A graph-based approach to assist TDOA based localization

Bin Yang and Martin Kreißig
Institute of Signal Processing and System Theory, University of Stuttgart, Germany
Email: {bin.yang, martin.kreissig}@iss.uni-stuttgart.de

***Abstract*—In multidimensional source localization based on time difference of arrival (TDOA), it is difficult to identify the direct-path TDOA among all TDOA estimates of one sensor pair and to find the corresponding TDOA values belonging to the same source among different sensor pairs. In this paper, we present a graph based approach to solve this problem. It relies on the concept of consistent graph whose sum of edge weights along all loops is zero. We introduce the concept of consistent graph and reformulate the above TDOA matching task as a synthesis of consistent graphs. We prove the feasibility of a bottom-up synthesis of consistent graphs by using fundamental loops. Finally, we present an efficient synthesis algorithm by exploiting a search in a compatibility-conflict graph.**

I. TDOA based localization

In multidimensional source localization, one is interested to estimate the position of one or multiple sources in space from the temporal measurements of several sensors which sample the time-space wave field induced by the sources [1]. Typically, K_s sources at the positions $\underline{q}_k \in \mathbb{R}^3$ transmit the source signals $s_k(t), 1 \le k \le K_s$ which induce the multidimensional wave field $x(t, \underline{p})$ at time t and position $\underline{p} \in \mathbb{R}^3$. An array of M sensors at the positions $\underline{p}_m$ sample this wave field spatially and collect M sensor signals $x_m(t) = x(t, \underline{p}_m), 1 \le m \le M$. In an ideal reflection-free environment, the source signal $s_k(t)$ arrives at mth sensor after a delay of $t_{mk,0} = \|\underline{p}_m - \underline{q}_k\|/c$ where c is the speed of propagation. A real environment causes Q_{mk} additional reflections of $s_k(t)$ also arriving at mth sensor . This results in the signal model

$$x_m(t) = \sum_{k=1}^{K_s} \sum_{\mu=0}^{Q_{mk}} a_{mk,\mu} s_k(t - t_{mk,\mu}) + n_m(t), \tag{1}$$

where $t_{mk,\mu}$ and $a_{mk,\mu}$ denote the delay and amplitude loss along μth path from kth source to mth sensor. $n_m(t)$ denotes the sensor noise. The task of localization is to estimate the source positions $\underline{q}_k$ from the measured signals $x_m(t)$ and the known sensor positions $\underline{p}_m$.

If the source signals $s_k(t)$ are known like in the global positioning system (GPS), a cross-correlation between $x_m(t)$ and $s_k(t)$ returns an estimate for the time of arrival (TOA) $t_{mk,0}$. If the source signals $s_k(t)$ are unknown as in acoustic source localization, there is no way to estimate the TOA. Instead, one estimates the time difference of arrival (TDOA) $\tau_{lm,k,00} = t_{lk,0} - t_{mk,0}$ between the direct paths from kth source to lth and mth sensor by a cross-correlation of $x_l(t)$ and $x_m(t)$ [2]. In the literature, the so called generalized cross-correlation with the phase transform GCC-PHAT [3], [4] is widely used for this purpose. The source positions $\underline{q}_k$ are then estimated from the TDOA estimates of different sensor pairs (l, m) and the source positions $\underline{p}_m$ [5], [6]. The two solid line boxes in Fig. 1 show the two major steps of TDOA based source localization: TDOA estimation and position estimation.

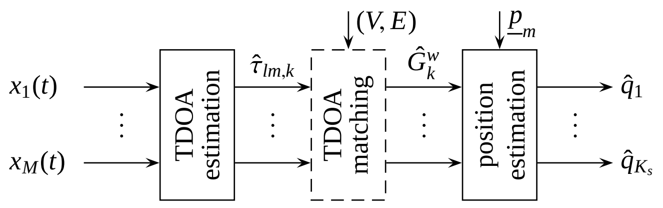

Fig. 1. Major steps of TDOA based localization

One serious problem of this approach is the ambiguity of TDOA estimation. According to Eq. (1), a cross-correlation of $x_l(t)$ and $x_m(t)$ yields a number of TDOA estimates at $t_{lk,\mu} - t_{mk,\nu}$ for one sensor pair (l, m). For the purpose of localization, only the direct path TDOA $\tau_{lm,k,00} = t_{lk,0} - t_{mk,0}$ is of interest and has to be identified among all TDOA estimates. In addition, in the multiple source case, one needs to know which TDOA estimates from different sensor pairs (l, m) belong to the same source. Using a set of TDOA estimates from different sources would lead to a wrong position estimate.

