EEPC: A Framework for Energy-Efficient Parallel Control of Connected Cars

Minghua Shen, *Member, IEEE*, Guojie Luo, *Member, IEEE*, and Nong Xiao, *Senior Member, IEEE*

Abstract—With the advanced communication sensors are deployed into the modern connected vehicles (CVs), large amounts of traffic information can be collected in real-time, which gives the chance to explore the various techniques to control the routing of CVs in a ground traffic network. However, the control of CVs often suffers from energy inefficiency due to the constant changes of network capacity and traffic demand. In this paper, we propose a cost-based iterative framework, named EEPC, to explore the energy-efficient parallel control of connected vehicles. EEPC enables the control of CVs to iteratively generate a feasible solution, where the control of each vehicle is guided in an energy-efficient way routing on its own trajectory. EEPC eliminates the conflicts between CVs with a limited number of iterations and in each iteration, EEPC enables each vehicle to coordinate with other vehicles for a same road resource of the traffic network, further determining which vehicle needs the resource most. Note that at each iteration, the imposed cost is updated to guide the coordination between CVs while the energy is always used to guide the control of CVs in EEPC. In addition, we also explore the parallel control of CVs to improve the real-time performance of EEPC. We provide two parallel approaches, one is fine grain and the other is coarse grain. The fine grain performs the parallel control of single-vehicle routing while the coarse grain performs the parallel control of multi-vehicle routing. Note that fine grain adopts multi-threading techniques and coarse grain adopts MPI techniques. The simulation results show that the proposed EEPC can generate a feasible control solution. Notably, we also demonstrate that the generated solution is effective in eliminating the resource conflicts between CVs and in suggesting an energy-efficient route to each vehicle. To the best of our knowledge, this is the first work to explore energy-efficient parallel control of CVs.

Index Terms—Cyber-physical system applications, connected vehicles, control and parallel control, energy-efficient control, real-time control

1 Introduction and Motivation

With the recent advances in electronics, sensors, and communication techniques are increasingly deployed in connected vehicles (CVs), modern transportation control systems have made significant progress during the past decade. Specifically, these advances have the ability to make it probable for CVs to apperceive their environments and to communicate with each other, further having more chances to perform the safe and efficient control of CVs. Moreover there are many companies and academic institutions have also started experimenting with modern CVs on intelligent transportation systems. While research on energy-efficient driving techniques, such as eco-driving, has already witnessed numerous efforts [1], it is still a challenge to design an autonomous control system, where each vehicle can coordinate with other vehicles so that the driving of each vehicle on a road network is the energy-efficient route with a safety guarantee.

The modern autonomous transportation control systems in metropolitan areas generally involve several phases as follows. First, users send their requests through the clients installed in their vehicles to the cloud servers. These requests, also called orders, mainly include the current locations and destinations. After receiving these orders, the control system running on the cloud servers will calculate and generate a feasible solution based on current network status and traffic demand in a holistic environment. And at meanwhile, the corresponding trajectory path of each vehicle will be published through wireless networks. With the changes in network status, the control system will update the current solution in real-time for vehicle safety and transportation efficiency. Although such control system has provided great convenience for vehicle driving, it still exists some non-negligible shortcomings. For instance, the choices are not always the most energy-efficient ones.

It is non-trivial to estimate the energy consumption of each vehicle in autonomous control systems [2]. The energy consumption should be estimated on each of the different road segments in ground traffic networks. Macroscopic and microscopic models are broadly applied to estimate the vehicle energy consumption [3]. In microscopic models, the vehicle acquires a larger amount of driving data to decide a statistical cost on each road segment. In macroscopic models, the vehicle only considers the driving time and road grade, which are typically easier to obtain through free or

- *M. Shen is with the School of Data and Computer Science, Sun Yat-Sen University, Guangzhou, Guangdong 510275, China, and also with the Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education, Guangzhou, Guangdong 510275, China. E-mail: shenmh6@mail.sysu.edu.cn.*
- *G. Luo is with the Center for Energy-Efficient Computing and Applications, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China. E-mail: gluo@pku.edu.cn.*
- *N. Xiao is with the School of Data and Computer Science, Sun Yat-Sen University, Guangzhou, Guangdong 510275, China. E-mail: xiaon6@mail.sysu.edu.cn.*

Manuscript received 19 Dec. 2018; revised 12 July 2019; accepted 16 July 2019. Date of publication 23 July 2019; date of current version 18 Dec. 2019. (Corresponding author: Minghua Shen.)
Recommended for acceptance by B. He.
Digital Object Identifier no. 10.1109/TPDS.2019.2930500

commercial historical databases. Specifically, the road-based macroscopic models depend heavily on the longitudinal dynamics of the vehicles and they are easier to calibrate using vehicle construction parameters [4]. And thus in this work, we mainly leverage the macroscopic road-based energy consumption model to explore the control of CVs so that each vehicle routes from one road segment to the adjacent ones.

Benefiting from the real-time updated data and information, we present EEPC, a cost-based iterative framework to provide the energy-efficient parallel control of CVs for autonomous control systems. EEPC maintains the energy-efficient control of CVs and supports the parallelization to obtain a good real-time performance. In EEPC framework, our control approach can balance the competing goals of eliminating conflicts among CVs and minimizing energy consumption of routing paths. The basic idea is to initially allow CVs to occupy the same road segment resource of ground traffic networks, but subsequently must coordinate with each other to determine which vehicle needs the occupied resource most. Note that our control approach imposes and adjusts the costs to the road resources to obtain a good resource assignment to these CVs. In addition, our EEPC supports parallelization to meet the real-time performance requirements of our control approach to CVs. Note that there are two grained parallel approaches, one is fine grain and the other is coarse grain. The fine grain focuses on the control of single-vehicle routing while the coarse grain studies the control of multi-vehicle routing. The contributions of this work are as follows:

- To the best of our knowledge, this is the first work to explore the energy-efficient parallel control of CVs for autonomous control systems.
- We present a cost-based iterative framework that supports energy-efficient control and its parallelization to CVs.
- We propose a conflict-aware control approach that not only eliminates the resource conflicts between vehicles but also provides an energy-efficient route to each vehicle.
- We present two multi-core parallel approaches, both of which enable the parallel control of CVs to obtain real-time performance improvement.
- We provide a simulation evaluation demonstrating the advantages of our EEPC framework over the widely used dataset.

This paper is an extension of our previous work[1] published at International Symposium on Low Power Electronics and Design (ISLPED) in 2017. Note that in this paper, we not only present novel insights into the initial control approach but also explore its parallelization to improve the real-time efficiency requirement of control approach for autonomous control systems.

The rest is organized as follows. The background and related work are in Section 2. The energy consumption model is described in Section 3. The framework is detailed in Section 4. The control approach is presented in Section 5 and its parallel exploration is shown in Section 6. The simulation results are given in Section 7 and the discussion is presented in Section 8, followed by the conclusion in Section 9.

2 Background and Related Work

2.1 Vehicle Control and Associated Terminology

The control of CVs is an important problem in modern autonomous transportation systems. It consists in assigning ground traffic resources to each vehicle of the autonomous control system in order to connect its current location to the destination. The resources in a ground traffic network and their connections are represented by a graph $G = (V, E)$. The set of vertices V corresponds to the road segment resources in the ground traffic network and the edges E to the feasible links that connect these nodes.

Typically, given a graph of ground traffic network, the weight associated with each node v is either the length of the road segment or vehicle travel time. In the energy-driven control study, each node of the graph is assigned a weight that represents the travel energy expenditure. Thus, we define a weighting function $w : V \rightarrow W$, which associates each node of the graph with a weight.

Given a vehicle i in the ground traffic network graph, the vehicle trajectory path T_i is the set of terminals, including the source terminal s_i and destination d_i. T_i forms a subset of V. A feasible solution to the control problem for vehicle i is the trajectory path T_i mapped onto the graph G and connecting s_i with its d_i.

2.2 Problem Statement

The goal of this work is to explore the energy-efficient control of CVs for modern autonomous transportation systems. Thus, the weight assigned to each node of the graph represents only the associated energy consumption. Furthermore, this weight can be easily extended to consider travel time as well, and the optimization would search then for a tradeoff solution between energy consumption and travel time minimization. In this paper, we solely focus on energy consumption aspects.

The control problem of CVs consists in the minimization of the energy consumption to route from a selected origin to a destination in the road network while eliminating the resource conflicts between vehicles. Minimization of an energy cost in a graph can be solved by means of a standard shortest path algorithm. However, it is challenge to eliminate the conflicts between vehicles in the same time to guarantee the traffic safety. Note that the control problem of CVs is a technology-specific variation of the disjoint path problem from graph theory, which is one of Karp's original NP-complete problems [7]. In a graph, two trajectory paths are disjoint if they do not occupy the same resource such as node or edge.

2.3 Parallel Programming Models

Multi-threading techniques enable parallel processing of multiple tasks within a shared memory. Multiple threads may operate at the same memory location, resulting in a conflict behavior. Synchronization is used to manage the

1. Minghua Shen and Guojie Luo. *Tiguan: Energy-Aware Collision-Free Control for Large-Scale Connected Vehicles*. in proceeding of IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED), 2017.

access of multiple threads to the shared memory and enables each thread to perform the processing of its own tasks at a specific program point. To explore the parallel control of single-vehicle routing, we can create the threads rapidly and synchronize the context between threads on a multi-core processor.

MPI techniques enable parallel processing of multiple tasks within a distributed memory. Each participated MPI process performs its own tasks and communicates with other MPI processes through MPI messages between different processor cores. During the communication, each MPI process sends and receives blocking or non-blocking MPI messages. The blocking message enables the MPI process to wait for the responds from other MPI processes and then performs the data synchronization between different MPI processes.

2.4 Related Work

Existing efforts related to our work are multi-agent control problem. Different from our work, these works leverage the idea of region partitioning and coverage control to control the routing of agents to perform the tasks in their specified regions [8], [9], [10]. Also, these works cannot fully overcome the challenge of conflicts between multiple agents, and the scale of agents is relatively small as well. Other works include multiple vehicles routing without communications and robust traffic flow management under uncertainty [11], [12]. Their task models and objectives are different from our energy-aware control problem. Moreover, these works do not consider the advances in communication techniques to enable the vehicle to real-time coordinate with each other to drive itself on a road with conflict free. Recently, the model predictive control has also been widely used to solve the problem of process control, task scheduling, cruise control, and multi-agent transportation networks [13], [14], [15]. These works provide solid results for related mobility scheduling and control problems. However, none of these works incorporates both the current and historical mobility patterns into their vehicle control design, leveraging the iteration scheme to solve the conflicts between vehicles.

