Joint Optimization of Cache Allocation and Content Placement in Urban Vehicular Networks

Tuo Liu, Sheng Zhou, Zhisheng Niu
Beijing National Research Center for Information Science and Technology
Department of Electronic Engineering, Tsinghua University, Beijing 100084, China
Email: t-liu16@mails.tsinghua.edu.cn, {sheng.zhou, niuzhs}@tsinghua.edu.cn

***Abstract*—Distributing popular contents, e.g., high precision digital maps and latest road conditions, through roadside units (RSUs) is a promising way to provide better driving safety and support for autonomous driving. Successful content download probability can be improved by allocating cache to RSUs and caching popular contents therein. In this paper, we consider a joint cache allocation and content placement problem in vehicular networks to maximize the overall successful content download probability, exploiting the moving information of vehicles. We first prove that the original problem is NP-hard, and then propose a low-complexity approximate algorithm which performs within a bounded gap (a multiplicity factor of $1-1/e$) to the optimum. The expression of the cache size allocated to each RSU is derived for the proposed algorithm when the content popularity obeys Zipf distribution. Extensive numerical experiments show that, the proposed strategy can significantly increase the successful content download probability as compared to existing solutions.**

I. Introduction

Data download via vehicle-to-infrastructure (V2I) in vehicular networks has been widely studied recently [1] [2]. Types of contents range from advertisements, road conditions, digital maps and multimedia contents. Once the vehicle enters the coverage of a roadside unit (RSU) and connects successfully to the RSU, it can request contents of interests. Then the RSU fetches the content from the Internet via the backhaul. However, both the connection duration when vehicles are moving and backhaul bandwidth are limited in vehicular networks, resulting in possible content download failure [3]. A promising solution is to allocate some cache for caching at RSUs, and thus each RSU can proactively fetch some popular contents and stores them in the cache. Notice that, cache size is always limited compared with the large volume of active contents, and vehicles can probably connect to multiple RSUs along their trajectories. Therefore, content placement strategies should be carefully designed and jointly optimized with cache allocation among RSUs, exploiting the moving information of vehicles.

Most existing works merely focus on content placement in vehicular networks. When a user requests for some content, the amount of downloaded data is limited by both the total cached data and its connection time with the associated RSU. The authors of [1] propose a heuristic content placement algorithm to maximize the file retrieval probability within a deadline and demonstrate its effectiveness by experiments. Reference [4] considers minimizing the content download time by caching at RSUs on highway. Reference [5] formulates the content distribution problem in the V2I manner into an integer linear programming (ILP) problem. Numerically solving the problem reveals the impact of various parameters, including the variance of connection time and vehicle speed, on the successful download probability. Reference [6] focuses on content placement in the vehicular network by addressing different wireless channel conditions of multiple RSUs.

These works take careful considerations on the strategy of distributing contents, while they assume cache size of every RSU as equal, or given. Some works about cache organization have been proposed in cellular networks [7] [8]. However, cache size optimization in vehicular networks has received little attentions which is different from that in cellular networks. First, vehicles move very fast while their trajectories are limited by roads, thus following certain patterns. Second, data traffic have great disparity on different roads. Intuitively, those RSUs on roads with heavier traffic should be equipped with larger cache. To this end, cache size should be optimized subject to a total cache budget, in order to achieve higher successful content download probability. Regarding this, we formulate a joint cache allocation and content placement problem in urban vehicular networks. Proving that the original problem is NP-hard, we propose a greedy cache allocation algorithm and it is proved to achieve $(1-\frac{1}{e})$ approximation of the optimum. The closed-form expression of the cache size when using our algorithm is derived. In comparison with other existing algorithms via numerical experiments, our algorithm can achieve 10% gain in terms of successful download probability, and we study both analytically and numerically on how different factors, for example the content popularity, successful transmission probability and cache budget, impact the successful content download probability.

The rest of this paper is organized as follows: Section II presents our system model and problem formulation. Section III explains the hardness of the original problem and proposes an approximate algorithm, with the resultant cache size expression. Section IV shows the performance of the proposed algorithms and compares it with other existing algorithms via extensive numerical experiments. Section V draws conclusions.

II. Problem Formulation

A typical architecture of urban vehicular network is illustrated in Fig. 1. We use similar notations in [9] and first define

a term "contact pattern" for better illustration.

