Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

CFCO: A Social-Welfare-Maximizing Collaborative Coalition Formation Method for Task Offloading in Vehicular Edge Computing

Kangli Zhao
Aba Teachers University
Huaming Gong
94305400@qq.com

Aba Teachers University
Wenpei Xiao
Aba Teachers University
Chunyan Zhou
Aba Teachers University
Qiyue Zhang
Aba Teachers University

Research Article

Keywords: Vehicular edge computing, task offloading, D2D collaboration, coalition formation game, social welfare, Jain fairness, NGSIM traces

Posted Date: August 25th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10210269/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

CFCO: A Social-Welfare-Maximizing Collaborative Coalition Formation Method for Task Offloading in Vehicular Edge Computing

Kangli Zhao[1], Huaming Gong[1*], Wenpei Xiao[1], Chunyan Zhou[1], Qiyue Zhang[1]

[1*]School of Computer Science and Technology, Aba Teachers University, Wenchuan, 623002, China.

*Corresponding author(s). E-mail(s): 94305400@qq.com;
Contributing authors: lia_zkl@foxmail.com; 3113899330@qq.com;
430349296@qq.com; 2997096460@qq.com;

Abstract

Vehicular edge computing (VEC) provides computation resources close to data sources for autonomous driving, cooperative perception, and intelligent vehicular services. However, limited on-board computing resources, fluctuating V2I reliability, and shared communication and computation resources at edge servers make it difficult for a single local-computing or MEC-offloading policy to achieve consistently high system utility. To address this problem, this paper proposes CFCO, a Collaborative Coalition Formation for Offloading method that maximizes social welfare. CFCO unifies local computing, D2D collaborative computing, and MEC offloading as three execution modes. It models cooperation among non-MEC vehicles as a coalition formation process and alternates coalition merge/split operations, MEC-mode refinement, and local vehicle-migration actions under a strict social-welfare improvement rule. Unlike policies that optimize only delay or energy, CFCO jointly captures early-completion benefit, energy cost, and delay penalty, thereby reflecting heterogeneous task preferences across vehicles. In this revised study, we evaluate CFCO on real-world vehicle trajectories from the Next Generation Simulation (NGSIM) dataset and report statistically robust results averaged over ten independent random seeds. In a representative one-dimensional NGSIM snapshot setting with $\boldsymbol{N = 60}$ vehicles and V2I availability probability **0.7**, CFCO achieves a social welfare of **481.277**, an average task delay of **0.225** s, a task success ratio of **0.998**, and a Jain fairness index of **0.716**. Compared with the second-best BR-LS baseline, CFCO improves social welfare by **0.2%** while reducing average delay by **1.9%**, and runs more than twelve times

faster. Compared with MEC-only, CFCO improves social welfare by **71.0%** and reduces average delay by **50.4%**. The final CFCO decision keeps **20.5%** of vehicles in local computing, offloads **22.5%** to MEC, and assigns **57.0%** to D2D collaborative coalitions. A small-scale exact optimality-gap study further shows that CFCO is within **0.03%** of the optimal ILP solution. Parameter sweeps further show that CFCO remains superior under varying local CPU capacity, D2D bandwidth, MEC capacity, vehicle density, and V2I reliability.

Keywords: Vehicular edge computing, task offloading, D2D collaboration, coalition formation game, social welfare, Jain fairness, NGSIM traces

1 Introduction

With the rapid development of autonomous driving, intelligent transportation systems, and the Internet of Vehicles (IoV), vehicles are evolving from simple communication terminals into mobile intelligent nodes integrating sensing, computation, storage, and decision making. High-definition map updates, cooperative perception, vehicular video analytics, augmented-reality navigation, and multi-task autonomous-driving services are typically computation intensive and delay sensitive [1–4]. If such services rely entirely on on-board processors, their execution quality is constrained by local CPU frequency, energy budget, and thermal limitations. If they rely entirely on roadside units (RSUs) or MEC servers, unstable V2I links, road blockage, vehicular handover, and multi-user bandwidth sharing introduce additional transmission and queueing delays [5–7].

Mobile edge computing (MEC) deploys computation resources at the network edge and can reduce cloud round-trip delay and core-network load [8, 9]. In VEC, vehicles may upload tasks to an RSU or an edge server through V2I links. Nevertheless, an MEC server is not an unlimited resource. When many vehicles offload simultaneously, upload bandwidth and MEC cores are shared among users. In addition, the distance between vehicles and the RSU and the availability of V2I links cause unequal offloading opportunities. In contrast, D2D/V2V communication allows neighboring vehicles to share idle computation resources and form horizontal collaborative-computing modes [4, 10–12]. Such collaboration can supplement MEC resources when V2I is unavailable or congested, but it also introduces coalition-head selection, cooperative-benefit evaluation, D2D connectivity, and coalition stability issues.

Existing studies can be broadly divided into three categories. The first category uses cloud-edge-end collaboration or deep reinforcement learning for offloading and resource allocation [2, 3, 13, 14]; these approaches can handle complex state spaces but often require substantial training cost and may be difficult to interpret online. The second category focuses on vehicular collaboration, clustering, or multi-hop offloading [10–12, 15]; these methods highlight the value of neighboring vehicular resources, but some of them still treat D2D collaboration and MEC offloading separately. The third category adopts game-theoretic, auction-based, or market-based mechanisms to

model selfish participants and resource trading [16–21]. These mechanisms help incorporate incentives and social welfare, but their complexity and deployability remain challenging in dynamic VEC environments. Overall, there remains a need for an interpretable collaborative offloading framework that jointly covers local computing, D2D collaboration, and MEC offloading while directly optimizing social welfare.

This paper proposes CFCO, which formulates the task offloading problem as coalition-structure optimization with an MEC mode set. Each vehicle has an independent task, a deadline, a delay-benefit weight, and a delay-penalty coefficient. The system objective is the sum of individual net welfare. CFCO first initializes the MEC-user set by comparing local welfare with congested MEC welfare. It then forms D2D coalitions among non-MEC vehicles and further adjusts the execution modes by mode refinement and local search.

The main contributions of this revised study are as follows:

- We construct a unified local-D2D-MEC model that is consistent with the implementation. In a D2D coalition, the vehicle with the highest CPU frequency acts as the coalition head; the coalition computation delay is determined by the maximum task workload, whereas the head's computation energy is accumulated over the total coalition workload. On the MEC side, V2I bandwidth is equally shared and finite MEC cores are captured by a congestion factor.
- We design CFCO as a social-welfare-improving coalition formation algorithm. Each merge, split, MEC-state toggle, and local vehicle migration is accepted only when it strictly increases social welfare. This produces monotonic improvement over a finite state space.
- We evaluate CFCO using the actual experimental outputs generated by the project code on real-world NGSIM vehicle trajectories [22]. Vehicle positions, velocities, vehicle classes, and physical dimensions are derived directly from the trace data, while local CPU frequencies and task deadlines are mapped from vehicle-class and velocity information, respectively. This trace-driven setting substantially increases the realism of the evaluation compared with purely synthetic topologies.
- We report statistically robust results averaged over ten independent random seeds, including mean values, standard deviations, 95% confidence intervals, and paired Wilcoxon signed-rank tests. A small-scale exact optimality-gap study demonstrates that CFCO is within 0.03% of the optimal ILP solution. In addition to the main result table, the paper analyzes convergence curves, delay CDFs, delay-energy trade-offs, coalition-size distributions, mode ratios, delay decomposition, runtime, and multi-dimensional parameter sweeps.

2 Related Work

2.1 Vehicular edge computing and collaborative offloading

MEC has become an important paradigm for supporting low-latency mobile applications by placing computing resources at the network edge [8, 9]. In vehicular networks, Zhao et al. [6] investigated offloading and resource allocation in cloud-assisted MEC and jointly modeled cloud and edge resources. Wu et al. [3] incorporated vehicles,

edge servers, and cloud servers into a multilayer reinforcement-learning framework for online asynchronous task offloading. Liu et al. [1] constructed a cloud-edge-end collaborative multi-objective model for large-scale vehicular task offloading. These studies demonstrate that a single MEC server or a single offloading path is insufficient for complex vehicular tasks.

D2D/V2V collaboration is a key feature distinguishing VEC from generic mobile edge computing. Hui et al. [10] studied collaborative vehicular task offloading in 5G heterogeneous networks with a game-theoretic framework. Liu et al. [11] exploited vertical and horizontal collaboration to maximize task completion rate and emphasized the value of idle vehicular resources. Huang et al. [12] studied multi-vehicle multi-hop task offloading. Shen et al. [5] generated task-offloading-oriented vehicular clusters while considering both collaboration performance and cluster stability. These works show that horizontal cooperation among vehicles can complement V2I/MEC resources, but collaboration formation and resource availability must be handled simultaneously.

