Yan, Fei; Dridi, Mahjoub; El Moudni, Abdellah

Combinatorial optimization algorithms for intelligent vehicle sequencing problem at an isolated intersection


International Journal of Combinatorial Optimization Problems and Informatics
Morelos, México

Available in: http://www.redalyc.org/articulo.oa?id=265224452001
Combinatorial optimization algorithms for intelligent vehicle sequencing problem at an isolated intersection

Fei Yan, Mahjoub Dridi and Abdellah El Moudni
Laboratoire Systèmes et Transports, France.
Université de Technologie de Belfort-Montbéliard, France.
{fei.yan, mahjoub.dridi, abdellah.el-moudni}@utbm.fr

Abstract. As the development of telecommunication technology and miniaturization technology, intelligent vehicles equipped with microprocessor devices become more and more popular. This encourages researchers to propose new strategies to efficiently improve the traffic situation, especially at intersections. In this study, we present a novel vehicle sequencing algorithm based on analysis of the information received in advance from each vehicle approaching an isolated intersection. The objective is to increase the throughput of the intersection. A Branch and Bound based algorithm is presented because of the combinatorial nature of the studied problem. Structural properties are carefully analyzed to simplify the search procedure for an optimal solution. Computational experiments and simulations are also carried out to evaluate the performance of the proposed algorithms.

Keywords: Isolated intersection, vehicular infrastructure integration, vehicle arrival time, Branch and Bound, heuristic.

1 Introduction

Since the first traffic controller was installed in London in 1868, various traffic control methods benefit significantly the development of the traffic systems. The most common means of traffic control at an isolated intersection is performed by traffic lights of different colors (green, amber and red) that are repeating periodically. Conflicts between traffic participants are prevented by dividing the cycle time in intervals allocated to traffic flows so that the conflicting flows don’t get the right-of-way in the same interval [1].

Normally, the traffic signal control for isolated intersection falls into two basic categories: pre-timed control strategy, which is also called fixed–time control, and the semi/fully traffic actuated control. For the first category, countless efforts have been made directed toward efficiently calculating and dividing a fixed cycle time for each intersection. For example, the well-known Webster’s report [2], the TRANSYT (Traffic Network Study Tool) system [3]. The second strategy, the traffic actuated control, which is also know as the traffic-responsive strategy, attracts more and more attention since the 80s of last century. This control strategy can change the divide of cycle time based on the real-time measurement like inductive loops. The most famous system of this kind is SCOOT (Split Cycle Offset Optimization Technique) [4]. It is also considered as the traffic-responsive version of TRANSYT. However, the two strategies are both 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 some differences between the last computed flow rates and actual vehicle arrivals.

Recent advances in wireless communication technology like Wi-Fi, WiMax, 3G, Zigbee, RFID, GPS and blue tooth provides major opportunities for the development of vehicle-to-vehicle (V2V) communication and Vehicle infrastructure Integration (VII). Meanwhile, miniaturization of computing devices and availability of Global Positioning System (GPS) make the vehicles with in-vehicle information system (IVIS) becoming more and more popular. Since the traffic control equipment has always followed technology development, it is nature to consider them in the traffic control strategies of isolated intersections.

Based on these developments of technology, many innovative traffic control strategies and algorithms were proposed and applied into reality. For instance, based on new embedded devices and communication technologies, car platooning investigations aiming at increasing the capacity of roads by reducing temporal distances between vehicles were extensively discussed [5] [6]. A reservation based system for alleviating traffic congestion at intersections is proposed in [7]. In [8], queuing
theory was implemented in traffic control, and the problem of scheduling traffic at an intersection was addressed by structuring the problem as a Markov decision process. A new approach of traffic control at intersection that requires communication with a center controller at intersection was proposed in [9]. This approach uses the information about the intended route of each vehicle that reaches the intersection.

In addition, there were also researches concerning predicting vehicle's accurate arrival time at the intersection with both the historical and real-time GPS vehicle location data [10]. Based on the knowledge of each vehicle's arrival time at intersection, algorithms in [11] aimed at scheduling approaching vehicles to pass the intersection taking into account each vehicle's arrival individually, but authors only considered the control strategy for a 2-way intersection.

One can note that with the in-vehicle information system, the knowledge of individual vehicle's precise status is realistic in modern traffic control. The utilization of this information changes the traffic control system to a discrete event problem. With the help of microprocessor-based controllers, the control devices can handle much more complex traffic situation. Meanwhile, since there are a great number of combinations of vehicles that can get the right-of-way simultaneously to pass an intersection, the optimal traffic control problem has more and more combinatorial nature. Consequently, the problem is transformed to a complex problem of combinatorial optimization.

This problem we study in this paper is based on the assumption of vehicle-infrastructure communication and the utilization of intelligent vehicles. Assuming that there is a controller in the center of the intersection and all vehicles equipped with IVIS are able to communicate with the center controller in some fashion. The vital telemetry data of each vehicle can be obtained by the controller once the vehicle enters the controller's control range. Basically, we assume knowledge of the vehicle's accurate arrival time at the intersection and the time each vehicle needs to pass through it. Decision of vehicle passing sequence can be made and broadcasted to all vehicles in the range. Vehicles will pass the intersection according to the received decisions instead of following the traditional traffic lights.

