Speeding Up Permutation Invariant Training for Source Separation

Thilo von Neumann,[1] *Christoph Boeddeker,*[1] *Keisuke Kinoshita,*[2] *Marc Delcroix,*[2]
Reinhold Haeb-Umbach[1]

[1]Paderborn University, Germany; [2]NTT Corporation, Japan
Email: {vonneumann,boeddeker,haeb}@nt.upb.de

Abstract

Permutation invariant training (PIT) is a widely used training criterion for neural network-based source separation, used for both utterance-level separation with utterance-level PIT (uPIT) and separation of long recordings with the recently proposed Graph-PIT. When implemented naively, both suffer from an exponential complexity in the number of utterances to separate, rendering them unusable for large numbers of speakers or long realistic recordings. We present a decomposition of the PIT criterion into the computation of a matrix and a strictly monotonously increasing function so that the permutation or assignment problem can be solved efficiently with several search algorithms. The Hungarian algorithm can be used for uPIT and we introduce various algorithms for the Graph-PIT assignment problem to reduce the complexity to be polynomial in the number of utterances.

1 Introduction

Speech source separation is an important pre-processing step when applying speech recognition or diarization to realistic recordings of meeting scenarios. It aims at estimating speech signals of individual speakers from a speech mixture. We can in general discern utterance-level algorithms [1–3] and Continuous Speech Separation (CSS) [4, 5] algorithms. Utterance-level separation works on relatively short recordings of the length of roughly one utterance and aims at outputting each source on an individual output channel. CSS algorithms operate on a continuous stream of audio of arbitrary length and allow output channels to contain different speakers as long as they do not overlap.

Recently, neural networks have shown strong separation performance, even if only single-channel recordings are available [2, 3, 6]. A state-of-the-art technique is Permutation Invariant Training (PIT) of a separation network. Utterance-level PIT (uPIT) [1] finds the optimal permutation between the target signals and the network output channels during training. This gives the network the freedom of selecting an output channel for each speaker.

We can use a network trained with uPIT to also realize CSS by use of a stitching scheme [4, 5] during test time. The input audio stream is segmented into relatively short, temporally overlapping segments so that separation can be performed on each segment independently. Neighboring segments are aligned using a similarity metric, which is computed on regions that are common to neighboring segments. The maximum number of active speakers must not exceed the number of output channels in any segment. This effectively limits the maximum segment size, because the number of output channels is typically chosen to be small, e.g., equal to two, and in multi-party meetings, long segments would typically comprise more than two speakers. The alignment of adjacent segments also introduces additional computational overhead.

Recently, a technique called Graph-PIT [7] was introduced to relax this limitation by training a neural network to output more than one speaker on a single output channel. Thus, Graph-PIT gives the freedom to increase the segment size of stitching-based CSS significantly without loss in performance, reducing the computational overhead, or even eliminating stitching completely in certain scenarios. We here focus only on these scenarios, i.e., we do not consider stitching. Graph-PIT works by placing separated utterances on output channels solely under the constraint that overlapping utterances are placed on different output channels. This transforms the permutation problem of uPIT to a more general many-to-one assignment of utterances to output channels which we formulate as a graph coloring problem, as visualized in Fig. 1.

However, graph coloring is an NP-hard problem and thus has an exponential complexity [8]. While the complexity is not problematic for small models with two output channels, it explodes for more outputs, limiting the use of the original formulation [7] to small numbers of output channels and short recordings.

Both, uPIT and Graph-PIT, have a factorial or exponential complexity in the number of utterances in the mixture signal when implemented naively [1, 7]. It was shown that for uPIT, the optimal assignment of target signals to output channels can be found with the Hungarian algorithm [9, 10] in polynomial time. This makes it practical to train a model to separate up to 20 speakers [11]. A similar approach is not possible for Graph-PIT.

We show that an elegant decomposition of the loss function into a score matrix and a strictly monotonously increasing function always allows us to use more efficient algorithms for solving the uPIT and the Graph-PIT assignment problems.[1] We review how the Hungarian algorithm can be used to solve the permutation problem in uPIT in polynomial time, and present a variety of assignment algorithms for Graph-PIT. We found that the assignment problem can be formulated in a way such that dynamic programming can be applied to provide a solution in linear time in the number of utterances.

