## A ADDITIONAL EXPERIMENTAL RESULTS

We present additional results for our sampling and mixture studies. As observed in the main body of the paper, the positive trends we observe for 3 bpc weaken for 4 bpc allocations.

| Dataset | Ember1 Bit Error Rate | | | | Ember2 Bit Error Rate | | | |
|---|---|---|---|---|---|---|---|---|
| | LA | FLA | LA+AC | FLA+AC | LA | FLA | LA+AC | FLA+AC |
| 25% | 11.0 ± 1.4% | 11.0 ± 1.4% | 11.0 ± 1.4% | 11.0 ± 1.3% | 10.0 ± 1.2% | 10.0 ± 1.2% | 9.8 ± 1.1% | 11.0 ± 1.1% |
| 50% | 10.0 ± 0.88% | 10.0 ± 0.88% | 9.3 ± 0.57% | 9.1 ± 0.48% | 11.0 ± 0.77% | 11.0 ± 0.77% | 8.8 ± 0.86% | 9.3 ± 0.83% |
| 75% | 9.4 ± 0.65% | 9.4 ± 0.65% | 8.6 ± 0.4% | 8.4 ± 0.37% | 9.5 ± 0.53% | 9.5 ± 0.53% | 8.0 ± 0.43% | 7.9 ± 0.35% |
| 90% | 9.1 ± 0.41% | 9.1 ± 0.41% | 8.1 ± 0.012% | 8.1 ± 0.012% | 9.8 ± 0.92% | 9.8 ± 0.92% | 7.8 ± 0.33% | 7.8 ± 0.41% |
| 100% | 9.1% | 9.1% | 8.1% | 8.1% | 9.0% | 9.0% | 7.7% | 7.6% |

**Table 7: Effect of dataset size on ECC overhead in level allocation (3 bpc).**

| Dataset | Ember1 Bit Error Rate | | | Ember2 Bit Error Rate | | |
|---|---|---|---|---|---|---|
| | LA | FLA | LA+AC | LA | FLA | LA+AC |
| 25% | 3.8 ± 0.17% | 4.1 ± 0.35% | 3.7 ± 0.1% | 4.0 ± 0.14% | 4.4 ± 0.21% | 3.9 ± 0.082% |
| 50% | 3.6 ± 0.22% | 4.1 ± 0.26% | 3.6 ± 0.17% | 3.9 ± 0.13% | 4.3 ± 0.22% | 3.7 ± 0.12% |
| 75% | 3.7 ± 0.19% | 3.9 ± 0.12% | 3.5 ± 0.11% | 3.9 ± 0.15% | 4.2 ± 0.16% | 3.8 ± 0.11% |
| 90% | 3.6 ± 0.1% | 4.0 ± 0.09% | 3.5 ± 0.11% | 3.8 ± 0.12% | 4.2 ± 0.2% | 3.7 ± 0.12% |
| 100% | 3.6% | 3.7% | 3.6% | 3.7% | 4.0% | 3.5% |

**Table 8: Effect of dataset size on bit error rate (BER) in level allocation (4 bpc).**

| Dataset | Ember1 Bit Error Rate | | | Ember2 Bit Error Rate | | |
|---|---|---|---|---|---|---|
| | LA | FLA | LA+AC | LA | FLA | LA+AC |
| 25% | 32.0 ± 1.1% | 32.0 ± 1.1% | 32.0 ± 0.68% | 34.0 ± 0.97% | 34.0 ± 0.97% | 33.0 ± 0.54% |
| 50% | 31.0 ± 1.4% | 31.0 ± 1.4% | 31.0 ± 1.1% | 33.0 ± 0.84% | 33.0 ± 0.84% | 32.0 ± 0.74% |
| 75% | 32.0 ± 1.2% | 32.0 ± 1.2% | 31.0 ± 0.73% | 33.0 ± 0.99% | 33.0 ± 0.99% | 32.0 ± 0.63% |
| 90% | 31.0 ± 0.67% | 31.0 ± 0.67% | 31.0 ± 0.76% | 33.0 ± 0.77% | 33.0 ± 0.77% | 32.0 ± 0.78% |
| 100% | 32% | 32% | 31% | 32% | 32% | 31% |

**Table 9: Effect of dataset size on ECC overhead in level allocation (4 bpc).**

| Dataset | Ember1 Bit Error Rate | | | | | | | | Ember2 Bit Error Rate | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LA | LA+FR | FLA | FLA+FR | LA+AC | LA+AC+FR | FLA+AC | FLA+AC+FR | LA | LA+FR | FLA | FLA+FR | LA+AC | LA+AC+FR | FLA+AC | FLA+AC+FR |
| 100/0 | 9.1% | 9.1% | .9.1% | 9.1% | 8.1% | 8.1% | 8.1% | 8.1% | 9.0% | 9.0% | 9.0% | 9.0% | 7.7% | 7.6% | 7.6% | 7.6% |
| 50/50 | 9.6% | 9.6% | 9.6% | 9.6% | 9.6% | 9.1% | 9.6% | 9.2% | 10.0% | 10.0% | 10.0% | 10.0% | 10.0% | 10.0% | 8.6% | 8.6% |
| 10/90 | 11.0% | 11.0% | 11.0% | 11.0% | 12.0% | 12.0% | 12.0% | 12.0% | 11.0% | 11.0% | 11.0% | 11.0% | 11.0% | 11.0% | 12.0% | 11.0% |
| 0/100 | 11.0% | 11.0% | 11.0% | 11.0% | 12.0% | 12.0% | 12.0% | 12.0% | 12.0% | 12.0% | 12.0% | 12.0% | 12.0% | 12.0% | 12.0% | 12.0% |

