An Approximation Analysis of Label Mapping Algorithms for Combining Speaker Diarization Systems

Desh Raj

August 8, 2021

Abstract

Speaker diarization (or "who spoke when") is the task of segmenting speech into homogeneous speaker-specific regions. Recently, there have been several advances in performing overlap-aware diarization, i.e., recognizing multiple speakers in overlapping speech. These include traditional clustering-based systems, as well as novel end-to-end approaches trained with supervision. Since machine learning tasks often benefit from an ensemble of systems, it may be desirable to develop a method that combines these complementary approaches for diarization. In this paper, we build upon a recently proposed paradigm for performing such combination, using a two-stage "label mapping" and "label voting" technique. In particular, we analyze the label mapping stage in the framework of a maximum orthogonal graph partitioning problem. We first show that the original algorithm used for this task is exponential in the input size, which makes it intractable. We then propose two new algorithms, based on a greedy and a randomized local search strategy, respectively, and derive approximation bounds for them. We empirically demonstrate the effectiveness of our methods on the AMI meeting corpus, where our system combination improves the diarization error rate by about 4% relative over the single best system. Our code is publicly available: `https://github.com/desh2608/dover-lap`.

1 Introduction

1.1 A background in speaker diarization

Speech is one of the most popular modalities for communication and dissemination of ideas. In everyday life, we come across a wide variety of scenarios where 2 or more people interact using the spoken word. These interactions may occur in the form of telephone conversations, multi-speaker meetings, or simply a conversation over a dinner party. In the digital age, video/audio conferencing has also become a popular mode of communication. Given its widespread use, a natural question is whether we can identify the segments of speech spoken by different speakers. Such segmentation is often useful for applications such as meeting transcription, and also enables downstream psycho-linguistic analysis such as the study of language acquisition in infants, pyschotherapy and human interaction, and observing collaborative learning in peer groups.

Speaker diarization (or "who spoke when") is defined as the task of segmenting speech into homogeneous speaker-specific regions [MBE$^+$12, TR06]. The input in this task is an audio recording (single or multi-channel) which may extend from a few minutes up to several hours (or days, such as in scenarios like child language acquisition). The desired output is a set of segments with associated speaker labels. Formally, given an audio recording R, the diarization system is a function f such that

$$f(R) = (U, K), \tag{1}$$

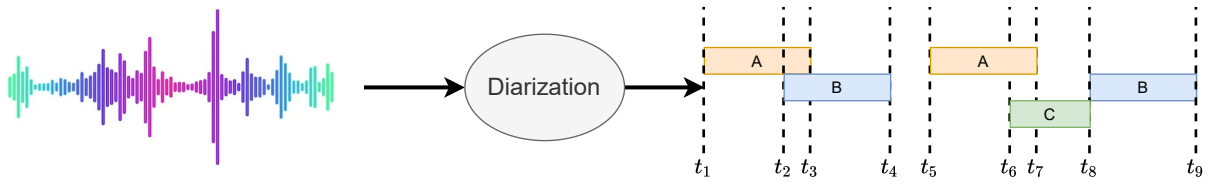

Figure 1: An illustration of the speaker diarization task. The system estimates $N = 3$ speakers in the recording.

where

$$U = \{(\Delta_j, u_j)\} \tag{2}$$

is a set of speaker-labeled time segments with $\Delta_j = (t_j, t_{j+1})$ and $u_j \in [N]$, and N is the estimated number of speakers in R. Depending on the use-case, the true value of N may or may not be known beforehand. The speaker labels u_j are *relative* labels, i.e., they are only consistent within the recording. Fig. 1 illustrates an example of the diarization task where the system predicts 3 speakers: A, B, and C, respectively. Note that some parts of the audio may be "non-speech," i.e., they do not have any assigned speaker (for e.g., the $t_4 - t_5$ segment), whereas some other parts may have "overlapping speech" and are assigned to multiple speakers (for e.g., the $t_2 - t_3$ and $t_6 - t_7$ segments).

1.2 Methods for speaker diarization

In the 2000s, NIST and DARPA organized several challenges to advance the state of speaker diarization. Since statistical learning-based approaches for diarization were still fairly new at the time, the evaluations were suitably constrained — participants were often provided oracle speech segmentation (i.e., non-speech regions were demarcated), the number of speakers was known beforehand, and the audio recordings were assumed to have zero overlapping speech. More recently, as the diarization task has matured, we have moved towards unconstrained evaluations. The CHiME-6 [WMBV20] and DIHARD [RCC$^+$19, RSK$^+$20] challenges, for example, contain challenging recordings with high noise and overlapping speech and a diverse number of speakers.

This shift in evaluation happened as a result of several advances in systems that perform speaker diarization. The traditional approach for speaker diarization involved a clustering of segment-level speaker embeddings, optionally followed by resegmentation [GRSS$^+$17, PHKN20, LPDB20]. This approach requires separately optimized speech activity detection components, and also assumes that the recording did not contain overlapping speech. To alleviate the latter problem, several methods have been proposed which seek to employ overlap detection modules and use some heuristics of the clustering process to assign extraneous speakers to the overlapping segments [RHK21, BBGP20]. More recently, supervised diarization methods such as region proposal networks (RPN), end-to-end neural diarization (EEND), and target-speaker voice activity detection (TS-VAD) have been proposed which inherently perform overlapping speaker assignment [HWF$^+$20, FWH$^+$20, MKP$^+$20]. An alternate paradigm for overlap-aware diarization involves using a continuous speech separation (CSS) module to first split the recording into 2 or 3 streams, followed by clustering of speakers across the streams [RDC$^+$21, XKC$^+$20]. Fig. 2 provides an overview of some of these diarization systems.

1.3 Ensembles of diarization systems

Machine learning tasks often benefit from an ensemble of systems with complementary strengths. For examples, ROVER (Recognition Output Voting Error Reduction) is a popular combination method for automatic speech recognition [Fis97]. With the availability of different diarization methods, there has recently been

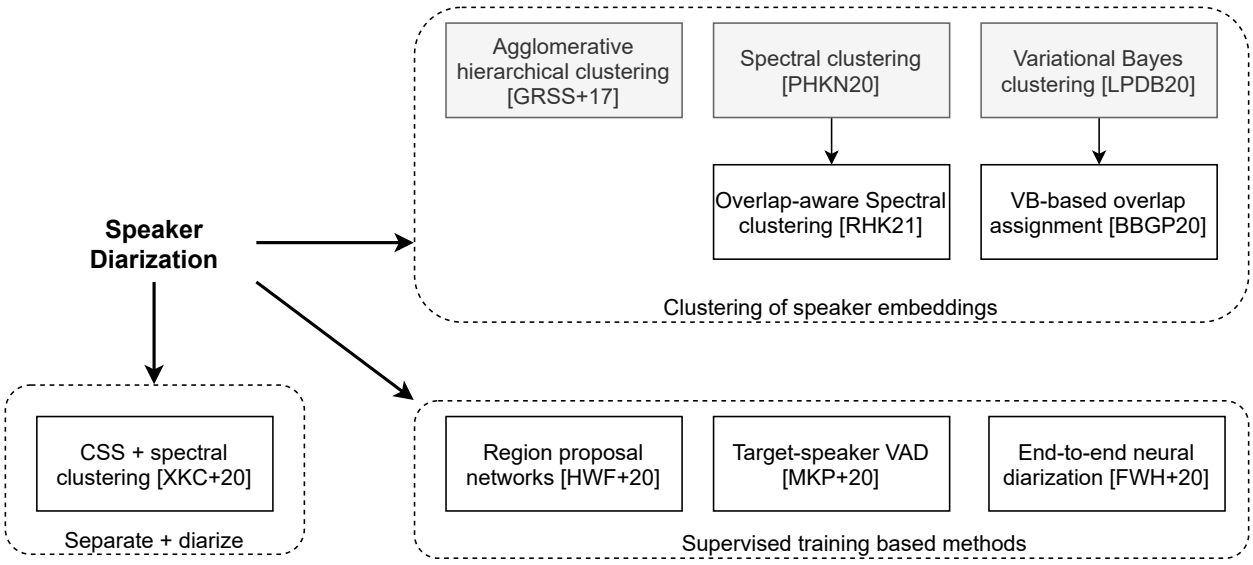

Figure 2: An overview of some of the diarization methods proposed in literature. The methods in grey boxes cannot handle overlapping speech.

some interest in developing a system combination technique for speaker diarization. However, this is inherently a hard problem for several reasons:

1. Different diarization outputs (or hypotheses) may have different number of speaker estimates for the same recording.

2. The diarization hypotheses are in different label spaces, and speaker identities are only consistent within a hypotheses. For example, systems A and B may assign labels (A1, A2), and (B1, B2, B3), respectively, for some recording, and there is no correspondence between these 2 sets of speaker labels.

