International Journal of Operations Research

International Journal of Operations Research, Vol. 8, No. 1, 57–68 (2011)

A Scheduling Approach for Autonomous Vehicle Sequencing Problem at Multi-Intersections

Fei Yan[1,*], Mahjoub Dridi[1], and Abdellah El Moudni[1]

[1] Laboratoire Systèmes et Transports, Université de Technologie Belfort-Montbeliard, Belfort - 90010, France

Received July 2010; *Revised December 2010, January 2011*; *Accepted January 2011*

Abstract— This paper addresses the problem of scheduling autonomous vehicles to traverse several adjacent intersections with the consideration of Vehicle to Infrastructure (V2I) communication technology. Due to the combinatorial nature of the control strategy, we decentralize the problem into several isolated intersection control problems and model each of them as a special single machine scheduling problem with family jobs that can be processed simultaneously. Chain constraints and release dates are also involved. Two frequently used traffic control measures average queue length and average vehicle waiting time are modeled as the number of late jobs and total tardiness, respectively. Efficient Branch and Bound algorithms are proposed for two objectives. A heuristic that serves as initial upper bound is presented. It can also be used independently. Computational experiments and simulations show the performance gain obtained by using the proposed schemes.

Keywords— Intelligent vehicle control, single machine scheduling, release date, branch and bound, heuristic.

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 in virtually all transportation modes (Hall 2003). Especially over the last decade, the traffic congestion attracted extensive attention because of the worldwide energy crisis and environmental concerns.

Since in modern cities, the congestions are usually caused by an isolated intersection with high volume of traffic flow or several adjacent intersections located in dense street networks, traffic control strategies for isolated intersections and intersection networks are mostly studied in literature to reduce the congestions.

In general, the common means of traffic control in modern cities is the control by traffic signals. They made it possible to "solve" conflicts between traffic flows at intersections (Guberinic, Senborn et al. 2008). The traffic signal control for intersections usually falls into two basic categories: pre-timed control strategy, which is also called fixed–cycle control, and the semi/fully traffic actuated control. The first category aims at calculating and dividing a fixed cycle time for each intersection. For example, 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, 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) (Hunt 1982). 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 big deviations between the last computed flow rates and actual vehicle arrivals. This encourages researchers to find new approaches for the traffic control at intersection.

With the significant progress achieved in the development of computer and telecommunication technologies during recent decades, more advanced control approaches used in traffic control systems were proposed under the framework of Intelligent Transportation System (ITS). For example, the development of wireless communication technology like Wi-Fi, WiMax, 3G, and blue tooth enabled the Vehicle to Vehicle communication (V2V) and Vehicle

* Corresponding author's email: fei.yan@utbm.fr

IJOR Vol. 8, No. 1, 57–68 (2011)

to Infrastructure (V2I) communication technologies. Meanwhile, miniaturization of computing devices and availability of Global Positioning System (GPS) made the intelligent vehicles with in-vehicle information system (IVIS) become more and more popular. Besides, there are also some researches that focus on predicting vehicle's accurate arrival time at intersection with both the historical and real-time GPS vehicle location data (see (Chin-Woo, Sungsu et al. 2008)). In this context, the autonomous vehicles start to attract more and more interests. Advanced strategies based on autonomous vehicles have been studied to solve the congestion problem (Wunderlich et al. 2008; Li et al. 2006).

In this paper, we study the autonomous vehicle sequencing problem at an intersection network with the consideration of V2I communication. All vehicles are assumed to be equipped with IVIS that can communicate with infrastructure of intersections, and they can traverse the intersection autonomously when informed. The traffic control strategy can be briefly described as follows. Suppose there is a small intersection network that contains several adjacent isolated intersections. A center controller is located in the network and all vehicles equipped with IVIS are able to communicate with the controller in some fashion. The vital telemetry data of each vehicle can be obtained by the controller once the vehicle enters the control range. At a basic level, we assume knowledge of each vehicle's accurate arrival time at the intersection which it is going to traverse and the time each vehicle needs to pass through that intersection. Decision of vehicle passing sequence can be made and broadcasted to all vehicles in the range. Vehicles will pass their intersection according to the received decisions instead of following the traditional traffic lights.

One can note that with this control strategy, the traffic control problem changes to a discrete vehicle sequencing problem. Each intelligent vehicle can be monitored and controlled individually with several parameters such as position, intended direction, arrival time, etc. The controller analyzes all the parameters and decides a final vehicle passing sequence with a specific objective to improve the traffic situation. At the same time, we can observe that the discreteness brings us a great number of vehicle passing combinations. Efficient algorithms are needed to find the optimal vehicle passing sequence.

Considering the discrete and combinatorial nature of the problem, we decentralize this problem to several vehicle sequencing problems at isolated intersection and model each of them as a special single machine scheduling problem. Different objectives are studied. Efficient Branch and Bound algorithms are proposed to find an optimal vehicle passing sequence. Heuristic that served as initial upper bound is also presented. It can also be used independently to find a satisfying solution in a very short computation time. Computational experiments and simulations show the improvement obtained by using the proposed schemes.

