Vehicle Sequence Reordering with Cooperative Adaptive Cruise Control

Ta-Wei Huang*, Yun-Yun Tsai*, Chung-Wei Lin†, and Tsung-Yi Ho*
National Tsing Hua University, Hsinchu, Taiwan*
National Taiwan University, Taipei, Taiwan†
E-Mails: da811050@gmail.com, alice103000004@gmail.com, cwlin@csie.ntu.edu.tw, tyho@cs.nthu.edu.tw

***Abstract*—With Cooperative Adaptive Cruise Control (CACC) systems, vehicles are allowed to communicate and cooperate with each other to form platoons and improve the traffic throughput, traffic performance, and energy efficiency. In this paper, we take into account the braking factors of different vehicles so that there is a desired platoon sequence which minimizes the platoon length. We formulate the vehicle sequence reordering problem and propose an algorithm to reorder vehicles to their desired platoon sequence.**

I. Introduction

Automated Highway System (AHS) is an intelligent transportation system technology designed for intelligent vehicles, and one of its applications is vehicle platooning which organizes highway traffic into groups of close-following vehicles. It has been shown that vehicle platooning can improve traffic throughput, energy efficiency, and even safety (due to a lower relative speed) [1, 2].

To facilitate platoons, two important technologies have been introduced: Adaptive Cruise Control (ACC) [3] and Vehicle-to-Vehicle (V2V) communication [4, 5]. An ACC system with laser or radar sensors can obtain the distance to and the speed of the preceding vehicle and maintain a predefined spacing from the preceding vehicle. An ACC system can be further enhanced by V2V communication and information exchange with preceding vehicles. The transmitted information includes the vehicles' positions, velocities, and accelerations. With the information, a vehicle can follow its preceding vehicles with even better performance (a closer gap, a higher traffic throughput, etc.). Such a system is referred as Cooperative Adaptive Cruise Control (CACC).

The majority of relevant research focused on the development and deployment of platoon protocols [6–11]. Most existing studies assume that all vehicles in platoon are homogeneous, which means that there is no difference between vehicles. In a heterogeneous case, the spacing between two adjacent vehicles is highly related to vehicles' types, settings, and capabilities. Therefore, [12] demonstrated that inter-vehicle spacing varies with different pairs of vehicles by introducing the braking factor. As a result, the sequence order of vehicles in a platoon will affect the platoon length.

Regarding the ordering problem, [13] developed a protocol for cooperative merging that allows a vehicle to merge into a platoon from side. This approach is also used for merging two platoons at a lane-reduction scenario. [14] proposed a strategy to automate on-ramp entrances to the main road and maximize the length that the corresponding platoon stays intact. However, it is very difficult to maintain a large platoon size without vehicle reordering on highway. [15] developed an energy-oriented protocol for intra-platoon vehicle sequence optimization by minimizing the impact of the disturbances when vehicles join and leave the platoon.

With these findings, the platoon length can be further reduced with an optimized platoon sequence. However, to the best of our knowledge, none of existing work makes use of reordering vehicles to minimize the platoon length. Therefore, we propose a vehicle sequence reordering algorithm to reorder vehicles to their desired platoon sequence and minimize the total operation time. It should be emphasized that the desired platoon sequence not only minimizes the platoon length by reordering the vehicles but also serves as a general goal to group vehicles in a platoon. Different from minimizing the platoon length, the reordering goals of other scenarios include grouping vehicles with the same destination together, merging vehicles on two lanes (there is a logical order of all vehicles based on their estimated arrival times to the merging point), and forming a computational group sharing computational resource to each other (for example, [16]).

The rest of this paper is organized as follows. Section II introduces the definitions and the formulation of the vehicle sequence reordering problem. Section III presents the proposed algorithm. Section IV shows the experimental results, and Section V concludes the paper.

II. Definitions and Formulation

We assume that all vehicles have CACC and the ability to share information with each other. All communications and maneuvers are coordinated by a platoon leader. Vehicles may be heterogeneous, with known vehicle lengths and braking capabilities. The notations used in our models are shown in Table I.

A. Target Sequence Determination

Each vehicle in a platoon tries to maintain a safe distance to its preceding vehicle. The safe distance g_{safe} is decided by the maximum decelerations, *i.e.*, braking factors, of the two vehicles. The length of a platoon consists of the length L_v of each vehicle and the safe distance g_{safe} of every two adjacent vehicles. Given that two vehicles have the same speed, the

TABLE I
DEFAULT PARAMETERS FOR CACC VEHICLE PLATOONING.