Fig. 2 shows the GCC-PHAT of two microphone signals recorded in a small room with the reverberation time $t_{60} \approx 300$ms where two speakers were talking simultaneously [7]. Ideally, one expects to see two peaks at the position of true TDOA values of the two sources (dashed lines). In practice, the cross-correlation shows many local maxima due to multipath propagation. One may select for each sensor pair the $K(\gg K_s)$ largest maxima resulting in K TDOA estimates $\hat{\tau}_{lm,k}, 1 \le k \le K$; but it is hard to identify the desired direct-path TDOA among them and to find the set of corresponding TDOA estimates from different sensor pairs belonging to the same source. Without this so called TDOA matching (dashed line box in Fig. 1), there will be an exponentially increasing number of possible TDOA combinations though we only need K_s sets of TDOA estimates for position estimation.

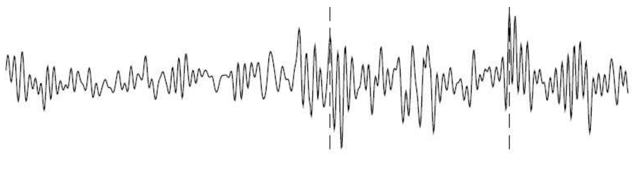

Fig. 2. GCC-PHAT of two microphone signals for two speakers in a room

Up to now, no efficient solutions are known for the TDOA matching. In this paper, we present a graph based approach to solve this problem. It relies on the concept of consistent graph [7], [8]. Each vertex (node) of the graph represents one sensor m and each edge of the graph denotes one sensor pair (l, m). In the ideal case, each edge has one associated edge weight corresponding to the TDOA $\tau_{lm} = t_l - t_m$ of exactly one source for that sensor pair. Since all edge weights (TDOA values) stem from the same source and same direct paths, their cyclic sum along any loop in the graph is zero by definition

$$\tau_{ij} + \tau_{jk} + \ldots + \tau_{li} = (t_i - t_j) + (t_j - t_k) + \ldots + (t_l - t_i) = 0. \quad (2)$$

This is equivalent to the 2. Kirchhoff law for electrical circuits where the sum of voltages along any loop in the circuit is zero. If the TDOA values in Eq. (2) stem from different sources and/or different paths, Eq. (2) does not hold with probability one. This simple observation is the basic idea for TDOA matching. It reduces a vast number of possible combinations of $\hat{\tau}_{lm,k}$ to only a few solutions $\hat{G}^w_k$ satisfying Eq. (2). Note that this idea is not only valid for difference of time arrival, but also for any other difference measurements between two sensors like difference of position, velocity, electrical potential etc. Hence it is widely applicable for a large number of sensor fusion applications [8].

II. Consistent graph

In this section, we review the concept of consistent graph introduced in [7], [8]. It is a mathematical abstraction of the TDOA matching problem.

We consider a directed graph $G = (V, E)$. $V = \{v_1, \ldots, v_M\}$ is the set of M vertices and $E = \{e_1, \ldots, e_N\}$ is the set of N directed edges. Since we need to check the zero sum condition (2) along all loops of the graph, we only consider connected graphs where each pair of vertices is connected by at least one path of the graph. One necessary condition for this is $N \geq M - 1$ [9]. If the graph is complete, i.e. each pair of vertices is connected by an edge, $N = \frac{M(M-1)}{2}$. Fig. 3 shows a complete graph with 4 vertices, 6 edges, and all 7 directed loops $l_1, \ldots, l_7$.

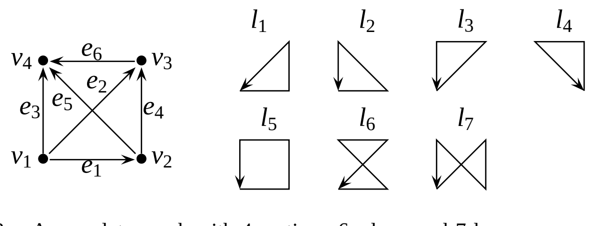

Fig. 3. A complete graph with 4 vertices, 6 edges, and 7 loops

In a weighted graph $G^w = (G, \underline{w}) = (V, E, \underline{w})$, a weight $w_n \in \mathbb{R}$ is assigned to each edge e_n and $\underline{w} = [w_1, \ldots, w_N]^T$. The sum of edge weights along a loop is calculated by taking the loop and edge direction into account. If an edge shows the opposite direction as the loop, its weight is negated before entering into the sum. For example, the sum of edge weights for loop l_1 in Fig. 3 is $w_1 + w_4 - w_2$. The graph G^w is said to be consistent if the sum of edge weights along all loops in G is zero.