Definition 1: A *contact pattern* is a set of RSU contact along the trajectory of a vehicle traveling through the network. Three different trajectories are shown in 1 and the corresponding contact patterns are $\{\text{RSU}_1, \text{RSU}_2, \text{RSU}_3, \text{RSU}_4\}$, $\{\text{RSU}_5, \text{RSU}_6, \text{RSU}_8\}$ and $\{\text{RSU}_1, \text{RSU}_6, \text{RSU}_7, \text{RSU}_8, \text{RSU}_9\}$ respectively.

Suppose there are a set of M RSUs $\{1, 2, \ldots, M\}$ deployed at the intersections. Denote the number of all possible contact patterns by K. And denote the set of RSUs of the k-th pattern and the set of patterns containing RSU m by $\mathcal{N}_k$ and $\mathcal{G}_m$ respectively. Then define l_k as the probability of the occurrence of contact pattern k among all possible contact patterns within the network, which depends on road traffic, RSU topology, etc. We assume that l_k can be learned or estimated beforehand considering the GPS data can be utilized to identify the actual trajectory of each vehicle.

The successful wireless transmission is not guaranteed under the moving conditions due to the stochastic characteristic of the wireless channel. Suppose a vehicle can successfully download a content from the cache of RSU m with probability q_m, which is defined as successful transmission probability. Though the successful transmission probability depends on various factors such as the location of RSUs, the speed of vehicles and the wireless capacity, etc, it can be learned through large amount of historical data.

Suppose there are a set of N contents of equal size $\{f_1, f_2, \ldots, f_N\}$ in the library and, without loss of generality, $f_1, f_2, \ldots, f_N$ are indexed in the descending order by their request probability which obeys Zipf distribution [10] [11], i.e., $P_n = n^{-a} / \sum_{k=1}^{N} k^{-a}$. Suppose there is a total cache budget C normalized by the content size, i.e., it can store at most C contents.[1]. The goal is to maximize the successful download probability among all request from all possible contact patterns and requests. Instead of assigning the cache uniformly to each RSU and simply designing content placement strategy, we aim to find the optimal cache allocation strategy as well as the content placement strategy by taking the request probability of contents, the probability of contact pattern and the wireless characteristics into considerations.

Denote the size of the cache allocated to RSU m by B_m and denote the set of contents stored in the cache of RSU m by $\mathcal{F}_m$. Accordingly, the problem is formulated as follows.

$$\max_{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_M} \quad 1 - \sum_{k=1}^{K} l_k \sum_{j=1}^{N} P_j \prod_{m \in \mathcal{N}_k} \left(1 - q_m \mathbb{1}_{\{f_j \in \mathcal{F}_m\}}\right) \quad (1)$$

$$\text{s.t.} |\mathcal{F}_m| <= B_m, m \in \{1, 2, \ldots, M\} \quad (2)$$

$$\sum_{m=1}^{M} B_m = C, \quad (3)$$

where $\mathbb{1}_{\{x\}}$ is an indicator function that equals 1 when x is true, and 0 otherwise. Inequality (2) denotes that contents stored in the cache of any RSU should not surpass its cache size. Equation (3) shows the constraint of the cache budget, i.e., the size of cache allocated to all RSUs adds up to C. The optimization variables in the above problem reflect the size of cache allocated to each RSU B_m and the combination of cached contents $\mathcal{F}_m$, therefore it is a joint optimization problem.

[1]The phrase "normalized cache size" refers to the cache size normalized by the content size in the rest of this work if without specification, e.g., if the normalized cache size equals 1000, then at most 1000 contents can be stored.

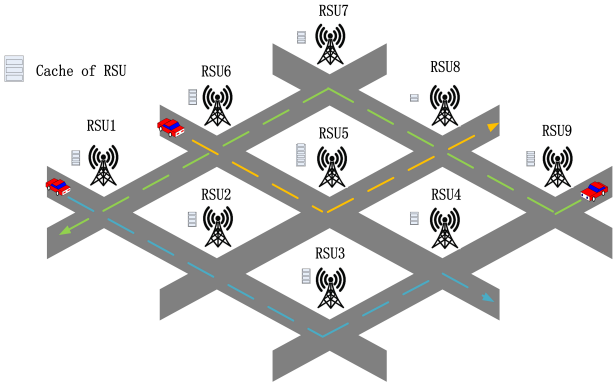


Fig. 1. An example of an urban vehicular network.