In this paper, we focus on vehicle energy consumption model and design a cost-based iterative framework that can minimize energy consumption while eliminating conflicts between vehicles. Moreover, our framework supports the parallelization to improve the real-time performance of the autonomous transportation control systems.

3 Macroscopic Energy Consumption Model

Macroscopic road-based energy consumption model is the effective way to estimate the energy consumption of each vehicle in autonomous control systems [4]. We focus on electric vehicles and develop an energy-efficient design scheme. In terms of electric vehicles, energy consumption model is required to capture the regenerative braking and electric drive efficiency. In general, the vehicle longitudinal dynamical model may be written as [5].

$$m\dot{v}(t) = F_w - F_a - F_f - F_s, \tag{1}$$

where m is the vehicle mass, $\dot{v}(t)$ is the vehicle acceleration, F_w is the force at the wheels, F_a denotes the aerodynamic force, F_f represents the rolling resistance force, and F_s is the gravity force. Thus, we have the vehicle model

$$\begin{cases} \dot{x}(t) = v(t) \\ m\dot{v}(t) = F_w - \frac{1}{2}\rho_a A_f c_d v(t)^2 - mgc_r - mgsin(\alpha(x)), \end{cases} \tag{2}$$

where ρ_a is the external air density, A_f is the vehicle frontal area, c_d is the aerodynamic drag coefficient, c_r is the rolling resistance coefficient, $\alpha(x)$ is the road slope as a function of the position, and g is the gravity.

Note that the sum of aerodynamic and rolling frictions, named road load force, is generally approximated as a second order polynomial in the speed v. Thus, we have

$$F_a + F_f = a_2 v(t)^2 + a_1 v(t) + a_0, \tag{3}$$

where a_0 , a_1 and a_2 are the constant parameters identified for given a considered vehicle. Thus, the force at the wheels can be also expressed as followed.

$$F_w = m\dot{v}(t) + a_2 v(t)^2 + a_1 v(t) + a_0 + mgsin(\alpha(x)). \tag{4}$$

For each node $n \in V$ of the graph, it is possible that we can obtain road segment length l_n and average traffic speed $\overline{v_n}$ of vehicle on this road. Because we can obtain the time of the day, and the road grade $\alpha_n(x)$, both of which varies within the used road segment depending on the position.

Specifically, since the time-variant speed or acceleration profile is not available, the energy consumption model can not be directly used to assign the weights to each node of the graph. Thus, we leverage average traffic speed $\overline{v}$ to replace the time-variant speed $v(t)$. All the vehicles on node n are supposed to travel at average traffic speed $\overline{v_n}$. Note that while it exists difference, it can efficiently reflect real driving conditions. The previous work [6] also give a validation analysis to verify the accuracy. Thus, the force expression in (4) is modified for each node n as follows:

$$\overline{F}_{w,n} = a_2 \overline{v}_n^2 + a_1 \overline{v}_n + a_0 + mgsin(\alpha_n(x)), \tag{5}$$

with no acceleration term. The torque requested from the electric motor to meet the force demand at the wheels is given as:

$$\overline{T}_{m,n} = \begin{cases} \frac{\overline{F}_{m,n} r}{\rho_t \eta_t}, & \text{if } \overline{F}_{w,n} \geq 0 \\ \frac{\overline{F}_{m,n} r \eta_t}{\rho_t}, & \text{if } \overline{F}_{w,n} < 0 \end{cases}, \tag{6}$$

where r is the wheel radius, ρ_t and η_t are the transmission ratio and efficiency, respectively. The electric motor rotational regime is also constant over time if constant speed is assumed:

$$\overline{w}_n = \frac{\overline{v}_n \rho_t}{r}. \tag{7}$$

Thus, the mechanical power available at the electric motor is written as followed.

$$\overline{P}_{m,n} = \begin{cases} T_{m,max} \cdot \overline{w}_n, & \text{if } \overline{T}_{m,n} \geq T_{m,max} \\ \overline{T}_{m,n} \cdot \overline{w}_n, & \text{if } T_{m,min} < \overline{T}_{m,n} < T_{m,max} \\ T_{m,min} \cdot \overline{w}_n, & \text{if } \overline{T}_{m,n} \leq T_{m,min} \end{cases}. \tag{8}$$

In the following, we assumed that the saturation torque is independent from the motor regime. Finally, the power demand at the battery of the electric vehicle, considering the electric drive efficiency η_b constant, can be written as:

$$\overline{P}_{b,n} = \begin{cases} \frac{\overline{P}_{m,n}}{\eta_b}, & \text{if } \overline{P}_{m,n} \geq 0 \\ \overline{P}_{m,n} \cdot \eta_b, & \text{if } \overline{P}_{m,n} < 0 \end{cases}, \tag{9}$$

and ultimately, we have the battery energy consumption over the generic travel time T_n.

$$\overline{E}_{b,n} = \int_0^{T_n} \overline{P}_{b,n} dt = \overline{P}_{b,n} T_n, \tag{10}$$

where $T_n = l_n / \overline{v}_n$ is the travel time on segment node n when traveling at the average traffic speed $\overline{v}_n$.

4 Our EEPC Framework

Our proposed EEPC framework is performed on the cloud server of autonomous control system. It is able to real-time simulate the vehicle routing process to fast generate a feasible control solution to guide the routing of CVs. As a result, each vehicle is based on its own available route and can drive from a selected origin to a destination in the current traffic network. Note that the route of each vehicle is the energy-efficient and all the participated CVs have no conflicts guaranteeing the traffic safety.

Specially, there are many impact factors resulting in the traffic safety problem, and the resource conflict is probably the most important factor in modern transportation control systems. The control of connected cars typically occupies the same routing resources of ground traffic network forming resource conflict. Such a conflict would cause collisions between connected cars, further resulting in a safety problem. Thus, the resource conflict is unacceptable especially in the macro-scale. Therefore, we devote to eliminate the resource conflict and propose the EEPC framework. In addition, our framework supports two different grained parallelism to improve the real-time performance.

Fig. 1 shows the overview of our EEPC framework to CVs of autonomous control system. Given a network graph and the participated CVs, EEPC generates a feasible solution so that each vehicle has a legal and energy-efficient route on the graph. EEPC consists of an inner layer and an outer layer which invokes the inner layer to implement a feasible solution for CVs. In inner layer, we perform the local control of CVs and we first control one vehicle routing and then update the local cost and then control the next vehicle routing until completing the routing of all the participated CVs. Note that each vehicle routing adopts the Dijkstra's search algorithm to find the shortest path measured by the costs of energy and conflict. If this solution is feasible, we will use it to guide the routing of CVs and otherwise, we perform the global control of CVs in outer layer. In the global control step, we first rip-up the previous routes of all CVs and then update the global cost so that we perform the next local control in inner layer. With global and local control steps, EEPC iteratively refines the route of each vehicle until generating a feasible control solution. In their respective iterations, EEPC performs the rip-up in global control step and performs the route or re-route in local control step for all the participated CVs in order to completely eliminate the resource conflicts between CVs. The implementation details will be demonstrated in Section 5.

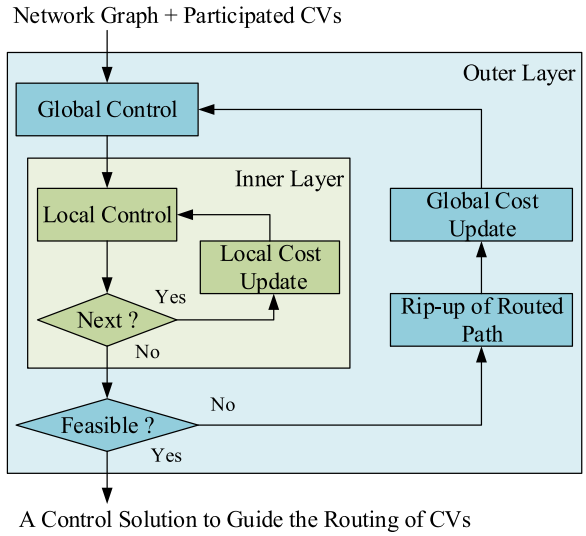

Fig. 1. The overview of our EEPC framework. In inner layer, we perform the local control to find a energy-efficient route to each vehicle. In outer layer, we perform the global control to eliminate the conflicts between CVs. Both of them have been combined to generate a feasible solution to control the routing of CVs. In addition, the local control can be parallelized in fine-grained scheme while the global control can be parallelized in coarse-grained scheme, both of which can improve the real-time performance of our EEPC framework.

In addition, our EEPC framework has a unique feature to allow parallel processing of control of CVs and has the potential to obtain a better real-time performance. EEPC supports fine- and coarse-grained parallelism. The fine grain is used to perform the parallelization of local control while the coarse grain performs the parallelization of global control. In fine-grained parallel scheme, we parallel control the vehicle routing one-by-one and specifically, we parallelize the low-level shortest-path exploration of ground traffic network graph. Multi-threading techniques are used as the fine-grained parallel programming paradigm running on the multi-core shared-memory platform. In coarse-grained parallel scheme, all the CVs are partitioned into several subsets and the routing of each subset is controlled by its own processor core. Note that the number of subsets is equal to the number of processor cores. MPI techniques are used as the coarse-grained parallel programming paradigm running on the multi-core distributed-memory platform. The implementation details will be demonstrated in Section 6.

In the EEPC framework, we assume that the real-time capacity of each resource node of traffic network graph is set to one, conflicting behaviours could occur at such a resource where more than one vehicle tries to route through this resource. If such a conflicting behaviour does occur, the idea of both the rip-up of global control and the re-route of local control will be employed to progressively release these conflicting resources occupied by the different vehicles in simulation process. In practice, the occupation of same resource should be less than or equal to the capacity to guarantee the conflict removal.

