. Solutions in GAs are usually encoded into a compact form to facilitate the use of reproduction operators including crossover and mutation. The encoded solution is usually referred to as an individual, and a group of individuals is referred to as a population. Starting from an initial population, some individuals are selected as parents and then produce new individuals through

crossover and mutation. Among the original and new individuals, some survive and others die. The surviving individuals form a new population, and we call the transition from one population to another a generation. Individuals whose corresponding solutions have better objective values usually have higher probability to be selected as parents and survivors. It is expected that optimal or near-optimal solutions will be obtained by evolving the population after a number of generations.

In this paper, we develop a genetic algorithm to guide the autonomous vehicles through several adjacent intersections. Two encoding schemes are designed based on fundamental mini-groups. A heuristic named Smallest Extra Time (*SET*) is also proposed for decoding the chromosome and obtaining complete solutions that can be evaluated by a fitness function. Besides, crossover and mutation operators suitable for each encoding scheme are also devised. As already mentioned in Section 3.1, solving the target problem is usually achieved by two types of decisions—vehicle passing group forming and vehicle passing group sequencing. The proposed GA will search for good vehicle passing group formation and the sequence of these vehicle passing groups by different ways corresponding to different encoding schemes.

Initialization of the GA involves generating the initial population randomly with our chromosome encoding scheme and also including the solution obtained by *SET*. After the initialization phase, genetic operators are used to improve the solution quality. The proposed GA is outlined step by step as follows:

Step 1: Apply the algorithm *FG* to form all vehicles into different *FG*s.

Step 2: Generate a random initial population that contains N_{POP} individuals with a chromosome encoding scheme, and also include the individual obtained by *SET*.

Step 3: Decode randomly generated individuals and evaluate them by a fitness function. Record the lowest OET obtained by individuals in the initial population by OET^*.

Step 4: Determine the pairs that define which chromosomes will undergo the crossover operation.

Step 5: Apply the crossover operator to the selected pairs in order to obtain new pairs of offspring with a probability P_c.

Step 6: Perform mutation on the produced offspring with probability P_m.

Step 7: Decode and evaluate the two offspring. The best two individuals among the two parents and two offspring will replace the parents.

Step 8: If any of the stopping criteria are reached, stop. Otherwise, go to Step 4.

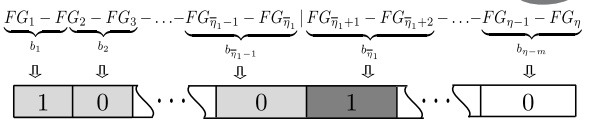


Fig. 4. Relation between binary bits and FGs for the GA.

4.1. Chromosome encoding. Based on fundamental mini-groups, we propose two encoding methods for genetic algorithms: a binary encoding scheme and a permutation encoding scheme. Comparisons between two schemes are presented in the next section.

4.1.1. Binary encoding scheme. Firstly, we adopt the primary binary encoding scheme for the proposed GA. Suppose that there are overall η fundamental mini-groups and the number of *FGs* in CSG_i is η_i. Certainly, we have

$$\eta = \sum_{i=1}^{m} \eta_i, \quad i = 1, 2, \ldots, m. \tag{7}$$

Define $\overline{\eta}_0 = 0$ and $\overline{\eta}_i$ $(i = 1, 2, \ldots, m)$ as

$$\overline{\eta}_i = \sum_{k=1}^{i} \eta_k. \tag{8}$$

Then we can label the *FGs* of each *CSG* as follows:

CSG_1: $FG_1, FG_2, \ldots, FG_{\overline{\eta}_1}$,
CSG_2: $FG_{\overline{\eta}_1+1}, FG_{\overline{\eta}_1+2}, \ldots, FG_{\overline{\eta}_2}$;
$\vdots$
CSG_m: $FG_{\overline{\eta}_{m-1}+1}, FG_{\overline{\eta}_{m-1}+2}, \ldots, FG_{\overline{\eta}}$;

In this binary encoding scheme for the GA, we need to decide whether to combine two adjacent FGs in the same CSG together to form a vehicle passing group. Since there are η_i fundamental mini-groups in CSG_i, $i = 1, 2, \ldots, m$, there will be $\eta_i - 1$ bits for CSG_i and the length of the binary string for chromosome is

$$L = \sum_{i=1}^{m} \{\eta_i - 1\} = \eta - m.$$

Then, if FG_{i-1} and FG_i belong to the same compatible stream group CSG_k, we define the binary bits between the two FGs as follows:

$$b_{i-k} = \begin{cases} 1 & \text{if } FG_{i-1} \text{ and } FG_i \text{ are combined,} \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

The relationship between b_{i-k} and FG_{i-1}, FG_i is shown in Fig. 4.

A numerical example is given as follows. Suppose there are four CSGs that contain 24 vehicles and the data of the vehicles are given in Table 2.

After the fundamental mini-group formation procedure, vehicles in four *CSG*s can form 5 FGs, 2 FGs, 3 FGs and 3 FGs, respectively. Then the length of the chromosome for the GA is $4 + 1 + 2 + 2 = 9$ bits, as shown in Fig. 5. For example, individual 1 in Fig. 5 represents the solution in which we let FG_1 and FG_2 in CSG_1 pass the intersection together without interruption, i.e., they form a vehicle passing group, then FG_3 and FG_4 pass together, and FG_5 will form a PG by itself at last. During the passing procedure of CSG_1, the right-of-way is changed two times to other *CSG*s, once after FG_2 and once after FG_4. Other *CSG*s can be deduced similarly.

After the chromosome encoding procedure, it is necessary to check whether a chromosome is feasible since in the chromosome we only consider the combination of fundamental mini-groups. For example, Fig. 6 shows an impossible individual.

With the formation of vehicle passing groups in Fig. 6, It is impossible to keep the 5 FGs in CSG_1 as 5 vehicle passing groups because the number of gaps among the 5 vehicle passing groups is smaller than the sum of vehicle passing groups in other *CSG*S. For any CSG_i, $i = 1, \ldots, m$, let NPG_i denote the number of vehicle passing groups contained in CSG_i according to a chromosome. We can have the following relation to make sure an individual is feasible. For all $CSG_i, i = 1, 2, \ldots, m$ there holds

$$NPG_i - 1 \leq \sum_{i'=1}^{m} NPG_{i'} \quad i' \neq i.$$

If this relation can be satisfied, the corresponding individual is ready to be used. If not, it is necessary to generate another feasible individual.

Table 2. Data of vehicles in four *CSG*s. Each vehicle has an integer arrival time a and an integer passing time p. Vehicles in each *CSG* are distributed on two or three lanes.

