When Locally Repairable Codes Meet Regenerating Codes — What If Some Helpers Are Unavailable

Imad Ahmad, Chih-Chun Wang; {ahmadi,chihw}@purdue.edu
School of Electrical and Computer Engineering, Purdue University, USA

***Abstract*—Locally rapairable codes (LRCs) are ingeniously designed distributed storage codes with a (usually small) bounded number of helper nodes participating in repair. Since most existing LRCs assume *exact repair* and allow *full exchange of the stored data* ($\beta = \alpha$), they can be viewed as a generalization of the traditional erasure codes (ECs) with a much desired feature of local repair. However, it also means that they lack the features of functional repair and partial information-exchange ($\beta < \alpha$) in the original regenerating codes (RCs). Motivated by the significant bandwidth (BW) reduction of RCs over ECs, existing works by Ahmad *et al* and by Hollmann studied "locally repairable regenerating codes (LRRCs)" that simultaneously admit all three features: local repair, partial information-exchange, and functional repair. Significant BW reduction was observed.**

One important issue for any local repair schemes (including both LRCs and LRRCs) is that sometimes designated helper nodes may be temporarily unavailable, the result of multiple failures, degraded reads, or other network dynamics. Under the setting of LRRCs with temporary node unavailability, this work studies the impact of different *helper selection methods*. It proves, for the first time in the literature, that with node unavailability, all existing methods of helper selection, including those used in RCs and LRCs, are strictly repair-BW suboptimal. For some scenarios, it is necessary to combine LRRCs with a new helper selection method, termed *dynamic helper selection*, to achieve optimal BW. This work also compares the performance of different helper selection methods and answers the following fundamental question: *whether one method of helper selection is intrinsically better than the other?* for various different scenarios.

I. Introduction

Erasure coding (EC) is efficient in terms of reliability vs. redundancy tradeoff in distributed storage. An (n,k) MDS code can be used to spread a file over a network of n nodes to tolerate $(n-k)$ simultaneous failures. When a node fails, it is repaired by accessing any k *surviving nodes*, downloading all the coded data, and then reconstructing the original data. As a result, we say that the repair of EC involves "full information exchange" and "exact repair". Regenerating codes (RCs) [3] were proposed to reduce communication during repair, termed the *repair-bandwidth* (BW). The key ideas that allow RCs to decrease repair-BW are: (1) contact as many nodes as possible during repair or in other words $d = n-1$ nodes, termed *helper nodes*, (2) download only a partial fraction of the data ($\beta < \alpha$) as opposed to full information-exchange ($\beta = \alpha$), and (3) allow *functional repair* instead of exact repair. These 3 ideas enable significant BW reduction of RCs over EC [3].

Another type of distributed storage codes is the locally repairable codes (LRCs) [5], [7], [10]–[12] that use a small number of helper nodes d during repair, which is in contrast with RCs that were originally designed for large d (i.e., $d \geq k$). A closer look at the properties of LRCs shows that LRCs resemble EC in that they operate with $\alpha = \beta$ and under exact repair, except that ECs access $d = k$ helpers while LRCs use a smaller d value (usually $\ll k$). For that reason, LRCs can be viewed as a generalization of EC with a much desired feature of local repair (small d). Inspired by the BW reduction of RCs over EC, it is thus natural to propose *locally repairable regenerating codes* (LRRCs), that simultaneously admit all three features: local repair ($d < k$), partial information-exchange ($\beta < \alpha$), and functional repair.

A critical component of any LRRC design is the underlying helper selection policy. Although not explicitly called LRRCs in [3], the results in [3] analyze RCs for arbitrary $d < n-1$, the RC thus becoming *locally repairable*, while assuming the newcomer "blindly chooses the d helpers", termed *blind helper selection* (BHS).[1] The works in [1] (the full version in [2]) and in [6] are the first works to study LRRCs with intelligent (non-blind) helper selection. For the special cases of $k = n-1$ and $\alpha = d\beta$ or $\alpha = \beta$, [6] upper bounds the achievable performance of any LRRCs regardless whether an intelligent or a BHS scheme is used. [1] answers the question: under what (n,k,d) values can intelligent helper selection strictly improve the performance of LRRCs when compared to the BHS-based LRRCs in [3]. This question was answered in [1] for any arbitrary (n,k,d) parameters. A new scheme termed the *family-repair* scheme was also devised [1] that successfully removes the $d \geq k$ limitation in [3] and demonstrates superior performance (very small repair BW) while admitting local repair (i.e., $d \ll k$) and partial information-exchange.