Parameter	Description	Value
L_v	Vehicle length	5.0 m
G_{min}	Minimum space-gap	1.0 m
A^{cf}	Comfort acceleration limit	2 m/s^2
D^{cf}	Comfort deceleration limit	3 m/s^2
V_{int}	Intended speed	20 m/s
V_{max}	Maximum speed	30 m/s
C_f	Communication frequency	10 Hz

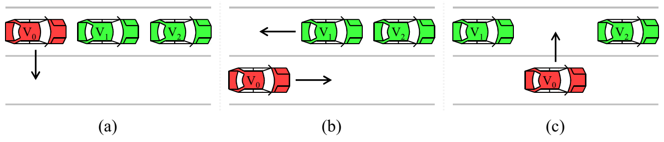

Fig. 1. Three steps of a change process: (a) V_0 first splits from the original platoon and performs lane change; (b) V_0 moves to the specific position on the adjacent lane, and V_1 also moves to the specific position and reserves an enough gap for V_0; (c) V_0 performs lane change again and merges back into the platoon.

minimum g_{safe} is obtained when the braking factor of a following vehicle is larger than that of the preceding vehicle. In order to minimize the platoon length, the target sequence is determined by the braking factors first and given as an input in our formulation in Section II-D.

B. Change Process

We assume that a special lane is reserved for platooned vehicles on the highway, and an adjacent lane is reserved for vehicles to change the sequence ordering. During a *change process*, some vehicles first split from the original platoon and perform lane change (Fig. 1(a)). Then, those vehicles move to the specific positions on the adjacent lane, and the other vehicles in the original platoon also move to the specific positions and reserve enough gaps for those vehicles on the adjacent lane (Fig. 1(b)). Lastly, those vehicles on the adjacent lane perform lane change again and merge back into the platoon (Fig. 1(c)). In the change process, platoon maneuvers are similar to those in previous work [8, 10, 13], including "merge," "split," "lane change," and "gap adjustment." The total operation time of the change process includes the time of the three steps in Fig. 1. For simplicity, we only consider the longest moving distance of each vehicle and transform the distance into the time by considering appropriate acceleration A^{cf}, deceleration D^{cf}, and maximum speed V_{max}. With minor modifications, the timing model can be generalized.

C. Move Constraint

The move constraint states that any two vehicles cannot collide while moving to their specific positions at the same time. Fig. 2 shows four example scenarios, and two of them violate the move constraint. In short, the vehicles that are chosen to move should keep the same ordering in the target sequence.

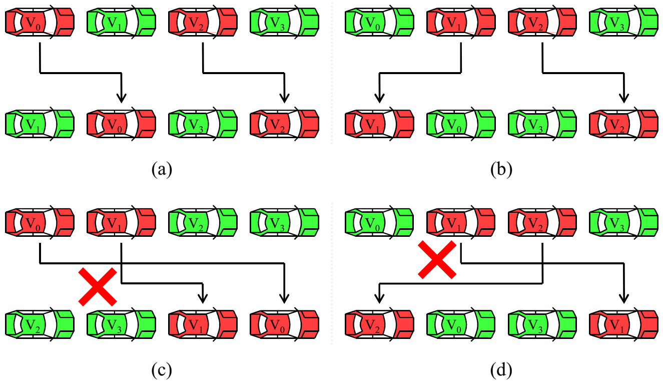

Fig. 2. Four example scenarios of change processes. (a)(b) The ordering of the chosen vehicles remains the same, so it satisfies the move constraint; (c)(d) the ordering of the chosen vehicles is different after the change process, so it violates the move constraint with a conflict.

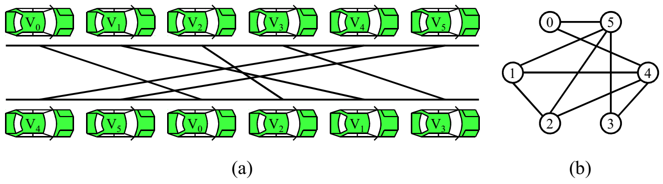

Fig. 3. (a) Intersection graph between two sequences and (b) its corresponding permutation graph.

D. Problem Formulation

Given the initial sequence S_i and the target sequence S_f (Section II-A), the vehicle sequence reordering problem is to transform S_i into S_f by the change processes (Section II-B) without violating the move constraint (Section II-C) and minimize the total operation time of all change processes.