3. For overlap-aware diarization systems, the hypotheses may disagree on which speech segments contain overlapping speakers, and so any combination system needs to have a mechanism to arrive at a consensus.

Next, we will formalize this task of combining diarization system outputs. We assume that we are given the hypotheses $U_1, \ldots, U_K$ from K diarization systems, where each system can perform overlapping speaker assignment and may contain different number of speakers, $c_1, \ldots, c_K$, respectively. Recall, from equation (2), that U_k is a set of tuples containing time intervals and the corresponding speaker assignment.

We define the problem of combining the hypothesis as finding a joint diarization hypothesis $\hat{U} = g(U_1, \ldots, U_K)$ such that $\hat{U}$ minimizes the chosen diarization error metric with respect to an unknown reference. A straightforward approach for solving this problem involves dividing the input hypotheses into small time durations, and performing majority voting individually with each such region. However, there are two issues with this solution. First, to perform any kind of voting (within a region), the system outputs need to be in the same label space. Second, overlap-aware systems may contain different number of speakers within any such region, and for overlap-aware combination, the voting mechanism needs to account for this possibility.

DOVER (Diarization output voting error reduction) [SY19] was the first method introduced for combining diarization systems. It proposed the two-stage framework of combination, involving label mapping and label voting, which are defined below.

1. **Label mapping:** Since diarization outputs do not have "absolute" speaker identities, the speaker labels of different outputs have no strict correspondence. In order to be able to make any "voting mechanism"

possible, the outputs have to be mapped to a common label space — we informally refer to this as the label mapping problem. We will formalize this task, in terms of the terminology we have developed, in § 3.1.

2. **Label voting:** Once all the hypotheses have been mapped to a common label space, voting is performed for time regions demarcated by timestamps within which every hypotheses has consistent speaker assignments (i.e., no speaker changes happen inside the region). For overlap-aware hypotheses, we additionally need to devise a strategy for estimating the number of speakers in a region in the output combination.

In this paper, we will focus our attention on the label mapping stage. In particular, we will formulate the label mapping task in the framework of a maximum orthogonal graph partitioning problem, and then propose and analyze several algorithms in this framework. We will design algorithms based on a greedy Hungarian strategy and a randomized local search technique, and derive approximation ratios for both algorithms. Finally, through experiments on the AMI meeting dataset, we will demonstrate the empirical strengths of our proposed algorithms.

2 Related work

The problem of label mapping that we analyze in this paper is analogous to the popular task of *index domain alignment* in distributed array processing [LC90]. It has also been studied to some degree in logistics [EJZ00] and manufacturing systems [ZZ02]. A related task is to enumerate maximal k-cliques in k-partite graphs, for which several branch-and-bound methods have been proposed [MK13]. Approximation algorithms for such problems have been used extensively in computational biology, such as for the alignment for metabolic pathways [CHS17] and for the global alignment of multiple protein interactions [SXB08].

In this paper, we borrow algorithmic and analytical ideas from two key papers. In [HLZ00], the authors presented a graph formalism demonstrating the equivalence between the maximum (minimum) orthogonal partition problem and the minimum (maximum) disjoint k-clique problem, and proposed greedy incremental algorithms which were based on maximum matching in a complete bipartite graph. We use this algorithm as our first method, and refer to it as the greedy Hungarian algorithm, since bipartite matching is the same as linear sum assignment. In [LPZ06], a greedy local search algorithm was proposed with approximation ratio similar to the method from [HLZ00], and then extended to a randomized version which achieved a close to optimal solution under some assumptions.

3 Label mapping as a graph partitioning problem

3.1 The label mapping task

Recall that diarization system combination involves a set of K input hypotheses $U_1, \ldots, U_K$, which needs to be mapped to a single diarization output $\hat{U}$, such that this combination optimizes the error on some unknown metric. To achieve this, the hypotheses must first be mapped to a common label space. Formally, the objective of "label mapping" is to find a set of N speaker labels $\{\hat{u}^1, \ldots, \hat{u}^N\}$ for our combined output $\hat{U}$, and a mapping function $\mathcal{S}$ that takes as input one of the speaker labels from the given hypotheses and maps it to a label in the combined output, i.e., $\mathcal{S}(u_k^{c_k}) = \hat{u}^n$.

For example, suppose we have two diarization hypotheses U_1 and U_2, and they assign speaker labels (u_1^1, u_1^2) and (u_2^1, u_2^2, u_2^3), respectively, for a recording R. Then, a label mapping function $\mathcal{S}$ may produce the mapping shown in Table 1.

Table 1: An example of label mapping for 2 hypotheses.

Hypothesis	Original label	Mapped label
U_1	u_1^1	A
	u_1^2	B
U_2	u_2^1	B
	u_2^2	C
	u_2^3	A

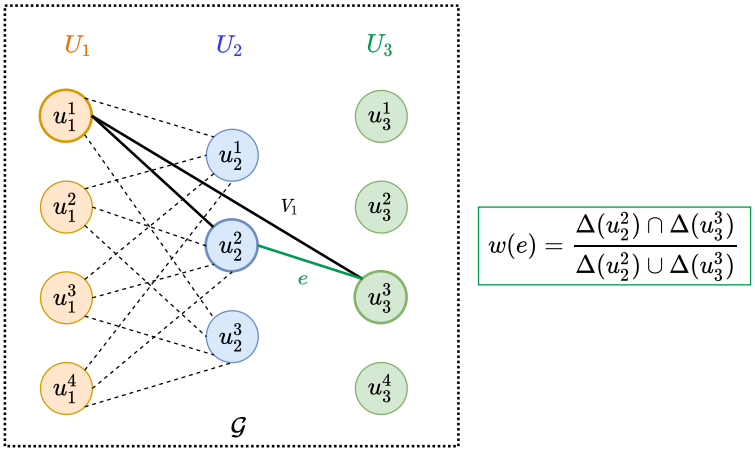

Figure 3: Illustration of the label mapping problem as a graph $\mathcal{G}$ for the case of $K = 3$. V_1 denotes the clique formed by vertices (u_1^1, u_2^2, u_3^3). $\Delta(u)$ represents the segments where speaker u is active in the recording.

Here, the speaker labels u_1^1 and u_2^3 are mapped to the same final speaker C (and similarly for u_1^2 and u_2^1). After this mapping, the hypotheses U_1 and U_2 are in the same label space (A, B, C), and so a majority voting can be performed. In the next section, we will present a reformulation of this task as a graph partitioning problem.

3.2 Problem formulation

Again, consider K diarization hypotheses $U_1, \ldots, U_K$, containing $c_1, \ldots, c_k$ speakers, respectively, such that $C = \max\{c_k, k \in [K]\}$. Let us denote each speaker as a node, i.e., u_k^i is the node corresponding to the i^{th} speaker in the k^{th} hypothesis, and $V = \{u_k^i\}$ is the set of all speaker nodes. Let $E = \{(u_k^i, u_\kappa^j) : \forall k, \kappa \in [K], i \in U_k, j \in U_\kappa, k \neq \kappa\}$ denote the set of all edges. Informally, this means that there is an edge between any two nodes if the nodes belong to different hypotheses. Additionally, we have a weight function $w : e \to \mathbb{R}^+$, where e denotes an edge. In practice, these edge weights are obtained by computing the relative overlap duration between the speakers in the recordings, i.e.,

$$w(u_k^i, u_\kappa^j) = \frac{\Delta(u_k^i) \cap \Delta(u_\kappa^j)}{\Delta(u_k^i) \cup \Delta(u_\kappa^j)}, \tag{3}$$

where $\Delta(u)$ is the set of all segments where speaker u is active in the recording. Clearly, $w : E \to [0, 1]$, and a higher w means that the corresponding speakers are more likely to occur in the same segments in the recording (i.e., they are more likely to represent the same speaker).