Vehicles	CSG_1		CSG_2		CSG_3		CSG_4	
	a	p	a	p	a	p	a	p
$v_{(i,1,1)}$	1	3	4	2	2	1	14	4
$v_{(i,1,2)}$	5	3	18	2	19	2	32	3
$v_{(i,1,3)}$	14	4	×	×	25	1	×	×
$v_{(i,1,4)}$	23	4	×	×	×	×	×	×
$v_{(i,1,5)}$	32	3	×	×	×	×	×	×
$v_{(i,2,1)}$	2	4	5	2	2	2	27	3
$v_{(i,2,2)}$	15	2	17	2	20	2	×	×
$v_{(i,2,3)}$	22	3	×	×	×	×	×	×
$v_{(i,3,1)}$	7	3	×	×	×	×	18	2
$v_{(i,3,2)}$	33	3	×	×	×	×	25	2

4.1.2. Permutation encoding scheme. To test the computational performance (convergence speed and accuracy), we give another chromosome encoding scheme for comparison. It adopts the permutation structure by encoding the combination and the sequence of FGs simultaneously in the chromosome. For example, as shown in Fig. 7, individual 1 represents the solution in which we let FG_1 pass the intersection first, then change the right-of-way to CSG_3 making FG_8 pass, ..., and finally FG_4, FG_5 in CSG_1. Different chromosomes represent different combinations of fundamental mini-groups and their sequences. We can observe that the binary encoding scheme only considers the combination of fundamental mini-groups in each *CSG*, and covers a sub-search space each time while the permutation encoding scheme indicates a specific solution. Besides, in order to get high-quality results, we need to carefully design the remaining components of the GA, which will be detailed in the following subsections.

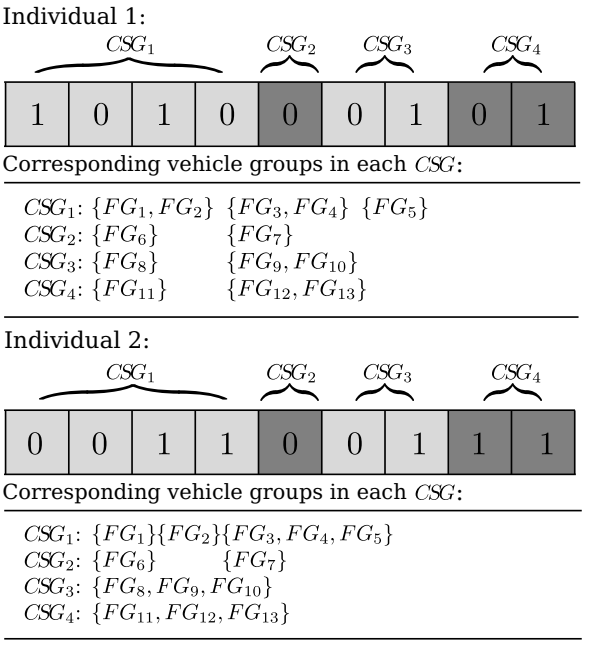


Fig. 5. Example of a binary encoding scheme of the GA.

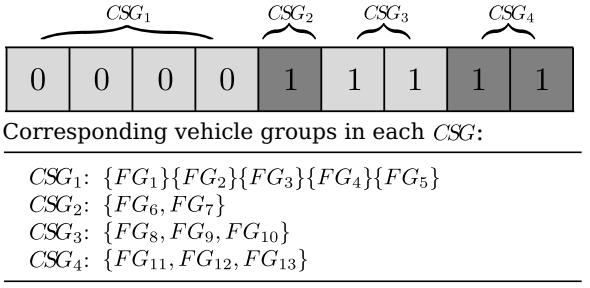


Fig. 6. Impossible individual generated by the proposed binary encoding scheme of the GA.

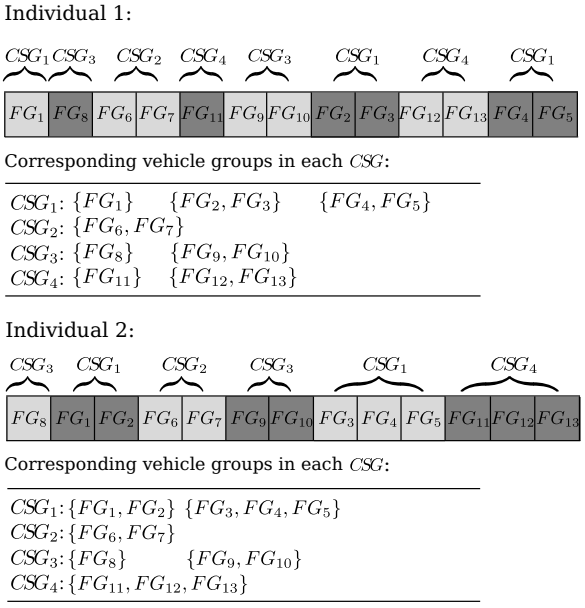


Fig. 7. Example of a permutation encoding scheme of the GA.

4.2. Chromosome decoding (*SET* heuristic). For a chromosome with a binary encoding scheme, the decoding scheme is responsible for constructing its corresponding schedule in order to calculate the concerned objective function. Since, in the encoding procedure, the vehicle passing groups of each *CSG* have already been formed, the remaining task is to decide on a passing group sequence. We propose the heuristic *SET* to achieve that end.

Suppose that PG_r (from CSG_j) is the last group of the partial vehicle passing group sequence already formed. In the group sequencing procedure, each time a new vehicle passing group PG_x (from CSG_i, $i \neq j$) is formed and appended to the partial sequence, the obtained new partial sequence will certainly have bigger completion time. Ideally, this increment will equal the sum of the passing time of all vehicles contained on the first lane $l_{(i,1)}$ of PG_x. However, the vehicle arrival time will probably introduce some extra time to the completion time of the partial sequence. Let q_i denote this extra time; it can be computed by following equation:

$$q_i = C_{PG_x} - (s_i + C_{PG_r}) - \mathcal{P}_{PG_x}, \qquad (10)$$

where $\mathcal{P}_{PG_x}$ is the sum of the passing time of vehicles on the first lane $l_{(i,1)}$ of PG_x. An example is given in Fig. 8.

Similarly, if we add some new vehicles (consider these vehicles are contained in a passing group called PG'_x) of the same *CSG*, i.e., CSG_j, into the last passing group PG_r of a partial sequence, i.e., $i = j$ of the case in Eqn. (10), the extra time caused by this PG'_x can be

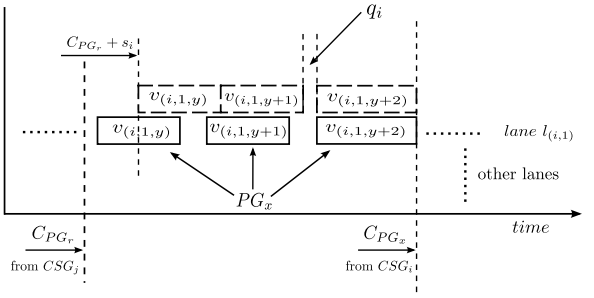


Fig. 8. Example of the extra time for SET: Case 1.

defined as

$$q_j = C_{PG'_x} - C_{PG_r} - \mathcal{P}_{PG'_x}. \qquad (11)$$

An example is given in Fig. 9.