III. Proposed Algorithm and Performance Analysis

Before we start to design the cache allocation algorithm, first observe the complexity of solving the problem (1)-(3), as shown in the following theorem.

Theorem 1: The problem described by (1)-(3) is NP-hard.

Proof: Consider a vehicular network composed of M RSUs and there are exactly 2 RSUs in each contact pattern. It is a special case of the considered problem. Denote the total number of contact patterns by K and assume the probability of the occurrence of each contact pattern is $\frac{1}{K}$. Suppose that there is only one content in the library and the request probability is 1. Assume the wireless channel is perfect and the successful transmission probability between any RSU and the associated vehicle is 1. Under these assumptions, once an RSU is allocated with a unit of cache and stores the very content, vehicles passing by can download it with probability 1. Given a total cache budget C, we investigate the problem of finding a cache allocation strategy, so that the probability of successful download for this content in any contact pattern equals 1, which means that there exists at least 1 RSU allocated with a cache in any contact pattern.

We can transform the above problem to an equivalent vertex cover problem in a graph $G = (V, E)$, as shown in Fig 2. Vertices in the graph correspond to RSUs and two nodes on each edge correspond to contact patterns. Here $|V| = M$ and $|E| = K$. Denote by V_0 the set of vertices whose counterpart RSU is allocated with a unit of cache, then $|V_0| \leq C$ holds. Since the probability of any contact pattern is equally $\frac{1}{K}$, the condition that the successful download probability within the whole vehicular network equals 1 holds if and only if $\forall (u, v) \in E$, either $u \in V_0$ or $v \in V_0$. The vertex cover problem is a known NP-hard problem. Therefore, given an instance of a

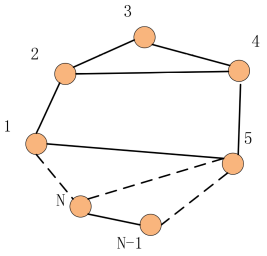


Fig. 2. An exemplary representation of a vehicular network.

vertex cover problem, it can be reducible to the problem in our model in polynomial time. In conclusion, the joint cache allocation and content placement problem is NP-hard. ■

Since it is a NP-hard problem and the number of variables can be very large, we resort to an approximate algorithm as illustrated in Algorithm 1.

Algorithm 1 A greedy cache allocation algorithm

1: **Input**: $C, M, N, K, P_i(i \in \{1,2,\ldots,N\})$,
2: $q_m(m \in \{1,2,\ldots,M\}), l_k(k \in \{1,2,\ldots,K\})$.
3: **initialization**: $\mathcal{F}_m = \emptyset(m \in \{1,2,\ldots,M\}), S = 0$.
4: **while** $S < \min\{C, MN\}$ **do**
5: **for** $m \in \{1,2,\ldots,M\}, j \in \{1,2,\ldots,N\}$ **do**
6: $V_{m,j} =$
7: $\sum_{k\in\mathcal{G}_m} l_k \prod_{t\in\mathcal{N}_k} (1 - q_t \mathbb{1}_{\{f_j \in \mathcal{F}_t\}}) P_j q_m \mathbb{1}_{\{f_j \notin \mathcal{F}_m\}}$.
8: **end for**
9: $(\hat{m}, \hat{j}) = \arg\max_{(m,j)} V_{m,j}$.
10: $D_{\hat{m},\hat{j}} = V_{\hat{m},\hat{j}}$.
11: $S \leftarrow S + 1$.
12: $\mathcal{F}_{\hat{m}} \leftarrow \mathcal{F}_{\hat{m}} \cup \{f_{\hat{j}}\}$.
13: **end while**

In the initialization stage of Algorithm 1 on line 2, each RSU has no cache, so the set of cached contents at each RSU is empty. Then in an iterative manner given from line 5 to line 8, the algorithm calculates the increment in successful download probability of distributing each content to each RSU and finds the content-RSU pair of the largest value. According to line 9 to line 11, one unit of cache is allocated to the corresponding RSU in the pair and the corresponding content is distributed to the cache. And $D_{\hat{m},\hat{j}}$ on line 10 records the increment in successful download probability due to the pair $(\hat{m}, \hat{j})$. The above operations repeat until the total cache budget is used up. The performance of the algorithm is guaranteed within a bounded gap to the optimum according to the following theorem.

Theorem 2: Algorithm 1 achieves at least $(1 - \frac{1}{e})$ approximation of the optimum.

