This is simply achieved in the proposed heuristic algorithm by sorting the ensemble partitions in descending order of their entropies, $H(C^i)$, select the first partition as the initial reference and then aggregate the remaining partitions in the sorted order.

An important feature that is achieved as a by-product of the proposed adaptive algorithm is that the aggregated partition becomes invariant of the order of the input partitions and of the initial partition, unlike the `Vote` algorithm. This invariability is a generally desirable property for an aggregation algorithm and it also saves the extra computations required to enhance the `cVote` algorithm by performing multiple passes. The steps of the `Ada-cVote` algorithm are outlined in Algorithm 3.

Algorithm 3 `Ada-cVote`

Function $\bar{\mathbf{U}} = \texttt{I-cVote}(\mathcal{U})$

1: Re-order $\mathcal{U}$, s.t. $\mathbf{U}^i$ are sorted in decreasing order of $I(C^i; X)$ ($\equiv H(C^i)$ for hard partitions)
2: Assign $\mathbf{U}^1$ to $\mathbf{U}^0$.
3: **for** $i = 2$ to b **do**
4: Compute $\mathbf{W}^i$ as given by Eq. 3.5.
5: $\mathbf{V}^i = \mathbf{U}^i \mathbf{W}^i$
6: $\mathbf{U}^0 = \frac{i-1}{i}\mathbf{U}^0 + \frac{1}{i}\mathbf{V}^i$
7: **end for**
8: $\bar{\mathbf{U}} = \mathbf{U}^0$.

4.2.3 Simulation Results

In this section, `cVote` and `Ada-cVote` are compared for the partition generation models described in Ch. 3. The two algorithms are compared by evaluating the obtained $I(C; X)$ and $\text{MSE}(\bar{\mathbf{U}}; \{\mathbf{U}_i\}_{i=1}^{b})$ for the aggregated partitions. Furthermore, the error rates Err^* are compared in the case of uniform ensembles to investigate if the adaptive aggregation may reduce the error rate for `cVote`, which tends to perform poorly for this type of ensemble, especially as p_e^i increases (as observed in Ch. 3). Finally, the adaptive aggregation is applied to the bipartite matching