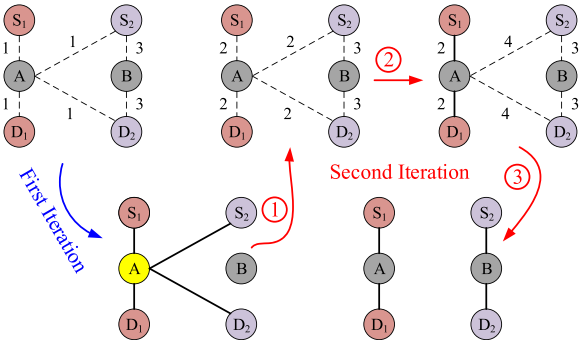

Fig. 2. An example of how to control the routing of CVs using cost function. There are two iterations to implement the feasible solution. In the first iteration, the initial solution is infeasible due to the conflict resource between two vehicles. In the second iteration, the subsequent solution is feasible due to the updates of global and local costs. The value of global cost increases 1 each iteration while the value of local cost is increased, depending on how many vehicles occupy the resource, in current iteration.

5 Energy-Efficient Control

In the proposed EEPC framework, the control of CVs is based on cost function to minimize the energy consumption of each vehicle and to eliminate the conflicts between different vehicles. We first demonstrate the cost function design motivated by an example and then we apply it into the EEPC framework. At last, we discuss the tradeoffs between energy consumption and conflict elimination when controlling the routing of CVs in our EEPC framework.

5.1 Cost Function Design

We start with an example to describe the control problem motivating the cost function design. Consider the example depicted in Fig. 2, we need to control two vehicles routing from their sources S_1 and S_2 to their destinations D_1 and D_1, respectively. Assuming that we adopt the basic shortest path algorithm, there is a conflict resource node A and this solution cannot satisfy the design requirements of control systems. To solve this problem, we consider designing the cost function to guide the control of CVs. The idea of imposing cost function is motivated by the other fields [16], [17], [18].

The proposed cost function consists of base cost, local cost, and global cost. These three terms can be iteratively updated to spread the conflicting resources between different vehicles for a feasible solution. In the ith iteration, the cost function for a resource v is given as follows:

$$c_v = (b_v + g_v^i) \times l_v, \tag{11}$$

where b_v is the base cost of using a road resource v, which can be used to reflect the basic energy consumption of the road resource. A reasonable choice for b_v is the intrinsic energy consumption e_v of the resource v, since minimizing the energy consumption of a vehicle is equal to minimize the road resources of a vehicle in nature. Thus in the remainder of our work, we set $b_v = e_v$. The local cost term l_v denotes the number of vehicles occupying the resource v in the local control. The global cost term g_v denotes the conflicts at a road resource v in the global control. Here we enable the global cost g_v to be increased in a fixed amount[2] iteration when a vehicle is re-routed through an already occupied resource node. In other words, this global cost g_v grows monotonically with each iteration in which the road resource is occupied.

By imposing three cost terms to each resource, and updating these costs within each iteration, the alternative routes can be explored to release conflicts between CVs. Essentially, in local control, the l_v enables the conflict vehicles to perform the coordination with each other for using an occupied resource. In global control, the g_v enable the non-conflict vehicles to release a legal resource for conflict vehicles. Our basic idea is to permit the conflicts initially and then iteratively update the cost until there are no conflicting resources.

Given a weight graph and CVs, we leverage the above cost function to obtain a feasible solution, further guiding the routing of CVs. We continue to select the example shown in Fig. 2 to demonstrate how to eliminate the conflicts between CVs. In this example, we need to perform two iterations so that each vehicle has its own route. In the first iteration, we use shortest-path fashion to control the routing of CVs on the graph within basic cost, thereby resulting in a conflict resource A between two CVs. In the second iteration, the cost function is to update the costs of the graph and we have three steps to guide the control of CVs. In the step ①, we rip-up the routes of previous iteration and perform the update of global cost g_A. The value of global cost g_A is increased from 0 to 1 while the total cost c_A of a conflict resource A is increased from 1 to 2. Note that the value of global cost adds 1 in each iteration. In the step ②, we perform the local control and route the first vehicle and then update the local cost l_A. The local cost is gradually increased, depending heavily on how many vehicles occupy the resource A, in current iteration. Thus for second vehicle, the total cost to use the resource A is increased from 2 to 4 which is large than the basic total cost 3. Thus in the step ③, when we continue to use shortest-path fashion to route the second vehicle, the alternated path will be selected for a feasible solution.

5.2 Routing Order Arrangement

The control of CVs involves assigning resources so that the destination is reachable from the source for each vehicle. When we control the routing of all participated CVs, the routing order becomes a very critical problem since some resources needed by a vehicle may be occupied by other vehicles, further resulting in the resource conflicts between different vehicles. To release the conflicts, the imposed cost function forces some vehicles to make a detour to select the alternative routes on the network graph. The conflicted vehicles adopt the alternative routes generating the longer paths, further impacting the total energy consumption.

To mitigate this problem, we focus on routing order and in our EEPC framework, we have the basic routing order arrangement as follows. All participated CVs are sorted first according to their distance from source to destination and we control the routing of CVs in such an order of decreasing distance. The longest distance vehicles are routed first because they tend to cross the whole region and moreover, they are much easier to route when there is no existing conflict. Short distance vehicles tend to be localized, so routing

2. Here, $g_v = g_v + 1$ if the resource v has a conflict in each iteration.

them later in the control process is not too difficult even in the presence of conflict.

In addition, when generating the resource conflicts between different vehicles, we perform the rip-up and re-route fashion for all participated CVs rather than the conflicted vehicles. Note that this rip-up and re-route fashion is executed at each iteration in the EEPC framework. Typically our EEPC framework can find the feasible solution with the limited number of iterations. To avoid the infinite iteration of EEPC framework, the total number of iterations is set to 50 in our simulation evaluation. Actually, our EEPC framework always uses the small number of iterations to find a feasible solution. It is effective to leverage cost-based rip-up and re-routed engine to eliminate the conflicts between vehicles.

5.3 Conflict-Aware Routing Control

Benefiting from conflict-based cost function above, our EEPC framework can generate the feasible solution so that each vehicle has its own route. In this framework, CVs are allowed to occupy the same resources and then iteratively performs the cost-based coordinate with other vehicles to determine which vehicle needs the resource most. The cost update is performed at each iteration to employ pressure continuously to control the routing of CVs. We gradually adjust the costs of resources to obtain a good resource assignment to CVs.

Algorithm 1. The Algorithm to Control the Routing of CVs

```
1:  Given(Graph G = ⟨V, E⟩, CVs {i})
2:
3:  Global-Control-Routing(G, {i})
4:  while incomplete or conflict exists do
5:    Local-Control-Routing({i})
6:    update global costs {g_v} of the resources in V
7:  end while
8:  end Global-Control-Routing
9:
10: Local-Control-Routing(vehicles {i})
11: for each uncontrolled or conflict vehicle i do
12:   rip-up T_i if exists
13:   T_i ← {s_i}
14:   T_i ← Find-Shortest-Path(T_i, d_i)
15:   update local costs {l_v} of the resources in T_i
16: end for
17: end Local-Control-Routing
18:
19: Find-Shortest-Path(T_i, d_i)
20: for each resource node v in T_i do
21:   enqueue v onto Q with key 0
22: end for
23: while a sink of d_i has not been found do
24:   dequeue node, p with lowest key from Q
25:   if p was not previously dequeued then
26:     for each neighbor v of p do
27:       enqueue v on Q with cost of v + key of p
28:     end for
29:   end if
30: end while
31: backtrace from sink to a node of T_i that is reached
32: return this path T_i
33: end Find-Shortest-Path
```

The algorithm to control the routing of CVs is shown in Algorithm 1. This algorithm can be divided into three nested iterations. In outermost iterations, we perform the global control of CVs. All of the participated CVs are encouraged to perform the cost-based coordination with each other to decide who will explore alternate route around the conflict resources, until all the conflicts are resolved to obtain a complete legal control solution. In middle iterations, we perform the local control of CVs. The vehicle route T_i from the previous outermost iterations is ripped-up and can be initialized to the vehicle source. and at the meanwhile, it will invoke the shortest-path algorithm, which computes a path from the source to the sink in the network resource graph. In innermost iterations, it employs the single source shortest path algorithm, which is implemented by Dijkstra's algorithm. After a sink is found, all nodes along a backtraced path from the sink to source are added to T_i, and at last, if the solution is feasible, the algorithm is complete.

In addition, the resources chosen by this algorithm is a challenge problem. Note that choosing the optimal or even near-optimal resources is not essential in this algorithm. The key of algorithm is successfully feasible in adjusting costs to eliminate the conflicts between CVs, further guiding the routing of CVs on the road with a safety guarantee.

5.4 Analysis of the Energy Consumption and Conflict Removal Tradeoffs

The idea of EEPC framework trades energy consumption for conflict removal. The emphasis of our algorithm is to adjust the costs of resources in a gradual and semi-equilibrium way to ensure a reasonable distribution of resources for all the participating CVs. Thus we analyze the tradeoffs of energy consumption and conflict removal when we control the routing of CVs in the EEPC framework. Thus we re-define the cost function of using resource v when we control the vehicle i routing from source s_i to destination d_i. The new cost function is given as follows:

$$C_v = \sigma_i e_v + (1 - \sigma_i) c_v, \tag{12}$$

where c_v is defined in (11) and σ_i is the balance ratio.

$$\sigma_i = E_i / E_{max}, \tag{13}$$

where E_i is the longest route from s_i to d_i, and E_{max} is the maximum over all routes, and here, we call *critical path energy consumption*. Thus, $0 < \sigma_i \leq 1$. The first term in Equation (12) is aware of the energy consumption. The second term is aware of conflict. Note that the long routing path will take more energy consumption when we control the vehicle routing. Fig. 3 shows that the vehicle may select the alternative path to route due to that there is a conflict existed in the shortest routing path. Actually in EEPC methodologies, we eliminate the resource conflict by imposing penalty cost to force the vehicle to explore the alternative path instead of selecting the original shortest path. Deprecating the shortest path is an efficient approach to eliminate the conflict of resources.

Our Equations (12) and (13) build an appropriate mix of *minimum-energy* and *minimum-cost* route, further forming a completing tradeoffs between energy consumption and conflict removal. If a source-sink pair relies on the critical route,

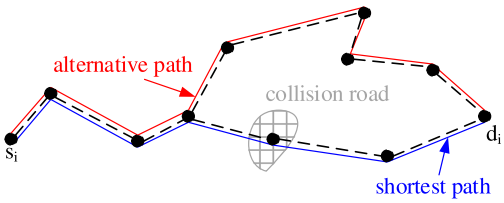

Fig. 3. The conflict of resources shows that there is a collision road. At that time, the route of vehicle selects the red alternative path rather than the blue shortest path.

then $\sigma_i = 1$ and the cost function for using resource v is simply energy term. Thus, a minimum-energy control scheme will be used and the conflict term will be ignored. If a source-sink pair belongs on a routing path whose energy is much smaller than the energy of critical route, its σ_i will be small and the conflict term will dominate, resulting in a solution which avoids conflict at the expense of extra energy.

