Memory Placement in Network Compression: Line and Grid Topologies

Mohsen Sardari, Ahmad Beirami, Faramarz Fekri
School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332
Email:{mohsen.sardari, beirami, fekri}@ece.gatech.edu

***Abstract*—Enabling intermediate nodes in networks with the capability of storing the past communication can offer several benefits. Recently, we have shown that by utilizing memory at intermediate nodes, one can compress the data stream sent from the source node with superior performance compared to the conventional end-to-end compression of individual sequences destined to each client. In other words, memorization or learning of past traffic at intermediate nodes provide extra compression gain. This gain comes from the fact that utilizing previous traffic shared between the source and intermediate nodes with memory helps to close the gap between the compression performance of universal compression techniques and entropy of each individual sequence. The gain of data traffic reduction depends on the number of memory units and their locations. Since in practical scenarios only a select number of nodes have the storage and computational capability to function as a memory unit, it is important to find the optimal location for such nodes. Furthermore, memory placement in the network poses some challenges to traditional shortest path routing algorithms, as the shortest path is not necessarily minimum cost route in networks with memory. In this paper, we investigate the memory placement problem and routing algorithms for networks featuring memory units for network compression. We derive the optimal memory placement strategy on line and grid networks. We further demonstrate how conventional routing algorithms should be modified when there are memory units in the network.**

I. Introduction

Recent studies have demonstrated that memory deployment in the network can result in reduction of network traffic [1]–[3]. Memory units are nodes in the network that can selectively store parts of the previous traffic passing through them. The memorized traffic can in turn help in the learning of the statistics of the unknown source model that generated the data. This learning of the statistics can be used in the memory-assisted universal compression of individual packets in order to suppress the redundancy in the universal compression [4].

In [5], we demonstrated that universal compression of finite length sequences has significant gap from the entropy of the sequence. However, using memory can significantly improve the compression performance for finite-length sequences and potentially close the gap. Motivated by this, in [4], [6], we introduced memory-assisted universal source coding and studied the fundamental gain g (defined as the ratio of the average codeword length resulting from the universal compression without memory to that of with memory) of memory-assisted source coding from an information-theoretic point of view where both the encoder and the decoder have access to shared data (memory) of length m from the unknown source.

This material is based upon work supported by the National Science Foundation under Grant No. CNS-1017234.

The deployment of memory units in the network gives rise to a number of questions and also brings some new challenges. In network compression, every traffic from source to the memory node benefits from the gain g. Then, the first question is how much gain in terms of reduction of the total traffic in the network one should expect. In [7], we defined a network-wide gain $\mathcal{G}$ for a general network topology as a function of the number of memory units in the network, and the fundamental gain g of memory-assisted source coding. We investigated the scaling behavior of the network-wide gain with the number of nodes in the network in a random graph, where we showed significant improvement is achievable even when the number of memory units is a small percentage of the total number of nodes in the network. In [3], we extended our study to Internet-like power-law network graphs and characterized the network-wide gain. Our results indicate that significant improvement can be obtained if one applies compression on the packets traveling in the Internet today.

However, as mentioned above, introducing memory units in the network brings some new challenges that need to be addressed for them to be useful in practice. In this paper, we try to address some of these challenges. In particular, we aim at answering two fundamental (and related) questions regarding memory-assisted network compression.

1) In practical scenarios where only a select number of nodes are capable of memorization and data processing, i.e., only certain number of nodes in the network are memory units, what would be the best strategy to choose the memory units? In other words, where should the memory units be placed in the network?
2) After the locations of the memory units are fixed, what would be the optimal routing algorithm for memory-assisted universal source coding? In other words, what is the best strategy to route packets between the source and destination nodes given the network topology, the location of the memories, and the fundamental gain g of memorization?

We stress that this problem is different from the en-route cache placement problem studied in the context of content caching [8], as we study the optimal placement without the en-route constraint and study the consequences of this extension.

The rest of this paper is organized as follows. In Section II, we present the necessary notations and the problem setup for

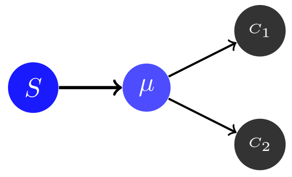

Fig. 1. The basic source, memory, destination configuration.

memory assisted source coding in network. In Section III, we demonstrate the challenges of memory placement and routing when certain number of nodes are capable of data processing as well as memorization. In Section IV, we state the memory placement problem for memory-assisted network compression and present our results for the class of line and grid networks. Finally, Section V concludes this paper.