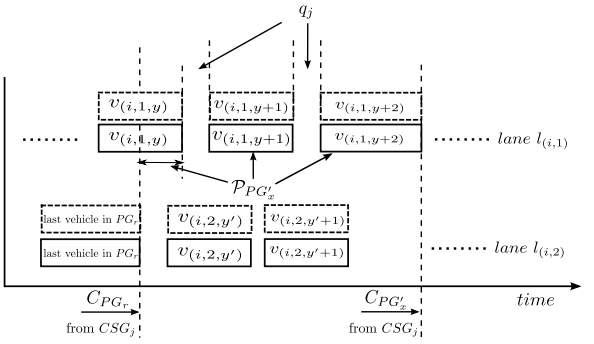


Fig. 9. Example of the extra time for SET: Case 2.

Note that, in this case, the first lane $l_{(i,1)}$ should be decided by the vehicle passing time that actually occurred after C_{PG_r} on and $\mathcal{P}_{PG'_x}$ indicates the sum of the vehicle passing time in PG'_x after C_{PG_r}. The pseudocode of this heuristic is also given.

Algorithm 2. Smallest Extra Time (*SET*).

while there are 'un-partitioned' vehicles **do**
  $C_{PG_r} \leftarrow 0$
  for CSG_i, i from 1 to m **do**
    Re-index the lane $l_{(i,1)}$ in each CSG_i;
    Form a passing group PG_x in each CSG_i by virtue of Properties 2–4; Count q_i by considering the PG_x as the last passing group;
  end for
  Append the PG_x with $\min\{q_i\}$ to the end of the partial sequence;
  $C_{PG_r} \leftarrow C_{PG_x}$;
end while

For the permutation encoding scheme, since it can already decide on a complete solution, the decoding procedure will be unnecessary.

4.3. Fitness evaluation and the selection strategy. In the proposed GA, the fitness of an individual is defined by the reciprocal of OET obtained by the decoding scheme. Mating selection is achieved through the roulette wheel. Crossover will be applied to produce two offspring and mutation will be applied probabilistically to the offspring. Among the parents and the offspring, two individuals with the highest fitness will replace the parents.

4.4. Crossover and mutation. Given two parents, the task of crossover is to generate the offspring through inheriting features (gene structures) from the parents. Since different encoding schemes have different gene structures and constraints, each encoding scheme has its own suitable crossover operators. For the binary encoding scheme, since a chromosome only contains binary bits, the two-point crossover is applicable, as shown in Fig. 10. Feasibility should also be checked after crossover. For the permutation encoding scheme, we apply linear

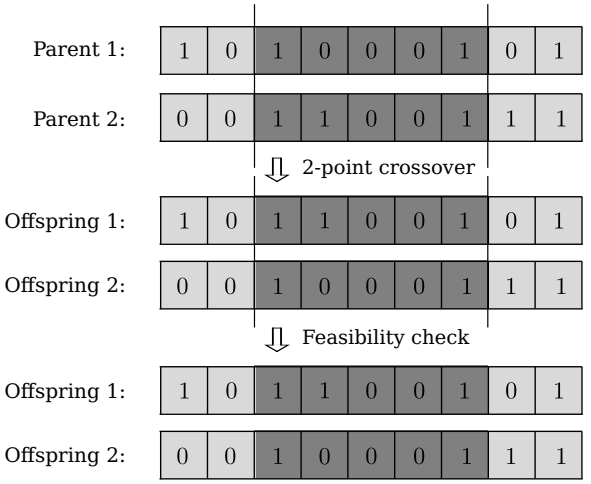


Fig. 10. Two-point crossover operation of a binary encoding scheme.

order crossover to the chosen parents. An example is given in Fig. 11. One should note that, after the crossover of chromosome with permutation encoding scheme, the sequence of *FG*s in each *CSG* may be changed, the FGs may not follow the FIFO order. Thus, another FIFO checking and adjustment procedure should be performed after the crossover. For example, in Fig. 11, the five FGs in CSG_1 should pass the intersection from FG_1 to FG_5. However, after the crossover, the FG_2 in the first offspring should pass the intersection after FG_3, FG_4 and FG_5. Thus, we shift FG_2 before the three FGs and next to the FG of the same *CSG*. If the FG we need to shift is the first FG in a *CSG*, we change its position to the beginning of the sequence.

For the binary encoding scheme, mutation is applied probabilistically. When a mutation is involved, we first

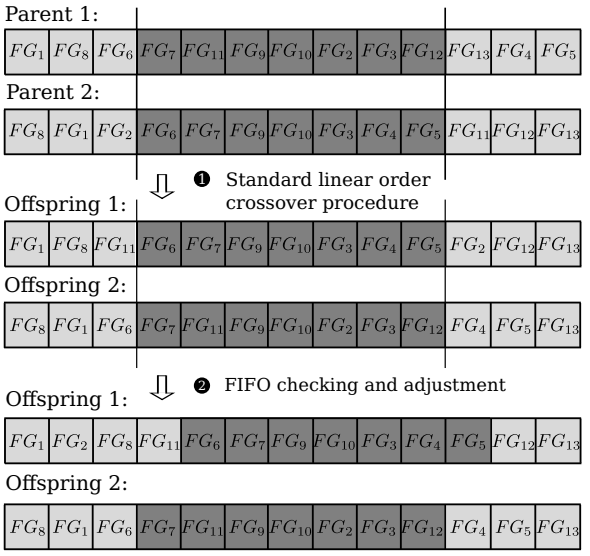


Fig. 11. Linear order crossover and checking of a permutation encoding scheme.

choose a random bit in the chromosome and then change the value of the bit form $0/1$ to $1/0$. Feasibility of the new individual should also be checked. In a permutation encoding scheme, we randomly choose two bits in the chromosome and change their position to generate a new individual. An illustration of mutation is shown in Fig. 12.

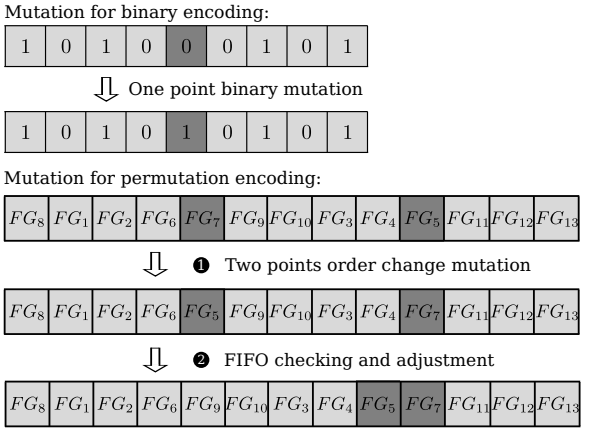


Fig. 12. Mutation operation of the GA.