The rest of this paper is structured as follows: a detailed description of the studied problem is provided in Section 2. We also give its scheduling model along with some useful notations that will be used in the sequel. Section 3 describes the Branch and Bound algorithms for each objective. Section 4 presents the results of computational experiments and simulations. Conclusions are drawn in Section 5.

2. PROBLEM DESCRIPTION AND SCHEDULING MODEL

2.1 Problem Description

The studied intersection network consists of several adjacent intersections that may have different layout. An illustration is given in Figure 1. There are four isolated intersections covered by the control range of a center controller. The information of each vehicle is gathered by the controller in real-time and vehicle passing sequences are decided inside the center controller. Since the layout between two adjacent intersections are fixed (number of lanes, distance, etc.), we can easily decentralize the multi-intersection control problem to several vehicle sequencing problems at isolated intersection. This also reduces the complexity of the decision making process in center controller, which can ensure the needs of an advanced traffic control system.

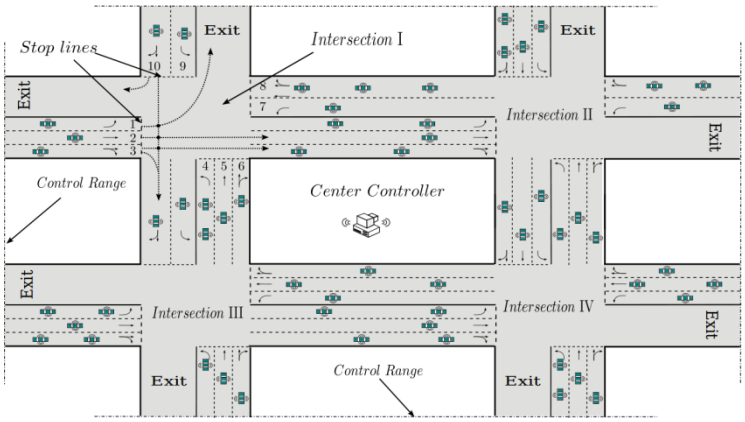

Figure 1. Schematics of the Intersection Network under Consideration

IJOR Vol. 8, No. 1, 57−68 (2011)

We now focus on the vehicle sequencing problem at an isolated intersection. At first, some basic notions should be introduced. Typically, an intersection consists of a number of approaches and the crossing area. Each approach may be used by several traffic streams. For example, the intersection I in Figure 1, the approach from 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 (with dashed lines). 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 while traversing the intersection (e.g., stream 3 and stream 10). The objective of optimal traffic control at 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. We call these two streams compatible streams. The lanes on which the two streams are moving are called compatible lanes. For example, stream 1 and stream 7 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 10), and their simultaneous movement through the intersection should not be permitted. Some conflict points of traffic streams are indicated in Figure 1. 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 10 streams into four compatible stream groups: CSG 1: stream 1, 7; CSG 2: stream 2, 3 and 8; CSG 3: stream 4 and 9, and CSG 4: stream 5, 6 and 10. 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 usually remains the same during a specific period.

Besides, there is always a lost time when we switch the right-of-way between two vehicles of different CSGs to avoid interference between incompatible streams. During the lost time, no vehicle behind the stop line is allowed to pass the intersection. The traffic control process is to decide a sequence of distributing 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 the intersection from different approaches, and the vital data of all vehicles in this range can be received by control device immediately. 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 lane (in same stream). This time interval 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.

Typically, there are three measures for evaluating the performance of a traffic control approach: the evacuation time, average queue size and average vehicle waiting time. The evacuation time of an intersection can be seemed as an objective related to the throughput. A Dynamic Programming algorithm and a Branch and Bound algorithm were proposed to find an optimal vehicle passing sequence with minimizing the evacuation time in earlier work (Yan, Dridi et al. 2009; Yan, Dridi et al. 2010). 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 traffic control strategy. In this paper, our objective is to decide a vehicle passing sequence to minimize the average queue size or average vehicle waiting time.

2.2 Scheduling Model

For our purpose, we model the studied 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. The time each job should finish the traversing procedure without any delay can be modeled as the job due date. It equals to the vehicle arrival time plus its passing time.

Jobs (vehicles) are partitioned into different families according to the group of compatible steams. All vehicles in same CSG form a job family. For example, in Figure 1, the vehicles approaching intersection I can be treated as four families corresponding to the four CSGs. 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

IJOR Vol. 8, No. 1, 57−68 (2011)

processed in parallel (vehicles in same compatible group but different streams can use the intersection simultaneously). In vehicle passing sequences, the lost time between two adjacent vehicles from different CSGs can be modeled as the family setup time which is only decided by the following family. Two measures average queue size and average vehicle waiting time are modeled as number of late jobs and total tardiness, respectively.

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

- l_i, the number of chains in family F_i.
- $l_{(i,l)}$, the l^{th} chain in F_i, where $1 \le l \le l_i$.
- $n_{(i,l)}$, the number of jobs in chain $l_{(i,l)}$.
- s_i, the setup time of F_i.
- $J_{(i,l,j)}$, the j^{th} job in chain $l_{(i,l)}$. j is indexed according the release sequence of jobs in each chain.
- $r_{(i,l,j)}$, the integer release date of $J_{(i,l,j)}$.
- $p_{(i,l,j)}$, the integer processing time of $J_{(i,l,j)}$.
- $d_{(i,l,j)}$, the due date of job $J_{(i,l,j)}$, i.e., $d_{(i,l,j)} = r_{(i,l,j)} + p_{(i,l,j)}$.
- $C_{(i,l,j)}$, the completion time of job $J_{(i,l,j)}$.