II. Problem Setup

The basic scenario for the memory-assisted universal compression is illustrated in Fig. 1. The source S (content server) is assumed to be a parametric information source [5] that generates sequences x^n of length n. Assume that client C_1 requests a set of finite-length sequences from the content server. Due to finite-length nature of the compression problem, traditional universal compression techniques (when applied to individual sequences) would achieve a compression rate which can be significantly larger than the entropy of the sequence [5].

However, if the node μ chooses to memorize the sequences destined to C_1 and accumulate a memory of total length m, this memory can be used for more effective compression of a new (unseen) sequence x^n that is to be transmitted to C_2. Denote $\mathbf{E}l_{n|m}(X^n)$ as the expected code length for a sequence of length n when both encoder (S) and the decoder (μ) have access to a memorized sequence of length m. Let $\mathbf{E}l_n(X^n)$ be the expected code length without memory. Then, we define the fundamental gain of memorization as the following [4], [6]:

$$g(n,m) \triangleq \frac{\mathbf{E}l_n(X^n)}{\mathbf{E}l_{n|m}(X^n)}. \tag{1}$$

In [4], we further characterized $g(n,m)$ for different sources as a function of the sequence length n and the memory length m and derived a closed form lower bound on $g(n,m)$. We clarify that there are two phases in the memory-assisted compression. The first is the memorization phase in which we may assume all memory units have accumulated some sufficiently long sequence from the source. This phase is realized in actual communication networks by observing the fact that a sufficient number of clients may have previously retrieved finite-length sequences from the server such that, via their routing, each of the memory units has been able to memorize the source. The compression happens in the second phase where, knowledge of the previous sequences helps for better compression of the new sequences. In this paper, we assume all memories have observed a total of m symbols from the source and hence we treat $g(n,m)$ as a given fixed constant. Thus, we focus on solving the memory placement and routing for a given fixed $g = g(n,m)$.

We represent a network by a connected graph $G(V,E)$ where V is the set of N nodes (vertices) and $E = \{uv : u, v \in V\}$ is the set of edges connecting nodes u and v. Associated with each edge uv is a cost (or length) c_{uv}, which is the cost of transferring one unit of flow along that edge. A simplified bit$\times$hop measure is equivalent to the case where $c_{uv} = 1$ for all edges. We consider a single source S which is the content server, and a set of memories $\boldsymbol{\eta} = \{\mu_i\}_{i=1}^M$ chosen out of the N nodes.

In a network with source S and a set of destinations $\mathbf{D} = \{D_i\}_{i=1}^N$, let f_D be the flow destined to $D \in \mathbf{D}$. The distance between any two nodes u and v is shown by $d(u,v)$. The distance is measured as the sum of the costs of the edges in the shortest path between two nodes. As we will see later, introducing memories to the network will change the lowest cost paths from the source to destinations, as there is a gain associated with the $S-\mu$ portion of the path. Therefore, we have to modify paths accounting for the gain of memories. Accordingly, for each destination D, we define *effective walk*, denoted by $W_D = \{S, u_1, \ldots, D\}$, which is the ordered set of nodes in the modified (lowest cost) walk between the source and D. We define $\mu_{D_i} = \arg\min_{\mu\in\boldsymbol{\eta}}\{\frac{d(S,\mu)}{g} + d(\mu, D_i)\}$. Then, we partition the set of destinations as $\mathbf{D} = \mathbf{D}_1 \cup \mathbf{D}_2$, where $\mathbf{D}_1 = \{D_i : \exists \mu_{D_i} \in W_{D_i}\}$ is the set of destinations observing a memory in their effective walk. The total flow $\mathcal{F}$ is then defined as

$$\begin{aligned}\mathcal{F} \;=\; & \textstyle\sum_{D_i\in\mathbf{D}_1}\left(\frac{f_{D_i}}{g}d(S,\mu_{D_i}) + f_{D_i}d(\mu_{D_i}, D_i)\right) + \\ & \textstyle\sum_{D_j\in\mathbf{D}_2} f_{D_j} d(S, D_j)\end{aligned}. \tag{2}$$

For simplicity, we assume $f_D = 1$ for all destinations. In other words, we assume that after traditional end-to-end compression, the length of the data to be transmitted is one bit. This has no consequence on our analysis but simplifies the notation.