2.2 Games, coalitions, and mechanism design

Coalition formation games are suitable for describing how participants form cooperative structures under local information. Saad et al. [23] applied coalition formation to distributed cooperation among vehicular RSUs, while Bogomolnaia and Jackson [24] studied the stability of hedonic coalition structures and provided theoretical foundations for distributed coalition formation. Recent vehicular cooperation and edge-market mechanisms further incorporate social welfare, incentive compatibility, and truthfulness constraints. For example, Wu et al. [19] studied a socially optimal marketplace for splittable task offloading; Liu et al. [20] investigated truthful mechanisms for computation offloading and resource sharing in vehicle computing; and Wu et al. [21] jointly modeled vehicle coalitions, model deployment, and task offloading. In contrast, this paper does not design payment or incentive-compatibility mechanisms. Instead, it uses coalition formation as an interpretable heuristic process for social-welfare-maximizing offloading decisions.

2.3 Difference from existing work

Compared with reinforcement-learning methods, CFCO does not require offline training. Its decision process consists of merge, split, mode switching, and local migration operations, which makes each performance change easier to interpret. Compared with pure D2D clustering, CFCO explicitly maintains an MEC offloading set and can switch among local, D2D, and MEC modes according to V2I availability and MEC congestion. Compared with auction or market mechanisms, CFCO does not rely on price design and directly optimizes the net welfare defined by benefit, energy cost, and delay penalty. This paper uses the Jain fairness index [25] to evaluate the distribution of individual welfare among vehicles, but fairness is not an explicit optimization constraint.

3 System Model and Problem Formulation

3.1 Network and task model

We consider a one-dimensional vehicular road topology. The configured synthetic-fallback road length is $L = 2000$ m, while the reported experiments use NGSIM trace coordinates directly: vehicle longitudinal positions are converted to SI units, shifted so that the minimum coordinate of the sampled snapshot is zero, and the RSU is placed at the center of the observed coordinate span. A total of N vehicles are sampled from the Next Generation Simulation (NGSIM) dataset [22]. The vehicle set is

$$\mathcal{V} = \{1, 2, \ldots, N\}, \tag{1}$$

where the default number of vehicles is $N = 60$.

NGSIM is a publicly available vehicle-trajectory dataset collected by the U.S. Department of Transportation. It records high-resolution vehicle states, including longitudinal and lateral positions, velocity, acceleration, vehicle class, vehicle length and width, lane identifier, and time headway. In our simulator, an NGSIM snapshot is defined by a unique combination of road location and frame identifier. We pre-process the raw CSV file, discard duplicate rows, convert imperial units to SI units, and retain only snapshots that contain at least $\max\{N, 100\}$ vehicles. For each experimental run, a snapshot is randomly selected and N vehicles are sampled from it without replacement in the reported configurations. The position of vehicle i is denoted by p_i and is obtained directly from the NGSIM trace after shifting the minimum longitudinal coordinate to zero.

The NGSIM vehicle-class field distinguishes three categories: motorcycle, automobile (sedan), and truck/SUV. We map each class to a realistic local CPU frequency range:

- Motorcycle: $f_i \sim \mathcal{U}(0.6, 1.2)$ GHz,
- Automobile: $f_i \sim \mathcal{U}(1.0, 2.0)$ GHz,
- Truck/SUV: $f_i \sim \mathcal{U}(0.8, 1.6)$ GHz.

Vehicles whose class is not mapped fall back to the default range $f_i \sim \mathcal{U}(0.8, 2.4)$ GHz. The CPU frequency of the MEC server is $f_{\text{mec}} = 40$ GHz, and the number of parallel MEC cores is $C_{\text{mec}} = 4$.

Each vehicle carries an independent task described by input data size L_i, computation density ρ_i, total workload D_i, deadline $\tau_i^{\max}$, delay-benefit weight w_i, and delay-penalty coefficient λ_i:

$$L_i \sim \mathcal{U}(10^5, 5 \times 10^5) \text{ bits}, \tag{2}$$

$$\rho_i \sim \mathcal{U}(500, 2500) \text{ cycles/bit}, \quad D_i = L_i \rho_i, \tag{3}$$

$$w_i \sim \mathcal{U}(5, 25), \quad \lambda_i \sim \mathcal{U}(0.5, 3.0). \tag{4}$$

The deadline is no longer purely random. Instead, it is coupled to the real velocity v_i observed in the NGSIM trace:

$$\tau_i^{\max} = \tau_{\text{high}}^{\max} - \beta \left(\tau_{\text{high}}^{\max} - \tau_{\text{low}}^{\max}\right) \frac{v_i}{v_{\max}} + \epsilon_i, \tag{5}$$

where $\tau_{\text{low}}^{\max} = 0.2$ s, $\tau_{\text{high}}^{\max} = 1.0$ s, $\beta = 0.5$ is the velocity-deadline coupling factor, $v_{\max} = \max(\max_j v_j, 1 \text{ m/s})$, and $\epsilon_i \sim \mathcal{U}(-0.03, 0.03)$ s is a small jitter. The deadline is then clipped to $[0.2, 1.0]$ s. Intuitively, faster vehicles are assigned tighter deadlines, reflecting the fact that high-speed driving tasks are typically more delay sensitive.

3.2 Communication model

The D2D link bandwidth is $B_d = 5$ MHz, the transmission-power parameter is $P_d = 0.1$ W, and the maximum D2D communication range is $R_d = 200$ m. The distance between vehicles i and j is

$$d_{ij} = \max(|p_i - p_j|, d_0), \quad d_0 = 1 \text{ m}. \tag{6}$$

The D2D path loss is

$$\text{PL}_{ij}^{dB} = \text{PL}_0^d + 10\alpha_d \log_{10}\left(\frac{d_{ij}}{d_0}\right), \tag{7}$$

where $\text{PL}_0^d = 38$ dB and $\alpha_d = 3.5$. The linear channel gain is $10^{-\text{PL}_{ij}^{dB}/10}$ and is combined with Rayleigh fading $h_{ij} \sim \text{Exp}(1)$ and log-normal shadowing $X_{ij}^{dB} \sim \mathcal{N}(0, 4^2)$. The noise power spectral density is $N_0 = -174$ dBm/Hz, converted in the code to $10^{(N_0-30)/10}$ W/Hz. The D2D rate is

$$R_{ij}^{d2d} = B_d \log_2\left(1 + \frac{P_d h_{ij} 10^{-\text{PL}_{ij}^{dB}/10} 10^{X_{ij}^{dB}/10}}{10^{(N_0-30)/10} B_d}\right). \tag{8}$$

If $d_{ij} > R_d$, then $R_{ij}^{d2d} = 0$.

The total V2I bandwidth is $B_v = 30$ MHz, the configured V2I power parameter is $P_v = 0.2$ W, the RSU coverage radius is $R_v = 400$ m, and the path-loss parameters are $\text{PL}_0^v = 35$ dB, $\alpha_v = 3.0$, and shadowing standard deviation 3 dB. The V2I rate is calculated similarly to the D2D rate. To strictly match the current implementation, the V2I SINR uses the configured power after the code's dBm-style transformation:

$$\tilde{P}_v = 10^{(10\log_{10} P_v + 30)/10}, \tag{9}$$

which is used to compute R_i^{v2i}. This transformation is retained here because the paper reports the results generated by the released simulator. In addition, each vehicle obtains V2I availability with probability $p_v = 0.7$ to emulate blockage, handover failure, and access instability. If a vehicle is outside RSU coverage or its V2I link is unavailable, it cannot use MEC offloading.

3.3 Local computing model

If vehicle i executes its task locally, its delay and energy consumption are

$$T_i^{local} = \frac{D_i}{f_i}, \tag{10}$$

$$E_i^{local} = \kappa f_i^2 D_i, \quad \kappa = 10^{-27}. \tag{11}$$

The local net welfare is

$$u_i^{local} = w_i \max(0, \tau_i^{\max} - T_i^{local}) - E_i^{local} - \lambda_i T_i^{local}. \tag{12}$$

The first term is the early-completion benefit, the second term is the energy cost, and the third term is the delay penalty.