Despite the promising preliminary results, the LRRCs in [1], [6] do not consider the following practical issue: Because of multiple failures or degraded reads or other network dynamics, some designated helper nodes may be temporarily unavailable. Therefore, for any locally repairable scheme to work in practice, including both LRCs and LRRCs, it needs to have an alternative set of helpers in case of node unavailability. For LRCs, temporary node unavailability has been studied in [9], [11], [12]. This work studies LRRC with temporary node unavailability and, in particular, its performance under different helper selection schemes.

Our studies are centered around three different classes of helper selection schemes. (i) BHS; (ii) The stationary helper selection (SHS) schemes; and (iii) A new scheme proposed

[1]Because of the use of BHS, the performance of RC in [3] becomes poor when applied to the unintended scenario of $d < k$.

in this work, called *dynamic helper selection* (DHS) and will be discussed in Section II. Suppose for each repair, among all $(n-1)$ surviving nodes, $\leq r$ of them are temporarily unavailable. BHS handles node unavailability naturally, since it blindly treats all helpers equally and can simply choose "any" d helpers out of $(n-1-r)$ "available" surviving nodes. SHS handles node unavailability in the following way. Each node is associated to a helper set of size $(d+r)$. If that node fails, then the newcomer accesses "any" d helpers out of its associated helper set. Since the helper set contains $(d+r)$ nodes to begin with, stationary repair can always find d helpers even with r unavailable nodes. Such a helper selection is *stationary* since the helper set of each node is fixed and does not change over time. *The helper selection policies of almost all existing designs of ECs, RCs, LRCs, and LRRCs [3], [5], [7], [9]–[12] are either BHS or SHS*. For example, if $r=0$ (all nodes being available), then each node in LRCs [5], [10] has a fixed helper set of d nodes. For $r=1$, the LRC design [7], [9], [12] also relies on fixed helper sets of size $(d+r)$ with the additional concept of *multiple-parity nodes per local group*.

Contribution 1: We prove, for the first time in the literature, that BHS and SHS can be strictly BW suboptimal. Specifically, we provide an example with $r=1$ showing that it is necessary to use DHS, which is designed based on a completely different principle, to achieve optimal BW while the performance of BHS/SHS is strictly suboptimal. Furthermore, the DHS scheme in our example is simultaneously Minimum Bandwidth Regenerating (MBR) and Minimum Storage Regenerating (MSR). Such an example demonstrates the benefit of DHS and calls for further research on DHS.

Contribution 2: BHS is the least powerful of the three helper selection policies and can thus be used as a baseline. We study the following fundamental question: Given any (n,k,d,r) value, can we *always* design an SHS or DHS scheme that strictly outperforms BHS? Surprisingly, for many (n,k,d,r) values, the answer is no. I.e., for some (n,k,d,r), even the best SHS or DHS scheme is no better than the simple well-studied BHS [3]. We examine all possible (n,k,d,r) values and prove that for a vast majority of (n,k,d,r) values, we can answer whether SHS or DHS can strictly outperform BHS. The small set of (n,k,d,r) values for which we cannot answer the question is also listed explicitly in this work.

Summary: The contribution of this work is mostly information-theoretic. The deceptively simple example in Contribution 1 sheds new and surprising insights on the fundamental performance limits of different helper selection schemes. The results in Contribution 2 provide valuable case-by-case guidelines whether it is worth spending time to design new SHS or DHS schemes or whether one should simply use the basic BHS. The exact percentage of BW improvement over existing solutions is not the main focus of our exploration.

II. Dynamic vs. Stationary Helper Selection

We will prove that DHS is strictly better than BHS and any SHS scheme by explicitly constructing an example. Consider a network of $n=5$ nodes. Each newcomer accesses $d=2$ helpers out of $(n-1-r)=3$ surviving nodes that are currently available. That is, $r=1$ node may be temporarily unavailable during repair but which node is unavailable may happen arbitrarily. Our goal is that any $k=3$ nodes can recover the original file. Using the notation in [3], this example has $(n,k,d,r)=(5,3,2,1)$. We also define α as the amount of data stored in each node, β as the amount of data sent by each helper, and $\mathcal{M}$ as the size of the original file.