To adapt to the energy consumption, Algorithm 1 can be changed as follows. First, the σ_i is initialized to 1 and the algorithm searches the minimum-energy route for every vehicle during the first iteration. In addition, the σ_i is re-calculated in each subsequent iteration. Second, the destination is reached in the decreasing order of σ_i. Third, the priority queue is initialized to T_i at cost $\sigma_i e_i$. The effect of this initialization is that nodes that belong to the partial route will have only the energy consumption component. These modifications will be referred in the algorithm to control the routing of CVs in the EEPC framework.

The algorithm terminates when no conflict resource exists. By re-calculating the σ_i, we have kept a tight domination on the critical route. Over the process of iterations, the critical route increases only to the situation that requires to resolve the conflict. The algorithm first reduces the energy consumption and then attempts to resolve the conflicts by re-executing the control of vehicle routing.

5.5 Energy Consumption Analysis

It is effective to trade the energy consumption for conflict removal and the problem is that how much is the bounds of energy consumption when we control the routing of CVs in the EEPC framework. Thus we need to analyze the bounds of energy consumption when employing the Algorithm 1 into the EEPC framework.

Here we consider that if global cost g_v is constrained by the base cost e_v, the Algorithm 1 will generate a worst-case energy-consumption route, which is the same to the minimum-energy critical route. It means that Algorithm 1 selects the fastest implementation in the network graph. In practice, the global cost g_v is allowed to increase in a gradual way until finding a complete solution. While the global cost g_v maybe exceed the base cost e_v in very difficult network graphs, Algorithm 1 still comes very close to this constraint in practice. Thus, when Algorithm 1 is adopted in the EEPC framework, we have the following theorem for the bounds of energy consumption.

Theorem 1. *If $g_v \leq e_v$ for all nodes of graph, the energy consumption of any vehicle route is constrained by E_{max}, the energy consumption of the longest minimum-energy route in the graph.*

Proof. When Algorithm 1 terminates successfully, the local cost l_v term in equation (12) is 1 and thus, $c_v = e_v + g_v$. Let R represents the most critical used route and S denotes the shortest-route energy consumption for R. The cost of S is given by:

$$C_S = \sum_{v \in S} C_v \tag{14}$$

$$= \sum_{v \in S} (\sigma_i e_v + (1 - \sigma_i)(e_v + g_v)) \tag{15}$$

$$= \sum_{v \in S} e_v + (1 - \sigma_i) \sum_{v \in S} g_v. \tag{16}$$

According to our assumption $g_v \leq e_v$,

$$C_S \leq \sum_{v \in S} e_v + (1 - \sigma_i) \sum_{v \in S} e_v \tag{17}$$

$$= E_i + (1 - \sigma_i) E_i \tag{18}$$

$$= (2 - \sigma_i) E_i \tag{19}$$

$$= (2 - \sigma_i)\sigma_i E_{max}. \tag{20}$$

Since $0 \leq \sigma_i \leq 1$, we have $0 \leq (2 - \sigma_i)\sigma_i \leq 1$ and

$$C_S \leq E_{max}. \tag{21}$$

The cost of R must be less than the cost of S, thus, the energy consumption of R must be less than the cost of S, which is less than E_{max}. □

6 Exploration of Parallel Control

The *energy-efficient conflict-free control* of CVs discussed in the previous section allows EEPC to progressively generate a feasible control solution such that each vehicle has its own energy-efficient route while eliminating the conflicts between CVs. In this section, we study the parallel control problem and our goal is that EEPC has a better real-time performance when controlling the routing of CVs, especially for large-scale CVs. In addition, the parallel control of CVs should be scalable support for multiple processor cores and the energy consumption should be small with the increasing number of processor cores.

It is non-trivial to perform the parallel control of CVs in the EEPC framework as it exhibits a dependent problem resulting in the inherent sequential processing order. To develop a good parallel control approach running on the multi-core processor system, we discuss several important problems as follows.

(1) How to parallelize the inherently sequential control of CVs and what is the granularity when performing the parallel control of CVs?
(2) How to synchronize the intermediate information between various processor cores to guarantee that the parallel control of CVs has a feasible solution as well?
(3) How to scale the available speedup and reduce the energy consumption when adopting the increasing number of processor cores?

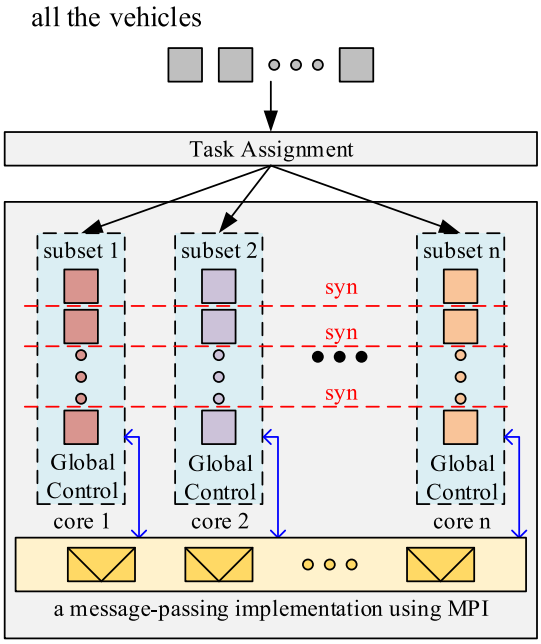

Fig. 4. The coarse-grained scheme is used to parallel control the routing of CVs. We assign the CVs to different processor cores and perform the parallel control of CVs. We use the MPI techniques to communicate with each other.

The EEPC framework performs the sequential control of CVs, consisting of global control and local control. The global control focuses on multi-vehicle routing while the local control focuses on single-vehicle routing. In global control, multiple CVs perform the cost-based coordination to eliminate the resource conflicts. In local control, each vehicle is based on the shortest-path algorithm to explore the alternative route. The parallelization of global control is at the task level and it means that we control the routing of multiple CVs in a concurrent way. The parallelization of local control is at the resource level and it means that we control the routing of single vehicle in a concurrent way. The former forms coarse-grained parallel scheme running on the multi-core distributed-memory platform using MPI techniques. The latter forms fine-grained scheme running on the multi-core shared-memory platform with multi-threading techniques.

A modern autonomous control system has more than thousands of CVs, so the sequential control algorithm can be parallelized if all the CVs's routing can be parallel controlled, thus, the available speedup will be limited only by the number of vehicles. The challenge is that the route used by each vehicle depends heavily on the knowledge of the other vehicles's routes since the traffic resources cannot be real-time occupied in vehicle routing. If the routes of two vehicles occupy the same resources in real-time in parallel control process, they will result in real-time conflicts and cannot generate a feasible solution. Thus, each processor core must have an up-to-date view of the intermediate results when performing the parallel control of CVs. Thus, each processor core requires to perform the synchronization to communicate the intermediate results, further avoiding the resource real-time sharing between different vehicles.

According to the discussions and analyses above, We leverage the coarse- and fine-grained schemes to explore the parallel control of CVs to further improve the real-time performance of the EEPC framework.

6.1 Coarse-Grained Parallel Control

Coarse-grained parallel control performs on the multi-core distributed-memory platform. Message passing interface (MPI) is used to synchronize the intermediate routes between processor cores when we parallel control the routing of CVs. Thus in the coarse-grained scheme, all the vehicles need to be assigned to several subsets and the number of subsets should be equal to the number of processor cores, further guaranteeing that each subset is executed on its own processor core. Thus, *assignment* becomes a critical step in the coarse-grained scheme. In order to design a good assignment approach, we define *the control of single-vehicle routing* as a *task*.

Basic Task Assignment and Parallelization. Fig. 4 gives an example of the coarse-grained scheme to parallelize the global control of the EEPC framework. This approach consists of assignment and parallelization. In assignment we distribute the tasks into different subsets and each subset has the roughly equal number of tasks for having a good load balance between processor cores. In parallelization we concurrent control the first task batch and synchronize their intermediate results and then concurrent control the next task batch until completing all the tasks. The intermediate results between processor cores can be communicated using MPI messages, further enabling each processor core to synchronize its own view of the overall control state. By using MPI messages, we not only avoid the resource conflicts between vehicles but also maintain most of the data structure used in sequential control approach.

Algorithm 2. Coarse-Grained Parallel Approach

```
1:  while task incomplete or conflict exists do
2:    assign the tasks to N subsets evenly
3:    for k such that k ≠ local do
4:      task[k] = first_task(subset[k])
5:    end for
6:    for local_task ∈ subset[local] do
7:      process the local task local_task
8:      synchronize the picture of local_task
9:      for k such that k ≠ local do
10:       the picture is obtained from core k for task[k].
11:       task[k] = next_task(subset[k])
12:     end for
13:   end for
14: end while
```

Algorithm 2 gives the pseudocode of our coarse-grained parallel approach and specifically, it only describes the key parallel portion of the implementation. In addition, this algorithm is from the perspective of one of N processor cores to perform its own subset of tasks. The variable k is the index of processor core and the variable $local$ is the index of the master processor core when performing the parallel approach. $subset[k]$ is a subset of tasks assigned to the processor core k for processing. The array $task[k]$ is used to record which task is currently being processed by each processor core. The first_task() function extracts the first task from the subset of processor core k to process. The next_task() function gives the next task to be processed by processor core k.

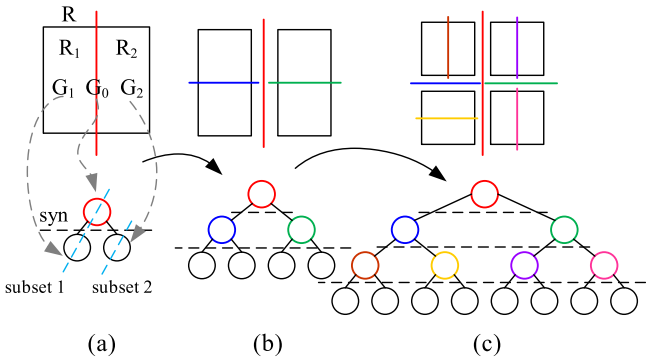

Fig. 5. The recursive partitioning is performed three times which forms a three-level perfect binary tree, further scaling to eight MPI processes for parallelization on multi-core processor systems. (a) The master MPI process processes the tasks of group G_0 of *subset 1* and then, combining with the slave MPI process, to parallel process the tasks of group G_1 of *subset 1* and the tasks of group G_2 of *subset 2*. (b) and (c) The initial binary tree can be recursively partitioned to support the scalable parallel processing.