Note that each job in family F_i belongs to one and only one chain, i.e., $\sum_{1 \le l \le l_i} \{n_{(i,l)}\} = n_i$. Only one job in same chain can be processed on the machine each time. The setup time s_i of family F_i should be incurred at start of the sequence and 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. Besides, since our definition of due date is the sum of job release date and its processing time, a job can be only on time or late. According to the standard classification scheme for scheduling problems (Graham, Lawler et al. 1979), we denote this problem by $1|\ p-jobs, chains, s_i, r_j\ |\ h$, where $p-jobs$ means that jobs contained in same family can be processed in parallel. h represents the U_j or T_j which correspond the number of late jobs or total tardiness, respectively.

3. PROPOSED ALGORITHMS

Over the last several decades, there has been significant interest in the single machine scheduling problems with jobs can be processed simultaneously. These problems always involve an element of batching (see (Monma and Potts 1989)). In 1992, (Potts and Wassenhove 1992) give a review that combines scheduling with batching together. (Webster and Baker 1995) present an overview of algorithms and complexity results for scheduling batch processing machines. They call the problem with processing jobs concurrently the burn-in model (or batching machine model), which is motivated by the problem of scheduling semiconductor burn-in operations for large scale integrated circuit manufacturing (Lee, Uzsoy et al. 1992). In this model, the processing time of each job is equal to the maximum processing time of any job assigned to it. All jobs in same batch start and complete at the same time. There are two variants of the burn-in model: bounded and unbounded, which depends on whether the size of batches is bounded by a constant or not. (Brucker, Gladky et al. 1998) give a detailed discussion about the two models.

In the recent decades, (Potts and Kovalyov 2000) present an updated review about the usually used algorithms of batching problem for the case of single machine, parallel machine, job shop and open shop. (Cheng and Kovalyov 2001) describes the complexity results for various related problems and objectives, also considering due dates, but not release dates. Some researches with several constraints like release dates, family setup times were also studied (see (Liu and Yu 2000; Cheng, Yuan et al. 2005) and (Brucker and Kovalyov 1996) for example).

However, for our case, the scheduling model has its distinct features. If we view this problem as processing jobs in passing groups (PG), a passing group PG can be defined as a set of jobs from same family and processed on the machine without the interruption of jobs in any other families. Note that only jobs in same family can be put into same passing group. Since there may be several chains in one PG, jobs in same chain of this passing group should be processed consecutively and jobs in different chains may be processed simultaneously. Thus, the processing time of this PG does not equal to the maximum processing time of any job in this group. Meanwhile, family setup times and job release dates are also involved. Thus, the frequently used models can not meet the needs of the studied problem.

IJOR Vol. 8, No. 1, 57−68 (2011)

In fact, with the definition of passing group PG, the final job sequence will have at least m PGs. We define the release date of a passing group r_{PG} as the earliest release date of jobs in it, i.e., $r_{PG} = \min_{j \in PG}\{r_j\}$ and the completion time of the passing group C_{PG} as the maximum completion time of the all jobs in this PG, i.e., $C_{PG} = \max_{j \in PG}\{C_j\}$. Under these considerations, the optimal solution of the problem can be viewed as the optimal Passing Group Sequence PGS, i.e., $PGS = (PG_1, PG_2, \cdots, PG_b)$, where $b \geq m$. These passing groups are separated by setups. The decision procedure can be viewed as "grouping" jobs of each family into several passing groups and sequencing all passing groups of different families to get an optimal vehicle sequence.

3.1 Minimizing Number of Late Jobs and Total Tardiness

In literature, the problem of single machine with minimizing the number of late jobs has already been studied extensively. The problem can be denoted by $1\|\sum U_j$. It can be solved in $O(n\log(n))$ time by algorithm proposed in (Moore 1968). However, if jobs have release dates ($1|r_j|U_j$), the problem becomes strongly NP-hard (Lenstra, Kan et al. 1968). Some researches also concerns about the case with family setup times. For example, (Bruno and Downey 1978) show that for the binary NP-hard problem $1|s_f|U_j$ with arbitrary number of families.

For minimizing the number of late jobs, (Monma and Potts 1989) and (Crauwels, Potts et al. 1996) give some useful properties than can be used to develop a Branch and Bound algorithm. However, with the job chain constraints in each family and the presence that jobs in different chains of same family can be processed simultaneously, those properties cannot be used any more and the optimal solution do not even follow the sequence of the form $S = (E, L)$ in literature, where a partial sequence E of early jobs is followed by a partial sequence L of late jobs. This urges us to find new properties of an optimal sequence to reduce the search space.

For the problem with minimizing total tardiness on a single machine, i.e., $1\|\sum T_j$, its computational complexity remained open until its NP-hardness in ordinary sense was established by 1987 (Du and Leung 1990). Researches with this objective always concern about some special cases such as the release date (Baptiste, Carlier et al. 2004), setup times (Luo and Chu 2006), etc.