One should note that each time after a crossover or a mutation operation the new individuals should be checked for their possibility. Only feasible offspring can be used for next operations. If any individual is not feasible, a crossover or a mutation operator should be applied again until feasible offspring are generated.

4.5. Generation of the initial population and the stopping criterion. The initial population of the proposed

GA is generated randomly according to the chromosome encoding schemes. Each individual under the framework of a binary encoding scheme should be checked after the generation, crossover and mutation. The initial population should also include the solution obtained by *SET*; this individual is encoded and placed as the first individual.

The algorithm will stop if the best solution is not updated after N_{NOBETTER} continuous generations or when a predefined maximum number of generations (N_{MAXGEN}) is reached.

4.6. Dynamic programming approach. In order to analyze errors of the proposed genetic algorithm, we also present a Dynamic Programming (DP) approach to find an optimal solution. This algorithm also uses the mini-groups as the minimum units of each *CSG*.

For the studied problem, suppose that the fundamental mini-group partition procedure has already been finished. Vehicles in each *CSG* are pre-partitioned into several FGs. Each *CSG* can be viewed as a sequence of FGs:

$$CSG_i = \{FG_1, FG_2, \ldots, FG_d\},$$

where $1 \le d \le n_i$. Vehicles in FG_i with $1 \le i \le d$ are also enumerated according to their arrival order. For example, let $v_{(i,j-1)}$ and $v_{(i,j)}$ be respectively the $(j-1)$-th and j-th vehicle in CSG_i (lanes of vehicles are not considered here); then $a_{(i,j-1)}$ will not be greater than $a_{(i,j)}$.

For a nonnegative integer x, where $x = |FG_1| + |FG_2| + \cdots + |FG_i|$, $1 \le i \le d$, define

$$CSG_i^{(x)} = \left\{v_{(i,j)}, 1 \le j \le x\right\}, \quad 1 \le i \le m.$$

For m integers $x_1, x_2, \ldots, x_m$, where $0 \le x_i \le n_i$ and $1 \le i \le m$, let

$$f_i(x_1, x_2, \ldots, x_i, \ldots, x_m)$$

denote the minimum evacuation time of the studied problem restricted to the m sub *CSG*s,

$$CSG_1^{(x_1)}, CSG_2^{(x_2)}, \ldots, CSG_m^{(x_m)},$$

with the last FG_r in the partial sequence being from CSG_i, $1 \le i \le m$. Then the finial minimum evacuation time of the studied problem is

$$f^* = \min_{1 \le i \le m} \{f_i(n_1, n_2, \ldots, n_m)\}.$$

Define a_{FG} as the arrival time of FG, i.e., $a_{FG} = \min_{j \in FG}\{a_j\}$, p_{FG} is the time for making all vehicles in FG pass the intersection. The recursion of our dynamic programming can be expressed as

$$f_i(x_1, x_2, \ldots, x_m) = \min \begin{cases} \min\limits_{1 \le i' \le m} \{\max\{f_{i'} + s_i, a_{FG_r}\} + p_{FG_r}\}, \\ \qquad i' \ne i; \\ \max\limits_{1 \le l \le l_i} \{C_{l_{(i,l)}}\}. \end{cases}$$

In the formula,

$$f_{i'} = f_{i'}(x_1, \ldots, x_i - |FG_r|, \ldots, x_m), \quad i' \ne i,$$

and a_{FG_r} is the arrival time of the last mini-group FG_r already considered, p_{FG_r} is the time used for making all vehicles in the FG pass after the time $\max\{f_{i'}(x_1, \ldots, x_i - |FG_r|, \ldots, x_m) + s_i, a_{FG_r}\}$, where $1 \le i' \le m, i' \ne i$; $C_{l_{(i,l)}}$ is the completion time of the last vehicle considered on lane $l_{(i,l)}$ after we add all the vehicles in FG_r ($i' = i$ in this case) into the partial sequence which corresponds to $f_i(x_1, \ldots, x_i - |FG_r|, \ldots, x_m)$. The initial condition is given by

$$f_i(0, \ldots, 0, |FG_1|, \ldots, 0) = s_i + p_{FG_1},$$

where FG_1 is the first FG of CSG_i and p_{FG_1} is the time used for making all vehicles pass in FG_1 after s_i.

Define NFG_i as the number of fundamental mini-groups in CSG_i, $1 \le i \le m$, and

$$NFG = \sum_{i=1}^{m} NMG_i.$$

The dynamic programming function has at most

$$m(NFG_1 + 1)(NFG_2 + 1) \cdots (NFG_m + 1) \le m\Big(\frac{NFG}{m} + 1\Big)^m$$

states. The complexity of this dynamic programming is $O(m(NFG/m + 1)^m)$.

Clearly, by using the fundamental mini-group partition procedure, the complexity of the dynamic programming algorithm can be dramatically reduced compared to the use of individual vehicles, especially when there are many lanes.

As we have mentioned before, the division of the compatible stream groups can be different in real-world traffic control, but it can be seen as constant for a specific duration of each day. This means, for real-world applications, that the problem usually needs less computational resources for finding an optimal vehicle passing sequence each time. For example, at the intersection I_1 in Fig. 2, there are four compatible groups waiting to pass the intersection. With the proposed dynamic programming algorithm, we can get an optimal sequence in $O(NFG^4)$ time for evacuating all detected vehicles, which can surely meet the needs of a real-time traffic control system.

4.7. Algorithm applications. Since at real-world intersections vehicles keep entering the control range of the center controller, the algorithm should be executed separately to each isolated intersection whenever there are new vehicles detected. However, if a vehicle passing group has the right-of-way to pass an intersection, the recalculation process should be postponed until all vehicles in that PG have passed through the intersection.

In the shared space between two neighbor intersections, for example, considering the lanes (from west to east) between intersection I_1 and intersection I_2 in Fig. 1, the output traffic streams of intersection I_1 is also the input traffic stream of intersection I_2. The vehicles that have already traversed Intersection I_1 (from west to east) can be seen as the new incoming vehicles for intersection I_2, and vice versa.

Moreover, since the layout between two adjacent intersections is fixed in both the number of lanes and the distance, we suppose that vehicles will use the same time to traverse from an upstream intersection to a downstream intersection. Therefore, when some vehicles are grouped together to pass the upstream intersection according to their arrival time and passing time, they can also be scheduled in the same passing group if their intentions are still the same (see Fig. 13). The computational time will also be reduced this way.

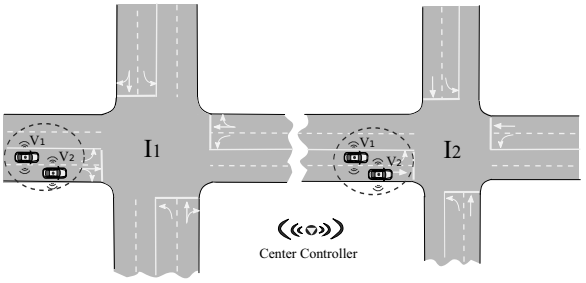


Fig. 13. Same vehicle passing group from an upstream to a downstream intersection.