The while loop is an iteration of parallel control approach and in task assignment, we distribute the tasks into N subsets in a load balance way, where N is the total number of processor cores. Each processor core is aware of the subset of tasks processing in every other processor core. In terms of each processor core k, besides the *local* processor core, we employ the first task of $subset[k]$ to initialize the variable $task[k]$. In the following outer loop, we process a task each time and this task is in the subset of tasks of local processor core. We first process the local task $local\_task$ and then synchronize the up-to-date picture to all other processor cores containing the path information and conflict state of $local\_task$. In the inner loop, we perform once for each processor core, k, beside from *local*. This loop is used to receive any update message pictures from other processor cores that have already been sent. Once we receive an update message, the next task will be extracted from $subset[k]$ to update the $task[k]$ and to process this task.

In this basic task assignment and parallelization, each processor core processes a task and synchronizes its results and then processes the next task and synchronizes its results until completing all tasks of its own subset. It is obvious that this approach has very frequent synchronization operations and its long communication time between processor cores will have an impacts on the available parallelism, further severely affecting the real-time performance of the EEPC framework. To reduce the expensive synchronization overheads, we leverage the *region partitioning* to guide the task assignment for parallelization.

Region-based Task Assignment and Parallelization. The basic idea is to partition the region R into two subregions (R_1, R_2) which forms three groups (G_1, G_0, G_2). The first group G_0 consists of tasks where its start and destination coordinates distribute in two subregions. The remaining two groups (G_1, G_2) consist of tasks where their start and destination coordinates are located in their own respective subregions. As the tasks of group G_0 are distributed in two subregions, they are processed in sequential. Correspondingly, as the tasks of group G_1 and the tasks of group G_2 are located in their own respective subregions, they are processed in parallel. The load balance can be maintained by ensuring that group G_1 and group G_2 have the roughly equal number of tasks. Note that load balance enables region partitioning to obtain an even distribution of tasks for better parallelism.[3] In addition, these subregions can be further partitioned in a recursive way and this recursive partitioning process forms a perfect binary tree which provides the opportunity to explore the parallelization.

Fig. 5 gives an example of region-based partitioning to guide the task assignment for parallelization. Fig. 5a shows the first-level partitioning for parallelization. The generated three groups form a binary tree where the parent node represents the group G_0 and the left child and right child represent the group G_1 and group G_2, respectively. The combination of group G_0 and group G_1 is regarded as a subset *subset 1* while the group G_2 is regarded as the other subset *subset 2*, both of which are parallel processed on two processor cores using MPI techniques. The master MPI process first processes the group G_0 of *subset 1* and synchronizes these results, and then, combining with the slave MPI process, to parallel process the group G_1 of *subset 1* and the group G_2 of *subset 2*. In this way, we perform the group-level synchronization only, thereby reducing the synchronization overheads significantly. Fig. 5b shows the second-level partitioning in a recursive way. The group G_1 is partitioned to form three groups (G_{11}, G_{10}, G_{12}) and the group G_2 are partitioned to form three groups (G_{21}, G_{20}, G_{22}), further creating a two-level perfect binary tree for parallelization on four processor cores. As shown in Fig. 5c, we perform the third-level recursive partitioning to form a three-level perfect binary tree for parallelization on eight processor cores. Note that the number of leaf nodes of perfect binary tree is equal to the number of processor cores.

In this way, the generated perfect binary tree is used to guide the task assignment for parallelization using MPI techniques on multi-core processor systems. Comparing with the previous basic assignment approach, this approach performs the group-level synchronization only and reduces the communication overheads significantly, further having the opportunity to obtain the better real-time performance for the EEPC framework.

6.2 Fine-Grained Parallel Control

Fine-grained parallelization focuses on the local control of CVs running on multi-core shared-memory platform. In the fine-grained scheme, we parallelize the shortest-path routing of each vehicle. Specifically, we parallelize aspects of the local-level path exploration on the ground traffic network graph. Threads are used to implement the fine-grained parallel programming paradigm. Actually, this is the parallelization of shortest-path algorithm in our EEPC framework.

Fig. 6 gives an example of the fine-grained approach to parallelize the single-vehicle routing of local control. Given the available N processor cores, we build one master thread and $N-1$ slave threads. In addition, we use N priority queues so that each thread only updates nodes in its own priority queue. In terms of thread-level parallelism, the master thread extracts the smallest cost node from the

3. If the vehicles are not distributed evenly, the available parallelism is very limited.

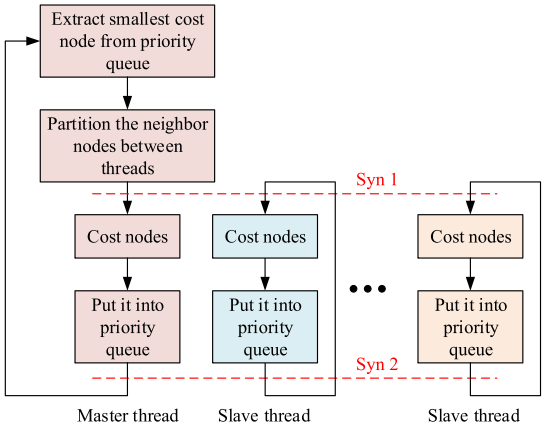

Fig. 6. The fine-grained scheme is used to parallelize the local control of the routing of CVs. We assign the resources in a round-robin way to different processor cores and perform the parallel control of single-vehicle routing. We use the multi-threading techniques to communicate with each other.

priority queue and leverages the round-robin fashion to partition its neighbor nodes between all threads for parallelization. Once the master thread updates the costs and puts the nodes into its priority queue, it communicates in the second barrier with all other slave threads before extracting the smallest cost node and repeating the loop. Two barriers are used to synchronize the intermediate state. The first synchronization indicates that the slave threads is ready while the second is used to indicate that the slave threads have completed their respective works.

7 Simulation and Evaluation

In this section, we conduct the extensive simulations to evaluate the impacts of the EEPC framework on the energy consumption and real-time performance when controlling the routing of CVs.

7.1 Simulation Setup

The proposed EEPC framework is implemented in the C++ programming language performing on the Intel Xeon 8-core processor system. The simulation study and experiments are conducted on ten different test cases, all of which are extracted from the ground traffic network graph of the California state (i.e., about 1,965,206 nodes and 2,766,607 links) [19]. Because there exists no available free dataset about the road segment length, road grade, and current traffic conditions as previously discussed in Section 3, we design a random algorithm to generate these data, including the corresponding average traffic speed of each vehicle, the number of vehicles with different origin-destination pairs in the considered network graph. By this way, we can calculate the energy-consumption weight of each node in traffic resource graph while it is synthetic.

Table 1 shows the extracted ground traffic network graph benchmarks. Specifically, the size of network graph is increased in a gradual manner, because the scalability of parallel control approach is very crucial to large-scale connected vehicles. In our simulation study, we mainly focus on the results of runtime and energy consumption of control approach of the EEPC framework. In general, we leverage the usage of road segment resources to evaluate the energy consumption.

TABLE 1
Road Network Information

Bench.	#Nodes	#Vehicles	Bench.	#Nodes	#Vehicles
case 1	27120	788	case 6	283792	5224
case 2	76386	1946	case 7	305082	6606
case 3	43872	2380	case 8	283338	7154
case 4	104176	3710	case 9	311112	7474
case 5	110250	3953	case 10	492570	8078

7.2 Sequential Control Study

Our EEPC is a cost-based iterative framework and in the EEPC framework, our sequential control approach attempts to balance the competing goals in terms of minimizing the energy consumption of CVs and eliminating the conflicting resources between CVs. In addition, the runtime of EEPC framework has the direct impact on the real-time performance of autonomous control systems. Thus in the first set of experiments, we focus on the conflicting resource, energy consumption, and runtime of sequential approach. Thus we study the iterative process of EEPC framework and analyze three metrics above at every iteration.

Our EEPC framework consists of global control and local control. In local control, the iteration is used to control the routing of each vehicle. In global control, the iteration is used to find a feasible control solution so that each vehicle has its own route. In this experiment, the number of iterations of global control is used as the total number of iterations of EEPC framework. Table 2 shows the total number of iterations to find a feasible solution for each case in the EEPC framework. In addition, we also give the total number of conflicting resource nodes of each iteration. In the first iteration, the network graph is weighted only by the basic cost of energy consumption. When performing the original shortest-path algorithm, there are large number of conflicting nodes. In the following iterations, the weight of graph is continually updated by global cost and local cost to find a feasible route to each vehicle. Note that the small number of iterations is sufficient to implement the feasible solution in the EEPC framework. On average, our EEPC framework takes about ten iterations to completely eliminate the resource conflicts between CVs, especially for large-scale CVs. We believe that the cost-based iterative scheme is effective to completely eliminate the resource conflicts, further guaranteeing the traffic safety.

