Chapter 3
Mobile Edge Caching

Abstract Edge caching is a key part of mobile edge computing. It not only can support the necessary task data for edge computing, but also enables powerful Internet of Things applications with massive amounts of data and various types of information in access networks. In this chapter, we present the architecture of the edge caching mechanism and introduce metrics for evaluating caching performance. We then discuss key issues in caching topology design, caching data scheduling, as well as caching server cooperation and present a case study of artificial intelligence–empowered edge caching.

3.1 Introduction

In recent years, wireless communication has witnessed an explosive growth of smart Internet of Things (IoT) devices and powerful mobile applications that greatly facilitate our daily life and improve manufacturing efficiency. The implementation of these applications requires massive content input and relies heavily on high-speed, low latency data transmission. However, the backhaul links between content servers and wireless access points are always constrained by bandwidth, making the stringent transmission requirements hard to meet. This data hungry characteristic poses significant challenges for mobile networks, especially in application scenarios with a large scale of smart devices. Mobile edge caching is a promising approach to alleviate backhaul load and address these challenges. It utilizes caching resources at the edge nodes and provides popular content access close to the end users.

3.2 The Architecture of Mobile Edge Caching

To clearly illustrate the components and operation mechanism of edge cache systems, we present a hierarchical mobile edge caching architecture, which is illustrated in Fig. 3.1. This architecture consists of four layers, namely, an application layer, a user node layer, an edge server layer, and a cloud service layer.

Y. Zhang, *Mobile Edge Computing*, Simula SpringerBriefs on Computing 9,
https://doi.org/10.1007/978-3-030-83944-4_3

Fig. 3.1 Hierarchical mobile edge caching architecture

The application layer contains a variety of wireless mobile applications with intensive data input requirements. For instance, the driving control of autonomous vehicles depends on high-resolution map data. Furthermore, the map data provided to the vehicles should be updated in real time as traveling vehicles reach different areas. Another example is the daily news broadcast. Although the data delivery does not have strict delay constraints, since the proportion of multimedia news continues to increase, a large amount of the media data will pose a heavy transmission burden on wireless networks. To distinguish different applications according to their caching service demands, we model the data caching task as $(f, t^{\max})$, where f is the amount of data required and $t^{\max}$ denotes the maximum latency tolerance before the data are received.

Above the application layer is the user node layer, which is composed of multiple forms and types of devices and physical entities, including mobile phones, wearable devices, tablets, and smart vehicles. It is worth noting that the user nodes can be data requesters while also acting as data providers. A typical example is a smartphone. When a video that has not been downloaded onto the phone is played, the phone can ask for data from a remote video server. During the video playback, the phone can provide the cached data to other phones in device-to-device (D2D) mode. Consequently, parts of user nodes can be regarded as special data servers and classified at the edge server level. The characteristics of the user nodes can affect the data caching performance, including the node's speed of motion, wireless transmission power, and communication topology.

The edge server layer focuses on providing data to user nodes in proximate areas at low cost. To do so, edge caching servers are usually installed in cellular network base stations, roadside units (RSUs), Wi-Fi access points, unmanned aerial vehicles, and other access network infrastructures. On the one hand, cached data can be delivered by directly using the wireless transmission capabilities of these infrastructures; on the other hand, the power supply facilities of these infrastructures can be utilized to provide adequate electricity to the edge servers. These servers are associated with two key performance evaluation indicators. One is the data storage capacity, and the other is the service coverage radius, which is determined by the transmission power of the attached wireless infrastructures and their operating environments.

When edge servers cannot meet data demands due to limitations in cache capacity, the caching nodes in the cloud service layer will provide users with a supplementary backup. These cloud servers are powerful and usually have vast amounts of storage space and can thus cache large amounts of data. Although the cloud server can be located as far away from the user nodes as an application server, there are essential differences between the two. As data generators, application servers are scattered throughout various networks. Access to their data can be severely restricted by their communication capabilities. In contrast, cloud servers are mostly located in the core network, equipped with high-speed input–output equipment and high-capacity transmission facilities, with, for example, optical fiber, facilitating high-speed data access for users.

The operation of edge caching requires efficient management, so a caching control module is introduced into the system that is responsible for tracking data popularity and scheduling storage resources. Considering that the caching scheduling relies on the collaboration of various types of entities, including both the data servers and the requesters, this module is implemented across multiple layers. With the advancement of artificial intelligence, powerful machine learning approaches have been implemented within the control module. They help extract data features in complex mobile application environments, predict data demand trends, and fully tap the data caching potential of large-scale heterogeneous service devices.