5. Performance evaluation

The algorithm performance of the GA is evaluated for both computational efficiency and accuracy. In a real-world traffic flow, new vehicles keep entering the control range of the controller, algorithms should be applied to each isolated intersection whenever a new vehicle enters the control range. For this reason, the computation time of the control algorithm should be short enough to satisfy the need of the real-time system. Since there are no existing benchmarks for this autonomous vehicle sequencing problem, we compare the computational performance of the proposed GA with different encoding schemes, the heuristic *SET*, the proposed DP and the branch and bound algorithm from our earlier work at an isolated intersection. We also evaluate the errors (average deviations) of the genetic algorithm with the optimal solution obtained by the proposed exact algorithms.

5.1. Computational results. The computation experiments are implemented at an isolated four-approach intersection that has four *CSG*s. Parameters of the experiments are summarized in Table 3.

Table 3. Parameters of computational instances.

Problem parameter	Values used
Number of *CSG*	4
Number of lanes in *CSG*	2, 3, 4
Number of vehicles	10, 25, 50, 75, 100
Vehicle passing time	Integers from 2 to 8
Lost time of *CSG*	Integers from 3 to 8
Population size	10, 25, 100
$N_{\mathrm{NOBETTER}}/N_{\mathrm{MAXGEN}}$	10/30

The number of vehicles approaching this intersection is varied from 10 to 100. Vehicles are equally distributed among the approaching lanes. Without loss of generality, we assume that there are 2 lanes (small intersection), 3 lanes (medium intersection) or 4 lanes (big intersection) in each of the four *CSG*s separately. For example, when number of lanes in each *CSG* is 2, there are overall $4 \times 2 = 8$ lanes leading to this intersection. Vehicles (50 vehicles, for instance) are equally distributed among the 8 lanes, i.e., around 6 vehicles on each lane: $8 \times 6 = 48 \approx 50$. This way, the test average number of vehicles on each lane is varied from 1 to 8. Besides, according to the usually used data of isolated intersections, we set the lost time of each *CSG* as randomly generated integers varied from 3 to 8 seconds, and the passing time of each vehicle as randomly generated integers varied from 2 to 8 seconds. Adjacent vehicles on the same lane should follow the "2 seconds rules" (at least 2 seconds between the arrival time of two adjacent vehicles). Population size (N_{POP}) is varied from 10 to 100. Different probabilities of crossover (P_c) and mutation (P_m) are also tested. Computational results are presented in Tables 4–6. In the tables, the average CPU time (AVE) in seconds is given as well as its error (ERR) in percentage comparing with the optimal value. All approaches are coded in C++ and run on a desktop computer with a Linux system (kernel 2.6.32).

The results illustrate the following facts:

- The proposed GA can handle 100 vehicles in its control range in a very short time.
- For the same number of vehicles, the more lanes one approach has, the less time the GA uses. This is because more vehicles can be formed as one

Table 4. Computational performance of the proposed GA with $N_{POP} = 10$, $P_c = 0.9$ and $P_m = 0.10$.

Lanes	Vehicles	Permutation encoding		Binary encoding		Branch and Bound		Dynamic Programming		Heuristic *SET*
		AVE	*ERR*	*AVE*	*ERR*	*AVE*	*MAX*	*AVE*	*MAX*	*ERR*
2	10	0.009	4.3%	0.023	4.3%	0.021	0.060	0.033	0.057	4.2%
	25	0.024	10.5%	0.048	8.2%	0.878	2.500	0.224	2.148	10.5%
	50	0.047	14.9%	0.109	11.3%	2.896	10.08	2.746	7.546	16.3%
	75	0.081	15.6%	0.165	13.8%	4.961	11.24	4.512	10.370	26.6%
	100	**0.119**	**16.9**%	**0.244**	**14.2**%	7.845	**14.520**	6.246	**12.140**	36.5%
3	10	0.006	0.0%	0.014	0.0%	0.061	0.198	0.113	0.475	4.1%
	25	0.018	5.1%	0.039	6.4%	0.241	1.104	0.388	0.966	6.4%
	50	0.027	9.6%	0.082	7.8%	0.260	0.963	0.912	2.105	10.4%
	75	0.033	10.7%	0.125	8.2%	0.398	1.612	1.390	3.503	20.1%
	100	**0.060**	**13.6**%	**0.231**	**10.6**%	0.580	**3.186**	2.221	**6.829**	27.8%
4	10	0.008	0.0%	0.014	0.0%	0.010	0.018	0.028	0.054	0.0%
	25	0.014	5.5%	0.033	0.0%	0.025	0.221	0.077	0.072	6.8%
	50	0.020	9.2%	0.076	6.1%	0.036	0.436	0.188	0.167	8.2%
	75	0.023	9.9%	0.128	7.8%	0.197	0.640	0.785	1.490	16.3%
	100	**0.057**	**11.2**%	**0.194**	**8.7**%	0.368	**1.845**	1.593	**3.967**	23.6%

Table 5. Computational performance of the proposed GA with $N_{POP} = 25$, $P_c = 0.7$ and $P_m = 0.2$.

Lanes	Vehicles	Permutation encoding		Binary encoding		Branch and Bound		Dynamic Programming		Heuristic *SET*
		AVE	*ERR*	*AVE*	*ERR*	*AVE*	*MAX*	*AVE*	*MAX*	*ERR*
2	10	0.039	0.0%	0.066	0.0%	0.021	0.060	0.033	0.057	4.2%
	25	0.088	7.4%	0.151	0.0%	0.878	2.500	0.224	2.148	10.5%
	50	0.174	9.2%	0.247	6.4%	2.896	10.08	2.746	7.546	16.3%
	75	0.298	11.9%	0.380	7.8%	4.961	11.24	4.512	10.370	26.6%
	100	**0.475**	**13.5**%	**0.594**	**9.3**%	7.845	**14.520**	6.246	**12.140**	31.1%
3	10	0.022	0.0%	0.049	0.0%	0.061	0.198	0.113	0.475	4.1%
	25	0.053	3.8%	0.114	0.0%	0.241	1.104	0.388	0.966	6.4%
	50	0.115	8.7%	0.213	5.2%	0.260	0.963	0.912	2.105	10.4%
	75	0.190	10.7%	0.309	7.5%	0.398	1.612	1.390	3.503	20.1%
	100	**0.402**	**13.1**%	**0.442**	**9.6**%	0.580	**3.186**	2.221	**6.829**	27.8%
4	10	0.017	0.0%	0.041	0.0%	0.010	0.018	0.028	0.054	0.0%
	25	0.044	0.0%	0.097	0.0%	0.025	0.221	0.077	0.072	6.8%
	50	0.106	6.1%	0.198	0.0%	0.036	0.436	0.188	0.167	8.2%
	75	0.164	9.9%	0.288	7.1%	0.197	0.640	0.785	1.490	16.3%
	100	**0.331**	**12.4**%	**0.465**	**8.7**%	0.368	**1.845**	1.593	**3.967**	23.6%

fundamental mini-group before the search procedure, and the computational time reduces in consequence.