Proof: Denote by A_{mj} the event that content f_j is cached in the cache of RSU m, and denote by $\mathcal{X}$ the feasible set of $A_{mj}, m \in \{1,2,\ldots,M\}, j \in \{1,2,\ldots,N\}$. We equivalently reformulate the optimization problem (1)-(3) into (4)-(6).

$$\max \quad g(\mathcal{X}) \tag{4}$$

$$\text{s.t. } g(\mathcal{X}) = 1 - \sum_{k=1}^{K} l_k \sum_{j=1}^{N} P_j \prod_{m\in\mathcal{N}_k} (1 - q_m \mathbb{1}_{\{A_{mj}\in\mathcal{X}\}}) \tag{5}$$

$$|\mathcal{X}| \le C. \tag{6}$$

Choose any two sets $\mathcal{X}_1, \mathcal{X}_2$ that satisfy $\mathcal{X}_1 \subseteq \mathcal{X}_2, |\mathcal{X}_2| \le C$, and choose one element $A_{m,j} \notin \mathcal{X}_2$, and $A_{m,j} \notin \mathcal{X}_1$. And

$$g(\mathcal{X}_1 \cup A_{mj}) - g(\mathcal{X}_1) = \sum_{k\in\mathcal{G}_m} l_k \prod_{t\in\mathcal{N}_k} (1 - q_t \mathbb{1}_{\{A_{kj}\in\mathcal{X}_1\}}) P_j q_m \ge 0. \tag{7}$$

And the following equation holds,

$$\begin{aligned}&(g(\mathcal{X}_1 \cup A_{mj}) - g(\mathcal{X}_1)) - (g(\mathcal{X}_2 \cup A_{mj}) - g(\mathcal{X}_2)) \\ &= \sum_{k\in\mathcal{G}_m} l_k \prod_{\substack{t\in\mathcal{N}_k \\ A_{tj}\in\mathcal{X}_1}} (1 - q_t)(1 - \prod_{\substack{h\in\mathcal{N}_k \\ A_{hj}\in\mathcal{X}_2 - \mathcal{X}_1}} (1 - q_h)) P_j q_m \\ &\ge 0. \end{aligned} \tag{8}$$

Therefore, the objective function is a submodular function according to the definition in [12], and the greedy property of the proposed algorithm guarantees $(1 - \frac{1}{e})$ approximation of the optimal algorithm. ■

The complexity of the algorithm above is $O(MN^2KC)$. The cost can still be very high especially when both the amount of contents and the cache budget are large. Meanwhile, the algorithm provides no insight into the critical factors that affect the cache allocation and content placement. Therefore we adopt further analysis on the algorithm and reveal following property.

Proposition 1: The result of Algorithm 1 indicates that although the size of cache allocated to RSUs may be different, for a given RSU $m, m \in \{1,2,\ldots,M\}$, it will store the most popular $|\mathcal{F}_m|$ contents in the cache after the implementation of Algorithm 1, i.e., $\mathcal{F}_m = \{f_1, f_2, \ldots, f_{|\mathcal{F}_m|}\}$.

Proof: Assume a virtual normalized cache budget C^* satisfying $C^* \ge MN$, then after Algorithm 1 completes, all contents will have been distributed to the cache of every RSU due to the abundance of cache budget. During the implementation of Algorithm 1, the content-RSU pairs on line 9 occur in certain order. For an arbitrary content f_j, we define a term "pair sequence" to simplify illustration.

Definition 2: a *pair sequence* of f_j is the sequence of content-RSU pairs picked out by line 9 of Algorithm 1 that contain f_j. The pairs are indexed according to their occurrence time during the implementation of Algorithm 1. Denote the pair sequence of f_j by $[(b_{j,1}, f_j), (b_{j,2}, f_j), \ldots, (b_{j,M}, f_j)]$. Then $b_{j,s}$ is the s-th RSU that Algorithm 1 distribute f_j to.

