Schlupkothen *et al. EURASIP Journal on Advances in Signal Processing* (2018) 2018:20
https://doi.org/10.1186/s13634-018-0536-x

EURASIP Journal on Advances in Signal Processing

RESEARCH Open Access

Backtracking-based dynamic programming for resolving transmit ambiguities in WSN localization

Stephan Schlupkothen* , Bastian Prasse† and Gerd Ascheid†

Abstract

The complexity of agent localization increases significantly when unique identification of the agents is not possible. Corresponding application cases include multiple-source localization, in which the agents do not have identification sequences at all, and scenarios in which it is infeasible to send sufficiently long identification sequences, e.g., for highly resource-limited agents. The complexity increase is due to the need to solve an additional optimization problem to resolve the indifferentiability of the agents and thus to enable their localization. In this work, we present a thorough analysis of this problem and propose a maximum a posteriori (MAP)-optimal algorithm based on graph decompositions and expression trees. The proposed algorithm efficiently exploits the fixed-parameter tractability of the underlying graph-theoretical problem and employs dynamic programming and backtracking. We show that the proposed algorithm is able to reduce the run time by up to 88.3% compared with a corresponding MAP-optimal integer linear programming formulation.

Keywords: Wireless sensor networks, Localization, Transmit ambiguities

AMS Subject Classification: 90C35, 90C39

1 Introduction

The unique identification of wireless sensors or agents is generally regarded as an axiomatic design criterion for wireless systems. However, in certain application cases, unique identification is either inherently impossible, such as in *multiple-source localization*, or infeasible because of application-specific constraints. In many scenarios, agent localization relies on pairwise inter-sensor distance measurements, for which precise information about the identification of the agents and the obtained measurements is required when using classical localization algorithms. In cases in which unique identification is not possible, the localization of wireless agents requires solving an additional optimization problem to overcome the non-unique identification of the agents. In this work, the corresponding problem is thoroughly analyzed and an efficient and optimal algorithm is presented. Subsequently, a relevant use case for these algorithms is presented.

*Correspondence: schlupkothen@ice.rwth-aachen.de
†Equal contributors
Integrated Signal Processing Systems, RWTH Aachen University, Templergraben 55, Aachen, Germany

An upcoming application case for wireless sensors in which non-unique identification will play a key role is the exploration of environments that are accessible only by miniature sensors[1]. Such environments include groundwater systems and, as considered in [1], hardly permeable *cold heavy oil production with sand* (CHOPS) mining areas. In both cases, the idea is to utilize wireless sensors to explore the environment and gather information regarding, e.g., the structure of the underground system, the physical properties of the sensor-carrying fluid, and the general resource richness. For these scenarios, the sensor motes need to satisfy the following requirements, which are mainly derived from or mentioned in [1]:

- An outer diameter of less than 5 mm to ensure that a significant number of the sensors remain intact after passing through the injection pumps.
- A robust shell able to withstand pressures of up to 6 MPa [2].
- Several tens of thousands of sensor motes to survey a significant portion of the environment while accounting for the fact that many sensors will be

destroyed and remain unrecovered in the environment. In this regard, [1] report an extraction ratio of approximately 8%. The large required number of agents is also motivated by the limited sensing range of each agent.
- Energy supply for an operating time of approximately 48 h.

To facilitate the localization of the agents, they are equipped with ultrasonic transceivers and perform ranging by measuring the round-trip times to other agents. These measurements are only stored locally and are not forwarded to nearby agents. The positioning, i.e., localization, of the sensor motes is performed offline after the extraction of the agents by a central unit, which can utilize significant computational resources for this purpose. This unit has access to all measurements recorded by the recovered agents.

In this scenario, the following conditions lead to the conclusion that unique identification of the agents is not feasible:

1. The small sensor size and the long operation time significantly constrain the consumable energy, particularly because the majority of the available space inside the agents is already occupied by hardware such as transceivers and environmental sensors.
2. Because of the small sensor mote size and its implications, Direct-Sequence (Spread Spectrum) Code-Division Multiple Access (DS-CDMA) approaches are needed because, e.g., frequency-division multiple access (FDMA) and TDMA require both a wide bandwidth and sharp filters[2] and accurate synchronization, respectively, and thus cannot be used. For DS-CDMA, it is known that the code length is equal to the number of users in the network[3]. Thus, the use of unique codes would result in excessive power consumption for the transmission of each ranging pulse.

Consequently, only shorter—meaning non-unique—DS-CDMA codes can provide a reduction in the energy requirements that is sufficient to enable this application scenario[4]. Therefore, the central unit (fusion center) needs to resolve the resulting ambiguities in the pairwise distance measurements, before positioning using classical localization algorithms will be possible.

It is believed that a computationally intensive tracking of the agents can be replaced by several quasi-static localizations whenever the swarm is distributed entirely within the environment. This is because on the one hand many agents are needed for the exploration and because on the other hand it may take long for the agents to pass through the environment. Consequently, a further important task is to estimate—based on the agent measurements—when these localizations should take place.

Note that the underlying problem of insufficient time or frequency resources to uniquely address all agents can likewise occur in other multiple-access schemes. Additionally, note that there is no negative impact regarding, e.g., the signal processing, hardware resources, or energy consumptions of the agents as a result of the non-unique identification. This is because the increased localization complexity that results from non-unique identification and the resolution of the resulting ambiguities are borne by the fusion center.

In this work, the implications of non-unique identification for the localization of sensor motes are investigated. In this regard, we show that non-unique identification results in an additional optimization problem beyond the actual localization itself. We refer to this additional problem as the *transmit-ambiguity resolution problem* (TARP). To address this problem, we propose an algorithm that is able to resolve the ambiguities optimally in the sense of the maximum a posteriori (MAP) probability and that is up to 8.55 times faster than an equivalent integer linear programming (ILP) formulation, which is $\mathcal{NP}$-hard in general and which serves as a baseline in this work. This corresponds to a run-time reduction of 88.5%. The algorithms presented in this work are designed to be used in conjunction with classical localization algorithms, such as those based on semidefinite programming [3–9], second-order cone programming (SOCP) [10], or least-squares methods [11, 12], for the described scenarios. Consequently, it is said that the presented algorithms *re-enable* the localization with existing localization algorithms.

1.1 Related work

Motivated by an initial feasibility study regarding the exploration of inaccessible environments [1] and related experiments in which the need for shared transmission identifiers has been foreseen, the authors of [13] presented the first procedure aimed at resolving the transmit ambiguities caused by non-unique identification for FDMA-based scenarios. The presented method was heuristic and was not derived from an optimality measure such as minimum mean-squared error (MMSE) or maximum-likelihood (ML). Moreover, its complexity was not analyzed, and measurement noise was not explicitly considered. The proposed method relied on a neighbor-matching procedure in which the ambiguities were resolved by comparing the distance measurements of neighboring sensors.

In [14], we extended this approach to purely additive noise scenarios using adjacency and neighborhood estimation. In [15], we considered a joint localization and transmit-ambiguity resolution approach in which a heuristic was used to resolve the ambiguities.

Moreover, in our conference work [16], we introduced the concept of dynamic programming on k-ambiguity trees by exploiting fixed-parameter tractability to efficiently solve the TARP. The run time of this method was compared with that of the integer programming implementation of the MMSE-optimal algorithm.

1.2 Contributions

The contribution of this work is fourfold: First, we present a detailed and comprehensive mathematical formulation of the TARP, which arises in the case of the non-unique identification of sensor motes, including the governing equations and its optimal ILP-based optimization problem. Second, we propose to use dynamic programming instead of solving the $\mathcal{NP}$-hard ILP problem to re-enable the localization of the agents. Third, we describe a new representation of the problem using k-ambiguity trees, which are derived from k-expression trees and are used in a dynamic programming manner to solve the problem optimally. This tree representation is based on a fixed-parameter tractability analysis, which is the basis for the efficient solution of the TARP. Fourth, we improve the dynamic programming approach by employing backtracking, which yields a further reduction in the computational complexity by incrementally evaluating partial solution candidates.

Consequently, we propose a methodology that compensates for the inability of these classical methods to address non-relatable distance measurements while maintaining their general applicability for positioning, cf. Fig. 1.

This work is partly based on our previous conference work [16], which is extended in this work in terms of the following: (1) detailed derivations of the presented algorithms, (2) an extensive evaluation and analysis of the algorithms, and (3) an algorithmic extension via backtracking, resulting in a new dynamic programming methodology.

Moreover, the framework developed in this work can also be extended, for example, to the multiple-source localization problem, which can be regarded as a special case of the problem at hand in which all sources have the same transmit ID.

1.3 Solution overview

This section is dedicated to a less formal description of the problem at hand and the solution steps described in this work. The sensory-agent-based approach for the exploration of unknown environments is depicted in Fig. 1. This figure shows the general procedure, from the introduction of the agents up through their (partial) extraction and the utilization of the agents' measurements for environmental data reconstruction.

The fusion center, which plays a key role in achieving the goal of obtaining precise information about the underground environment, must address the non-unique identification of the agents and their consequently non-

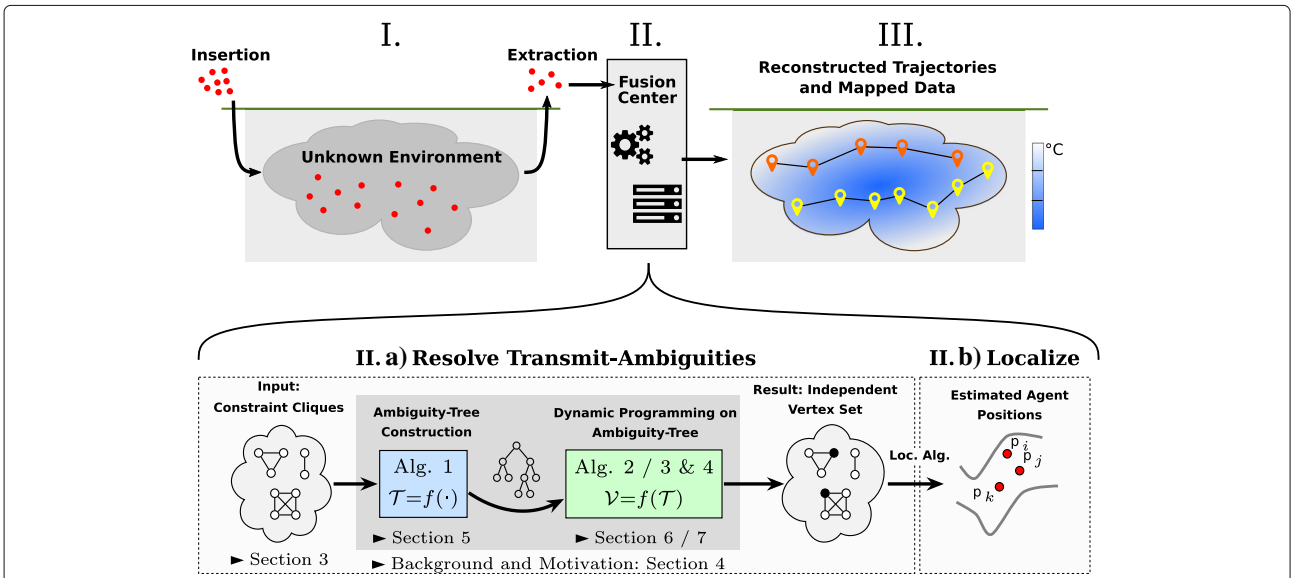


Fig. 1 Illustration of the exploration procedure. In step I, agents (visualized in red) are introduced into an unknown environment and perform environmental and distance measurements for mapping and localization, respectively. The agents are carried by the fluid in the environment. Then, the agents are extracted from the environment, and their data are used by a fusion center (step II) to compute the agents' trajectories (illustrated for two agents). The steps executed by the fusion center for this purpose are as follows: (a) resolve the transmit ambiguities, e.g., with the presented algorithms, and (b) perform positioning using classical localization algorithms. In step III, the environmental measurements and estimated trajectories are used to compute, e.g., the temperature profile of the environment

relatable distance measurements. In this regard, the fusion center follows a two-step approach, where the transmit ambiguities are first resolved and then localization is conducted:

1. The resolution of the transmit ambiguities: Conceptually, the purpose of this step is to estimate the proper mapping between the measurements and the agents. In this work, however, a mapping between pairs of measurements and pairs of agents is sought. Such a pair of measurements is called an *assignment* or *solution candidate* for the two bidirectional measurements between the corresponding pair of agents. Because of the combinatorial nature of this problem, it is $\mathcal{NP}$-hard in general. As shown in Section 3, this problem is related to the *independent set problem* and can be described as an *ILP problem*. However, Section 4 motivates the use of dynamic programming to solve this problem more efficiently based on the fact that the problem is *fixed-parameter tractable*. In this context, it is shown that the *clique-width* of the graph that describes the resolution problem from a graph-theoretical perspective exerts a significant influence on the solution complexity. For this reason, the following procedure is used in this work to solve the problem:
   (a) Based on a thorough analysis of the graph-theoretical problem presented in Section 4, the constraints that govern the combinatorial problem are derived. These constraints are described by vertex sets specifying mutual exclusivity, called *constraint cliques* in the following.
   (b) These constraint cliques, which are the basic entities of the graph-theoretical problem, are used to obtain a tree structure that serves as the basis for computing the complexity-determining *clique-width* parameter. This computation is performed such that, with a reasonable computational complexity, a complexity parameter k close to the actual *clique-width*—which is the lowest possible k—is achieved. The corresponding theory and algorithm are introduced in Sections 4 and 5.
   (c) Utilizing the previously computed tree structure, dynamic programming methods are used to obtain a solution. The solutions correspond to *selected* vertices of the vertex sets that specify the mutual exclusivity of the constraint cliques. Consequently, a solution to the combinatorial problem is found, and a mapping between the measurements and the agents is obtained. Two different dynamic programming algorithms are presented in Sections 6 and 7, of which the latter uses a backtracking procedure.
2. The localization of the agents: Based on the mapping between the measurements and the agents, classical localization algorithms can be used for positioning. See, e.g., the last paragraph of Section 1 for an overview of possible solution methods.

1.4 Organization

Based on the steps outlined in Fig. 1, the remainder of this work is structured as follows. In Section 2, the system model and problem formulation are presented. Section 3 introduces the constraints that govern the class of input problems and their graph-theoretical description (Fig. 1, II.a first step). This special graph structure will be exploited in the proposed algorithm to efficiently solve the TARP. Finally, this section presents an overview of the ILP-based problem formulation. Section 4 summarizes the theoretical foundations of the proposed algorithms and introduces the concepts of, e.g., fixed-parameter tractability and expression trees, which are needed for the subsequent discussion of the algorithm design. The proposed algorithms consist of two steps, which are described in Section 5 (Fig. 1, II.a second step) and Section 6 (Fig. 1, II.a third step). Section 5 shows how the graph-theoretical problem can be described using a special variant of the *clique-width k-expression tree*. In Section 6, this expression tree is used in a dynamic programming algorithm to resolve the transmit ambiguities. An improved backtracking version of this dynamic programming algorithm is described in Section 7 (Fig. 1, II.a third step). A numerical evaluation of the presented methods is presented in Section 8, and Section 9 gives a summary and an outlook regarding future work. To illustrate how the presented algorithms operate, Appendix 1 provides corresponding examples.

2 Problem introduction and system model

In this section, the system model and notation are introduced. Then, a summary of the implications of non-unique identification for agent localization is presented. It is assumed that each sensor has—in addition to its DS-CDMA code, which is henceforth called its ID—a unique serial number (SN) that is only accessible through physical access to the sensors. Such access is possible once the sensors have been extracted from the target environment and centralized localization is being performed. A sensor's ID, henceforth denoted as $I_{\mathcal{J}}$ for the Jth ID, is inherently encoded into its ranging pulse, as it serves as the DS-CDMA spreading sequence. The centralized localization unit also knows which IDs are used by which sensors. Hence, the mapping from the SNs to the IDs is known

and fixed. The following steps are performed to enable the localization of the sensor motes:

1. The agents, equipped with ultrasonic transducers, are introduced into the environment.
2. The agents record measurements of physical properties such as pressure and temperature. In addition, the agents perform round-trip time (RTT) measurements to facilitate distance estimates to neighboring agents. Because of the strict energy limitations, there is no forwarding or transmission of any data aside from the RTT measurements.
3. The agents are recovered from the environment, and their RTT measurements are jointly processed by the transmit-ambiguity solver to overcome the ambiguities in the distance measurements.
4. Classical localization algorithms are used to estimate the positions of the agents for each time instance when an RTT measurement was made.

The inaccuracy of the RTT ranging procedure is modeled as multiplicative noise. In this model, which has also been adopted by, for example, [7], it is assumed that distance estimates to farther sensors are less accurate. Consequently, the random variable (RV) $d_{j\to i}$, which denotes the distance measured by sensor i based on a ranging pulse sent by sensor j, is given by

$$d_{j\to i} = d_{i,j}\left(1 + n_{i,j}\right) \tag{1}$$

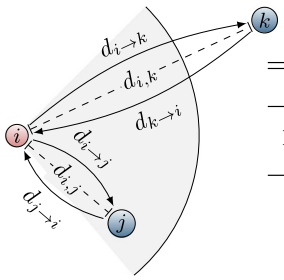
where all $n_{i,j} \sim \mathcal{N}\left(0, \sigma_{i,j}^2\right)$ are independently and identically distributed (iid) Gaussian variables with variance $\sigma_{i,j}^2$. Moreover, it is assumed that bidirectional measurements are made. Note that bidirectional communication is also a requirement for RTT measurements. Hence, it is assumed that if sensor i measured a distance to sensor j, then sensor j also measured a similar distance with respect to sensor i. An illustration of the notation is given in Fig. 2. The set of RVs referring to the range measurements received by sensor mote i, which uses ID $I_{\mathcal{L}}$, from nodes sending with ID $I_{\mathcal{K}}$ is denoted by $\mathcal{D}_i = \{d_{j\to i} | j \in \mathcal{K}\}$. The set of distances that sensor i has measured with respect to sensors using $\mathcal{I}_K$ is denoted by $\mathcal{M}_i = \left\{m_i^1, m_i^2, \ldots\right\}$, where m_i^l denotes the corresponding lth range measurement of sensor i[5]. As full connectivity is not necessarily given, the following holds: $|\mathcal{M}_i| \leq |\mathcal{D}_i|$. Consequently, $\mathcal{M}_i$ is an unordered or permuted subset of the random variates, i.e., realizations, of the RVs $\mathcal{D}_i$.