To make this problem more understandable and rigorous, we model this problem as a special single machine scheduling problem in which some jobs can be processed in parallel. Then we propose a Branch and Bound type algorithm and an efficient heuristic for finding an optimal or approximate vehicle passing sequence. Because of the real-time property is one of the most important demands of a traffic control system. The proposed algorithms need to find a good solution is a very short time. For this purpose, we study carefully the structural properties of this optimization problem to simplify the search procedure.

The rest of this paper is structured as follows: In section 2, a detail description of the studied problem is provided, along with some useful notations that will be used in the paper. Section 3 describes the algorithms. A numerical example is also given. Section 4 presents computational experiments and simulation results. Conclusions are drawn in Section 5. An outline for future work is also proposed.

2 Problem description

2.1 Intersection Configuration

In The intersection under consideration is presented in Fig. 1. This intersection appears frequently in real-word traffic networks. It is a four-approach intersection with through lanes (some also serve as right-turn lanes) and exclusive left-turn lanes. The controller is located in the center of intersection with same control range for four directions. Vehicles approaching the intersection should inform the center controller their data after entering the control range. Vehicle overtaking on same lane is not allowed and pedestrian is not considered.

Before studying this problem, some basic notions should be introduced. Typically, an intersection consists of a number of approaches and the crossing area. The objective of optimal traffic control at intersection is to transform input traffic flows into output ones while preventing traffic conflicts and maximizing the throughput (minimizing the vehicle delays).
At an isolated intersection, each approach may be used by several traffic streams. For example, in Fig. 2, the approach form west to east consists of three traffic streams (stream 1, 2 and 3). Each stream has its own lane and an independent queue (overtaking is not allowed). The path used by a traffic stream to traverse the intersection is called the trajectory. Trajectories of all streams are illustrated with dashed lines in Fig. 2. 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 when traversing the intersection (e.g., stream 3 and stream 6).

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, we call these two streams compatible streams. The lanes on which the two streams are moving are called compatible lanes. For example, stream 4 and stream 10 are compatible streams. On the other hand, when trajectories of two traffic streams do cross, the streams are in a conflict (e.g., stream 2 and 12), and their simultaneous movement through the intersection should not be permitted. All conflict points of traffic streams are indicated in Fig. 2. When several streams are compatible with each other, we call the set of these streams a compatible stream group. In this example, we can partition the 12 streams into four compatible stream groups:
Group 1: stream 1, 7
Group 2: stream 2, 3, 8 and 9
Group 3: stream 4, 10
Group 4: stream 5, 6, 11, and 12.

One should note that the division of these groups is not constant when traffic flow of a specific stream is much greater during a peak time (e.g., morning peak time or evening peak time). However, this division remains the same during each period. Thus, we consider in the following that the division of traffic groups does not change. During the traffic control process, we need to give the right-of-way to each compatible stream group and a corresponding time to make vehicles get through the intersection. This time is often called a phase (or stage). The object of traffic control at isolated intersection is to decide the sequence of phases and their corresponding time to maximize the throughput or minimize the vehicle delay. Typically, there is always a lost time between two phases to avoid interference between incompatible streams of consecutive phases. An example of the traffic control process is given in Fig. 3.

With the application of Vehicle infrastructure Integration (VII) technology, precise status of approaching vehicles is available for consideration. For our purposes, we suppose that at a start time $t_0 = 0$, there are $n$ vehicles in the control range approaching the intersection from different approaches, and the vital data of all vehicles in this range have been received by control device in no time. At a basic level, the information from each vehicle contains follow parts:

- Vehicle identification (ID): used to identify individual vehicle.
- Stream number: which stream the vehicle belongs to, i.e., which lane it is moving on.
- Precise vehicle arrival time: the precise time vehicle arrives at the stop line from $t_0$ without interference.
- Vehicle passing time: time interval 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 stop line until it reaches a safe distance with its follower on same line (in same stream). The time duration depends on the type of vehicle that is getting through the intersection. For example, trucks are slower than small vehicles, they need more time to change the speed and therefore, more time to accelerate until it can reach a safe distance for the following vehicle to move.

Our objective is to decide a vehicle passing sequence to minimize the evacuation time, i.e., the time to make all the vehicles in the control range get through the intersection. The decision procedure will perform each time when there is new vehicle detected in the range. The minimization of the evacuation time will naturally increase the throughput of intersection in consequence. It should be noted that the control technique may be applied to any intersection layout.