E. Permutation Graph

The permutation graph is the model that we will utilize in our algorithm. S_i and S_f can form the permutation graph, as shown in Fig. 3, denoted by $G = (V, E)$. Each vertex $v \in V$ represents a vehicle in the platoon. Each edge $e \in E$ connecting v_i and v_j represents that v_i and v_j are reversed in two sequences. The number of edges in the permutation graph may increase or decrease during a change process, and the goal is equivalent to reducing the number of edges to zero.

III. PROPOSED ALGORITHM

A. Enumeration Method

We first introduce an enumeration method for the vehicle sequence reordering problem. In a change process, some chosen edges are removed, and some other edges become "restricted," *i.e.*, they cannot be removed in the current change process due to the move constraint. We continue doing these until each edge is either removed or restricted. The enumeration method tries to consider all combinations of removing edges in a change process. However, we may not obtain a solution by the enumeration method because removing edges may generate some additional edges. To prevent this and guarantee convergence, we perform the change process that the number

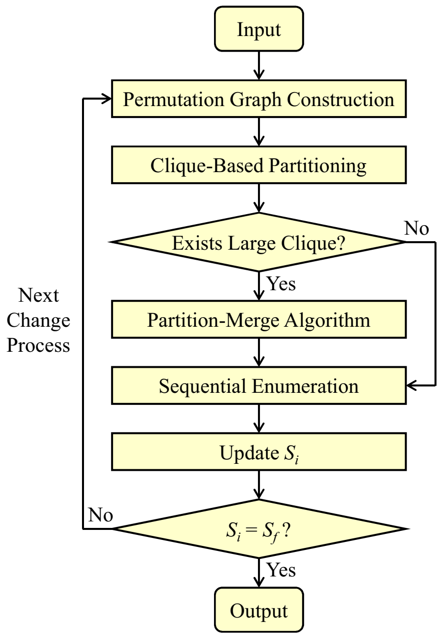

Fig. 4. Algorithm flowchart.

of edges will decrease, and the change process that the number of edges will increase is suspended.

B. Clique-Based Partitioning

The enumeration method is time-consuming for large graphs. To address this, we partition the graph into several subgraphs and apply the enumeration method iteratively. A clique can be seen as a representative set in the graph because the vertices in the clique are highly connected. The graph can be partitioned into several cliques by finding the largest clique and removing the clique from the graph iteratively, where it only takes polynomial time to find the largest clique in a graph. We then choose cliques and their connected edges as the subgraphs and apply the enumeration method.

C. Partition-Merge Algorithm

The enumeration method guarantees to obtain an optimal solution, but it is time-consuming. The clique-based partitioning is more efficient in general, but it still suffers the efficiency issue. Therefore, we further improve them and propose a more efficient Partition-Merge Algorithm. The flowchart is summarized in Fig. 4.

To deal with long runtime caused by large cliques, we propose a two-stage algorithm to adjust the size of cliques, *i.e.*, partition stage and merge stage, and both of them adjust the sizes of cliques. A given permutation graph is partitioned into several cliques which have different sizes. For a clique whose size is too large, it is partitioned into two smaller cliques. After partitioning, the remaining cliques are examined if some of them can be merged into a larger subgraph. By these two stages, we can improve time efficiency by adjusting the sizes of subgraphs. The detailed settings of the two stages are introduced in the following paragraphs.

Partition Stage. If the size of a clique is larger than an upper limit, then the clique will be partitioned into two parts iteratively until their sizes are smaller than the upper limit. This partition statge is based on the Fiduccia-Mattheyses (FM) algorithm [17]. The FM algorithm is a linear-time heuristic algorithm for weighted partitions. Each edge (v_i, v_j) in the clique is evaluated by a weight function $W_{i,j}$ as

$$W_{i,j} = t_{i,j} + N_r(i,j) - N_s(i,j), \tag{1}$$

where $t_{i,j}$ the time cost of moving the vehicle from position i to position j, $N_r(i,j)$ is the number of restricted edges caused from moving the vehicle from position i to position j, and $N_s(i,j)$ is the number of solved (removed) edges caused by moving the vehicle from position i to position j. The partition stage partitions the graph into two sides, and the objective is to minimize the total edge weight between two sides, implying the difficulty to solve those edges. To keep both sides balanced, the steps of the partition stage are: (1) randomly initialize both sides with the same number of vertices, (2) pick a vertex and change it to the other side if it reduces total edge weight between two sides, (3) lock the vertex, and (4) repeat until all vertices are locked.