In a general network where every node can be a destination, we define a generalized network-wide gain of memory deployment as a function of memorization gain g, as follows:

$$\mathcal{G}(g) = \frac{\mathcal{F}_0}{\mathcal{F}}, \tag{3}$$

where $\mathcal{F}_0$ is the total flow in the network with end-to-end universal compression of sequence without using memory, i.e.,

$$\mathcal{F}_0 = \sum_{D\in\mathbf{D}} d(S,D).$$

Note that with the above definition, the gain $\mathcal{G}(g)$ is the benefit offered by the memory-assisted compression over the end-to-end universal compression without memory.

III. Hardness of Optimal Deployment of Memory and Routing in Network Compression

Let the total number of memory units be M. The goal of the memory deployment is to find the best set of M out of

N vertices in the network such that $\mathcal{G}(g)$, i.e., the network-wide gain of memory, is maximized. In general, this is a hard problem as we summarize below.

A. *Hardness of Memory Deployment*

It can be shown that the memory placement is equivalent to the well-known k-median problem. Hence, the memory deployment problem on a general graph is an NP-hard problem. However, a solution to the deployment problem can be obtained for certain network topologies, which can be helpful in finding approximate solutions for general networks. In this section, we demonstrate the challenges of the memory deployment problem by considering the class of line networks.

Another challenge in finding the gain $\mathcal{G}(g)$ comes from the difficulty of finding minimum cost paths in a network. Below we summarize these challenges and introduce a modified routing algorithm that can help finding the effective walks in a network and hence calculating $\mathcal{G}(g)$.

B. *Modified Shortest Path Routing for Memory-Assisted Compression*

In a regular network without memory nodes, the shortest path problem can be solved using the well-known Bellman-Ford algorithm which relies on the so-called principle of optimality: if a shortest path from u to v passes through a node w, then the portion of the path from w to v is also a shortest path. It is important to note that, there is another statement for principle of optimality as follows: if w is a node on the shortest path from u to v, this knowledge implies the knowledge of the shortest path from u to w. This latter statement only holds in networks without memory and in general is *not* true for networks with memory (Fig. 2). Therefore, the shortest path problem requires more attention in networks with memory. The well-known routing algorithms like Dijkstra's algorithm, in their original form are not applicable to networks with memory. This limitation is due to the fact that when the algorithm runs into a memory node on the path, all the previous edges' costs along the path should be divided by g, and hence needs to recalculate the whole path.

Although, having memory units in a network changes the shortest paths dramatically (an example is shown in Fig. 2), the principle of optimality sill holds and this will enable us to find the shortest path using the well-known Bellman-Ford algorithm which is in effect the repeated application of the principle of optimality. The Bellman-Ford algorithm is used in distance-vector routing protocols. The distributed version of the algorithm is used within an Autonomous System (AS), a collection of IP networks typically owned by an ISP. While Bellman-Ford algorithm solves the shortest path problem in networks with memory and the solution for routing within an AS is readily provided by this algorithm, the more efficient Dijkstra's algorithm is more used in practice. The Dijkstra's Algorithm is widely used in network routing protocols, most notably *IS-IS* and *OSPF* (Open Shortest Path First) and hence it is important to visit the challenges of finding the shortest paths in networks with memory using the Dijkstra's algorithm.

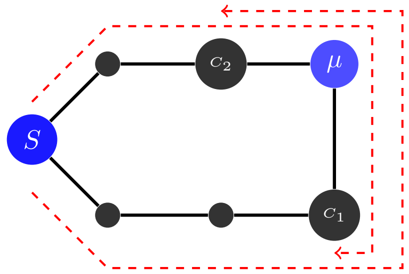

Fig. 2. Routing in networks featuring memory. Memories change the shortest paths dramatically but the principle of optimality sill holds.

Here, we present a modified version of Dijkstra's algorithm that finds the effective walk from all the nodes in a network to a destination D, in a network with a single memory. Iterating over all nodes will provide the effective walk between every pair of nodes in the network. To handle the memory node, we define a node-marking convention by defining a set $\mathcal{M}$ which contains the marked nodes. We say that a node is marked if it is either itself a memory node, or a node through which a compressed flow is routed. The modified Dijkstra's algorithm starts with finding a node ν closest to node D. Then, we iteratively update the effective distance of the nodes to D. For nodes not directly connected to D, the distance is initialized to infinity and then iteratively updated as in Dijkstra's algorithm. After finding the effective distance between every pair of vertices via the modified Dijkstra algorithm, we can calculate $\mathcal{F}$ and then $\mathcal{G}$.