3.4 MEC offloading model

Let $\mathcal{M}$ be the set of MEC-offloading vehicles and $n_{\text{mec}} = |\mathcal{M}|$. The implementation abstracts MEC upload congestion by equal sharing of the total V2I bandwidth. The available V2I rate of vehicle i is

$$r_i^{v2i} = \min\left(R_i^{v2i}, \frac{B_v}{\max(n_{\text{mec}}, 1)}\right). \tag{13}$$

If $r_i^{v2i} \leq 10^3$ bit/s, the MEC link is treated as unusable. The finite-core MEC congestion factor is

$$\eta = \max\left(1, \frac{n_{\text{mec}}}{C_{\text{mec}}}\right). \tag{14}$$

Thus, the MEC offloading delay and energy consumption are

$$T_i^{mec} = \frac{L_i}{r_i^{v2i}} + \eta \frac{D_i}{f_{\text{mec}}}, \tag{15}$$

$$E_i^{mec} = P_v \frac{L_i}{r_i^{v2i}}. \tag{16}$$

The corresponding net welfare is

$$u_i^{mec} = w_i \max(0, \tau_i^{\max} - T_i^{mec}) - E_i^{mec} - \lambda_i T_i^{mec}. \tag{17}$$

3.5 D2D coalition computing model

Non-MEC vehicles can form a D2D coalition $S \subseteq \mathcal{V}$. If $|S| = 1$, the vehicle computes locally. If $|S| > 1$, the coalition head is the vehicle with the highest CPU frequency:

$$h(S) = \arg\max_{j \in S} f_j. \tag{18}$$

Consistent with the code, the D2D coalition computation stage uses a parallel/batch-processing abstraction: when the head processes the tasks in the coalition, the computation delay is determined by the maximum task workload, while the head's computation energy is accumulated over the total coalition workload. Hence, the execution delay is

$$T^S_{\text{exec}} = \frac{\max_{j\in S} D_j}{f_{h(S)}}. \tag{19}$$

The delay of the head vehicle h is

$$T^S_h = T^S_{\text{exec}}. \tag{20}$$

Other members $i \in S \setminus \{h\}$ must first upload their input data to the head through D2D links:

$$T^S_i = \frac{L_i}{R^{d2d}_{ih}} + T^S_{\text{exec}}. \tag{21}$$

The computation energy of the head and the transmission energy of a non-head member are

$$E^S_h = \kappa f_h^2 \sum_{j\in S} D_j, \tag{22}$$

$$E^S_i = P_d \frac{L_i}{R^{d2d}_{ih}}, \quad i \neq h. \tag{23}$$

If the D2D rate from a member to the head is unavailable, the corresponding delay and energy are set to infinity and the coalition is considered infeasible by the optimizer.

3.6 Social welfare and optimization problem

For any execution mode, the net welfare of vehicle i is unified as

$$u_i = w_i \max(0, \tau_i^{\max} - T_i) - E_i - \lambda_i T_i. \tag{24}$$

Let $\mathcal{P}$ be a partition of the vehicle set and $\mathcal{M}$ be the set of MEC users. Vehicles in $\mathcal{M}$ execute in MEC mode. Non-MEC singleton coalitions execute locally, and non-MEC multi-member coalitions execute through D2D collaboration. The system social welfare is

$$W(\mathcal{P}, \mathcal{M}) = \sum_{i\in\mathcal{V}} u_i. \tag{25}$$

This paper solves

$$\max_{\mathcal{P},\mathcal{M}} \; W(\mathcal{P}, \mathcal{M}), \tag{26}$$

subject to

$$\mathcal{P} \text{ is a partition of } \mathcal{V}, \quad \mathcal{M} \subseteq \mathcal{V}, \tag{27}$$

$$\forall i \in \mathcal{M}, \; R^{v2i}_i > 0 \text{ and V2I is available}, \tag{28}$$

$$\forall S \in \mathcal{P}, \; |S| > 1, \; \forall i \in S \setminus \{h(S)\}, \; R^{d2d}_{ih(S)} > 0. \tag{29}$$

The problem simultaneously includes set partitioning, discrete mode selection, and link-feasibility constraints. Its search space grows exponentially with the number of vehicles. Therefore, this paper uses a coalition-formation heuristic rather than exhaustive optimal partitioning.

4 CFCO Algorithm Design

4.1 Overall idea

The core idea of CFCO is to alternately optimize execution-mode selection and D2D coalition formation. The algorithm maintains two state variables: the D2D coalition partition $\mathcal{P}$ and the MEC-user set $\mathcal{M}$. A candidate action is accepted only if it strictly increases $W(\mathcal{P}, \mathcal{M})$, so the algorithm trajectory is a monotonic social-welfare improvement sequence.

Initially, each vehicle is a singleton coalition. For vehicle i, the algorithm compares its local welfare u_i^{local} with its MEC welfare $u_i^{mec}(N)$ under the congested condition $n_{\text{mec}} = N$. If the MEC welfare is higher and finite, vehicle i is inserted into $\mathcal{M}$. This initialization is conservative because it assumes that all candidate vehicles use MEC simultaneously, avoiding over-selection of MEC at the initial stage.

4.2 Merge and split

The merge phase traverses all pairs of non-MEC coalitions (S_1, S_2) and computes the D2D coalition merge gain:

$$\Delta W(S_1, S_2) = W(S_1 \cup S_2) - W(S_1) - W(S_2). \tag{30}$$

The coalition pair with the largest positive gain is merged. In the current experimental configuration, the flag `CFCO_INDIVIDUAL_STABILITY` is disabled; therefore, the merge rule is mainly based on social-welfare improvement. The code still retains an optional individual-stability check requiring that no member's welfare decreases after merging.

The split phase applies a conservative test to each multi-member non-MEC coalition S. If splitting it into singleton coalitions yields higher social welfare,

$$\sum_{i \in S} u_i^{local} > W(S), \tag{31}$$

then the coalition is split into singletons. This implementation does not enumerate all subpartitions of S; instead, it uses the low-complexity rule of splitting the whole coalition into singleton coalitions to reduce online computation.

4.3 Mode refinement and local search

After several merge/split rounds, CFCO performs MEC-mode refinement. Vehicles are traversed in ascending order of local welfare. If a vehicle is currently an MEC user, the algorithm attempts to remove it from $\mathcal{M}$ and re-evaluate it as a non-MEC singleton. If a vehicle is currently not an MEC user, the algorithm attempts to add it to $\mathcal{M}$ and keep it as a singleton. A trial state is accepted only if it strictly increases social welfare:

$$W(\mathcal{P}_{trial}, \mathcal{M}_{trial}) > W(\mathcal{P}_{current}, \mathcal{M}_{current}). \tag{32}$$

In the main loop, CFCO performs at most two merge-split steps followed by one mode-refinement step. If mode refinement brings no improvement, the main loop terminates.

Algorithm 1 CFCO: collaborative coalition formation for offloading

Require: System model, vehicle set $\mathcal{V}$, maximum iteration count $I_{\max}$
Ensure: Coalition partition $\mathcal{P}$ and MEC-user set $\mathcal{M}$

Initialize $\mathcal{P} \leftarrow \{\{1\}, \ldots, \{N\}\}$ and $\mathcal{M} \leftarrow \emptyset$
for each vehicle i **do**
    if $u_i^{mec}(N) > u_i^{local}$ and MEC offloading is feasible **then**
        $\mathcal{M} \leftarrow \mathcal{M} \cup \{i\}$
    end if
end for
repeat
    Find the non-MEC pair (S_1, S_2) with the largest positive merge gain and merge it
    Split any multi-member non-MEC coalition if singleton splitting improves welfare
    Try MEC-state toggles for all vehicles and accept strictly improving states
until $I_{\max}$ is reached or mode refinement yields no improvement
Perform up to two rounds of local search over MEC toggles and non-MEC coalition migrations
return $\mathcal{P}, \mathcal{M}$

Finally, the algorithm performs at most two rounds of deterministic local search. For each vehicle, local search tries two actions: toggling its MEC state or moving a non-MEC vehicle to another non-MEC coalition (or to a new singleton). The best improving action for the current vehicle is immediately applied.

4.4 Pseudocode

4.5 Convergence and complexity

For a finite vehicle set, the number of feasible coalition partitions and MEC-user sets is finite. Every merge, split, mode switch, and local migration in CFCO requires a strict improvement in social welfare; hence, the algorithm cannot cycle in the state space. Because the welfare sequence strictly increases over a finite set of states, the algorithm terminates in finite steps.

In terms of complexity, one merge round checks $O(N^2)$ coalition pairs, and each welfare evaluation may involve $O(N)$ vehicles in the worst case, giving $O(N^3)$ complexity. Split checking is between $O(N)$ and $O(N^2)$ depending on coalition sizes. Mode refinement tries MEC toggling for all vehicles and performs multiple global welfare evaluations, leading to $O(N^2)$ to $O(N^3)$ worst-case cost. Local search attempts vehicle migration among candidate coalitions and is $O(N^3)$ in the worst case. With the maximum iteration count $I_{\max} = 200$ treated as a constant in the experimental configuration, the overall complexity can be written as $O(I_{\max} N^3)$.

Table 1 Experimental parameter settings.

