Constraint-Based Distance Estimation in Ad-Hoc Wireless Sensor Networks

Urs Bischoff, Martin Strohbach, Mike Hazas, and Gerd Kortuem

Lancaster University, Lancaster LA1 4YW, UK

Abstract. We propose a lightweight localisation approach for supporting distance and range queries in ad hoc wireless sensor networks. In contrast to most previous localisation approaches we use a distance graph as spatial representation where edges between nodes are labelled with distance constraints. This approach has been carefully designed to satisfy the requirements of a concrete application scenario with respect to the spatial queries that need to be supported, the required accuracy of location information, and the capabilities of the target hardware. We show that this approach satisfies the accuracy requirements of the example application using simulations. We describe the implementation of the algorithms on wireless sensor nodes.

1 Introduction

Cooperative localisation algorithms play an important role for wireless sensor networks. In cooperative localisation, nodes work in a peer-to-peer manner to compute a map of the network using only the resources provided by the nodes themselves [1]. This makes cooperative algorithms especially suited for sensor network deployments that require true ad hoc localisation without external infrastructure.

The design of cooperative localisation algorithms requires a careful tradeoff between *fidelity* and *complexity*. Fidelity refers to quality of the computed result and includes precision and accuracy, as well as timeliness. Because measurements used as input to localisation algorithms are to some degree unreliable or inaccurate, node locations cannot be determined with absolute certainty. Complexity refers to the amount of resources consumed by an algorithm. In the wireless sensor network literature, complexity is most often considered in the context of energy efficiency and network use, although hardware, memory and time requirements are also important. Complexity is important because typical sensor network components possess very limited resources for processing and communication.

In this paper, we describe a novel approach to cooperative localisation that focuses on supporting distance and range queries, and trades fidelity against complexity. A *distance query* retrieves the distance between two given nodes, and a *range query* retrieves the IDs of nodes that are within a given distance from a given node. This approach has been carefully designed to satisfy the requirements of a concrete application scenario with respect to the accuracy

K. Römer, H. Karl, and F. Mattern (Eds.): EWSN 2006, LNCS 3868, pp. 54–68, 2006.

of location information, the spatial queries that need to be supported and the capabilities of the target hardware. Our approach can be characterized as in the following ways.

1. In contrast to most previous approaches we use a distance graph as spatial representation where edges between nodes are labelled with distance constraints. Node coordinates (absolute or relative) are not represented.
2. The distance constraint of each edge is represented as an interval [a,b] indicating that the distance between two nodes is at least a and at most b. The distance intervals are used for modelling uncertainty and localisation errors.
3. A constraint propagation algorithm is used to compute an overall consistent distance graph.

The next section discusses the localisation requirements of our application scenario. Following that is a description of the spatial model and the cooperative algorithm. We then evaluate the accuracy of our approach for range and distance queries using simulations, and we describe the resource requirements of an implementation of the algorithm on a specific type of wireless sensor node. Finally, we contrast our approach to some well-known algorithm in wireless sensor networks, and provide a comparative quantification of the computational and memory requirements of our algorithm.

2 Motivating Application Example

The application scenario that motivates this research is taken from the chemical industry work place in which safe handling and storage of chemical containers is of key importance. The goal here is to assist trained staff in detecting hazards of inappropriately stored chemicals. These hazards are defined by safety rules that are defined in terms of the storage conditions of the chemicals. For example, they prescribe that incompatible materials, such as reactive chemicals, must not be stored in immediate proximity. The meaning of "proximity" depends on a number of parameters including the type of involved chemicals.

We have developed a wireless sensor network for detecting possible safety hazards [2, 3]. Chemical containers are moved around quite frequently and may end up in a location without global networking capabilities, for example during transport inside the hull of ships or trucks. Moreover, it is generally unrealistic to rely on centralised services in environments that deal with chemical containers. Thus the key design goal was to enable the detection of safety hazards without the involvement of external infrastructure. This required a cooperative approach in which the containers themselves are able to sense their environment and interpret their storage conditions with respect to predefined safety rules. We achieved this goal by embedding perception, safety rules and higher-level reasoning capabilities into sensor nodes attached to chemical containers.

