A. Proactive Caching The reactive caching policy determines whether to cache a particular content after it has been requested. It typically happens when the network is at peak-traffic hour and cannot effectively cope with the peak traffic. On the other hand, a proactive caching policy determines which contents should be cached before they are requested based on the prediction of user demands [96]. Proactive caching usually utilizes the estimations of request patterns (e.g., user mobility patterns, user preferences and social relationships) to improve caching performance and guarantee QoS requirements. As machine learning and big data analytics advance, it is advantageous to cache popular contents locally before the requests truly arrive [32], [49]. Proactive caching improves the caching efficiency by pre-downloading popular contents during offpeak times and serving predictable peak-hour demands. Bastug et al. [32] proposed a proactive networking paradigm which leverages social networks and content popularity distributions to improve the caching performance in terms of the number of satisfied requests and the offloaded traffic. They demonstrated that proactive caching performs better than reactive caching. Tadrous et al. [97] considered the system in which the popularity of services can be predicted. Cache nodes can proactively cache services during off-peak hours according to their popularities. They explored the proactive caching scheme by considering the resource allocation to maximize the cost reduction which is related to the offloaded traffic incurred by proactive caching. To further improve the performance of proactive caching, it is desirable to jointly optimize the caches among multiple nodes. Hou et al. [98] exploited a learning-based approach for proactive caching to maximize the cache hit ratio. In their system model, different caches can share information and contents. They first estimated the content popularity by a learning method and then designed a greedy algorithm to obtain the suboptimal content distribution solutions. However, the caching performance highly depending on the prediction accuracy is the major drawback of proactive caching. Prediction errors can gravely degrade the caching performance [99]. B. Distributed Caching Centralized caching uses a central controller, which possesses a global view of all network status, to determine caching schemes. The central controller usually tracks the information of user mobility patterns and the channel state information (CSI) by extracting and analyzing the received requests. Hence, the centralized caching is able to achieve the optimum caching performance with optimum caching decisions (e.g., content placement). However, obtaining full network information is challenging especially in the context of dynamic 5G wireless networks, which are expected to serve an increasing number of mobile users [100]. Furthermore, the central controller has to process a large amount of traffic, which incurs a great burden on the controller as well the links between the controller and network entities. In that case, the central controller can be the bottleneck of the mobile caching system. In distributed caching, which is also referred to as decentralized caching, cache nodes make decisions (e.g., content placement and update) only based on their local information and the information from adjacent nodes. Distributed caching is applied in [49] where adjacent BSs are jointly optimized to increase the cache hit probability. By fetching contents from multiple neighboring caches, the total cache size seen from the user can be increased. The believe propagation (BP) method has been proposed as an efficient way to distributively solve the resource allocation problems in wireless networks. In BP, the complex global optimization problem is usually decomposed into multiple subproblems, which can be effectively addressed in the parallel and distributed manner. A tutorial of BP can be found in [101]. Li et al. [102] discussed the file placement problem to minimize the average file downloading delay. Their network architecture consists of a MBS and cache-enabled SBSs to which UEs’ requests are preferentially forwarded. They divided the files in the file library into several file groups and assumed that each SBS can only cache one file group. A distributed BP algorithm was proposed with the aid of a factor graph, which is a bipartite graph consisting of factor nodes and variable nodes. A factor node refers to the utility function of a user and is related to the average file download delay. Each variable node indicates a cache status vector of each SBS. Only if a UE is under the coverage of a SBS, there can be an edge connecting the UE (factor node) to the SBS (variable node). The BP algorithm is then implemented by iteratively passing messages between the factor nodes and variable nodes. In each iteration, the message is represented by a probability mass function based on the UE’s utility function; each variable node updates its message to be sent to connected factor nodes and each factor node updates its message to be sent to connected variable nodes. The BP algorithm terminates when the messages do not change. Different from [102] which assumed that each UE can only be served by one BS, Liu et al. [92] proposed a distributed BP algorithm to minimize the average download delay in cellular networks where each user can be served by multiple cache-enabled BSs. The data transmission scheme depends on the cache placement. If only one BS caches the requested file, the BS will transmit the file to the user directly; otherwise, multiple BSs can transmit the file via cooperative beamforming. In their BP model, each BS iteratively collects local information (e.g., user requests and CSIs), runs computations, and exchanges messages with the neighboring BSs until convergence. They demonstrated that the distributed BP algorithm requires less calculations than the centralized one. C. Cooperative Caching Since the caching space in a BS is relatively small, designing a caching policy for each BS independently may result in an insufficient utilization of caches. This happens when some of the caches are overly used while others have many vacant spaces. In order to address this issue, cooperative caching policies have been proposed to improve the caching efficiency. In