Let us now apply LRCs [7], [12], which use SHS. That is, each node i is associated with a fixed helper set D_i of size $(d+r)=3$. If $r=1$ node of the D_i is temporarily unavailable, then node i can still access d helpers from[2] D_i. Unfortunately, with the constraint of "full information-exchange" and "exact repair" in LRCs, it can be proved (see Proposition 1) that the smallest achievable noramlized BW of any LRCs is still $\frac{d\beta}{\mathcal{M}}=1$ regardless how we design the helper sets $\{D_i\}_{i=1}^{5}$ and the underlying local repair group.

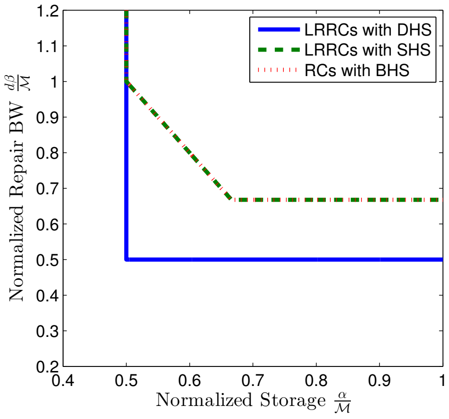

Fig. 1. Storage-bandwidth tradeoff curves of LRRCs with DHS, LRRCs with SHS, and RCs with BHS for $(n,k,d,r)=(5,3,2,1)$.

One may wonder how much we can improve by applying the two important concepts: (i) partial information exchange and (ii) functional repair in [3]. With (i) and (ii), [3] proves that the storage-BW tradeoff curve for RCs with BHS satisfies

$$\min(2\beta,\alpha)+\min(\beta,\alpha)\geq\mathcal{M}, \tag{1}$$

where $\mathcal{M}$ is the file size. A normalized storage-BW tradeoff curve is plotted in Fig. 1. Namely, by using a larger normalized storage size $\frac{\alpha}{\mathcal{M}}=\frac{2}{3}$ rather than $\frac{1}{2}$, we can reduce the normalized BW to $\frac{d\beta}{\mathcal{M}}$ from 1 to $\frac{2}{3}$. Since BHS is used, RCs can access any $d=2$ out of $3=n-r-1$ available nodes and thus naturally handle node unavailability.

In this work, we first generalize the $r=0$ LRRCs in [1] for the case of $r=1$, which combines (i) and (ii) while removing the limitation of $k\leq d$ in RCs. Proposition 1 will later prove that the best tradeoff with (i) and (ii) is still identical to the BHS tradeoff (1). It is thus natural to ask whether we have already reached the fundamental barrier of this problem $(n,k,d,r)=(5,3,2,1)$ or it is because concepts (i) and (ii)

[2]If the unavailable node is not in D_i, then node i chooses "any" d out of $(d+r)$ nodes in D_i and we analyze the worst-case performance in the same way as analyzing BHS. Without loss of generality, we can assume that the unavailable node is always within D_i since the worst-case is used otherwise.

alone are not sufficient. It turns out that it is the latter. To devise a scheme that strictly outperforms the BHS tradeoff curve, *it is essential to add the third ingredient: (iii) DHS*. The proposed LRRCs with DHS has a new tradeoff curve, see Fig. 1, that is guaranteed to be optimal, i.e., no scheme can do better, and strictly outperforms BHS and any SHS.

Note that by combining (i)–(iii), our scheme achieves the corner point $(0.5, 0.5)$ in Fig. 1, which simultaneously minimizes both storage and BW. It is thus the first known scheme that is provably both MSR and MBR with $r = 1$.

We discuss our findings more rigorously in the following.