A useful matrix representation of all loops in a graph is the loop matrix $\mathbf{B}$. It is given by

$$\mathbf{B} = \left[\begin{array}{c|ccccccc} & l_1 & l_2 & l_3 & l_4 & l_5 & l_6 & l_7 \\ e_1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ e_2 & -1 & 0 & 1 & 0 & 0 & -1 & 1 \\ e_3 & 0 & -1 & -1 & 0 & -1 & 0 & -1 \\ e_4 & 1 & 0 & 0 & 1 & 1 & 0 & -1 \\ e_5 & 0 & 1 & 0 & -1 & 0 & 1 & 1 \\ e_6 & 0 & 0 & 1 & 1 & 1 & -1 & 0 \end{array}\right] \quad (3)$$

for the graph in Fig. 3. It describes the relationship between all loops in columns and all N edges in rows. If an edge is contained in a loop, the corresponding element in $\mathbf{B}$ is marked with 1 if the edge and loop direction coincide or -1 otherwise. A zero element in $\mathbf{B}$ indicates that the edge is not contained in that loop. By using the loop matrix, a consistent graph is easily defined by

$$\mathbf{B}^T \underline{w} = \underline{0}. \quad (4)$$

The inner product of $\underline{w}$ with each column of $\mathbf{B}$ is the sum of edge weights along that loop.

Unfortunately, even a medium-size graph has a huge number of loops. For a complete graph with M vertices, the total number of loops is

$$N_\text{L} = \sum_{m=3}^{M} \binom{M}{m} \frac{(m-1)!}{2}, \quad (5)$$

see Appendix A for a proof. Table I lists the number of edges and loops for a complete graph with M vertices. Obviously, checking the consistency of $\underline{w}$ along all loops is not practical.

$M = \sharp$ vertices	3	4	5	10	20
$N = \sharp$ edges	3	6	10	45	190
$N_\text{L} = \sharp$ loops	1	7	37	556014	$1.7 \cdot 10^{17}$
$N_\text{IL} = \sharp$ ILs	1	3	6	36	171

TABLE I. Number of edges, loops, and independent loops for a complete graph with M vertices

Fortunately, all loops (columns) of the loop matrix $\mathbf{B}$ can be written as linear combinations of a small set of linearly independent loops (IL) of $\mathbf{B}$. According to [10], the number of linearly independent loops or the rank of $\mathbf{B}$ for a connected graph with M vertices and N edges is $N - M + 1$. This means, we can always find a subset of $N_\text{IL} = N - M + 1$ columns in $\mathbf{B}$ which span the complete column space of $\mathbf{B}$. Let $\mathbf{B}_\text{IL}$ be an $N \times (N - M + 1)$ submatrix of $\mathbf{B}$ containing a set of ILs. A necessary and sufficient condition for $\underline{w}$ being consistent is then

$$\mathbf{B}_\text{IL}^T \underline{w} = \underline{0}. \quad (6)$$

If the edge weights $\underline{w}$ (TDOA estimates) suffer from estimation errors or noise, the sum of edge weights along a loop is not exactly zero. In this case, the condition (6) is replaced by $\|\mathbf{B}_\text{IL}^T \underline{w}\| \leq \delta$ where $\|\cdot\|$ is a suitable vector norm and $\delta > 0$ is a small threshold value. Since $N_\text{IL} \ll N_\text{L}$, see Table I, the computational complexity of (6) is significantly reduced in comparison to (4). For the graph in Fig. 3, the rank of $\mathbf{B}$ is 3.

Note that for a given loop matrix $\mathbf{B}$, there are many different possibilities to choose a set of ILs. A special case of ILs is a set of fundamental loops (FL). While the ILs are defined as linearly independent columns of the loop matrix $\mathbf{B}$, the FLs are typically defined in a geometrical way by using the concept of spanning tree and cotree [9]. A spanning tree

$G_s = (V_s, E_s)$ of a given connected graph $G = (V, E)$ is a subgraph with $V_s = V$ and $E_s \subset E$. It has $M - 1$ edges connecting all vertices in G without closing any loop. The subgraph of G containing the remaining $N - M + 1$ edges is called the cotree $G_c = (V_c, E_c)$ of G with $E_c = E \backslash E_s$. By adding each edge of the cotree to the spanning tree, one FL is closed. Since the cotree has $N - M + 1$ edges, a connected graph with M vertices and N edges has exactly $N - M + 1$ FLs. The number of FLs is identical to the rank of $\mathbf{B}$ and the number of ILs.