The following definition of the derived problem is used.

Definition 1 (transmit-ambiguity resolution problem (TARP)) *The TARP is the problem of overcoming the inability to differentiate among agents that use the same transmit code during communication. In the context of RTT ranging, e.g., for the sake of agent localization, the TARP is the problem of finding the correct mapping between the distance or RTT measurements m_i^l and the RVs $d_{j\to i}$ and, thus, the agents. Thereby, a decision is made regarding the source of a ranging pulse and its resulting distance or RTT measurement.*

An important aspect of the algorithm derivation and the henceforth used notation is that the TARP can be decomposed into independent subproblems, which are given by all pairs of—not necessarily distinct—IDs. In this regard, notice that the ambiguities occurring between two (distinct) groups of agents, each with possibly individual IDs, are not affected by measurements or ambiguities with any other groups of agents. This fact is also illustrated in Fig. 3a. Because of the independence of the subproblems and without loss of generality, only the measurements between sensor motes using, as an example, the IDs $I_{\mathcal{K}}$ and $I_{\mathcal{L}}$ are considered in the following.

Example 1 *An example of the consequences of the reuse of DS-CDMA codes with respect to localization is illustrated in Fig. 3a. This figure shows two sensor motes, "5" and "6," that are both using ID $I_{\mathcal{L}}$. A third sensor mote ("7") is using ID $I_{\mathcal{K}}$. Because the sensor motes can*



	$i \leftrightarrow j$	$i \leftrightarrow k$
Distance	$d_{i,j} = d_{j,i}$	$d_{i,k} = d_{k,i}$
Random variable	$d_{i\to j} = d_{i,j}(1 + n_{i,j})$ $d_{j\to i} = d_{i,j}(1 + n_{j,i})$	$d_{i\to k} = d_{i,k}(1 + n_{i,k})$ $d_{k\to i} = d_{i,k}(1 + n_{k,i})$
Measurement	$m_j^1 \dot{\sim} d_{i\to j}$ $m_i^1 \dot{\sim} d_{j\to i}$	— —

Fig. 2 Illustration of the notation and the effects of the transmit ambiguities. Considered here are an agent i, which is assumed to use ID $I_{\mathcal{R}}$ (*red*), and two surrounding agents (using ID $I_{\mathcal{B}}$ (*blue*)), of which one is within the communication range of i. Also illustrated are the actual distances between i and the other two agents and the RVs that describe potential measurements of those distances. For both cases, the table given on the right-hand side clarifies the notation and the fact that no measurement of the distance between i and k has been made. Here, $m \dot{\sim} d$ is used to denote that measurement m is a single realization, i.e., a sample, of the distribution d

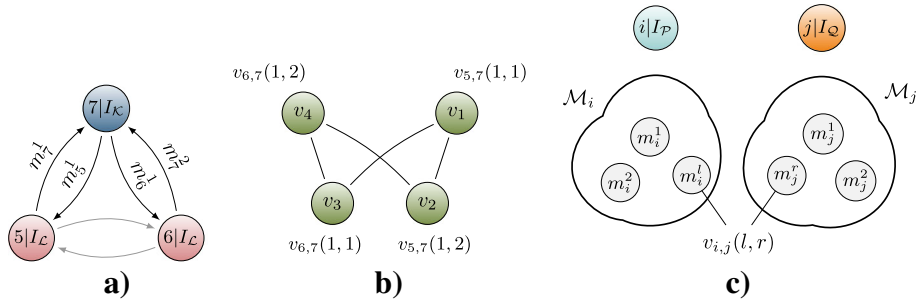


Fig. 3 a Illustration of a network with non-unique identification sequences $I_{\mathcal{K}}$ and $I_{\mathcal{L}}$ and their respective distance measurements as seen by the receiving node [14]. The gray arrows and their corresponding measurements are not considered in the subproblem between $I_{\mathcal{K}}$ and $I_{\mathcal{L}}$. **b** Ambiguity corresponding to **a**, where v_5 is given by the pair of measurements indicated in gray between agents "5" and "6." For clarity, the edges are annotated with the measurements that give rise to them (cf. items 1(a) and 1(b) in the listing in Section 3). Independent of the previous example, **c** provides a general illustration of how vertices $v_{i,j}(l,r)$ are formed based on the measurement sets $\mathcal{M}_i$ and $\mathcal{M}_j$ of agents i and j, respectively

differentiate only between different DS-CDMA codes and not between different sensor motes that are using the same code, sensor mote "7" measures two distances that cannot be related to sensor motes "5" and "6." More precisely, sensor mote "7" measures $\mathcal{M}_7 = \{m_7^1 m_7^2\}$ *(realizations of the RVs contained in* $\mathcal{D}_7 = \{d_{5\to7} d_{6\to7}\}$*), and before offline localization can be performed, the mapping between these measurements and sensors "5" and "6" needs to be estimated.*

Consequently, *classical* localization methods, which do not consider application scenarios with transmit ambiguities—such as [3, 6, 7], which use semidefinite programming, or [11], which uses a linearized least-squares approach—cannot be used without first applying algorithms to resolve the TARP such as those presented in this work.

3 Graph representation and graph problem formulation

In this section, a graph-theoretical formulation of the problem resulting from DS-CDMA code reuse is given. Moreover, a detailed analysis of the problem's properties, which will be exploited in Section 4, is presented.

The following description utilizes the particular properties of a type of graph whose instances are henceforth called *ambiguity graphs*. Such a graph is denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, and its components, i.e., the vertex set, edge set and weight function, are defined in the following.

Every vertex of the ambiguity graph is uniquely identified by the notation $v_{i,j}(l,r)$, where i and j refer to agents i and j and l and r refer to their respective lth and rth measurements. Therefore, the vertex $v_{i,j}(l,r)$ refers to the RVs $(d_{j\to i}, d_{i\to j})$ and their potential realization given by the measurements $\left(m_i^l, m_j^r\right)$. In the following, this relation is denoted by $v_{i,j}(l,r) \overset{\text{ref.}}{\sim} \left(m_i^l, m_j^r\right)$. Moreover, this vertex has a weight, which corresponds to the likelihood that the measurements m_i^l and m_j^r were both measured between agents i and j. This section describes how this weight is calculated and presents the definition of the edges of this graph.

Example 2 *A simple example of an ambiguity graph for a network is shown in Fig. 3a. For simplicity, a linear index* v_k *is used to denote the vertices of the ambiguity graph. The relationship between this linear notation and the* $v_{i,j}(l,r)$ *notation is as follows:* $v_1 = v_{5,7}(1,1)$*, which relates to* $\left(m_5^1, m_7^1\right)$*;* $v_2 = v_{5,7}(1,2)$*, which relates to* $\left(m_5^1, m_7^2\right)$*;* $v_3 = v_{6,7}(1,1)$*, which relates to* $\left(m_7^1, m_6^1\right)$*; and* $v_4 = v_{6,7}(1,2)$*, which relates to* $\left(m_6^1, m_7^2\right)$*. With this short-hand notation, the edge and vertex sets for this example are given as follows:* $\mathcal{E}_{1,a}(5,1) = \{(v_1, v_2)\}$, $\mathcal{E}_{1,a}(6,1) = \{(v_3, v_4)\}$, $\mathcal{E}_{1,b}(7,1) = \{(v_1, v_3)\}$, $\mathcal{E}_{1,b}(7,2) = \{(v_2, v_4)\}$, $\mathcal{E}_2(5,7) = \{(v_1, v_2)\}$, $\mathcal{E}_2(6,7) = \{(v_3, v_4)\}$, $\mathcal{V}_{1,a}(5,1) = \{v_1, v_2\}$, $\mathcal{V}_{1,a}(6,1) = \{v_2, v_4\}$, $\mathcal{V}_{1,b}(7,1) = \{v_1, v_3\}$, $\mathcal{V}_{1,b}(7,2) = \{v_2, v_4\}$, $\mathcal{V}_2(5,7) = \{v_1, v_2\}$, *and* $\mathcal{V}_2(6,7) = \{v_3, v_4\}$.

The set of edges $\mathcal{E}$ of the ambiguity graph is the union of three edge sets that satisfy the criteria given below.

1. In the following problem descriptions, edges are used to denote complementary constraints, and any distance measurement m_i^l can be assigned to only one RV $d_{\bullet\to i}$; therefore, edges are introduced as follows:
   (a) Between $v_{i,j}(l,r) \overset{\text{ref.}}{\sim} \left(m_i^l, m_j^r\right)$ and any other vertex $v_{i,j'}(l,r') \overset{\text{ref.}}{\sim} \left(m_i^l, m_{j'}^{r'}\right)$ that relates to m_i^l. The set of all edges created based on this condition is denoted by $\mathcal{E}_{1,a}(i,l)$.

$$\mathcal{E}_{1,a}(i,l) = \left\{(v_x, v_y) \in \mathcal{V} \times \mathcal{V} \middle|\, v_x \overset{\text{ref.}}{\sim} \left(m_i^l, m_j^r\right) \wedge v_y \overset{\text{ref.}}{\sim} \left(m_i^l, m_{j'}^{r'}\right)\right\} \tag{2}$$

(b) Between $v_{i,j}(l,r) \overset{\text{ref.}}{\sim} \left(m_i^l, m_j^r\right)$ and any other vertex $v_{i',j}(l',r) \overset{\text{ref.}}{\sim} \left(m_{i'}^{l'}, m_j^r\right)$ that relates to m_j^r. The set of all edges created based on this condition is denoted by $\mathcal{E}_{1,b}(j,r)$.

$$\mathcal{E}_{1,b}(j,r) = \left\{(v_x, v_y) \in \mathcal{V} \times \mathcal{V} \mid v_x \overset{\text{ref.}}{\sim} \left(m_i^l, m_j^r\right) \wedge v_y \overset{\text{ref.}}{\sim} \left(m_{i'}^{l'}, m_j^r\right)\right\} \tag{3}$$

2. Similarly, a pair (tuple) of RVs $(d_{j\to i}, d_{i\to j})$ can only be matched to at most one tuple of measurements, i.e., one vertex of the ambiguity graph. Therefore, edges are introduced between any vertex $v_{i,j}(l,r)$ and $v_{i,j}(l',r')$, i.e., any other vertex that corresponds to the same bidirectional link between sensors i and j. The set of all edges created based on this condition is denoted by $\mathcal{E}_2(i,j)$.

$$\mathcal{E}_2(i,j) = \left\{(v_x, v_y) \in \mathcal{V} \times \mathcal{V} \mid v_x \overset{\text{ref.}}{\sim} \left(m_i^l, m_j^r\right) \wedge v_y \overset{\text{ref.}}{\sim} \left(m_i^{l'}, m_j^{r'}\right)\right\} \tag{4}$$

Consequently, the edge set $\mathcal{E}$ is given by $\mathcal{E} = \mathcal{E}_{1,a} \cup \mathcal{E}_{1,b} \cup \mathcal{E}_2$, where

$$\mathcal{E}_{1,a} = \bigcup_{i\in\mathcal{K}} \bigcup_{l=1,\dots,|\mathcal{M}_i|} \mathcal{E}_{1,a}(i,l) \tag{5a}$$

$$\mathcal{E}_{1,b} = \bigcup_{j\in\mathcal{L}} \bigcup_{r=1,\dots,|\mathcal{M}_j|} \mathcal{E}_{1,b}(j,r) \tag{5b}$$

$$\mathcal{E}_2 = \bigcup_{(i,j)\in\mathcal{K}\times\mathcal{L}} \mathcal{E}_2(i,j) \tag{5c}$$

Note that the index of each edge set, such as $_{1,a}$ or $_2$, refers to the corresponding (sub)item and its constraints as given in the listing above.

The objective is to maximize the probability $\mathbb{P}$ of correctly matching the RVs to the distance measurements given all measurements $\mathcal{M}_i$ from all sensors i, i.e.,

$$\mathbb{P}\left\{(d_{j\to i}, d_{i\to j}) = (m_i^l, m_j^r), \forall i,j,l,r \middle| d_{j\to i}\in\mathcal{M}_i, \forall j,i\right\} \tag{6}$$

by choosing the weights $\mathcal{W}(v_{i,j}(l,r))$ of the vertices according to

$$\mathcal{W}(v_{i,j}(l,r)) = -\log f_{d_{j\to i},d_{i\to j}}\left(m_i^l, m_j^r\right) \tag{7}$$

where $f_{d_{j\to i},d_{i\to j}}(x,y)$ denotes the joint probability density function (PDF) of the RVs $d_{j\to i}$ and $d_{i\to j}$ and, hence, of the noisy distance measurements. A derivation of the PDF used in the numerical evaluation of this work is given in Appendix 2.

Thus, the construction of the ambiguity graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ can be summarized as follows:

- Every vertex $v_{i,j}(l,r)$ is uniquely identified by a pair of measurements $\left(m_i^l, m_j^r\right)$. This vertex describes a possible solution candidate for the measurement assignment problem. More precisely, it reflects the probability that m_i^l and m_j^r are the realizations of $d_{i\to j}$ and $d_{j\to i}$, respectively. In the subsequently described optimization problem, the goal will therefore be to select a subset of these solution candidates.
- The likelihood that the mapping denoted by $v_{i,j}(l,r)$ is correct is given by the corresponding weight $\mathcal{W}(v_{i,j}(l,r))$.
- The edges (cf. Eq. (5)) govern the criteria under which a solution candidate $v_{i,j}(l,r)$ may be selected. Note that the vertices that are subject to particular constraints, such as $\mathcal{E}_{1,a}(i,l)$, $\mathcal{E}_{1,b}(j,r)$, or $\mathcal{E}_2(i,j)$, are the vertices that are the endpoints of the edges in these sets. The corresponding sets of vertices and their subgraphs are denoted by $\mathcal{V}_{1,a}(i,l)$, $\mathcal{V}_{1,b}(j,r)$, and $\mathcal{V}_2(i,j)$ and by $\mathcal{G}_{1,a}(\cdot,\cdot) = \left(\mathcal{V}_{1,a}(\cdot,\cdot), \mathcal{E}_{1,a}(\cdot,\cdot)\right)$, $\mathcal{G}_{1,b}(\cdot,\cdot) = \left(\mathcal{V}_{1,b}(\cdot,\cdot), \mathcal{E}_{1,b}(\cdot,\cdot)\right)$, and $\mathcal{G}_2(\cdot,\cdot) = (\mathcal{V}_2(\cdot,\cdot), \mathcal{E}_2(\cdot,\cdot))$.
- Every vertex $v_{i,j}(l,r) \overset{\text{ref.}}{\sim} \left(m_i^l, m_j^r\right)$ is part of the three subgraphs $\mathcal{G}_{1,a}(i,l)$, $\mathcal{G}_{1,b}(j,r)$, and $\mathcal{G}_2(i,j)$. This is because by the definition of item 1(a) in the listing above, $v_{i,j}(l,r)$ is adjacent to any other vertex that relates to the measurement m_i^l, and by the definition of item 1(b), $v_{i,j}(l,r)$ is also adjacent to any other vertex that relates to the measurement m_j^r. Moreover, $v_{i,j}(l,r)$ is a solution candidate for the measurements between agents i and j and hence is also an endpoint of the edges in $\mathcal{E}_2(i,j)$ and is therefore adjacent to the vertices in $\mathcal{V}_2(i,j)$.

For brevity and simplicity, in the following, we may refer to the vertices $v_{i,j}(l,r)$ as v_k, $k \in \{1,\dots,|\mathcal{V}|\}$, if the additional information provided by the indices i, j, l, and r of $v_{i,j}(l,r)$ is not relevant.

Based on the definition of the ambiguity graph given above, it is easy to verify that the optimal resolution of the transmitter ambiguities, in the sense of maximizing (6) for all assignments $\left(m_i^l, m_j^r\right)$, is equivalent to solving the following *minimum-weight independent set problem with size constraints* (MW-ISP-SC):

$$\arg\min_{\boldsymbol{x}} \quad \boldsymbol{w}^T\boldsymbol{x} \tag{8a}$$

$$\text{subject to} \quad \sum\nolimits_{v_k\in\mathcal{V}_{1,a}(i,l)} x_k = 1, \quad \forall\, \mathcal{V}_{1,a}(i,l) \in \mathcal{V}_{1,a} \tag{8b}$$

$$\sum\nolimits_{v_k \in \mathcal{V}_{1,b}(j,r)} x_k = 1, \quad \forall\, \mathcal{V}_{1,b}(j,r) \in \mathcal{V}_{1,b} \tag{8c}$$

$$\sum\nolimits_{v_k \in \mathcal{V}_2(i,j)} x_k \leq 1, \quad \forall\, \mathcal{V}_2(i,j) \in \mathcal{V}_2 \tag{8d}$$

$$x_i + x_j \leq 1, \quad \forall\, (i,j) \in \mathcal{E} \tag{8e}$$

$$x_k \in \{0,1\}, \quad \forall\, k = 1, \ldots, |\mathcal{V}| \tag{8f}$$

where x_k is either one or zero and indicates whether vertex k of the ambiguity graph is in the solution set or not, respectively. This means that any measurements $\left(m_i^l, m_j^r\right)$ that are in the solution set, i.e., whose corresponding vertices $v_{i,j}(l,r)$ are in the independent set, are chosen as the realizations of the RVs $(d_{j\to i}, d_{i\to j})$. Furthermore, w_k denotes the weight of vertex $v_k \in \mathcal{V}$, i.e., $w_k = \mathcal{W}(v_k)$. Constraint (8e) ensures the independence of the set by allowing at most one of two adjacent vertices to be included in the solution set. Constraints (8b) to (8d) refer to the conditions described in items 1(a), 1(b), and 2 of the listing above and constitute additional constraints compared with the classical *independent set problem*; they are called *size constraints* because they enforce independent sets of a particular size.

Note that because of the choice of the vertex weights (cf. Eq. (7)), the solution to Problem (8) corresponds to the feasible solution with the highest probability in the MAP sense (cf. Eq. (6)). In addition, note that Problem (8) is $\mathcal{NP}$-hard in general [17, 18].