2 Problem Formulation

We consider the task of source separation, i.e., separating a speech mixture signal $\mathbf{y}$ containing overlapping speech of K speakers into its individual speech components. We here only consider single-channel separation for simplicity, but the presented algorithms can as well be used to train multi-channel systems. The speakers utter U utterances $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_U] \in \mathbb{R}^{T \times U}$ where each utterance signal $\mathbf{s}_u \in \mathbb{R}^T$ is represented as a vector of samples and zero-padded to the full length of the meeting T. The meeting signal $\mathbf{y}$ is the sum of these zero-padded utterance signals

$$\mathbf{y} = \sum_{u=1}^{U} \mathbf{s}_u. \tag{1}$$

The general task is to separate $\mathbf{y}$ into C estimated signals $\hat{\mathbf{S}} = [\hat{\mathbf{s}}_1, \hat{\mathbf{s}}_2, ..., \hat{\mathbf{s}}_C] \in \mathbb{R}^{T \times C}$ so that no speech parts overlap. We here only consider neural network-based separation with PIT objectives where the network has a fixed number of C output channels and is trained with a Signal-to-Distortion Ratio (SDR)-based loss function. We look into two scenarios: utterance-level and meeting-level separation.

Utterance-level separation

Utterance-level separation targets mixtures in which each speaker says exactly one utterance, and a separation system with $C = K$ output channels. The task here is to place one utterance $\mathbf{s}_k$ onto each output channel.

Meeting-level separation

The case of meeting-level CSS allows for $K > C$ as long as never more than C speakers speak at the same time. Exclusive placement of utterance different output channels, as is done in utterance-level separation, is not possible due to $K > C$ in general. Instead, the placement of separated utterances on output channels is chosen such that no two utterances overlap in a channel.

[1]Code is available at https://github.com/fgnt/graph_pit

3 Variants of SDR

A commonly used loss function for training of time-domain source separation networks is the SDR [2, 3, 12, 13]. It is generally defined in the logarithmic domain as

$$\mathcal{L}^{(\text{SDR})}(\hat{\mathbf{s}}, \mathbf{s}) = -10 \log_{10} \frac{\|\mathbf{s}\|^2}{\|\mathbf{s} - \hat{\mathbf{s}}\|^2}. \tag{2}$$

As a loss function for a source separator, output channels are usually treated independently. The loss is aggregated as the average over the output channels [2]. Neglecting the permutation ambiguity for the moment, this gives the averaged SDR (a-SDR)

$$\mathcal{L}^{(\text{a-SDR})}(\hat{\mathbf{S}}, \mathbf{S}) = -\frac{1}{C} \sum_{c=1}^{C} 10 \log_{10} \frac{\|\mathbf{s}_c\|^2}{\|\mathbf{s}_c - \hat{\mathbf{s}}_c\|^2}. \tag{3}$$

Although $\mathcal{L}^{(\text{a-SDR})}$ is commonly used, it becomes problematic in certain cases, e.g., for low volume or short utterances [14]. It does not allow speeding up Graph-PIT as we discuss later in Section 5. These problems can be eliminated by, instead of averaging the SDRs, summing the energies of the target and error signals [14]:

$$\mathcal{L}^{(\text{sa-SDR})}(\hat{\mathbf{S}}, \mathbf{S}) = -10 \log_{10} \frac{\sum_{c=1}^{C} \|\mathbf{s}_c\|^2}{\sum_{c=1}^{C} \|\mathbf{s}_c - \hat{\mathbf{s}}_c\|^2}. \tag{4}$$

We call Eq. (4) the source-aggregated SDR (sa-SDR). Similar modifications are possible for other objectives, e.g., SI-SDR [13] or log-MSE [12], but we restrict ourselves to the SDR here. We write $\sum_{c=1}^{C} \|\mathbf{s}_c\|^2$ as a matrix operation $\text{Tr}(\mathbf{S}^\mathsf{T}\mathbf{S})$ in the following equations.

4 Speeding up utterance-level PIT

One approach to utterance-level separation is uPIT [1], where the network outputs one signal for each speaker. We assume for simplicity that $C = K = U$ so that the signals $\mathbf{s}_u$ can be directly used as targets. The permutation between speakers and output channels is ambiguous during training; uPIT postulates to choose the permutation that yields the smallest loss. It is often defined for losses $\mathcal{L}^{(\text{pair})}$, such as Eq. (2), that sum over losses between pairs of targets and estimates [1]

$$\mathcal{J}^{(\text{uPIT})}(\hat{\mathbf{S}}, \mathbf{S}) = \min_{\pi \in \Pi_C} \sum_{c=1}^{C} \mathcal{L}^{(\text{pair})}(\hat{\mathbf{s}}_c, \mathbf{s}_{\pi(c)}), \tag{5}$$

where Π_C is the set of all permutations of length C. This formulation is not compatible with $\mathcal{L}^{(\text{sa-SDR})}$ in Eq. (4). A more general definition can be found with a permutation matrix [15]:

$$\mathcal{J}^{(\text{uPIT})}(\hat{\mathbf{S}}, \mathbf{S}) = \min_{\mathbf{P} \in \mathcal{P}_C} \mathcal{L}(\hat{\mathbf{S}}, \mathbf{S}\mathbf{P}), \tag{6}$$

where $\mathcal{L}$ is a loss function over multiple pairs of targets and estimations, such as Eqs. (3) and (4), and $\mathcal{P}_C$ is the set of all permutation matrices $\mathbf{P} \in \{0,1\}^{C \times C}$. All entries in $\mathbf{P}$ are 0 except for exactly one 1 in each row and each column.