According to line 4-10 in Algorithm 1 and the definition above, $D_{b_{j,s},j}$ can be expressed as the following:

$$D_{b_{j,s},j} = \sum_{k\in\mathcal{G}_{b_{j,s}}} l_k \prod_{t\in\mathcal{N}_k \cap \{b_{j,1},\ldots,b_{j,s-1}\}} (1 - q_t) P_j q_{b_{j,s}}. \tag{9}$$

As the algorithm selects the content-RSU pairs in a greedy manner, the following equation holds,

$$b_{j,1} = \arg\max_{m} \sum_{k \in \mathcal{G}_m} l_k q_m P_j. \tag{10}$$

Observe that there is a shared term P_j in the calculation of $\sum_{k \in \mathcal{G}_m} l_k q_m P_j$ for every RSU m. Then P_j can be eliminated. Therefore, the first selected RSUs in Algorithm 1 for any content are exactly the same one. Then we replace $b_{j,1}, j \in \{1, 2, \ldots, N\}$ by b_1, and the following equation holds:

$$b_{j,2} = \arg\max_{m} \sum_{k \in \mathcal{G}_m} l_k \prod_{t \in \mathcal{N}_k \cap \{b_1\}} (1 - q_t) q_m. \tag{11}$$

According to (11), the second selected RSU is still the same for any content f_j using the same method above. In the same manner, $b_{j,2}, j \in \{1, 2, \ldots, N\}$ can be replaced by b_2. The recursion can iterate until all RSUs have been picked. As a result, the selection order of RSUs is the same for any content by Algorithm 1, and the order can be rewritten as an array $[b_1, b_2, \ldots, b_M]$. It is noticeable that $\{b_1, b_2, \ldots, b_M\} = \{1, 2, \ldots, M\}$.

On the other hand, from the perspective of an arbitrary RSU b_m, it receives contents in a certain order during the implementation of Algorithm 1. According to line 10 in Algorithm 1, for any content j, the following equation holds,

$$D_{b_m,j} = \sum_{k \in \mathcal{G}_{b_m}} l_k \prod_{t \in \mathcal{N}_k \cap \{b_1, \ldots, b_{m-1}\}} (1 - q_t) q_{b_m} P_j. \tag{12}$$

According to (12), a more popular content will always be cached prior to a less popular one for any RSU b_m. Therefore, after the algorithm completes, $\mathcal{F}_m = \{f_1, f_2, \ldots, f_{|\mathcal{F}_m|}\}$. As a result, the proof is completed. ■

Algorithm 2 Identifying $b_m, m \in \{1, 2, \ldots, M\}$.

1: **Input**: $C, M, K, q_m (m \in \{1, 2, \ldots, M\})$,
2: $l_k (k \in \{1, 2, \ldots, K\})$.
3: **initialization**: $m = 1, H = \emptyset, T = \{1, 2, \ldots, M\}$.
4: **while** $m \leq M$ **do**
5: **for** $y \in T - H$ **do**
6: $W_y = \sum_{k \in \mathcal{G}_y} l_k \prod_{t \in \mathcal{N}_k \cap H} (1 - q_t) q_y$.
7: **end for**
8: $g = \arg\max_{g} W_g$.
9: $b_m \leftarrow g$.
10: $H \leftarrow H \cup \{g\}$.
11: $m \leftarrow m + 1$.
12: **end while**

Then the objective is to find the sequence $[b_1, b_2, \ldots, b_M]$. Algorithm 2 shows the process to identify $b_m, m \in \{1, 2, \ldots, M\}$. It calculates the increment in successful download probability within the whole network of assigning the same content to each RSU, according to line 4 to line 7, and then on line 8 and line 9, the algorithm picks the RSU contained in the pair with the largest increment. Then the RSU is removed from the set of candidates. Above operations loop until the selection order of RSUs is determined. Here we define a new term $G_{b_m} = D_{b_m,j}/P_j$ for simplicity, and according to (12), the following equation holds,

$$G_{b_m} = \sum_{k \in \mathcal{G}_{b_m}} l_k \prod_{t \in \mathcal{N}_k \cap \{b_1, \ldots, b_{m-1}\}} (1 - q_t) q_{b_m}. \tag{13}$$

Since the request probability obeys Zipf distribution, the following equation holds for $b_m, m \in \{1, 2, \ldots, M\}$ and $f_j, j \in \{1, 2, \ldots, N\}$,

$$D_{b_m,j} = G_{b_m} \frac{j^{-a}}{\sum_{r=1}^{N} r^{-a}}. \tag{14}$$

According to Proposition 1 and (14), the normalized cache size of each RSU can be acquired by solving the following problem.