Parameter	Symbol	Default value
Number of vehicles	N	60
Synthetic-fallback road length	L	2000 m
NGSIM coordinate span		sampled from trace snapshot
Task input size	L_i	$\mathcal{U}(10^5, 5 \times 10^5)$ bits
Computation density	ρ_i	$\mathcal{U}(500, 2500)$ cycles/bit
Deadline range	$[\tau_{\text{low}}^{\max}, \tau_{\text{high}}^{\max}]$	$[0.2, 1.0]$ s
Velocity-deadline coupling	β	0.5
Delay-benefit weight	w_i	$\mathcal{U}(5, 25)$
Delay-penalty coefficient	λ_i	$\mathcal{U}(0.5, 3.0)$
Local CPU frequency (motorcycle)	f_i	$\mathcal{U}(0.6, 1.2)$ GHz
Local CPU frequency (automobile)	f_i	$\mathcal{U}(1.0, 2.0)$ GHz
Local CPU frequency (truck/SUV)	f_i	$\mathcal{U}(0.8, 1.6)$ GHz
MEC CPU frequency	f_{mec}	40 GHz
Number of MEC cores	C_{mec}	4
D2D bandwidth	B_d	5 MHz
D2D transmit power	P_d	0.1 W
D2D communication range	R_d	200 m
Total V2I bandwidth	B_v	30 MHz
V2I power parameter	P_v	0.2 W
V2I coverage range	R_v	400 m
V2I availability probability	p_v	0.7
Maximum iteration count	$I_{\max}$	200
Random seeds (main experiment)		10 seeds
Random seeds (scenario sweeps)		5 seeds

5 Experimental Setup

All experiments are implemented in the Python simulator of the project. The main experiment is repeated over ten independent random seeds $\{2024, 42, 123, 7, 99, 314, 2023, 1, 88, 555\}$, and the reported metrics are averaged with 95% confidence intervals. Scenario sweeps use five seeds, and the small-scale exact optimality-gap study also uses ten seeds. The default setting uses $N = 60$ vehicles sampled from NGSIM snapshots, one-dimensional trace coordinates, and V2I availability probability $p_v = 0.7$. The configured road length $L = 2000$ m is used for the synthetic fallback topology, whereas the NGSIM experiments use the observed longitudinal coordinate span of each sampled snapshot. The main experiment includes nine algorithms: CFCO, Singleton, MEC-only, Random, SNR-Greedy, Cap-Greedy, Matching, BR-LS, and SA.

The baselines are defined as follows. Singleton executes all tasks locally. MEC-only offloads all vehicles with feasible MEC links under the congested $n_{\text{mec}} = N$ rate model to the RSU and keeps the remaining vehicles local. Random forms feasible D2D clusters randomly in position order. SNR-Greedy and Cap-Greedy both use MEC-preference initialization and select feasible D2D heads according to CPU frequency while accepting only positive merge gains; the two differ in the member traversal order. Matching is a conservative head-matching baseline in which MEC-preferred vehicles are excluded from D2D participation, while the remaining vehicles select feasible D2D

Table 2 Main experimental results (mean and 95% confidence interval).

Algorithm	SW	D_{avg} (s)	$D_{\max}$ (s)	E_{avg} (J)	Succ.	Jain	Time (s)
CFCO	**481.28**	**0.225**	**0.633**	0.935	**0.998**	**0.716**	9.17
	[439.86,522.69]	[0.217,0.233]	[0.563,0.702]	[0.894,0.976]	[0.995,1.002]	[0.688,0.744]	[8.59,9.75]
Singleton	378.05	0.318	0.958	1.084	0.967	0.606	0.75
	[339.12,416.97]	[0.306,0.331]	[0.902,1.014]	[1.029,1.139]	[0.957,0.976]	[0.569,0.644]	[0.73,0.77]
MEC-only	281.45	0.455	0.923	**0.409**	0.940	0.505	0.76
	[241.04,321.87]	[0.432,0.478]	[0.902,0.945]	[0.381,0.436]	[0.919,0.961]	[0.445,0.566]	[0.74,0.79]
Random	263.10	0.415	0.816	1.415	0.973	0.410	0.13
	[226.38,299.82]	[0.391,0.438]	[0.707,0.925]	[1.326,1.503]	[0.947,0.999]	[0.347,0.473]	[0.11,0.14]
SNR-Greedy	398.27	0.256	0.638	1.435	**0.998**	0.571	0.35
	[358.57,437.96]	[0.248,0.264]	[0.568,0.708]	[1.370,1.500]	[0.995,1.002]	[0.533,0.609]	[0.32,0.39]
Cap-Greedy	397.83	0.258	0.635	1.428	**0.998**	0.576	0.37
	[358.61,437.04]	[0.249,0.266]	[0.565,0.706]	[1.363,1.493]	[0.995,1.002]	[0.535,0.618]	[0.34,0.40]
Matching	400.70	0.295	0.810	1.044	0.990	0.646	0.77
	[363.68,437.72]	[0.284,0.306]	[0.737,0.883]	[0.990,1.098]	[0.978,1.002]	[0.611,0.680]	[0.73,0.80]
BR-LS	480.24	0.230	0.637	0.910	**0.998**	0.713	110.85
	[439.19,521.29]	[0.222,0.237]	[0.570,0.704]	[0.867,0.952]	[0.995,1.002]	[0.687,0.739]	[108.11,113.60]
SA	428.88	0.284	0.800	0.819	0.987	0.683	11.80
	[391.12,466.65]	[0.274,0.293]	[0.695,0.906]	[0.741,0.897]	[0.973,1.000]	[0.650,0.716]	[11.51,12.08]

heads according to individual welfare and positive merge gain. SA is a mode-aware simulated annealing baseline that searches over random move, merge, split, and MEC-toggle actions. BR-LS is a deterministic best-response local-search baseline that, at each step, applies the single MEC-toggle or coalition-migration move yielding the largest positive increase in mode-aware social welfare; it serves as a strong non-CFCO comparator that is independent of CFCO's merge/split flow.

6 Experimental Results and Analysis

6.1 Main results

Table 2 shows that CFCO achieves the best social welfare, average delay, task success ratio, and Jain fairness among all evaluated algorithms. Its social welfare is 481.28, improving over BR-LS (480.24) by 0.2%, over SA (428.88) by 12.2%, over Matching (400.70) by 20.1%, and over MEC-only (281.45) by 71.0%. The average delay of CFCO is 0.225 s, which is 1.9% lower than BR-LS, 20.7% lower than SA, and 50.4% lower than MEC-only. The success ratio of CFCO is 0.998, which is 0.058 higher than MEC-only and 0.031 higher than Singleton.

It should be noted that CFCO is not the best on every single metric. MEC-only has the lowest average energy (0.409 J), because the feasible MEC-offloading vehicles use the RSU transmission-energy model and the remaining vehicles fall back to local execution. SA also has lower average energy (0.819 J) than CFCO. Thus, CFCO does not simply pursue the lowest energy or the minimum maximum delay. Instead, it selects a tradeoff under the social-welfare objective jointly defined by early-completion benefit, energy cost, and delay penalty. This objective explains why CFCO obtains a more balanced performance in average delay, success ratio, and fairness.

A notable observation is the strong performance of BR-LS, which achieves social welfare 480.24, essentially matching CFCO. However, BR-LS requires an average runtime of 110.85 s, whereas CFCO converges in only 9.17 s. In other words, CFCO

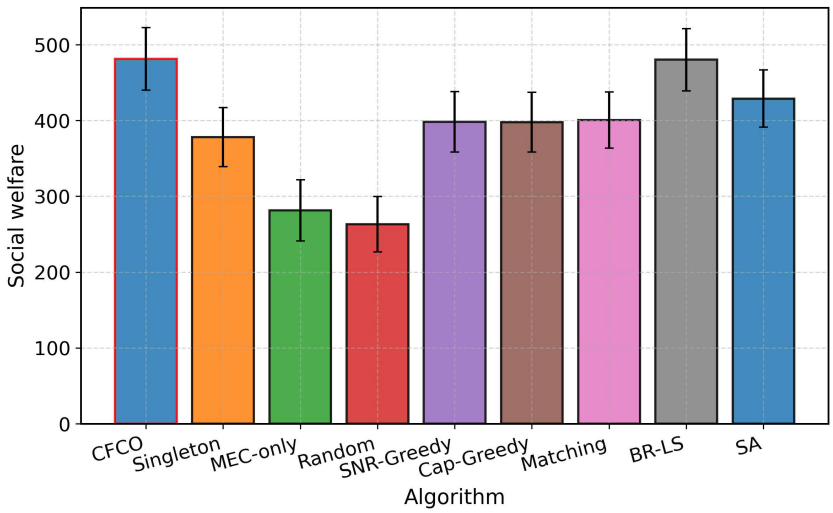


Fig. 1 Social welfare comparison among algorithms.