### 2.2 Scheduling model and useful notations

For the own properties of the problem and its combinatorial nature, we can easily treat this vehicle sequencing problem as a single machine scheduling problem. We model this isolated intersection as a single machine that can process parallel jobs. Each vehicle is modeled as a job and its arrival time and passing time can be modeled as the job release date and processing time, respectively. Jobs (vehicles) are partitioned into different families according to the group of compatible steams. All vehicles in each compatible stream group form a job family. For example, the vehicles in Fig. 2 can be treated as four families corresponding to the four groups. Since overtaking in same stream is not allowed, vehicles on same lane should traverse intersection in First-In-First-Out way. This can be modeled as the chain constraints of the single machine problem. Vehicles in same stream are treated as jobs in same chain. Jobs in different chains but same family can be processed in parallel, i.e., vehicles in same compatible group but different streams can use the intersection simultaneously. The lost time between two phases can be modeled as the family setup time which is only decided by the following family.
Therefore, the studied problem is modeled as a special single machine scheduling problem with that jobs can be processed in parallel. Since we need to evacuate the detected vehicles (jobs) as soon as possible, the objective of the scheduling problem is to minimize the makespan.

Suppose there are \( n \) vehicles that are partitioned into \( m \) families \( F_1, F_2, \ldots, F_m \) according to the compatible stream groups. The number of jobs in family \( F_i \) is \( n_i \), where \( 1 \leq i \leq m \). In each family, jobs are partitioned into at least one chain. For the reason of rigor and mathematical expression, we give the following notations:

- \( s_i \) \( \), the setup time of \( F_i \).
- \( r_{i,j} \) \( \), the \( j \)th integer release time of \( F_i \).
- \( p_{i,j} \) \( \), the processing time of \( F_i \). \( j \) is indexed according to the compatible release expectations of jobs in this family.

Note that each job in family \( F_i \) belongs to one and only one chain, i.e., \( \sum_{l=1}^{n_i} \{ n_{i,l,j} \} = n_i \). Only one job in same chain can be processed in the machine each time. The setup time \( s_i \) of family \( F_i \) should be incurred whenever there is a switch from processing a job in another family to a job in this family. Jobs in different chains of same family can be processed concurrently. The objective is to minimize the makespan. According to the standard classification scheme for scheduling problems, we denote this problem as \( 1 | p - \) jobs, chains, \( s, r, p | C_{\text{max}} \), where \( p - \) jobs means that jobs contained in same family can be processed in parallel.

The single machine scheduling problem with parallel jobs was first studied with the problem of scheduling semiconductor burnin operations for large scale integrated circuit manufacturing [12]. The problem is also called batching problem. [13] gave a comprehensive review about the usually used algorithms for this kind of problem. Besides, the single batching machine problem with constraints like release dates, family jobs have also been widely discussed. For example, the unbounded batching machine problem with family jobs and release dates to minimize makespan [14]. Some different constraints were studied in [13] [15].

Most of these researches about single machine scheduling with parallel jobs are based on the assumption of batch availability, which indicates a batch process is complete when the last job in same batch is processed. However, the vehicle sequencing problem presented above has its own distinct properties, such as a job (vehicle) process is complete when its own process is finished and the single machine can process only one job of same chain each time, etc. For these reasons, we suppose in our problem that each job can be processed after its own release date and will be available for later processing when its process completes. Meanwhile, family jobs, chain constraints and family setup times are also considered.

In fact, we can view the problem as processing jobs in groups. A job group \( G \) is defined as a set of jobs from same family and will be processed on the machine without the interruption of jobs in any other families. Jobs in a group \( G \) correspond with vehicles in one phase. Note that only jobs in same family can be put into one group. Thus, the final sequence will have at least \( m \) groups. We define the release date of a group \( G_6 \) as the earliest release date of jobs in it, i.e., \( r_6 = \min_{j \in G} \{ r_j \} \); the completion time of the group \( C_6 \) as the maximum completion time of all jobs in this group, i.e., \( C_6 = \max_{j \in G} \{ C_j \} \). Under these considerations, the optimal solution of the problem can be viewed as the optimal group sequence \( GS \). We need to partition jobs into different groups and then schedule these groups to minimize the makespan. The optimal solution of the problem can be noted as \( GS = \{ G_1, G_2, \ldots, G_b \} \), where \( b \geq m \).
3 Structural properties of the problem

As stated above, the optimization of the problem consists of two parts: partitioning jobs into different groups and scheduling these groups to minimize the makespan. In the decision making process, we can have an incomplete group sequence in which some jobs are already formed into groups and scheduled, other jobs are still waiting for the decision. We call this sequence partial group sequence in the following. Jobs in this sequence are 'grouped' jobs and the rest jobs are 'un-grouped' jobs.

Considering the combinatorial nature of the problem, we adopt a Branch and Bound based algorithm to find the optimal solution among all feasible solutions. Since the traffic-responsive control strategy has tight real-time constraints, e.g. each decision making process within 2s for advanced control systems, the search procedure should be limited in a very short time. For such purpose, A Branch and Bound algorithm can be improved in two ways: diminishing the search node with the own properties of the problem and providing efficient lower and upper bounds to cut branches. To achieve that end, the structural properties of this scheduling model are carefully studied in this section. These properties will be used to simplify the search procedure of the Branch and Bound algorithm present in next section.