We define the graph as $\mathcal{G} = (V, E, w)$. It is easy to see that $\mathcal{G}$ is K-partite, and if $c_k = C$, $\forall k \in [K]$, then it is also complete. Fig. 3 illustrates this graphical formulation of the label mapping problem.

Table 2: Notations used in the paper.

Symbol	Definition
$U_1, \ldots, U_K$	Diarization hypotheses
c_k	Number of speakers in hypotheses k
C	Maximum number of speakers in any hypothesis
u_k^i	Speaker i in hypothesis k
V	Set of all hypotheses speakers
E	Set of all edges $\{(u_k^i, u_\kappa^j)\}$
w_e	Weight on edge $e = \{(u_k^i, u_\kappa^j)\}$
$(V_1, \ldots, V_C)$	Clique set (output of label mapping)
$\mathcal{G}$	Graph formed by vertex set V and edge set E with weights w

Each U_k in the graph is an *independent set* (set of vertices with no edges between any pair), and the label mapping problem can be defined as: partition V into C vertex-disjoint cliques $\Phi = (V_1, \ldots, V_C)$, such that the partition maximizes

$$J(\Phi) = w(\Phi) = \sum_{c \in C} w(V_c) = \sum_{c \in C} \sum_{e \in E(V_c)} w(e), \tag{4}$$

where $E(V_c)$ represents edges in the sub-graph induced by V_c. Intuitively, the objective maximizes the sum of all edge weights within the cliques. The partition is *orthogonal* since it may contain at most 1 vertex from every U_k. Table 2 summarizes these notations.

It may not immediately be clear why maximizing the objective in (4) provides an optimal label mapping. Since the partition is orthogonal, each V_c may represent a mapped speaker label. By maximizing the total edge weights within cliques, we maximize the total relative overlap between speaker turns for speakers that are mapped to the same label.

3.3 Empirical validation for the objective

The objective we are trying to maximize in (4) maximizes the total relative overlap between speakers mapped to the same label. However, diarization systems are actually evaluated using the *diarization error rate* (DER) metric, which is computed as:

$$DER = \frac{\text{Missed speech}(MS) + \text{False alarm}(FA) + \text{Speaker confusion}(CF)}{\text{Total reference speaking time}(T)}. \tag{5}$$

If we divide the recording R into S regions (where no speaker changes occur within a region), the components of the DER between the reference and the hypotheses can be computed using the following equations. Suppose, for any region s, $N_{ref}(s)$ and $N_hyp(s)$ denote the number of speakers estimated by the reference and the hypotheses, respectively, and $N_{correct}(s)$ denotes the number of correctly identified speakers.

1. **Missed speech**: fraction of scored time that a hypothesized non-speech segment corresponds to a reference speaker segment. It can be expressed as

$$MS = \frac{1}{T} \sum_{s=1}^{S} \text{dur}(s)(\max(0, N_{ref}(s) - N_{hyp}(s)))$$

2. **False alarm**: fraction of scored time that a hypothesized speaker is labelled as a non-speech in the reference. It can be formulated as

$$FA = \frac{1}{T} \sum_{s=1}^{S} \text{dur}(s)(\max(0, N_{hyp}(s) - N_{ref}(s)))$$

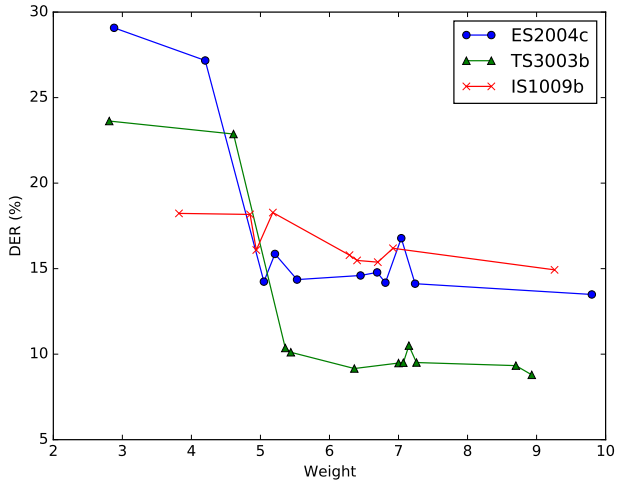

Figure 4: Partition weight $w(\Phi)$ versus diarization error rate (DER) for three arbitrarily chosen recordings from the AMI evaluation set, showing that DER tends to improve with weight.

3. **Speaker confusion**: fraction of scored time that a speaker ID is assigned to the wrong speaker. It is computed as

$$CF = \frac{1}{T}\sum_{s=1}^{S} \text{dur}(s)(\min(N_{ref}(s), N_{hyp}(s)) - N_{correct}(s))$$

Since the reference and hypotheses have speaker labels in different label spaces, they are first aligned using the Hungarian algorithm [Kuh55], which solves a linear sum assignment problem. [1] It is hard to demonstrate a theoretical correspondence between the objective in (4) and the DER metric. However, we empirically demonstrate a correspondence using some recordings from the AMI dataset in Fig. 4. We note from the figure that as the objective (x-axis) improves, the DER decreases, suggesting that our proposed objective is a good proxy for minimizing the final DER.

3.4 $\mathcal{G}$ is a Turán graph

In this section, we show that the graph $\mathcal{G}$ constructed in § 3.2 is a Turán graph, and in particular, it is $T(CK, K)$. First, we define a Turán graph below.

The **Turán graph** $T(n, r)$ is a complete multi-partite graph formed by partitioning a set of n vertices into r subsets, with sizes as equal as possible, and connecting two vertices by an edge if and only if they belong to different subsets.

Lemma 3.1. *Every graph $\mathcal{G}$ is equivalent to some $T(CK, K)$ Turán graph, i.e., it is equivalent to a complete K-partite graph $K_{C,C,C,\ldots}$.*

Proof. Given $\mathcal{G}$, if all independent sets U_k contain exactly C nodes, then the statement is trivially true. Otherwise, for any U_k containing $c_k < C$ nodes, we add $C - c_k$ dummy nodes to U_k, and connect each dummy node with vertices in all other independent sets. We assign weight 0 to all the newly added edges. Let us call this complete graph $\mathcal{G}'$. It is easy to see that $\mathcal{G}'$ is $T(CK, K)$ since it contains CK vertices divided equally into K subsets (each independent set is such a subset). Now, we only need to show that any solution Φ for the graph $\mathcal{G}$ is equivalent to some solution Φ' to $\mathcal{G}'$.

[1] Our first algorithm for label mapping is motivated by this application of the Hungarian algorithm, and is described in § 5.

Algorithm 1: DOVER-Lap label mapping

Input: Graph $\mathcal{G} = (V, E, w)$
Output: Partition $\Phi = V_1, \ldots, V_C$

1 $\Phi = \{\}$
/* Loop until no vertices remaining */
2 **while** $V \neq \phi$ **do**
/* Enumerate all maximal cliques */
3 S = set of all maximal cliques in V
/* Get maximum weighted clique */
4 V_c = max(S, key=$\sum_{e \in S_i} w(e)$)
/* Add clique to partition */
5 $\Phi = \Phi \cup \{V_c\}$
/* Remove clique vertices from V */
6 $V = V \setminus \{V_c\}$

Suppose Φ maximizes the objective in (4) for graph $\mathcal{G}$. Clearly, Φ must contain exactly C cliques, and each clique has size at most K. We can extend Φ to Φ' by incrementally adding dummy nodes to the cliques until they become maximal. Since all the added edges are zero-weighted, $w(\Phi') = w(\Phi)$.

Similarly, if we have a solution Φ' for $\mathcal{G}'$, we can obtain a solution Φ for $\mathcal{G}$ with the same total weight by simply removing the dummy nodes from all the cliques in Φ'. □

Lemma 3.2. *$\mathcal{G}$ has an exponential number of maximal cliques.*

Proof. The proof is through a simple combinatorial argument. Since $\mathcal{G}$ is a complete K-partite graph, any maximal clique of $\mathcal{G}$ contains exactly 1 vertex from all its K independent sets. Since each independent set has C vertices, there are C^K possibilities for a maximal clique. □

4 The DOVER-Lap label mapping algorithm

Before we propose new algorithms for label mapping, we analyze a previously proposed method in the graphical framework from § 3.2, and show that it is exponential in the input size. Algorithm 1 presents the label mapping algorithm proposed earlier as part of the DOVER-Lap algorithm [RGPH$^+$21]. Note that the algorithm has been reformulated in terms of the graph problem we described earlier. The algorithm roughly follows the following steps after the graph construction.