The permutation matrix $\hat{\mathbf{P}}$ that minimizes Eq. (6) can be found naively by computing the loss for all possible permutations and selecting the one that yields the smallest loss. This has a complexity of $\mathcal{O}(C!)$ because $|\mathcal{P}_C| = C!$ [1], so it is only applicable for relatively small numbers of speakers.

If, for a certain $\mathcal{L}$, uPIT can be expressed with Eq. (6), we can find a score matrix $\mathbf{M} \in \mathbb{R}^{C \times C}$ with the elements $m_{i,j} = \mathcal{L}^{(\text{pair})}(\hat{\mathbf{s}}_i, \mathbf{s}_j)$ so that [15]

$$\mathcal{J}^{(\text{uPIT})}(\hat{\mathbf{S}}, \mathbf{S}) = \min_{\pi \in \Pi_C} \sum_{c=1}^{C} \mathcal{L}^{(\text{pair})}(\hat{\mathbf{s}}_c, \mathbf{s}_{\pi(c)}) = \min_{\mathbf{P} \in \mathcal{P}_C} \text{Tr}(\mathbf{M}\mathbf{P}). \tag{7}$$

Finding the permutation matrix $\hat{\mathbf{P}}$ that minimizes Eq. (7) becomes a linear sum assignment problem [15], i.e., it is equivalent to selecting exactly one entry in each row and column of $\mathbf{M}$ such that their sum is minimized. This can be solved by the Hungarian algorithm in $\mathcal{O}(C^3)$ time [9, 10, 15].

If a representation as in Eq. (7) is not possible, e.g., as for Eq. (4), we can still find $\hat{\mathbf{P}}$ with the Hungarian algorithm if $\mathcal{J}$ can be expressed as[2]

$$\mathcal{J}^{(\text{uPIT})}(\hat{\mathbf{S}}, \mathbf{S}) = f(\min_{\mathbf{P} \in \mathcal{P}_C} \text{Tr}(\mathbf{M}\mathbf{P}), \hat{\mathbf{S}}, \mathbf{S}), \tag{8}$$

with $\mathbf{M} \in \mathbb{R}^{C \times C}$ and a strictly monotonously increasing function f, as we will demonstrate in the following. The idea for the f follows argmax rules and the inner representation for the permutation problem is known from [15]. Eq. (8) equals Eq. (7) for $f(x, \hat{\mathbf{S}}, \mathbf{S}) = x$ and $m_{i,j} = \mathcal{L}^{(\text{pair})}(\hat{\mathbf{s}}_i, \mathbf{s}_j)$ with $x = \min_{\mathbf{P} \in \mathcal{P}_C} \text{Tr}(\mathbf{M}\mathbf{P})$. The decomposition is often not unique and the choice of f and $\mathbf{M}$ impacts the computational cost. Further, some decompositions that can be found for uPIT are not usable for Graph-PIT.

Decomposing the Signal-to-Distortion Ratio (SDR)

The averaged SDR, Eq. (3), can be expressed with $f(x, \hat{\mathbf{S}}, \mathbf{S}) = x$ and $m_{i,j} = \text{SDR}(\hat{\mathbf{s}}_i, \mathbf{s}_j)$. For the alternative, $\mathcal{L}^{(\text{sa-SDR})}$, we get

$$\mathcal{J}^{(\text{uPIT})} = \min_{\mathbf{P} \in \mathcal{P}_C} -10 \log_{10} \frac{\text{Tr}(\mathbf{S}^\mathsf{T}\mathbf{S})}{\text{Tr}\big((\mathbf{S}\mathbf{P} - \hat{\mathbf{S}})^\mathsf{T}(\mathbf{S}\mathbf{P} - \hat{\mathbf{S}})\big)} \tag{9}$$

$$= -10 \log_{10} \frac{\text{Tr}(\mathbf{S}^\mathsf{T}\mathbf{S})}{\min_{\mathbf{P} \in \mathcal{P}_C} \text{Tr}\big((\mathbf{S}\mathbf{P} - \hat{\mathbf{S}})^\mathsf{T}(\mathbf{S}\mathbf{P} - \hat{\mathbf{S}})\big)}. \tag{10}$$