Energy consumption is a second critical metric and to have a better comparison in this experiment, we focus on the total energy consumption of each case and we leverage the *energy consumption ratio* to demonstrate the effectiveness of our EEPC framework. Note that the ratio is between the sum of energy consumption of all participated CVs and the available energy provided by its own overall network graphs. Table 3 shows the energy consumption ratio to implement a practical solution for each case in our EEPC framework. We also give the ratio of each iteration. In the first iteration, we use the shortest-path algorithm to control the route of each vehicle on the original graph. After several iterations, the original route is updated to the new route in

TABLE 2
Analyses of the Conflicting Resource Nodes of Each Iteration Across Ten Different Benchmarks

Name	The Number of conflicting Resource Nodes at Every Iteration.													Total
Iteration	1	2	3	4	5	6	7	8	9	10	11	12	13	Num. of Ite.
case 1	743	253	97	24	9	7	5	2	0	—	—	—	—	9
case 2	1,127	674	226	22	9	3	0	—	—	—	—	—	—	7
case 3	865	376	152	95	41	16	8	3	1	0	—	—	—	10
case 4	1,754	742	313	154	85	27	9	7	4	2	1	0	—	12
case 5	2,572	855	428	217	96	22	11	2	0	—	—	—	—	9
case 6	1,836	438	271	112	65	21	13	8	5	2	0	—	—	11
case 7	8,375	5,421	3,629	1,753	223	32	7	3	0	—	—	—	—	9
case 8	7,581	3,612	1,095	833	506	317	25	9	6	0	—	—	—	10
case 9	9,765	2,241	853	242	73	19	2	0	—	—	—	—	—	8
case 10	13,563	5,423	2,117	564	233	98	42	21	10	7	3	1	0	13

TABLE 3
Analyses of the Energy Consumption of Each Iteration Across Ten Different Benchmarks

Name	The Ratio of Energy Consumption of Each Iteration.													Final Ene.
Iteration	1	2	3	4	5	6	7	8	9	10	11	12	13	Con. Rat.
case 1	0.21	0.23	0.24	0.24	0.24	0.24	0.25	0.26	0.26	—	—	—	—	0.26
case 2	0.26	0.28	0.28	0.29	0.32	0.32	0.33	—	—	—	—	—	—	0.33
case 3	0.11	0.14	0.14	0.14	0.16	0.16	0.15	0.17	0.17	0.17	—	—	—	0.17
case 4	0.30	0.32	0.32	0.32	0.33	0.33	0.33	0.34	0.34	0.34	0.35	0.35	—	0.35
case 5	0.28	0.31	0.32	0.33	0.33	0.34	0.35	0.35	0.36	—	—	—	—	0.36
case 6	0.06	0.06	0.06	0.06	0.07	0.07	0.07	0.08	0.08	0.09	0.10	—	—	0.10
case 7	0.32	0.34	0.34	0.35	0.35	0.37	0.38	0.38	0.39	—	—	—	—	0.39
case 8	0.33	0.34	0.34	0.35	0.35	0.36	0.37	0.38	0.38	0.38	—	—	—	0.38
case 9	0.29	0.30	0.31	0.32	0.32	0.33	0.34	0.35	—	—	—	—	—	0.35
case 10	0.24	0.24	0.24	0.24	0.25	0.25	0.25	0.26	0.26	0.26	0.27	0.27	0.27	0.27

TABLE 4
Analyses of the Runtime of Each Iteration Across Ten Different Benchmarks *(Time: second)*

Name	The Execution Time of Each Iteration.													Total
Iteration	1	2	3	4	5	6	7	8	9	10	11	12	13	Runtime
case 1	0.27	0.46	0.52	0.53	0.53	0.42	0.47	0.48	0.46	—	—	—	—	4.59
case 2	2.11	4.26	4.42	3.75	4.01	4.73	3.82	—	—	—	—	—	—	32.31
case 3	0.53	0.72	0.61	0.56	0.57	0.61	0.61	0.60	0.56	0.58	—	—	—	7.15
case 4	3.15	5.06	5.77	4.28	5.61	5.63	5.72	4.72	4.55	5.10	5.34	4.92	—	64.86
case 5	5.71	8.23	7.55	8.19	8.43	7.27	9.46	8.91	8.44	—	—	—	—	79.36
case 6	10.58	26.72	28.44	29.35	29.46	30.07	38.22	31.15	32.43	33.61	32.07	—	—	353.54
case 7	37.26	52.65	58.72	56.66	55.31	57.81	61.52	59.31	53.17	—	—	—	—	544.73
case 8	16.43	22.71	26.66	28.52	31.95	33.73	35.71	37.07	37.34	39.48	—	—	—	360.03
case 9	12.25	23.46	24.72	25.11	27.73	26.24	28.03	29.64	—	—	—	—	—	275.73
case 10	36.32	42.73	48.61	55.42	61.74	65.45	71.33	73.42	77.06	79.41	82.35	83.57	84.26	878.63

order to eliminate the resource conflict. And note that the ratio of last iteration is slightly larger than the ratio of first iteration. The solution of last iteration is used as the final solution and thus, the final energy consumption ratio is the same to the ratio of last iteration. The results show that our approach not only completely eliminates the conflicts but also minimizes the energy consumption, further generating an energy-efficient route to each vehicle.

The execution time is the third critical metric and it is directly involved in the real-time performance of the control framework. In this experiment, we analyze the execution time of the EEPC framework, including graph construction and initialization, algorithm execution. Meanwhile, we also analyze the execution time of each iteration when performing the EEPC framework. Table 4 shows the execution time of each iteration and the total execution time of the EEPC framework. In the first iteration, the executing time is the shortest-path algorithm running on the original graph. With the iteration proceedings, some vehicles need to explore the alternative routes to avoid the resource conflicts. Note that in general, the length of new route is longer than the length of original routing path. Thus, the execution time of each iteration increases as the length of routing path increases. In terms of small-scale cases, it is very fast to find a practical solution, where each vehicle has its own route running on the network graph with a conflict-free guarantee. In terms

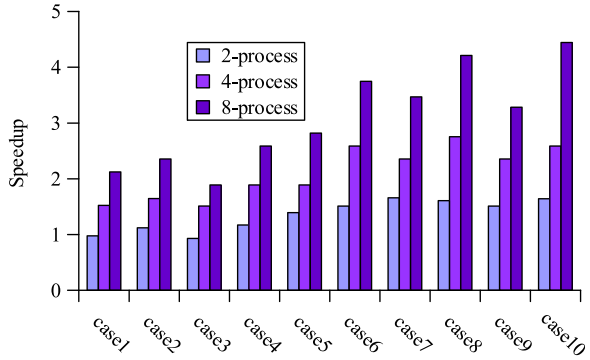

Fig. 7. Speedups of coarse-grained parallel approach using 2, 4, and 8 cores, respectively. This approach uses basic task assignment running on multi-core distributed-memory platform.

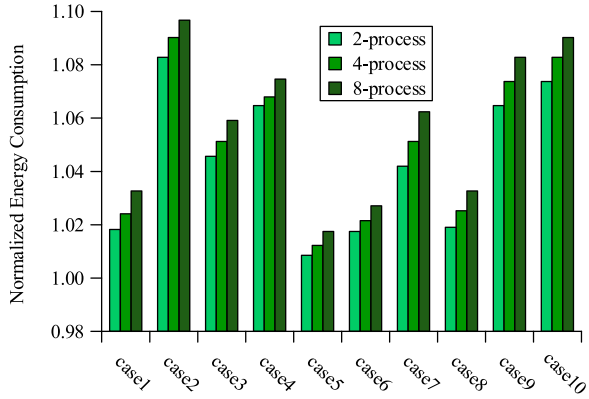

Fig. 8. The normalized energy consumption results of coarse-grained parallel approach using 2, 4, and 8 cores, respectively. This approach uses basic task assignment running on multi-core distributed-memory platform.

of large-scale cases, it needs to take a slightly longer time to find a feasible solution impacting the real-time efficiency of autonomous control systems. While there is a slightly impact on the real-time performance, our control approach is effective to minimize the total energy consumption of all CVs and to completely eliminate the resource conflicts between vehicles, especially for large-scale CVs.

From Tables 2, 3, and 4, EEPC uses the limited number of iterations to find a feasible control solution in that each vehicle has its own non-conflicting route. With the basic shortest-path control scheme, each vehicle has the shortest route further generating the minimum energy consumption and execution time. It is a pity that the route has the resource conflict problem, impacting the traffic safety. To solve the problem, EEPC imposes cost function to encourage the conflicted vehicle to make a detour to select the alternative route. The alternative route typically is longer than the original shortest route, further increasing the energy consumption and execution time. Comparing with the basic shortest-path control scheme, the nature of EEPC trades the energy consumption and execution time for conflict elimination, further guaranteeing the traffic safety.

In addition, the execution time of EEPC to find a feasible solution is relatively real time especially for large-scale CVs in real-time traffic applications. To obtain a better real-time performance, we further accelerate the execution time of EEPC by multi-core parallel techniques. Actually, the execution time of EEPC is typically better than the traffic jam time of time-variant traffic applications. Because the time-variant traffic applications are uncontrollable and their traffic jam time is very length and unacceptable in practical scenarios.

In summary, in the EEPC framework, our cost function can iteratively update the weight of the network graph to determine which vehicle needs the resource most, further avoiding the conflicts between vehicles. Moreover, the core of EEPC framework is the shortest-path algorithm, which is used to connect the original node to the destination node, further minimizing the energy consumption. The shortcoming is to take a slightly longer time to implement the feasible solution in the EEPC framework. To improve real-time efficiency, we adopt multi-core parallel techniques to accelerate the runtime of EEPC framework.

7.3 Coarse-Grained Parallel Control Study

Our coarse-grained parallel approaches enable our EEPC framework running on the multi-core distributed-memory computing platforms. In the coarse-grained scheme, all the tasks are assigned to different processor cores and the data communication between processor cores is performed by MPI-based messages. In the second set of experiments, we focus on the basic task assignment approach and evaluate its impacts on the speedup and energy consumption of parallel control approach in the EEPC framework.

Considering that our used multi-core processor system is equipped with eight cores, we select two, four, and eight cores to parallelize the control approach, respectively. Fig. 7 shows the available speedups of parallel control approach using basic task assignment, comparing to the total runtime of the sequential control approach. In terms of two cores, our parallel control approach can provide about 1.3× speedup on average. The achieved speedup is slow when the size of benchmark increases. In terms of four cores, our parallel control approach achieves about 2.1× speedup on average. This is a slightly improvement comparing with the speedup of two cores and its speedup is still slow when we increase the benchmark size. In terms of eight cores, the speedup of about 3.2× is achieved on average and it is about 2.5× faster than the speedup of two cores. Notably, the maximum speedup is about 4.5× using eight cores. These results show that the basic task assignment is effective to parallelize the control approach in the EEPC framework, although its scalable speedup is slow.