1. Enumerate all the maximal cliques in $\mathcal{G}$. Let this set be denoted by S.
2. Find the clique V_c with the maximum weight in S.
3. Add V_c to the partition Φ. Remove the vertices in V_c from $\mathcal{G}$ and the associated edges.
4. Repeat from Step 1 until no vertices remain in $\mathcal{G}$.

While this method leads to strong diarization performance by the ensemble, it has a glaring problem — it is exponential in the number of input hypotheses, since there are C^K maximal cliques in the graph (for K hypotheses and a maximum of C speakers per hypothesis), which we have to compute at line 3 of the algorithm. This is a direct consequence of Lemma 3.2.

Theorem 4.1. *The DOVER-Lap label mapping algorithm (Algorithm 1) has time complexity $\mathcal{O}(C^K)$, where K is the number of input hypotheses and C is the maximum number of speakers in any hypothesis.*

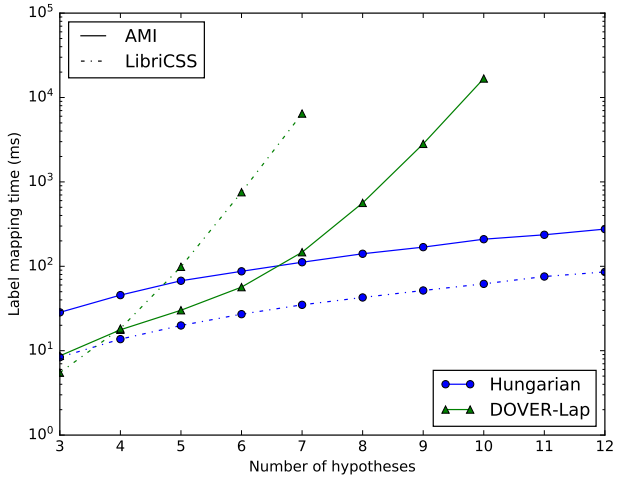

Figure 5: Label mapping time (in ms) for combining different number of hypotheses on the AMI and LibriCSS data. The y-axis is logarithmic.

Proof. From Lemma 3.2, since $\mathcal{G}$ has C^K maximal cliques, it takes at least $\mathcal{O}(C^K)$ time to simply enumerate all the cliques, as is required in the first step of the algorithm. Hence, the mapping algorithm has complexity exponential in the size of the input. □

Due to this exponential dependency, the algorithm quickly becomes computationally intractable, as shown in Fig. 5. We computed the label mapping time for an increasing number of input hypotheses for the AMI and LibriCSS evaluation sets. For AMI (which contains 4 speakers; solid green line), the algorithm became infeasible beyond $K = 10$. For LibriCSS (which contains 8 speakers; dotted green line), this limit was reached for an even smaller value of K, making combination impossible beyond 7 hypotheses. As a comparison, we also show (in blue) the mapping time for the greedy Hungarian algorithm we will describe later in § 5.

In the following sections, we will describe our proposed algorithms for label mapping, and derive approximation ratios for them. In all our derivations, we will use the multiplicative definition of approximation ratio, i.e., for a maximization problem, we say an algorithm is α-optimal if it generates solutions with objective value at least $\frac{1}{\alpha}$ of the optimal value, for any instance of the problem.

5 Greedy Hungarian mapping

Our first algorithm greedily combines independent sets in pairs using the Hungarian algorithm for linear sum assignment. Let us first set up some terminology and describe some sub-methods that will be used in the algorithm.

5.1 Definitions

Consider the subgraph $\mathcal{G}_{ij}$ induced by considering all the vertices in sets U_i and U_j, where U_i and U_j are two independent sets in $\mathcal{G}$. By construction, such a subgraph $\mathcal{G}_{ij}$ is a complete bipartite graph. For our graphical formulation, we define a local and global mapping as follows.

- A **local label mapping** ψ is a matching on the bipartite graph $\mathcal{G}_{ij}$.
- A **global label mapping** Ψ is a function that assigns a vertex u_k^i in the independent set U_k to any one of the final cliques V_c.

Our algorithm works by incrementally constructing the global label mapping using the local maps between pairs of independent sets. The local maps ψ are computed using the Hungarian method as a subroutine, which is described in the next section. We develop the following terminology to describe this method. For the sake of brevity, we drop the subscript ij for the bipartite graph $\mathcal{G}_{ij}$, and just refer to it as $\mathcal{G}$ by abuse of notation.

A *labeling* of $\mathcal{G}$ is a function $l : V \to \mathbb{R}$ such that

$$\forall \{u, v\} \in E, l(u) + l(v) \geq w(u, v).$$

An *equality subgraph* is a subgraph $\mathcal{G}_l = (V, E_l) \subseteq \mathcal{G} = (V, E)$, fixed on labeling l, such that

$$E_l = \{(u, v) \in E : l(u) + l(v) = w(u, v)\}.$$

Lemma 5.1 (The Kuhn-Munkres theorem)**.** *Given labeling l, if ψ is a perfect matching on $\mathcal{G}_l$, then ψ is a maximum-weight matching on $\mathcal{G}$.*

Proof. Let ψ' be any perfect matching in $\mathcal{G}$. By definition of a labeling function, and since ψ' is perfect,

$$w(\psi') = \sum_{(u,v)\in\psi'} w(u, v) \leq \sum_{(u,v)\in\psi'} l(u) + l(v) = \sum_{v\in V} l(v).$$

This means that $\sum_{v\in V} l(v)$ is an upper bound for any perfect matching ψ' of $\mathcal{G}$. Now, let us consider ψ.

$$w(\psi) = \sum_{(u,v)\in\psi} w(u, v) = \sum_{(u,v)\in\psi} l(u) + l(v) = \sum_{v\in V} l(v) \geq w(\psi').$$

Thus, ψ is a maximum-weight matching in $\mathcal{G}$. □

Finally, given a bipartite graph $\mathcal{G}$ and a matching ψ, the graph is said to contain an *alternating path* if there exists a path in the graph which has alternating edges in ψ. An *augmenting path* is a path in such a graph which has its endpoints (start and end vertices) unmatched (i.e., not in the matching).

5.2 The Hungarian method

An important component of our algorithm is the Hungarian method for computing the maximum-weighted matching in bipartite graphs, due to Kuhn and Munkres [Kuh55, Mun57]. In our algorithm, we use this as a subroutine to compute the local label mapping ψ In this section, we describe the algorithm and prove that it returns a maximum bipartite matching in $\mathcal{O}(n^3)$ time.

The idea behind the algorithm is to find a perfect matching on some labeling l on an equality subgraph, and use Lemma 5.1 to claim that it is a maximum-weighted matching on $\mathcal{G}$. To achieve this, we start with an empty matching $\psi = \phi$ and a valid l given as

$$l ::= \forall x \in X, y \in Y : l(y) = 0, l(x) = \max_{y'\in Y} w(x, y')$$

We then repeat the steps of *augmenting the matching* and *improving the labeling*, until we obtain a perfect matching. These two steps are outlined below.

5.2.1 Augmenting the matching

Given $\mathcal{G}_l$ and some matching ψ, we find an unmatched vertices $u, v \in V$ such that there is an augmenting path α from u to v. If such a pair of vertices exist, we create this augmenting path and flip the edges in the matching, i.e., we replace the edges in ψ with edges in the augmenting path that are in $E_l \setminus \psi$. This process increases the size of the matching, since we added previously unmatched vertices.