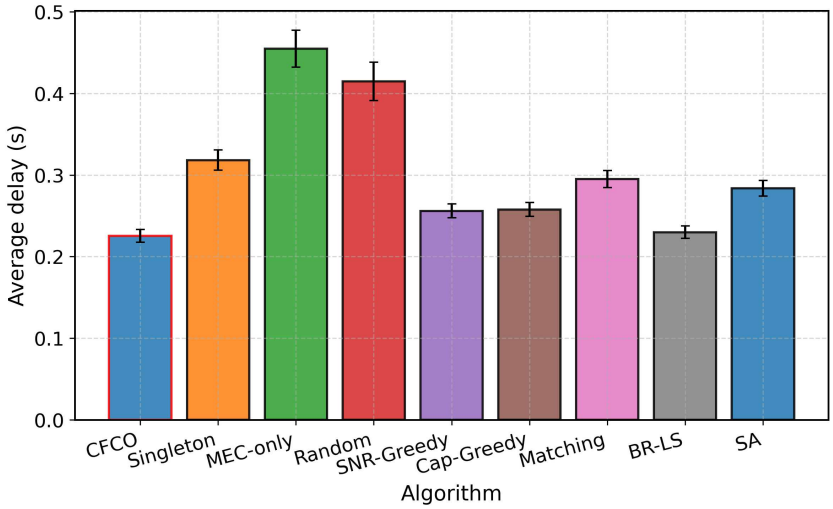


Fig. 2 Average task delay comparison among algorithms.

obtains comparable solution quality more than twelve times faster than the exhaustive best-response local search. This efficiency advantage is important for online VEC decision making.

6.2 Statistical significance and optimality gap

To validate the reliability of the observed differences, we conduct paired Wilcoxon signed-rank tests comparing CFCO with each baseline over the ten random seeds.

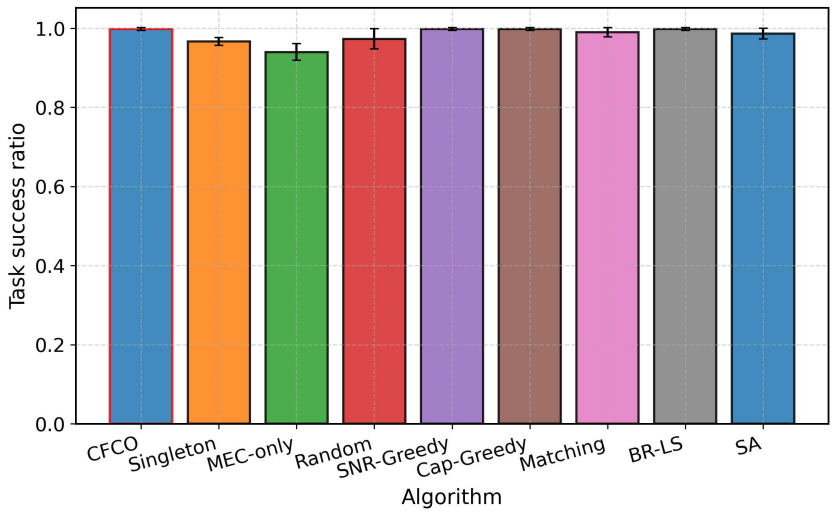


Fig. 3 Task success ratio comparison among algorithms.

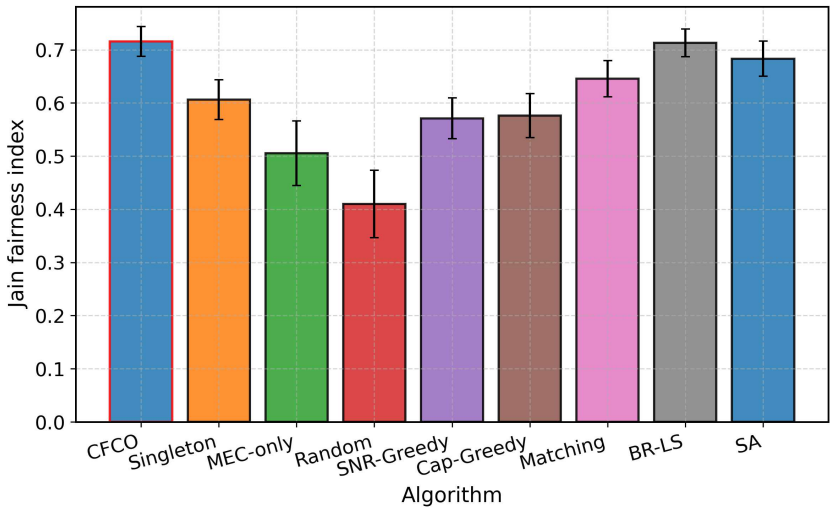


Fig. 4 Jain fairness comparison among algorithms.

Table 3 reports the p-values for social welfare, average delay, success ratio, and Jain fairness. CFCO is statistically better than all baselines in social welfare and average delay at the 0.05 significance level. The advantage over BR-LS is also significant ($p = 0.024$ for social welfare and $p = 0.005$ for average delay), confirming that the small mean difference is consistent across seeds rather than being driven by a single run. For success ratio, the differences against SNR-Greedy, Cap-Greedy, and BR-LS are not significant because these algorithms reach the same mean success ratio (0.998). For

Table 3 Paired Wilcoxon signed-rank tests comparing CFCO with each baseline.

Baseline	SW p	Delay p	Success p	Jain p
Singleton	9.77×10^{-4}	9.77×10^{-4}	9.77×10^{-4}	9.77×10^{-4}
MEC-only	9.77×10^{-4}	9.77×10^{-4}	9.77×10^{-4}	9.77×10^{-4}
Random	9.77×10^{-4}	9.77×10^{-4}	1.94×10^{-2}	9.77×10^{-4}
SNR-Greedy	9.77×10^{-4}	9.77×10^{-4}	5.00×10^{-1}	9.77×10^{-4}
Cap-Greedy	9.77×10^{-4}	9.77×10^{-4}	5.00×10^{-1}	9.77×10^{-4}
Matching	9.77×10^{-4}	9.77×10^{-4}	3.77×10^{-2}	9.77×10^{-4}
BR-LS	2.44×10^{-2}	4.88×10^{-3}	5.00×10^{-1}	1.88×10^{-1}
SA	9.77×10^{-4}	9.77×10^{-4}	3.77×10^{-2}	1.95×10^{-3}

Table 4 Small-scale exact optimality gap against Optimal-ILP ($N = 12$, ten seeds).

Algorithm	Algorithm SW	Optimal SW	Gap (%)	95% CI
CFCO	112.37	112.40	0.03	$[-0.02, 0.08]$
Singleton	69.70	112.40	38.25	$[31.89, 44.62]$
MEC-only	110.88	112.40	1.35	$[0.84, 1.86]$
Random	53.58	112.40	52.49	$[42.75, 62.23]$
SNR-Greedy	19.00	112.40	82.58	$[75.86, 89.29]$
Cap-Greedy	19.00	112.40	82.58	$[75.86, 89.29]$
Matching	110.99	112.40	1.26	$[0.68, 1.85]$
BR-LS	112.36	112.40	0.03	$[-0.01, 0.08]$
SA	112.31	112.40	0.06	$[-0.16, 0.27]$

Jain fairness, the difference against BR-LS is also not significant ($p = 0.188$), which is consistent with their nearly identical fairness values.

To quantify how close CFCO is to the global optimum, we solve an exact integer linear programming (ILP) formulation for $N = 12$ over ten seeds. Table 4 summarizes the optimality gap for each algorithm. CFCO achieves an average optimality gap of only 0.03% (95% CI $[-0.02, 0.08]$), which is statistically indistinguishable from zero and much smaller than all baselines except BR-LS. This result provides strong evidence that the coalition-formation heuristic in CFCO locates near-optimal partitions in the tractable regime.

6.3 Mode selection, delay decomposition, and runtime

Figure 5 shows the newly generated three-panel plot of mode ratios, delay decomposition, and runtime based on the Excel result data. In the final CFCO decision, 20.5% of vehicles execute locally, 22.5% are offloaded to MEC, and 57.0% join D2D collaborative coalitions. The high D2D ratio reflects the NGSIM trace topology: vehicles are spatially clustered on a real road, which creates favorable short-range D2D links and makes collaborative computing attractive. In contrast, MEC-only sends 66.8% of vehicles to MEC and keeps 33.2% local; its lower welfare and higher delay show that feasible V2I access alone is insufficient when shared bandwidth and finite MEC

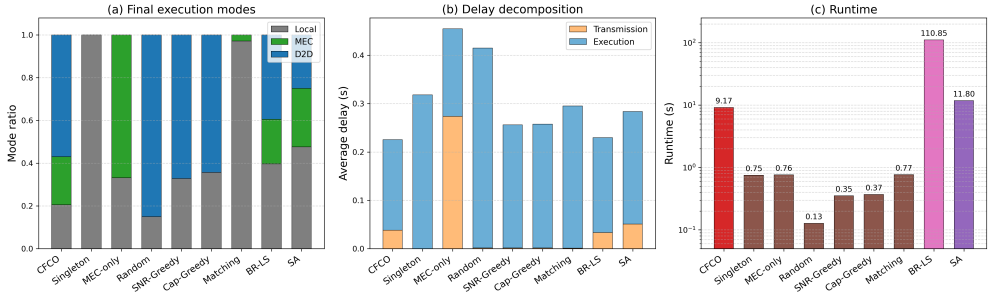


Fig. 5 Newly generated result figure: final execution modes, delay decomposition, and runtime.

cores create congestion. Random assigns a very large fraction of vehicles to D2D coalitions (85.0%), but because random collaboration does not adequately consider head capacity or welfare gain, it obtains the lowest social welfare.