3.3 Caching Performance Metrics

Many studies are devoted to edge cache optimization, adopting various technical approaches to address different problems in server layout, resource scheduling, data management, and so forth. To quantitatively evaluate the pros and cons of these approaches, the following caching performance metrics are introduced.

3.3.1 Hit Rate Ratio

The hit rate ratio is one of the most important performance metrics in edge caching. To understand this metric, we first need to define the concept of a hit. In the edge caching process, if a user can directly obtain the requested content from an edge cache server, without the help of cloud service facilities or remote data sources, this data access is called a hit. Thus, the hit rate ratio can be calculated by dividing the number of cache hits by the total number of cache hits and misses, as follows:

$$r = \frac{h}{h + m} \tag{3.1}$$

where h is the number of cache hits and m is the number of cache misses. It is worth noting that m contains not only content obtained from the cloud or remote servers, but also content that was not successfully obtained due to transmission failure.

The hit rate ratio reflects how effective a cache is at fulfilling requests for content; in other words, it measures the utilization efficiency of edge service resources in satisfying end users. To improve the performance metric, content popularity–based caching strategies are usually adopted to determine the type of cached data and data update frequency.

3.3.2 Content Acquisition Latency

Content acquisition latency is another chief measurement in caching services that indicates the total time cost, from the user generating a data request to obtaining the complete data. Content acquisition latency can be formally represented as

$$T_a = t_{\text{req}} + \sum_{k \in \mathcal{K}} t_{\text{res},k} \tag{3.2}$$

where t_{req} is the time for a request to be transmitted from a user to the chosen server. Since the request message is small, t_{req} can usually be ignored. If the cache server directly requested by the user does not have the required data, it needs to obtain the resource from a remote server that is $\mathcal{K}$ hops away. The variable $t_{\text{res},k}$ is the one-hop transmission delay of the data response.

Many mobile applications are now delay sensitive. For instance, in mobile navigation, digital map data need to be delivered to the user's smartphone for route planning when the user first enters an area. To improve the timeliness of news, breaking news needs to be pushed to the user end as soon as possible.

The latency metric can be used in two ways in caching service management: one way is to minimize it as an optimization objective, and the other is to make it a constraint of an optimization problem. The content acquisition delay is affected by the location of the data source server and the transmission rate. Approaches to reduce latency can therefore be considered in three aspects: the effective placement of edge caching servers, the timely update of stored data, and the optimized scheduling of communication resources.

3.3.3 Quality of Experience (QoE)

The QoE is a metric that evaluates the performance of edge caching services from the perspective of data users. Since multiple users can have diverse data requirements, QoE can be reflected in different aspects. When a user downloads a large video through cellular networks, the user might mainly concern about the data transmission

cost. For content that is urgently needed, users will focus on the timeliness of the data acquisition. A user receiving an advertisement push will care about whether the content meets his or her shopping demands. Considering that users can have multiple preferences at the same time, QoE can also be formed as a combination of multiple factors, as follows:

$$QoE = \sum_{n=1}^{N} \alpha_n \cdot q_n \tag{3.3}$$

where q_n is the nth type of metric value in the QoE evaluation, and α_n is a coefficient that reflects the user's subjective measurement of the importance of different metrics. Since a positive data acquisition experience will encourage users to spend more on edge caching services, which will increase the revenue potential of the edge server operator, QoE has become an important criterion for cache design and management.

3.3.4 Caching System Utility

Unlike QoE, which only considers user-side gains, a system utility metric introduces service-side revenue factors into the caching performance evaluation. Edge cache servers are usually deployed and managed by operators, and the servers' operation process produces energy, maintenance, and other costs. On the other hand, operators can obtain rewards by providing data services to end users. Rational operators aim to reduce costs while maximizing profits. From the perspective of the service side, caching utility can be defined as the difference between the profits and costs. Furthermore, since an edge caching system includes both users and services, we define the system utility as the sum of the user- and service-side utilities, which can be presented as

$$U_{\text{sys}} = R_{\text{server}} - C_{\text{server}} + G_{\text{user}} - C_{\text{user}} \tag{3.4}$$

where R_{server} and C_{server} are the reward and operation costs of the caching servers, respectively; G_{user} denotes the user utility gained from the edge caching service, such as improved QoE or savings in remote transmission consumption; and C_{user} is the cost paid by the users to access the service, for example, the price paid to the server operator. Generally, the value of C_{user} is equal to R_{server} in a given edge caching system. To improve the caching system utility, we can adopt a game-theoretic approach and form a cooperative game between the users and servers that helps find caching scheduling strategies that benefit both sides.