From these researches, we can easily deduce that our problem with minimizing the number of late jobs and total tardiness are at least binary NP-hard for arbitrary number of families. However, according to the isolated intersection configuration, family number is usually constant during a specific time, we intend to use Branch and Bound algorithm to find an optimal solution for both the problem with U_j and T_j.

Suppose PG_x is a passing group in an optimal sequence and jobs in PG_x are from family F_i. The completion time of the last passing group before PG_x is C_{PG_r}. We can have the following property about the first "UN-grouped" job in each chain of same family after the completion time C_{PG_r} of a partial sequence.

Property 1. *For problem* $1|p-jobs, chains, s_i, r_j|h$, *there exists an optimal passing group sequence, in which any passing group* PG_x *contains the first "UN-grouped" job with the minimum completion time of all chains.*

Proof. By contradiction. The proof is presented for the problem $1|p-jobs, chains, s_i, r_j|U_j$, and the problem with total tardiness is similar. Suppose there is a partial sequence, in which some passing groups have been formed, but no decision has been taken yet on grouping the remaining "UN-grouped" jobs. The completion time of the last passing group in the partial sequence is C_{PG_r}. Assume that there are only two chains in F_i, i.e., $l_{(i,1)}$ and $l_{(i,2)}$. Let the first "UN-grouped" job after C_{PG_r} in these two chains be $J_{(i,1,y)}$ and $J_{(i,2,y')}$, respectively. See Figure 2 as an example. In the example, the completion time of $J_{(i,1,y)}$ is the smallest.

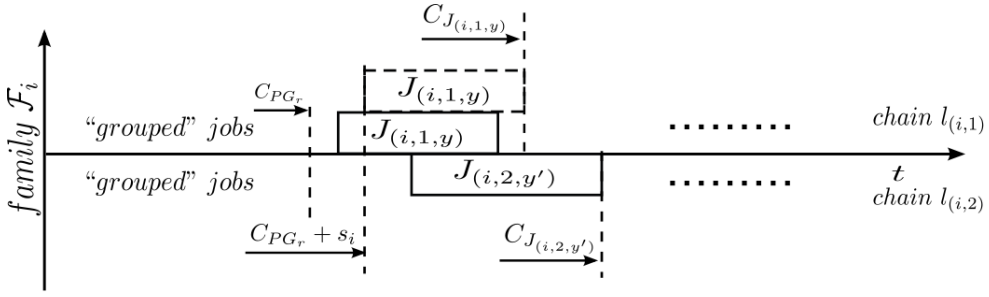

Figure 2. Example of Property 1.

IJOR Vol. 8, No. 1, 57−68 (2011)

Suppose that there exists an optimal sequence PGS in which PG_x contains only one job $J_{(i,2,y')}$, we can obtain a new passing group sequence PGS' by inserting job $J_{(i,1,y)}$ into PG_x. Obviously, this change does not increase the number of late jobs after C_{PG_r}, and then PGS' is optimal too. Continue this procedure; we will get an optimal passing group sequence as the property above.

One can clearly observe that this property stands even F_i has more than two chains. By Property 1, we can easily have the following corollary.

Corollary 2. *For problem* $1|\ p-jobs, chains, s_i, r_j\ |h$, *if there are several jobs of same family that have the same starting time and processing time, there exists an optimal sequence that these jobs are contained in same passing group.*

Note that for the studied problem, the jobs of same family with same starting time or release dates must be located indifferent chains. The proof is similar as that in Property 1.

3.2 Branch and Bound Algorithm and Heuristic

Based on the results given above, we propose a Branch and Bound algorithm for the problem $1|\ p-jobs, chains, s_i, r_j\ |h$ as follows.

3.2.1 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 last group just formed should be inserted more jobs of same family if there are still "UN-grouped" jobs in it; other $k-1$ branches indicate that we give the authorization of using machine to jobs in other $k-1$ families that still have jobs waiting for process, where $k \le m$. Each time considering jobs of family F_i, $1 \le i \le m$, the new node is going to contain the job/jobs that satisfy with Property 1 or Corollary 2. The search tree is constructed in a depth-first fashion.

3.2.2 Fathoming and Backtracking

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

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

In searching process, if a complete solution has smaller objective value (i.e., smaller number of late jobs or total tardiness) than the current upper bound is found, this value 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.

3.2.3 Lower Bound and Initial Upper Bound

For a partial group sequence with the last passing group PG_r from family F_i, suppose the group adjacently before PG_r is PG_p. C_{PG_r} and C_{PG_p} is the completion time of PG_r and PG_p, respectively. Clearly, PG_p is not from F_i. The proposed lower bound of this partial sequence consists of two parts:

- LBS (or LBS'): the number of late jobs (or the total tardiness) in the partial sequence that already scheduled, which can be easily counted by checking the starting time and release time of each job in the partial sequence. Note that since the last passing group in partial sequence PG_r may contain more jobs from F_i, we only count LBS (or LBS') from the beginning of the partial sequence to the last job in PG_p.
- LBR (or LBR'): the lower bound of late jobs (or the total tardiness) for the rest "UN-grouped" jobs,. This can be computed by comparing the release date with the minimum starting time of each "UN-grouped" job. If the minimum starting time is bigger than its release date, we count the job as a job that will be late (or count its minimum tardiness). This process is given in Figure 3.