$$\max_{\{C_{b_1}, C_{b_2}, \ldots, C_{b_M}\}} \sum_{m=1}^{M} G_{b_m} \sum_{j=1}^{C_{b_m}} \frac{j^{-a}}{\sum_{r=1}^{N} r^{-a}} \tag{15}$$

$$\text{s.t.} \sum_{m=1}^{M} C_{b_m} = C, \forall m \tag{16}$$

$$C_{b_m} \in \{0, 1, 2, \ldots\}, \forall m, \tag{17}$$

where C_{b_m} is the normalized size of cache allocated to RSU b_m. We use integral to approximate the second term in (15) and relax the integer constraint of (17) to (20). It is reasonable since the normalized cache budget C and the amount of contents can be very large. The transformed problem is reformulated below.

$$\max_{\{C_{b_1}, C_{b_2}, \ldots, C_{b_M}\}} \sum_{m=1}^{M} G_{b_m} \frac{\int_1^{C_{b_m}} x^{-a} dx}{\int_1^{N} x^{-a} dx} \tag{18}$$

$$\text{s.t.} \sum_{m=1}^{M} C_{b_m} = C, \forall m \tag{19}$$

$$C_{b_m} \geq 0, \forall m. \tag{20}$$

Using Lagrange multiplier, the normalized size of cache allocated to each RSU can be obtained as:

$$C_{b_m} = \left\lfloor \frac{G_{b_m}^{\frac{1}{a}} C}{\sum_{k=1}^{M} (G_{b_k})^{\frac{1}{a}}} \right\rfloor, \tag{21}$$

where $\lfloor x \rfloor$ represents the floor function. b_m is derived from Algorithm 2, then G_{b_m} can be derived from (13). According to (21), the normalized size of cache allocated to b_m is proportional to $G_{b_m}^{\frac{1}{a}}$, in which the impact of successful transmission probability and road traffic at RSU m is incorporated into the term G_{b_m}, and the term $\frac{1}{a}$ reflects the effect of skewness of content popularity distribution. From the expression, cache will be allocated in a more uniform manner when α grows large. Compared with Algorithm 1, the complexity decreases from $O(MN^2KC)$ to $O(M^3K)$. The impact of the number of content and cache budget size is eliminated, the values of which two are usually quite large in reality.

TABLE I
MAJOR PARAMETERS FOR EXPERIMENTS

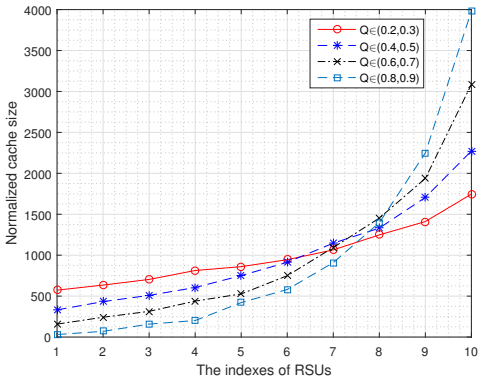
$M = 10$	Number of RSUs
$K = 100$	Number of contact patterns
$Q \in (0, 1)$	Successful transmission probability
$a \in (0.1, 1.3)$	Zipf parameter for content popularity
$N = 20000$	Number of contents in library
$C \in (1000, 10000)$	Normalized cache budget
$V = 20000$	Number of vehicles in the network

IV. NUMERICAL EXPERIMENTS

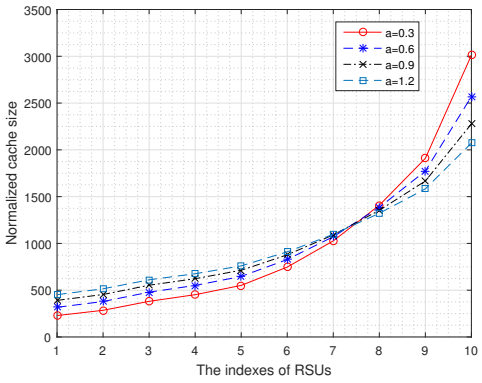
In this section, we present numerical experiments to evaluate the performance of the proposed algorithms and compare it with the greedy algorithm on equal cache (GAEC) from [6], and popularity-based caching algorithm on equal cache (PAEC). Both of them allocate equal-sized cache to each RSU. The GAEC scheme chooses the content in a greedy manner, while the PAEC scheme distributes most popular contents to each cache. The two algorithms differ because a vehicle may encounter several RSUs in certain contact pattern, then distributing most popular contents to every RSU may not be greedy. We investigate how different parameters including the content popularity, successful transmission probability of each RSU and total cache budget affect the successful download probability of the whole network. Major simulation parameters are listed in Table 1. Here the probability of the occurrence of each contact pattern is assumed to obey Zipf distribution, i.e., the probability of contact pattern i is $l_i = i^{-r} / \sum_{k=1}^{K} k^{-r}$ with $r = 1$, which is verified by [5].