3.4 Caching Service Design and Data Scheduling Mechanisms

Edge caching is expected to provide data storage services for users to access low latency content, while releasing the burden on backhaul networks between the data generator and end users. Thanks to the evolution of wireless communication and the development of IoT technology, pervasive smart nodes can interact with each other and become integrated so that edge cache services can be implemented on diverse types of nodes and in various network segments. In this section, we investigate an edge cache design from the perspective of how the edge caching service is deployed at heterogeneous nodes, and we then discuss the corresponding caching data scheduling mechanisms. Without loss of generality, we divide the nodes by infrastructures and user devices and correspondingly classify the cache service modes into three types, namely, infrastructure supported, user device sharing enabled, and a hybrid type, as illustrated in Fig. 3.2.

3.4.1 Edge Caching Based on Network Infrastructure Services

The network infrastructure is a vital part of a communication system, and it comprises hardware and software that enables communication connectivity between user devices and service facilities. In this section, infrastructure refers specifically to the base stations of cellular networks, the RSUs in vehicular networks, and the wireless access points in Wi-Fi networks, which are always managed by network operators.

The above-mentioned infrastructures have common characteristics. First, they have a large data cache space that usually reaches several megabytes or even gigabytes. Second, the infrastructures serve a wide coverage area, delivering cached data to multiple user nodes. Through the coordination of multiple infrastructure nodes to form a mesh network, their service range can be further improved. Finally, the infrastructures have fixed locations, and it is hard to spatially adjust the caching service pattern of an individual infrastructure node.

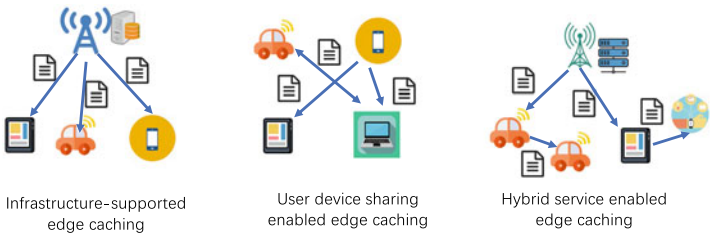

Fig. 3.2 Edge caching service modes

In view of the above characteristics, infrastructure-based edge caching mechanisms mainly focus on data popularity determination, target server selection, and cache data updates. It is worth noting that, although infrastructures cannot change their locations, they can coordinate with multiple servers distributed in different areas and adjusting their caching strategies to meet the dynamic spatial distribution of data demands. The caching strategy of infrastructure-based servers can generally be described as $a^t_{m,n} = \{0, 1\}$, where m and n are the server node index and the type of data, respectively, and t denotes the time slot for the caching strategy. Here $a^t_{m,n} = 1$ indicates the data are cached, and vice versa.

A few previous works have examined infrastructure-based edge caching. Xu, Tao, and Shen [35] used small base stations as edge caching servers and investigated cache placement optimization to minimize the system's long-term transmission delay without knowledge of user data preferences. The optimization scheme was formulated as a sequential multi-agent decision problem. Moreover, the authors proposed a direct online learning cache strategy in both stationary and non-stationary environments that achieves a good balance between gains in base station coordination and computational complexity. Wang et al. [36] presented an edge caching scenario for mobile video streaming in which base stations distributed citywide provide video storage capacity. Based on the analysis of viewers' request patterns behind both spatial and temporal dimensions, the authors proposed a multi-agent deep reinforcement learning–empowered caching scheme that minimizes both content access latency and traffic costs.

3.4.2 Edge Caching Based on D2D Services

Although infrastructure-based edge caching has produced a promising paradigm to push data closer to the network edge with low transmission latency, there are still problems to be addressed. For instance, the data delivery of infrastructures usually has a large wireless coverage range and works in broadcast mode. Excessive transmission distances can weaken the space-division multiplexing of the spectrum, thereby reducing the efficiency of data caching services. Furthermore, multiple user nodes can have different types of data demands. However, it is hard for the broadcast delivery mode to provide edge services with demand recognition and differentiation.