We can observe that the lower bound LBR / LBR' consists of two parts: the number of late jobs (sum of tardiness) which are going to have in family F_i, and that in other families. The final lower bound LB / LB' can be computed by the sum of LBS / LBS' and LBR / LBR', i.e.,

$$LB = LBS + LBR,\ LB' = LBS' + LBR' \tag{1}$$

IJOR Vol. 8, No. 1, 57−68 (2011)

The algorithm for the lower bound LBR / LBR' is given as follows:

Algorithm 1. Lower bound LBR / LBR'

Begin

/* $LBR \leftarrow 0$, $LBR' \leftarrow 0$, $start \leftarrow 0$ */

For $f = 1$ to m with $f \neq i$

For $l = 1$ to l_f

$start \leftarrow C_{PG_r} + s_f$ if $f \neq i$; otherwise $start \leftarrow C_{PG_p} + s_f$;

For $j = y$ to $n_{(f,l)}$, /* $J_{(f,l,y)}$ is the first "UN-grouped" job in $l_{(f,l)}$ */

If $start > r_{(f,l,j)}$

$LBR \leftarrow LBR + 1$ (for the problem U_j);

$LBR' \leftarrow LBR' + (start - r_{(f,l,j)})$ (for the problem T_j);

$start \leftarrow start + p_{(f,l,j)}$;

End

Figure 3. Algorithm of the Lower Bound LBR / LBR'

For the proposed Branch and Bound algorithm, we give the following algorithm to find an initial group sequence. The initial upper bound can be obtained from this initial sequence to increase the rate with which nodes are fathomed.

Each time considering jobs in a family, we group the jobs that satisfy Property 1 or Corollary 2 to a temporary group. Then add the temporary group that has the earliest release date (ERD) of all families. Let PGS be the partial sequence that already grouped; this procedure can be described in Figure 4.

Algorithm 2. Initial Sequence for U_j **and** T_j

Begin

/* $C_{PG_r} \leftarrow 0$, $PGS \leftarrow \varnothing$ */

while there still are 'un-grouped' jobs, **do**

$J \leftarrow \varnothing$;

for family F_i , $i \in [1, m]$

Add the job/jobs in F_i which can satisfy the Property 1 or Corollary 2 to J ;

Add the job with earliest release date in J to PGS ;

$C_{PG_r} \leftarrow C_{PGS}$;

End

Figure 4. Algorithm for Finding an Initial Sequence

The initial upper bound of number of late jobs and total tardiness can be easily obtained from this initial sequence. One should note that we can also use this procedure as an independent heuristic.

3.2.4 A numerical example

In order to illustrate the search procedure, a numerical example of problem $1|\, p-jobs, chains, s_i, r_j \,|\, U_j$ is present as follows. Suppose there are 15 vehicles at time $t_0 = 0$ approaching the intersection and vehicles are partitioned into

IJOR Vol. 8, No. 1, 57−68 (2011)

three CSGs. The lost time of each CSG is $s_1 = 1s,\ s_2 = 2s,\ s_3 = 3s$. The arrival time and passing time of vehicles are given in Table 1.

Table. 1. Arrival time and passing time of vehicles in example

F_i	F_1						F_2				F_3				
J	$J_{(1,1,1)}$	$J_{(1,1,2)}$	$J_{(1,1,3)}$	$J_{(1,2,1)}$	$J_{(1,2,2)}$	$J_{(1,3,1)}$	$J_{(2,1,1)}$	$J_{(2,1,2)}$	$J_{(2,2,1)}$	$J_{(2,2,2)}$	$J_{(3,1,1)}$	$J_{(3,1,2)}$	$J_{(3,1,3)}$	$J_{(3,2,1)}$	$J_{(3,2,2)}$
r	1	5	23	2	15	7	4	18	5	17	2	19	25	2	20
p	3	3	4	4	2	3	2	2	2	2	1	2	1	1	2

By following the initial upper bound of the given algorithm, we can have an initial sequence with 13 late vehicles. The branching scheme will then start from root node with three new branches; each branch indicates the vehicle sequence will start from a CSG. The lower bound of the three new branches is 4, 5 and 7 late vehicles, respectively. Then the branching scheme will continue from the first branch (the branch with only one passing group from the first CSG) since it has the smallest number of late jobs.

Finally, we can have the optimal sequence with 8 late vehicles. The final sequence is given in Figure 5, in which the late vehicles (jobs) are $J_{(2,1,1)}$, $J_{(2,2,1)}$, $J_{(1,2,2)}$, $J_{(3,1,1)}$, $J_{(3,1,2)}$, $J_{(3,1,3)}$, $J_{(3,2,1)}$, $J_{(3,2,2)}$.