2.1 Peer-to-Peer Position Sensing

The sensor nodes make use of a simplified version of Relate, which is an ultrasonic peer-to-peer positioning system for determining the relative locations of a set of

mobile computing devices [4]. The simplified version does not require a connection to a host PC and delivers accurate range information between nodes of a wireless sensor network. Using the Relate technology we can directly measure the distance between co-located nodes and do not have to rely on network measurements such as received radio signal strength. The maximum distance that can be measured by Relate is about two meters. Relate nodes cooperate by broadcasting distance measurements over the network. Thus each node has access to distance measurements between a wide range of nodes and using the algorithm described in this paper can build a spatial model of its network neighbourhood.

2.2 Localisation Requirements

In our application scenario a container A tries to detect whether there are any reactive chemicals in its proximity, whereby proximity is defined as a circular area around A with a domain-specific radius. In this situation A needs to determine which containers are located within the proximity zone and which ones are located outside of it. The cooperative reasoning process described in [2] involves all nodes located within the proximity zone, regardless of their absolute location. Consequently, the spatial model for this application example must be able to support range queries, i.e. queries that return all nodes located in a specified range around a given node.

The required precision of the distance and range information depends in large measures on the physical dimensions of chemical containers. A chemical container as used in our application scenario is barrel-shaped and has a diameter between 60 and 80 cm. We assume that a range measurement precision of about half the diameter of a container is high enough. The Relate system provides measurements with 10 cm granularity or better, but only within the relatively short detection range of its ultrasound transducers (2 m). Thus a localisation algorithm must be able to indirectly compute distances between nodes that are beyond this range.

2.3 Hardware Requirements

The hardware used for instrumenting a chemical container is comparable to that used in other wireless sensor networks nodes such as the Berkeley Motes. The hardware consists of two separate components. The Relate component implements peer-to-peer distance sensing, and the arteFACT component implements the reasoning framework. Both are based on the Particle Smart-its, an embedded wireless sensing platform [5]. Particle Smart-Its incorporate a PIC 18F6720 microcontroller with 128 KB of program flash memory and 4 KB of RAM. Particle Smart-its communicate via a slotted RF protocol and can provide effective data rates of up to 39 kbit/s. In the following we will describe the localisation system we designed for this application scenario.

3 Spatial Model and Algorithm

Our approach is based on the idea of representing localisation information in the form of a constraint network that expresses restrictions on the distance between

network nodes. Node coordinates (absolute or relative) are not represented. To represent constraints we use a graph where edges between nodes are labelled with distance intervals. More formally, a *distance graph* is an undirected labelled graph $G = (N, E, C)$ where N is a set of network nodes, E is a set edges and C is a constraint function which assigns to each edge a *distance interval* $[u, v]$ with $u, v \in \mathbb{R}$ and $u \leq v$. Distance intervals are used to represent knowledge about the real-world distance between nodes. If the edge between nodes A and B is labelled with the interval $I_{AB} = [u, v]$ and $d(A, B)$ is the real-world distance between A and B, then:

$$u \leqslant d(A, B) \leqslant v\,. \tag{1}$$

Thus an interval $[u, v]$ indicates that the distance between the two respective network nodes is at least u and at most v. An interval $[u, u]$ indicates that the distance is exactly u. An interval $[0, \infty]$ is the most generic (or empty) constraint since it does not limit the distance between nodes in any meaningful way. An edge labelled with an empty constraint can be omitted from the graph without any loss of information. Figure 1 shows an example of a distance graph.