4 Exploiting the graph structure

Because of the $\mathcal{NP}$-hardness of Problem (8), new algorithms are needed that can solve the TARP optimally but more efficiently than regular ILP solvers. Less computationally complex algorithms are particularly necessary to accommodate the large number of agents needed for the considered application case. The idea that serves as the basis for the following sections is to employ dynamic programming to obtain a highly tailored algorithm to facilitate efficient solution generation. This is enabled by exploiting the special graph structure described in Section 3. The underlying principle of this exploitation is the *fixed-parameter tractability* (FPT) of the problem.

In this regard, Section 4.1 introduces the concept of FPT and a theorem that renders it applicable to the TARP and thereby forms the basis of our chosen dynamic programming approach. Section 4.2 presents an overview of the fundamental graph-theoretical concepts, including graph decomposition, that enable the application of the FPT theorem to the TARP. Based on these concepts, Section 5 introduces a novel variant of this decomposition that is tailored to ambiguity graphs.

4.1 Fixed-parameter tractability

The concept of FPT is generally used to analyze problems and, in particular, to find which parameters significantly influence the computational complexity. This analysis can then be used to design efficient algorithms for solving $\mathcal{NP}$-hard problems such as the TARP.

The idea of FPT is to describe the complexity of an $\mathcal{NP}$-hard problem as a combination of two aspects: the first depending only on the input, such as the number of vertices of a graph, and the second being given by some *fixed parameter* of the problem. Generally, a problem is said to be fixed-parameter tractable if some algorithm capable of solving this problem has a complexity of $\mathcal{O}(\text{poly}(|x|)f(k))$, where $|x|$ denotes the size of the input, $\text{poly}(\cdot)$ denotes some polynomial function, and $f(k)$ is some arbitrary function of the fixed parameter k [19].

The algorithm proposed in this work is based on a theorem presented by [20], which states that finding the independence number, i.e., the size of the largest independent set, is a fixed-parameter tractable problem in which the complexity parameter k refers to a graph property or parameter known as the *clique-width*. The *clique-width* is introduced in detail in Section 4.2 below. Regarding the applicability of this theorem to the TARP, note that in Problem (8), the goal is to find an independent set with a particular independence number that is given by the number of distinct sets of types $\mathcal{E}_{1,a}(\cdot,\cdot)$ and $\mathcal{E}_{1,b}(\cdot,\cdot)$[6]. Because the *decision* variant of *finding the independence number* can be interpreted as answering the question "is there an independent set with a size of at least n?", the following theorem is applicable to Eq. (8).

Theorem 1 (The problem of finding the independence number is fixed-parameter tractable [20]) *The problem of determining the independence number of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a clique-width of at most k has a complexity of $\mathcal{O}(|\mathcal{V}|f(k))$ if $\mathcal{G}$ is given by a clique-width k-expression.*

Consequently, Theorem 1 states that the clique-width k of graph $\mathcal{G}$ is an important parameter when the desire is to find the independence number of $\mathcal{G}$. This is highlighted by the fact that for the problem of finding the independence number, [21] specifies the particular function $f(k) = 2^k$ to concretize the complexity in the FPT theorem above [22]. Consequently, the objective in the following is to find a clique-width k-expression for instances of the ambiguity graph with the minimum possible clique-width k. However, computing the clique-width k of a graph is an $\mathcal{NP}$-hard problem in general [23]. Therefore, sub-optimal methods for finding some approximation $k' \geq k$ of the clique-width are presented in this work.

To this end, Section 4.2 reviews the concept of the clique-width and shows how graphs can be described by certain operations that yield *expressions* that can then be

modeled using a tree. In this way, a graph is said to be decomposed, e.g., into a tree structure.

Remark 1 *The motivation for using tree-structure decompositions of graphs, such as tree-decompositions or clique-width k-expression trees, stems mainly from the fact that the independent set problem, for example, can be efficiently solved by applying dynamic programming on such a representation (tree structure)* [20, 24]. *Our particular interest in decomposing graphs into clique-width k-expression trees is mainly driven by the FPT theorem introduced above and by the fact that the clique-width is upper bounded by the tree-width* [25]. *Furthermore, note that in our case, in which the decomposition of ambiguity graph instances is desired, the edge set $\mathcal{E}$ is given by a union of cliques*[7] *(cf. Eq. (5)), which turns out to simplify the formulation of an expression for the clique-width.*

4.2 Graph theory definitions and notations: clique-width

Recall that, motivated by Theorem 1, the goal is to find clique-width k-expressions that correspond to ambiguity graph instances to enable efficient algorithms for solving the TARP. To this end, this section reviews the definitions of the clique-width and clique-width k-expression trees; this discussion is mainly based on [20, 21].

The clique-width is calculated based on recursive operations on vertex-labeled graphs [20]. These operations construct a new clique-width k-expression tree. In the following, $\mathcal{G} = (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G}, lab_\mathcal{G})$ denotes a k-labeled graph where each vertex label is given by the mapping $lab_\mathcal{G} : \mathcal{V}_\mathcal{G} \to [k]$, with $[k]$ being the set of natural numbers $[k] := \{1, \ldots, k\}$ [21].

Definition 2 (Clique-width [20, 21]) *The clique-width of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is the minimum number of vertex labels $lab : \mathcal{V} \to [k]$ such that $\mathcal{G}$ can be constructed by the following operations:*

- *Disjoint union of vertex-labeled graphs: Let $\mathcal{G} = (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G}, lab_\mathcal{G}) \in CW_k$ and $\mathcal{J} = (\mathcal{V}_\mathcal{J}, \mathcal{E}_\mathcal{J}, lab_\mathcal{J}) \in CW_k$ be two vertex-disjoint labeled graphs; then, $\mathcal{G} \oplus \mathcal{J} := (\mathcal{V}', \mathcal{E}', lab')$ is also in CW_k, with $\mathcal{V}' := \mathcal{V}_\mathcal{G} \cup \mathcal{V}_\mathcal{J}$, $\mathcal{E}' := \mathcal{E}_\mathcal{G} \cup \mathcal{E}_\mathcal{J}$, and*

$$lab'(u) := \begin{cases} lab_\mathcal{G}(u) & \text{if } u \in \mathcal{V}_\mathcal{G} \\ lab_\mathcal{J}(u) & \text{if } u \in \mathcal{V}_\mathcal{J} \end{cases}$$

- *Edge introductions $\eta_{i,j} : \eta_{i,j}(\mathcal{G}) := (\mathcal{V}_\mathcal{G}, \mathcal{E}'_\mathcal{G}, lab_\mathcal{G})$, where*

$$\mathcal{E}'_\mathcal{G} := \mathcal{E}_\mathcal{G} \cup \big\{\{u, v\} \big| u, v \in \mathcal{V}_\mathcal{G}, u \neq v, lab_\mathcal{G}(u) = i, lab_\mathcal{G}(v) = j\big\}$$

- *Vertex relabeling $\rho_{i \to j} : \rho_{i \to j}(\mathcal{G}) := (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G}, lab'_\mathcal{G})$, where*

$$lab'_\mathcal{G}(u) := \begin{cases} lab_\mathcal{G}(u) & \text{if } lab_\mathcal{G}(u) \neq i \\ j & \text{if } lab_\mathcal{G}(u) = i \end{cases}, \forall u \in \mathcal{V}_\mathcal{G}$$

The resulting expression is called a clique-width k-expression or a CW_k-expression.

Using Definition 2, a CW_k-tree can also be defined, where the root of the tree corresponds to the original unlabeled ambiguity graph $\mathcal{G}$. Consequently, at the root r of the expression tree, which corresponds to the complete expression, all vertices must have the same label such that the root of the tree is equal to the unlabeled graph $\mathcal{G}$.

Table 1 presents examples of CW_2-expressions and the corresponding graphs. Note that only $k = 2$ labels are required to represent $\mathcal{G}_6$, which has four vertices.

Example 3 *Figure 4 shows the CW_2-tree for the labeled graph $\mathcal{G}_6$ given in Table 1. The vertices of the tree are annotated with the induced subgraphs of the tree nodes*[8]. *Note that in this example, the original graph, $\mathcal{G}_6$, was already labeled, and hence, the root of the CW_k-tree is identical to the original graph, with the same labels.*

Generally, the decompositions into CW_k-expressions and CW_k-trees are not unique, i.e., multiple possible CW_k-trees exist for a given graph. The main benefit of representing a graph according to Definition 2 is that the CW_k-expression is directly related to the complexity of the MW-ISP-SC, which is of $\mathcal{O}(|\mathcal{V}|2^k)$.

Moreover, the attempt to solve such a problem using the CW_k-tree $\mathcal{T}$ is motivated by the property that the vertex set of subgraph $\mathcal{G}_s$, which corresponds to tree node s, is partitioned into $k \leq |\mathcal{V}_s|$ sets, as given by the k labels [20]. The reason for this is that during the decomposition of a graph into a CW_k-expression tree, the labeling will be performed such that, in the dynamic programming phase,

Table 1 CW_2-expressions and their corresponding 2-labeled graphs [20]

Graph		CW_2-expression
$\mathcal{G}_1$	①	$\bullet_1$
$\mathcal{G}_2$	②	$\bullet_2$
$\mathcal{G}_3$	① ②	$\mathcal{G}_1 \oplus \mathcal{G}_2$
$\mathcal{G}_4$	① — ②	$\eta_{1,2}(\mathcal{G}_3)$
$\mathcal{G}_5$	① — ①	$\rho_{2 \to 1}(\mathcal{G}_4)$
$\mathcal{G}_6$	① ① ① ②	$\eta_{1,2}(\mathcal{G}_5 \oplus \mathcal{G}_3)$

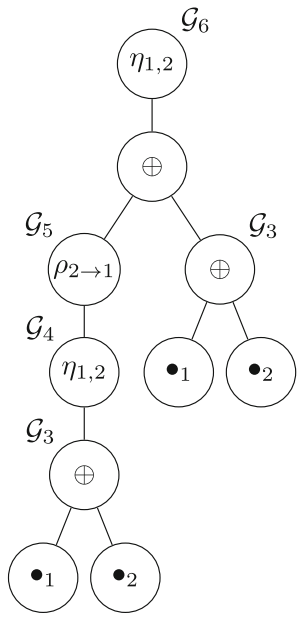


Fig. 4 CW_2-expressions and their corresponding 2-labeled graphs [20]

it is sufficient to differentiate between groups of vertices that have different labels.

Example 4 *Considering the independent set problem as an example, the independence criterion does not need to be checked for every vertex because all vertices that have the same properties in this regard will have the same label. This is ensured by the design of the labeling function, which is described in the following section. Consequently, for a group of vertices with the same label, a check for independence requires only a single check rather than one check for each vertex in that group.*

5 The *k*-ambiguity tree

To enable the exploitation of the property mentioned in the FPT theorem, i.e., that the clique-width of the input graph is a critical parameter in determining the complexity of the TARP, an algorithm for obtaining the tree description—often called a graph decomposition in the literature, as it disassembles parts of the graph into a tree structure—and related operations are presented in this section. To this end, Section 5.1 introduces a shorthand, tailored version of the operations used in the CW_k-expressions. The main difference compared with the CW_k operations is from a special design criterion that is subsequently introduced, which leads to the introduction of additional predefined edges. The tree corresponding to this tailored description is called the k-ambiguity tree in the following. Contrary to the notation of the CW_k-tree, the leaves of the k-ambiguity tree are chosen to be the subgraphs that correspond to the edge constraint sets $\mathcal{E}_{1,a}(\cdot,\cdot)$, $\mathcal{E}_{1,b}(\cdot,\cdot)$, and $\mathcal{E}_2(\cdot,\cdot)$, cf. Eqs. (5a), (5b), and (5c). These subgraphs are denoted by $\mathcal{G}_{1,a}(\cdot,\cdot) = \big(\mathcal{V}_{1,a}(\cdot,\cdot),\mathcal{E}_{1,a}(\cdot,\cdot)\big)$, $\mathcal{G}_{1,b}(\cdot,\cdot) = \big(\mathcal{V}_{1,b}(\cdot,\cdot),\mathcal{E}_{1,b}(\cdot,\cdot)\big)$, and $\mathcal{G}_2(\cdot,\cdot) = (\mathcal{V}_2(\cdot,\cdot),\mathcal{E}_2(\cdot,\cdot))$, respectively. Note that according to the definition of the edge constraint sets, they are complete induced[9] subgraphs, i.e., cliques[10].

Recall, from the definition of CW_k-expressions, that any expression with the operators $\oplus$, $\eta_{i,j}$, or $\rho_{i\to j}$, e.g., in the form of $\mathcal{X}_1 \oplus \mathcal{X}_2$, in the CW_k-tree corresponds to the *merging* of two trees $\mathcal{T}_{\mathcal{X}_1}$ and $\mathcal{T}_{\mathcal{X}_2}$ that, in turn, correspond to the expressions $\mathcal{X}_1$ and $\mathcal{X}_2$, respectively. Consequently, we use $\mu(\mathcal{T}_1,\mathcal{T}_2)$ to denote a joint operator that applies a sequence of $\oplus$, $\eta_{i,j}$, or $\rho_{i\to j}$ operations to serve as a *merging* operator. This joint operator is motivated by the canonical form of the clique-width expression [26], in which after each disjoint union $\oplus$, first, several edge introductions $\eta_{i,j}$ and then several relabeling operations $\rho_{i\to j}$ must occur [20].

During this merging process, the key idea is to choose an operation such that its output is an *induced* subgraph of $\mathcal{G}$. This is because the number of labels required for induced subgraphs is lower than that for non-induced subgraphs[11]. This *merging* operation is described in more detail below.

5.1 The merging operator

The merging operator performs the steps listed below, thereby merging the labeled subgraphs $\mathcal{G}_{c_1(s)}$ and $\mathcal{G}_{c_2(s)}$ to yield the output $\mathcal{G}_s = (\mathcal{V}_s,\mathcal{E}_s,lab_s)$. In the following, we use the index $c_i(s)$, $i = 1,2$, to represent the edge or vertex set of the first or second child of tree node s (see also Fig. 6a).

1. Union of the children's vertex sets: $\mathcal{V}_s = \mathcal{V}_{c_1(s)} \cup \mathcal{V}_{c_2(s)}$.
2. Union of the children's edge sets and a further edge set $\mathcal{E}_{c_1(s),c_2(s)}$: $\mathcal{E}_s = \mathcal{E}_{c_1(s)} \cup \mathcal{E}_{c_2(s)} \cup \mathcal{E}_{c_1(s),c_2(s)}$. The additional edge set $\mathcal{E}_{c_1(s),c_2(s)}$ is chosen such that $\mathcal{G}_s$ is an induced subgraph, which—as mentioned above—is the desired criterion for the merging operator. Note that $\mathcal{G}_{c_1(s)}$ and $\mathcal{G}_{c_2(s)}$ are both also induced subgraphs, as they are also the results of merging or are cliques corresponding to leaves of the tree. Therefore, the set $\mathcal{E}_{c_1(s),c_2(s)}$ contains only edges of the form $(v_i,v_j), v_i \in \mathcal{V}_{c_1(s)}, v_j \in \mathcal{V}_{c_2(s)}$.
3. Vertex relabeling resulting in the fewest possible labels, $\min_{lab_s} |\{lab_s(v)|v \in \mathcal{V}_s\}|$.

The labeling function required to perform steps 2 and 3 is defined in the following. Recall from Section 3 that every vertex $v_{i,j}(l,r)$ is an element of exactly three cliques, namely, $\mathcal{G}_{1,a}(i,l)$, $\mathcal{G}_{1,b}(j,r)$, and $\mathcal{G}_2(i,j)$. Because of this property, all vertices that are adjacent to $v_{i,j}(l,r)$ are in the sets $\mathcal{V}_{1,a}(i,l)$, $\mathcal{V}_{1,b}(j,r)$, and $\mathcal{V}_2(i,j)$.

Because the k-ambiguity tree corresponding to the ambiguity graph needs to reflect an increasing number of the properties of the original graph as the considered tree node becomes closer to the root of the tree, and in particular because the root of the tree represents the original graph, a tool is needed for describing which properties of a vertex of the original graph are not currently considered at a given tree node. To this end, a function $R_s(v_{i,j}(l,r))$ is defined that describes the properties of a vertex $v_{i,j}(l,r)$ with respect to its originally adjacent vertices. More precisely, $R_s(v_{i,j}(l,r))$ indicates the membership of the three cliques corresponding to $v_{i,j}(l,r)$ with respect to the subgraph $\mathcal{G}_s$ at tree node s and is defined as follows:

$$R_s(v_{i,j}(l,r)) = \big((i,l)\cdot\overline{\mathbb{1}}_{\mathcal{V}_s}(\mathcal{V}_{1,a}(i,l)),\ (j,r)\cdot\overline{\mathbb{1}}_{\mathcal{V}_s}(\mathcal{V}_{1,b}(j,r)),\ (i,j)\cdot\overline{\mathbb{1}}_{\mathcal{V}_s}(\mathcal{V}_2(i,j))\big) \tag{9}$$

where $\overline{\mathbb{1}}_{\mathcal{V}_s}(\mathcal{V})$ is related to the inverted indicator function and is defined as $\overline{\mathbb{1}}_{\mathcal{V}_s}(\mathcal{V}) = 0$ if $\mathcal{V} \subseteq \mathcal{V}_s$ and 1 otherwise. Thus, a missing edge to a vertex to which $v_{i,j}(l,r)$ was originally connected is indicated by a non-zero tuple entry. An illustration of this function is provided in Fig. 5.

Considering the labeling of the vertices, every occurring three-tuple value of R_s is chosen to correspond to a distinct label in the subgraph of tree node s. Thus, the tuple elements of two vertices, $R_{c_1(s)}(v_1)$ and $R_{c_2(s)}(v_2)$, are sufficient to determine whether an edge must be inserted between v_1 and v_2 by the operator μ (or $\eta_{i,j}$): iff the values and positions of at least one tuple entry in each of $R_{c_1(s)}(v_1)$ and $R_{c_2(s)}(v_2)$ are equal and non-zero, then an edge is inserted between v_1 and v_2. This is because these vertices are part of the same clique in the ambiguity graph, cf. Section 3. Recall that a tuple entry of zero indicates that all edges for the corresponding clique have already been inserted.