For the studied problem, the optimal group sequence can be described as

$$\pi = (G_1, G_2, L, G_k), b \geq m$$  \hspace{1cm} (1)

Suppose there is a partial group sequence, in which some groups have been formed, but no decision has been taken yet on grouping the remaining 'un-grouped' jobs. Let $C_{G_i}$ be the completion time of the last job group $G_r$ in the partial group sequence that already processed (i.e., the makespan of the partial sequence), and job $J_{(i,j,y)}$ be the first 'un-grouped' job in chain $l_{(i,j)}$ of family $F_i$ after $C_{G_i}$. Then $\sum_{j=y}^{n_{(i,j)}} p_{(i,j,y)}$ is the sum of processing time of jobs from $J_{(i,j,y)}$ to the last job in chain $l_{(i,j)}$, where $n_{(i,j)}$ is the number of jobs in $l_{(i,j)}$. A natural lower bound of processing all 'un-grouped' jobs in family $F_i$ should be the setup time of $F_i$ plus the maximum $\sum_{j=y}^{n_{(i,j)}} p_{(i,j,y)}$ of all chains in family $F_i$, i.e.,

$$S_i = \max_{1 \leq i \leq l} \{ \sum_{j=y}^{n_{(i,j)}} p_{(i,j,y)} \}$$  \hspace{1cm} (2)

Set $P_{(i,j)} = \sum_{j=y}^{n_{(i,j)}} p_{(i,j,y)}$ and re-index the chain that has the maximum $P_{(i,j)}$ after $C_{G_i}$ as the first chain of family $F_i$, i.e., $l_{(i,1)}$. We can then have the following properties.

**Property 1.** There is an optimal solution, in which any group $G_x$ from family $F_i$ should have at least one job belonging to the first chain $l_{(i,1)}$ after $C_{G_i}$, where $1 \leq i \leq m$ and $C_{G_i}$ is the completion time of the partial group sequence before $G_x$.

**Proof.** By contradiction. Suppose that there are only two chains in family $F_i$. $G_x$ is a group from $F_i$ in the optimal solution and the completion time of the group before $G_x$ is $C_{G_i}$. Let $J_{(i,j,y)}$ and $J_{(i,j,y)}'$ denote the first 'un-grouped' job in the first chain $l_{(i,1)}$ and chain $l_{(i,1)}$ of family $F_i$ after $C_{G_i}$, respectively. In this case, $l = 2$, $J_{(i,j,y)}$ means $J_{(i,2,y)}$. An example is given in Fig. 4.
Suppose that \( G_x \) contains only one job \( J_{(i,y)} \). It's easy to see that after group \( G_x \) is formed, the lower bound of processing all 'un-grouped' jobs in family \( F_i \) is the maximum \( P_{(i,j)} \) of these two chains plus the setup time of \( F_i \), i.e., \( P_{(i,1)} + s_i \). We can obtain a new group sequence \( GS' \) by moving \( J_{(i,y)} \) to the group that contains \( J_{(i,1)} \). One can notice that this change can reduce the makespan. Thus, \( GS' \) is also optimal. Continuing this procedure, we can eventually have an optimal group sequence with the property above. The proof can be easily applied to the family with more than two chains.

In other words, this property indicates that there is at least one optimal solution, in which each group contains at least one job from the chain has the maximum \( P_{(i,j)} \). Note that the chain has the maximum \( P_{(i,j)} \) may change each time a new group is formed. Thus the maximum \( P_{(i,j)} \) should be re-calculated after the process of a new group in family \( F_i, 1 \leq i \leq m \). Since there is only one chain that has maximum \( P_{(i,j)} \) at a time, one can clearly observe that this property stands even \( F_i \) has more than two chains. For reason of clarity, we only consider two chains in the proof of the following properties.

With this property, we can also have a property about relations between jobs with different chain number of same family.

Property 2. Suppose \( G_x \) is a group from family \( F_i \) in the required group sequence and \( G_x \) is the last group before it, where \( 1 \leq x \leq b \) and \( 1 \leq i \leq m \). If any of the following conditions stands:

1. \( C_{(i,y)} \leq C_{(i,y)} \) or
2. \( 0 \leq C_{(i,y)} - C_{(i,y)} \leq P_{(i,j)} - (P_{(i,1)} - p_{(i,1,y)}), l \neq 1 \),

there is an optimal solution in which \( J_{(i,y)} \) and \( J_{(i,y)} \) are both contained in \( G_x \), where \( C_{(i,y)} \) and \( C_{(i,y)} \) are the completion time of the first job in the first chain \( \ell_{(i,1)} \) and chain \( \ell_{(i,1)}, (l \neq 1) \) after \( C_{G_x} \), respectively.

**Proof.** Suppose there are only two chains in family \( F_i \). We can deduce that:

\[
C_{(i,y)} = \max\{C_{G_x} + s_i, r_{(i,1,y)}\} + p_{(i,1,y)}
\]

and

\[
C_{(i,y)} = \max\{C_{G_x} + s_i, r_{(i,2,y)}\} + p_{(i,2,y)}
\]

By property 1 we know that there is an optimal group sequence in which \( G_x \) contains job \( J_{(i,y)} \). We discuss the two cases separately.
Case 1. $C_{l(i,y)} \leq C_{l(i,y)}$ (See Fig. 5 as an example). It is clear to see that the containing of job $J_{i(y)}$ in group $G_x$ does not cause any extra time to process the rest jobs. Otherwise, it will probably cause extra time to process $G_x$ in one of the groups after $G_x$.