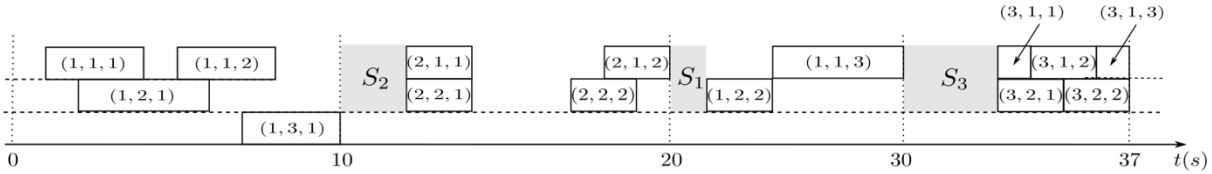

Figure 5. Final optimal sequence of example

The search procedure is similar with the problem $1|\ p-jobs, chains, s_i, r_j\ |T_j$.

3.3 Application of Algorithms and Coordination of Intersections

Since at the 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 approaching this intersection. However, if a group of vehicles have the right-of-way to pass an intersection, the recalculation process should be postponed until all vehicles in that group have passed through the intersection.

Since mentioned in (Hall 2003), for an advanced traffic control system, the decision making process should be executed for about each 2 seconds (for the efficiency of traffic control and the safety of drivers), the recalculation procedure should follow the rules below:

- If there are not any vehicles detected in 2 seconds, the recalculation will be executed when vehicle comes.
- If there are one or more vehicles detected within 2 seconds, the recalculation will be executed only once at end of the 2 seconds with the consideration of all new vehicles.

Besides, in the shared space between two neighbor intersections, for example, considering the lanes (from west to east) between intersection I and intersection II in Figure 1, the output traffic streams of intersection I is also the input traffic stream of intersection II. The vehicles already traversed Intersection I (from west to east) can be seemed as the new coming vehicles for intersection II, and vice versa. The average queue length and average vehicle waiting time can be obtained by the average of all vehicles in this intersection network.

Moreover, since the layout between two adjacent intersections are fixed in both number of lanes and distance. The time used by vehicles from upstream intersection to downstream intersection can be seemed as the same. Thus, when some vehicles are grouped together to pass the upstream intersection according to their arrival times and passing time, they can still be scheduled in same group if their intentions keep the same with each other (Figure 6). The computational time will also be reduced this way.

IJOR Vol. 8, No. 1, 57–68 (2011)

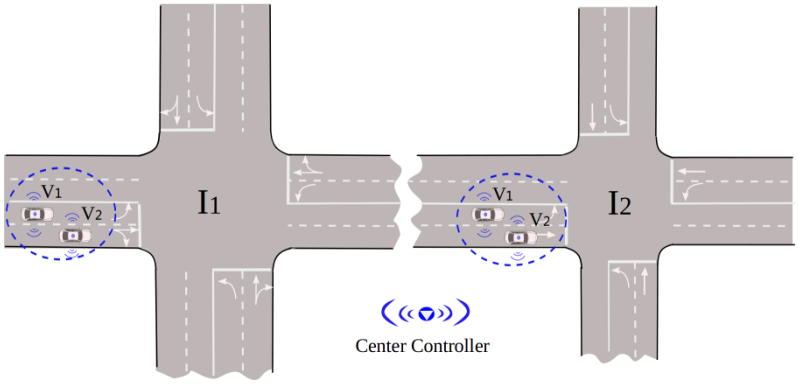

Figure 6. Illustration of same vehicle passing group from upstream to downstream intersection

4. COMPUTATIONAL EXPERIMENTS AND SIMULATIONS

4.1 Computational experiments

A traffic control system should satisfy the need of a real-time system; for example, the decision making process should be completed within 2 seconds. Thus, the running time of the proposed control algorithms should be tested. In this section, experiments are carried out to evaluate the computational performance of the proposed Branch and Bound algorithms and the heuristics. Accuracy of the heuristic is also calculated. The computational experiments are performed at an isolated intersection with four approaches and four compatible stream groups.

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 compatible stream groups. Table 2 shows the computation time for different objectives with the proposed Branch and Bound algorithms and heuristic. We assume that the number of vehicles approaching the intersection is varied from 10 to 50; the loss time (setup time) for each compatible group is randomly generated integers varied from 3 to 8 seconds and passing time of vehicles (processing time) are 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 2.6.32).

The results illustrate that for most cases, the proposed Branch and Bound algorithms can handle nearly 50 vehicles in its control range in a very short time. The CPU time augments with the number of lanes contained by each compatible stream group from 0.018s to 0.622s for minimizing U_j; and from 0.065s to 1.047s for minimizing the total tardiness. The efficient computation time makes it possible to apply the proposed Branch and Bound algorithms to a real-time control system. Heuristic performs much better than the Branch and Bound algorithm in running time with the maximum deviation of 9.82% in each computation.

All the computational experiments are performed for an upstream intersection, the Branch and Bound algorithms for downstream intersection will be more efficient since some vehicle groups from upstream intersection can be used directly.