To address these problems, we resort to D2D caching services. Driven by the advancement of IoT technology, smart devices are becoming increasingly popular. These devices are equipped with a certain data caching capacity and multiple types of wireless communication interfaces, such as cellular, Wi-Fi, and Bluetooth, making the devices potential data carriers and deliverers.

Technical challenges, however, arise in using smart devices to provide efficient caching services. One of the challenges is the dynamic network topology caused by device movement, which makes the stable maintenance of data services for given areas very difficult. In addition, multiple device pairs could concurrently communicate and deliver cached data. It is not easy to efficiently schedule communications in

a distributed scenario. Last but not least, constrained by its limited cache resources, an individual user's device cannot provide cache services for large files.

In response to the above challenges, academics have carried out in-depth research. Some works have focused on the analysis of device mobility patterns and leveraged the mobility to expand caching service coverage. For instance, Qiao, Leng, et al. [37] introduced a paradigm to jointly manage content placement and delivery in vehicular edge networks. The caching optimization problem was formulated as a double time scale Markov decision process, based on the consideration that content popularity changes are less frequent compared to those associated with vehicle mobility. The authors proposed a deep deterministic policy gradient (DDPG) approach to achieve minimum system costs and content delivery latency. Regarding complex wireless edge networks, research efforts have been devoted to coordinating multiple user devices and improving the efficiency of data services.

Karasik, Simeone and Shamai [38] leveraged the benefits of out-of-band broadcast D2D communication for caching data delivery in a complex fog radio access network. To minimize the normalized delivery time, a compress-and-forward–based edge caching and D2D communication scheme was proposed, which proved to be an information-theoretically optimal approach. Moreover, researchers have studied the integration of multiple user devices into an edge service node with strong caching capabilities to overcome the weakness of an individual device's constrained storage space. For example, Zhang, Cao, et al. [39] exploited the relations between caching-empowered vehicles in content dispatch services and proposed a social–aware mobile edge caching scheme that leverages deep reinforcement learning to organize social characteristic–aware vehicles in content processing and caching and maximizes dispatch utility.

3.4.3 Hybrid Service–Enabled Edge Caching

Although both infrastructure service-based and D2D service-based edge caching approaches can distribute popular data in proximity to mobile users via local storage capabilities, inherent shortcomings still remain due to the location, caching, and communication characteristics of the edge servers. As mentioned above, the location of the edge cache–enabled infrastructure is fixed, so its data service coverage is difficult to adjust. In D2D edge caching, though user device locations can be flexibly changed, the devices' storage space is usually small. Moreover, due to the large-scale distribution and independent control of different devices, the collaborative caching of multiple devices is sometimes inefficient.

Combining the infrastructure-based approach with the D2D caching approach to build a hybrid edge caching mechanism has emerged as a promising paradigm to address the above-mentioned problems. On the one hand, user devices can spread data to far places as they move, making up for the fixed coverage of infrastructures. On the other hand, an infrastructure uses its large storage capacity to remedy the inability of user devices to store large files. In some cases, cloud data servers are also

Table 3.1 Comparison of edge caching modes

	Infrastructure supported	User device sharing enabled	Hybrid service enabled
Capacity	High	Low	High
Latency	Low	High	Medium
Cost	High	Low	Medium

integrated into data caching systems to provide powerful data source support for the various types of edge servers.

Being a key enabling technique in enhancing data delivery efficiency, the hybrid edge caching mechanism has attracted a great deal of research interest. Wu, Zhang, et al. [40] introduced a D2D-assisted cooperative edge caching scheme in millimeter-dense networks where the cache resources of users and small base stations are jointly utilized for data storage and delivery according to content popularity. Zhao, Liu, et al. [41] designed a caching scheme that combines caching placement and the establishment of D2D with the aid of small base stations. Popular files are prefetched in the local cache during off-peak periods and served to users at peak times, thereby reducing communication pressure on the backhaul link. Zhang, Yu, et al. [42] considered motivations and security in caching service and proposed a blockchain-based cache and delivery market that guarantees the expected reward of both user devices and edge nodes in data sharing. Saputra, Hoang, et al. [43] introduced a proactive cooperative caching approach that uses a content server to predict the content demand for the entire network and that optimizes the distributed caching service of edge nodes. Kwak, Kim, Le and Chong [44] extended the hybrid caching mechanism to the cloud end and proposed a content caching algorithm for joint content caching control in central cloud units and base stations in a hierarchical cellular network.