IV. Optimal Deployment of Memory on Line and Grid Networks

A. *Memory Deployment on Line Networks*

Consider a line network with the source node S placed at one end of the line and the destinations placed along the line as shown in Fig. 3. Therefore, we have a total number of N nodes on the line and the total length of the line is N hops. As mentioned before, we assume traditional universal compression would give one unit of flow, to be sent to each destination. We consider the deployment of M memory units on the line such that the memory μ_i is placed at t_i from the source, as shown in Fig. 3. We find t_i's such that total flow $\mathcal{F}$ is minimized (or equivalently, $\mathcal{G}(g)$ is maximized). The solution to the special case of "en-route" memory deployment on line networks is studied in [8]. En-route memories are those which are only located along routes from source to receivers. An en-route memory intercepts any request that passes through it along the regular routing path. The solution to the en-route memory placement problem as discussed in [8] is $t_i = \frac{1}{M} \quad \forall \mu_i$.

However, the memory deployment problem for network compression on a line network is more challenging. The difficulty comes from the fact that each memory can serve some of the destinations closer to source than the memory itself. In other words, the shortest effective walk from source to the destinations is not necessarily the same as the shortest

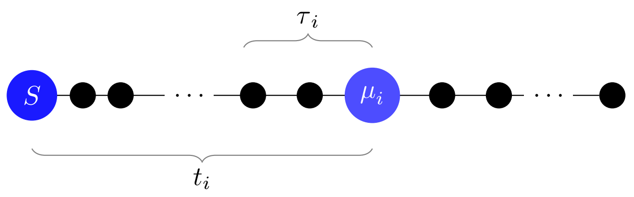

Fig. 3. The placement of memory units on a line network: the source node S is placed at one end of the line and the i-th memory is placed at t_i from the source and τ_i is its left-coverage.

hop distance. As shown in Fig. 3, for a memory μ_i located at t_i, there is a left-coverage hop-length of τ_i towards the source to cover the destinations on the left side of the memory. The following lemma shows how t and τ change for different values of g.

Lemma 1 *For the simple case of $M = 1$ and a line of hop-length N, the optimal memory location t and coverage τ is given by*

$$\begin{aligned} t &= \frac{2g}{3g+1}N + O(1), \\ \tau &= \frac{g-1}{3g+1}N + O(1). \end{aligned} \tag{4}$$

As shown in Fig. 4, as the gain g increases, the memory is placed on 2/3 distance from the source and the left coverage approaches 1/3.

Lemma 2 *The gain of single memory placement on line is $\mathcal{G}(g) = \frac{(3g+1)^2}{3g^2+10g+3}$, and for $g \gg 1$ we have*

$$\mathcal{G} \approx 3. \tag{5}$$

Proof: The proof is immediate from Lemma 1 and the fact that for line $\mathcal{F}_0 = 1/2$. ■

Following the results of deployment of a single memory on line, we can extend the result and solve for the general problem of deployment of M memory nodes.

Theorem 3 *Consider deployment of M memories on a line where memory μ_i is placed at t_i with τ_i left-coverage. Then,*

$$\begin{cases} t_i & \approx \frac{m}{M}N + O(1) \\ \tau_i & \approx \frac{g-1}{2gM}N + O(1) \end{cases}. \tag{6}$$

By definition, $t_0 = \tau_0 = 0$. Furthermore, $\mathcal{G}(g) = \frac{2g^2M}{2g(M+1)+g^2+1}$, and for $g \gg 1$ we have

$$\mathcal{G} \approx 2M. \tag{7}$$

Proof: Proof is provided in the appendix. ■

Next, we extend the result of memory deployment for a line network to grid topologies.

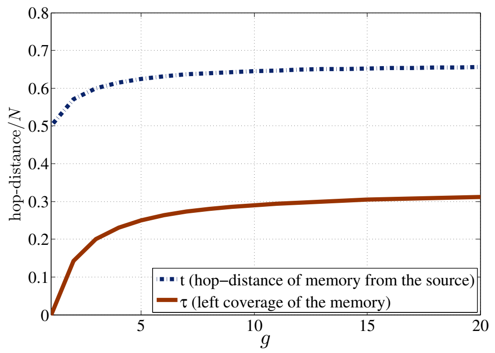

Fig. 4. Variations of t and τ vs. g for a line network.