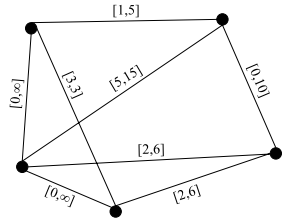

Fig. 1. Example distance graph

The algorithm for computing and updating the distance graph is a modified Floyd-Warshall algorithm for computing all-pairs shortest paths. Instead of adding distances and selecting a minimum distance in each step, it infers and adds distance intervals.

1. Initialize all edges with the empty constraint $[0, \infty]$
2. Receive distance measurements from Relate positioning system.
3. For all edges perform the modified Floyd-Warshall iteration:
   (a) infer new distance interval, and
   (b) combine interval with interval inferred in previous iteration.
4. Go to step 2

Step 2 of this algorithm takes a raw distance measurement d from the Relate system and transforms it into a distance interval $[d-\epsilon, d+\epsilon]$ which accounts for the inaccuracy of the measurement d. ϵ is a constant derived from an error model of the Relate positioning technology. More details on this transformation can be found in Sect. 4. The resulting distance interval $[d-\epsilon, d+\epsilon]$ is added to the graph.

Step 3 of this algorithm traverses the whole graph to infer new or more specific constraints. This is done using a number of transitive inference steps which derive information about I_{AC} from I_{AB} and I_{BC}. Figure 2 illustrates an inference

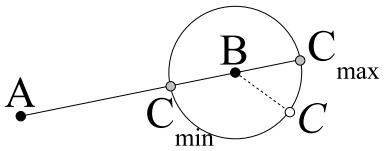

Fig. 2. Two intervals are used to infer a third one

step assuming the position of A and B. For simplicity reasons we assume zero-length (i.e. $u = v$) intervals. The distances $d(A, B)$ and $d(B, C)$ are already known, either because they have been inferred in a previous step or because they represent measurements. $d(A, C)$ must be inferred in this step. We know that C must lie on the circle with radius $d(B, C)$ and centre B. Hence we can represent all possible distances between A and C as $[d(A, C_{\min}), d(A, C_{\max})]$. Equations (2)-(4) illustrate the inference steps for the general case.

$$I_{AB} = [u_1, u_2],\ I_{BC} = [v_1, v_2] \\ \Rightarrow \quad I_{AC} = [v_1 - u_2, u_2 + v_2] \quad \text{if} \quad u_2 \leqslant v_1\,. \tag{2}$$

$$I_{AB} = [u_1, u_2],\ I_{BC} = [v_1, v_2] \\ \Rightarrow \quad I_{AC} = [u_1 - v_2, u_2 + v_2] \quad \text{if} \quad v_2 \leqslant u_1\,. \tag{3}$$

$$I_{AB} = [u_1, u_2],\ I_{BC} = [v_1, v_2] \\ \Rightarrow \quad I_{AC} = [0, u_2 + v_2] \quad \text{if} \quad I_{AB} \cap I_{BC} \neq \varnothing\,. \tag{4}$$

These inference steps for all interval cases are also illustrated in Fig. 3. If the distance intervals I_{AB} between A and B and I_{BC} between B and C do not overlap all nodes have a different position. Hence, $d(A, C) > 0$ and the interval boundaries are calculated according to case (a) (cf. (2)) and (b) (cf. (3)) respectively. Otherwise A and C could have the same position and the minimum distance is chosen to be 0 in the inferred interval (c) (cf. (4)). There could exist several paths between pairs of nodes. In the second iteration step the inferred interval I_{inf} is therefore compared to the previously inferred interval I_{pre} to obtain the smallest consistent interval I_{new}. Consistency means that there is a position for each node in the graph so that all calculated distances would lie in their respective intervals. Thus, if the graph is consistent before the Floyd-Warshall iteration step, the intervals in step 3b will overlap and the smallest consistent