Table 3.1 compares the key performance of the service modes cited above. Since the infrastructure is deployed and maintained by operators, it has a high capacity and low transmission delay. However, content subscribers need to pay the operators for the caching and transmission services, so infrastructure-supported edge caching usually has a high cost. In contrast, user device sharing–enabled caching utilizes the caching capacity of IoT devices. Although the performance of D2D cache services can be poor, the direct delivery of data without operator participation saves greatly on costs. The hybrid service mode incorporates the previous two approaches and achieves high caching capacity and medium delivery latency, while reducing the cost compared to the infrastructure-supported mode.

3.5 Case Study: Deep Reinforcement Learning–Empowered Social–Aware Edge Caching

To further elaborate the edge caching mechanism and performance evaluation metrics, in this section we present a case study that focuses on deep reinforcement learning–empowered social–aware edge caching management.

3.5.1 System Model

We consider an edge service–empowered vehicular social network with M_i RSUs located in an area i, $i \in \mathcal{I}$. The RSUs' caching space is limited, and their maximum caching capacities are denoted as $\{s_1^r, s_2^r, \ldots, s_{M_i}^r\}$, respectively. In addition, each RSU is equipped with a mobile edge computing server that helps process computation tasks offloaded to it. The computing capacities of these servers are $\{c_1^r, c_2^r, \ldots, c_{M_i}^r\}$, respectively.

The vehicular network consists of smart vehicles. These vehicles generate content, including road congestion status, driving control indications, and vehicle sensing information. This content needs to be processed and delivered among vehicles and roadside infrastructures. The processing and delivery of content are always undertaken jointly and consequently. We use the term *content dispatch process* to represent the combination of content processing and the delivery process. The content involves different data sizes and diverse computing demands. We consider K types of content and denote type k content as $\{f_k, c_k, t_k^{\max}\}$, where f_k and c_k are the content size and required amount of computation, respectively. Variable $t_k^{\max}$ is the maximum latency tolerance of type k content to be dispatched to receiving vehicles. The vehicular network operates within a discrete time model with time slots of fixed length.

To satisfy the computing demand for particular content, the content can either be processed on a vehicle with its on-board computing resources or offloaded to and executed on a mobile edge computing server. Let $x_{v,k}$ be the probability that a given vehicle processes type k content and $1 - x_{v,k}$ the probability that the newly generated content is offloaded to an RSU. Moreover, in the vehicular network, both vehicle-to-vehicle (V2V) and vehicle-to-RSU (V2R) modes can be exploited for content transmission. We use $y_{v,k}$ and $y_{r,m,k}$ to denote the probabilities of a vehicle choosing to deliver type k content through the V2V or V2R mode, respectively, after the content is processed.

In the content dispatch process, social relations and mobility aspects of smart vehicles are jointly exploited. The contact rate is taken as a key indicator to characterize the social relationships between vehicles. Vehicular pairwise inter-contact times follow an exponential distribution. The contact rate between two vehicles in area i is $\lambda_{v,i}$. In addition, RSUs equipped with caching resources can also act as content distribution relays. The various transmission ranges and different locations

of these RSUs lead to different contact rates between the RSUs and the vehicles, which are denoted as $\{\lambda_1, \ldots, \lambda_{M_i}\}$, respectively.

3.5.2 Problem Formulation and a DDPG-Based Optimal Content Dispatch Scheme

The content can have different levels of dispatch performance due to various computation offloading targets, caching strategies, and transmission modes. The number of vehicles in an area to which type k content, $k \in \mathcal{K}$, has been dispatched under delay constraint $t_k^{\max}$ can be expressed as

$$n_k^{\text{total}} = x_{v,k} y_{v,k} n_{1,k} + x_{v,k} \sum_{m}^{M} y_{r,m,k} n_{2,k,m} + \left(1 - x_{v,k}\right) \sum_{m}^{M} \rho_{m,k} n_{3,k,m} \tag{3.5}$$

where $n_{1,k}$, $n_{2,k,m}$, and $n_{3,k,m}$ are the numbers of vehicles that obtain content within their delay constraints through on-board computing and caching, on-board computing with RSU caching, and RSU computing and caching approaches, respectively.