Moreover, the function R_s determines the relabeling according to the operator μ (or $\rho_{i\to j}$). Iff $R_s(v_1) = R_s(v_2)$, i.e., if the complete tuples are equal, then v_1 and v_2 are assigned the same label at s, as illustrated in the following example.

Example 5 *We consider two vertices v_1 and v_2 at tree node s and assume that $R_s(v_1) = \big((0,0),(1,1),(0,0)\big) = R_s(v_2)$. Then, v_1 and v_2 are both lacking edges to all vertices in clique $\mathcal{V}_{1,b}(1,1)$ that are not in $\mathcal{V}_s$, i.e., to all $v \in \mathcal{V}_{1,b}(1,1)\backslash\mathcal{V}_s$. Consider the merging of tree node s with another tree node s' by operator μ: if an edge is inserted between v_1 and $v_3 \in \mathcal{V}_{s'}$, then an edge must also be inserted between v_2 and v_3 because v_1 and v_2 have the same properties with respect to their current clique memberships. Therefore, no distinction is needed between vertices v_1 and v_2 with respect to the edge insertion operation, and consequently, the same label is assigned to both vertices. This label is uniquely determined by $R_s(v_1) = R_s(v_2)$.*

Note that no two vertices are part of the same three cliques (cf. Section 3), and hence, two vertices can have the same label only if at least one tuple entry of R_s is zero for both vertices. At the root r of the ambiguity tree, which corresponds to the entire ambiguity graph $\mathcal{G}$, all edges are already inserted and $R_r(v_i) = ((0,0),(0,0),(0,0))\forall v_i \in \mathcal{V}$; therefore, only a single label is needed.

5.2 Building the k-ambiguity tree

Using the methods and, in particular, the merging operator defined previously, this section introduces an

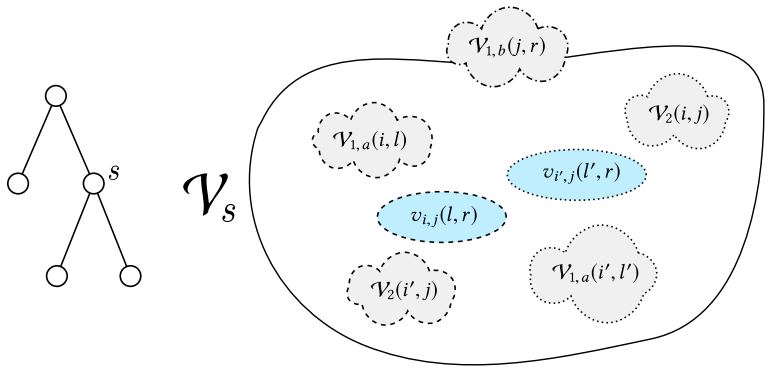


Fig. 5 Illustration of a tree node s and its vertex set $\mathcal{V}_s$, which is used to visualize $R_s(v)$ for the two vertices $v_{i,j}(l,r)$ and $v_{i',j}(l',r)$. The corresponding vertex sets for $v_{i,j}(l,r)$ are $\mathcal{V}_{1,a}(i,l)$, $\mathcal{V}_{1,b}(j,r)$, and $\mathcal{V}_2(i,j)$ and are indicated by dashed contours. The sets $\mathcal{V}_{1,a}(i',l')$, $\mathcal{V}_{1,b}(j,r)$, and $\mathcal{V}_2(i',j)$ for $v_{i',j}(l',r)$ are shown with dotted contours. The set $\mathcal{V}_{1,b}(j,r)$, which is considered for both vertices, therefore has a dash-dotted contour. The purpose of visualizing $R_s(v)$ for the two vertices is to check whether all vertices of the abovementioned sets $\mathcal{V}_\bullet(\bullet,\bullet)$ are already included in the vertex set $\mathcal{V}_s$ at the current tree node s. For the two vertices in the example above, we have $R_s(v_{i,j}(l,r)) = \big((0,0),(j,r),(0,0)\big) = R_s(v_{i',j}(l',r))$ because for both vertices, the corresponding set $\mathcal{V}_{1,b}(\bullet,\bullet)$ is not fully included in $\mathcal{V}_s$. Consequently, both vertices share the same property, which is captured by $R_s(v_{i,j}(l,r)) = R_s(v_{i',j}(l',r))$. Therefore, they also have the same label (color)

algorithm that is able to generate k-ambiguity trees based solely on the subgraphs $\mathcal{G}_{1,a}(\cdot,\cdot)$, $\mathcal{G}_{1,b}(\cdot,\cdot)$, and $\mathcal{G}_2(\cdot,\cdot)$.

Because finding the clique-width decomposition of a graph $\mathcal{G}$ is an $\mathcal{NP}$-hard problem in general, the decomposition of an ambiguity graph into a k-ambiguity tree can also be efficiently done only in a sub-optimal way. Consequently, only decompositions that require a number of labels greater than the clique-width are of computational interest. However, note that for such a decomposition, there is no need to build the actual ambiguity graph because the equivalent information is also provided by all subgraphs of types $\mathcal{G}_{1,a}(\cdot,\cdot)$, $\mathcal{G}_{1,b}(\cdot,\cdot)$, and $\mathcal{G}_2(\cdot,\cdot)$. A subset of these basic entities is therefore chosen as the leaves of the k-ambiguity tree. Note that choosing these cliques as the leaves results in an entry of zero in $R_s(v)$ for each vertex $v \in \mathcal{V}_s$. The purpose of this heuristic is to reduce the number of distinct three-tuples at tree nodes closer to the root of the tree and thereby reduce the number of labels.

To simplify the notation in the following steps, $\mathcal{C}$ is used to denote the set that contains all cliques of types $\mathcal{G}_{1,a}(\cdot,\cdot)$, $\mathcal{G}_{1,b}(\cdot,\cdot)$, and $\mathcal{G}_2(\cdot,\cdot)$. Similarly, all *labeled* cliques are denoted by $\mathcal{C}_{lab}$.

Example 6 $\mathcal{G}_{1,a}(i,l) = \big(\mathcal{V}_{1,a}(i,l), \mathcal{E}_{1,a}(i,l)\big) \in \mathcal{C}$ *and* $\big(\mathcal{V}_{1,a}(i,l), \mathcal{E}_{1,a}(i,l), lab_{1,a}(i,l)\big) \in \mathcal{C}_{lab}$.

The set $\mathcal{T}$ is the set of all partial ambiguity trees, i.e., all possible branches of the ambiguity tree that relate to a certain subgraph of the ambiguity graph $\mathcal{G}$. Furthermore, the neighborhood $\mathcal{N}(\mathcal{G}_i)$ of a labeled subgraph $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i, lab_i)$ consists of those subgraphs $\mathcal{G}_j = (\mathcal{V}_j, \mathcal{E}_j, lab_j)$ that have a corresponding element in $\mathcal{T}$ and are adjacent to each other in the ambiguity graph $\mathcal{G} = (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G})$:

$$\mathcal{G}_j \in \mathcal{N}(\mathcal{G}_i) \Leftrightarrow \exists (v_i, v_j) \in \mathcal{V}_i \times \mathcal{V}_j \quad \text{such that} \quad (v_i, v_j) \in \mathcal{E}_\mathcal{G} \tag{10}$$

Note that when two subgraphs $\mathcal{G}_i$ and $\mathcal{G}_j$ are merged that are not in each other's neighborhoods, the merging outcome $\mu(\mathcal{G}_i, \mathcal{G}_j)$ requires $lab_i + lab_j$ labels. In the opposite case, fewer labels may be required.

The greedy k-ambiguity tree construction algorithm is presented in Algorithm 1. On Line 7, it makes use of the remainder graph $(\mathcal{V}_\Re, \mathcal{E}_\Re, lab_\Re) = \Re(\mathcal{G}', \mathcal{T})$, which has the following properties:

- It is the labeled induced subgraph with vertices $\mathcal{V}_\Re = \mathcal{V}_{\mathcal{G}'} \backslash \mathcal{V}_\text{in}$, where $\mathcal{V}_\text{in} \subset \mathcal{V}_{\mathcal{G}'}$ is the subset of vertices that already occur in subgraphs corresponding to elements of $\mathcal{T}$.
- Because it is an induced subgraph, its edge set is given by the edges in ambiguity graph $\mathcal{G}$ whose endpoints are both in the vertex set $\mathcal{V}_\Re$ [27].
- Moreover, $\Re(\mathcal{G}', \mathcal{T})$ is a complete graph, and hence, its edge set can be easily determined.

6 Solving the TARP on the *k*-ambiguity tree

As shown in Fig. 1, the TARP is solved in two steps, of which the first—i.e., the computation of the k-ambiguity tree—has been presented in the previous section. The second step, in which dynamic programming algorithms are used on the k-ambiguity tree to solve the TARP, is presented in this section.

As mentioned in Section 3, the optimal solution to the TARP corresponds to the solution to the MW-ISP-SC with respect to the ambiguity graph. To solve this problem using dynamic programming on the ambiguity tree, an algorithm similar to the method presented by [20] for determining the independence number of a graph is used. However, two major extensions are made to obtain an algorithm suitable for solving the TARP:

1. Accounting for the weight of an independent set in addition to its size
2. Ensuring feasibility with respect to constraints (8b) to (8d)

In the following, additional notations and concepts are introduced that are needed to describe the dynamic programming algorithm.

The power set of all labels available at a tree node s is denoted by $\mathcal{P}([\kappa(s)])$ and lists all $2^{\kappa(s)}$ possible combinations of labels, including the empty set, e.g.,

$$\mathcal{P}([\kappa(s)]) = \{\emptyset, \{1\}, \ldots, \{\kappa(s)\}, \{1,2\}, \ldots, \{1, \ldots, \kappa(s)\}\} \tag{11}$$

Moreover, $\mathscr{L}_s(\mathcal{V}') = \{lab_s(v) | v \in \mathcal{V}'\}$ denotes the set of labels of all vertices in $\mathcal{V}' \subseteq \mathcal{V}_s$.

The algorithm proposed hereafter utilizes the following tuple structure. At each tree node s, which corresponds to the subgraph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s, lab_s)$, we define

$$F(s) = \big(a_\emptyset, a_{\{1\}}, \ldots, a_{\{1,\ldots,\kappa(s)\}}\big) \tag{12}$$

which is a tuple of length $2^{\kappa(s)}$, where each element $a_\mathscr{L}$, with label set $\mathscr{L} \in \mathcal{P}([\kappa(s)])$, denotes the minimum weight of the MW-ISP-SC such that $\mathscr{L}_s(\mathcal{V}_\text{IS}) = \mathscr{L}$.

The corresponding most generic optimization problem for computing the data structure $F(s)$ is given in the following. Note that for consistency with the following sections, the elements of $F(s)$ are denoted by $d_\mathscr{L}$. The data structures of the children of tree node s are denoted by $F(c_1(s)) = (\ldots, a_\mathscr{L}, \ldots)$ and $F(c_2(s)) = (\ldots, b_\mathscr{L}, \ldots)$, cf. Fig. 6a. The elements are given by

$$d_\mathscr{L} = \min_{\mathcal{V} \subseteq \mathcal{V}_s} \sum_{v_i \in \mathcal{V}} \mathcal{W}(v_i) \tag{13a}$$

$$\text{s.t.} \quad \mathscr{L}_s(\mathcal{V}) = \mathscr{L} \tag{13b}$$

Algorithm 1: Greedy k-ambiguity tree construction

Input: Clique membership information, i.e., $\mathcal{C}_{lab}$
Output: $\ell_{\max}$-ambiguity tree $\mathcal{T}$

1 $\mathcal{T} \leftarrow \emptyset$; /* initialize partial tree set */
2 $\mathcal{S}(n) \leftarrow \left\{\mathcal{G}_s = (\mathcal{V}_{\mathcal{G}_s}, \mathcal{E}_{\mathcal{G}_s}, lab_{\mathcal{G}_s}) \in \mathcal{C}_{lab} \middle| |\mathcal{V}_{\mathcal{G}_s}| = n\right\}, \forall n \in \mathbb{N}$; /* given subgraphs that can be labeled with n labels */
3 $\ell_{\max} \leftarrow \min_{\mathcal{G}_s \in \mathcal{C}_{lab}} |\mathcal{V}_{\mathcal{G}_s}|$; /* upper bound on number of labels */
4 **repeat**
5 **foreach** $\mathcal{G}_s \in \mathcal{S}(\ell_{max})$ **do** /* update $\mathcal{S}(\ell_{\max})$ */
6 If there is no vertex in $\mathcal{V}_{\mathcal{G}_s}$ that is part of a subgraph whose tree is in $\mathcal{T}$, add the ambiguity tree corresponding to $\mathcal{G}_s$—which is a single leaf—to $\mathcal{T}$. Remove $\mathcal{G}_s$ from $\mathcal{S}(\ell_{\max})$ and continue the loop;
7 If some but not all vertices of $\mathcal{V}_{\mathcal{G}_s}$ are part of a subgraph corresponding to a tree in $\mathcal{T}$, add the remainder graph $\Re(\mathcal{G}_s, \mathcal{T})$ to $\mathcal{S}(\ell_{\max})$. Remove $\mathcal{G}_s$ from $\mathcal{S}(\ell_{\max})$ and continue the loop;
8 If all vertices in $\mathcal{V}_{\mathcal{G}_s}$ are part of a subgraph corresponding to a tree in $\mathcal{T}$, remove $\mathcal{G}'$ from $\mathcal{S}(\ell_{\max})$ and continue the loop;
9 **end**
10 **foreach** $(T_{\mathcal{G}_i}, T_{\mathcal{G}_j}) \in \mathcal{T} \times \mathcal{T}$ *where* $\mathcal{G}_i \in \mathcal{N}(\mathcal{G}_j)$ **do** /* merge neighboring graphs */
11 $\mathcal{G}_{\text{merge}} \leftarrow \mu(\mathcal{G}_i, \mathcal{G}_j)$;
12 If $\mathcal{G}_{\text{merge}}$ has at most $\ell_{\max}$ labels, remove $T_{\mathcal{G}_i}$ and $T_{\mathcal{G}_j}$ from $\mathcal{T}$. Add the tree $T_{\mathcal{G}_{\text{merge}}}$ corresponding to $\mathcal{G}_{\text{merge}}$ to $\mathcal{T}$;
13 **end**
14 If Line 12 has never been executed, i.e., no trees from $\mathcal{T}$ have been merged, increase the label bound $\ell_{\max}$ by one;
15 **until** $\mathcal{T}$ *contains only one element, which represents the complete ambiguity graph* $\mathcal{G}$;

where the weights $\mathcal{W}(v_i)$ are given by Eq. (7). To enable the recursive computation of $F(s)$, $d_{\mathscr{L}}$ is obtained as

$$d_{\mathscr{L}} = \min_{\mathcal{V}_1, \mathcal{V}_2} \sum_{v_i \in \mathcal{V}_1 \cup \mathcal{V}_2} \mathcal{W}(v_i) \tag{14a}$$

$$\text{s.t.} \quad \mathscr{L}_s(\mathcal{V}_1 \cup \mathcal{V}_2) = \mathscr{L} \tag{14b}$$

$$\mathcal{V}_1 \subseteq \mathcal{V}_{c_1(s)}, \mathcal{V}_2 \subseteq \mathcal{V}_{c_2(s)} \tag{14c}$$

because the vertex set corresponding to tree node s, $\mathcal{V}_s$, is the union of the vertex sets of the children of s. By the definition of the CW_k-tree (cf. Definition 2), for the objective function of Problem (14), it holds that

$$\min_{\mathcal{V}_1, \mathcal{V}_2} \sum_{v_i \in \mathcal{V}_1 \cup \mathcal{V}_2} \mathcal{W}(v_i) = \min_{\mathcal{V}_1, \mathcal{V}_2} \sum_{v_i \in \mathcal{V}_1} \mathcal{W}(v_i) + \sum_{v_i \in \mathcal{V}_2} \mathcal{W}(v_i) \tag{15}$$

and for the first constraint of Problem (14), it holds that

$$\mathscr{L}_s(\mathcal{V}_1 \cup \mathcal{V}_2) = \mathcal{R}_{c_1(s) \to s}\big(\underbrace{\mathscr{L}_{c_1(s)}(\mathcal{V}_1)}_{=:\mathscr{L}_1}\big) \cup \mathcal{R}_{c_2(s) \to s}\big(\underbrace{\mathscr{L}_{c_2(s)}(\mathcal{V}_2)}_{=:\mathscr{L}_2}\big) \tag{16}$$

where $\mathcal{R}_{c_i(s) \to s}(\mathscr{L})$ denotes the label set at tree node s of the vertices labeled with $\mathscr{L}$ at $c_i(s)$.

In the following, the notation that infeasible sets have a weight of $d_{\mathscr{L}} = \infty$ is adopted. Hence, we regard sets that satisfy $d_{\mathscr{L}} < \infty$ as solution candidates. Using formulation Eqs. (14), (15), and (16), $F(s)$ will be computed using dynamic programming in a bottom-up fashion, starting from the leaves of the ambiguity tree. The final formulation with additional constraints, which are introduced in the following, is given in Eq. 17 and referenced in Algorithm 2.