Although Eq. (10) looks complicated due to the matrix notation, the denominator is the Mean Squared Error (MSE) between targets and permuted estimations. The permutation $\mathbf{P}$ can be found by minimizing the MSE. Comparing with Eq. (8) and factorizing the MSE, we get

$$f^{(\text{sa-SDR-MSE})}(x, \hat{\mathbf{S}}, \mathbf{S}) = -10 \log_{10} \frac{\text{Tr}(\mathbf{S}^\mathsf{T}\mathbf{S})}{T \cdot x}$$

$$= -10 \log_{10} \frac{\sum_{c=1}^{C} \|\mathbf{s}_c\|^2}{T \cdot x}, \tag{11}$$

$$m_{i,j}^{(\text{sa-SDR-MSE})} = \frac{1}{T} \big\|\mathbf{s}_i - \hat{\mathbf{s}}_j\big\|^2, \tag{12}$$

where T is the time length of the signals. We can further simplify the denominator in Eq. (10) to find

$$f^{(\text{sa-SDR-dot})}(x, \hat{\mathbf{S}}, \mathbf{S}) = -10 \log_{10} \frac{\text{Tr}(\mathbf{S}^\mathsf{T}\mathbf{S})}{\text{Tr}(\mathbf{S}^\mathsf{T}\mathbf{S}) + \text{Tr}\big(\hat{\mathbf{S}}^\mathsf{T}\hat{\mathbf{S}}\big) + 2x}, \tag{13}$$

$$\mathbf{M}^{(\text{sa-SDR-dot})} = -\hat{\mathbf{S}}^\mathsf{T}\mathbf{S}. \tag{14}$$

The elements of $\mathbf{M}$ are the dot products between pairs of estimations and targets $m_{i,j} = \hat{\mathbf{s}}_i^\mathsf{T}\mathbf{s}_j$. They are computed with a matrix multiplication which is efficient on modern computer hardware. We show in Section 5 that the decomposition in Eqs. (13) and (14) is required for speeding up Graph-PIT.

5 Speeding up Graph-PIT

Graph-PIT [7] solves the meeting-level separation problem under the assumption that never more than C speakers are active at the same time, but in general $C < K$. It allows placement of utterances from different speakers on the same output channel and distributing utterances from the same speaker across different output channels as long as they never overlap during training.

Using the matrix notation, we can define Graph-PIT as a natural extension of uPIT (Eq. (6)):

$$\mathcal{J}^{(\text{Graph-PIT})} = \min_{\mathbf{P} \in \mathcal{B}_{\mathcal{G},C}} \mathcal{L}(\hat{\mathbf{S}}, \mathbf{S}\mathbf{P}). \tag{15}$$

[2] The decomposition Eq. (8) does in general not exist, but it exists for most relevant separation objectives we are aware of.

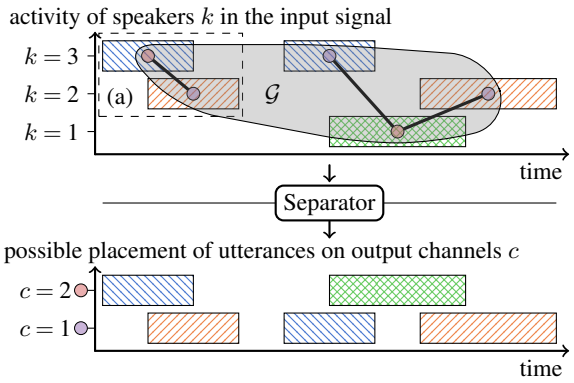

Figure 1: Example of processing a three-speaker scenario using Graph-PIT with a two-output separator. Each box represents one utterance. *Top*: Utterances in the meeting and the colored overlap graph $\mathcal{G}$. Graph-PIT is equivalent to uPIT for an activity pattern as marked with (a). *Bottom*: A possible assignment of utterances to output channels. Taken from [7].

The matrix $\mathbf{P} \in \{0,1\}^{U\times C}$ is no longer a square permutation matrix but an assignment matrix that assigns utterances to output channels, i.e., several utterances can be assigned to each output channel. This means that each row in $\mathbf{P}$ contains exactly one 1. The term $\mathbf{SP} \in \mathbb{R}^{T\times C}$ represents the target signals when the utterances are assigned to output channels with $\mathbf{P}$. The columns of $\mathbf{SP}$ are no longer only permutations of $\mathbf{S}$ but disjoint sums of utterance signals from $\mathbf{S}$.