A. Stationary Helper Selection Schemes Performance

Proposition 1: Consider $(n, k, d, r) = (5, 3, 2, 1)$ and any arbitrarily given SHS scheme with fixed helper sets D_1 to D_5, each having $|D_i| = d+r = 3$ nodes. Even when allowing (i) partial information exchange and (ii) functional repair, the storage-BW tradeoff is again characterized by (1). Namely, LRRCs with SHS is no better than RC with BHS.

Proof: Suppose $D_1 = \{2, 3, 4\}$, which can be done by relabeling the nodes. Now consider D_2 which, by definition, is a subset of $\{1, 3, 4, 5\}$. Since $|D_2| = 3$, we must have $|D_2 \cap D_1| \geq 1$. The intersection set $D_2 \cap D_1$ can either have node 3, node 4, or both. Without loss of generality, say $D_1 \cap D_2 \ni 3$.

Fail node 3 first and repair it by accessing 2 nodes in D_3. Then, fail node 2 and suppose one node $a \in D_2 \backslash \{3\}$ is unavailable. Hence, node 2 will access node 3 and another node $b \in D_2 \backslash \{a, 3\}$ for repair. Next, fail node 1 and assume node 4 is unavailable. Since $D_1 = \{2, 3, 4\}$, node 1 will access nodes 2 and 3 for repair. Now consider a data collector that accesses nodes $\{1, 2, 3\}$ for reconstruction. We now consider the min-cut in the *information flow graph* [3] that separates the source (root) from $\{1, 2, 3\}$. We observe that node 3 will contribute $\min(2\beta, \alpha)$ to the min-cut and node 2 will contribute $\min(\beta, \alpha)$, since node 3 was the helper of node 2. The min-cut value is thus (1). The proof is hence complete. ■

B. Dynamic Schemes Outperform Stationary Schemes

We now present a DHS scheme for $(n, k, d, r) = (5, 3, 2, 1)$ which has the following strictly better tradeoff curve:

$$2 \min(2\beta, \alpha) \geq \mathcal{M}. \tag{2}$$

We describe in the following a code with $\beta = 1$, $\alpha = 2$, and $\mathcal{M} = 4$ that achieves the corner point of (2), also see Fig. 1. The optimality of the proposed code is established in Proposition 3. The scheme consists of two parts. **Part I:** How to choose the helper nodes? **Part II:** What is the coded data sent by each helper after the helpers are decided?

To describe **Part I**, we need the following notation. We say node i is the *parent* of node j if (i) node i was the helper of node j, and (ii) node i has not been repaired since the failure of node j. For example, say node 1 fails and accesses nodes 2 and 3 as helpers. Then node 2 fails and accesses nodes 3 and 4. After the above two repairs, node 3 is a parent of node 1 but node 2 is not since node 2 has been repaired. On the other hand, both nodes 3 and 4 are parents of node 2.

The main idea of the proposed DHS scheme is to choose helpers such that no 3 nodes ever form a "triangle", i.e., we avoid the forming 3 nodes $\{a, b, c\}$ such that a is the parent of both b and c; and b is the parent of c. We term this DHS scheme *a Clique-Avoiding (CA) scheme*.

We now prove by induction that CA is always possible. In the beginning, all nodes are intact and there is no triangle. Suppose there is no triangle after $(t_0 - 1)$ repairs. Suppose also that node a fails at $t = t_0$. Since there was no triangle in time $(t_0 - 1)$, we only need that a does not participate in any triangle *after* the repair. Denote the helper choice for $t = t_0$ by $\{b, c\}$. Therefore, we only need to choose $\{b, c\}$ such that nodes $\{a, b, c\}$ do not form a triangle after repair.

To that end, we observe that out of $n - 1 = 4$ surviving nodes $r = 1$ node is unavailable, and a has 3 nodes to choose $d = 2$ helpers from. Call these nodes $\{i, j, k\}$. Since there is no triangle at $(t_0 - 1)$, among $\{(i, j), (j, k), (i, k)\}$ one of them, say (j, k), is not "connected". I.e., neither j is the parent of k, nor k is the parent of j. We then choose nodes j and k to be nodes b and c. As a result, nodes $\{a, b, c\}$ do not form a triangle after repair. By induction CA is always possible.