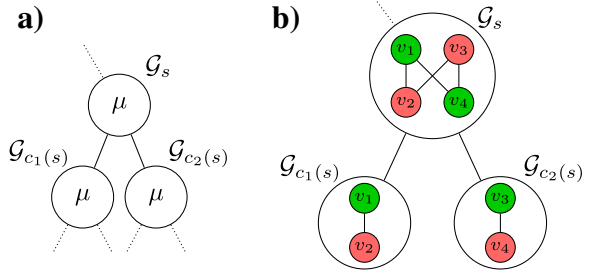


Fig. 6 Examples of k-ambiguity trees. **a** shows a part of a tree in which the tree nodes consist of the merging operator μ. This visualization is analogous to the general description of clique-width k-expression trees, in which the annotated ambiguity subgraphs $\mathcal{G}$ are often referred to as *bags*. **b** shows the same tree, with the tree nodes—for clarity—replaced by the corresponding labeled ambiguity subgraphs

Algorithm 2: TARP-Solver based on dynamic programming on a k-ambiguity tree

Input: k-ambiguity tree T of ambiguity graph $\mathcal{G}$
Output: Vertex set corresponding to the optimal solution to Problem (8)

Start bottom-up dynamic programming;
repeat
  Determine the subgraph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s, lab_s)$ corresponding to tree node s;
  if *s is a leaf of T* **then**
    Compute $F(s) = \left(a_\emptyset, a_{\{1\}}, \ldots, a_{\{1,\ldots,\kappa(s)\}}\right)$ as follows:
    if *$\mathcal{G}_s$ is of type $\mathcal{G}_{1,a}$ or $\mathcal{G}_{1,b}$, i.e., $\mathcal{G}_s \in \mathcal{C}_{1,ab}$* **then**
      $a_{\mathscr{L}} = \begin{cases} \sum_{v \in \mathcal{V}_s(\mathscr{L})} \mathcal{W}(v) & \text{if } |\mathscr{L}| = 1 \\ \infty & \text{else} \end{cases}$
    else `/* G_s is of type G_2 */`
      $a_{\mathscr{L}} = \begin{cases} \sum_{v \in \mathcal{V}_s(\mathscr{L})} \mathcal{W}(v) & \text{if } |\mathscr{L}| = 1 \\ 0 & \text{if } |\mathscr{L}| = 0 \\ \infty & \text{else} \end{cases}$
    end
  else `/* s has two children, c_1(s) and c_2(s) */`
    As $F(c_1(s)) = \left(a_\emptyset, a_{\{1\}}, \ldots, a_{\{1,\ldots,\kappa(c_1(s))\}}\right)$ and $F(c_2(s)) = \left(b_\emptyset, b_{\{1\}}, \ldots, b_{\{1,\ldots,\kappa(c_2(s))\}}\right)$ are known from previous iterations, compute all $d_{\mathscr{L}}$ of $F(s) = \left(d_\emptyset, d_{\{1\}}, \ldots, d_{\{1,\ldots,\kappa(s)\}}\right)$ according to Problem (17).
  end
until *root r of T reached*;
Based on $F(r) = (a_{\{1\}})$ at the root r of the tree, recursively obtain the label sets $\mathscr{L}_1$ and $\mathscr{L}_2$ that minimize the objective function of Problem (17). At the leaves, the mapping between the labels and vertices is unique.

6.1 Ensuring the independent set size in the k-ambiguity tree

Constraints (8b) and (8c), which determine the size of the independent set, are equivalent to the requirement that each clique in $\mathcal{C}_{1,ab}$ must have exactly one vertex in the solution to the independent set problem. The clique set $\mathcal{C}_{1,ab}$ is defined as a subset of the clique set $\mathcal{C}$, which was introduced in Section 5.2: $\mathcal{C}_{1,ab}$ consists of all cliques (complete subgraphs) that are due to ambiguity graph constraints 1(a) and 1(b), i.e., of the $\mathcal{G}_{1,a}(\cdot,\cdot)$ and $\mathcal{G}_{1,b}(\cdot,\cdot)$ types. To enable a feasibility check of the size constraint at each tree node s, let $\mathcal{C}_{1,ab}(s) \subseteq \mathcal{C}_{1,ab}$ denote those cliques whose vertices are completely included in $\mathcal{V}_s$. Now, set $a_{\mathscr{L}} = \infty$ if there is no vertex set $\mathcal{V}'$ with $\mathscr{L}_s(\mathcal{V}') = \mathscr{L}$ that contains a vertex from *every* clique in $\mathcal{C}_{1,ab}(s)$.

Example 7 *If, e.g., $R_{c_1(s)}(v) = \left((\alpha_1,\alpha_2),(\beta_1,\beta_2),(\gamma_1,\gamma_2)\right)$ with $\alpha_1,\alpha_2 > 0$ (cf. Eq. (9)) and $R_s(v) = \left((0,0),(\beta_1,\beta_2),(\gamma_1,\gamma_2)\right)$, then $\mathcal{G}_{1,a}(\alpha_1,\alpha_2)$ is in $\mathcal{C}_{1,ab}(s)$, and hence, at tree node s, exactly one vertex $v \in \mathcal{V}_{1,a}(\alpha_1,\alpha_2)$ must be selected as part of the independent set solution. Therefore, it is said that $\mathcal{G}_{1,a}(\alpha_1,\alpha_2)$ imposes a size constraint (SC) at tree node s.*

Because each non-leaf tree node s in the k-ambiguity tree is the result of the merging of its two children, i.e., $\mathcal{G}_s = \mu(\mathcal{G}_{c_1(s)}, \mathcal{G}_{c_2(s)})$, the SCs are formulated in terms of the children to enable recursive procedures. This reduces the SC clique set that needs to be checked at parent node s to those cliques that are not included in the constraint clique sets of its children $c_i(s)$: $\mathcal{C}'_{1,ab}(s) = \mathcal{C}_{1,ab}(s) \backslash \{\mathcal{C}_{1,ab}(c_1(s)), \mathcal{C}_{1,ab}(c_2(s))\}$.

The dynamic programming procedure operates on the k-ambiguity tree and, therefore, on label sets instead of vertex sets. Hence, the SCs in $\mathcal{C}'_{1,ab}(s)$ must be formulated using label sets. Recall that each label corresponds to a three-tuple of the form given in Eq. (9) and that the first two elements refer to vertex sets of the $\mathcal{V}_{1,a}(\cdot,\cdot)$ and $\mathcal{V}_{1,b}(\cdot,\cdot)$ types. If $\ell_1 \in \mathscr{L}_{c_1(s)}$ and $\ell_2 \in \mathscr{L}_{c_2(s)}$ correspond to three-tuples $R_{c_1(s)}$ and $R_{c_2(s)}$ that have equal and non-zero entries at the same position, then edges are inserted between vertices with labels of ℓ_1 and ℓ_2 by the operator μ at tree node s. Therefore, these vertices cannot be part of an independent set and are infeasible. The set of feasible label-set pairs $(\mathscr{L}_1, \mathscr{L}_2) \subset \mathcal{P}([\,c_1(s)\,]) \times \mathcal{P}([\,c_2(s)\,])$ at tree node s is denoted by $\mathscr{C}(s)$.

6.2 Dynamic-programming-based algorithm for solving the TARP

A complete algorithmic description of the dynamic programming procedure is given in Algorithm 2[12]. The algorithm is performed in a bottom-up fashion, i.e., starting from the leaves of the k-ambiguity tree. Depending on whether the current tree node is a leaf, $F(s)$ must be computed either from scratch or based on its children $c_1(s)$ and $c_2(s)$:

- In the case that s is a leaf, its corresponding subgraph $\mathcal{G}_s$ is an element of clique set $\mathcal{C}$. If $|\mathscr{L}| > 1$, $\mathscr{L}$ addresses more than one vertex, which is not feasible, as indicated by $a_{\mathscr{L}} = \infty$. The case in which $|\mathscr{L}| = 0 (\mathscr{L} = \emptyset)$ is feasible only for non-SC cliques, i.e., if $\mathcal{G}_s \in \mathcal{C}_2$, we set $a_{\mathscr{L}} = 0$.
- In the case that s has two children $c_i(s)$, $i = 1, 2$, the tuple elements of $F(s)$ are computed based on $F(c_i(s))$, $i = 1, 2$, by solving Problem (17). The label set $\mathscr{L}$ is obtained as the union of the relabeled label sets of the children, cf. constraint (17b). Constraint (17c) ensures that the SC is satisfied (cf. Section 6.1), and constraint (17d) ensures that $\mathcal{V}_s(\mathscr{L})$ is an independent set of ambiguity graph $\mathcal{G}$. Thus, $\mathcal{E}^{lab}_{c_1(s),c_2(s)}$ is the labeled variant of $\mathcal{E}_{c_1(s),c_2(s)}$, i.e., it contains the edges that are introduced during the merging of the labels of $\mathcal{V}_{c_1(s)}$ and $\mathcal{V}_{c_2(s)}$.

$$d_{\mathscr{L}} = \min_{\mathscr{L}_1,\mathscr{L}_2} a_{\mathscr{L}_1} + b_{\mathscr{L}_2} \tag{17a}$$

$$\text{s.t. } \mathscr{L} = \bigcup_{i=1,2} \mathcal{R}_{c_i(s)\to s}(\mathscr{L}_i) \tag{17b}$$

$$(\mathscr{L}_1, \mathscr{L}_2) \in \mathscr{C}(s) \tag{17c}$$

$$(\ell_1, \ell_2) \notin \mathcal{E}^{lab}_{c_1(s),c_2(s)}, \forall \ell_1 \in \mathscr{L}_1, \ell_2 \in \mathscr{L}_2 \tag{17d}$$

7 Backtracking-based dynamic programming algorithm for solving the TARP

This section introduces a new dynamic programming algorithm that employs backtracking to reduce the number of label-set combinations *tested* before the optimal solution is found. In this regard, note that in the general formulation of Problem (17), the complete data structure $F(s) = (\ldots, a_{\mathscr{L}}, \ldots)$ is computed for every tree node s. However, to solve the TARP and, hence, to compute $F(r)$ for the root r of the tree, only specific elements $a_{\mathscr{L}}$ of $F(s)$ from each tree node s are needed. Run-time reductions can be achieved by performing only on-demand computations of the necessary weight elements of the data structure $F(s)$. To this end, bounds $\beta_{c_1(s)}$ and $\beta_{c_2(s)}$ on the lengths of the data structures of the children of tree node s are introduced. The partial data structure is denoted by $F(s, \beta_s) = F(s, \beta_{c_1(s)}, \beta_{c_2(s)})$, where β_s is the bound on the cardinality of $F(s, \beta_s)$, which is controlled by the parent nodes of s. Initially, every leaf of the k-ambiguity tree is initialized with a bound of ∞, and all other tree nodes have an initial bound of 1. In general, the bounds will be increased only if additional label-set combinations $(\mathscr{L}_1, \mathscr{L}_2)$ need to be considered, i.e., if the currently considered combinations—which, by definition, have the lowest possible weight—are infeasible with respect to Problem (17). In general, it is possible to increase the bounds in an arbitrary way. In the following, we let $\beta_{c_i(s)} = f_\beta(u_{c_i(s)})$ denote the function that determines the new bound, depending on the update counter $u_{c_i(s)}$ of child $c_i(s)$. This update counter is increased by one whenever a new set of label-set combinations is requested by the parent node.

Recall that each entry $a_{\mathscr{L}}$ of $F(c_i(s)) = (\ldots, a_{\mathscr{L}}, \ldots)$ is the weight of the local MW-ISP-SC at the corresponding tree node, where the solution set only consists of vertices labeled with $\mathscr{L}$. The bounds $\beta_{c_i(s)}$, $i = 1, 2$, specify the maximum cardinality $|F(s, \beta_{c_i(s)})| \leq \beta_{c_i(s)}$ of the label sets that are considered when computing $F(s, \beta_{c_1(s)}, \beta_{c_2(s)})$ for the parent tree node s.

The following variables are used to perform the backtracking:

- a_{next} and b_{next} denote the next smallest values in $F(c_1(s))$ and $F(c_2(s))$, respectively, that are *not* currently members of $F(c_1(s), \beta_{c_1(s)})$ and $F(c_2(s), \beta_{c_2(s)})$, respectively.
- $a_{\min}$ and $b_{\min}$ denote the smallest weights in $F(c_1(s), \beta_{c_1(s)})$ and $F(c_2(s), \beta_{c_2(s)})$, respectively.
- $d_{\text{next}} = \min\{a_{\text{next}} + b_{\min}, b_{\text{next}} + a_{\min}\}$ is a lower bound on the weight of the next label combinations that are in $F(s) = F(s, \infty)$ but not in $F(s, \beta_s)$ for the current value of β_s[13]. Depending on the minimizer of d_{next}, i.e., either $a_{\text{next}} + b_{\min}$ or $b_{\text{next}} + a_{\min}$, the corresponding bound, i.e., either $\beta_{c_1(s)}$ or $\beta_{c_2(s)}$, is increased according to f_β by the corresponding update counter.
- $\mathscr{C}(s, \beta_{c_1(s)}, \beta_{c_2(s)}) \subseteq \mathscr{C}(s)$ denotes the set of label set combinations that are feasible with respect to the size constraints.

$$d_{\mathscr{L}} = \min_{\mathscr{L}_1,\mathscr{L}_2} a_{\mathscr{L}_1} + b_{\mathscr{L}_2} \tag{18a}$$

$$\text{s.t. } \mathscr{L} = \bigcup_{i=1,2} \mathcal{R}_{c_i(s)\to s}(\mathscr{L}_i) \tag{18b}$$

$$(\mathscr{L}_1, \mathscr{L}_2) \in \mathscr{C}(s, \beta_{c_1(s)}, \beta_{c_2(s)}) \tag{18c}$$

$$a_{\mathscr{L}_1} + b_{\mathscr{L}_2} \leq d_{\text{next}} \tag{18d}$$

$$(\ell_1, \ell_2) \notin \mathcal{E}^{lab}_{c_1(s),c_2(s)}, \forall \ell_1 \in \mathscr{L}_1, \ell_2 \in \mathscr{L}_2 \tag{18e}$$

See Algorithm 3 for a complete description.

8 Numerical results and discussion

Simulations are reported here to enable the comparison of three different algorithms with different parameters. These algorithms are briefly summarized in Table 2. The first algorithm is denoted by `ILP`. It solves Problem (8) directly using the commercial BARON solver [28], interfaced from MATLAB.

Table 2 Algorithm overview

Abbreviation		Parameters	Description
`ILP`		—	Solves Problem (8) using the branch-and-bound-type solver BARON.
`Tree`	`Dec.+Rec`	$a_\beta = \infty$	Non-backtracking version. Uses Algorithms 1 and 2.
	`Dec.+BT`	$a_\beta \in (0,\infty), f_\beta(x)=a_\beta 1.25^{x-1}$	Backtracking version. Uses Algorithms 1 and 3.
	`Dec.+BT*`	—	*Genie-aided* version of the backtracking algorithm, where the optimal bounds on the data structures are known for each tree node. Uses Algorithms 1 and 3.

Algorithm 3: Backtracking-based TARP-Solver using dynamic programming on a k-ambiguity tree

Input: k-ambiguity tree T of ambiguity graph $\mathcal{G}$, SC cliques $\mathscr{C}(s)$ for all tree nodes s

Output: Vertex set corresponding to the solution to Problem (8)

1 Let r denote the root of T;
2 Set $u_s \leftarrow 0$ for all tree nodes s; `/* initialize all update counters */`
3 $(d_{\text{next}}, F(r, \beta_{c_r(s)})) \leftarrow$ Algorithm 4 $\left(r, \beta_{c_r(s)} = 1, \mathscr{C}\right)$;
4 Based on $F(r) = (a_{\{1\}})$ at the root r of the tree, recursively obtain the label sets $\mathscr{L}_1$ and $\mathscr{L}_2$ that minimize the objective function of Problem (17). At the leaves, the mapping between the labels and vertices is unique.

The algorithms proposed in this work are denoted by `Tree: Dec. + BT` and `Tree: Dec. + DP` to highlight the fact that these methods consist of two parts, namely, tree decomposition (Algorithm 1) and tree recursion using dynamic programming (Algorithms 2 or 3). In the case of the backtracking (BT) algorithm, the exponential bound function $f^*_\beta(x) = a_\beta * 1.25^{x-1}$ is used.

To enable the analysis of the theoretical performance limits of the backtracking version of the proposed algorithm, a method denoted by `Tree: Dec. + BT*` is additionally evaluated. This method differs from `Tree: Dec. + BT` only in the dynamic programming (DP) part: here, it is assumed that a genie-aided bound function $f^*_\beta(\cdot)$ is already used in the first step to specify the optimal bounds on the data structure $F(s, \beta_s)$. Note that in this way, the overhead resulting from the computation of too many or too few data structure elements is eliminated.

Table 2 gives a brief summary of the algorithms and the abbreviations used.

8.1 Simulation setup

The simulations were performed by randomly placing N agents in the two-dimensional unit box $[0,1]^2$ following a uniform distribution. This setup is widely adopted in the literature and can be found, e.g., in [7, 29, 30] because it facilitates scaling of the obtained results. To ensure that the IDs were assigned to the agents as equally as possible, the following procedure was used: First, $\lfloor \frac{N}{I} \rfloor$ instances of each ID were placed in the *ID pool*, where I is the number of IDs. Then, one additional ID instance from among the first $N - I\lfloor \frac{N}{I} \rfloor$ IDs was placed in the pool. Finally, the IDs in the pool were randomly assigned to the agents. Because the complexity of the algorithms increases with increasing connectivity among the agents, a communication range of $R = \sqrt{2}$ was chosen for all agents in some scenarios to demonstrate the algorithms' limits in such cases. The additive noise on the initial position estimate was assumed to have a standard deviation of $\sigma_p = \sigma_{p_i} = 0.1, \forall i$. All figures show averages over 100 simulations.

8.2 Simulation results

Because all presented algorithms are optimal in the MAP sense, only run-time comparisons are given in the following. Figure 7 shows a comparison between the `ILP` algorithm and the non-backtracking algorithm in terms of the total simulation time for a fixed sensing radius of $R = \sqrt{2}$. It can be seen that for lower code-reuse ratios ($N \to 30$ for Fig. 7a and $N \to 60$ for Fig. 7b), the run times are approximately equal, although slightly worse for the dynamic-programming-based algorithm. This is mainly because the tree decomposition requires most of the computing time and the benefits of the dynamic programming cannot show to advantage in this case. Although the code-to-sensor-mote ratio is the same in both figures, the absolute run time of the `ILP` algorithm increases from approximately 356 s ($N = 40$) in the case of 10 distinct codes to 6630 s ($N = 80$) in the case of 20 codes. Similarly, for the proposed algorithm without backtracking, the run time increases from approximately 168 s ($N = 40$) to 775 s ($N = 80$). The maximum run-time reductions achieved by the proposed algorithm are 52.8% ($N = 40$) and 88.3% ($N = 80$), respectively.