Case 2. $0 \leq C_{l(i,y)} - C_{l(i,y)} - p_{l(i)} + (P_{l(i)} - p_{l(i,y)})$, $\forall i \neq 1$. Note that after time $C_{G_{l(i,y)}}$, a lower bound of processing all `un-grouped' jobs in $F_1$ is $P_{l(i)} + s_i$. If $C_{l(i,y)} < C_{l(i,y)}$ (see Fig. 6 as an example) and group $G_x$ contains both $J_{i(y)}$ and $J_{i(y)}$, this lower bound will be increased from $P_{l(i)} + s_i$ to $P_{l(i)} + s_i + (C_{l(i,y)} - C_{l(i,y)})$ because of containing of $J_{i(y)}$. However, if $G_x$ does not contain $J_{i(y)}$, this lower bound will be increased to $P_{l(i)} + s_i + (P_{l(i)} - p_{l(i,y)})$. Since we have $0 \leq C_{l(i,y)} - C_{l(i,y)} - p_{l(i)} + (P_{l(i)} - p_{l(i,y)})$, we can deduce that there is an optimal group sequence in which both $J_{i(y)}$ and $J_{i(y)}$ are contained in one group.

One should note that this proof can be easily applied to the family with more than two chains.

The two properties above are mainly about the first `un-grouped' job in each chain of same family after the completion time $C_{G_{l(i,y)}}$ of a partial sequence. We now give a property with regard to all the `un-grouped' jobs of same family.

Property 3. There is an optimal sequence, in which a group $G_x$ contain all `un-grouped' jobs in $F_i$ after time $C_{G_{l(i,y)}}$ if we have $2s_i \geq C_{G_{l(i,y)}} - C_{G_{l(i,y)}} - p_{l(i,y)}$, where $C_{G_{l(i,y)}}$ is the completion time of $G_x$ when it contains all rest jobs of family $F_i$ after $C_{G_{l(i,y)}}$.

Proof. Suppose there are only two chains in family $F_i$, the last group before $G_x$ is $G_{l(i,y)}$ and its completion time is $C_{G_{l(i,y)}}$. An example is given in Fig. 7. In the example, there are three `un-grouped' jobs in chain $l_{(i,1)}$ and two jobs in chain $l_{(i,2)}$ after time $C_{G_{l(i,y)}}$. By property 1, there is an optimal sequence in which $J_{i(y)}$ is contained in $G_x$. 

Fig.5. Example of Property 2, case 1.

Fig.6. Example of Property 2, case 2.
Since $P_{(i,1)} + S_i$ is a lower bound of processing all 'un-grouped' jobs in $F_i$ after $C_G$, it's easy to see that $C_G - (C_G + S_i + P_{(i,1)})$ indicates the extra time caused by processing all 'un-grouped' jobs in $F_i$ if we put these jobs into $G_x$. Thus, if $S_i \geq C_G - (C_G + S_i + P_{(i,1)})$, the final makespan $C_{max}(GS)$ will be increased if we do not group all the five jobs into group $G_x$ because it will cost another $S_i$ to process rest jobs. Then the property stands.

The proposed three properties present the relations among jobs of the same family. In fact, we can consider that these properties may bind some jobs together as a new 'big' job in the search procedure. With these 'big' jobs, the number of elements will be sharply reduced.

4 Branch and Bound algorithm and a heuristic

In this section, we present a branch-and-bound algorithm for finding an optimal group sequence. An efficient heuristic is proposed to find a low initial upper bound. It can also be used independently to get satisfactory approximate solutions.

At beginning, the $P_{(i,1)}$ of each chain of all families are computed and we index the chain that has maximum $P_{(i,1)}$ as $(i,1)$ of family $F_i$, where $i \in [1,l]$ and $l \in [1,m]$. This computation and re-index procedure should be done each time after a group is formed during the searching process.

4.1 Branching scheme

In this branch-and-bound scheme, each node denotes a partial group sequence and it is partitioned into $k$ branches: one branch indicates the group just formed should add more jobs in same family if there are still jobs in it, other $k-1$ branches indicate that we give the authorization of using machine to jobs in other $k-1$ families, where $k \leq m$. The search tree is constructed in a depth-first fashion.

4.2 Fathoming and backtracking

In the proposed Branch and Bound algorithm, a node is fathomed if:

1. It is a leaf node, i.e., a complete solution.
2. The lower bound exceeds or equals the incumbent upper bound.

In searching process, if a complete solution has smaller makespan than the current upper bound is found, this makespan should be regarded as a new upper bound. Fathoming initiates backtracking to the first node that still not fathomed. If no such node is found, the search terminates.