Merge Stage. After the partition stage, the graph has been split into several subgraphs. However, solution quality may become bad if the sizes of subgraphs are too small. Therefore, to balance the sizes of subgraphs, each pair of subgraphs $(C_{i'}, C_{j'})$ is evaluated by an evaluation function $U_{i',j'}$ as

$$U_{i',j'} = \sum_{v_i \in C_{i'}, v_j \in C_{j'}} (e_{i,j}), \tag{2}$$

where $e_{i,j}$ is a 0-1 variable to represent if there is an edge between v_i and v_j. The evaluation function counts the edges between subgraphs as the evaluation value of merging the two subgraphs. The subgraphs of a pair with the highest evaluation value are merged into a larger subgraph which is not guaranteed to be a clique. After the merging, the evaluation repeats, and the merge stage stops when no subgraph can be merged. At the same time, the size of a new subgraph cannot exceed the upper limit after the merging. By the above stages, the graph is partitioned and merged into subgraphs with reasonable sizes.

IV. Experimental Results

A. Baselines

We compare our algorithm with two baselines. The first baseline is the Sequential Move which chooses one vehicle based on the target sequence and moves it to the specific position. Only one vehicle can move in a change process, and the other vehicles stay in the platoon. The second baseline is the Multi-Vehicle Move that chooses several vehicles in a change process without violating the move constraint. The Multi-Vehicle Move follows the Sequential Move but continues checking if the next change process can be combined with the current change process. If it does not violate the move constraint, the two change processes are combined as one change process.

TABLE II
EXPERIMENTAL RESULTS OF SEQUENTIAL MOVE, MULTI-VEHICLE MOVE, OUR ALGORITHM, WHERE THE OBJECTIVE IS THE TOTAL OPERATION TIME.

Scenario (Platoon size)	Sequential Move		Multi-Vehicle Move		Ours	
	Objective (s)	CPU Runtime (s)	Objective (s)	CPU Runtime (s)	Objective (s)	CPU Runtime (s)
Case 1 (10)	147.1	< 0.01	90.1	< 0.01	77.6	0.03
Case 2 (12)	194.7	< 0.01	118.3	< 0.01	95.1	0.05
Case 3 (14)	237.2	< 0.01	114.9	< 0.01	102.8	0.26
Case 4 (16)	310.1	< 0.01	187.2	< 0.01	136.2	3.74
Case 5 (18)	361.1	< 0.01	213.5	< 0.01	155.9	2.63
Case 6 (20)	470.2	< 0.01	285.0	< 0.01	207.2	10.19
Normalized Average	2.220	~ 0	1.302	~ 0	1	1

TABLE III
PLATOON LENGTH REDUCTION AFTER VEHICLE SEQUENCE REORDERING.

Length (m)	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
Original	33.3	41.1	48.5	56.1	62.9	70.1
Reordered	27.0	33.0	39.0	45.0	51.0	57.0
Reduced	6.3	8.1	9.5	11.1	11.9	13.1
Reduction	18.9%	19.9%	19.5%	19.7%	18.9%	18.6%

B. Experimental Settings

We implemented the basedlines and our algorithm in the C++ programming language on Xeon 3.50GHZ CPU with 128GB memory and tested them with six synthetic scenarios. For each scenario, we fix the number of vehicles and randomly generate 10 testcases (randomly generate braking factors and initial sequences). According to [18], the number of vehicles in the platoon can be as large as 20.

C. Experimental Results

Table II compares the objective (the total operation time) and CPU runtime of the Sequential Move, the Multi-Vehicle Move, and our algorithm, where the values are the averages of 10 testcases of each scenario. The objective of our algorithm is 2.220X better than Sequential Move and 1.302X better than the Multi-Vehicle Move. It shows that our algorithm can achieve better solution quality with reasonable runtime, but we agree that there is still space for the runtime improvement. It should be noted that the Partition-Merge Algorithm is needed as the clique-based partitioning cannot obtain solutions in some larger cases due to long runtimes, although the clique-based partitioning can get better solutions than the Sequential Move and the Multi-Vehicle Move. Table III shows the platoon length reduction after vehicle sequence reordering. The length is reduced by almost 20% on each case, which is beneficial to traffic throughput and traffic performance.