The delay decomposition shows that CFCO has an average transmission delay of only 0.038 s and an average execution delay of 0.187 s. This indicates that CFCO does not blindly increase transmission by choosing D2D or MEC; instead, it exploits external computation only when the transmission cost is controlled. In terms of runtime, the current Python implementation of CFCO takes 9.17 s on average. It is higher than the static baselines (all below 0.8 s) and the Random baseline (0.13 s), but it is 22.3% faster than SA (11.80 s) and more than twelve times faster than BR-LS (110.85 s), while achieving comparable or better social welfare. The overhead mainly comes from repeated global welfare evaluation and local migration search. It can be reduced in future work by caching coalition welfare, incremental updates, and parallel candidate evaluation.

6.4 Convergence and stability

Figure 6 shows the social welfare trajectory over iterations, averaged across the ten random seeds with shaded error regions. CFCO starts from the initial MEC-preference assignment and quickly improves through merge/split operations, converging after approximately 9–12 recorded iterations. SA also improves gradually, but its final value remains lower than CFCO on average. The shaded region shows that CFCO is robust across seeds: although the absolute social welfare varies with the NGSIM snapshot, the convergence behavior is consistent.

For stability, the individual-stability ratio of CFCO is 0.765 across the ten seeds, compared with 0.850 for SA, 0.805 for SNR-Greedy, 0.807 for Cap-Greedy, 0.825 for BR-LS, and 0.437 for Random. Singleton, MEC-only, and Matching have stability ratios of 1.000 under this D2D-only check mainly because their reported partitions are close to singleton structures; the check evaluates unilateral coalition deviations in the final partition and does not re-optimize MEC assignment. Therefore, stability must be interpreted together with welfare, delay, and success ratio. The lower stability ratio of CFCO compared with SA and BR-LS is acceptable because CFCO achieves

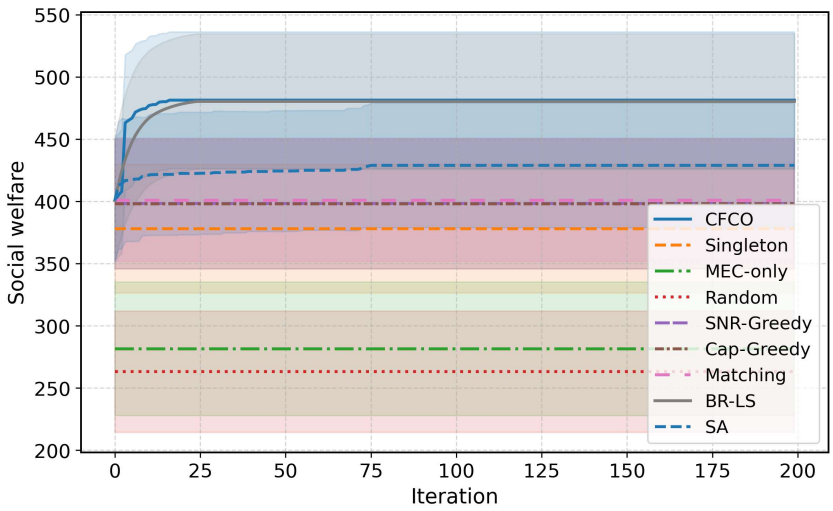


Fig. 6 Social welfare convergence curves.

substantially higher social welfare than SA and nearly the same social welfare as BR-LS with much lower runtime.

6.5 Delay-energy relationship and delay distribution

Figure 7 shows the relationship between average delay and average energy. MEC-only has the lowest average energy, but its average delay and task success ratio are inferior to CFCO. Random has the highest average delay and average energy 1.415 J, indicating that random coalition formation may select poor collaborators. CFCO lies in a low-delay and moderate-energy region, reflecting the tradeoff induced by the welfare objective. BR-LS occupies a similar region, confirming that both structured coalition formation and best-response local search identify high-quality delay-energy tradeoffs.

Figure 8 gives the empirical CDF of task delays. The mean 95th-percentile delay of CFCO is 0.481 s, lower than SA (0.570 s), Matching (0.598 s), Singleton (0.675 s), Random (0.663 s), and MEC-only (0.791 s). SNR-Greedy and Cap-Greedy have slightly higher 95th-percentile delays than CFCO (0.519 s for both), while their social welfare and fairness remain much lower. Together with the average-delay result, this indicates that CFCO reduces average response time while avoiding a large number of tasks falling into a long-delay region.

6.6 Coalition size and cooperation ratio

Figure 9 shows the final coalition-size distribution of CFCO. Because the D2D range is 200 m and NGSIM vehicles form natural platoons on the road, CFCO tends to form several small collaborative coalitions rather than one large coalition. Figure 10 further shows the offloading/cooperation ratio. CFCO uses D2D collaboration for 57.0% of vehicles, which is higher than SA's 25.0% and reflects CFCO's stronger ability to exploit favorable D2D opportunities in the trace-driven topology.

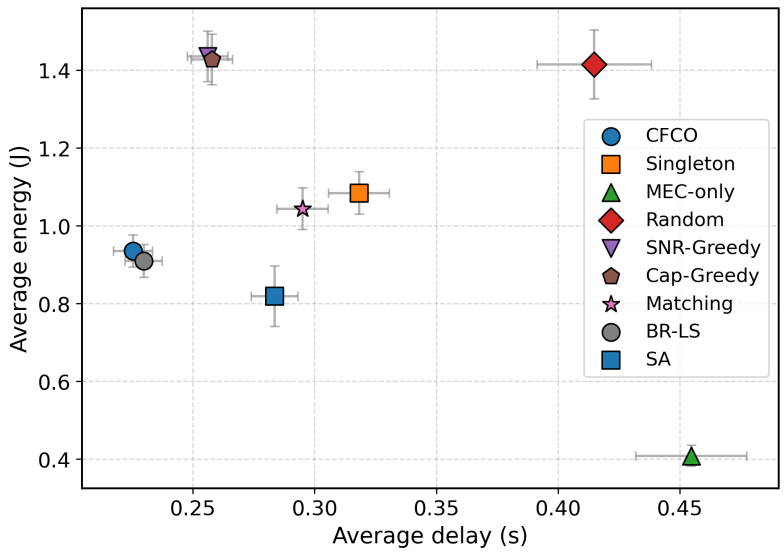


Fig. 7 Average delay versus average energy.

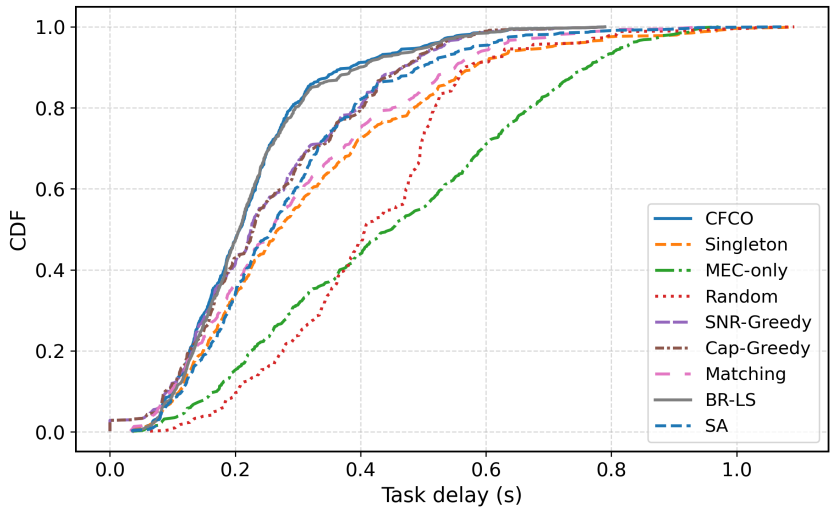


Fig. 8 Empirical CDF of task delays for all algorithms.

6.7 Parameter sensitivity

Figure 11 shows four parameter sweeps: local CPU frequency, D2D bandwidth, MEC CPU frequency, and number of vehicles. The scenario sweeps use five independent random seeds, and error bars indicate 95% confidence intervals.