the cooperative caching, BSs are able to share cached contents with each other [99]. However, the delay of searching and retrieving contents from other caches may also be significant and hence should be taken into consideration. In order to actualize cooperative caching, network nodes should be aware of the caching status of other nodes by information exchanges that may induce significant signaling overheads. Hence, we need to find a solution to share the caching status with the minimum overhead. Jiang et al. [103] developed the cooperative caching policy for HetNets where users can fetch contents from FBSs, D2D communications or MBS. They formulated the cooperative content placement and delivery problem as an integer linear programming (ILP) problem to minimize the average downloading latency. A Lagrangian relaxation algorithm was then designed to decouple the original problem into two smaller subproblems which can be solved more efficiently. Additionally, the content delivery problem was also formulated and solved by the Hungarian algorithm. Most researches on cooperative caching assume the static popularity; the joint consideration of the cooperation and learning of the time-varying popularity still requires further investigation. Song at el. [104] explored the content caching problem with an unknown popularity distribution. They incorporated the learning of the popularity distribution, and then jointly optimized the content caching, content sharing and cost of content retrieving. D. Coded Caching In a traditional switching network, the network node forwards packets one after another: two packets are present in the node at the same time; one of the two packets is forwarded while the other one is queued even if both are headed for the same destination. This traditional packet forwarding mechanism requires separate transmissions and hence decreases the network efficiency. Network coding is a technique which merges two separate messages into one coded message and forwards them to the destination. After receiving the coded message, the network node separates them into two original messages. To enable the network coding technique, transmitted data are encoded at network nodes and then decoded at the destinations. Hence, the network coding technique requires fewer transmissions to transmit all the data. However, this scheme requires coding and decoding processes, and hence incurs more processing overheads to the network nodes. The complexity of network coding can be lowered by efficient packet transmissions [105]. In network coded caching, files in the file library are usually divided into coded packets and then any linear combination of these code packets can reconstruct the entire original object [43]. For example, the file library has the file C, which is divided into C1 ⊕ C2. Owing to the cache storage limitation, a user, who requests file C for the first time, only caches packet C1 after having received file C. When the user requests the same file C for the second time, the BS only needs to transmit C2 to the user. On the contrary, in uncoded caching, file C has to be transmitted for both the first and second time. Therefore, coded caching helps reduce network traffic (C + C2 < C + C). Maddah-Ali and Niesen [106] jointly optimized the caching and coded multicast delivery and demonstrated that the joint optimization problem can improve the caching gain when the demand for the cached content is uniformly distributed. They further showed the near-optimal performance of coded caching achieved by a random caching scheme [107]. They also presented that caching gain can be exploited from coded multicast transmissions in [108]. They proposed in [108] a decentralized coded caching scheme and discussed how to handle scenarios with asynchronous user demands, nonuniform content popularity, and online cache updating. Most works only consider the single layer coded caching, Karamchandani et al. [109] proposed a hierarchical coded caching scheme by considering a two-layer hierarchical cache. They first utilized the coded caching schemes in each layer and then combined the two layers by providing coded multicasting opportunities across different layers. E. Probabilistic Caching Different from wired networks with fixed and known topologies, wireless networks face the uncertainty about which user will connect to which BS due to undetermined user locations and the variance of user requests. Caching in wireless networks becomes more complex when a user moves from one cell to another during the content delivery. An approach to solve this problem is to employ a probabilistic caching policy in which the content can be placed in the caches according to some random distributions. To reflect the uncertainty, Blaszczyszyn and Giovanidis [85] modeled the user locations as a spatial random process. They optimized the probability of each content being cached at each BS with the aim to maximize the cache hit probability. They also demonstrated that the widely used greedy algorithm, which caches the most popular files, cannot always guarantee optimization in a general network unless no BS coverage overlaps exist. Ji et al. [43] discussed the random caching strategy with the aid of coded multicasting in D2D networks where UEs are uniformly distributed in a grid network and can share contents with each other. They pointed out that the drawback of deterministic caching is that the optimal cache placement cannot always be implemented without errors especially when D2D caching is considered. They demonstrated that their random caching strategy, where users make arbitrary requests for files, performs better as the network size grows. F. Game Theory based Caching In wireless networks, multiple parties coexist, including the service providers (SPs) who provide contents, mobile network operators (MNOs) who manage the radio access networks (RANs), and mobile users who consume different contents. When applying a specific caching strategy, the benefits of different parties could conflict with each other. For example, bringing more contents to BSs is beneficial to users while increasing the cost of MNOs due to the additional storages and power consumption. Since each party only cares about its own profit, competitions among them are unavoidable. To effectively cope with the competition and guarantee high overall user