Weighted Vertex Cover Using Disjoint Set Data Structures for the Memory Reconfiguration Problem

***Abstract*—We propose a minimum weighted vertex cover method relying on modified disjoint data set structures and Kruskal algorithm for purposes of the memory reconfiguration problem. With column-muxing, the problem of redundancy allocation and planning with spare columns and spare rows must account for the column versus row redundancy block costs. The proposed algorithm is comprehensive and computationally efficient. It enables reducing the budget of repair by upto 70% compared to an all row repair solution. It also highlights the difference in redundancy planning requirements based on the underlying column muxing strategy for fixed array size.**

I. Introduction

MEmory redundancy planning is key for design manufacturability and yield. A large or small quota can result in extra spare parts or yield loss. Planning around the expected fail probability results in proper usage of resources. The authors in [1], [2] proposed heuristics that determine the number of rows and columns for proper yield. Targetting fail rate within the repairable range, enables solving the NP complete problem with average case polynomial time. The authors, however, did not incorporate the cost for column versus row redundancy. We propose a weighted vertex cover approach that takes into consideration the unbalanced cost between spare rows and columns for proper planning.

II. Proposed Approach

Figure 1 presents the Disjoint set data structure [3]. It relies on merge and find functions to detect cycles for minimum spanning tree algorithms such as Kruskal's algorithm. Figure 2 presents a sketch of a 4x8 array with two faults. It assumes that the column muxing is 4-to-1. This reduces to a 4x2 array: two column blocks (cb1, and cb2) and 4 rows. A row comprises eight bit cells, and a column block 16 bit cells. It is represented by a graph of 4x2 vertices and two edges. Vertex cover (r1, r2) is the least costly cover. To find the cover, the array is mapped to a bipartite graph. Given a bipartite graph G(V, E), Kruskal's algorithm [4], is an efficient algorithm for finding the minimum spanning forests, that runs in $O(|E|log|E|)$, where $|E|$ is the cardinality of E; E is the set of edges and V is the set of vertices. A repairable nxn arrays has a fault probability $p_f = c/n$ where $c < 1$, and the number of non-tree vertices is log(n) [1]. Figure 3 presents the proposed flow. We rely on Kruskal's algorithm to find the forest; the edges are equally weighted. We find for each tree a minimum weighted vertex cover (Fig. 4) incorporating the verteices' cost. We handle the non-tree vertices heuristically. We study mxn arrays, where $m \in \{64, ..., 512\}$, $n \in \{128, .., 1024\}$; $p_f = 0.5/\sqrt{(mxn)}$ and column muxing scenarios $\{1{-}to{-}1, 2{-}to{-}1, 4{-}to{-}1\}$. We analyze the benefit of adding column redundancy versus an all row-redundancy solution to help with redundancy planning. We note that the advantage (Fig. 5acd entries < 1) varies with the column block cost vs row cost and muxing strategy.

References

[1] W. Shi,and W. Fuchs. "Probabilistic analysis and algorithms for reconfiguration of memory arrays." IEEE TCAD (1992)
[2] — removed for blind review
[3] K. Taha, IEEE Transactions on Knowledge and Data Engineering, 2019
[4] Ayegba, Peace, et al. "A comparative study of minimal spanning tree algorithms." IEEE ICMCECS, 2020.

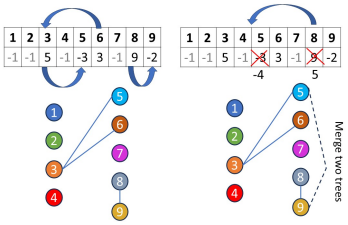

Fig. 1. Disjoint set data structure helps detect cycles. Merging trees example. A root is labeled by its size (times '-1'). A node is labeled by its parent.

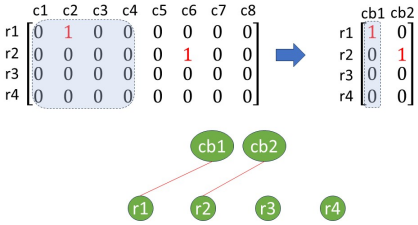

Fig. 2. Example array faults. An edge represents the fault.

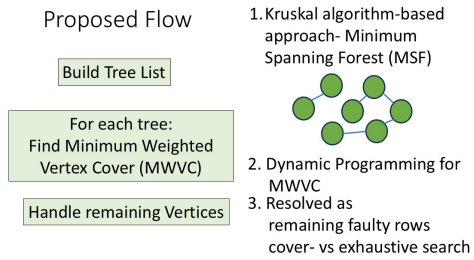

Fig. 3. Overall proposed flow.

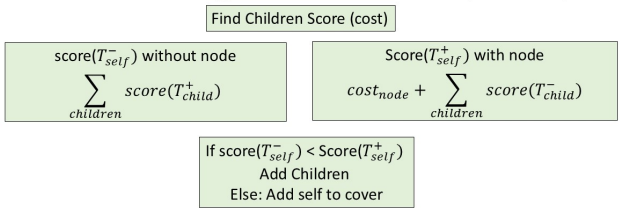

Fig. 4. Minimum weighted vertex cover (MWVC): dynamic programming for self node (T_{self}) and children (T_{child}).

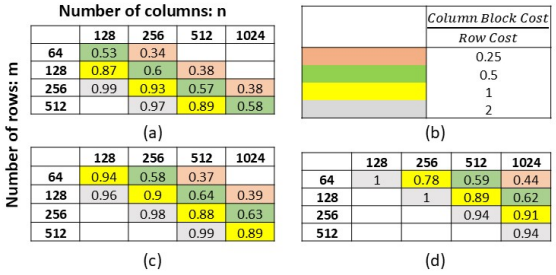

Fig. 5. Normalized redundancy budget cost based on proposed MWVC (smaller better) for added column redundancy vs row only redundancy planning. a) 1-to-1, c) 2-to-1, and d) 4-to-1 column muxing. b) cells per column block to row block ratio. Budget planning function of array size, and muxing; e.g., fixed array size 256x1024 adding column redundancy incorporates 30% (53%) cost reduction when muxing is 2-to-1 (1-to-1) vs 4-to-1.