Note that the CA scheme needs to know the *repair history* before deciding which 2 of the 3 available nodes $\{i, j, k\}$ to be the helpers, a significant departure from the principle of associating each node a with a *fixed helper set* D_a. Because the CA scheme has to *dynamically* select the helpers based on repair history, we call CA a Dynamic Helper Selection (DHS) policy. Although the most general form of DHS policies may be complicated (since it can use all history information), the CA scheme herein is quite simple and only requires the controller to keep track of the latest parent-child relationship among all nodes. This can be easily done by updating a table and the optimal CA does not need to track the entire information-flow graph of the past.

We now describe **Part II**, which only uses the *binary field* rather than high-order GF(q). **Initialization:** Recall that $\alpha = 2$, $\beta = 1$. and $\mathcal{M} = 4$. Consider a file of 4 packets X_1 to X_4. Initially, we let nodes 1 and 2 store $\{X_1, X_2\}$ and $\{X_3, X_4\}$, respectively. We then let nodes 3 and 4 store packets $\{X_1, X_3\}$ and $\{X_2, X_4\}$, respectively. Finally, let node 5 store coded packets $\{[X_1 + X_2], [X_3 + X_4]\}$. Artificially, we say nodes 1 and 2 are parents of node 3 since we can view node 3 as being failed first and then repaired from nodes 1 and 2. Similarly, we also artificially say that nodes 1 and 2 are the parents of node 4 (resp. node 5). The initialization phase is now complete. One can see that even with the artificially defined parent-child relationship, there is no triangle after initialization. We can thus use the same induction proof to show that CA is always possible after initialization.

The regular repair operations: Suppose node a fails and one other node is temporarily unavailable. We run the CA scheme to find the helpers b and c. Denote the two non-helper nodes by d and e. Each of b and c will send 1 packet to a since $\beta = 1$. The packets are constructed as follows. **Step 1:** Denote the two packets stored in b by $Y_1^{(b)}$ and $Y_2^{(b)}$. Among three candidate packets $Y_1^{(b)}$, $Y_2^{(b)}$, and the binary sum $[Y_1^{(b)} + Y_2^{(b)}]$, choose one, call it Z_b^*, that satisfies simultaneously: (i) if nodes c and d jointly contain 4 *linearly independent* packets, then we require that Z_b^* cannot be expressed as a linear combination of the two packets stored in d. Otherwise, we require that Z_b^* cannot be expressed as a linear combination of the packets stored in c and d; and (ii) is identical to (i) except that we replace d by the remaining node e.

Step 2: Denote the two packets stored in c by $Y_1^{(c)}$ and $Y_2^{(c)}$. Among three candidate packets $Y_1^{(c)}$, $Y_2^{(c)}$, and the binary sum $[Y_1^{(c)} + Y_2^{(c)}]$, choose one, call it Z_c^*, that satisfies simultaneously: (i) Z_c^* cannot be expressed as a linear combination of Z_b^* and the two packets stored in d; and (ii) Z_c^* cannot be expressed as a linear combination of Z_b^* and the two packets stored in e. Node c then sends packet Z_c^* to a.

Proposition 2: Using the CA scheme, we can always find (Z_b^*, Z_c^*) that satisfies the desired conditions. Moreover, under such construction, any $k = 3$ nodes can always reconstruct the original 4 packets X_1 to X_4. Such a binary code construction $(\alpha, \beta, \mathcal{M}) = (2, 1, 4)$ is thus legitimate.

Let us use an example to illustrate our construction. Suppose after initialization, node 3 fails and node 2 is unavailable. Newcomer 3 thus has to access $\{1, 4, 5\}$ for repair. Since node 1 is the parent of both nodes 4 and 5, the CA scheme will avoid choosing $\{1, 4\}$ and $\{1, 5\}$ and select helpers $\{4, 5\}$ instead. I.e., $a = 3$, $b = 4$, and $c = 5$; and $d = 1$ and $e = 2$.