Next we compare the energy consumption results of coarse-grained parallel control approach with two, four, and eight processor cores, respectively. Fig. 8 shows their energy consumption results normalized to the original sequential control approach. Our parallel approach has the impacts on the final energy consumption and in general, it increases about $2\% \sim 10\%$ energy consumption. Note that in terms of two cores, its energy consumption is smaller than the results of four cores as well as the results of eight cores. With the increasing number of cores, the energy consumption is worsening. On average, the energy consumption of four cores is about 1.1× larger than the energy consumption of two cores. And in terms of eight cores, on average, its energy consumption is about 1.3× higher than the energy consumption of two cores. These results display that the energy consumption increases when the number of used cores increases in the EEPC framework. It means that

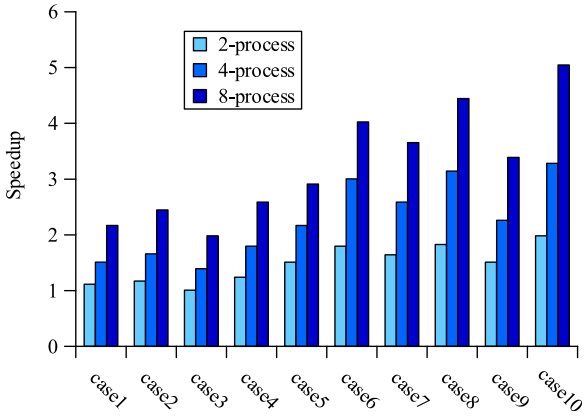

Fig. 9. Speedups of region-based parallel control approach using 2, 4, and 8 cores, respectively. This approach uses region-based partitioning to guide the task assignment running on multi-core distributed-memory platform.

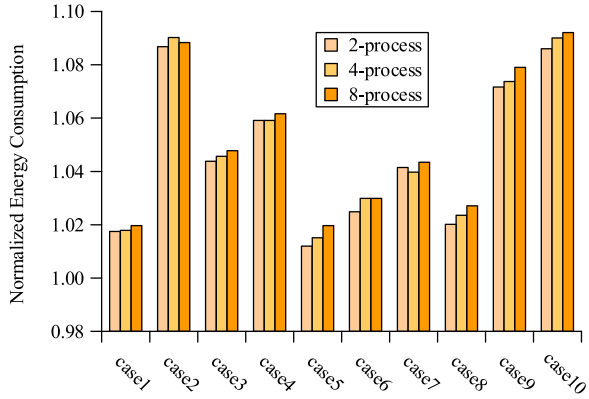

Fig. 10. Normalized energy consumption of region-based parallel control approach using 2, 4, and 8 cores, respectively. This approach uses region-based partitioning to guide the task assignment running on the multi-core distributed-memory platform.

our basic parallel control approach reduces the execution time of EEPC framework at the cost of energy consumption.

In this experiment, we leverage the basic task assignment to control the routing of CVs in parallel. We demonstrate the effectiveness of parallel control approach to reduce the runtime, further improving the real-time performance. It is a pity that the basic task assignment enables the parallel control approach to perform task-level synchronization generating the expensive communication overheads, thereby impacting the scalable speedup. In addition, using basic task assignment, our parallel approach has an impact on energy consumption. To improve the parallel approach, the region-based task assignment will be discussed in the following section.

7.4 Region-Based Parallel Control Study

In the EEPC framework, the region-based parallel control approach is the other coarse-grained scheme and it leverages the region-based recursive partitioning to guide the task assignment. In the next series of experiments, we study the region-based task assignment approach and evaluate its impacts on the speedup and energy consumption of parallel control approach in the EEPC framework.

It is the same to the basic coarse-grained scheme, we still select two, four, and eight cores to perform the parallel control of CVs. Fig. 9 shows the available speedups of the coarse-grained region-based parallel control approach when comparing to the total runtime of the sequential control approach. On average, our region-based parallel control approach achieves about 1.4×, 2.3×, and 3.6× speedups using two, four, and eight cores, respectively. Benefiting from region-based recursive partitioning, the tasks in their respective subregions are parallelized without task-level synchronization. This is due to that each subregion has the independent tasks. Thus, we perform the region-level synchronization rather than task-level synchronization, further reducing the communication overheads. Thus, the available speedup is improved significantly. We believe that leveraging region recursive partitioning to guide the task assignment is an attractive approach to parallel control the routing of CVs in the EEPC framework.

We then study and evaluate the energy consumption of region-based parallel control approach using two, four, and eight processor cores, respectively. Fig. 10 gives their normalized energy consumption to the sequential control approach. Overall, the region-based partitioning has also a slightly impact on the energy consumption results of parallel control approach. This impact is the similar to the basic task assignment of parallel control approach. In terms of two cores, our region-based parallel approach increases about 3.5 percent energy consumption on average. When adding the number of cores, the energy consumption is slightly increased only. We believe that this impact is still acceptable when improving the real-time performance in large-scale autonomous transportation systems. Comparing with basic task assignment, it is more efficient to leverage the region-based recursive partitioning to guide the task assignment so that we control the routing of CVs in parallel. Thus in the EEPC framework, our region-based parallel control is more effective to improve the real-time performance with an acceptable energy consumption.

Figs. 9 and 10 demonstrate the effectiveness of our energy-efficient parallel control approach by exploiting region-based recursive partitioning. In this experiment, our approach scales to eight processor cores to achieve about 3.6× speedup on average. In terms of large-scale case10, it can achieve the maximum speedup of 5× using 8 cores and it only takes about 170 seconds to find an energy-efficient conflict-free control solution in the EEPC framework. We believe that it is very attractive to deploy our EEPC framework into the modern autonomous transportation systems to control the routing of large-scale CVs.

7.5 Fine-Grained Parallel Control Study

Our fine-grained scheme enables the EEPC running on the multi-core shared-memory platform. In the scheme, the task of controlling a single-vehicle routing is parallelized using multi-threading techniques. In the following series of experiments, we evaluate the available speedup and energy consumption of fine-grained parallel control approach in the EEPC framework.

Fig. 11 shows the speedup as a function of the number of threads for fine-grained parallel control approach. In terms of two threads, a speedup of about 1.2× is achieved on average. We also observe that the available speedup is slow down with the number of threads increases. Note that our

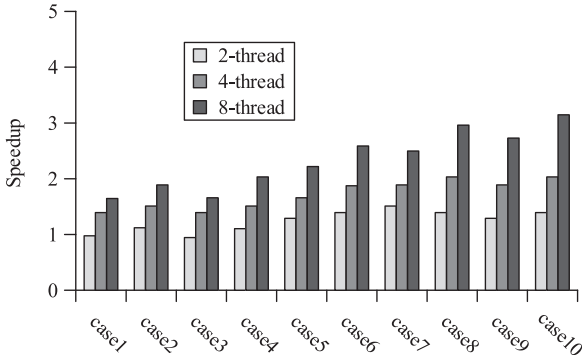

Fig. 11. The available speedups of fine-grained parallel control approach using 2, 4, and 8 threads, respectively. This approach is performed on the multi-core shared-memory platform.

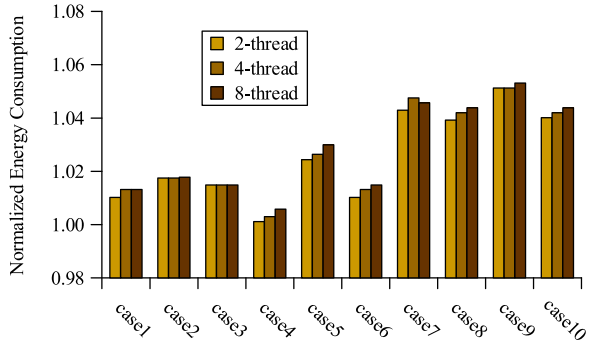

Fig. 12. The normalized energy consumption results of fine-grained parallel control approach using 2, 4, and 8 threads, respectively. This approach is performed on the multi-core shared-memory platform.

fine-grained scheme achieves about 1.7× and 2.3× speedups on average using four and eight threads, respectively. The results show that the scalable speedup of the fine-grained scheme is very slow, depending heavily on the memory architecture of multi-core processor systems. In fine-grained scheme, pairs of cores will share the data in the same memory. When we use two threads, one thread will communicate with the other thread through the shared cache. When we adopt four or eight threads, the communication between threads will be probably occur at the main memory, severely impacting the available speedup. The speedup of fine-grained scheme is poor in comparison with the speedup of coarse-grained scheme. Coarse-grained scheme is the more attractive parallel approach to provide a better speedup in the EEPC framework.

Fig. 12 shows the energy consumption of fine-grained parallel control approach normalized to the original sequential control approach. The fine-grained scheme has also a slightly impact on the final energy consumption and we observe that in all cases, the maximum change to energy consumption is less than 6 percent. The results demonstrate that the fine-grained scheme is feasible to generate the energy-efficient control solution. In terms of small-scale cases, from case 1 to case 4, the maximum energy consumption is less than 2 percent. In terms of all the cases, the fine-grained scheme does not significantly impact the final results of energy consumption with more than two threads. In comparison with coarse-grained scheme, the fine-grained scheme has the better results of energy consumption. Fine-grained scheme is the more attractive parallel approach to provide a smaller energy consumption in the EEPC framework.

In the EEPC framework, our control approach and its parallel techniques are very effective to large-scale CVs. our sequential control approach can enable each vehicle to have an energy-efficient conflict-free route and it requires to take a long time to iteratively find a feasible control solution, especially for large-scale CVs. Our parallel techniques can accelerate the runtime to improve the real-time performance with a slightly energy degradation.

7.6 Comparison Study on Parallel Control in EEPC

EEPC performs the sequential cost-based iterative control process to eliminate the resource conflicts while minimizing the energy consumption. The execution time is relatively long especially for large-scale CVs. This motivates us to leverage multi-core parallel methodologies to accelerate the execution time to improve real-time performance. As shown in Tables 5 and 4, all of the proposed parallel control approaches have a better performance speedup comparing to the proposed sequential control approach. And at the meanwhile, these parallel control approaches have the slight degradation on energy efficiency. Here we provide more detailed analysis.