experience, game theory is adopted to analyze the interactions among these parties. An auction game is suitable to characterize the competition among SPs. In this setting, the cache storages are considered as objects to be auctioned and the price should be paid to MNOs by SPs. MNO should be the one who is in charge of the auction process. Hu at el. [110] applied game theory to analyze how the selfishness of different parties may impact the overall caching performance by considering the relations and interactions among different parties. They considered two scenarios including the SBS caching and D2D caching. In the former one, multiple SPs aim to cache their own contents into SBSs with limited cache storages, and an auction game is proposed to solve the problem. For the latter one, they adopted a coalition game to analyze how a cooperative group can be formed to download contents together. They extended their work by introducing the concept of caching as a service in [12], where they utilized the wireless network virtualization technology and each SP has to pay for the SBS cache storages owned by the MNO. A multi-object auction mechanism was proposed to characterize the competition among SPs. Since all SPs tend to cache more contents to improve the service performance, they intented to act as the bidders and compete for limited cache storages. The utility function is related to the average content download file. Their mechanism was carried out by a series of auctions, which are solved by the market matching algorithm [111]. Hamidouche et al. [112] assumed that all SBSs in a cache enabled small cell network could choose their backhaul link types among wired links, mmW and sub6 GHz bands. They formulated a backhaul management minority game where the SBSs are the players and independently decide their backhaul link types and the numbers of files to download and cache from the MBS without sacrificing the current requests’ QoS. The characteristic of a minority game is that players prefer the action selected by the minority group. The existence of a unique Nash equilibrium was then proved. By considering the social ties among UEs, Hamidouche et al. [113] utilized the game theoretic approach to determine the content placement strategies to SPs. A many to many matching game was formulated between SPs and SBSs, where each file in SPs can be matched to a set of SBSs. SPs specify their preferences based on the average file download delay while SBSs prefer to store more popular files. The stable solution can be obtained by iteratively update the matching solution according to SPs’ and SBSs’ preferences, until neither of them can find a better preference. G. Summary and Discussion In this section, we survey several caching schemes and compare their pros and cons, including proactive caching, distributed caching, cooperative caching, coded caching, probabilistic caching and game theory based caching. Proactive caching, contrary to reactive caching, caches the contents prior to receiving the requests. It helps improve the caching efficiency by pre-downloading popular contents during off-peak hours and serving users during peak hours. Hence, accurate prediction of user demands plays an important role in proactive caching. Most works characterize the user demands by estimating user mobility patterns, content popularity distributions and user social relationships via machine learning and big data analytics. Further research is still required to provide higher estimation accuracy. Distributed caching, contrary to centralized caching, does not rely on the central controller to make caching decisions. Hence, it avoids the great burden on the single control node. In distributed caching, the caching strategies are usually made based on the local information (e.g., user requests and CSIs) and that from the neighboring ones, and hence can be addressed in the parallel and distributed manner. Most works investigate how to utilize the local and neighboring information to solve the content placement problem. However, unlike centralized caching, which owns a global view of all network status, distributed caching usually cannot obtain the optimal solutions. Hence, designing distributed caching strategies with performance guarantee still requires further research. Cooperative caching allows multiple caches to share contents with each other, and hence it can alleviate the shortage of caching storages. In cooperative caching, a cache is usually aware of the caching status of its neighboring caches by exchanging information; this may incur significant signal overheads. Most works on cooperative caching do not consider these overheads. Hence, further research is needed to minimize the content retrieval latency while minimizing the overhead. Coded caching allows files to be divided into coded chunks with coding, which are then cached in different cache nodes. Users obtain different coded chunks from different cache nodes and then decode these chunks into a complete requested file. Coded caching is usually coupled with a multicast technique to provision content delivery. However, coded caching aggravates the network system complexity and introduces more processing at network intermediary and terminal nodes. This drawback of coded caching is neglected by most works and hence needs further investigation. Probabilistic caching allows contents to be cached at different caches with different probabilities. It is proposed to address the uncertainty of wireless networks caused by varying wireless channel conditions and user mobilities. The objective of probabilistic caching is usually to maximize the cache hit probability by optimizing the probabilities with which contents are cached at different locations. Most works assume the deterministic network status and so probabilistic caching is still an ongoing research. Game theory based caching investigates the interactions of multiple coexisted parties (e.g., service providers and mobile network operators). Each party selfishly optimizes its own benefits which may conflict among different parties. A typical case is the auction game where the service providers act as the bidders and compete for the limited caching storages in order to improve their own caching performances. Most works only consider the non-cooperative game, and so the cooperative game requires further investigation.