- For most cases, the binary encoding scheme outperforms the permutation encoding scheme in the sense of accuracy. It achieves a better solution than the permutation encoding scheme. However, it needs more computational time because of the decoding procedure.
- With the population size increase, the computational time of the genetic algorithm increases quickly, but the the errors do not decrease significantly as the calculation time does. Since a real traffic control system needs the decision process be very fast, we can choose the population size with a different traffic situation.

5.2. Simulations with a continuous traffic flow. To test the performance of the proposed control strategy for a continuous traffic flow, simulations are implemented at a 4-intersection network. The overall evacuation time of these adjacent intersections is examined by using the

Table 6. Computational performance of the proposed GA with $N_{POP} = 100$, $P_c = 0.9$ and $P_m = 0.1$.

Lanes	Vehicles	Permutation encoding		Binary encoding		Branch and Bound		Dynamic Programming		Heuristic *SET*
		AVE	*ERR*	*AVE*	*ERR*	*AVE*	*MAX*	*AVE*	*MAX*	*ERR*
2	10	0.075	0.0%	0.114	0.0%	0.021	0.060	0.033	0.057	4.2%
	25	0.149	5.3%	0.190	0.0%	0.878	2.500	0.224	2.148	10.5%
	50	0.363	7.1%	0.531	5.6%	2.896	10.08	2.746	7.546	16.3%
	75	0.612	8.7%	1.152	6.9%	4.961	11.24	4.512	10.370	26.6%
	100	**1.243**	**10.0**%	**2.314**	**7.6**%	7.845	**14.520**	6.246	**12.140**	31.1%
3	10	0.060	0.0%	0.087	0.0%	0.061	0.198	0.113	0.475	4.1%
	25	0.122	0.0%	0.153	0.0%	0.241	1.104	0.388	0.966	6.4%
	50	0.298	7.0%	0.441	3.5%	0.260	0.963	0.912	2.105	10.4%
	75	0.467	8.8%	1.032	6.9%	0.398	1.612	1.390	3.503	20.1%
	100	**0.904**	**10.6**%	**1.876**	**7.6**%	0.580	**3.186**	2.221	**6.829**	27.8%
4	10	0.048	0.0%	0.059	0.0%	0.010	0.018	0.028	0.054	0.0%
	25	0.091	0.0%	0.104	0.0%	0.025	0.221	0.077	0.072	6.8%
	50	0.188	5.1%	0.312	0.0%	0.036	0.436	0.188	0.167	8.2%
	75	0.379	7.1%	0.753	4.3%	0.197	0.640	0.785	1.490	16.3%
	100	**0.642**	**10.6**%	**1.142**	**6.2**%	0.368	**1.845**	1.593	**3.967**	23.6%

proposed genetic algorithms. We also test the influence of using the proposed GA on two other frequently used measures. One is the average vehicle waiting time, which indicates the average waiting time of vehicles before traversing an intersection; another is the average queue length, which means the average number of vehicles waiting to cross an intersection. These two measures are evaluated separately at four intersections and we take the average value from four intersections as the value of these intersections.

Suppose the distance between any two adjacent intersections is the same and all vehicles need 7 s to travel from an upstream intersection to a downstream intersection. Each intersection has four approaches and each approach contains two lanes for incoming vehicles. The cover range of the controller for each approach is 50 meters. Vehicles approaching the intersection are partitioned into four *CSG*s. According to statistics, we assume that the maximum traffic load for each of the four approaches is 1800 vehicles/h (about one vehicle every 2 seconds). Other configurations like vehicle passing time and lost time are the same as in the computational experiments. Each data point is obtained by taking the average over three separate simulations. Each simulation runs 60 minutes of the traffic flow.

The proposed genetic algorithms were applied with the population size 100. The performances of the GA, the dynamic programming algorithm, the heuristic and our earlier branch and bound algorithm are compared together with the following two control strategies:

1. The traditional fix-cycle time, in which the famous Webster formula (Webster, 1958) is applied to calculate the cycle length and the green time according to the estimated traffic volume.

2. Adaptive control system: a traditional traffic control method which is proven efficient in the current traffic system. Here, the method presented by Fang and Elefteriadou (2006) is used for comparison.

Simulation results of evacuation time are presented in Fig. 14. The effect for average queue size and average vehicle waiting time after applying different algorithms is presented in Figs. 15 and 16, respectively.

In the results, we can note that the two exact algorithms (branch and bound and dynamic programming) have different values for some traffic flow rate. The reason is that, for several adjacent intersections, the exact algorithms may be not fast enough to get an optimal solution for all intersections at some traffic load. The passing sequence may not be uploaded in consequence.

However, the proposed genetic algorithms performed well for several adjacent intersections. We can observe that the proposed GA can reduce the overall evacuation time for nearly 60 s (from 3691 s to 3632 s) with a high traffic flow rate (0.5 vehicle/s). One should note that the evacuation time is bounded by the simulation time, i.e., 3600 s. This means the improvement obtained by using the new control strategy and proposed algorithms is $(3691 - 3632)/(3691 - 3600) \approx 65\%$.

For the average queue size, we can find that the proposed GA reduces the average queue length for more than 75% for certain traffic flow rates. Meanwhile, the average vehicle waiting time is also decreased

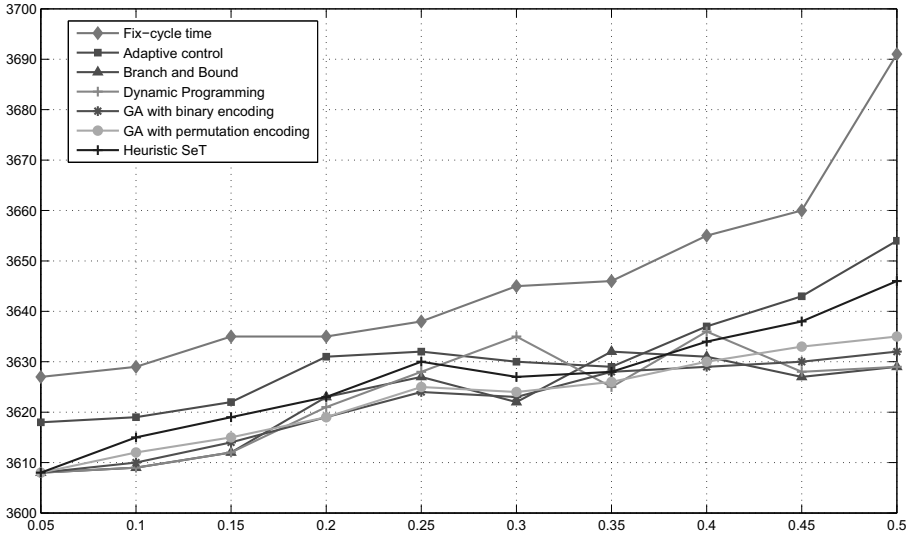


Fig. 14. Overall evacuation time of simulated adjacent intersections.

significantly from 35.7 s to 8.2 s, which means vehicles will spend 77% less time before traversing this area.