Algorithm 2: Hungarian label mapping

Input: Graph $\mathcal{G} = (V, E, w)$, $U = \{U_1, \ldots, U_K\}$
Output: Partition $\Phi = V_1, \ldots, V_C$

$\Psi = \{\}$ /* Global label map */
$\upsilon = U_1$
for k *in* $[2, K]$ **do**
  /* Compute local map */
  ψ = Hungarian(υ, U_k)
  /* Merge pair w.r.t. local map */
  υ = Merge(υ, U_k, ψ)
  /* Update global map */
  Ψ = Update(Ψ, ψ)
/* Compute partition using global map */
Φ = Partition(U,Ψ)

5.2.2 Improving the labeling

Let $S \subseteq X$ and $T \subseteq Y$ represent the set of vertices on either side of an "almost" augmenting path in ψ. Let $N_l(S) = \{v : \forall u \in S, (u, v) \in E_l\}$. If $N_l(S) = T$, then we cannot increase the alternating path and augment, so we must improve the labeling.

Let $\delta_l = \min_{u \in S, v \notin T}(l(u) + l(v) - w(u, v))$. We improve l to l' as

$$l'(r) = \begin{cases} l(r) - \delta_l & \text{if } r \in S, \\ l(r) + \delta_l & \text{if } r \in T, \\ l(r) & \text{otherwise.} \end{cases} \tag{6}$$

It is easy to show that l' is a valid labeling by examining all modified edges.

Using the two subroutines of augmenting and improving described above, the Hungarian method iterates until ψ is a perfect matching for $\mathcal{G}_l$.

Lemma 5.2. *The Hungarian method runs in $\mathcal{O}(n^3)$ time, where n is the number of vertices in $\mathcal{G}$.*

Proof. Each of the subroutines of augmenting the matching and improving the labeling increases the size of the matching by 1 edge. Since there can be at most $\frac{n}{2}$ edges in a matching, it takes $\mathcal{O}(n)$ rounds. Augmenting the matching requires $\mathcal{O}(n)$ time to find the right vertex, if one exists, and another $\mathcal{O}(n)$ to flip the matching. Improving the labeling also requires $\mathcal{O}(n)$ to find δ_l. However, it can occur $\mathcal{O}(n)$ times if no augmenting path is found, therefore it requires $\mathcal{O}(n^2)$ steps in a single round. Hence, the total running time is $\mathcal{O}(n^3)$. □

5.3 Algorithm

We now describe the whole algorithm for label mapping in Algorithm 2. The algorithm starts with a pair of hypotheses (independent sets), and computes a matching (local map) for them using the Hungarian method described in the previous section. It then *merges* the pair w.r.t. the map ψ. This *merge* operation is described next.

Let $\mathcal{G}_{ij}$ be a bipartite graph induces from the graph $\mathcal{G}$ by considering the independent sets U_i and U_j, as described earlier. Let ψ be a matching on $\mathcal{G}_{ij}$, s.t. $\psi = \{u_1v_1, u_2, v_2, \ldots, u_C, v_C\}$, where $u_c \in U_i$ and $v_c \in U_j$. We merge U_j with U_i by identifying the vertex pairs v_c with u_c. By Lemma 3.1, we can assume that we have performed graph completion on $\mathcal{G}$ and so there are no unmatched vertices. Let $U_{i(j)}$ denote the new vertex set. We remove the loops and edges within $U_{i(j)}$, and replace multiple edges by a single edge

with weight equal to the sum of weights of the edges it is replacing. The new weighted graph $(\mathcal{G}_{i(j)}, w')$ is called the *merge* of U_j to U_i from $\mathcal{G}$ along ψ. Clearly, $\mathcal{G}_{i(j)}$ is a (k-1)-partite graph.

In each iteration of the algorithm, we reduce the number of independent sets by 1 because of this merge operation. Furthermore, after every merge, we update the global mapping Ψ using the local map ψ. This process simply involves creating a transitive map of the form $\psi_K(\dots(\psi_2(\psi_1(\cdot))$ by composing the local label mappings at each iteration.

Theorem 5.3. *Algorithm 2 runs in polynomial time.*

Proof. From Lemma 5.2, each iteration of the loop runs in $\mathcal{O}(C)$ time, and there are a total of $K-1$ iterations. Hence, the algorithm finishes in $\mathcal{O}(CK)$ time. □

Theorem 5.4. *Algorithm 2 is a $\frac{1}{c}$-approximation for the label mapping problem.*

Proof. By Lemma 3.1, W.L.O.G, suppose $\mathcal{G}$ is complete. First we will show, by induction on K, that $w(\Phi) \geq \frac{w(\mathcal{G})}{C}$. For $K = 2$, $\mathcal{G}$ is bipartite, so the Hungarian method provides an optimal solution, i.e., Ψ is a maximum-weighted matching of $\mathcal{G}$. Since there are $C!$ perfect matchings in $\mathcal{G}$ and each edge appears in exactly $(C-1)!$ of these matchings, we have

$$\text{average weight of matching} = \frac{\sum_{e\in E} w(e)(C-1)!}{C!} = \frac{\sum_{e\in E} w(e)}{C} = \frac{w(\mathcal{G})}{C}.$$

Since the Hungarian method returns the maximum-weight matching, we have

$$w(\Phi) \geq \frac{w(\mathcal{G})}{C}.$$

For the inductive case, suppose the statement holds for some $K-1$. Let ψ_1 be the matching in the first iteration (i.e., between U_1 and U_2), and Φ' be the remaining matching. Let $\mathcal{G}'$ be the graph obtained after the first merge operation. Then, by applying the statement on ψ_1 and Φ', we have

$$\begin{aligned} w(\Phi) = w(\psi_1) + w(\Phi') &\geq \frac{1}{C} \sum_{e\in[U_1,U_2]} w(e) + \frac{w(\mathcal{G}')}{C} \\ &= \frac{1}{C}\left(\sum_{e\in[U_1,U_2]} w(e) + \sum_{e\notin[U_1,U_2]} w(e)\right) = \frac{w(\mathcal{G})}{C}. \end{aligned}$$

Now suppose Φ^* is an optimal solution. Then, since $w(\Phi^*) \leq w(\mathcal{G})$, we have

$$\begin{aligned} \frac{w(\Phi)}{w(\Phi^*)} \geq \frac{w(\Phi)}{w(\mathcal{G})} &\geq \frac{1}{C} \\ \implies w(\Phi) &\geq \frac{w(\Phi^*)}{C}. \end{aligned}$$

Hence, the algorithm is a $\frac{1}{C}$-approximation. □

6 Randomized local search

In this section, we will develop a randomized algorithm using local search which obtains a better approximation in expectation compared with the greedy Hungarian algorithm described earlier. We use the same graphical formulation that was developed previously, so we do not repeat the construction here. We will present a deterministic local search algorithm to understand the concept of local improvements in this setting, and then extend it to a randomized version.

6.1 Preliminary: deterministic local search

Again, W.L.O.G., we will assume that we have a complete graph $\mathcal{G}$, i.e., all the independent sets have exactly C vertices. The algorithms and analysis can be easily extended to the more general case without breaking any guarantees, due to Lemma 3.1. Let us first define the concepts of a feasible partition, an exchangeable pair, and neighborhood of a partition.

- A partition Φ of $\mathcal{G}$ is called *feasible* if it divides the graph into exactly C cliques, each of size K.
- A pair of vertices u_i and u_j are called *exchangeable* if both u_i and u_j belong to the same U_k for some $k \in [K]$.
- A *neighbor* Φ' of a partition Φ is a partition that can be obtained by swapping an exchangeable pair in Ψ. We denote the *neighborhood* of Φ by $N(\Phi)$.

With this terminology, we can now define local optimum for the label mapping problem. We say that a feasible partition Φ is a *local optimum* for label mapping if $w(\Phi) \geq w(\Phi')$ for any $\Phi' \in N(\Phi)$.

Lemma 6.1. *For any feasible partition Φ,*

$$|N(\Phi)| = K \cdot \binom{C}{2}.$$

Proof. To obtain a neighbor of a partition, we first choose a set U_k in K ways, and then choose an exchangeable pair in U_k in $\binom{C}{2}$ ways. Thus, there are a total of $K \cdot \binom{C}{2}$ neighbors. □

Lemma 6.2. *For any feasible partition Φ, we have*

$$\frac{1}{|N(\Phi)|} \sum_{\Phi' \in N(\Phi)} \left(w(\Phi') - \frac{w(\mathcal{G})}{C} \right) = \left(1 - \frac{2C}{|N(\Phi)|} \right) \left(w(\Phi) - \frac{w(\mathcal{G})}{C} \right).$$

Proof. Consider some neighbor Φ' of Φ, obtained by swapping the exchangeable pair (u_i, u_j) in the independent set U_k. Suppose that u_i and u_j were originally in the cliques V_i and V_j, respectively. This means that after swapping, in the partition Φ', u_i is in V_j and u_j is in V_i.