Fig 3(a) to Fig 3(c) show how different factors affect the size of cache allocated to RSUs given by our algorithm. Fig. 3(a) indicates that the larger successful transmission probability attains, the steeper cache size allocation among RSUs will be, because when the successful transmission probability is low, cache should be more evenly distributed, in order to have multiple backups for popular contents. Fig. 3(b) shows that when the Zipf parameter a of content popularity is large, more requests are concentrating on a few popular contents, then cache should be allocated more uniformly to store popular contents on more RSUs, which is in consistence with our theoretical results. Fig. 3(c) shows that if there is strong variance of successful transmission probability of different RSUs, then larger cache is supposed to be allocated to those with better wireless channel conditions.

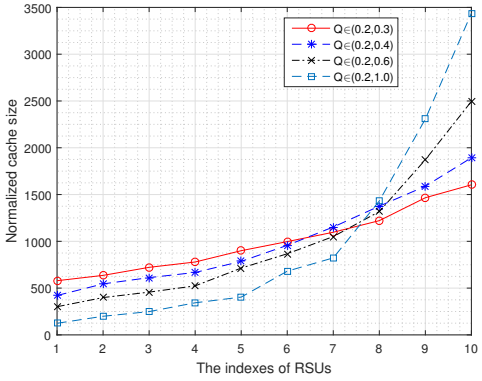
Fig. 4(a) displays how Zipf parameter a of content popularity impacts the successful download probability. In these experiments, Q is randomly chosen in the range of $(0.8, 0.9)$ and C is set to 10000. According to the figure, the proposed algorithm achieves 10% higher of successful download probability within the whole network than GAEC and PAEC when $a < 0.8$, indicating that the cache size should be carefully considered especially when the content popularity is more evenly distributed. As a grows, requests concentrate on a small set of very popular contents and the gap between these



(a)



(b)

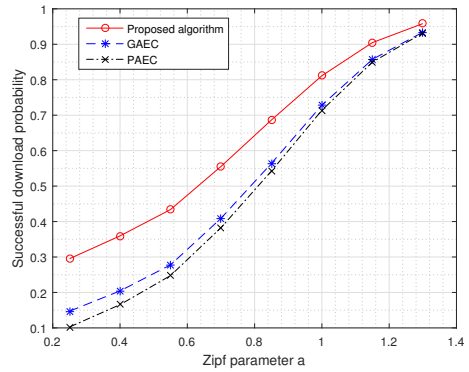


(c)

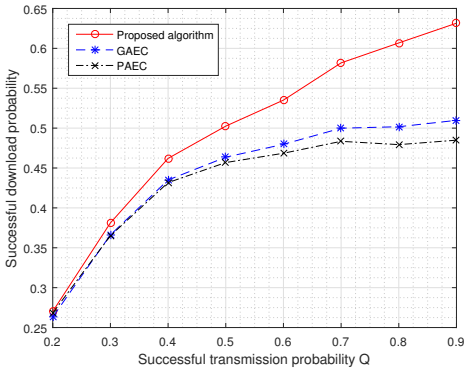
Fig. 3. The impact of different factors on the size of cache allocated to each RSU. (a) The value of the successful transmission probability. (b) The value of Zipf parameter of the request probability. (c) The variance of the successful transmission probability.

algorithms becomes narrow. Also, the performance of GAEC, which is based on cooperative caching of different contents on equal-sized cache, provides little gain over PAEC when a gets large.

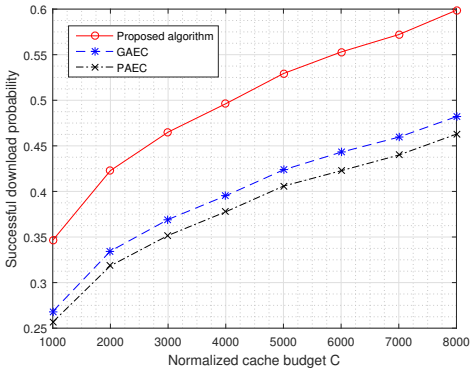
The effect of successful transmission probability Q on the successful download probability within the network is given by Fig. 4(b). Here the normalized cache budget C is set to 10000 and the Zipf parameter of content popularity a is set



(a)



(b)



(c)

Fig. 4. The impact of different factors on the successful download probability. (a) The impact of the content popularity. (b) The impact of the successful transmission probability. (c) The impact of the total cache budget.

to 0.8. Fig. 4(b) reveals that the performance of the three algorithms are all degraded by low successful transmission probability, with the proposed algorithm slightly better than the other two. However, the performance gap broadens as the successful transmission probability grows.