We can also note that the proposed GA can response quickly for new detected vehicles with different traffic flow rates. Passing sequences can be uploaded in time. Moreover, the binary encoding scheme performs a little better than the permutation encoding scheme. The heuristic SET also performs better than traditional adaptive control with traffic lights in the sense of different measures.

6. Conclusions

In this paper, we studied the autonomous vehicle sequencing problem at adjacent intersections under the framework of AIM. The VII technology was used to obtain each vehicle's information. The several intersections considered were decentralized to several isolated intersections and at each isolated intersection; vehicles were treated as discrete individuals. Our objective was to minimize the overall evacuation time at each decision.

In order to obtain an optimal or a near-optimal vehicle passing sequence, a fundamental mini-group algorithm based on the properties our earlier work was devised to combine certain vehicles of the same compatible stream group together as the minimum unit. Then we designed a genetic algorithm which adopts the binary encoding scheme or permutation encoding scheme to decide on the vehicle passing sequence each time. A heuristic *SET* was also proposed to decode the chromosome obtained by the binary encoding scheme. It can also be used independently to find a vehicle passing sequence. A dynamic programming algorithm was also presented for finding an optimal solution.

Two encoding schemes of the GA, the heuristic *SET*, dynamic programming and the branch and bound algorithm in our earlier work, were compared in computational experiments. The results showed that the proposed genetic algorithm can decide on a vehicle passing sequence in less than 0.5 seconds for 100 vehicles with small deviations from optimal solutions. Simulations with a continuous traffic flow at a four-intersection network were also implemented. The results indicated that the proposed algorithm can efficiently improve the traffic condition.

In future work, differences between normal vehicles and special used vehicles such as ambulances or police cars will be considered. Special use vehicles should have the privileges to pass through intersection. Each vehicle will have a weight of priority and we need to make vehicles with high weights pass the intersection as quickly as possible while satisfying certain objectives. More constraints and rules will be considered in the next stage of our work to make the control strategy more suitable for real applications.

Acknowledgment

The authors are grateful to the anonymous referees for their constructive remarks and suggestions.

References

Akpinar, S. and Bayhan, G.M. (2010). A hybrid genetic algorithm for mixed model assembly line balancing problem with parallel workstations and zoning constraints, *Engineering Applications of Artificial Intelligence* **24**(3): 449–457.

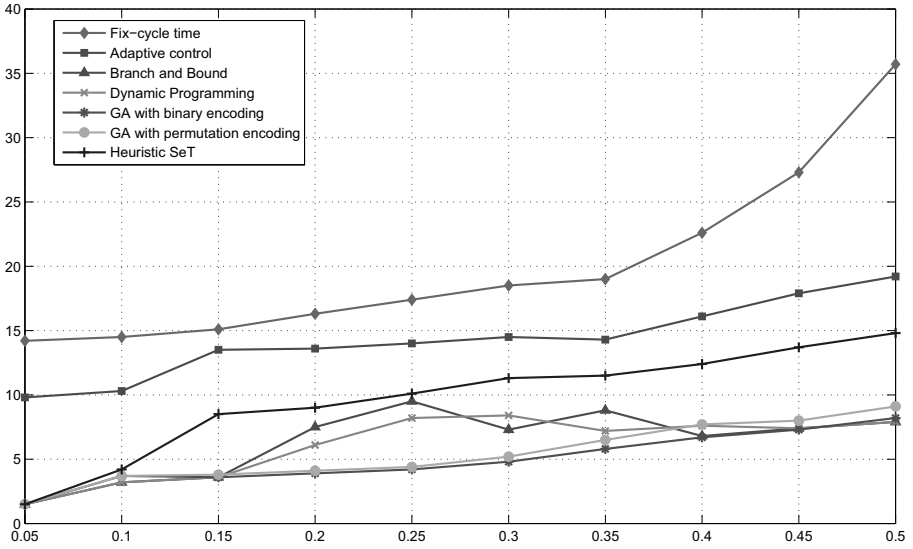


Fig. 15. Average queue length of simulated adjacent intersections.

Aotani, T., Yamaoka, S. and Tajima, T. (2002). Research & development of driving safety support systems, *Proceedings of the 41st SICE Annual Conference, Osaka, Japan*, Vol. 3, pp. 1792–1797.

Aytug, H., Khouja, M. and Vergara, F.E. (2003). Use of genetic algorithms to solve production and operations management problems: A review, *International Journal of Production Research* **41**(17): 3995–4009.

Belter, D. and Skrzypczyński, P. (2010). A biologically inspired approach to feasible gait learning for a hexapod robot, *International Journal of Applied Mathematics and Computer Science* **20**(1): 69–84, DOI: 10.2478/v10006-010-0005-7.

Bertolazzi, E., Biral, F., Da Lio, M., Saroldi, A. and Tango, F. (2010). Supporting drivers in keeping safe speed and safe distance: The SASPENCE subproject within the European Framework Programme 6 Integrating Project Prevent, *IEEE Transactions on Intelligent Transportation Systems* **11**(3): 525–538.

Chisalita, L. and Shahmehri, N. (2002). A peer-to-peer approach to vehicular communication for the support of traffic safety applications, *Proceedings of the IEEE 5th International Conference on Intelligent Transportation Systems, Singapore*, pp. 336–341.

Dresner, K. and Stone, P. (2004). Multiagent traffic management: A reservation-based intersection control mechanism, *Proceedings of Autonomous Agents and Multiagent Systems AAMAS'04, New York, NY, USA*, pp. 530–537.

Dresner, K. and Stone, P. (2006). Traffic intersections of the future, *Proceedings of the 21st National Conference on Artificial Intelligence, Boston, MA, USA*, pp. 1593–1596.

Dridi, M. and Kacem, I. (2004). A hybrid approach for scheduling transportation networks, *International Journal of Applied Mathematics and Computer Science* **14**(3): 397–409.

Fang, F. and Elefteriadou, L. (2006). Development of an optimization methodology for adaptive traffic signal control at diamond interchanges, *Journal of Transportation Engineering* **132**(8): 629–637.

Gradinescu, V., Gorgorin, C., Diaconescu, R., Cristea, V. and Iftode, L. (2007). Adaptive traffic lights using car-to-car communication, *Proceedings of the IEEE 65th Vehicular Technology Conference, VTC2007-Spring, Dublin, Ireland*, pp. 21–25.

Hall, R. W. and Papageorgiou, M. (1999). *Handbook of Transportation Science*, Springer, New York, NY/Boston, MA/Dordrecht/London/Moscow.

Hart, E., Ross, P. and Corne, D. (2005). Evolutionary scheduling: A review, *Genetic Programming and Evolvable Machines* **6**(2): 191–220.