In the local CPU frequency sweep, CFCO's average social welfare per vehicle increases from 4.97 at 0.6 GHz to a peak of 8.04 at 2.2 GHz and then decreases slightly

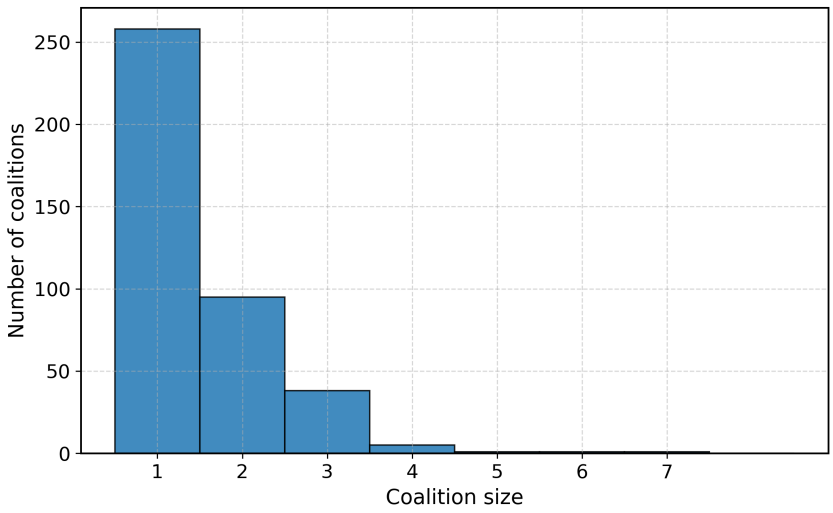


Fig. 9 Coalition-size distribution of CFCO.

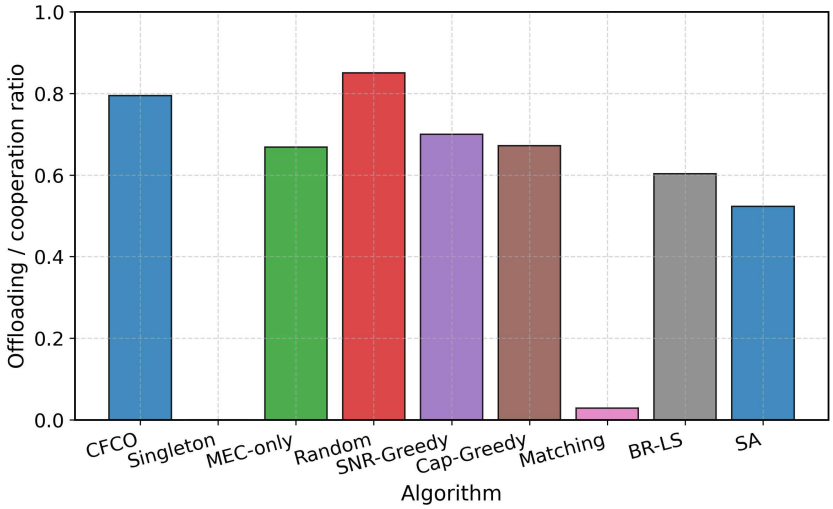


Fig. 10 Offloading/cooperation ratio of different algorithms.

to 7.46 at 3.0 GHz. The decrease is related to the energy model: local energy contains the quadratic term $\kappa f_i^2 D_i$. Once the delay benefit of increasing CPU frequency becomes saturated for some tasks, the quadratic energy term can reduce net welfare. Nevertheless, CFCO remains above all baselines at every frequency point.

In the D2D bandwidth sweep, CFCO's average social welfare per vehicle is relatively flat, increasing slightly from 7.85 at 1 MHz to 7.90 at 10 MHz. This indicates that, under the current NGSIM topology, CFCO already exploits D2D collaboration effectively even at low bandwidth, and additional bandwidth yields diminishing

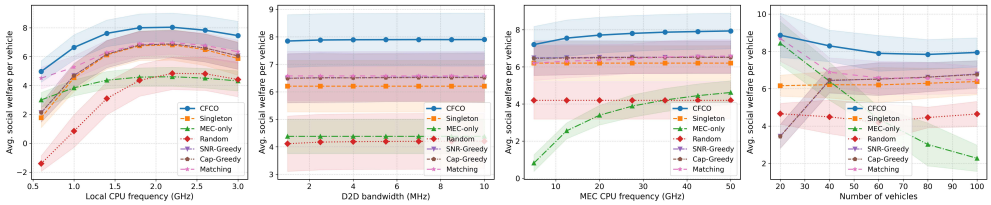


Fig. 11 Parameter sensitivity analysis.

returns. MEC-only and Singleton do not depend on D2D bandwidth and therefore remain flat.

In the MEC CPU frequency sweep, CFCO's average social welfare per vehicle increases steadily from 7.20 at 5 GHz to 7.94 at 50 GHz. MEC-only also improves from 0.82 to 4.62 as MEC capacity increases, but it remains well below CFCO because many vehicles still lack feasible V2I links. This shows that CFCO can exploit stronger MEC resources while preserving local and D2D paths when MEC is unavailable or congested.

In the vehicle-number sweep, CFCO obtains average social welfare per vehicle of 8.87, 8.29, 7.90, 7.84, and 7.94 for $N = 20, 40, 60, 80, 100$, respectively, and is above the baselines at all points. The decreasing trend reflects increased contention for MEC and D2D resources, while the slight rebound at $N = 100$ remains within the confidence interval.

6.8 Impact of V2I reliability

Figure 12 shows the average social welfare per vehicle when the V2I availability probability p_v increases from 0.1 to 1.0. CFCO increases from 7.13 to 8.17, while MEC-only decreases from 6.73 to 0.92. The decrease in MEC-only occurs because, when p_v is high, more vehicles attempt to offload simultaneously, which intensifies V2I bandwidth sharing and MEC core congestion. CFCO avoids this congestion by dynamically switching vehicles to local or D2D modes. Figure 13 further shows average delay under different V2I reliability values. CFCO keeps lower average delay than Singleton and Random at all reliability points, lower delay than MEC-only for $p_v \geq 0.2$, and only slightly higher delay than MEC-only at $p_v = 0.1$, where few vehicles compete for MEC resources.

6.9 Discussion and limitations

The experiments show that CFCO achieves higher social welfare and lower average delay in the NGSIM trace-driven configuration. Four limitations remain. First, although we use real NGSIM snapshots, each experimental run samples a single static snapshot rather than a continuous mobility trace; vehicles do not move between successive decision intervals. Extending the evaluation to time-varying vehicle trajectories is an important direction for future work. Second, the current D2D coalition computation model uses a parallel/batch-processing abstraction at the head, where execution delay is determined by the maximum task workload. If a real system can only process coalition tasks strictly sequentially, the model and results should be rerun. Third, the

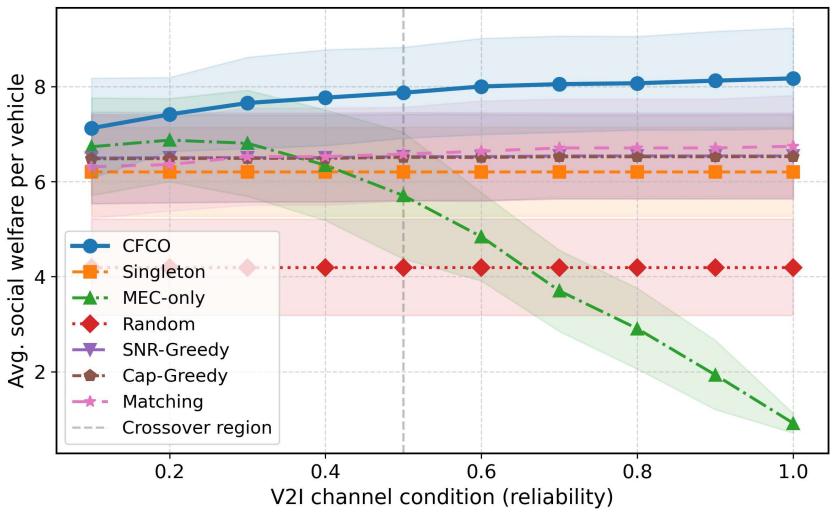


Fig. 12 Average social welfare per vehicle under different V2I reliability values.