4.3 Lower bound

Finding a tight lower bound is crucial to the Branch and Bound algorithm. We can obtain the lower bound of each node by three parts.
1. **LBS**: Time used by the jobs already grouped. It is the completion time of last group that already processed in a partial group sequence. We can be easily obtained by the jobs already partitioned processed (partitioned).

2. **LBR**: The lower bound of the time that will be used by rest jobs. In a partial group schedule, where some jobs have been formed in groups, but no decision has been taken yet on grouping the remaining ‘un-grouped’ jobs, LBR can be obtained by the following equation:

   \[
   \text{LBR} = \sum_{i=1}^{m} (S_i + P_{(i,1)}) - S_j
   \]  

   (3)

   Where \(S_j\) is the setup time of the family containing the last job in the partial schedule that already formed.

3. **LBP**: In order to enhance the exclusion rate of the search procedure, a value named Group Penalty is found attaching to each partial group sequence. Before presenting the Group Penalty algorithm, some further notations are given.

Suppose that \(G_r\) (from \(F_j\)) is the last group of the partial group sequence that already formed. In the job-grouping and group-sequencing procedure, each time a new group \(G_x\) (from \(F_i\), \(i \neq j\)) is formed and added to the partial sequence, the obtained new partial sequence will certainly has a bigger LBS. Ideally, this increment will equal to the sum of processing time of all jobs contained in the first chain \(l_{(i,1)}\) of group \(G_x\). However, the job release date will probably cause an extra time to LBS. Let \(q_i\) denotes this extra loss time, it can be computed by following equation.

\[
q_i = C_{G_x} - (S_o + C_{G_r}) - P_{G_x}
\]  

(4)

where \(P_{G_x}\) is the sum of processing time of jobs in the first chain \(l_{(i,1)}\) of group \(G_x\). Similarly, if we add some new jobs (contained in \(G_x\)) of same family into the last group \(G_r\) of partial sequence, i.e., \(i = j\) of the case above, define \(q_j\) as the extra time caused by \(G_x\).

\[
q_i = C_{G_x} - C_{G_r} - P_{G_x}
\]  

(5)

Then, for any partial group sequence with the last group \(G_r\) from family \(F_j\), the Group Penalty of this partial sequence, which noted as \(\Delta\) can be obtained by the following algorithm.
Algorithm 4. Group Penalty

Begin
/* initialization parameters, \( \Delta \leftarrow \infty \), \( \Delta' \leftarrow \infty \) */
do
for \( F_i, i \in [1, m] \) and \( i \neq j \)
  Form a group \( G_x \) from the ‘un-grouped’ jobs in \( F_i \), add to the partial sequence, count \( q_i \);
  if there are ‘un-grouped’ jobs in \( F_j \)
    \[ \Delta_i = q_i + q_j + s_j \]
  else
    \[ \Delta_i = q_i + q_j \]
  \[ \Delta' \leftarrow \min\{\Delta, \Delta'\} \]
\[ \Delta \leftarrow \min\{\Delta, \Delta'\} \]
Add new jobs to \( G_x \) from \( F_j \) by virtue of property 1~3. Re-count \( C_{G_x} \);
while there still are ‘un-grouped’ jobs in \( F_j \)
End

For each partial group sequence, the group penalty is the estimation of the minimum extra time wasted by the rest jobs. It can be used to form the lower bound of the following Branch and Bound Algorithm.

4.4 Initial upper bound (heuristic)

Providing a low initial upper bound is important for the efficiency of a Branch-and-Bound algorithm, i.e., it increases the rate with which nodes are fathomed. Thus, it is worth expending some computational effort to achieve that end. Moreover, such an upper bound can be used as a stand-alone heuristic for solving the problem. The algorithm of finding initial upper bound is given as follows.

Algorithm 5. Initial upper bound (IUB) (Heuristic)

Begin
/* \( C_{G_x} \leftarrow 0 \) */
while there still are ‘un-grouped’ jobs, do
for family \( F_i, i \in [1, m] \),
  Re-index the chain \( l_{(i, j)} \) in \( F_i \);
  Form a group \( G_x \) in \( F_i \) by virtue of property 1~3, count \( q_i \);
  Add \( G_x \) of family \( F_i \) that has \( \min\{q_i\} \);
  \( C_{G_x} \leftarrow C_{G_x} \);
IUB \leftarrow C_{G_x} ;
End
Since the proposed initial upper bound can find a complete solution in a very short time with a low deviation, algorithm 5 can be used as an independent heuristic for this problem, the computational experiments will present the performance of the heuristic.

4.5 Numerical example

To make the algorithm more understandable, we give the following example to illustrate its searching procedure.

Consider from a start time, there are 15 jobs going to be processed on the single machine and jobs are partitioned in three families. The setup time of three families are $S_1 = 1, S_2 = 2, S_3 = 3$. All parameters are measured in seconds. The release date and the processing time of jobs are given in Table 1.