Table 2. Computation Time for Different Objectives with B&B and Heuristic

l_i*	N	Minimizing U_j (in seconds)			Minimizing T_j (in seconds)			Heuristic	
		B&B average	B&B min	B&B max	B&B average	B&B min	B&B max	CPU time	Deviation
2	10	0.018	0.009	0.027	0.065	0.020	0.181	0.010	1.47%
	25	0.107	0.036	0.198	0.263	0.097	0.456	0.025	3.65%
	50	0.254	0.098	0.329	0.450	0.178	0.874	0.108	4.02%
3	10	0.164	0.062	0.245	0.111	0.093	0.255	0.036	2.86%
	25	0.259	0.104	0.377	0.379	0.174	0.646	0.072	4.72%
	50	0.366	0.153	0.625	0.612	0.286	1.024	0.186	6.65%
4	10	0.198	0.104	0.480	0.228	0.122	0.471	0.098	4.06%
	25	0.289	0.147	0.687	0.582	0.229	0.938	0.165	6.97%
	50	0.622	0.276	1.023	1.047	0.419	1.675	0.264	9.82%

* l_i represents the number of lanes contained in each compatible stream group (family).

IJOR Vol. 8, No. 1, 57–68 (2011)

4.2 Simulation Experiments

In this subsection, we analyze the two measures with different relative traffic loads. Comparisons are given among the Branch and Bound algorithm, the heuristic, an optimized fixed cycle and an actuated traffic light control scheme.

The simulation is implemented at an intersection network with four intersections (see Figure 1 for example). The distance between any two adjacent intersections is the same and all vehicles need 15s to travel from an upstream intersection to a downstream intersection. Each intersection has four approaches and each approach has two lanes for incoming vehicles. Vehicles approaching each intersection are partitioned into four CSGs. 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 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 several separate simulations. Each simulation runs 10 minutes of traffic flow.

By applying the Branch and Bound algorithm and heuristic with minimizing the number of late jobs, the simulation result of average queue size is presented in Figure 7. We can find that the new control strategy reduces the average queue length for more than 60% at certain traffic flow rates.

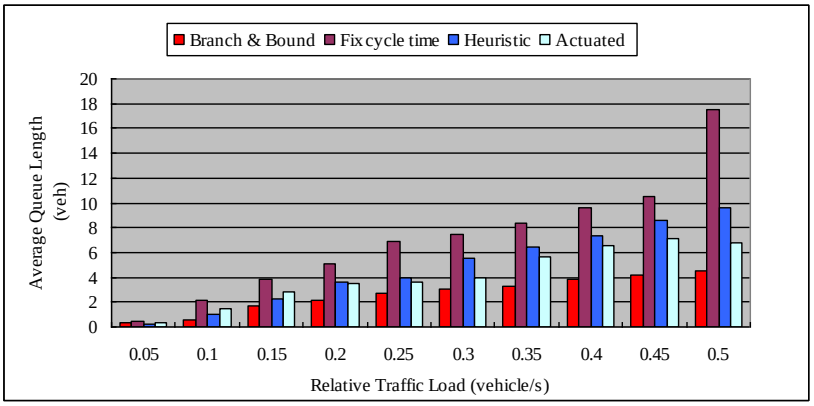

Figure 7. Average Queue Length for 10 minutes Traffic Flow

Meanwhile, the average vehicle waiting time of all vehicles in this network during 10 minutes by applying algorithms with minimizing total tardiness are shown in Figure 8. The results show that even for high traffic load (0.5 vehicle/s), the average vehicle waiting time before traversing the intersection network decreased from 67.1s to 20.4s. This means drivers of intelligent vehicles will spend about 70% less time in this area.

5. CONCLUSIONS

This paper proposed a new approach to sequence intelligent vehicles to pass an intersection network via V2I communications. The considered multi-intersections were decentralized to several isolated intersections. At each isolated intersection, vehicles were treated as discrete individuals in the control strategy and our objective was to minimize the average queue length or average vehicle waiting time.

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. Two objectives were considered. Branch and Bound algorithms and a heuristic were developed based on the analysis of structural properties of the problem. Results showed that their average running time can satisfy the need of a real-time control system. Simulations showed the significant improvement by applying the proposed algorithms.

IJOR Vol. 8, No. 1, 57–68 (2011)

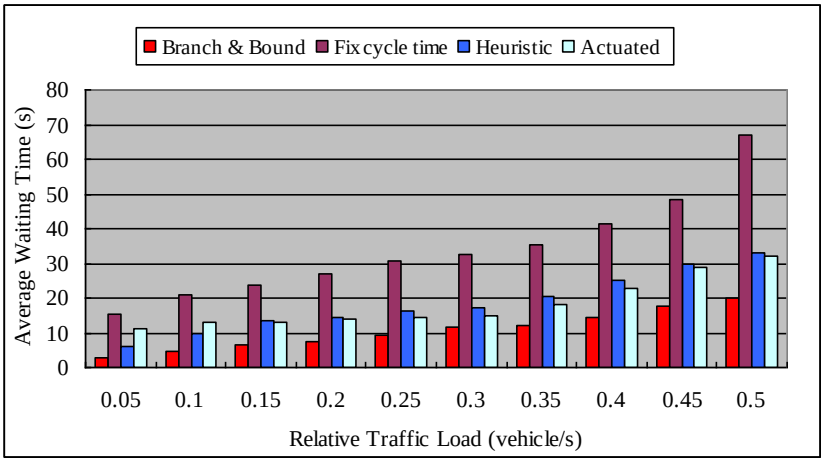

Figure 8. Average Waiting Time for 10 minutes Traffic Flow

6. ACKNOWLEDGEMENT

The authors are grateful for the constructive comments of the referees on an earlier version of this paper.