The sequential control approach is heuristic and this results in noise in the sequential control approach. Thus the parallel approaches are difficult to generate the same energy

TABLE 5
The Total Runtime of Proposed Parallel Control Approaches Across Ten Different Benchmarks *(Time: second)*

Methods	Coarse-Grained Parallel Approach			Region-Based Parallel Approach			Fine-Grained Parallel Approach		
Name	2-process	4-process	8-process	2-process	4-process	8-process	2-thread	4-thread	8-thread
case 1	4.67	3.04	2.13	4.17	3.06	2.09	4.78	3.26	2.85
case 2	28.83	20.32	13.46	28	20.19	12.92	28.86	21.54	17.01
case 3	7.85	4.76	3.8	7.15	5.1	3.58	7.74	5.11	4.47
case 4	55.76	34.14	24.85	54	38.17	24.9	57.38	43.26	32.43
case 5	57.93	41.55	27.43	52.82	35.24	27.37	61.06	49.55	36.07
case 6	235.69	139.75	93	207.86	117	88.32	252.57	196.41	135.92
case 7	336.25	226.03	155.6	341.4	209.52	147.22	363.15	302.63	217.71
case 8	229.32	134.81	84.17	200.3	112.47	80	257.16	180.5	120.4
case 9	182.6	113.94	81.82	183.46	102.2	82.5	212	153.18	102.12
case 10	539.04	348.66	195.57	455.25	266.39	171.61	627.37	439.32	283.49

consumption as the sequential approach. In sequential control approach, EEPC routes a current vehicle and then routes the next vehicle and so on. Note that before routing the next vehicle, EEPC knows which resources are used by the previous vehicles. In coarse-grained approach, EEPC concurrently routes a batch of current vehicles and then performs the data synchronization and then concurrently routes the next batch of vehicles and so on. Note that in a batch of vehicles, EEPC does not know which resources are used each other, resulting in some detours. This lengthens the routes of some vehicles, further increasing the energy consumption. Similarly, in fine-grained approach, EEPC uses multiple threads to parallelize the control of a single-vehicle routing and they also lengthen the single-vehicle route increasing energy consumption. Moreover, the energy consumption of fine-grained approach is slightly smaller than coarse-grained approach. Thus it is difficult to design other parallel approaches to keep same energy efficiency as the sequential approach.

8 Discussion and Possible Limitations

The simulation results show that our EEPC framework is a desirable choice for autonomous control systems in the context of cyber-physical systems applications. The EEPC framework can iteratively eliminate the resource conflict while minimizing the energy consumption. Benefiting from multi-core parallel techniques, EEPC also performs better in terms of processing time. It is convincing that our EEPC framework is an attractive choice for modern intelligent transportation systems.

Our key goal is to reduce the energy consumption while eliminating the conflicts between CVs in autonomous control systems. We leverage a cost function to iteratively update the resource weight and further to determine which vehicle needs the resource most. Since the iterative process is time-consuming especially for large-scale CVs, our second goal is to reduce the processing time to improve the real-time performance of the EEPC framework. We present two parallel approaches: coarse-grained scheme and fine-grained scheme. In coarse grain we explore the task assignment to parallelize the multi-vehicle routing while in fine grain we parallelize the task of controlling a single-vehicle routing. Coarse-grained scheme has better speedup and fine-grained scheme has smaller energy consumption. We can believe that the EEPC framework meet the requirement of autonomous control systems for cyber-physical systems applications very well.

There are some limitations about our cost-based iterative EEPC framework at this stage. First, we use average speed to replace the practical time-variant speed for building the energy consumption model. This enables us to have the chance to explore the energy-efficient conflict-free control solutions. Our framework can fast generate the feasible solution and the processing time typically is on the order of seconds or minutes. While it is probably different from the practical solutions, it is still meaningful to guide the control of CVs in modern transportation systems. In the future work, we attempt to deploy the EEPC framework into the intelligent transportation system of Guangzhou city, China. Second, our work is based on macroscopic model and it does not consider much about other features of CVs running on the ground traffic network. We simplify the network graph construction and explore how to reduce the conflict and energy consumption. Thus, the technical contribution in this paper is rather limited. We improve the basic EEPC framework considering the microscopic features of the network graph in our future work. Third, developing the efficient autonomous control system is non-trivial. Previous work focus on the time-variant control approach and they cannot always generate the most energy-efficient solutions. Moreover, they also cannot completely eliminate the conflicts between CVs. In addition, the traffic jam time is very length and unacceptable in practical time-variant scenarios. In contrast, our work does not consider the time-variant problem in the EEPC framework. Thus we improve the EEPC framework considering the time-variant problem in our future work. We believe that our EEPC framework will be an important part of autonomous control systems in the context of modern cyber-physical systems applications.

9 Conclusion

It is a very important problem to control the routing of CVs in the cyber-physical system applications. We propose a cost-based iterative control framework able to generate the energy-efficient route, eliminate the conflict, and leverage the multi-core parallel techniques to improve the real-time performance for autonomous control systems. We leverage cost function to iteratively eliminate the resource conflicts between vehicles, further ensuring the safety driving. At the meanwhile, energy-efficient route is obtained by minimizing the usage of road segment resources. In addition, we explore the different grained parallel approaches to accelerate the processing time, further improving the real-time performance. Simulation experiments demonstrate the effectiveness of our control framework.

Acknowledgments

We appreciate the insightful comments and feedbacks from anonymous reviewers. This work is partly supported by the National Natural Science Foundation of China (NSFC) under Grant No. 61433019 and No. 61802446. This work is also partly supported by the Projects for Guangdong Introducing Innovative and Entrepreneurial Teams under Grant No. 2016ZT06D211, and Guangdong Basic and Application Basic Research Teams under Grant No. 2018B030312002.

References

[1] A. Sciarretta, G. Nunzio, and L. Ojeda, "Optimal ecodriving control: Energy-efficient driving of road vehicles as an optimal control problem," *IEEE Control Syst. Mag.*, vol. 35, no. 5, pp. 71–90, Oct. 2015.

[2] M. Kubicka, J. Klusacek, A. Sciarretta, A. Cela, H. Mounier, L. Thibault, and S. Niculescu, "Performance of current eco-routing methods," in *Proc. IEEE Intell. Vehicles Symp.*, 2016, pp. 472–477.

[3] C. Guo, B. Yang, O. Andersen, C. S. Jensen, and K. Torp, "EcoMark 2.0: Empowering eco-routing with vehicular environmental models and actual vehicle fuel consumption data," *Geoinformatica*, vol. 19, no. 3, pp. 567–599, Jul. 2015.

[4] Z. Chen, L. Li, B. Yan, C. Yang, C. Martnez, and D. Cao, "Multimode energy management for plug-in hybrid electric buses based on driving cycles prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 10, pp. 2811–2821, Oct. 2016.

[5] L. Guzzella and A. Sciarretta, *Vehicle Propulsion Systems: Introduction to Modeling and Optimization*, Berlin, Germany: Springer, 2013.
[6] G. Nunzio, L. Thibault, A. Sciarretta, "A model-based eco-routing strategy for electric vehicles in large urban networks," in *Proc. IEEE 19th Int. Conf. Intell. Transp. Syst.*, 2016, pp. 2301–2306.
[7] R. Karp, "Reducibility among combinatorial problems," *Complexity Comput. Comput.*, In: Miller R.E., Thatcher J.W., Bohlinger J.D. (Eds), The IBM Research Symposia Series. Springer, Boston, MA. 1972. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4684-2001-2_9#citeas
[8] I. Amundson and X. Koutsoukos, "A survey on localization for mobile wireless sensor networks," in *Proc. Int. Workshop Mobile Entity Localization Tracking GPS-Less Environments*, 2009, pp. 235–254.
[9] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Trans. Robot. Autom.*, vol. 20, no. 2, pp. 243–255, Apr. 2004.
[10] F. Miao, S. Han, A. M. Hendawi, M. E. Khalefa, J. A. Stankovic, and G. J. Pappas, "Data-driven distributionally robust vehicle balancing using dynamic region partitions," in *Proc. ACM/IEEE 8th Int. Conf. Cyber-Phys. Syst.*, Apr. 2017, pp. 261–271.
[11] A. Arsie, K. Savla, and E. Frazzoli, "Efficient routing algorithms for multiple vehicles with no explicit communications," *IEEE Trans. Autom. Control*, vol. 54, no. 10, pp. 2302–2317, Oct. 2009.
[12] Y. Wan and S. Roy, "A scalable methodology for evaluating and designing coordinated air-traffic flow management strategies under uncertainty," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 4, pp. 644–656, Dec. 2008.
[13] G. Naus, J. Ploeg, M. Molengraft, W. Heemels, and M. Steinbuch, "Design and implementation of parameterized adaptive cruise control: An explicit model predictive control approach," *Control Eng. Practice*, vol. 18, no. 8, pp. 882–892, Aug. 2010.
[14] R. Negenborn, B. Schutter, and J. Hellendoorn, "Multi-agent model predictive control for transportation networks: Serial versus parallel schemes," *Eng. Appl. Artif. Intell.*, vol. 21, no. 3, pp. 353–366, Apr. 2008.
[15] K. Kim, "Collision free autonomous ground traffic: A model predictive control approach," in *Proc. ACM/IEEE Int. Conf. Cyber-Phys. Syst.*, 2013, pp. 51–60.
[16] L. McMurchie and C. Ebeling, "Pathfinder: A negotiation-based performance-driven router for FPGAs," in *Proc. ACM Int. Symp. Field-Programmable Gate Arrays*, 1995, pp. 111–117.
[17] D. Chen, J. Cong, and P. Pan, "FPGA design automation: A survey," *J. Foundations Trends Electron. Des. Autom.*, vol. 1, no. 3, pp. 139–169, Jan. 2006.
[18] A. Kahng, I. Markov, and J. Hu, *VLSI Physical Design: From Graph Partitioning to Timing Closure*, Berlin, Germany: Springer, 2011.
[19] J. Leskovec, K. Lang, A. Dasgupta, M. Mahoney, "Statistical properties of community structure in large social and information networks," in *Proc. 17th Int. Conf. World Wide Web*, 2008, pp. 695–704.

Minghua Shen received the PhD degree in computer science from the Peking University, in 2017. He is currently an associate researcher with the School of Data and Computer Science, Sun Yat-sen University, China. His research interests include FPGA synthesis, heterogeneous and parallel computing, and cyber-physical systems. He is a member of the IEEE and ACM.

Guojie Luo received the BS degree in computer science from Peking University, Beijing, China, in 2005, and the MS and PhD degrees in computer science from UCLA, in 2008 and 2011, respectively. He is currently an associate professor with the School of EECS, Peking University. His research interests include electronic design automation, heterogeneous computing and emerging devices, and cyber-physical systems. He is a member of the IEEE and ACM.

Nong Xiao received the BS and PhD degrees in computer science from the College of Computer, National University of Defense Technology (NUDT), China, in 1990 and 1996, respectively. He is currently a professor with the School of Data and Computer Science, Sun Yat-sen University. His current research interests include parallel and distributed computing, computer storage system, and computer architecture. He is a senior member of the IEEE and a member of ACM.