<table>
<thead>
<tr>
<th>$F_1$</th>
<th>$F_1$</th>
<th>$F_2$</th>
<th>$F_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J_{(1,1,1)}$</td>
<td>$J_{(1,1,2)}$</td>
<td>$J_{(1,2,1)}$</td>
<td>$J_{(1,3,1)}$</td>
</tr>
<tr>
<td>$r$</td>
<td>1</td>
<td>5</td>
<td>23</td>
</tr>
<tr>
<td>$p$</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

Initial lower bound: At the beginning, lower bound $LBS = 0s$. Lower bound of ‘un-grouped’ jobs $LBR$ can be computed by equation (3) with $LBR = 24s$. Lower bound $LBP$ is obtained by algorithm 4. Since there is no scheduled group at beginning, we suppose that the previous groups is from family $F_1$, $F_2$, $F_3$ separately, and choose the smallest group penalty. Finally, we have $LBP = 3s$. This penalty is obtained by processing jobs as $\{J_{(1,1,1)}\}; \{J_{(2,1,1)}, J_{(2,2,1)}\}; \{J_{(1,1,2)}, J_{(1,2,1)}\}; \{J_{(1,3,1)}\}; \{J_{(2,1,2)}, J_{(2,2,2)}\}; \{J_{(3,1,2)}\}; \{J_{(3,2,1)}\}; \{J_{(3,2,2)}\}$ (note that jobs enclosed by $|$ are jobs in one group). Thus, the initial lower bound is $LB = LBS + LBR + LBP = 27s$.

Initial upper bound: Calculated by algorithm 5, $IUB = 32s$. It is obtained by the sequence: $\{J_{(1,1,1)}\}; \{J_{(2,1,1)}, J_{(2,2,1)}\}; \{J_{(1,1,2)}, J_{(1,2,1)}, J_{(1,3,1)}\}; \{J_{(2,1,2)}, J_{(2,2,2)}\}; \{J_{(3,1,2)}\}; \{J_{(3,2,1)}\}; \{J_{(3,2,2)}\}; \{J_{(1,1,3)}, J_{(1,2,2)}\}$.

Search procedure: The search tree is presented in Fig. 8 and the index of the node corresponds to the sequence of the search procedure. Some important nodes are listed as follows. An explanation is provided whenever necessary.
Node 1: Group obtained: \((1,1,1)\), \(LBS = 4\), \(LBR = 20\), \(LBP = 3\), \(LB = 27\).

Node 2: Group obtained: \((1,1,1), (1,2,1)\), \(LBS = 8\), \(LBR = 17\), \(LBP = 3\), \(LB = 28\).

Node 5: Group obtained: \((1,1,1), (1,2,1)\), \(LBS = 13\), \(LBR = 14\), \(LBP = 4\), \(LB = 31\).

Node 7: Group obtained: \((1,1,1), (1,2,1)\), \(LBS = 20\), \(LBR = 11\), \(LBP = 0\), \(LB = 31\).

Node 9: Group obtained: \((1,1,1), (1,2,1)\), \(LBS = 26\), \(LBR = 5\), \(LBP = 0\), \(LB = 31\).

Node 10: A complete group sequence: \((1,1,1), (1,2,1)\), \(LBS = 20\).

After the first leaf node is found, the incumbent upper bound is changed to 31s. Other nodes are fathomed by this upper bound in the following search procedure. A final optimal solution is obtained by Node 10. This optimal solution is illustrated in Fig. 9.

**4.6 Algorithm application**

The proposed Branch and Bound algorithm and heuristic aim at finding an optimal or approximate solution of the vehicle sequencing problem, i.e., minimizing the evacuation time of a set of vehicles in the control range.

Since at the real-world intersections, vehicles keep entering the control range of the center controller; the algorithm should be executed whenever new vehicles are detected. However, if a group of vehicles are authorized to pass intersection, the recalculation process should be postponed until all vehicles in that group have passed through the intersection.

Consider the numerical example above. At initial time 0s, 15 vehicles (jobs) are detected in the control range, algorithm is executed to get an passing sequence. If at 12s, a new vehicle (job) of compatible group 2 (family \(F_2\)) enters the control range. Since at \(t = 12s\), the vehicles in family \(F_3\) have the right-of-way (\((3,1,1)\) and \((3,2,1)\)), the algorithm should be re-executed without these two vehicles, and the new detected vehicle should be considered to decide a new passing sequence.

**4.7 Computational experiments**

Since algorithms should be applied each time a new vehicle enters the control range in the real-world traffic flow, the running time of the control algorithm should be short enough to satisfy the need of the real-time system. Experiments are done to evaluate the computational performance of the Branch and Bound algorithm and the heuristic in running time. Accuracy of the heuristic is also calculated.

The computation experiments (See Table. 2) are implemented at an isolated four-approach intersection. Without loss of the generality, we assume that there are 2 lanes, 3 lanes and 4 lanes for incoming vehicles in each of the four approaches. Number of vehicle compatible groups is 4. Table 2 indicates the average computation time of the Branch and Bound algorithm and the heuristic with the assumption that the number of vehicles in the range is varied from 10 to 100. Besides, we assume the loss time (setup time) for each compatible group is randomly generated integers varied from 3 to 8 seconds. Passing time of vehicles (processing time) is varied from 2 to 8 seconds. Vehicles in each compatible group are equally distributed among the streams.
To illustrate the performance of the heuristic in terms of accuracy, experiments are also carried out to test the maximum deviation between heuristic and Branch and Bound algorithm for each computation.