Since node b stores $\{X_2, X_4\}$, the three candidates are X_2, X_4, and $[X_2 + X_4]$. Since node c stores $\{[X_1 + X_2], [X_3 + X_4]\}$, the three candidates are $[X_1 + X_2]$, $[X_3 + X_4]$, and $[X_1 + X_2 + X_3 + X_4]$. Recall that node d stores $\{X_1, X_2\}$ and node e stores $\{X_3, X_4\}$. Out of the $3^2 = 9$ combinations, only the combination $(Z_b^*, Z_c^*) = ([X_2 + X_4], [X_1 + X_2 + X_3 + X_4])$ can simultaneously satisfy [Cond.1] and [Cond.2]. $Z_b^* = [X_2 + X_4]$ will then be sent to node a from node b and $Z_c^* = [X_1 + X_2 + X_3 + X_4]$ will be sent to node a from node c. Newcomer a will then store both packets in its storage. The same repair process can then be repeated and applied to any arbitrary next newcomer.

Since node b stores $\{X_2, X_4\}$, the three candidates are X_2, X_4, and $[X_2 + X_4]$. Since node c stores $\{[X_1 + X_2], [X_3 + X_4]\}$, the three candidates are $[X_1 + X_2]$, $[X_3 + X_4]$, and $[X_1 + X_2 + X_3 + X_4]$. Recall that node d stores $\{X_1, X_2\}$ and node e stores $\{X_3, X_4\}$. Out of the $3^2 = 9$ combinations, only the combination $(Z_b^*, Z_c^*) = ([X_2 + X_4], [X_1 + X_2 + X_3 + X_4])$ can simultaneously satisfy [Cond.1] and [Cond.2]. $Z_b^* = [X_2 + X_4]$ will then be sent to node a from node b and $Z_c^* = [X_1 + X_2 + X_3 + X_4]$ will be sent to node a from node c. Newcomer a will then store both packets in its storage. The same repair process can then be repeated and applied to any arbitrary next newcomer.

We close our discussion by the following proposition.

Proposition 3: There exists no distributed storage code for $(n, k, d, r) = (5, 3, 2, 1)$ if (2) does not hold. Namely, the proposed scheme, which achieves (2), is absolutely optimal.

The proof is by a min-cut analysis that applies to any possible helper selection policy and even non-linear codes.

III. The Min-Cut-Based Analysis and the Proposed Modified Family Repair Scheme

For $(n, k, d, r) = (5, 3, 2, 1)$, we have proved that even the best SHS is no better than BHS, see Proposition 1, while DHS is strictly better, i.e., DHS$\succ$SHS=BHS. In this section, we study the following fundamental question: Can we characterize the order of the performance of BHS, SHS, and DHS in a similar way for other (n, k, d, r) combinations?

A. When can LRRCs do better than Blindly-Repaired RCs under node unavailability?

Due the nature of the distributed storage problem, we only consider (n, k, d, r) values that satisfy

$$2 \leq n;\ 1 \leq k \leq n-1;\ 1 \leq d;\ \text{and}\ d + r \leq n - 1. \tag{3}$$

We now recall that for general (n, k, d, r) the tradeoff curve of any LRRC (and RC) with BHS is characterized in [3]:

$$\sum_{i=0}^{k-1} \min((d-i)^+\beta, \alpha) \geq \mathcal{M}, \tag{4}$$

where $(x)^+ = \max(x, 0)$. We then have the following new propositions.

Proposition 4: If $k \leq \left\lceil \frac{n-r}{n-d-r} \right\rceil$, then BHS is absolutely optimal. Namely, even the most intelligent helper selection will have the same tradeoff curve (4) as BHS.

Proposition 5: If $\min(d+1, k) > \left\lceil \frac{n}{n-d-r} \right\rceil$, then there exists an SHS scheme A such that the tradeoff curve of A is strictly better[3] than that of (4).

One can verify that some (n, k, d, r) values satisfy neither Propositions 4 nor 5. Namely, for those (n, k, d, r), we do not know whether a carefully designed SHS can be strictly better than BHS. There is a gap between Propositions 4 and 5

B. The Case $r = 1$

For the most interesting case of $r = 1$, we can close the gap between Propositions 4 and 5 when $d \in \{1, 2\}$.

Proposition 6: Consider any fixed (n, k, d, r) vector with $d = 1$ and $r = 1$. If the vector satisfies neither Propositions 4 nor 5, then one of the following four cases must be true: (i) $k = 3$; (ii) $k = 4$ and $(n \bmod 3) \neq 0$; (iii) $k = 4$ and $(n \bmod 3) = 0$; and (iv) $k \geq 5$. Furthermore, in cases (i) and (ii) BHS is absolutely optimal. In cases (iii) and (iv), there exists an SHS that is strictly better than BHS.