B. Memory Deployment on a Grid Network

In this section, we extend our study to the analysis of memory placement in grid networks. Let G be a $(2L+1)\times(2L+1)$ grid network with a single source placed at $(0,0)$, as shown in Fig. 5. Again, the goal is to obtain the optimal placement of M memories in this topology and derive the corresponding network-wide memorization gain, defined in Sec. II. Note that in this topology, since the cost is measured in bit$\times$hop, the geometry of the problem implies that ℓ_1 distance would be the proper interpretation of the distance metric. Therefore, for example, in Fig. 5, all the nodes located on the diamond square share the same distance from the source S located at the center point. We look at two memory placement strategies on the grid. First, we consider the scaling of the gain under a uniform memory placement. Then, we look at a case where all the memories are placed at distance t from th source.

Now, consider the uniform placement of the memory units on the grid. For simplicity let the number of memories be $M = k^2$. Note that since our goal is to obtain the scaling behavior of the gain with the number of memories, as the grid size and M grow large, this assumption is not restrictive. The memories could be placed uniformly on the grid being at distance $\frac{L}{k+1}$ hops from each other. Note that the uniform memory placement provides a lower bound on the network-wide gain since it is not the optimal placement strategy. It is straightforward to demonstrate that in this case, due to symmetry, the problem breaks down to solving the routing problem in a square whose corner points are all memory units. We need to determine which of the corner-point memory units would be the best choice for serving the node inside the square given that one of the corner points are closest to the source with distance r hops, two located at distance $r + k$ from the source, and another corner at distance $r + 2k$ from the source depending on which quarter the corners are located at. Our main result on uniform placement on the grid is the following theorem.

Theorem 4 *In a grid network, where the number of the memory units is M, as the total number of nodes grows, i.e.,*

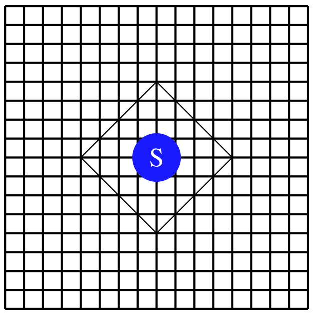

Fig. 5. Grid Network with the source S is placed at $(0,0)$. All the nodes located on the diamond square have the same distance from S.

$N \to \infty$, the network-wide gain is lower bounded by

$$\mathcal{G} \geq g \frac{\sqrt{M}}{\sqrt{M} + 2(g-1)}. \tag{8}$$

According to this result, even with a finite number of memory units, a non-vanishing network-wide gain is obtained even as the total number of nodes grows large. Further, note that when the number of memory elements also grows large (i.e., $M \to \infty$), the network-wide gain approaches its upper limit g. Thus, for large M, the uniform memory deployment on a grid network is near-optimal.

Next, we analyze a different strategy. We assume that all memories are *uniformly* at distance t hops from the source. Again, similar to the analysis of the line network in the previous section, define τ to be the distance (in bit×hops) that a memory serves backward as discussed in the previous section. Our goal is to obtain the optimal placement under this assumption, i.e., to find the optimal t and τ. Intuitively, based on the geometry of the problem along with the symmetry of the memory placement, this placement is near optimal when the number of memory units is very small. From Thm. 4, when the number of the memory units grows, the uniform placement of memories will be optimal.

As shown in Fig. 5, the number of nodes at distance h hops from the source is $4h$. Therefore, we can extend the analysis for the line networks with the difference that the density of the nodes at distance h being $4h$ instead of 1 in the line networks. When there is no memory unit, we have $\mathcal{F}_0 = \int_0^1 4x^2 \,\mathrm{d}x = \frac{4}{3}$. By placing memories at distance t from the source we have

$$\begin{aligned} \mathcal{F} &= \int_0^{t-\tau} 4x^2 \,\mathrm{d}x \\ &+ \int_0^{1-t} 4(t+x)x \,\mathrm{d}x + \frac{t}{g}\int_t^1 4x \,\mathrm{d}x \\ &+ \int_0^{\tau} 4(t-x)x \,\mathrm{d}x + \frac{t}{g}\int_0^{\tau} 4(t-x) \,\mathrm{d}x. \end{aligned} \tag{9}$$

To find the optimal t and τ, we take the derivative of $\mathcal{F}$ with respect to t and τ and equate that to zero. In a result similar to Lemma 2, for $g \gg 1$ we have that $\mathcal{G} \approx 3.3$. As we mentioned before, when the number of memories can grow large, the uniform placement is a better strategy.