Fig. 4 shows the spanning tree (solid line), cotree (dashed line) and the corresponding FLs for the graph in Fig. 3. Note that there are different choices for the spanning tree, resulting in different cotrees and different sets of FLs. In Fig. 4, the breadth-first-search (BFS) spanning tree [9] is shown top and the depth-first-search (DFS) spanning tree is shown bottom. The corresponding set of FLs is $\{l_1, l_2, l_3\}$ for BFS and $\{l_1, l_4, l_5\}$ for DFS.

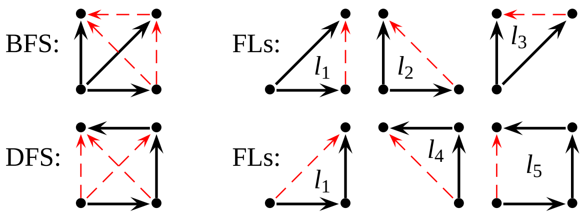

Fig. 4. Spanning tree (solid line), cotree (dashed line) and fundamental loops for the graph in Fig. 3

It is well known that each set of FLs is also a set of ILs [10]. This means, each set of FLs can also be used in Eq. (6) to check the consistency of a given edge weight vector $\underline{w}$. The opposite, however, is not true. A set of ILs is not always a set of FLs. The reason is that the FLs have some special geometrical properties not shared by all ILs. By construction, each FL has exactly one edge (from the cotree) not shared by the other FLs. The 3 loops $\{l_1, l_6, l_7\}$ in Eq. (3) and Fig. 3, for example, are linearly independent and thus form a set of ILs. But they are not a set of FLs because each edge of the loop l_1 also occurs in l_6 and/or l_7. This means, we cannot find a spanning tree and cotree resulting in the set of FLs l_1, l_6, l_7. In the next section, we will see that this difference between FLs and ILs will play a key role for the bottom-up synthesis of consistent graphs.

III. Bottom-up synthesis of consistent graphs

The previous section showed an efficient way to examine the consistency of a given edge weight vector $\underline{w}$ with respect to a graph $G = (V, E)$. $\underline{w}$ contains exactly one weight w_n for each edge e_n. But this is not the problem we want to solve. In TDOA based localization, not a single TDOA value or weight w_n, but a large set of K_n edge weights $W_n = \{w_{n,1}, \ldots, w_{n,K_n}\}$ is generated for each edge e_n. They correspond to the TDOA estimates or positions of peaks in the cross-correlation of two sensor signals, see section I. The number K_n of selected peaks is usually chosen large enough to ensure that W_n contains the K_s desired direct-path TDOA values of the K_s sources. The remaining TDOA values in W_n originate from multipath propagation and should be discarded.

Given the sets $W_1, \ldots, W_N$ of edge weights, the TDOA matching problem is to find matching combinations of edge weights $\underline{w} \in W_1 \times \ldots \times W_N$ in the sense of (6). This is a combinatorial problem. We call it the synthesis of consistent graphs because, starting with $W_1, \ldots, W_N$, a number of consistent graphs $\hat{G}_k^w = (V, E, \underline{w}_k)$ are synthesized where each $\underline{w}_k$ satisfies the consistency condition (6). Each consistent graph $\hat{G}_k^w$ is the input to estimate the position of one source.

A brute force approach for the synthesis of consistent graphs is to examine all $\prod_{n=1}^{N} K_n$ (or K^N if $K_n = K$) weight combinations for $\underline{w}$. The number of possible combinations increases exponentially with N, making this approach intractable. For a small array of $M = 5$ sensors and $N = 10$ sensor pairs, if one chooses the $K_n = K = 10$ largest maxima of each cross-correlation, $K^N = 10^{10}$ possible weight combinations have to be examined in order to localize a few sources.

To avoid this huge complexity, we propose to use a bottom-up synthesis approach. Starting with some small-size connected subgraphs of G, we first look for consistent weight combinations for each subgraph by checking all possible weight combinations. The complexity of this exhaustive search is small because each subgraph has a small number of edges. Then the consistent subgraphs are successively combined together to larger consistent graphs. Here two important questions have to be answered: a) What kind of subgraphs should we choose to start with? They should be easy to find for any given graph G, should have a small number of edges, and should enable a bottom-up synthesis of consistent graphs. b) How can we ensure that a combination of consistent subgraphs automatically leads to a larger consistent graph? The answer for both questions is the use of FLs.