If we consider the changes from Φ to Φ' in terms of edge weights, we removed the edges from u_i to all other vertices in V_i (and similarly from u_j to all other vertices in V_j), and added edges from u_i to other vertices in V_j (and similarly for u_j to other vertices in V_i). By slight abuse of notation, suppose $w(V_i \setminus \{u_i\})$ denotes the sum of all edge weights in V_i excluding the node u_i. Then, we have

$$\begin{aligned} w(\Phi') - w(\Phi) = &\left(\sum_{u \in V_j \setminus \{u_j\}} w(u_i, u) + \sum_{u \in V_i \setminus \{u_i\}} w(u_j, u) \right) \\ &- \left(\sum_{u \in V_i \setminus \{u_i\}} w(u_i, u) + \sum_{u \in V_j \setminus \{u_j\}} w(u_j, u) \right) \end{aligned}$$

Summing over all neighbors Φ' of Φ, we get

$$\sum_{\Phi' \in N(\Phi)} \big(w(\Phi') - w(\Phi) \big) = \sum_{\Phi' \in N(\Phi)} \Bigg(\sum_{u \in V_j \setminus \{u_j\}} w(u_i, u) + \sum_{u \in V_i \setminus \{u_i\}} w(u_j, u)$$

$$- \sum_{u \in V_i \setminus \{u_i\}} w(u_i, u) - \sum_{u \in V_j \setminus \{u_j\}} w(u_j, u)\Big)$$
$$= 2 \sum_{c_i, c_j \in [C]} w(E(V_{c_i}, V_{c_j})) - 2(C-1) \sum_{c \in [C]} w(E(V_c)),$$

where $E(V_{c_i}, V_{c_j})$ denotes the set of edges going between cliques V_{c_i} and V_{c_j} in the partition Φ, and $E(V_c)$ denotes the edges inside the clique V_c. Since $w(\Phi) = \sum_{c \in [C]} w(E(V_c))$ and $w(\mathcal{G}) - w(\Phi) = \sum_{c_i, c_j \in [C]} w(E(V_{c_i}, V_{c_j}))$, we get

$$\sum_{\Phi' \in N(\Phi)} \big(w(\Phi') - w(\Phi)\big) = 2\left(w(\mathcal{G}) - w(\Phi)\right) - 2(C-1)w(\Phi)$$
$$\implies \sum_{\Phi' \in N(\Phi)} w(\Phi') - \sum_{\Phi' \in N(\Phi)} w(\Phi) = 2w(\mathcal{G}) - 2(C-1)w(\Phi)$$
$$\implies \sum_{\Phi' \in N(\Phi)} w(\Phi') - K \cdot \binom{C}{2} w(\Phi) = 2w(\mathcal{G}) - 2(C-1)w(\Phi)$$

Finally, subtracting $\frac{w(\mathcal{G})}{C}$ from both sides and dividing by $|N(\Phi)|$ (from Lemma 6.1, we obtain

$$\frac{1}{|N(\Phi)|} \sum_{\Phi' \in N(\Phi)} \left(w(\Phi') - \frac{w(\mathcal{G})}{C}\right) = \left(1 - \frac{2C}{|N(\Phi)|}\right)\left(w(\Phi) - \frac{w(\mathcal{G})}{C}\right).$$

□

We can now give a deterministic local search algorithm for label mapping. Given the graph $\mathcal{G}$, we initialize a feasible partition Ψ and then follow the steps below.

1. Search $N(\Phi)$ until we find Φ' such that
$$\left(w(\Phi') - \frac{w(\mathcal{G})}{C}\right) \geq \left(1 - \frac{2C}{|N(\Phi)|}\right)\left(w(\Phi) - \frac{w(\mathcal{G})}{C}\right).$$
2. Let $\Phi = \Phi'$.
3. If $w(\Phi) < \frac{w(\mathcal{G})}{C}$, find a new feasible partition $\Phi' \in N(\Phi)$ s.t. $w(\Phi') > w(\Phi)$. Let $\Phi = \Phi'$.

We repeat these steps until we find Φ such that $w(\Phi) \geq \frac{w(\mathcal{G})}{C}$. Note that Lemma 6.2 guarantees the existence of Φ' satisfying the conditions in both steps 1 and 3.

Theorem 6.3. *Starting from an arbitrary initialization, the local search algorithm reaches a feasible partition Φ with weight $w(\Phi) \geq \frac{w(\mathcal{G})}{C}$ in polynomial number of steps.*

Proof. In step 1 of the algorithm, we can find a feasible partition Φ' in at most $|N(\Phi)|$ searches, which is at most $K \cdot \binom{C}{2}$. So we now need to count how many such iterations are required until we obtain a solution satisfying the stopping criterion.

Let us denote $f(\Phi) = w(\Phi) - \frac{w(\mathcal{G})}{C}$, for ease of notation. This means that in step 1, our local improvements are of the form $f(\Phi') \geq \left(1 - \frac{2C}{|N(\Phi)|}\right) f(\Phi)$. Suppose we initialize with a partition Φ_0, and the consecutive partitions obtained in step 1 are Φ_1, Φ_2, and so on. Then,

Algorithm 3: Randomized local search

Input: Graph $\mathcal{G} = (V, E, w)$
Output: Partition $\Phi = V_1, \ldots, V_C$

1 $\Phi = \{\}$
/* Repeat for N epochs */
2 **for** n *in* $[N]$ **do**
/* Initialize a partition at random */
3 $\tilde{\Phi}$ = Random(V)
/* Repeat for M iterations */
4 **for** m *in* $[M]$ **do**
/* Select edge between cliques */
5 $u_p v_q$ = Sample($E(\tilde{\Phi}^C)$, $p = \frac{w(u_p v_q)}{w(\tilde{\Phi}^C)}$)
/* Swap incident vertex */
6 Swap u_p and u'_p with probability $\frac{1}{2}$
7 Swap v_q and v'_q with probability $\frac{1}{2}$
/* Update if weight increases */
8 Φ = max(Φ, $\tilde{\Phi}$)

$$f(\Phi_i) \geq f(\Phi_0) \left(1 - \frac{2C}{|N(\Phi_0)|}\right)^i.$$

This means that for sufficiently large i, the RHS in the inequality becomes diminishingly small, and $f(\Phi_i) \geq 0$, which is our desired stopping criterion. Otherwise, step 3 of the algorithm will produce a different feasible solution Φ' with weight $w(\Phi') > w(\Phi)$. We can also solve for i to bound the number of iterations as $\log_b w(\mathcal{G}) + 1$, where $b = \frac{|N(\Phi)|}{|N(\Phi)|-2C}$, and so $i* \leq n \ln w(\mathcal{G})$, which implies that the algorithm is polynomial if the weights are bounded. In our case, $w(\mathcal{G}) \leq |E|$, so the algorithm is polytime. □

Clearly, the local search algorithm is also a $\frac{1}{C}$-approximation, using similar arguments as in Theorem 5.4. Next, we will present a randomized version of this algorithm that provides a better solution in expectation.

6.2 Extension to randomized case

To improve the approximation ratio of local search, we make two changes. First, instead of arbitrarily selecting a neighbor Φ' of Φ by choosing some exchangeable pair to swap, we make this selection based on some probability distribution. Second, we repeat the local search process for enough iterations and choose the solution with the maximum weight. We will show that this process gives us a $(1 - \epsilon)$-approximate solution with high probability.

First, we describe the selection of an exchangeable pair. Let $E(\Phi^C)$ denote the complement set of the matching, i.e., the set of edges which are not in the matching Φ. Let $e = (u_{ik}, u_{j\kappa})$ denote some edge in $E(\Psi^C)$, s.t. $u_{ik} \in U_k$ and $u_{j\kappa} \in U_\kappa$. We select edge e with probability $\frac{w(e)}{w(\Phi^C)}$. This means that a higher-weighted edge not in the matching is more likely to get selected. Once we have selected an edge e, we choose either of its incident vertices u_{ik} or $u_{j\kappa}$ with probability $\frac{1}{2}$. If u_{ik} is selected, we select a vertex $v_k \in U_k$ ($v_k \neq u_{ik}$) with uniform probability distribution (and analogously for $u_{j\kappa}$). Then, our exchangeable pair is (u_{ik}, v_k).