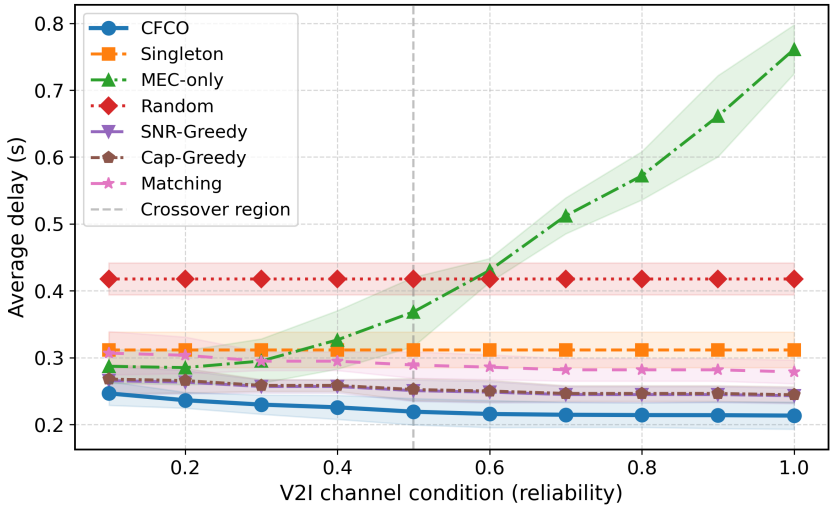


Fig. 13 Average task delay under different V2I reliability values.

strong performance of BR-LS suggests that much of the gain comes from the welfare objective and the neighborhood structure; future work can add even stronger baselines, such as auction or market mechanisms, to further differentiate CFCO. Fourth, the reported social welfare values are not directly comparable with those in the previous synthetic-topology version because the NGSIM traces introduce realistic spatial clustering, measured velocities, and velocity-dependent deadlines; the improvement should be interpreted as the combined effect of the trace-driven setting and the algorithm, not as a purely algorithmic advance.

7 Conclusion

This paper proposed CFCO, a social-welfare-maximizing collaborative coalition formation algorithm for task offloading in VEC. CFCO unifies local computing, D2D collaboration, and MEC offloading, and monotonically improves social welfare in a finite state space through merge, split, mode refinement, and local migration operations. In this revised study, we evaluated CFCO on real-world NGSIM vehicle trajectories and reported statistically robust results over ten independent random seeds. Experimental results generated by the actual code show that, in the default $N = 60$ scenario, CFCO achieves social welfare 481.28, average delay 0.225 s, task success ratio 0.998, and Jain fairness index 0.716, outperforming eight baselines overall. A small-scale exact ILP study confirms that CFCO is within 0.03% of the optimal solution. Parameter sweeps and V2I reliability experiments further indicate that CFCO adapts well to changing infrastructure reliability and system resources. Future work will extend this study with continuous mobility traces, multi-RSU cooperation, task dependencies, sequential coalition-head processing, and incentive-compatible payment mechanisms.

Acknowledgements. We thank Google Gemini for assistance with English language polishing and Moonshot AI Kimi for support in experimental parameter tuning. AI tools were used solely to assist with language editing and technical optimization; the authors retain full responsibility for the content and conclusions of this work.

Funding. This research was funded by the research team of Aba Teachers University (grant number AS-KCTD2026-1), the Sichuan Provincial College Students' Innovation and Entrepreneurship Training Program (Grant No. To be assigned).

Data availability. To facilitate future research and ensure reproducibility, the complete implementation code and the NGSIM trace data used in this study will be released as an open-source repository on GitHub during the manuscript revision process.

Declarations. The authors declare that they have no competing interests.

References

[1] Liu, X., Li, W., Cao, B., Wang, S., Lyu, Z.: Large-scale multiobjective vehicle task offloading optimization based on cloud-edge-end collaboration for 6g enabled transport systems. IEEE Transactions on Intelligent Transportation Systems (2025). Early access

[2] Li, P., Xiao, Z., Gao, H., Wang, X., Wang, Y.: Reinforcement learning based edge-end collaboration for multi-task scheduling in 6g enabled intelligent autonomous transport systems. IEEE Transactions on Intelligent Transportation Systems (2025). Early access

[3] Wu, J., Tang, M., Jiang, C., Gao, L., Cao, B.: Cloud-edge-end collaborative task offloading in vehicular edge networks: A multilayer deep reinforcement learning approach. IEEE Internet of Things Journal **11**(22), 36272–36290 (2024)

[4] Wang, S., Tan, X., Pan, Y., Li, O., Sun, M., Zheng, Q., Yang, J.: Multi-device adaptive computing offloading for real-time video analytics in mec-enabled vehicular networks. IEEE Transactions on Vehicular Technology (2025) https://doi.org/10.1109/TVT.2025.3577032 . Early access

[5] Shen, R., Gao, M., Li, W., Li, Y.: Task offloading oriented cluster generation scheme in vehicular networks. IEEE Transactions on Vehicular Technology **73**(6), 9142–9146 (2024)

[6] Zhao, J., Li, Q., Gong, Y., Zhang, K.: Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks. IEEE Transactions on Vehicular Technology **68**(8), 7944–7956 (2019)

[7] Xue, Z., Liu, C., Wen, F., Han, G.: Joint optimization of task offloading and resource allocation for cooperative perception in vehicular edge computing systems. IEEE Transactions on Vehicular Technology (2025). Early access

[8] Mao, Y., You, C., Zhang, J., Huang, K., Letaief, K.B.: A survey on mobile edge computing: The communication perspective. IEEE Communications Surveys & Tutorials **19**(4), 2322–2358 (2017)

[9] Mach, P., Becvar, Z.: Mobile edge computing: A survey on architecture and computation offloading. IEEE Communications Surveys & Tutorials **19**(3), 1628–1656 (2017)

[10] Hui, Y., Su, Z., Luan, T.H., Li, C., Mao, G., Wu, W.: A game theoretic scheme for collaborative vehicular task offloading in 5g hetnets. IEEE Transactions on Vehicular Technology **69**(12), 16044–16056 (2020)

[11] Liu, L., Zhao, Z., Feng, J., Xu, F., Zhang, Y., Pei, Q., Xiao, M.: Distributed collaborative computing for task completion rate maximization in vehicular edge computing. IEEE Transactions on Intelligent Transportation Systems (2025). Early access

[12] Huang, W., Zhao, Z., Min, G., Wang, Y., Chang, Z.: Mmto: Multi-vehicle multi-hop task offloading in mec-enabled vehicular networks. IEEE Transactions on Mobile Computing (2025) https://doi.org/10.1109/TMC.2025.3576154 . Early access

[13] Zhang, Y., Wang, W., Huang, R., Zhou, J., Xu, Y., Ren, J., Zhang, Y.: Collaborative edge and cloud computing: Optimal configuration and computation management. IEEE Transactions on Mobile Computing (2025) https://doi.org/10.1109/TMC.2025.3584524 . Early access

[14] Bi, S., Zhang, Y.-J.A.: Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading. IEEE Transactions on Wireless Communications **17**(6), 4177–4190 (2018)

[15] Deng, M., Liang, W., Qiu, B., Xie, X.: Federated deep reinforcement learning-based task offloading strategy with game theory in vehicular edge computing. Physical Communication **72**, 102802 (2025) https://doi.org/10.1016/j.phycom.2025.102802

[16] Pu, X., Lei, T., Wen, W., Feng, W., Wang, Z., Chen, Q., Jin, S.: Incentive mechanism and resource allocation for collaborative task offloading in energy-efficient mobile edge computing. IEEE Transactions on Vehicular Technology **72**(10), 13775–13780 (2023)

[17] Yu, J., Liu, S., Zou, Y., Wang, G., Hu, C.: Auction theory and game theory based pricing of edge computing resources: A survey. IEEE Internet of Things Journal **12**(16), 32394–32418 (2025)

[18] Liu, J., Wei, J., Luo, R., Yuan, G., Liu, J., Tu, X.: Computation offloading in edge computing for internet of vehicles via game theory. Computers, Materials & Continua **81**(1), 1–25 (2024)

[19] Wu, L., Sun, P., Wang, Z., Chen, H., Luo, J., Zuo, Y., Yang, Y.: A socially optimal marketplace for splittable task offloading in multi-user multi-server edge computing networks. IEEE Transactions on Networking **34**, 1451–1463 (2026)

[20] Liu, X., Liu, J., Li, W.: Truthful mechanism for computation offloading and resource-sharing in vehicle computing. IEEE Transactions on Intelligent Transportation Systems **27**(6), 6800–6815 (2026)

[21] Wu, Y., Fang, Z., Huang, J., Guo, L., Wu, J.: Vehicle coalition-based incentive algorithm for model deployment and task offloading. IEEE Transactions on Network and Service Management **23**, 3351–3364 (2026)

[22] Federal Highway Administration: NGSIM next generation simulation (NGSIM) vehicle trajectories and supporting data. Technical report, U.S. Department of Transportation (2006). Accessed: 2025. https://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm

[23] Saad, W., Han, Z., Debbah, M., Hjorungnes, A., Basar, T.: Coalition formation games for distributed cooperation among roadside units in vehicular networks. IEEE Journal on Selected Areas in Communications **29**(1), 48–60 (2011)

[24] Bogomolnaia, A., Jackson, M.O.: The stability of hedonic coalition structures. Games and Economic Behavior **38**(2), 201–230 (2002)

[25] Jain, R.K., Chiu, D.-M.W., Hawe, W.R.: A quantitative measure of fairness

and discrimination for resource allocation in shared computer systems. Technical Report DEC-TR-301, Digital Equipment Corporation (1984)

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- plotdata.xlsx
- log.txt