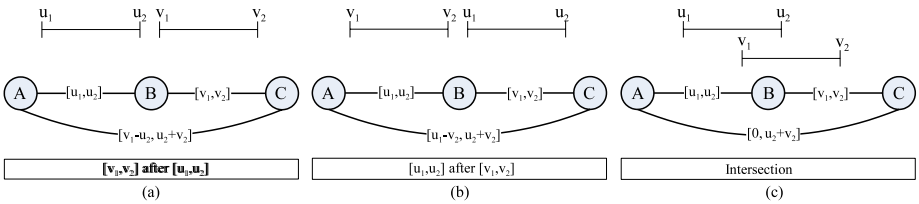

Fig. 3. Three possible inference steps for input intervals $[u_1, u_2]$ and $[v_1, v_2]$

interval is obtained by intersecting them (6). Otherwise, we create a consistent graph by choosing the smallest interval that contains both input intervals (5).

$$I_{\text{pre}} = [u_1, v_1] \quad \text{and} \quad I_{\text{inf}} = [u_2, v_2]$$

$$\text{IF } ((v_1 \leqslant u_2) \text{ OR } (v_2 \leqslant u_1)) \Rightarrow I_{\text{new}} = [\min(u_1, u_2),\ \max(v_1, v_2)] \tag{5}$$

$$\text{ELSE} \Rightarrow I_{\text{new}} = [\max(u_1, u_2),\ \min(v_1, v_2)]\,. \tag{6}$$

4 Evaluation

In this section we evaluate the feasibility and accuracy of our algorithm and spatial model. Using a simulation environment, we characterise the algorithm results in terms of range query success and in terms of distance accuracy. We also show that our algorithm can be implemented on a resource-constrained wireless sensing platform such as the Particle Smart-Its.

4.1 Accuracy

Accuracy was evaluated in a simulation environment. For the purposes of comparison, our simulation parameters were similar to that described by Shang and Ruml [6].[1] Our test networks consist of two hundred sensor nodes. The nodes are randomly placed, using a uniform distribution, within a $10r \times 10r$ square area. Next, a measurement range is chosen; our standard range is $2r$. Each node can measure the distance to neighbouring nodes that are within the measurement range. These distance measurements build the input to our inference algorithm. A single experiment simulates one hundred random networks each consisting of two hundred nodes.

In early experiments, we used an error model for modelling measurements more realistically. The error model was based on experiments with Relate USB dongles [4] that have similar characteristics as the Relate devices we used. In one experiment it was shown that 90% of the true distances lie in the interval $[\widetilde{d} - 2cm, \widetilde{d} + 4.5cm]$; $\widetilde{d}$ is the measured distance. However, tests showed that the error model does not have a big influence on the final errors. It is the algorithm itself that introduces the biggest errors; so the influence of the initial measurement errors and interval lengths is negligible for the overall end result.

Thus, we decided not to use this error model. This allows us to produce results that are more easily comparable to other algorithms because they are independent of the underlying error model which is based on a specific technology. In the following we present the evaluation results of these experiments.

[1] Despite this, the results of our algorithm cannot be *directly* compared to that of Shang and Ruml. Whereas we measure the errors between estimated node-to-node distances and the ground truth distances, Shang and Ruml measure the Euclidean error between estimated locations and ground truth locations. However, it can be shown that for a given spatial configuration of nodes, the distance estimation error (our quantity) roughly corresponds to the location estimation error (computed by Shang and Ruml); this gives some basis for comparison.