Proposition 7: Consider any fixed (n, k, d, r) vector with $d = 2$ and $r = 1$. If the vector satisfies neither Propositions 4 nor 5, then the following condition must hold: (i) $(n, k) = (5, 4)$ or $(5, 3)$. Furthermore, in (i), no SHS can do better than BHS but a carefully designed DHS is strictly better than BHS.

Some simple calculation can show that the gap for $d = 3$ and $r = 1$ contains only the point $(n, k, d, r) = (7, 3, 3, 1)$; and for $d = 4$ and $r = 1$ it contains only the two points $(9, 3, 4, 1)$ and $(7, 4, 4, 1)$. In general, the gap between Propositions 4 and 5 is quite small for $d \geq 3$ and $r = 1$.

Remark: [1] proves that for $r = 0$ and arbitrary d, if the optimal SHS is no better than BHS, then no scheme can ever do better than BHS. Namely, the surprising phenomenon

[3]Namely, when we plot the tradeoff curves as in Fig. 1, some portion of the curve of A is strictly below that of BHS while no portion of the curve of A is strictly above that of BHS.

of DHS≻SHS=BHS in Section II cannot be observed when $r = 0$. Propositions 6 and 7 show that when $r = 1$ and $d \leq 2$ there are exactly 2 scenarios exhibiting "DHS≻SHS=BHS," one being the example in Section II and the other being $(n, k, d, r) = (5, 4, 2, 1)$. Since "DHS≻SHS=BHS" does not happen when $r = 0$ and happens rarely when $r = 1$ and $d \leq 2$, the example in Section II was obtained by carefully searching through a wide range of (n, k, d, r).

C. The Modified Family Repair (MFR) Schemes

When proving that BHS is absolutely optimal in Proposition 4 and cases (i) and (ii) of Proposition 6, we use a min-cut analysis similar to [1]. When proving that a carefully designed SHS is strictly better than BHS in Proposition 5 and cases (iii) and (iv) of Proposition 6, we explicitly construct an SHS scheme, find its tradeoff curve, and prove that it is strictly better than BHS. Herein, we omit the min-cut analysis of the former and outline the SHS construction for the latter.

The main ingredient of our SHS construction is a new scheme, called the *modified family repair (MFR)* scheme. The MFR scheme divides the nodes into *complete families*, each having $(n-d-r)$ nodes. If $\frac{n}{n-d-r}$ is not integer, we have one incomplete family with size $n \bmod (n-d-r)$ nodes. If node i is in a complete family, then we choose D_i to contain all the nodes *not* in the family of node i. If i is in the incomplete family, then we choose D_i to contain node 1 to node $(d+r)$.

For example, suppose that $(n, d, r) = (8, 4, 1)$. Each complete family has $n - d - r = 3$ nodes. There are 2 complete families, $\{1, 2, 3\}$ and $\{4, 5, 6\}$, and 1 incomplete family, $\{7, 8\}$. Since node 4 is in complete family $\{4, 5, 6\}$, we have $D_4 = \{1, 2, 3, 7, 8\}$ being the nodes outside its family. Since node 7 is in the incomplete family $\{7, 8\}$, we have $D_7 = \{1, 2, 3, 4, 5\}$. All the other D_i can be found similarly.

If node 2 is unavailable when repairing node 4, then newcomer 4 will access $D_4 \backslash 2 = \{1, 3, 7, 8\}$ for repair. Similarly, if node 3 is unavailable when repairing node 7, then newcomer 7 will access $D_7 \backslash 3 = \{1, 2, 4, 5\}$ as helpers. Also see our discussion in footnote 2. Note that in MFR, each newcomer only requests help from *outside* its own family. The intuition is that we would like *each family to preserve as diverse information as possible during repair.*

Finally, we provide a formula that characterizes the storage-BW tradeoff of the MFR scheme. Unfortunately, due to the space limit, we directly use some notations in [2] with no detailed definitions. The following tradeoff curve is also the backbone of proving Propositions 5 and 6.