Huang, Q. and Miller, R. (2003). The design of reliable protocols for wireless traffic signal systems, *Technical report*, Department of Computer Science and Engineering, Washington University, Saint Louis, MO.

Hunt, P. (1982). The scoot on-line traffic signal optimization technique, *Traffic Engineering & Control* **23**(4): 190–192.

Kashan, A., Karimi, B. and Jenabi, M. (2008). A hybrid genetic heuristic for scheduling parallel batch processing machines with arbitrary job sizes, *Computers & Operations Research* **35**(4): 1084–1098.

Kato, S., Tsugawa, S., Tokuda, K., Matsui, T. and Fujii, H. (2002). Vehicle control algorithms for cooperative driving with automated vehicles and intervehicle communications, *IEEE Transactions of Intelligent Transportation Systems* **3**(3): 155–161.

Lachner, R. (1997). Collision avoidance as a differential game: real-time approximation of optimal strategies using higher derivatives of the value function, *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Orlando, FL, USA*, Vol. 3, pp. 2308–2313.

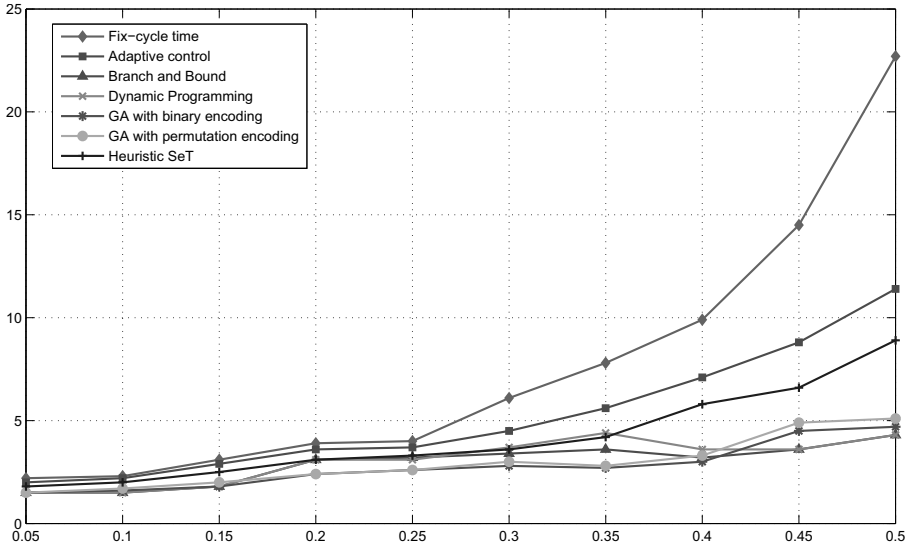


Fig. 16. Average waiting time of simulated adjacent intersections.

Li, L. and Wang, F. (2006). Cooperative driving at blind crossings using intervehicle communication, *IEEE Transactions on Vehicular Technology* **55**(6): 1712–1724.

Nadeem, T., Dashtinezhad, S. and Liao, C. (2004). TrafficView: A scalable traffic monitoring system, *Proceedings of the IEEE International Conference on Mobile Data Management, Berkeley, CA, USA*, pp. 13–26.

Robertson, D. (1969). TRANSYT: A traffic network study tool, *Technical Report TRRL-LR-253*, Transport and Road Research Laboratory, Crowthorne.

Shladover, S., Desoer, C., Hedrick, J., Tomizuka, M., Walrand, J., Zhang, W.-B., McMahon, D., Peng, H., Sheikholeslam, S. and McKeown, N. (1991). Automated vehicle control developments in the path program, *IEEE Transactions on Vehicular Technology* **40**(1): 114–130.

Shladover, S.E. (2007). Path at 20—History and major milestones, *IEEE Transactions on Intelligent Transportation Systems* **8**(4): 1_22–1_29.

Wang, D., Gen, M. and Cheng, R. (1999). Scheduling grouped jobs on single machine with genetic algorithm, *Computers & Industrial Engineering* **36**(2): 309–324.

Webster, F. (1958). Road research technical paper, *Technical report*, Road Research Laboratory, London.

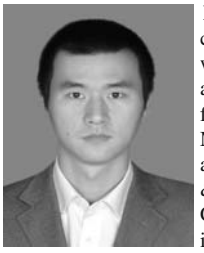
Witkowska, A., Tomera, M. and Śmierzchalski, R. (2007). A backstepping approach to ship course control, *International Journal of Applied Mathematics and Computer Science* **17**(1): 73–85, DOI: 10.2478/v10006-007-0007-2.

Wu, J., Abbas-Turki, A. and El Moudni, A. (2009). Discrete methods for urban intersection traffic controlling, *Proceedings of the IEEE 69th Vehicular Technology Conference, Barcelona, Spain*, pp. 1–5.

Xing, L., Chen, Y., Yang, K., Hou, F., Shen, X. and Cai, H.-P. (2008). A hybrid approach combining an improved genetic algorithm and optimization strategies for the asymmetric traveling salesman problem, *Engineering Applications of Artificial Intelligence* **21**(8): 1370–1380.

Yan, F., Dridi, M. and El-Moudni, A. (2009). A branch and bound algorithm for new traffic signal control system of an isolated intersection, *39th International Conference on Computers & Industrial Engineering, CIE39, Troyes, France*, pp. 999–1004.

Yan, F., Dridi, M. and El-Moudni, A. (2012). New vehicle sequencing algorithms with vehicular infrastructure integration for an isolated intersection, *Telecommunication Systems* **50**(4): 325–337.



Fei Yan received the M.Sc. degree in mechanical engineering and automation from Northwestern Polytechnical University, China, in 2004 and the Ph.D. degree in computer engineering from the University of Technology of Belfort–Montbeliard (UTBM), France. He is currently a lecturer at the School of Information Science & Technology, Southwest Jiaotong University, China. His main research interests are artificial intelligence, combinatorial optimization, traffic control of intelligent vehicles, and performance analysis.

Mahjoub Dridi was born in Menzel Bourguiba, Tunisia, in 1976. He received the Eng. Dip. degree from ENSAIT, France, in 2000 and the M.Sc. degree in control and computer sciences from the University of Lille 1, Villeneuve d'Ascq, France, in 2001. He is currently an assistant professor at the University of Technology of Belfort–Montbeliard (UTBM) at the Systems and Transport Laboratory. His research is related to traffic control, production planning, computer science, optimization methods for discrete events systems, and operational research.

Abdellah El Moudni received the Ph.D. degree in automatic control from Lille University, France, in 1985. He is currently a full professor at the University of Technology of Belfort–Montbeliard (UTBM), France, where he is in charge of courses in automatic control. His research interests include transportation systems, nonlinear control theory, dynamic systems, control of discrete-event systems, and singular perturbation methods in discrete time.

Received: 28 July 2011
Revised: 21 February 2012
Re-revised: 29 June 2012