Let $G_i^w = (V_i, E_i, \underline{w}_i)$ $(i = 1, 2)$ be two weighted subgraphs. They are said to be compatible if they have at least one common edge and all common edges of G_1^w and G_2^w have identical edge weights in $\underline{w}_1$ and $\underline{w}_2$. A combination of two compatible weighted subgraphs results in a larger weighted graph $G_{12}^w = (V_{12}, E_{12}, \underline{w}_{12})$. It is defined mathematically by union of sets: $V_{12} = V_1 \cup V_2, E_{12} = E_1 \cup E_2$ and $\underline{w}_{12} = \underline{w}_1 \cup \underline{w}_2$, i.e. the union graph G_{12}^w contains all vertices, edges and edge weights from G_1^w and G_2^w. If G_1^w and G_2^w do not have any common edges or at least one of their common edges has different weights in $\underline{w}_1$ and $\underline{w}_2$, they cannot not be combined.

Fig. 5 shows two examples for the union of compatible graphs. Clearly, a union of graphs introduces new loops not present previously, e.g. those in dashed line. Thus the central question is whether the union graph G_{12}^w is consistent if both G_1^w and G_2^w are consistent.

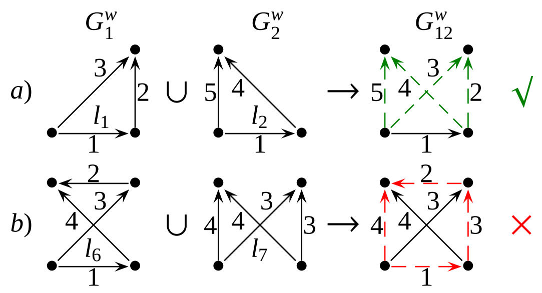

Fig. 5. a) a union of two consistent FLs is again consistent; b) a union of two consistent ILs is not always consistent

Fig. 5a shows the union of two loops l_1 and l_2 which are members of the set of FLs $\{l_1, l_2, l_3\}$ in Fig. 4. They are consistent and compatible because the common edge of l_1 and l_2 share the same weight 1. It is easy to see that the union graph G_{12}^w including the new dashed loop is consistent again. Fig. 5b shows the union of two other loops l_6 and l_7. They are ILs, but not FLs as explained at the end of the previous section. They are consistent and compatible because the two common diagonal edges of l_6 and l_7 have the same weights 3 and 4. Interestingly, the new dashed loop in the union graph G_{12}^w is not consistent.

This example illustrates one important difference between ILs and FLs. A union of compatible and consistent FLs seems to result automatically in a larger consistent graph. The following theorem states that this is indeed generally true. By combining all compatible and consistent FLs of a given graph, a largest possible consistent graph can be synthesized. This bottom-up synthesis is not possible by combining compatible and consistent ILs.

Theorem 1: Given a connected graph G and a set of FLs $L = \{l_1, \cdots, l_{N_{\text{IL}}}\}$. Let L_i $(i = 1, 2)$ be two disjoint subsets of L and $G_i = \cup_{l \in L_i} l$ be the union of FLs from L_i. If G_1 and G_2 are connected and have at least one common edge, then a) $G_1 \cup G_2$ is connected as well; b) all common edges in G_1 and G_2 are connected; c) $L_1 \cup L_2$ forms a set of FLs for $G_1 \cup G_2$; d) If G_1^w and G_2^w are consistent and compatible, then $G_1^w \cup G_2^w$ is also consistent.

Proof: see Appendix A.

IV. Compatibility-conflict graph

The previous section proved a theorem which enables a bottom-up synthesis of consistent graphs by combining compatible and consistent FLs. But the theorem does not tell us how to do this combination. This section presents an algorithm for this purpose. The basic idea of the algorithm is to represent the relationship between all found consistent FLs in terms of a new graphical representation, a called compatibility-conflict graph or simply cc-graph.

Conflict graphs are well known in the graph theory [11], [12], [13]. They are also known under the name independent set or stable set [14]. Two vertices in conflict are connected by an edge and vertices without conflict are disconnected. Our cc-graph is an extension of conflict graph and introduces a third new state between two vertices.

For illustration, Fig. 6a shows an incomplete graph G with $M = 6$ vertices and $N = 9$ edges. Each vertex represents a sensor and each edge represents a sensor pair. The spanning tree of G is drawn in solid line and the cotree in dashed line. By appending one edge of the cotree to the spanning tree, one FL is closed. In total, G has $N_{\text{IL}} = N - M + 1 = 4$ FLs $l_1, \ldots, l_4$, see Fig. 6b. Obviously, each pair of FLs share at least one common edge except for (l_2, l_3).