V. Conclusion

In this paper, we investigated the memory placement and routing algorithms for networks where intermediate nodes are capable of the memorization of the past communication. While the placement problem is hard in general, we derived the optimal memory placement strategy on line networks and also presented results for grid topologies. We further demonstrated how routing should be modified when there are nodes that memorize the past traffic.

Appendix

Sketch of Proof of Lemma 1: By approximating Fig. 3 with a continuous line and normalizing the length, we have

$$\begin{aligned} \mathcal{F} = \int_0^{t-\tau} x \,\mathrm{d}x \quad &+ \quad \left(\frac{t(1-t)}{g} + \int_0^{1-t} x \,\mathrm{d}x\right) \\ &+ \quad \left(\frac{t\tau}{g} + \int_0^{\tau} x \,\mathrm{d}x\right) \end{aligned} \tag{10}$$

The first term in (10) is the flow to all points on the line not covered by memory. The second term is for the right coverage of memory and the third term accounts for the left coverage of memory (τ). The result in (4) follows by taking the derivative of $\mathcal{F}$ and equating to zero, i.e., $\frac{\partial}{\partial t}\mathcal{F} = 0$ and $\frac{\partial}{\partial \tau}\mathcal{F} = 0$. ■

Sketch of Proof of Thm. 3: Similar to the proof of Lemma 1, we can write

$$\begin{aligned} \mathcal{F} \;=\; &\sum_{m=1}^{M} \left[\frac{t_i}{g}\tau_i + \int_0^{\tau_i} x \,\mathrm{d}x \right. \\ &+ \left. \frac{t_i}{g}(t_{m+1} - \tau_{m+1} - t_i) + \int_0^{t_{m+1}-\tau_{m+1}-t_i} x \,\mathrm{d}x \right]. \end{aligned}$$

Again, by taking the derivative of $\mathcal{F}$ with respect to t_i and τ_i and solving the system of equations we arrive at

$$\left\{ \begin{array}{lcl} t_i & \approx & \frac{t_{i+1}+t_{i-1}}{2} \\ \tau_i & = & \frac{g-1}{2g}(t_i - t_{i-1}) \end{array} \right. , \tag{11}$$

where (11) results in a tridiagonal matrix which in turn results in (6) for large M. Further, (7) follows from (6) and $\mathcal{F}_0 = 1/2$ for line networks. ■

References

[1] Z. Zhuang, C.-L. Tsao, and R. Sivakumar, "Curing the amnesia: Network memory for the internet," Tech Report, 2009. [Online]. Available: http://www.ece.gatech.edu/research/GNAN/archive/tr-nm.pdf

[2] A. Anand, A. Gupta, A. Akella, S. Seshan, and S. Shenker, "Packet caches on routers: the implications of universal redundant traffic elimination," *SIGCOMM*, vol. 38, pp. 219–230, 2008.

[3] M. Sardari, A. Beirami, and F. Fekri, "Memory-assisted universal compression of network flows," in *IEEE INFOCOM*, Orlando, FL, March 2012, pp. 91–99.

[4] A. Beirami, M. Sardari, and F. Fekri, "Results on the fundamental gain of memory-assisted universal source coding," in *IEEE International Symposium on Information Theory (ISIT)*, Boston, MA, July 2012.

[5] A. Beirami and F. Fekri, "Results on the redundancy of universal compression for finite-length sequences," in *IEEE Intl. Symp. Info. Theory (ISIT)*, July 2011, pp. 1504–1508.

[6] ——, "Memory-assisted universal source coding," in *Data Compression Conference (DCC)*, Snowbird, UT, April 2012, p. 392.

[7] M. Sardari, A. Beirami, and F. Fekri, "On the network-wide gain of memory-assisted source coding," in *2011 Information Theory Workshop (ITW)*, 2011, pp. 476–480.

[8] P. Krishnan, D. Raz, and Y. Shavitt, "The cache location problem," *IEEE/ACM Transactions on Networking*, vol. 8, pp. 568–582, 2000.