We have to find the set $\mathcal{B}_{\mathcal{G},C}$ of matrices that are valid assignments of utterances to output channels, i.e., so that no two utterances overlap on the same output channel. This is equivalent to a graph vertex coloring problem of the undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, whose vertices $\mathcal{V}$ represent utterances and edges $\mathcal{E}$ overlaps between two utterances. Such a coloring is shown for an example graph in Fig. 1. The assignment of a color to a vertex represents the assignment of an utterance to an output channel. An edge is added between two vertices if the corresponding utterances overlap in time:

$$\mathcal{V} = \{1,...,U\}, \tag{16}$$

$$\mathcal{E} = \{\{u,v\} : u,v \in \mathcal{V}, \text{utterances } u \text{ and } v \text{ overlap}\}. \tag{17}$$

Its adjacency matrix $\mathbf{A}_\mathcal{G} \in \{0,1\}^{U\times U}$ is given by its elements

$$a_{u,v} = \begin{cases} 1, & \text{if } \{u,v\} \in \mathcal{E}, \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

A vertex coloring is usually represented as a function $\pi : \mathcal{V} \to \{1,...,C\}$ so that $\pi(u) \neq \pi(v)$ if $\{u,v\} \in \mathcal{E}$ [8]. To be consistent with Eq. (6), we represent a coloring with the assignment matrix $\mathbf{P} \in \{0,1\}^{U\times C}$, where its entries are $p_{i,\pi(i)} = 1$ for $i \in \{1,...,U\}$ and all other entries are 0. If $\mathbf{P}$ is a valid coloring it satisfies

$$\mathrm{Tr}(\mathbf{P}^\mathsf{T}\mathbf{A}_\mathcal{G}\mathbf{P}) = 0. \tag{19}$$

We denote by $\mathcal{B}_{\mathcal{G},C}$ the set of all valid C-colorings of $\mathcal{G}$.

Similar to Eq. (8) , we are interested in a decomposition for Graph-PIT where $\mathcal{L}$ can be expressed as a computation of a score matrix $\mathbf{M} \in \mathbb{R}^{C\times U}$ and a strictly monotonously increasing f:

$$\mathcal{J}^{\text{(Graph-PIT)}}(\hat{\mathbf{S}},\mathbf{S}) = f(\min_{\mathbf{P}\in\mathcal{B}_{\mathcal{G},C}} \mathrm{Tr}(\mathbf{MP}),\hat{\mathbf{S}},\mathbf{S}). \tag{20}$$

Note that $\mathbf{P}$ is no longer square and that some or all target signals $\mathbf{SP}$ in Eq. (15) are now the sum of multiple utterances. Because of this we have to carefully select our objective since we no longer know a decomposition for every relevant source separation objective, as it was the case for uPIT. We specifically introduced $\mathcal{L}^{\text{(sa-SDR)}}$ because we are not aware of a decomposition for $\mathcal{L}^{\text{(a-SDR)}}$. Further, while Eqs. (11) and (12) work for uPIT, this decomposition is not applicable for Graph-PIT because the MSE cannot be factorized if the targets are the sums of multiple target utterances. The matrix product in Eqs. (13) and (14), however, works, as it directly uses utterance signals $\mathbf{S}$ instead of the target sum signals $\mathbf{SP}$.

5.1 Finding the assignment matrix

Finding $\hat{\mathbf{P}} \in \mathcal{B}_{\mathcal{G},C}$ that minimizes Eq. (15) for a general loss function $\mathcal{L}$ requires computing the full loss for all $\mathbf{P} \in \mathcal{B}_{\mathcal{G},C}$. The decomposition in Eq. (20) on the other hand, if it exists, allows for more efficient search, where the assignment is solved on $\mathbf{M}$ instead of computing the objective for each assignment. If $\mathbf{M}$ is treated as a score matrix assigning a score to each pair of vertex and color, $\hat{\mathbf{P}}$ is the coloring of $\mathcal{G}$ with the minimal score. Compared to uPIT, the problem of finding $\hat{\mathbf{P}}$ is no longer a balanced linear sum assignment problem, since multiple utterances can be assigned to a single output channel. The overlap graph further imposes constraints. This prevents us from using the Hungarian algorithm. The following sub-sections discuss different approaches for finding $\hat{\mathbf{P}}$.

5.1.1 Optimal: Brute-force exhaustive search

The naive way for solving the assignment problem is brute-force enumeration of all possible solutions and selecting the best one. This assignment algorithm is used in [7]. It has an exponential complexity limited by the number of colorings $\mathcal{O}(C(C-1)^{U-1})$.

5.1.2 Greedy: Depth First Search

A greedy but not necessarily optimal solution can be found with (incomplete) Depth First Search (DFS) [16] in the space of partial solutions. The algorithm colors vertices sequentially so that each newly selected color adds the smallest possible score, respecting the constraints imposed by the overlap graph. If at any point a vertex cannot be colored, the latest coloring is undone and the next possible color adding the smallest score is tested. This algorithm has a best-case complexity of $\mathcal{O}(UC)$.