Assume that $\tilde{K}_i$ consistent weight combinations or consistent FLs $\hat{l}_{i,k}^w$ have been found for the i-th FL l_i $(1 \le k \le \tilde{K}_i, 1 \le i \le N_{\text{IL}})$. For the 4 FLs in Fig. 6b, we assume $\tilde{K}_1 = \tilde{K}_2 = \tilde{K}_3 = 2$ and $\tilde{K}_4 = 1$. Each consistent FL $\hat{l}_{i,k}^w$ represents a vertex in the cc-graph $G_{cc} = (V_{cc}, E_{cc}, \bar{E}_{cc})$ and there is a total number of $\sum_{i=1}^{N_{\text{IL}}} \tilde{K}_i = 7$ vertices, see Fig. 7.

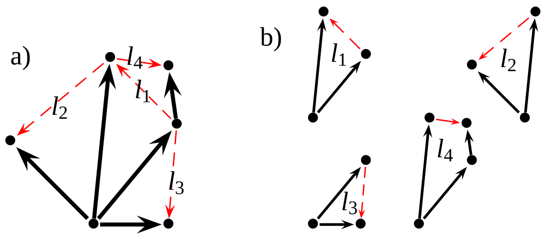

Fig. 6. a) A graph with 6 vertices and 9 edges; b) 4 FLs

Each pair of vertices in G_{cc} can take exactly one of three possible states: a) compatibility state: Both consistent FLs associated to the pair of vertices have at least one common edge and all common edges have identical edge weights. These two compatible vertices are connected by a solid line edge, a compatibility edge, and can be combined together, e.g. $(\hat{l}_{1,1}^w, \hat{l}_{2,2}^w)$. The set E_{cc} contains all compatibility edges. b) conflict state: Both consistent FLs have at least one common edge with different edge weights. They are connected by a dashed line edge, a conflict edge, and are not allowed for combination at all, e.g. $(\hat{l}_{1,1}^w, \hat{l}_{2,1}^w)$. The set $\bar{E}_{cc}$ contains all conflict edges. c) open state: Both consistent FLs have no common edges. They are not connected in the cc-graph and have an open relationship, e.g. $(\hat{l}_{2,2}^w, \hat{l}_{3,1}^w)$ or $(\hat{l}_{2,2}^w, \hat{l}_{3,2}^w)$. They can be combined later over other compatible consistent FLs (e.g. $\hat{l}_{2,2}^w - \hat{l}_{1,1}^w - \hat{l}_{3,2}^w$) or not (e.g. $\hat{l}_{2,2}^w, \hat{l}_{3,1}^w$).

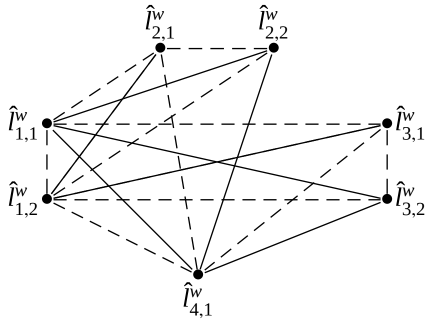

Fig. 7. The cc-graph G_{cc} for the consistent FLs in Fig. 6

The cc-graph is a comprehensive graphical description about which consistent FLs are compatible and combinable, which ones are in conflict and not combinable, and which ones have disjoint edges. The task now is to find maximally connected sets of vertices in G_{cc} which are connected by compatibility edges only. Each such solution set is a union of compatible and consistent FLs and is, according to Theorem 1, a larger consistent graph $\hat{G}_k^w$. It satisfies the consistency condition (6) and is the input for the source position estimation, see Fig. 1. In Fig. 7, there are two such solution sets $\hat{G}_1^w = \{\hat{l}_{1,1}^w, \hat{l}_{2,2}^w, \hat{l}_{3,2}^w, \hat{l}_{4,1}^w\}$ and $\hat{G}_2^w = \{\hat{l}_{1,2}^w, \hat{l}_{2,1}^w, \hat{l}_{3,1}^w\}$. The first solution set $\hat{G}_1^w$ results in a fully consistent graph containing all 9 edges of the given graph in Fig. 6a. The second solution set $\hat{G}_2^w$ leads to a partially consistent graph containing only 8 edges of the graph in Fig. 6a because the top cotree edge of FL l_4 is missing.

To find these solution sets from the cc-graph is again a non-trivial combinatorial problem. The desired algorithm has to fulfill the following requirements:

- The solution set $\hat{G}_k^w$ should be maximally connected, i.e. as large as possible. In the ideal case, a solution set leads to a fully consistent graph containing all given edges (sensor pairs) of G. Frequently, a partially consistent graph is found which contains only a subset of the edges of G. Such a partially consistent graph is also useful for source localization.

- All possible solution sets should be found.
- No redundant solution sets which are part of other solution sets should be found.