Figure 8 shows complementary results for the case wherein the sensing radius R is varied and the code-to-sensor-mote ratio is fixed to 10:40 and 20:80. In addition, in this scenario, it can be observed that the run-time complexity of the algorithms is approximately the same when the problem size is small, i.e., when the connectivity and thus the communication range are low ($R \to 0$). However, with increasing connectivity, the complexity of the `ILP` algorithm becomes significantly higher compared to

Algorithm 4: Backtracking function for Algorithm 3

Input: The currently considered tree node s and the corresponding bound β_s, SC cliques $\mathscr{C}(s)$ at s

Output: d_{next} and data structure $F(s, \beta_s)$

Determine $F(s)$ for the subgraph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s, lab_s)$ corresponding to tree node s;

if *s is a leaf of T* **then** /* For leaves, no bound β_s is imposed */

  Compute $F(s, \beta_s = \infty) = F(s) = \left(a_\emptyset, a_{\{1\}}, \ldots, a_{\{1,\ldots,\kappa(s)\}}\right)$ as follows:

  if *$\mathcal{G}_s$ is of type $\mathcal{G}_{1,a}$ or $\mathcal{G}_{1,b}$, i.e., $\mathcal{G}_s \in \mathcal{C}_{1,ab}$* **then**

    $a_{\mathscr{L}} = \begin{cases} \sum_{v \in \mathcal{V}_s(\mathscr{L})} \mathcal{W}(v) & \text{if } |\mathscr{L}| = 1 \\ \infty & \text{else} \end{cases}$

  else// $\mathcal{G}_s$ is of type $\mathcal{G}_2$

    $a_{\mathscr{L}} = \begin{cases} \sum_{v \in \mathcal{V}_s(\mathscr{L})} \mathcal{W}(v) & \text{if } |\mathscr{L}| = 1 \\ 0 & \text{if } |\mathscr{L}| = 0 \\ \infty & \text{else} \end{cases}$

  end

  Set $d_{\text{next}} \leftarrow \infty$, which indicates that $F(s, \beta_s) = F(s)$;

else /* s has two children $c_1(s)$ and $c_2(s)$ with bounds $\beta_{c_1(s)}$ and $\beta_{c_2(s)}$ */

  if $u_s = 0$ **then**

    // Note: $u_{c_1(s)} = u_{c_2(s)} = 0$ here

    $\left(a_{\text{next}}, F(c_1(s), \beta_{c_1(s)})\right) \leftarrow$ Algorithm 4 $\left(c_1(s), f_{\beta_{c_1(s)}}(u_{c_1(s)}), \mathscr{C}(s)\right)$;

    $\left(b_{\text{next}}, F(c_2(s), \beta_{c_2(s)})\right) \leftarrow$ Algorithm 4 $\left(c_2(s), f_{\beta_{c_2(s)}}(u_{c_2(s)}), \mathscr{C}(s)\right)$;

  end

  Set $a_{\min} \leftarrow \min_i \left[F(c_1(s), \beta_{c_1(s)})\right]_i$;

  Set $b_{\min} \leftarrow \min_i \left[F(c_2(s), \beta_{c_2(s)})\right]_i$;

  Determine $\mathscr{C}(s, \beta_{c_1(s)}, \beta_{c_2(s)})$;

  Compute $F(s, \beta_{c_1(s)}, \beta_{c_2(s)}) = \left(d_\emptyset, d_{\{1\}}, \ldots, d_{\{1,\ldots,\kappa(s)\}}\right)$ using $d_{\mathscr{L}}$ according to Problem (18);

  if $a_{next} + b_{min} < b_{next} + a_{min}$ **then**

    Increase update counter $u_{c_1(s)} \leftarrow u_{c_1(s)} + 1$;

    $\left(a_{\text{next}}, F(c_1(s), \beta_{c_1(s)})\right) \leftarrow$ Algorithm 4 $\left(c_1(s), f_{\beta_{c_1(s)}}(u_{c_i(s)}), \mathscr{C}(s)\right)$;

  else

    Increase update counter $u_{c_2(s)} \leftarrow u_{c_2(s)} + 1$;

    $\left(b_{\text{next}}, F(c_2(s), \beta_{c_2(s)})\right) \leftarrow$ Algorithm 4 $\left(c_2(s), f_{\beta_{c_2(s)}}(u_{c_2(s)}), \mathscr{C}(s)\right)$;

  end

  if *any in* $F(s, \beta_{c_1(s)}, \beta_{c_2(s)}) > \min\{a_{next} + b_{min}, b_{next} + a_{min}\}$ **then**

    Remove elements from $F(s, \beta_{c_1(s)}, \beta_{c_2(s)})$ that are larger than d_{next};

  end

  if $|F(s, \beta_{c_1(s)}, \beta_{c_2(s)})| > \beta_s$ **then** /* too many elements computed */

    Assign to $F(s, \beta_s)$ the smallest β_s values of $F(s, \beta_{c_1(s)}, \beta_{c_2(s)})$;

    $d_{\text{next}} \leftarrow (\beta_s+1)$-th smallest value of $F(s, \beta_{c_1(s)}, \beta_{c_2(s)})$;

  else /* too few or exact number of elements computed */

    $F(s, \beta_s) \leftarrow F(s, \beta_{c_1(s)}, \beta_{c_2(s)})$;

    $d_{\text{next}} \leftarrow \min\{a_{\text{next}} + b_{\min}, b_{\text{next}} + a_{\min}\}$;

    /* Note that if all elements are computed, i.e., if $F(c_i(s), \beta_{c_i(s)}) = F(c_i(s))$ for both $i = 1, 2$, then $a_{\text{next}} = b_{\text{next}} = \infty$ and, therefore, $d_{\text{next}} = \infty$ as well. */

  end

end

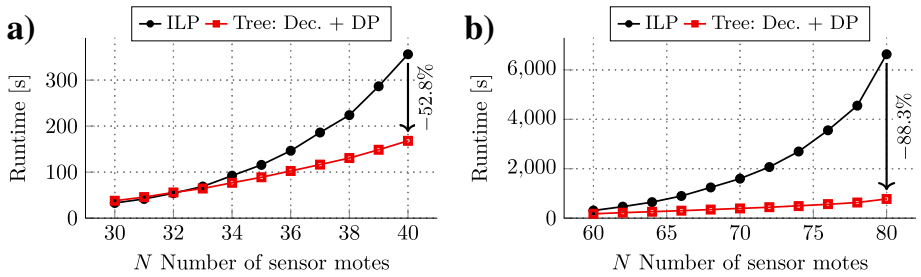


Fig. 7 Figures **a** and **b** show the total run times for 10 and 20 distinct DS-CDMA codes, respectively. Moreover, the measurement noise and sensing range has been set to $\sigma = 0.05$ and $R = \sqrt{2}$, respectively

the proposed algorithm. For Fig. 8a, the reduction in run time for $R = 1.3$ is from 246 *s* (ILP) to 128 *s* (Tree: Dec. + BT). Correspondingly for Fig. 8b, the reduction is from 2983 *s* (ILP) to 576 *s* (Tree: Dec. + BT). The increase in run time with larger sensing radius R is mainly due to the increased connectivity and the consequently larger quantity of ambiguous measurements that need to considered and mapped.

Simulation results for variable measurement noise are given in Fig. 9, where the code-to-sensor-mote ratio is fixed to 10:40 and 20:80, and the sensing radius has been fixed to $R = 0.2$. Consequently, this case can be considered as having low connectivity or low agent density. Due to the low connectivity, the overall run time is low and its variability high. Nevertheless, it can be observed that the `ILP` algorithm takes roughly twice as long as the proposed algorithm. Moreover, the run time varies only slightly with the measurement noise.

A comparison among the tree-decomposition-based algorithms is shown in Fig. 10. Here, the run-time analysis is restricted to the run time of the dynamic programming part because all methods share the same tree-decomposition algorithm. Figure 10a and b again present the results for 10 and 20 distinct DS-CDMA codes, respectively, with the same code-to-sensor-mote ratio. It can be seen that the backtracking procedure proposed in this work decreases the run time of the dynamic programming part by up to 45.8% ($N = 40$) in the case of 10 codes and 48% ($N = 80$) in the case of 20 codes. Note that these reductions were achieved using the simple function $f^*_\beta(x) = 700 \times 1.25^{x-1}$.

The theoretical minimum run time, as simulated using the genie-aided bound function, leads to additional reductions of 66.3% ($N = 40$) and 57.7% ($N = 80$). For future work, we are interested in conducting further research on how to effectively benefit from these theoretical values for the practically relevant Tree: Dec. + BT algorithm. A brief analysis of important properties that are relevant for the implementation of this algorithm is given in Section 8.2.1.

To highlight the significance of the complexity reductions achieved using the newly proposed algorithms, the run-time trends for a fixed sensor-to-ID ratio of four to one are shown in Fig. 11. Together with the results in Fig. 7, this figure suggests that the proposed algorithms should provide a good basis on which to derive sub-optimal algorithms to solve this $\mathcal{NP}$-hard problem even more efficiently.

8.2.1 *Analysis of tree and backtracking parameters*

Figure 12 shows the impact of the number of sensor motes in a scenario with a fixed number of IDs. The figure visualizes the relative frequency of the minimum num-

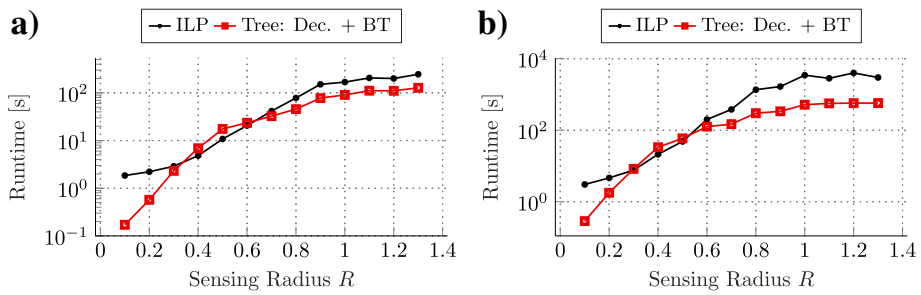


Fig. 8 Figures **a** and **b** show the total run times for 10 DS-CDMA codes in the case of $N = 40$ agents and 20 CD-CDMA codes for $N = 80$ agents. Moreover, the measurement noise has been set to $\sigma = 0.05$

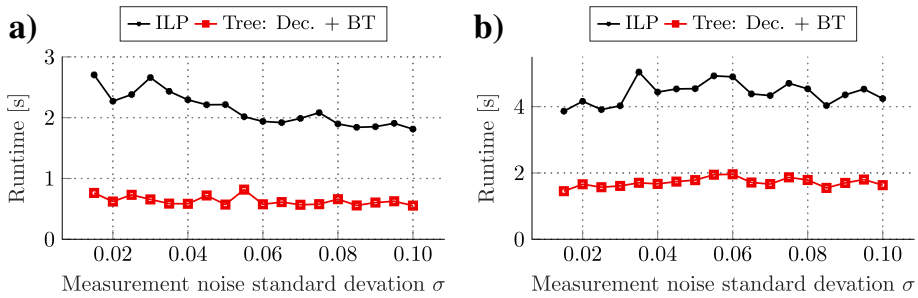


Fig. 9 Figures **a** and **b** show the total run times for 10 DS-CDMA codes in the case of $N = 40$ agents and 20 CD-CDMA codes for $N = 80$ agents. Moreover, the sensing range has been set to $R = 0.2$

ber of label-set combinations that need to be computed at each tree node to find the optimal solution. For 30 sensor motes, more than 10 combinations need to be computed for only 8.4% of the tree nodes, whereas this percentage rises to 28.6 and 46.2% for $N = 35$ and $N = 40$, respectively. Thus, when more agents are present in the network, a larger proportion of the tree nodes require the computation of many label-set combinations to find the optimal solution. However, it is not possible to describe the dependency between these cases using only the number of agents as a parameter. Consequently, a more optimal bound function $f^*_\beta(\cdot)$ should not depend solely on general system parameters such as the number of sensor motes; it should also capture the individual properties of the tree node to which this function is applied. These properties could include, e.g., the level of the currently considered tree node or the number of vertices at this node.

Hence, additional run-time reductions could be achieved by applying tree-node-specific bound functions rather than a global bound function. In future work, we intend to conduct further research on this topic and to investigate potential use cases of machine learning to push the performance of the backtracking algorithm farther toward its theoretically achievable limit.

Moreover, given the complexity as well as the size of the search space of the problem, meta-heuristic optimization techniques, such as evolutionary algorithms, are also promising candidates to solve the TARP efficiently.

8.2.2 *Impact of transmit ambiguities on localization*

Although this work focuses on the resolution of the transmit ambiguities and efficient algorithms for doing so, a brief illustration of the impact of transmit ambiguities on localization is given in the following. Figure 13 presents a visual comparison of the degradation in localization performance caused by ID reuse. Black diamonds indicate the anchor node positions, red crosses indicate the estimated sensor mote positions, and green circles indicate the actual sensor mote positions. The blue ellipses denote the 1-standard-deviation regions obtained from 100 simulations of the same true sensor mote positions but with different measurement noise realizations. Figure 13a shows the localization results in the absence of transmit ambiguities for 40 sensor motes. The localization is very stable with respect to measurement noise, and only the outermost sensor motes have visible standard-deviation ellipses. Figure 13b shows the corresponding results for the case in which only 10 DS-CDMA codes are available. Here, the different measurement noise realizations

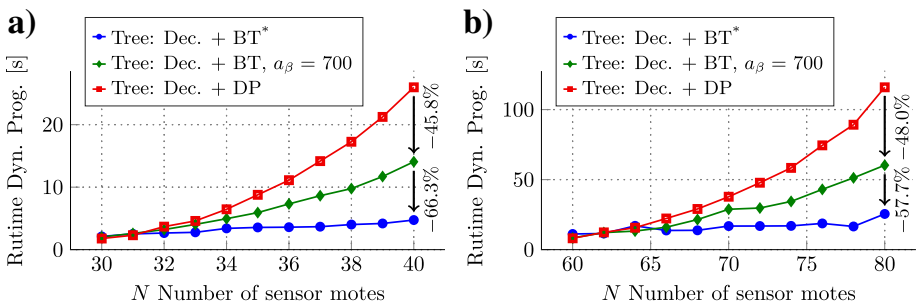


Fig. 10 Figures **a** and **b** show the run times of the dynamic programming part for 10 and 20 distinct DS-CDMA codes, respectively. The measurement noise and sensing range have been set to $\sigma = 0.05$ and $R = \sqrt{2}$, respectively

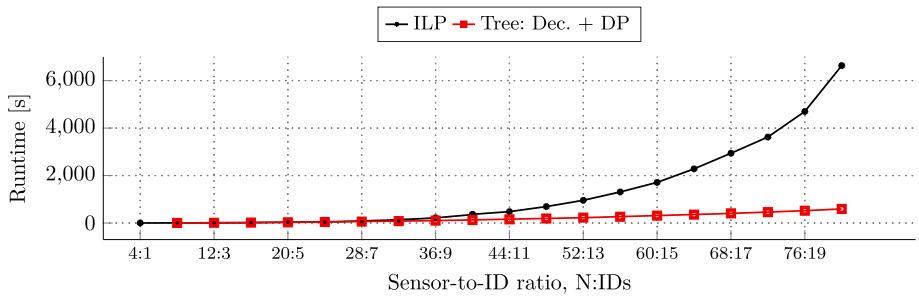


Fig. 11 Run-time simulation for a fixed sensor-to-ID ratio of 4 : 1, starting with 4 agents and 1 ID and ending with 80 agents and 20 IDs. The measurement noise has been set to $\sigma = 0.05$

lead to different solutions for the resolution of the transmit ambiguities and, thus, to larger standard-deviation ellipses.

9 Conclusion

In this work, we considered application cases in which unique agent identification is not possible but where localization of the sensor motes is nevertheless required. To address this problem, we proposed an MAP-optimal solution approach that restores the ability to perform localization using well-known and established algorithms. The proposed approach is highly tailored to the considered class of problem instances and, in our MATLAB implementation, shows a speed-up factor of up to 8.55 compared with a commercial ILP solver. The obtained run-time reductions become increasingly relevant as more agents are deployed in the system.

More precisely, we developed an algorithm based on graph decomposition and dynamic programming, and we further extended this algorithm with a backtracking scheme, which achieved additional run-time reductions for the dynamic programming part of up to 48%. Moreover, we investigated the theoretical performance limits of the proposed backtracking scheme, which revealed the potential for additional run-time reduction. Consequently, the proposed framework facilitates efficient solutions to the TARP and offers potential for further enhancements.

9.1 Outlook and future work

In our future work, we are particularly interested in finding more generic and more optimal methodologies to close the gap with respect to the theoretically achievable performance; to this end, we intend to investigate, e.g., machine-learning-based approaches. In addition to improving the optimal algorithms, we also intend to use the presented framework to derive sub-optimal algorithms. Further interest lies in the investigation of extensions to the presented approach to mobile agents.

Endnotes

[1] In this work, we use the terms *sensor mote*, *sensor*, and *agent* interchangeably where the context permits.

[2] The limiting factors for FDMA are the high transmission frequency that is required because of the small agent size and the limited bandwidth of the corresponding ultrasonic transducers.