5.1.3 Optimal: Branch-and-Bound Search

The solution space can be searched in a more elegant way by using a branch-and-bound [17] algorithm to always find the optimal assignment. The algorithm works in the same way as the DFS algorithm, but continues search until the best solution is found. During the search process, partial solutions that have a worse score than the currently best solution are discarded immediately as extending those can never yield the lowest score.

5.1.4 Optimal: Dynamic Programming

We can significantly reduce the number of considered partial colorings by the use of Dynamic Programming (DP) [18]. Given our scenario, we can sort the utterances u by their start times and traverse $\mathcal{G}$ in that order from the first utterance $u = 1$ to the last one $u = U$. Let us denote the set of already visited utterances $v < u$ that overlap with u as $\mathcal{N}_u = \{v : v < u, \{u,v\} \in \mathcal{E}\}$. Generally, we have to consider all possible valid colorings of $\{v \in \mathcal{V} : v < u\}$ in the u-th step. It turns out to be enough to only remember one coloring, the one with the best accumulated score, for each set of colorings that share the same colors of the utterances in $\mathcal{N}_u$, due to the structure of $\mathcal{G}$. Let us call this set of colorings obtained in the u-th step Q_{u-1}. We construct an intermediate set $\tilde{Q}_u$ by extending each coloring in Q_{u-1} with each valid color c of u. The colors c must be unequal to all colors of the utterances in $\mathcal{N}_u$ for this coloring because u overlaps with all utterances in $\mathcal{N}_u$. The accumulated score is increased by $m_{c,u}$. The set Q_u, for the next step, is obtained from $\tilde{Q}_u$ by only keeping the coloring with the best accumulated score for each set of colorings that share the same coloring of $\mathcal{N}_{u+1}$. The colorings are extended by one vertex in every step, so that the best coloring of $\mathcal{G}$ is obtained after U steps as the best coloring in Q_U.

The algorithm always performs U steps and $|Q_u| \leq |\tilde{Q}_u| \leq C^{C-1}$ because we consider colorings with C colors of $|\mathcal{N}_u| < C$ vertices in Q. We know that $|\mathcal{N}_u| < C$ because we assume that at most C utterances overlap at a time. The complexity $\mathcal{O}(UC^{C-1})$ is thus overall linear in the number of utterances.

5.2 General remarks on speeding up

A straightforward optimization method for graph coloring is to color each connected component of $\mathcal{G}$ independently. Since there cannot be an edge between two vertices in different connected

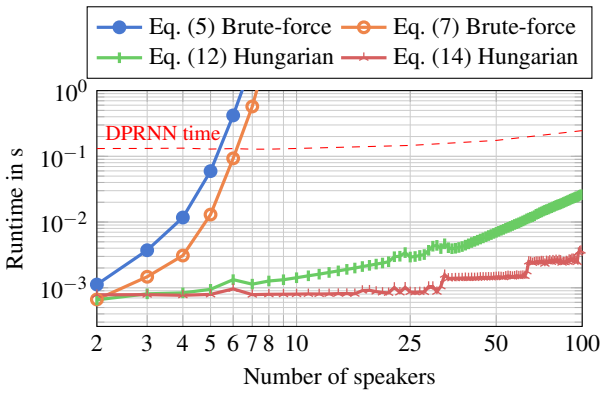

Figure 2: Run-time over number of speakers for different variants for computing the SDR for signals of 4 s length. The run-time required for a forward and backward pass of a corresponding DPRNN network is shown as a dashed red line.

components, the assignment for one of them does not influence the others. The number of utterances in one connected component is typically limited, so that the complexity becomes linear in the number of connected components. It stays unaffected within a connected component, e.g., exponential. This acceleration can be combined with any of the above algorithms. Note that the DP assignment algorithm from Section 5.1.4 does this implicitly.

6 Experiments

We use a Dual-Path Recurrent Neural Network (DPRNN)-TasNet [3] as the separator. Its configuration is kept similar to [3] except for the depth. Our network uses three stacked blocks instead of six, to keep the computational cost for training low [7].

6.1 Separation performance

To validate that $\mathcal{L}^{(\text{sa-SDR})}$ does not impair the performance compared to $\mathcal{L}^{(\text{a-SDR})}$, we compare their performance in terms of SDR [19] for fully overlapped data from the WSJ0-2mix database [20] and Word Error Rate (WER) for artificially generated meeting-like data. We use the same artificially generated meetings based on WSJ [21] as [7]. Each meeting is roughly 120 s long and comprises 5 to 8 speakers, so that the overall speaker distribution is uniform across all meetings. The meetings have an overlap ratio of 0.2 to 0.4 and utterances are selected uniformly. A logarithmic weight between 0 dB to 5 dB is applied to all utterances to simulate volume differences. We use a sample rate of 8 kHz.