**Table 10: Interchip level allocation (3 bpc) comparison on ECC overhead. A mix of $x/y$ = $x$% target chip's dataset plus $y$% from another chip.**

| Dataset | Ember1 Bit Error Rate | | | | | | Ember2 Bit Error Rate | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LA | LA+FR | FLA | FLA+FR | LA+AC | LA+AC+FR | LA | LA+FR | FLA | FLA+FR | LA+AC | LA+AC+FR |
| 100/0 | 3.6% | 3.5% | 3.7% | 3.7% | 3.6% | 3.5% | 3.7% | 3.7% | 4.0% | 3.9% | 3.5% | 3.5% |
| 50/50 | 3.6% | 3.6% | 4.1% | 4.2% | 3.7% | 3.5% | 4.3% | 4.3% | 5.5% | 5.1% | 3.9% | 3.9% |
| 10/90 | 4.1% | 4.0% | 4.4% | 4.3% | 4.1% | 4.0% | 4.8% | 4.6% | 5.3% | 5.3% | 4.3% | 3.9% |
| 0/100 | 4.0% | 4.1% | 4.2% | 4.9% | 4.1% | 4.1% | 5.1% | 5.0% | 5.5% | 5.3% | 4.7% | 4.4% |

**Table 11: Interchip level allocation (4 bpc) comparison on bit error rate (BER). A mix of $x/y = x\%$ target chip's dataset plus $y\%$ from another chip.**

| Dataset | Ember1 Bit Error Rate | | | | | | Ember2 Bit Error Rate | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LA | LA+FR | FLA | FLA+FR | LA+AC | LA+AC+FR | LA | LA+FR | FLA | FLA+FR | LA+AC | LA+AC+FR |
| 100/0 | 32.0% | 32.0% | 32.0% | 32.0% | 31.0% | 30.0% | 32.0% | 32.0% | 32.0% | 32.0% | 31.0% | 31.0% |
| 50/50 | 31.0% | 31.0% | 31.0% | 31.0% | 32.0% | 31.0% | 36.0% | 36.0% | 36.0% | 36.0% | 33.0% | 33.0% |
| 10/90 | 34.0% | 34.0% | 34.0% | 34.0% | 35.0% | 34.0% | 39.0% | 39.0% | 39.0% | 39.0% | 36.0% | 33.0% |
| 0/100 | 34.0% | 34.0% | 34.0% | 34.0% | 34.0% | 35.0% | 41.0% | 41.0% | 41.0% | 41.0% | 38.0% | 36.0% |

**Table 12: Interchip level allocation (4 bpc) comparison on ECC overhead. A mix of $x/y = x\%$ target chip's dataset plus $y\%$ from another chip.**

## B ADDITIONAL PSEUDOCODE

Additionally, we present pseudocode for both of our heuristic improvements, i.e., our `FindAllCliques` and `FlexibleRefine` subroutines.

### B.1 `FindAllCliques`

**Algorithm 2** `FindAllCliques` pseudocode.

```
function FindAllCliques(G, n, ϵ)
    γ = LevelAlloc(n, ϵ)
    Cands = CandidateGen(γ)
    Cliques = FindMaximalCliques(G)
    BestBER = 1, BestClique = None
    for c ∈ Cliques do
        if Size(c) == n then
            Levels = Sorted(List(c))
            BER = GetBERFromAlloc(Levels)
            if BER < BestBER then
                BestBER = BER, BestClique = Levels
    return BestClique

function ConstructGraph(Cands)
    G = InitializeGraph()
    for node ∈ Cands do
        G.AddNode(node)
    for CandPair ∈ Combinations(Cands, 2) do
        if Not IsOverlap(CandPair) then
            G.AddEdge(CandPair)
    return G
```

### B.2 `FlexibleRefine`

**Algorithm 3** `FlexibleRefine` (`FR`) pseudocode.

```
function FlexibleRefine(L, n)
    for i ∈ [0, ..., Len(L)] do
        MinBer = GetBER(L, n)
        MinBoundary = L[i][xh]
        for j ∈ [L[i][xh] + 1, ..., L[i + 1][xl]] do
            L[i][xh] = L[i + 1][xl] = j
            BER = GetBER(L)
            if BER < MinBER then
                MinBoundary = j
                MinBER = BER
        L[i][xh] = L[i + 1][xl] = MinBoundary
    return L
```