Fig. 4(c) demonstrates the impact of normalized cache budget on the successful download probability. The normalized cache size of each RSU in GAEC and PAEC is set to $\lfloor C/M \rfloor$. Here Q is randomly chosen within the range of $(0.8, 0.9)$ and a is set to 0.8. Fig. 4(c) reveals that larger cache renders higher successful download probability for all three algorithms. The figure also show that using the proposed algorithm, the successful download probability is more than 10% higher than the other two and the performance gap is immune to the variance of the cache budget.

V. Conclusions

This paper investigates nonuniform cache allocation and content placement in vehicular networks exploiting the moving information of vehicles. The problem is proved to be NP-hard and thus a low-complexity approximate algorithm is proposed. The performance of the algorithm is guaranteed to achieve a factor of $(1 - \frac{1}{e})$ within the optimal strategy. The expression of the cache size from the proposed algorithm is derived by theoretical analysis, according to which, more concentration of content popularity and smaller value of successful wireless transmission probability result in more even distribution of cache. Numerical experiments show that the proposed algorithm performs 10% better than baseline algorithms due to the consideration of non-uniformity of cache size, and the joint optimization of cache allocation and content placement.

Acknowledgement

This work is sponsored in part by the Nature Science Foundation of China (No. 91638204, No. 61571265, No. 61621091), and Qualcomm Technologies, Inc.

References

[1] Y. Huang, Y. Gao, K. Nahrstedt and W. He, "Optimizing File Retrieval in Delay-Tolerant Content Distribution Community," *IEEE ICDCS'09*, 2009.

[2] K. Liu, J. K.-Y. Ng, J. Wang, V. Lee, W. Wu, and S. H. Son, "Network-coding-assisted Data Dissemination via Cooperative Vehicle-to-vehicle/-infrastructure Communications," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 6, pp. 1509–1520, Jun. 2016.

[3] D. Zhang and C. Kiat Yeo, "Enabling Efficient WiFi-Based Vehicular Content Distribution," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 3, pp. 479-492, March. 2013.

[4] R. Ding, T. Wang, L. Song, Z. Han and J. Wu, "Roadside-unit Caching in Vehicular Ad Hoc Networks for Efficient Popular Content Delivery," *IEEE WCNC'09*, 2009.

[5] G. Mauri, M. Gerla, F. Bruno, M. Cesana and G. Verticale, "Optimal Content Prefetching in NDN Vehicle-to-Infrastructure Scenario," *IEEE Trans. Veh. Technol.*, vol. 66, no. 3, pp. 2513-2525, March. 2017.

[6] R. Kim, H. Lim and B. Krishnamachari, "Prefetching-Based Data Dissemination in Vehicular Cloud Systems," *IEEE Trans. Veh. Technol.*, vol. 65, no. 1, pp. 292-306, Jan. 2016.

[7] X. Peng, J. Zhang, S. H. Song and K. B. Letaief, "Cache Size Allocation in Backhaul Limited Wireless Networks," *IEEE ICC'16*, 2016.

[8] Vu, Thang X. and Chatzinotas, Symeon and Ottersten, Bjorn, "Coded Caching and Storage Planning in Heterogeneous Networks," *IEEE WCNC'17*,2017.

[9] Y. Huang, Gao, K. Nahrstedt and W. He, "Optimizing file retrieval in delay-tolerant content distribution community,"*IEEE ICDCS'09*, 2009.

[10] W. Li, "Random Texts Exhibit Zipf's-law-like Word Frequency Distribution," *IEEE Trans. Inf. Theory.*, vol. 38, no. 6, pp. 1842-1845, Nov 1992.

[11] L. Breslau, Pei Cao, Li Fan, G. Phillips and S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications," *IEEE INFOCOM '99*, 1999.

[12] M. L. Fisher, G. L. Nemhauser, L. A. Wolsey, "An analysis of approximations for maximizing submodular set functions—II" *Mathematical Programming*, vol. 14, no. 1, pp. 265-294, 1978.