In [15], such an algorithm was presented which is an extension of the conflict graph algorithm in [16] to a cc-graph. It is shown as cc-graph search (CCGsearch) in Algorithm 1. It is a recursive algorithm CCGsearch(G_{cc}, V, L, X) operating on a given cc-graph G_{cc}. During any instance of the recursion, V is the set of currently considered vertices from G_{cc} (not the set of vertices of the consistent graph in section II), L is the current solution set of found compatible vertices, and X is the set of of vertices already visited which should be skipped now. The algorithm is initiated with CCGsearch($G_{cc}, V_{cc}, \emptyset, \emptyset$) where V_{cc} is the set of all vertices in G_{cc}. The term $N_V(L)$ in CCGsearch denotes the set of vertices from V which are compatible to at least one vertex from the solution set L and have no conflict to any vertices from L. Similarly, $\bar{N}_V(L)$ denotes the set of vertices from V which have conflict to at least one vertex in L. For more details, we refer to [15].

Algorithm 1 CCGsearch algorithm

```
 1: CCGsearch($G_{cc}, V, L, X$)
 2: determine $N_V(L)$ and $\bar{N}_V(L)$
 3: if $N_V(L) = \emptyset$ then
 4:     save $L$                      % save solution $L$
 5: else
 6:     $V = V \setminus \bar{N}_V(L)$        % remove conflict neighbours
 7:     for $n \in N_V(L) \setminus X$ do
 8:         CCGsearch($G_{cc}, V \setminus n, L \cup n, X$)
 9:         $X = X \cup n$            % mark vertex $n$ as visited
10:     end for
11: end if
```

Fig. 8 illustrates the steps of synthesis of consistent graphs for the TDOA matching in Fig. 1. Starting with a given connected graph $G = (V, E)$ with M vertices and N edges, a suitable set of $N_{\mathrm{IL}} = N - M + 1$ FLs l_i is chosen. For each FL $l_i, 1 \leq i \leq N_{\mathrm{IL}}$, the TDOA estimates $\hat{\tau}_{lm,k}$ from the cross-correlations of different sensor pairs (l, m) are examined to look for consistent weight combinations or consistent FLs $\hat{l}^w_{i,k}, 1 \leq i \leq N_{\mathrm{IL}}, 1 \leq k \leq \tilde{K}_i$. All $\sum_{i=1}^{N_{\mathrm{IL}}} \tilde{K}_i$ consistent FLs form the vertices of a cc-graph G_{cc} whose three types of edges (compatibility, conflict, open) define the relationship between all consistent FLs. Finally, the CCGsearch algorithm is applied to find all maximally connected sets of compatible and consistent FLs. They are the desired consistent graphs $\hat{G}^w_k$. Each of them is the input to localize one source. The complete MATLAB code for synthesis of consistent graphs can be downloaded from www.iss.uni-stuttgart.de/download/.

Extensive experiments have shown that this graph based approach for TDOA matching has a low computational complexity and significantly reduces the number of possible TDOA

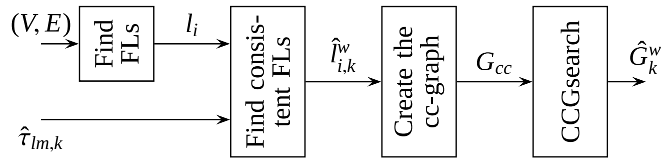

Fig. 8. Steps of synthesis of consistent graphs

combinations and wrong position estimates. They are, however, not reported in this paper due to limited space.

V. Conclusions

In this paper, we presented a graph based approach to find matching TDOA estimates from different sensor pairs to assist TDOA based source localization. We reviewed the concept of consistent graphs and fundamental loops. We gave a proof for the bottom-up synthesis of consistent graphs and presented an efficient synthesis algorithm.

Appendix

Proof of Eq. (5)

We consider a loop containing $m \geq 3$ vertices. There are $\binom{M}{m}$ different sets of m from M vertices. For each set of m vertices, say $\{v_1, \ldots, v_m\}$, there are $m!$ possible permutations. But many of these permutations result in the same loop: a) m cyclic permutations $v_1 v_2 \ldots v_m$; $v_2 \ldots v_m v_1$; $\ldots$; $v_m v_1 \ldots v_{m-1}$ result in the same loop; b) 2 reversed permutations $v_1 v_2 \ldots v_m$ and $v_m v_{m-1} \ldots v_1$ also lead to the same loop. Hence there are $\frac{m!}{2m} = \frac{(m-1)!}{2}$ different loops for each set of m vertices. The total number of loops is given in Eq. (5).

Proof of Theorem 1

The fact a) is trivial. Since G_1 and G_2 are connected and they share at least on common edge, $G_1 \cup G_2$ is connected, too.