With this selection of exchangeable pair, our entire randomized local search algorithm for label mapping is shown in Algorithm 3. The local search runs for M *iterations*, and we repeat the whole process for N *epochs*. Finally, we return the partition with the maximum weight among all the N epochs.

Theorem 6.4. *With probability* $1 - \frac{1}{e}$*, Algorithm 3 returns a* $(1-\epsilon)$*-approximate solution for*

$$N = \frac{(C!)^{K-1}}{\left(1 + \epsilon^2/(4(C-1)^2(1-\epsilon)^2)\right)^{C(K-1)}}$$
$$\textit{and} \quad M = C(K-1) \cdot \frac{(C!)^{K-1}}{\left(1 + \epsilon^2/\left(4(C-1)^2(1-\epsilon)^2\right)\right)^{C(K-1)}}.$$

Proof. Consider an optimum partition Φ_{OPT} with weight l. If, at some moment, $w(\Phi) = u \geq (1-\epsilon)l$, then we are done; otherwise $w(\Phi) = u < (1-\epsilon)l$. Let Φ^C_{OPT} and Φ^C denote the complement edge sets of the optimal partition and the current partition, respectively. We have,

$$\begin{aligned}
& u < (1-\epsilon)l \\
\implies & w(\mathcal{G}) - u > w(\mathcal{G}) - (1-\epsilon)l \\
\implies & w(\Phi^C) > (w(\mathcal{G}) - l) + \epsilon l \\
\implies & w(\Phi^C) > w(\Phi^C_{OPT}) + \epsilon l \\
\implies & w(\Phi^C) - w(\Phi^C_{OPT}) > \frac{\epsilon u}{1-\epsilon}.
\end{aligned}$$

We note that any edge in the set $\Phi^C \setminus \Phi^C_{OPT}$ is *wrongly* assigned, since it is present within cliques in the optimal partition. If such an edge is selected in line 5 of the algorithm, then the algorithm would correct it with probability $\frac{1}{2} \times \frac{1}{C-1}$. Since each edge is chosen with probability $\frac{w(e)}{w(\Phi^C)}$, the probability that the algorithm corrects at least one wrong entry is

$$\begin{aligned}
\sum_{e \in \Phi^C \setminus \Phi^C_{OPT}} \frac{1}{2(C-1)} \frac{w(e)}{w(\Phi^C)} &= \frac{1}{2(C-1)} \frac{w(\Phi^C \setminus \Phi^C_{OPT})}{w(\Phi^C)} \\
&> \frac{1}{2(C-1)} \frac{\epsilon u}{(1-\epsilon)u} \\
&= \frac{\epsilon}{2(C-1)(1-\epsilon)}.
\end{aligned}$$

Suppose in some epoch, we choose a initial partition Φ with t_k wrongly assigned vertices in set U_k, for $2 \leq k \leq K$ (i.e., t_k vertices are assigned to wrong cliques). This happens with probability $\prod_{k=2}^K p_k$, where

$$p_k = \begin{cases} \binom{C}{t_k} \frac{(t_k-1)!}{C!}, & t_k \geq 2 \\ \frac{1}{C!} & t_k = 0 \end{cases}$$

For such a partition, the algorithm finds an optimum partition with probability at least

$$\prod_{k=2}^{K} \left(\frac{\epsilon}{2(C-1)(1-\epsilon)} \right)^{t_k}.$$

Summing over all possible choices of $(t_2, \ldots, t_K)$, we conclude that the probability of using local search to find one initial partition is at least

$$\begin{aligned}
& \tfrac{1}{(C!)^{K-1}} \textstyle\prod_{k=2}^K \left(1 + \sum_{t_k=2}^K \left(\binom{K}{t_k} \right) (t_k - 1)! \left(\frac{\epsilon}{2(C-1)(1-\epsilon)} \right)^{t_k} \right) \\
& \geq \tfrac{1}{(C!)^{K-1}} \left(1 + \tfrac{\epsilon^2}{4(k-1)^2(1-\epsilon)^2} \right)^{C(K-1)} = \tfrac{1}{N}
\end{aligned}$$

Since we choose $N = \frac{(C!)^{K-1}}{(1+\epsilon^2/(4(C-1)^2(1-\epsilon)^2))^{C(K-1)}}$ initial independent partitions, the probability of fail is at most

$$(1 - \frac{1}{N})^N < \frac{1}{e}.$$

Hence, we obtain a $(1-\epsilon)$-approximate solution with probability at least $1 - \frac{1}{e}$.

□

7 Experimental results

We performed experiments on the AMI meeting corpus [CAB+05]. AMI consists of 100 hours of recorded meetings containing 4 or 5 speakers per session, with speech from close-talking, single distant microphone (SDM), and array microphones. About 20% of the speech duration contains overlaps, on average. For our experiments, we used the mixed-headset recordings, which are obtained by summing the individual headset signals from the participants in the meeting. We combined the diarization outputs from the following overlap-aware diarization methods.

1. **VB-based overlap assignment (VB)** [BBGP20]: This method leverages Variational Bayes (VB)-HMM used originally for diarization in [DBM18]. The model is a Bayesian HMM, in which states represent speaker specific distributions and transitions between states represent speaker changes. Using the output of an externally trained overlap detector, overlapping frames are assigned the top two speakers from the posterior matrix computed using VB inference. We used a single-speaker spectral clustering system to initialize the matrix with hard probabilities [PHKN20], and an oracle speech activity detector (SAD) to remove non-speech segments.

2. **Overlap-aware spectral clustering (SC)** [RHK21]: This method also uses an external overlap detector, but unlike VB, overlap assignment happens in the first-pass clustering itself, instead of during resegmentation. This is done by reformulating spectral clustering as a constrained optimization problem, and then discretizing it under the overlap constraints. We used the same SAD, x-vector extractor, and overlap detector for the VB and SC methods.

3. **Region proposal networks (RPN)** [HWF+20]: It combines segmentation and embedding extraction into a single neural network, and jointly optimizes them using an objective function that consists of boundary prediction and speaker classification components. The region embeddings are then clustered (using K-means clustering) and a non-maximal suppression is applied. For AMI, we trained the RPN on force-aligned data from the AMI training set; for LibriCSS, it was trained on simulated meeting-style recordings with partial overlaps generated using utterances from the LibriSpeech [PCPK15] training set. Since we used K-means clustering, we assumed that the oracle number of speakers for each recording is known. A post-processing step was applied using oracle SAD segments to filter non-speech.

For the randomized local search algorithm, we adopted an early stopping method where we stopped the procedure when the objective function value did not increase for 100 epochs. For the greedy Hungarian method, we also experimented with sorting the hypotheses in increasing order of average DER with all other hypotheses before starting the mapping process. For both the algorithms, we used the label voting technique described in the DOVER-Lap paper [RGPH+21] for the final combination after the mapping stage. We used `spyder`[2] for DER-based sorting in the Hungarian method, and also for evaluating the final performances.

[2] `https://github.com/desh2608/spyder`

Table 3: Comparison of Hungarian (modified DOVER) and randomized local search label mapping methods. Results are shown on the AMI evaluation set, in terms of missed speech (MS), false alarm (FA), speaker error (SE), and diarization error rate (DER). We combined 3 overlap-aware hypotheses: overlap-aware spectral clustering (SC), VB-based overlap assignment, and regional proposal networks (RPN).

Method	**MS**	**FA**	**SE**	**DER**
Overlap-aware SC [RHK21]	11.48	2.27	9.81	23.56
VB-based overlap assignment [BBGP20]	9.84	**2.06**	9.60	21.50
Region proposal networks [HWF$^+$20]	**9.49**	7.68	8.25	25.42
Greedy Hungarian	9.91	2.71	8.56	21.58
+ DER-based sorting	9.79	2.93	8.20	20.92
Randomized local search (RLS)	9.69	3.21	**7.84**	**20.74**

Table 3 shows the results for the proposed label mapping algorithms on the AMI evaluation set. We combined the 3 overlap-aware diarization systems. As expected, RLS outperforms the greedy Hungarian algorithm, and most of the gains come from lower speaker error (7.84% compared with 8.20%). However, this difference in performance is fairly small, especially when we consider that the RLS method requires a longer processing time. This may be because the theoretical bounds are designed to hold in the setting when the size of inputs is fairly large. In our setting of combining diarization hypothesis, these "large number" assumptions are violated. Furthermore, improving the objective in (4) is not monotonically related to an improvement in DER, as seen in Fig. 4.