Proposition 8: Consider any (n, k, d, r) and the corresponding MFR scheme, the storage-BW tradeoff is

$$\min_{\forall \pi_f} \sum_{i=1}^{k} \min\left((d - y_i(\pi_f))^+ \beta, \alpha \right) \geq \mathcal{M}, \tag{5}$$

where the detailed definitions of π_f and $y_i(\pi_f)$ are the same as the ones used in [2, Proposition 3].

It is worth mentioning that Proposition 8 is weaker than the result in Section II-B in the following sense. The storage-BW tradeoff curve in Proposition 8 is based on min-cut analysis as in [3], which means that we do not know whether for general (n, k, d, r) there exists a finite field code that can meet the value in (5). Also see the discussion in [13]. In contrast, the code existence result in Section II-B is in the strongest sense since we first provide an explicit binary code construction and then prove its optimality in Proposition 3. In [2], explicit LR-RCs are provided for all $(n, k, d, r = 0)$, termed *generalized fractional repetition codes*, based on fractional repetition codes [4], [8]. Fractional repetition codes can indeed be thought of as LRRCs with $\alpha = d\beta$. For some $(n, k, d, r > 0)$ combinations, we have strengthened Proposition 8 by providing an explicit code construction but we omit the details.

IV. Conclusion

For the first time in the literature, we have shown that stationary helper selection (SHS) is suboptimal by carefully constructing an optimal binary code for $(n, k, d, r) = (5, 3, 2, 1)$ based on *dynamic helper selection* (DHS), where r represents how many nodes can be temporarily unavailable. For general (n, k, d, r) values, we have answered the question whether SHS/DHS can outperform blind helper selection (BHS) or not, for a vast majority of (n, k, d, r) values. The results thus provide valuable guidelines for each (n, k, d, r) whether it is beneficial to spend time and design new SHS/DHS schemes or whether one should simply use the basic BHS.

References

[1] I. Ahmad and C.-C. Wang, "When and by how much can helper node selection improve regenerating codes?" in *Proc. 52nd Annu. Allerton Conf. Communication, Control, and Computing.*

[2] ——, "When and by how much can helper node selection improve regenerating codes?" *[Online]. Available: arXiv:1401.4509 [cs.IT].*

[3] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, 2010.

[4] S. El Rouayheb and k. Ramchandran, "Fractional repetition codes for repair in distributed storage systems," in *Proc. 48th Annual Allerton Conf. on Comm., Contr., and Computing.*

[5] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, "On the locality of codeword symbols," *IEEE Trans. Inf. Theory*, vol. 58, no. 11, pp. 6925–6934, 2012.

[6] H. D. L. Hollmann, "On the minimum storage overhead of distributed storage codes with a given repair locality," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, Honolulu, HI, Jun. 2014, pp. 1041–1045.

[7] G. M. Kamath, N. Prakash, V. Lalitha, and P. V. Kumar, "Codes with local regeneration," in *IEEE Information Theory and Applications Workshop (ITA)*, San Diego, CA, Feb. 2013, pp. 1–5.

[8] O. Olmez and A. Ramamoorthy, "Replication based storage systems with local repair," in *International Symposium on Network Coding (NetCod)*, June 2013, pp. 1–6.

[9] L. Pamies-Juarez, H. D. L. Hollmann, and F. Oggier, "Locally repairable codes with multiple repair alternatives," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, Istanbul, Turkey, Jul. 2013, pp. 892–896.

[10] D. S. Papailiopoulos and A. G. Dimakis, "Locally repairable codes," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, Cambridge, MA, Jul. 2012, pp. 2771–2775.

[11] N. Prakash, G. M. Kamath, V. Lalitha, and P. V. Kumar, "Optimal linear codes with a local-error-correction property," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, Cambridge, MA, Jul. 2012, pp. 2776–2780.

[12] A. S. Rawat, O. O. Koyluoglu, N. Silberstein, and S. Vishwanath, "Optimal locally repairable and secure codes for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 60, no. 1, pp. 212–236, 2012.

[13] Y. Wu, "Existence and construction of capacity-achieving network codes for distributed storage," *IEEE J. Select. Areas Commun.*, vol. 28, no. 2, pp. 277–288, 2010.