Optimal content dispatching should maximize the total number of vehicles that receive various types of content under specified delay constraints and can be formulated as

$$\begin{aligned}
&\max_{\{x_{v,k}, y_{r,m,k}, y_{v,k}, \rho_{m,k}\}} \sum_{k=1}^{K} n_k^{\text{total}} \\
&\text{suchthat} \quad \text{C1}: \sum_{k=1}^{K} x_{v,k} y_{r,m,k} f_k + \sum_{k=1}^{K} f_k \left(1 - x_{v,k}\right) \cdot \rho_{m,k} \le s_m^r, \quad m \in \mathcal{M} \\
&\qquad \text{C2}: \; 0 \le x_{v,k}, y_{v,k}, y_{r,m,k}, \rho_{m,k} \le 1, k \in \mathcal{K}, m \in \mathcal{M} \\
&\qquad \text{C3}: \sum_{m=1}^{M} y_{r,m,k} + y_{v,k} = 1, \quad k \in \mathcal{K} \\
&\qquad \text{C4}: \sum_{m=1}^{M} \rho_{m,k} = 1, \quad k \in \mathcal{K}
\end{aligned} \tag{3.6}$$

where constraint C1 indicates that the amount of data cached on RSU m should not exceed its cache capacity; C2 gives the ranges of the decision variables $x_{v,k}$, $y_{v,k}$, $y_{r,m,k}$, and $\rho_{m,k}$; C3 implies that either content should be transmitted V2V or an RSU should be selected for V2R data delivery; and constraint C4 indicates that one of the M RSUs should be selected when edge computing services are utilized to process content.

To address this problem, we design a DDPG-based dispatch scheme by leveraging a deep reinforcement learning approach. The DDPG is a policy gradient deep reinforcement learning algorithm that concurrently learns policy and value functions in the learning process. The DDPG agent learns directly from unprocessed observation spaces through a policy gradient method that estimates the policy weights,

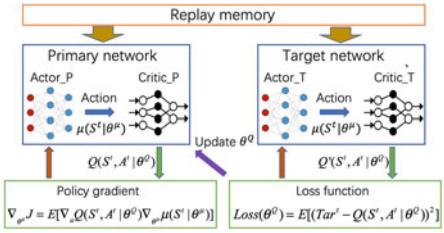

Fig. 3.3 Architecture of the DDPG-based scheme

and employs an actor–critic model to learn the value function and update the actor model. Since the DDPG utilizes a stochastic behavior policy for strategy exploration but estimates a deterministic target policy, it greatly reduces learning complexity.

In our designed DDPG learning approach, the action set taken in slot t is $\mathcal{A}^t = \{x^t_{v,k}, y^t_{r,m,k}, y^t_{v,k}, \rho^t_{m,k}, \alpha^{t,i}_k, p^t_{i,j,k}\}$, where $k \in \mathcal{K}$, $m \in \mathcal{M}_i$, and $i \in \mathcal{I}$. The state at time slot t is $\mathcal{S}^t = \{S^t_{1,k,i}, S^t_{2,k,m,i}, S^t_{3,k,m,i}, S^t_{4,k,i}, S^{t,j,i}_{5,k,m,i}, S^{t,j,i}_{6,k,m,i}\}$, which represents the number of vehicles that have obtained content through various ways. Moreover, we introduce two neural network parameters, namely, θ^μ and θ^Q, in the DDPG learning process. The parameter θ^μ is updated by the primary actor neural network using the sampled policy gradient, which can be shown to be

$$\nabla_{\theta^\mu} J = \mathrm{E}[\nabla_a Q(S^t, A^t|\theta^Q) \nabla_{\theta^\mu} \mu(S^t|\theta^\mu)] \tag{3.7}$$

where $Q(S^t, A^t|\theta^Q) = \mathrm{E}[u^t + \eta Q(S^{t+1}, \mu(S^{t+1})|\theta^Q)]$ is an action value function, and $\mu(S^t|\theta^\mu)$ is the explored policy. The term θ^Q is updated by a primary critic neural network by minimizing a loss function, which is defined as

$$Loss(\theta^Q) = \mathrm{E}[(Tar^t - Q(S^t, A^t|\theta^Q))^2] \tag{3.8}$$

where the target value $Tar^t = u(S^t, A^t) + \eta Q'(S^{t+1}, \mu'(S^{t+1}|\theta^{\mu'})|\theta^Q)$. The DDPG-based content dispatch scheme concurrently learns an action value function and dispatch policies. It uses off-policy data and the Bellman equation to learn the action value function, and it utilizes the action value function to learn the policy.