8 Conclusion

By formulating label mapping as a graph partitioning problem, we showed that the algorithm used previously in DOVER-Lap is exponential in the input size and becomes intractable as the number of hypotheses increases. We then proposed two algorithms for this problem — a greedy algorithm based on the Hungarian method, and a randomized local search algorithm. We derived approximation ratios for both algorithms and showed that the RLS algorithm obtains a better approximation in expectation. We also demonstrated our algorithms empirically by combining 3 overlap-aware diarization systems on the AMI meeting data, where they outperformed the single best system.

References

[BBGP20] Latané Bullock, Hervé Bredin, and L. P. García-Perera. Overlap-aware diarization: Resegmentation using neural end-to-end overlapped speech detection. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7114–7118, 2020.

[CAB$^+$05] Jean Carletta, Simone Ashby, Sebastien Bourban, Mike Flynn, Maël Guillemot, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Wessel Kraaij, Melissa Kronenthal, Guillaume Lathoud, Mike Lincoln, Agnes Lisowska Masson, Iain McCowan, Wilfried Post, Dennis Reidsma, and Pierre Wellner. The AMI meeting corpus: A pre-announcement. In *MLMI*, 2005.

[CHS17] W. Chen, W. Hendrix, and N. Samatova. The application of the weighted k-partite graph problem to the multiple alignment for metabolic pathways. *Journal of computational biology : a journal of computational molecular cell biology*, 24 12:1195–1211, 2017.

[DBM18] Mireia Díez, Lukás Burget, and Pavel Matejka. Speaker diarization based on Bayesian HMM with eigenvoice priors. In *Proc. Odyssey Speaker and Language Recognition Workshop*, pages 147–154, June 2018.

[EJZ00] Geoffrey Exoo, JL Jiang, and Cheng Zhao. An application of multi-dimensional matching to logistics. *Utilitas Mathematica*, 57:227–235, 2000.

[Fis97] J. Fiscus. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover). *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*, pages 347–354, 1997.

[FWH+20] Yusuke Fujita, Shinji Watanabe, Shota Horiguchi, Yawen Xue, and Kenji Nagamatsu. End-to-end neural diarization: Reformulating speaker diarization as simple multi-label classification. *ArXiv*, abs/2003.02966, 2020.

[GRSS+17] Daniel Garcia-Romero, David Snyder, Gregory Sell, Daniel Povey, and Alan McCree. Speaker diarization using deep neural network embeddings. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4930–4934, 2017.

[HLZ00] George He, J. Liu, and C. Zhao. Approximation algorithms for some graph partitioning problems. *J. Graph Algorithms Appl.*, 4:1–11, 2000.

[HWF+20] Zili Huang, Shinji Watanabe, Yusuke Fujita, Paola García, Yiwen Shao, Daniel Povey, and S. Khudanpur. Speaker diarization with region proposal network. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6514–6518, 2020.

[Kuh55] Harold W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, May 1955.

[LC90] J. Li and M. Chen. Index domain alignment: minimizing cost of cross-referencing between distributed arrays. *[1990 Proceedings] The Third Symposium on the Frontiers of Massively Parallel Computation*, pages 424–433, 1990.

[LPDB20] F. Landini, Jan Profant, M. Díez, and L. Burget. Bayesian HMM clustering of x-vector sequences (VBx) in speaker diarization: theory, implementation and analysis on standard tasks. *ArXiv*, abs/2012.14952, 2020.

[LPZ06] J. Liu, Y. Peng, and C. Zhao. Generalized k-multiway cut problems. *Journal of Applied Mathematics and Computing*, 21:69–82, 2006.

[MBE+12] Xavier Anguera Miró, Simon Bozonnet, Nicholas W. D. Evans, Corinne Fredouille, Gerald Friedland, and Oriol Vinyals. Speaker diarization: A review of recent research. *IEEE Transactions on Audio, Speech, and Language Processing*, 20:356–370, 2012.

[MK13] Mohammad Mirghorbani and P. Krokhmal. On finding k-cliques in k-partite graphs. *Optimization Letters*, 7:1155–1165, 2013.

[MKP+20] I. Medennikov, M. Korenevsky, Tatiana Prisyach, Yuri Y. Khokhlov, Mariya Korenevskaya, I. Sorokin, T. Timofeeva, A. Mitrofanov, A. Andrusenko, Ivan Podluzhny, A. Laptev, and A. Romanenko. Target-speaker voice activity detection: a novel approach for multi-speaker diarization in a dinner party scenario. *ArXiv*, abs/2005.07272, 2020.

[Mun57] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of The Society for Industrial and Applied Mathematics*, 10:196–210, 1957.

[PCPK15] Vassil Panayotov, Guoguo Chen, D. Povey, and S. Khudanpur. Librispeech: An ASR corpus based on public domain audio books. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210, 2015.

[PHKN20] Tae Jin Park, Kyu J. Han, Manoj Kumar, and Shrikanth S. Narayanan. Auto-tuning spectral clustering for speaker diarization using normalized maximum eigengap. *IEEE Signal Processing Letters*, 27:381–385, 2020.

[RCC+19] Neville Ryant, Kenneth Ward Church, C. Cieri, Alejandrina Cristia, J. Du, Sriram Ganapathy, and M. Liberman. The second dihard diarization challenge: Dataset, task, and baselines. In *INTERSPEECH*, 2019.

[RDC+21] Desh Raj, Pavel Denisov, Z. Chen, Hakan Erdogan, Zili Huang, Mao-Kui He, Shinji Watanabe, Jun Du, T. Yoshioka, Yi Luo, Naoyuki Kanda, Jinyu Li, S. Wisdom, and J. Hershey. Integration of speech separation, diarization, and recognition for multi-speaker meetings: System description, comparison, and analysis. *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 897–904, 2021.

[RGPH+21] Desh Raj, L. P. García-Perera, Zili Huang, Shinji Watanabe, Daniel Povey, A. Stolcke, and S. Khudanpur. Dover-lap: A method for combining overlap-aware diarization outputs. *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 881–888, 2021.

[RHK21] Desh Raj, Zili Huang, and Sanjeev Khudanpur. Multi-class spectral clustering with overlaps for speaker diarization. *IEEE Spoken Language Technology Workshop (SLT)*, 2021.

[RSK+20] Neville Ryant, Prachi Singh, Venkat Krishnamohan, Rajat Varma, Kenneth Church, C. Cieri, Jun Du, Sriram Ganapathy, and M. Liberman. The third dihard diarization challenge. *ArXiv*, abs/2012.01477, 2020.

[SXB08] R. Singh, J. Xu, and B. Berger. Global alignment of multiple protein interaction networks with application to functional orthology detection. *Proceedings of the National Academy of Sciences*, 105:12763 – 12768, 2008.

[SY19] Andreas Stolcke and Takuya Yoshioka. DOVER: A method for combining diarization outputs. *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 757–763, 2019.

[TR06] Sue Tranter and Douglas A. Reynolds. An overview of automatic speaker diarization systems. *IEEE Transactions on Audio, Speech, and Language Processing*, 14:1557–1565, 2006.

[WMBV20] Shinji Watanabe, Michael Mandel, J. Barker, and E. Vincent. Chime-6 challenge: Tackling multispeaker speech recognition for unsegmented recordings. *ArXiv*, abs/2004.09249, 2020.

[XKC+20] Xiong Xiao, Naoyuki Kanda, Z. Chen, Tianyan Zhou, T. Yoshioka, Sanyuan Chen, Y. Zhao, Gang Liu, Y. Wu, J. Wu, Shujie Liu, Jinyu Li, and Yifan Gong. Microsoft speaker diarization system for the voxceleb speaker recognition challenge 2020. *ArXiv*, abs/2010.11458, 2020.

[ZZ02] M. Zhou and Chen Zhao. An optimization model and multiple matching heuristics for quality planning in manufacturing systems. *Computers & Industrial Engineering*, 42:91–101, 2002.