We observed a slight improvement in SDR for WSJ0-2mix from 15.7 dB to 15.9 dB when switching from $\mathcal{L}^{(\text{a-SDR})}$ to $\mathcal{L}^{(\text{sa-SDR})}$. The WER improved for the meeting-like data from 13.7 % to 13.4 %. This confirms that the proposed modification of the loss does not harm the performance.

6.2 Run-time evaluation

The run-time evaluation is performed on a GTX1080 graphics card, while the assignment algorithms are executed on the CPU. Graph-PIT algorithms are implemented in Python. The run-times shown here are averaged over 500 measurements.

6.2.1 uPIT

Fig. 2 shows an analysis of the run-time of the different uPIT optimizations for $\mathcal{L}^{(\text{sa-SDR})}$. The time required for a forward and backward step of a separation network as described in Section 6 is marked with a red dashed line. Solving the permutation problem with a brute-force search has a factorial complexity and exceeds the run-time of the network already for 6 speakers, even when most of the heavy-lifting is deferred to computation of a score matrix ("Eq. (7) Brute-force"). Both variants that use the Hungarian algorithm are polynomial in time. Their run-time is dominated by the computation of $\mathbf{M}$. Computing the MSE for $\mathbf{M}$ as in Eq. (12) is slower than the dot product in Eq. (14). A carefully tuned low-level implementation could speed up the MSE to be faster than the

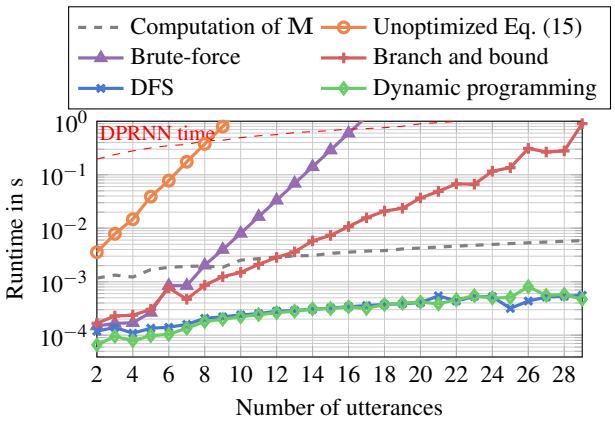

Figure 3: Comparison of the different assignment algorithms for different sizes of connected components on toy example data, computing sa-SDR. All algorithms use a pre-computed score matrix (gray dashed line) except for the "Unoptimized" one. The separator has three outputs. All simulated utterances have a length of 2 s with 0.5 s overlap. The red dashed line indicates the time required for a forward and backward step of the separation network.

dot product because the squared error can be implemented without rather slow floating point multiplications. Since the run-times of both variants are more than one order of magnitude smaller than the time required for the network, such an implementation is not required. The permutation problem can be solved for 100 speakers in a fraction of a second with Eq. (14) and the Hungarian algorithm, taking up a negligible amount of time in practical applications.

6.2.2 Graph-PIT

We compare the run-time of the different assignment algorithms for Graph-PIT in Fig. 3 for a single connected component with varying numbers of utterances and a separator with three output channels. The algorithms compute $\mathcal{L}^{(\text{sa-SDR})}$ since a decomposition for $\mathcal{L}^{(\text{a-SDR})}$ is not possible for Graph-PIT. All algorithms use a pre-computed loss matrix, except for the "Unoptimized" algorithm, which computes the full loss for all assignments (Eq. (15)). The time required for computing $\mathbf{M}$ is drawn with a dashed gray line. Any algorithm that needs more time than the computation of $\mathbf{M}$ dominates the overall run-time.

Both brute-force assignment algorithms ("Unoptimized" and "Brute-force") show an exponential behavior and dominate the run-time for relatively small numbers of utterances. The branch-and-bound algorithm has a better run-time, but is still exponential. The run-times of the DP and DFS algorithms are linear in the number of utterances where DP finds the optimal solution. Their run-time is negligible compared to the computational cost required to compute $\mathbf{M}$. A connected component contains on average 3 utterances in the CHiME-6 corpus [22], with a few outliers contain more than 30 utterances. These would dominate the run-time for all exponential-time algorithms; the "Unoptimized" algorithm, for example, takes more then 40 s for 15 utterances.

A further speedup could be realized by implementation in a lower-level language, such as C. Our implementation of the DP algorithm has a negligible run-time for practical examples.

7 Conclusions

We present a general framework for decomposing loss functions for efficiently finding the optimal assignment of targets to output channels in uPIT or Graph-PIT training of a separation network. We observe a large speedup compared to the naive implementations, so that the run-time of the assignment algorithm becomes negligible even for large numbers of speakers or utterances.

8 Acknowledgements

Computational resources were provided by the Paderborn Center for Parallel Computing.