[3] For example, Walsh-Hadamard codes, which are commonly used for DS-CDMA, have a code length of 2^n bits for separating 2^n users in the network [32]. Recall that the

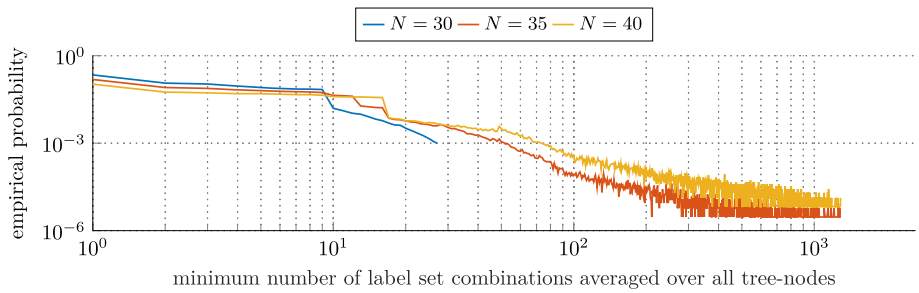


Fig. 12 Statistics of the *k*-ambiguity trees for 10 IDs and a varying number of sensor motes *N*

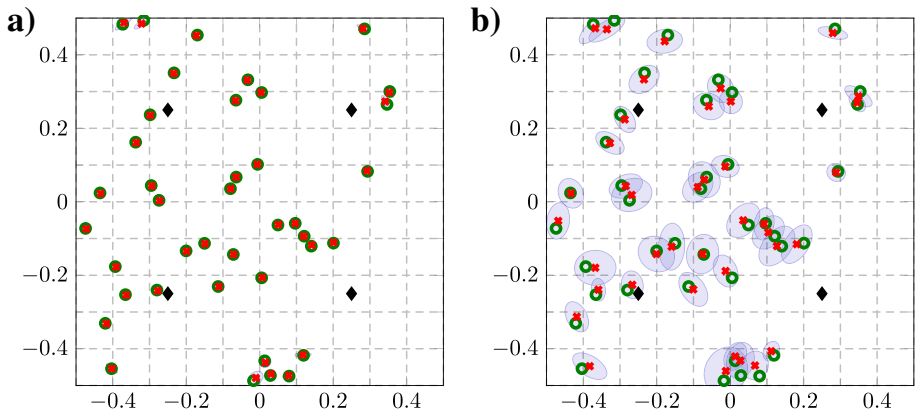


Fig. 13 Visualization of localization performance for 40 sensor motes, where **a** represents the case of unique identification (40 IDs) and **b** represents the case of only 10 IDs. These localization results were obtained by first resolving the transmit ambiguities and then applying the least-squares approximation presented in [11], where the measurement noise has been set to $\sigma = 0.05$

use of several tens of thousands of sensors is required for the considered application case [1].

[4] The authors of [1] also briefly mention that because of the limited available energy, non-unique identification of the sensors plays a key role in enabling such application cases.

[5] Note that the lth measurement denotes the lth entry in set $\mathcal{M}_i$ and that the value of l does not imply a chronological order.

[6] Note that each measurement belongs to exactly one set of type $\mathcal{V}_{1,a}(\cdot,\cdot)$ and one of type $\mathcal{V}_{1,b}(\cdot,\cdot)$, as each vertex corresponds to two distance measurements. Because the solution to the TARP is given in the form of vertices (pairs of measurements) and because every measurement must be mapped exactly once, the size of the independent set corresponds to the number of sets of type $\mathcal{E}_{1,a}(\cdot,\cdot)$, which is equal to the number of sets of type $\mathcal{E}_{1,b}(\cdot,\cdot)$.

[7] Note that items 1(a), 1(b), and 2, as listed in Section 3, introduce edges among specified sets of vertices. These vertices form the cliques.

[8] In the following, the term *tree node* is used to refer to a vertex of the tree, whereas a *subgraph* is the graph that is induced by the edge and vertex sets at a given tree node.

[9] An induced subgraph consists of a subset of the vertex set of the graph and all edges whose endpoints are both in the vertex set [27].

[10] For simplicity, and with a slight misuse of common terminology, we also refer to single vertices as cliques.

[11] This becomes apparent when one notices that the CW_k-tree needs to reflect an increasing number of the properties of the original graph as the considered tree node becomes closer to the root of the tree. At the root of the tree, the original graph must be obtained. Ensuring that the subgraphs are induced at each tree node is an easy way to satisfy this requirement.

[12] Note that this algorithm is easily parallelizable by virtue of the tree structure.

[13] Assume, e.g., $d_{\text{next}} = a_{\text{next}} + b_{\min}$, and let $\mathscr{L}_{c_1(s),\text{next}}$ and $\mathscr{L}_{c_2(s),\min}$ denote the label sets corresponding to a_{next} and $b_{\min}$, respectively. If $\mathscr{L}_{c_1(s),\text{next}}$ and $\mathscr{L}_{c_2(s),\min}$ are feasible with respect to Problem (18) at tree node s, then the bound d_{next} is tight.

Appendix 1

Detailed example: from cliques to backtracking

In this appendix, we present a detailed example to illustrate the discussed algorithms, including the following:

1. Input: An exemplary problem instance that is specified by cliques and the vertices' clique memberships.
2. k-ambiguity tree construction: For a specific choice of cliques as leaves of the k-ambiguity tree, the computation of the labels, and additionally introduced edges.
3. Backtracking: An exemplary computation of a data structure element by the backtracking algorithm.

The problem input

The cliques and the vertices' clique memberships are given in Fig. 14 and Table 3, respectively. For brevity, we restrict ourselves to only one branch of the complete tree, which contains the vertices listed in Table 3.

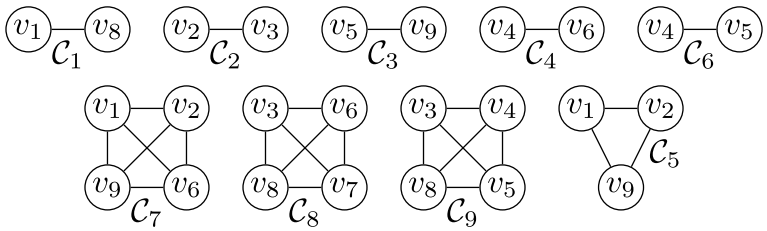


Fig. 14 Example input: cliques $\mathcal{C}_i$, $i = 1, \ldots, 9$. Note that these cliques are the actual input to the algorithm and do not need to be obtained beforehand

k-ambiguity tree construction

As mentioned in Section 5.2, cliques are chosen as leaves of the tree to minimize the number of labels. A realization of the k-ambiguity tree is presented in Fig. 15. Note that at every tree node, different labels (colors) are used to identify the vertices.

The edge sets and vertex sets of the subgraphs $\mathcal{G}_i$, $i = 1, \ldots, 4$, are given directly by the choice of the tree leaves. For clarity, their labeling and the edge, vertex, and label sets of the other subgraphs are computed in this example using the three-tuples $R_s(v)$ (cf. Eq. (9)). All three-tuples that are needed for the computation are given in Table 4.

In Table 4, a slight change in notation is adopted for simplicity: The lengthy but descriptive notation of Eq. (9) is replaced with $R_s(v) = (\mathcal{C}_i, \mathcal{C}_j, \mathcal{C}_k)$ if the cliques $\mathcal{C}_i$, $\mathcal{C}_j$, and $\mathcal{C}_k$ are not fully included at subgraph $\mathcal{G}_s$ of tree node s. Likewise, if $\mathcal{G}_s$ includes $\mathcal{C}_i$, which is a clique of type $G_{1,a}(\cdot, \cdot)$ because it appears in the first position in the three-tuple, then $R_s(v) = (0, \mathcal{C}_j, \mathcal{C}_k)$ and so forth for the other tuple elements and cliques.

Recall that two vertices v_i and v_j are connected by an edge iff there is at least one pair of equal and non-zero entries in the same position in the three-tuples $R_{c_1(s)}(v_i)$ and $R_{c_2(s)}(v_j)$. This relationship is denoted by $R_{c_1(s)}(v_i) \sim R_{c_2(s)}(v_j)$. Moreover, recall that two vertices v_i and v_j will have the same label at tree node s iff the three-tuples of the vertices are equal, i.e., iff $R_s(v_i) = R_s(v_i)$. In this table, $\ell(v)$ denotes the label of vertex v and the obtained edges and labels refer to Fig. 15, which shows the result up through $\mathcal{G}_7$, which is not the root of the tree.

Table 3 Vertices' clique memberships for the example in Fig. 14

	v_1	v_2	v_3	v_4	v_5	v_6	v_8	v_9
$\mathcal{G}_{1,a}$	$\mathcal{C}_7$	$\mathcal{C}_7$	$\mathcal{C}_8$	$\mathcal{C}_8$	$\mathcal{C}_8$	$\mathcal{C}_7$	$\mathcal{C}_8$	$\mathcal{C}_7$
$\mathcal{G}_{1,b}$	$\mathcal{C}_5$	$\mathcal{C}_5$	$\mathcal{C}_9$	$\mathcal{C}_6$	$\mathcal{C}_6$	$\mathcal{C}_9$	$\mathcal{C}_9$	$\mathcal{C}_5$
$\mathcal{G}_2$	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_2$	$\mathcal{C}_4$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_1$	$\mathcal{C}_3$

Remark: Because the example considered in Appendix 1 focuses on only a part of the complete k-ambiguity tree in which v_7 is irrelevant, the clique memberships of v_7 are not given above

Dynamic programming: weight computation via backtracking

In this section, an exemplary computation of an element of the data structure $F(s)$ for $\mathcal{G}_7$ is presented. Figure 16 shows the details of the k-ambiguity tree that are needed for this example. The figure shows the vertex weights that are assigned to the vertices. The tables on the right-hand side visualize the mapping of the label sets with corresponding colors. Because the weight computation proceeds in a bottom-up fashion, the data structures of the children of $\mathcal{G}_7$ are also given in the tables. Note that in the actual algorithm, the complete data structures will not be known, as they are computed only on demand in the backtracking approach.

For brevity, only the computation of $d_{\mathscr{L}}$ (with $\mathscr{L} = \{\ell_3^1, \ell_3^2\}$) using the backtracking algorithm is summarized below.

- Assumptions: The procedure starts from $\beta_s = \beta_{c_1(s)(s)} = \beta_{c_2(s)(s)} = 1$. Hence, the following partial data structures are known (cf. lines 12 and 13 in Algorithm 4):
  $F(c_1(s), 1) = (1), a_{\text{next}} = 3, F(c_2(s), 1) = (2), b_{\text{next}} = 4.5$
- Constraints:
  - Active SC cliques at $\mathcal{G}_7$: $\mathcal{C}_5$, $\mathcal{C}_7$, and $\mathcal{C}_9$
  - The SC cliques are given by $\mathcal{V}(\mathcal{C}_5) = \{1, 2\}$, $\mathcal{V}(\mathcal{C}_7) = \{1, 2, 6, 9\}$, and $\mathcal{V}(\mathcal{C}_9) = \{3, 4, 5, 8\}$ and by the label sets $\{\ell_1^1\}$, $\{\ell_1^1, \ell_2^2, \ell_2^3\}$, and $\{\ell_1^2, \ell_2^1\}$, respectively
  - The only label set that satisfies all SCs is given by $\mathscr{C}(7) = \{(\ell_1^1, \ell_2^1)\}$, as it contains at least one vertex from each of the three cliques
  - $\mathcal{E}^{lab}_{c_1(s), c_2(s)} = \{(\ell_1^1, \ell_2^2), (\ell_1^1, \ell_2^3), (\ell_2^2, \ell_2^3), (\ell_2^1, \ell_1^2)\}$, cf. Table 4, bottom right cell
- Task: Computation of $d_{\{\ell_3^1, \ell_3^2\}}$ according to lines 12 and 13
  - Line 15: $a_{\min} = \min\{1\} = 1$

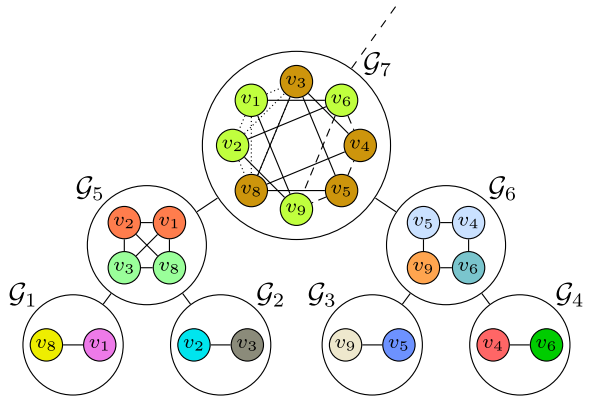


Fig. 15 *k*-ambiguity tree corresponding to the cliques shown in Fig. 14. Note that the tree nodes shown here do not correspond to the complete *k*-ambiguity tree; the root *r* of the tree is not shown

- Line 16: $b_{\min} = \min\{2\} = 2$
- $d_{\text{next}} = \min\{a_{\text{next}} + b_{\min}, b_{\text{next}} + a_{\min}\} = 5$
- Line 17: $d_{\{\ell_3^1,\ell_3^2\}} = a_{\ell_1^1} + b_{\ell_2^1} = 3$ is feasible because
  - ℓ_1^1 and ℓ_2^1 clearly yield label set $\{\ell_3^1, \ell_3^2\}$ at $\mathcal{G}_7$
  - $(\ell_1^1, \ell_2^1) \in \mathscr{C}(7)$
  - $a_{\ell_1^1} + b_{\ell_2^1} < 5$
  - $(\ell_1^1, \ell_2^1) \notin \mathcal{E}^{lab}_{c_1(s),c_2(s)}$

Note that by virtue of the backtracking procedure, it becomes unnecessary to compute the other 20 combinations that yield $\mathscr{L} = \{\ell_3^1, \ell_3^2\}$, which all have higher weights.

Appendix 2

MAP-optimal PDF and vertex weights

To derive the MAP-optimal PDF for the transmission model defined in Eq. (1), which is used to calculate the vertex weights according to Eq. (7), we make the following assumptions.

1. An initial position estimate $\mathbf{p}_{0,i}$ is given for each sensor i
2. $\mathbf{p}_{0,i}$, $\forall i$, is normally distributed around the sensor's true position, i.e., $\mathbf{p}_{0,i} = \mathbf{p}_i + \mathbf{n}_i$, $\mathbf{n}_i \sim \mathcal{N}\left(\mathbf{0}, \sigma_{p_i}^2 \mathbf{I}_2\right)$, with $\sigma_{p_i} = \sigma_p, \forall i$

Note that these assumptions are common, particularly for Kalman-filter-based tracking algorithms, for which the conditional PDF $f_{\mathbf{p}_i|\mathbf{p}_{i,0}}(x)$ is often assumed to be Gaussian.

Thus, for the conditional RV $d_{j\to i}$, we find

$$\begin{aligned}(d_{j\to i} | \{\mathbf{p}_{0,k}\}_k) &\sim \|(\mathbf{p}_{0,i}+\mathbf{n}_{p_{0,i}}) - (\mathbf{p}_{0,j}+\mathbf{n}_{p_{0,j}})\|_2 \cdot (1+n_i) \\ &\sim \|\Delta_{ij} + \mathbf{m}_{p_{ij}}\|_2 (1+n_i)\end{aligned}$$

where $\Delta_{ij} = \mathbf{p}_{0,i} - \mathbf{p}_{0,j}$ is the distance vector between the positions of sensors i and j and $\mathbf{m}_{p_{ij}} \sim \mathcal{N}\left(\mathbf{0}, 2\sigma_p^2 \mathbf{I}_2\right)$ is the effective noise [31]. Furthermore, note that

$$\left(\frac{\|\Delta_{ij} + \mathbf{m}_{p_{ij}}\|_2}{\sqrt{2}\sigma_p} \middle| \{\mathbf{p}_{0,k}\}_k\right) \sim \chi_2(\lambda) \tag{19}$$

$$(1+n_i) \sim \mathcal{N}(1, \sigma^2) \tag{20}$$

where $\chi_2(\lambda)$ is the non-central chi distribution with two degrees of freedom and a non-centrality parameter of $\lambda = d_{ij}/\sqrt{2}\sigma_p$. Therefore, we can calculate the conditional PDF of $d_{j\to i}|\{\mathbf{p}_{0,k}\}_k$ as

$$f_{d_{j\to i}|\{\mathbf{p}_{0,k}\}_k}(z) = \int_{\mathbb{R}} \sqrt{2}\sigma_p f_{\chi|2,\lambda}(\sqrt{2}\sigma_p x) \cdot f_{\mathcal{N}|0,\sigma^2}(z/x) \frac{1}{|x|} dx \tag{21}$$

where $f_{\chi|2,\lambda}(x)$ denotes the PDF of the non-central chi distribution with two degrees of freedom and a non-centrality parameter of λ and $f_{\mathcal{N}|0,\sigma^2}(x)$ denotes the normal distribution with zero mean and a variance of σ^2.