REFERENCES

1. Baptiste, P., Carlier, J. and Jouglet, A.. (2004). A Branch-and-Bound procedure to minimize total tardiness on one machine with arbitrary release dates, *European Journal of Operational Research,* 158(3): 595-608.
2. Brucker, P., Gladky, A., Hoogeveen, H., Kovalyov, M., Potts, C.N., Tautenhahn, T. and Velde, SVD.. (1998). Scheduling a batching machine, *Journal of Scheduling*, 1: 31-54.
3. Brucker, P. and Kovalyov, M. Y. (1996). Single machine batch scheduling to minimize the weighted number of late jobs, *Mathematical Methods of Operations Research,* 43: 1-8.
4. Bruno, J. and Downey P. (1978). Complexity of task sequencing with deadlines, set-up times and changeover costs, *SIAM Journal on Computing,* 7: 393-404.
5. Cheng, T. C. E. and Kovalyov, M. Y. (2001). Single machine batch scheduling with sequential job processing, *IIE Transactions,* 33(5): 413-420.
6. Cheng, T. C. E., Yuan, J. J. and Yang, A. F.. (2005). Scheduling a batch-processing machine subject to precedence constraints, release dates and identical processing times, *Computers and Operations Research,* 32(4): 849-859.
7. Chin-Woo, T., Sungsu, P., Liu, H., Qing, X. and Lau, P. (2008). Prediction of Transit Vehicle Arrival Time for Signal Priority Control: Algorithm and Performance, *IEEE Transactions on Intelligent Transportation Systems,* 9(4): 688-696.
8. Crauwels, H. A. J., Potts, C. N. and Van Wassenhove, L.N. (1996). Local search heuristics for single-machine scheduling with batching to minimize the number of late jobs, *European Journal of Operational Research,* 90(2): 200-213.
9. Du, J. and Leung, J.Y.T. (1990). Minimizing total tardiness on one machine is NP-hard, *Mathematics of Operations Research*, 15: 483-495.
10. Graham, R., Lawler, E., Lenstra, J. and Rinnooy Kan, A. (1979). "Optimization and approximation in deterministic machine scheduling: A survey, *Annuals of Discrete Mathematics,* 5: 287-326.
11. Guberinic, S., Senborn, G. and Lazic, E. (2008). *Optimal Traffic Control: Urban Intersections*, CRC Press, Taylor & Francis Group.
12. Hall, R. W. (2003). *Handbook of Transportation Science*, Springer.
13. Hunt, P. B. (1982). The SCOOT on-line traffic signal optimisation technique, *Traffic Engineering & Control,* 23: 190-192.
14. Lee, C.Y., Uzsoy, R. and Martin-Vega, L. A. (1992). Efficient algorithms for scheduling semiconductor burn-in operations, *Operations Research,* 40: 764-775.
15. Lenstra, J. K., Rinnoy Kan, A. H. G. and Brucker, P.. (1968). Complexity of machine scheduling problems, *Annuals of Discrete Mathematics,* 1: 343-362.
16. Li, L. and Wang, F.Y. (2006). Cooperative driving at blind crossings using intervehicle communication, Vehicular Technology, *IEEE Transactions on Vehicular Technology,* 55(6): 1712-1724.

IJOR Vol. 8, No. 1, 57−68 (2011)

17. Liu, Z. and Yu, W. (2000). Scheduling one batch processor subject to job release dates, *Discrete Applied Mathematics,* 105(1-3): 129-136.
18. Luo, X. and Chu, F. (2006). A branch and bound algorithm of the single machine schedule with sequence dependent setup times for minimizing total tardiness, *Applied Mathematics and Computation,* 183(1): 575-588.
19. Monma, C. L. and Potts, C. N. (1989). On the complexity of scheduling with batch setup times, *Operations Research,* 37: 798-804.
20. Moore, J. M. (1968). An n job one machine algorithm for minmizing the number of late jobs, *Management Science,* 15: 102-109.
21. Potts, C. N. and Kovalyov, M. Y. (2000). Scheduling with batching: A review, *European Journal of Operational Research,* 120(2): 228-249.
22. Potts, C. N. and Wassenhove, L. N. V. (1992). Integrating scheduling with batching and lotsizing: A review of algorithms and complexity, *Journal of Operational Research Society,* 43: 395-406.
23. Robertson, D. I. (1969). TRANSYT: A traffic network study tool, Crowthorne, England, Ministry of Transport.
24. Webster, S. and Baker, K. R. (1995). Scheduling groups of jobs on a single machine, *Operations Research,* 43: 692-703.
25. Wunderlich, R., Liu, C., Elhanany, I. and Urbanik, T. (2008). A novel signal-scheduling algorithm with quality-of-service provisioning for an isolated Intersection, *IEEE Transactions on Intelligent Transportation Systems,* 9(3): 536-547.
26. Yan, F., Dridi, M. and EI Moudni, A. (2009). Autonomous vehicle sequencing algorithm at isolated intersections, *12th International IEEE Conference on Intelligent Transportation Systems, ITSC '09*, 2009.
27. Yan, F., M. Dridi and EI Moudni, A. (2010). New vehicle sequencing algorithms with vehicular infrastructure integration for an isolated intersection, *Telecommunication Systems*, In press.