References

[1] M. Kolbæk, D. Yu, Z.-H. Tan, and J. Jensen, "Multitalker Speech Separation With Utterance-Level Permutation Invariant Training of Deep Recurrent Neural Networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, pp. 1901–1913, Oct. 2017.

[2] Y. Luo and N. Mesgarani, "TaSNet: Time-Domain Audio Separation Network for Real-Time, Single-Channel Speech Separation," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 696–700, Apr. 2018. ISSN: 2379-190X.

[3] Y. Luo, Z. Chen, and T. Yoshioka, "Dual-Path RNN: Efficient Long Sequence Modeling for Time-Domain Single-Channel Speech Separation," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 46–50, May 2020. ISSN: 2379-190X.

[4] Z. Chen, T. Yoshioka, L. Lu, T. Zhou, Z. Meng, Y. Luo, J. Wu, X. Xiao, and J. Li, "Continuous Speech Separation: Dataset and Analysis," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7284–7288, May 2020. ISSN: 2379-190X.

[5] T. Yoshioka, H. Erdogan, Z. Chen, X. Xiao, and F. Alleva, "Recognizing Overlapped Speech in Meetings: A Multichannel Separation Approach Using Neural Networks," in *Interspeech 2018*, pp. 3038–3042, ISCA, Sept. 2018.

[6] Y. Luo and N. Mesgarani, "Conv-TasNet: Surpassing Ideal Time–Frequency Magnitude Masking for Speech Separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, pp. 1256–1266, Aug. 2019. Conference Name: IEEE/ACM Transactions on Audio, Speech, and Language Processing.

[7] T. von Neumann, K. Kinoshita, C. Boeddeker, M. Delcroix, and R. Haeb-Umbach, "Graph-PIT: Generalized permutation invariant training for continuous separation of arbitrary numbers of speakers," in *Interspeech 2021*, ISCA, 2021.

[8] B. Bollobás, "Colouring," in *Graph Theory: An Introductory Course* (B. Bollobás, ed.), Graduate Texts in Mathematics, pp. 88–102, New York, NY: Springer, 1979.

[9] H. W. Kuhn, "The Hungarian Method for the Assignment Problem," in *50 Years of Integer Programming 1958-2008*.

[10] J. Munkres, "Algorithms for the assignment and transportation problems," *J. Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[11] S. Dovrat, E. Nachmani, and L. Wolf, "Many-Speakers Single Channel Speech Separation with Optimal Permutation Training," *arXiv:2104.08955 [cs, eess]*, Apr. 2021. arXiv: 2104.08955.

[12] J. Heitkaemper, D. Jakobeit, C. Boeddeker, L. Drude, and R. Haeb-Umbach, "Demystifying TasNet: A Dissecting Approach," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6359–6363, May 2020. ISSN: 2379-190X.

[13] J. Le Roux, S. Wisdom, H. Erdogan, and J. R. Hershey, "SDR–half-baked or well done?," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 626–630, IEEE, 2019.

[14] T. von Neumann, K. Kinoshita, C. Boeddeker, M. Delcroix, and R. Haeb-Umbach, "Graph-PIT: Enabling Segment-less Continuous Speech Separation of Meetings," *to be submitted to IEEE Transactions on Audio, Speech, and Language Processing*, 2021.

[15] P. Tichavsky and Z. Koldovsky, "Optimal pairing of signal components separated by blind techniques," *IEEE Signal Processing Letters*, vol. 11, pp. 119–122, Feb. 2004. Conference Name: IEEE Signal Processing Letters.

[16] S. Even, *Graph Algorithms*. WH Freeman & Co., 1979.

[17] A. H. Land and A. G. Doig, "An Automatic Method for Solving Discrete Programming Problems," in *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art* (M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, eds.), pp. 105–132, Berlin, Heidelberg: Springer, 2010.

[18] S. Dreyfus, "Richard Bellman on the Birth of Dynamic Programming," *Operations Research*, vol. 50, no. 1, pp. 48–51, 2002. Publisher: INFORMS.

[19] C. Févotte, R. Gribonval, and E. Vincent, "BSS_eval Toolbox User Guide – Revision 2.0," 2005. type:techreport.

[20] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 31–35, IEEE, 2016.

[21] J. Garofolo, D. Graff, D. Paul, and D. Pallett, "Csr-i (wsj0) complete ldc93s6a," *Web Download. Philadelphia: Linguistic Data Consortium*, vol. 83, 1993.

[22] S. Watanabe, M. Mandel, J. Barker, E. Vincent, A. Arora, X. Chang, S. Khudanpur, V. Manohar, D. Povey, and D. Raj, "CHiME-6 Challenge: Tackling multispeaker speech recognition for unsegmented recordings," in *6th International Workshop on Speech Processing in Everyday Environments (CHiME 2020)*, 2020.