V. CONCLUSION

In this paper, we formulated the vehicle sequence reordering problem for platooning and proposed the Partition-Merge algorithm to minimize the total operation time. Experimental results showed the effectiveness of the proposed algorithm. This paper provided an initial solution to vehicle reordering for many different applications, such as grouping vehicles with the same destination together, merging vehicles on two lanes, and forming a computational group sharing computational resource to each other.

ACKNOWLEDGMENT

This work is partially supported by the Ministry of Education of Taiwan under Grant Number NTU-107V0901.

REFERENCES

[1] M. Michaelian and F. Browand, “Field experiments demonstrate fuel savings for close-following,” *UC Berkeley: California Partners for Advanced Transportation Technology*, 2000.

[2] B. Van Arem, C. J. Van Driel, and R. Visser, “The impact of cooperative adaptive cruise control on traffic-flow characteristics,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 4, pp. 429–436, 2006.

[3] R. Rajamani and S. E. Shladover, “An experimental comparative study of autonomous and co-operative vehicle-follower control systems,” *Transportation Research Part C: Emerging Technologies*, vol. 9, no. 1, pp. 15–31, 2001.

[4] T. Taleb, A. Benslimane, and K. B. Letaief, “Toward an effective risk-conscious and collaborative vehicular collision avoidance system,” *IEEE Transactions on Vehicular Technology*, vol. 59, no. 3, pp. 1474–1486, 2010.

[5] F. Bai and B. Krishnamachari, “Exploiting the wisdom of the crowd: localized, distributed information-centric vanets [topics in automotive networking],” *IEEE Communications Magazine*, vol. 48, no. 5, 2010.

[6] F. Bu, H.-S. Tan, and J. Huang, “Design and field testing of a cooperative adaptive cruise control system,” in *American Control Conference (ACC)*, pp. 4616–4621, IEEE, 2010.

[7] G. Naus, R. Vugts, J. Ploeg, R. van de Molengraft, and M. Steinbuch, “Cooperative adaptive cruise control, design and experiments,” in *American Control Conference (ACC)*, pp. 6145–6150, IEEE, 2010.

[8] V. Milanés, S. E. Shladover, J. Spring, C. Nowakowski, H. Kawazoe, and M. Nakamura, “Cooperative adaptive cruise control in real traffic situations,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 1, pp. 296–305, 2014.

[9] M. di Bernardo, A. Salvi, and S. Santini, “Distributed consensus strategy for platooning of vehicles in the presence of time-varying heterogeneous communication delays,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 102–112, 2015.

[10] M. Amoozadeh, H. Deng, C.-N. Chuah, H. M. Zhang, and D. Ghosal, “Platoon management with cooperative adaptive cruise control enabled by vanet,” *Vehicular communications*, vol. 2, no. 2, pp. 110–123, 2015.

[11] S. E. Shladover, C. Nowakowski, X.-Y. Lu, and R. Ferlis, “Cooperative adaptive cruise control: Definitions and operating concepts,” *Transportation Research Record: Journal of the Transportation Research Board*, no. 2489, pp. 145–152, 2015.

[12] Z. Wang, G. Wu, and M. J. Barth, “Developing a distributed consensus-based cooperative adaptive cruise control system for heterogeneous vehicles with predecessor following topology,” *Journal of Advanced Transportation*, 2017.

[13] E. S. Kazerooni and J. Ploeg, “Interaction protocols for cooperative merging and lane reduction scenarios,” in *18th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 1964–1970, 2015.

[14] R. Hall and C. Chin, “Vehicle sorting for platoon formation: Impacts on highway entry and throughput,” *Transportation Research Part C: Emerging Technologies*, vol. 13, no. 5-6, pp. 405–420, 2005.

[15] P. Hao, Z. Wang, G. Wu, K. Boriboonsomsin, and M. Barth, “Intra-platoon vehicle sequence optimization for eco-cooperative adaptive cruise control,” in *Proceedings of the 20th International IEEE Conference on Intelligent Transportation Systems (ITSC’17)*, 2017.

[16] T. Higuchi, J. Joy, F. Dressler, M. Gerla, and O. Altintas, “On the feasibility of vehicular micro clouds,” in *2017 IEEE Vehicular Networking Conference (VNC)*, pp. 179–182, Nov 2017.

[17] C. M. Fiduccia and R. M. Mattheyses, “A linear-time heuristic for improving network partitions,” in *19th Conference on Design Automation*, pp. 175–181, IEEE, 1982.

[18] P. Varaiya, “Smart cars on smart roads: problems of control,” *IEEE Transactions on automatic control*, vol. 38, no. 2, pp. 195–207, 1993.