All approaches are coded in C++ and run on a desktop computer with Linux system (kernel 2.6.28).

### Table 2. Computation time and deviation of proposed algorithms

<table>
<thead>
<tr>
<th>Number of lanes</th>
<th>Number of vehicles</th>
<th>Branch and Bound</th>
<th>Heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>B&amp;B aver</td>
<td>B&amp;B min</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>0.023</td>
<td>0.013</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>0.040</td>
<td>0.040</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.320</td>
<td>0.120</td>
</tr>
<tr>
<td></td>
<td>75</td>
<td>1.922</td>
<td>0.330</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>5.643</td>
<td>0.600</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>0.014</td>
<td>0.008</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>0.027</td>
<td>0.010</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.018</td>
<td>0.030</td>
</tr>
<tr>
<td></td>
<td>75</td>
<td>0.144</td>
<td>0.040</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.260</td>
<td>0.060</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>0.013</td>
<td>0.009</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>0.028</td>
<td>0.026</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.035</td>
<td>0.020</td>
</tr>
<tr>
<td></td>
<td>75</td>
<td>0.149</td>
<td>0.100</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.173</td>
<td>0.138</td>
</tr>
</tbody>
</table>

The results illustrate the following facts:

1. For most cases, the proposed Branch and Bound algorithm can handle nearly 100 vehicles in its control range in a very short time.
2. For the same number of vehicles, the more streams a compatible group has, the less time algorithm uses. This is because more vehicles can be partitioned into same group during the search procedure, and the computation time reduces in consequence.
3. Heuristic performs much better than the Branch and Bound algorithm in running time with the maximum deviation of 8.35% in each computation.

### 4.8 Algorithms simulation

The fundamental measures for evaluating the performance of a traffic control algorithm at isolated intersection include the evacuation time, average queue size and average vehicle waiting time. The average queue size indicates the number of vehicles on each lane waiting to cross the intersection at same time. Average vehicle waiting time measures how long a vehicle has to wait before traversing the intersection. All the three measures are frequently used to evaluate the performance of a control algorithm. In this subsection, we analyze the three measures with different relative traffic loads. Comparisons are done among the Branch and Bound algorithm, the heuristic and an optimized traditional fix-cycle traffic light control scheme.

The simulation is implemented at an isolated four-approach intersection; each approach has two lanes for incoming vehicles. Vehicles approaching the intersection are partitioned into four compatible stream groups. According to statistics, we consider that the maximum traffic load for each of the four approaches is 1800 vehicles/h (one vehicle every 2 seconds) and the traffic load for each incoming lane is quarter of the maximum load of one approach. Other configurations are the same as in the computational experiments. In addition, each data point is obtained by taking the average over several separate simulations. Each simulation runs 30 minutes of traffic flow.

Since the algorithm aims at evacuating approaching vehicles as soon as possible, we first simulate the overall evacuation time of all vehicles passed the intersection during the 30 minutes. The result is presented in Fig. 10. We observe that the Branch and Bound algorithm and the heuristic reduce significantly the overall evacuation time for about 150 seconds, and the throughput can be improved in consequence.
The comparisons of average queue size and average vehicle waiting time are given in Fig. 11 and Fig. 12. One can notice that the Branch and Bound algorithm still performs much better than heuristic in reducing the average queue length and average vehicle waiting time. Both algorithms improve remarkably the traffic situation. However, for some data points, the heuristic performs a little better than the Branch and Bound which is an exact searching method, this is because our objective function is to minimize the overall evacuation time, not the average queue size or average vehicle waiting time.

Fig. 10. Simulation results of evacuation time for 10 min of traffic flow

Fig. 11. Simulation results of average queue length
The results illustrate that with the proposed control strategy and algorithms, we can reduce sharply the evacuation time of vehicles approaching an isolated intersection, as well as the average queue length and waiting time. The algorithms can perform smoothly during testing time without any breakout.

5 Conclusions

This paper proposed a new approach to sequence intelligent vehicles to pass an isolated intersection via V2I communications. Vehicles were treated as discrete individuals in the control strategy and our objective was to evacuate detected vehicles as soon as possible. We modeled this problem as a special single machine scheduling problem that jobs in same family can be processed in parallel, setup times and chain constraints were also considered. A Branch and Bound algorithm and a heuristic were developed based on the carefully analysis of structural properties of the problem. The efficiency of both methods was studied. Results showed the efficiency of the proposed algorithms, and their average running time can satisfy the need of a real-time control system.

Our future work will study the differences between normal vehicles and special used vehicles such as ambulances, police cars. Special used vehicles should have the privileges to pass through intersection as quickly as possible. Pedestrian crossings will also be taken into consideration. Further more, several neighbor intersections will be controlled together by one controller to save the computational resources as well as guarantee a more global optimization.

References