Figure 3.3 shows the architecture of the DDPG-based dispatch scheme. The compositions of the primary and target networks are similar, and both have an actor and a critic. The actor and critic are two deep neural networks. In the primary network, the actor explores the content dispatch policy $\mu(S^t|\theta^\mu)$, while the critic helps the actor learn a better policy through a gradient approach. The target network is an old version of the primary network. It generates a target value to train the critic in the primary network, where the policy θ^Q is updated through the calculated function $Loss(\theta^Q)$. The replay memory stores the learning experience used to update the actor and critic parameters.

3.5.3 Numerical Results

We evaluate the performance of the proposed content dispatch schemes based on a real traffic dataset that contains mobility traces of approximately 500 taxi cabs in the San Francisco Bay Area.

Figure 3.4 shows the convergence of the proposed DDPG-based dispatch scheme with different values of λ_v, the average vehicular contact rates of the different combinations of areas. In different scenes with different λ_v values, the DDPG-based scheme converges within 8,000 iterations. In addition, the figure indicates that λ_v significantly affects the number of vehicles that can receive content within the specified constraint. A larger λ_v means a higher probability of vehicles meeting and interacting with each other. Content dispatch efficiency improves as λ_v increases. It is worth noting that, in practice, the proposed DDPG-based dispatch scheme is executed offline. For a given steady vehicular traffic flow, we could obtain the optimal content dispatch strategies for vehicles with various states in advance. The strategy set is stored in the control center and can be accessed and applied to vehicles directly, without very many learning iterations.

Figure 3.5 compares the content dispatch performance of the proposed DDPG-based scheme under two social–aware data forwarding approaches, that is, Epidemic and ProPhet. In the Epidemic scheme, a vehicular data carrier forwards data to its contacted vehicle in V2V mode if the target vehicle does not have the data. In the

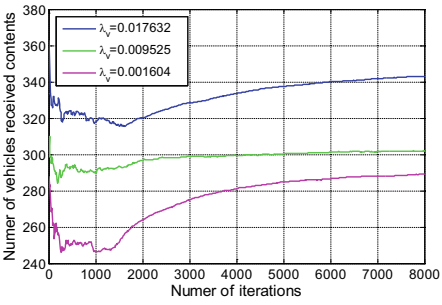

Fig. 3.4 Convergence of the proposed DDPG-based dispatch scheme

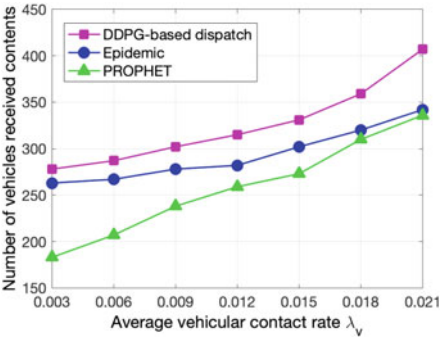

Fig. 3.5 Comparison of the content dispatch performance under different approaches

ProPhet scheme, the vehicular carrier only forwards the data if its contacted vehicles have higher contact rates compared to its own. We use the number of vehicles that can receive content under their delay constraints as the metric to evaluate the caching performance. It is worth nothing that this metric can be considered a special form of the hit rate ratio metric. The figure shows that the DDPG scheme is the most efficient, since it jointly exploits the data dispatch capabilities of both vehicles and RSUs, while adaptively optimizing dispatch strategies according to the various vehicular contact rates of different areas. In contrast, under the Epidemic approach, vehicles forward cached content to contacted vehicles only if their caching space is unoccupied. This approach uses only V2V delivery and ignores the data distribution capabilities of RSUs. In ProPhet, a vehicle only forwards cached content if the contacted node has a contact rate higher than its own. Consequently, ProPhet relies only on RSUs for content distribution when λ_v is small, but relies on V2V content delivery when λ_v exceeds the contact rate metrics of the RSUs. The single data delivery mode of each of these two approaches seriously decreases their dispatch efficiency.