Assuming the stochastic independence of $d_{j\to i}$ and $d_{i\to j}$, we obtain the desired PDF as follows:

$$\begin{aligned}f_{d_{j\to i},d_{i\to j}|\{\mathbf{p}_{0,k}\}_k}(u,v) &= f_{d_{j\to i}|\{\mathbf{p}_{0,k}\}_k}(u) \cdot f_{d_{i\to j}|\{\mathbf{p}_{0,k}\}_k}(v) \\ &= 2\sigma_p^2 \int_{\mathbb{R}} f_{\chi|2,\lambda}(\sqrt{2}\sigma_p x) f_{\mathcal{N}|0,\sigma^2}(u/x) \frac{1}{|x|} dx \cdot \\ &\quad \int_{\mathbb{R}} f_{\chi|2,\lambda}(\sqrt{2}\sigma_p x) f_{\mathcal{N}|0,\sigma^2}(v/x) \frac{1}{|x|} dx\end{aligned} \tag{22}$$

Nomenclature

sensor i	A sensor with a unique serial number (SN) i
$I_{\mathcal{J}}$	The Jth non-unique transmission identifier (ID)
$d_{j,i}$	The actual distance between sensors j and i
$d_{j\to i}$	A random variable (RV) denoting the distance measured by sensor i based on a signal from sensor j
$\mathcal{D}_i$	A set of RVs $d_{\bullet\to i}$ representing range measurements at sensor i, where $\bullet$ serves as a placeholder
m_i^k	The kth measurement at sensor i (realization of $d_{j\to i}$ for some unknown sensor j)
$\mathcal{M}_i$	A set of measurements $m_i^\bullet$ (a subset of the realizations of RV $\mathcal{D}_i$) at sensor i
$v_{i,j}(l,r)$	A vertex in the ambiguity graph that refers to a measurement tuple $\left(m_i^l, m_j^r\right)$ that may have been measured between agents i and j
$\mathcal{E}_{1,a}, \mathcal{E}_{1,b}, \mathcal{E}_2$	Components of the edge set of the ambiguity graph, cf. Eqs. (5a), (5b), and (5c)

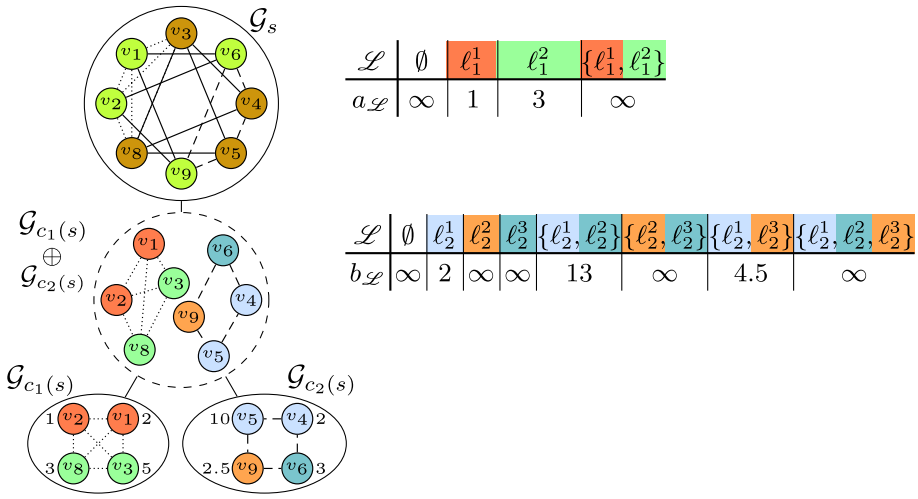
Table 4 Three-tuples used to determine the edges and labels for the example in Fig. 14

$\mathcal{G}_s$	Constraint cliques	Three-tuples of the vertices of $\mathcal{G}_i$	Action/comment
$\mathcal{G}_1$	$\mathcal{C}_1$	$R_1(v_1) = (\mathcal{C}_7, \mathcal{C}_5, 0)$	• $R_1(v_1) \neq R_1(v_8) \Rightarrow \ell(v_1) \neq \ell(v_8)^\dagger$
		$R_1(v_8) = (\mathcal{C}_8, \mathcal{C}_9, 0)$	
$\mathcal{G}_2$	$\mathcal{C}_2$	$R_2(v_2) = (\mathcal{C}_7, \mathcal{C}_5, 0)$	• $R_2(v_2) \neq R_2(v_3) \Rightarrow \ell(v_2) \neq \ell(v_3)^\dagger$
		$R_2(v_3) = (\mathcal{C}_8, \mathcal{C}_9, 0)$	
$\mathcal{G}_3$	$\mathcal{C}_3$	$R_3(v_5) = (\mathcal{C}_8, \mathcal{C}_6, 0)$	• $R_3(v_5) \neq R_3(v_9) \Rightarrow \ell(v_5) \neq \ell(v_9)^\dagger$
		$R_3(v_9) = (\mathcal{C}_7, \mathcal{C}_5, 0)$	
$\mathcal{G}_4$	$\mathcal{C}_4$	$R_4(v_4) = (\mathcal{C}_8, \mathcal{C}_6, 0)$	• $R_4(v_4) \neq R_4(v_6) \Rightarrow \ell(v_4) \neq \ell(v_6)^\dagger$
		$R_4(v_6) = (\mathcal{C}_7, \mathcal{C}_9, 0)$	
$\mathcal{G}_5$	$\mathcal{C}_1, \mathcal{C}_2$	$R_5(v_1) = (\mathcal{C}_7, \mathcal{C}_5, 0)$	• $R_5(v_i) \sim R_5(v_j), i,j \in \{1,2,3,8\} \Rightarrow$ complete graph
		$R_5(v_2) = (\mathcal{C}_7, \mathcal{C}_5, 0)$	• $R_5(v_1) = R_5(v_2) \Rightarrow \ell(v_1) = \ell(v_2)$
		$R_5(v_3) = (\mathcal{C}_8, \mathcal{C}_9, 0)$	• $R_5(v_3) = R_5(v_8) \Rightarrow \ell(v_3) = \ell(v_8)$
		$R_5(v_8) = (\mathcal{C}_8, \mathcal{C}_9, 0)$	
$\mathcal{G}_6$	$\mathcal{C}_3, \mathcal{C}_4, \mathcal{C}_6$	$R_6(v_4) = (\mathcal{C}_8, 0, 0)$	• $R_6(v_4) \sim R_6(v_5) \Rightarrow (v_4, v_5)$
		$R_6(v_5) = (\mathcal{C}_8, 0, 0)$	• $R_6(v_6) \sim R_6(v_9) \Rightarrow (v_6, v_9)$
		$R_6(v_6) = (\mathcal{C}_7, \mathcal{C}_9, 0)$	• $R_6(v_4) = R_6(v_5) \Rightarrow \ell(v_4) = \ell(v_5)$
		$R_6(v_9) = (\mathcal{C}_7, \mathcal{C}_5, 0)$	
$\mathcal{G}_7$	$\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4, \mathcal{C}_5, \mathcal{C}_6, \mathcal{C}_7, \mathcal{C}_9$	$R_7(v_1) = (0,0,0)$	• $R_7(v_1) = R_7(v_2) = R_7(v_6) = R_7(v_9) \Rightarrow \ell(v_1) = \ell(v_2) = \ell(v_6) = \ell(v_9)$
		$R_7(v_2) = (0,0,0)$	• $R_7(v_3) = R_7(v_4) = R_7(v_5) = R_7(v_8) \Rightarrow \ell(v_3) = \ell(v_4) = \ell(v_5) = \ell(v_8)$
		$R_7(v_3) = (\mathcal{C}_8, 0, 0)$	• $R_7(v_i) \sim R_7(v_j), (i,j) \in \{1,2,6,9\}^2, i > j \Rightarrow (v_i, v_j) \forall (i,j) \in \{1,2,6,9\}^2, i > j$
		$R_7(v_4) = (\mathcal{C}_8, 0, 0)$	• $R_7(v_i) \sim R_7(v_j), (i,j) \in \{3,4,5,8\}^2, i > j \Rightarrow (v_i, v_j) \forall (i,j) \in \{3,4,5,8\}^2, i > j$
		$R_7(v_5) = (\mathcal{C}_8, 0, 0)$	• Note that the last two operations introduce edges between the labels $\{\ell_1^1, \ell_2^2, \ell_2^3\}$ and between the labels $\{\ell_2^1, \ell_1^2\}$
		$R_7(v_6) = (0,0,0)$	
		$R_7(v_8) = (\mathcal{C}_8, 0, 0)$	
		$R_7(v_9) = (0,0,0)$	

†The edge sets of the leaves are given directly, as the leaves are chosen to be cliques

•The red-circled cliques are *binding* SC cliques, i.e., fully included SC cliques that have not been included before by either of the child nodes

- $\mathcal{E}_i(\cdot,\cdot)$ A particular part of the edge set, which is due to constraint item 1(a) if $i = 1, a$; item 1(b) if $i = 1, b$; and item 2 if $i = 2$ (here, $(\cdot)$ serves as a placeholder)
- $\mathcal{V}_i(\cdot,\cdot)$ The set of the vertices that are the endpoints of the edges in $\mathcal{E}_i(\cdot,\cdot)$
- $\mathcal{G}_i(\cdot,\cdot)$ The subgraph formed by the vertices $\mathcal{V}_i(\cdot,\cdot)$ and the edges $\mathcal{E}_i(\cdot,\cdot)$
- $[\kappa(s)]$ The set of labels $\{1, \ldots, \kappa(s)\}, \kappa(s) \in \mathbb{N}$, at tree node s
- $c_i(s)$ Child i of tree node s, $i = 1, 2$
- $\mathcal{C}, \mathcal{C}_{lab}$ The set of all cliques/labeled cliques due to ambiguity graph constraints
- $\mathcal{T}$ A set of partial ambiguity trees that refer to subgraphs of the ambiguity graph
- $\mathcal{N}(\mathcal{G}_i)$ The neighborhood of a labeled subgraph $\mathcal{G}_i$, i.e., the set of *adjacent* labeled subgraphs in $\mathcal{T}$
- $\oplus$ An operator that computes the union of two labeled graphs
- $\eta_{i,j}(\mathcal{G})$ The introduction of edges between the vertices labeled with i and those labeled with j
- $\rho_{i\to j}(\mathcal{G})$ The relabeling of all vertices in $\mathcal{V}$ that are currently labeled with i to label j
- $\mu(\mathcal{G}_1, \mathcal{G}_2)$ An operator that merges two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ as described in Section 5.1
- $\Re(\mathcal{G}, \mathcal{T})$ The remainder graph of $\mathcal{G}$ with respect to the subgraph corresponding to the elements in partial tree set $\mathcal{T}$



$\mathscr{L}$	$\emptyset$	ℓ_1^1	ℓ_1^2	$\{\ell_1^1, \ell_1^2\}$
$a_{\mathscr{L}}$	∞	1	3	∞

$\mathscr{L}$	$\emptyset$	ℓ_2^1	ℓ_2^2	ℓ_2^3	$\{\ell_2^1, \ell_2^2\}$	$\{\ell_2^2, \ell_2^3\}$	$\{\ell_2^1, \ell_2^3\}$	$\{\ell_2^1, \ell_2^2, \ell_2^3\}$
$b_{\mathscr{L}}$	∞	2	∞	∞	13	∞	4.5	∞

Fig. 16 Two child nodes, each with four vertices and two or three different colors, are merged at their parent tree node s. The colors of the vertices correspond to their labels, and the annotated numbers represent the vertex weights. The middle step shows the union of the vertices and edges and is usually omitted in visualizations. At the top, $\mathcal{G}_s$ is the merging result, which contains only two labels. Note that different labels (colors) are used at tree note s compared to at $c_i(s)$, $i = 1, 2$. The tables on the right-hand side show the labels used: the top table for the left child $\mathcal{G}_{c_1(s)}$ and the bottom table for the right child. Several label sets are infeasible and thus are assigned a weight of ∞

$\mathscr{L}$	A set of labels
$\mathcal{P}(\mathcal{X})$	The power set of set $\mathcal{X}$, i.e., the set that contains all subsets of set $\mathcal{X}$
$\mathscr{L}_s(\mathcal{V})$	The set of all labels of the vertices in $\mathcal{V}$ at tree node s
$\mathcal{R}_{c_i(s)\to s}(\mathscr{L})$	The label set of the vertices labeled with $\mathscr{L} \subseteq \mathscr{L}_{c_i(s)}$ at child i after relabeling at tree node s
$\mathcal{E}_{c_1(s),c_2(s)}$	The set of edges introduced by merging using μ
$\mathbf{p}_i$	The actual position of sensor i
$[F]_i$	The ith entry of tuple F
$\mathscr{C}, \mathscr{C}(s)$	The size-constraint (SC) clique set $\mathscr{C}$ and its subset $\mathscr{C}(s)$ denoting the SC cliques at tree node s

Abbreviations

BT: Backtracking; CHOPS: Cold heavy oil production with sand; CW: Clique-width; DS-CDMA: Direct-Sequence (Spread Spectrum) Code-Division Multiple Access; FDMA: Frequency-division multiple access; FPT: Fixed-parameter tractability; ID: Identification number or sequence; ILP: Integer linear program; MW-ISP-SC: Minimum-weight independent set problem with size constraints; MMSE: Minimum mean-squared error; MAP: Maximum a posteriori; RTT: Round-trip time; RV: Random variable; SN: Serial number; TA: Transmit ambiguity; TDMA: Time-division multiple access

Acknowledgements

We gratefully acknowledge the computational resources provided by the RWTH Compute Cluster from RWTH Aachen University under project RWTH0118.

Funding

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement no. 665347.

Authors' contributions

All authors contributed equally to this work. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Received: 31 July 2017 Accepted: 7 February 2018
Published online: 27 March 2018

References

1. E Talnishnikh, J van Pol, HJ Wörtche, in *Intelligent Environmental Sensing. Smart Sensors, Measurement and Instrumentation*, ed. by H Leung, S Chandra Mukhopadhyay. Micro motes: a highly penetrating probe for inaccessible environments, vol. 13 (Springer International Publishing, Cham, 2015), pp. 33–49
2. MB Dusseault, CHOPS—cold heavy oil production with sand in the Canadian heavy oil industry (2004). https://open.alberta.ca/dataset/2815953
3. P Biswas, Y Ye, in *Information Processing in Sensor Networks, 2004. IPSN 2004. Third International Symposium On*. Semidefinite programming for ad hoc wireless sensor network localization, (2004), pp. 46–54. https://doi.org/10.1109/IPSN.2004.1307322
4. AM-C So, Y Ye, in *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2005*. Theory of semidefinite programming for sensor network localization (Society for Industrial and Applied Mathematics, Philadelphia, 2005), pp. 405–414
5. Z Wang, S Zheng, S Boyd, Y Ye, Further relaxations of the sdp approach to sensor network localization (2006). Technical report
6. P Biswas, T-C Lian, T-C Wang, Y Ye, Semidefinite programming based algorithms for sensor network localization. ACM Trans. Sen. Netw. **2**(2), 188–220 (2006). https://doi.org/10.1145/1149283.1149286
7. P Biswas, T-C Liang, K-C Toh, Y Ye, T-C Wang, Semidefinite programming approaches for sensor network localization with noisy distance measurements. **3**(4), 360–371 (2006). https://doi.org/10.1109/TASE.2006.877401

8. H Chen, G Wang, Z Wang, HC So, HV Poor, Non-line-of-sight node localization based on semi-definite programming in wireless sensor networks. IEEE Trans. Wireless Commun. **11**(1), 108–116 (2012). https://doi.org/10.1109/TWC.2011.110811.101739
9. Q Shi, C He, H Chen, L Jiang, Distributed wireless sensor network localization via sequential greedy optimization algorithm. IEEE Trans. Signal Process. **58**(6), 3328–3340 (2010). https://doi.org/10.1109/TSP.2010.2045416
10. P Tseng, Second-order cone programming relaxation of sensor network localization. SIAM J. Optim. **18**(1), 156–185 (2007). https://doi.org/10.1137/050640308
11. GC Calafiore, L Carlone, M Wei, in *18th Mediterranean Conference on Control Automation (MED), 2010*. Position estimation from relative distance measurements in multi-agents formations, (2010), pp. 148–153. https://doi.org/10.1109/MED.2010.5547601
12. B Liu, H Chen, Z Zhong, HV Poor, Asymmetrical round trip based synchronization-free localization in large-scale underwater sensor networks. IEEE Trans. Wireless Commun. **9**(11), 3532–3542 (2010). https://doi.org/10.1109/TWC.2010.090210.100146
13. E Duisterwinkel, L Demi, G Dubbelman, E Talnishnikh, HJ Wörtche, JW Bergmans, Environment mapping and localization with an uncontrolled swarm of ultrasound sensor motes. Proc. Meet. Acoust. **20**(1), 030001 (2013). https://doi.org/10.1121/1.4879264
14. S Schlupkothen, G Ascheid, in *2015 14th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET) (Med-Hoc-Net'15)*. Localization of wireless sensor networks with concurrently used identification sequences, (Vilamoura, Portugal, 2015), pp. 1–7. https://doi.org/10.1109/MedHocNet.2015.7173166
15. S Schlupkothen, G Ascheid, in *2016 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*. Joint localization and transmit-ambiguity resolution for ultra-low energy wireless sensors, (2016), pp. 48–53. https://doi.org/10.1109/WiSEE.2016.7877302
16. S Schlupkothen, B Prasse, G Ascheid, in *2016 Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*. A dynamic programming algorithm for resolving transmit-ambiguities in the localization of wsn, (2016), pp. 1–8. https://doi.org/10.1109/MedHocNet.2016.7528420
17. CA Floudas, PM Pardalos, *Encyclopedia of Optimization, Second Edition*. (Springer, Secaucus, 2009)
18. DZ Du, KI Ko, *Theory of Computational Complexity. Wiley Series in Discrete Mathematics and Optimization*. (Wiley, New York, 2011)
19. J Flum, M Grohe, *Parameterized Complexity Theory. Texts in Theoretical Computer Science. An EATCS Series*. (Springer, Berlin Heidelberg, 2006)
20. F Gurski, I Rothe, J Rothe, E Wanke, *Exakte Algorithmen Für Schwere Graphenprobleme. eXamen.press*. (Springer, Berlin Heidelberg, 2010)
21. F Gurski, A comparison of two approaches for polynomial time algorithms computing basic graph parameters. CoRR. **abs/0806.4073** (2008). http://arxiv.org/abs/0806.4073
22. S Oum, SH Sæther, M Vatshelle, Faster algorithms for vertex partitioning problems parameterized by clique-width. Theor. Comput. Sci. **535**, 16–24 (2014)
23. MR Fellows, FA Rosamond, U Rotics, S Szeider, Clique-width is np-complete. SIAM J. Discrete Math. **23**(2), 909–939 (2009). https://doi.org/10.1137/070687256
24. HL Bodlaender, A tourist guide through treewidth. Acta Cybern. **11**(1-2), 1–21 (1993)
25. B Courcelle, S Olariu, Upper bounds to the clique width of graphs. Discrete Appl. Math. **101**(1–3), 77–114 (2000)
26. W Espelage, F Gurski, E Wanke, in *Algorithms and Data Structures: 7th International Workshop, WADS 2001 Providence, RI, USA, August 8–10, 2001 Proceedings*, ed. by F Dehne, J-R Sack, and R Tamassia. Deciding clique-width for graphs of bounded tree-width (Springer, Berlin, Heidelberg, 2001), pp. 87–98
27. EW Weisstein, Vertex-induced subgraph. From MathWorld—A Wolfram Web Resource (2016). Last visited on 02/02/2016. http://mathworld.wolfram.com/Vertex-InducedSubgraph.html
28. M Tawarmalani, NV Sahinidis, A polyhedral branch-and-cut approach to global optimization. Math. Program. **103**, 225–249 (2005)
29. N Patwari, AO Hero III, in *Proceedings of the 2Nd ACM International Conference on Wireless Sensor Networks and Applications. WSNA '03*. Using proximity and quantized rss for sensor localization in wireless networks (ACM, New York, 2003), pp. 20–29. https://doi.org/10.1145/941350.941354
30. JA Costa, N Patwari, AO Hero III, Distributed weighted-multidimensional scaling for node localization in sensor networks. ACM Trans. Sen. Netw. **2**(1), 39–64 (2006). https://doi.org/10.1145/1138127.1138129
31. EW Weisstein, Normal difference distribution. From MathWorld—A Wolfram Web Resource (2016). http://mathworld.wolfram.com/NormalDifferenceDistribution.html. Accessed 02 Feb 2016
32. A Goldsmith, *Wireless Communications*. (Cambridge University Press, New York, 2005)