The proof of b) is done by a contradiction. Fig. 9 shows two common edges e_1 and e_2 (solid line) in two graphs G_1 and G_2. We assume that e_1 and e_2 are not connected. Since G_1 and G_2 are unions of two subsets of the same FLs, e_1 and e_2 must originate from the spanning tree of G. Since the spanning tree is connected by definition, there must be a path p as a part of the spanning tree connecting e_1 and e_2. If G_1 and G_2 share the same path p, e_1 and e_2 are connected by p which contradicts the above assumption. If G_2 has a different path q from the spanning tree connecting e_1 and e_2, then there is a loop $p - q$ in the spanning tree which contradicts the definition of the spanning tree. Thus e_1 and e_2 must be connected.

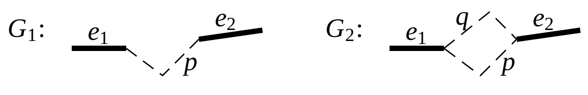

Fig. 9. Two common edges e_1, e_2 in two graphs G_1, G_2

The fact c) follows by counting the number of vertices, edges, and FLs in G_1, G_2, and $G_1 \cup G_2$. Assume that G_i has M_i vertices, N_i edges and $N_i - M_i + 1$ FLs ($i = 1, 2$). We also assume that G_1 and G_2 have c common edges. These c common edges are connected according to b), but do not close any loops because they are part of the spanning tree of G.

Hence c common edges imply $c+1$ common vertices in G_1 and G_2. The union graph $G_1 \cup G_2$ has thus $M_{12} = M_1 + M_2 - (c+1)$ vertices, $N_{12} = N_1 + N_2 - c$ edges, and $N_{12} - M_{12} + 1 = (N_1 - M_1 + 1) + (N_2 - M_2 + 1))$ FLs. The number of FLs in $G_1 \cup G_2$ is exactly the sum of the numbers of FLs in G_1 and G_2. In other words, the FLs in G_1 and G_2 also form a set of FLs for $G_1 \cup G_2$.

The fact d) immediately follows from c).

References

[1] F. Gustafsson and F. Gunnarsson, "Mobile positioning using wireless networks," *IEEE Signal Processing Magazine*, vol. 22, pp. 41–53, 2005.

[2] M. Brandstein and D. Ward, Eds., *Microphone Arrays*. Springer, 2001.

[3] W. R. Hahn and S. A. Tretter, "Optimum processing for delay-vector estimation in passive signal arrays," *IEEE Trans. Information Theory*, vol. 19, pp. 608–614, 1973.

[4] C. H. Knapp and G. C. Carter, "The generalized correlation method for estimation of time delay," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 24, pp. 320–327, 1976.

[5] R. O. Schmidt, "A new approach to geometry of range difference location," *IEEE Trans. Aerospace and Electronic Systems*, vol. 8, pp. 821–835, 1972.

[6] J. O. Smith and J. S. Abel, "Closed-form least-squares source location estimation from range-difference measurements," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 35, pp. 1661–1669, 1987.

[7] J. Scheuing and B. Yang, "Disambiguation of TDOA estimation for multiple sources in reverberant environments," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 16, pp. 1479–1489, 2008.

[8] B. Yang and M. Kreißig, "An introduction to consistent graphs and their signal processing applications," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, 2011, pp. 2740–2743.

[9] J. Bang-Jensen and G. Gutin, *Digraphs: Theory, algorithms and applications*, 2nd ed. Springer, 2002.

[10] N. Balabanian and T. A. Bickart, *Electrical network theory*. John Wiley & Sons, 1969.

[11] U. Pferschy and J. Schauer, "The Knapsack problem with conflict graphs," *Journal of Graph Algorithms and Applications*, vol. 13, pp. 233–249, 2009.

[12] V. Bafna and V. Bansal, "The number of recombination events in a sample history: Conflict graph and lower bounds," *IEEE Trans. Computational Biology and Bioinformatics*, vol. 1, pp. 78–90, 2004.

[13] F. Eisenbrand, S. Funke, and J. Reichel, "Packing a trunk," in *European Symposium on Algorithms*, 2003.

[14] S. Rebennack, "Stable set problem: Branch & cut algorithms," in *Encyclopedia of Optimization*, C. A. Floudas and P. Pardalos, Eds. Springer, 2008, pp. 3676–3688.

[15] M. Kreißig and B. Yang, "Reliable simultaneous TDOA assignment in multi-speaker and reverberant environments," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, 2013.

[16] C. Bron and J. Kerbosch, "Algorithm 457: Finding all cliques of an undirected graph," *Communications of the ACM*, vol. 16, pp. 575–577, 1973.