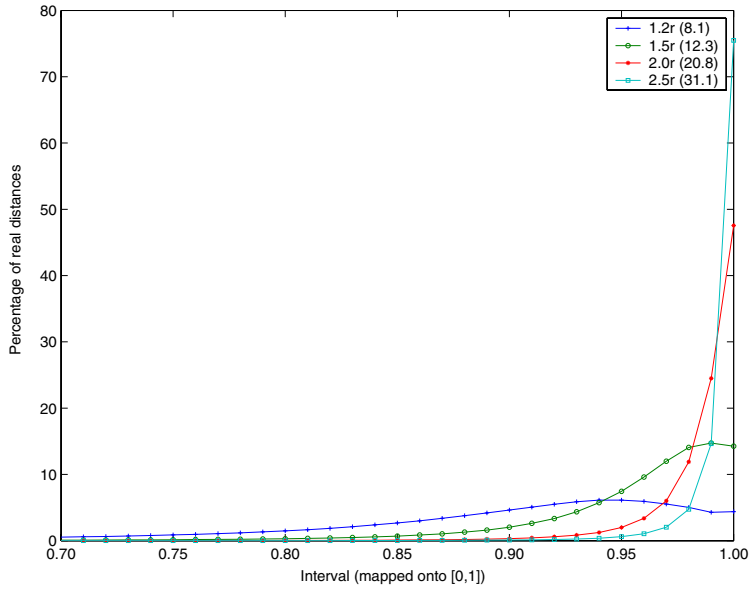

Fig. 4. Distribution of distances within intervals. As shown, this distribution depends on the measurement radius (connectivity).

We first analysed the characteristics of the intervals. As mentioned earlier the algorithm computes intervals; hence, if we want to do range queries, a position within each interval has to be chosen as the represented distance of the interval. We observed that the ground truth distances are not uniformly distributed within the intervals produced by our algorithm. We projected all intervals onto a standard interval [0,1]. These intervals were subdivided into one hundred subintervals. Then, we counted the number of times a real distance falls within the limits of each of the hundred sub-intervals; Fig. 4 shows that most real distances are close to the upper boundary of the interval results.

There are two main reasons for this. It can be shown that the lower boundary decreases faster than the upper boundary increases with each inference step. So, it is more probable that the real distance is not around the centre of the interval but closer to the upper boundary. This effect is intensified if several steps are necessary to infer the distance interval to another node. Because distant nodes are in the majority, we can clearly see this tendency in the overall distribution. There is also a geometric explanation. In Fig. 2, we showed that A sees C as lying on a circle around B. If we assume that the position of C is uniformly distributed on the periphery of the circle, we can see that the distance $\overline{AC}$ is not uniformly distributed between $\overline{AC}_{\min}$ and $\overline{AC}_{\max}$; the probability density is higher toward the boundaries.

By choosing an interval position close to the upper boundary as a representative for the real distance, the total error rate can be minimised. We use values between 0 and 1 to refer to interval positions as a fraction of the interval size. The distance between two nodes A and B can be calculated according to (7).

We have experimented with various interval positions and measurement ranges. An optimal choice of p depends, among other reasons, on the connectivity of the graph. Because we do not know the connectivity in advance, we have to find a good trade-off. We chose the interval position $p = 0.98$ for our next experiments that are based on different sets of simulated networks.

$$I_{AB} = [u, v] \quad \Rightarrow \quad d(A, B) \approx v - p(v - u) \quad \text{with} \quad p \in [0, 1]\,. \tag{7}$$

Range Query Classification Error. In our range query scenario we are interested in the number of miss classifications. Therefore we distinguish between false positives and false negatives to characterize the accuracy of our algorithm. If a range query classifies a node as being inside the range even though it is outside, it is a false positive. On the other hand, a false negative is a node reported to be outside the query range even though it is inside in reality.

Figure 5 shows the error rate of false negatives, false positives and the total error as percentage of the number of nodes classified as being inside the respective circular query range. The measurement range is chosen to be $2r$. The error rate is displayed with respect to the range specified in the query. Figure 6 depicts the same error but as percentages of the total number of nodes. The maximum false positive error ratio is 1.5% with respect to the number of nodes classified inside the query range. So for a query range of $4r$, 1.5% of the nodes are wrongly classified as being inside the query range. With respect to the total number of nodes we have a maximum of 0.65% false positives, 0.45% of false negatives and a maximum total error of 1.1%.

For small ranges we have direct ultrasound distance measurements. For larger query ranges we depend on inferred distances. In both graphs we can therefore see an increase of the error rate at the beginning. Then, we observe a sharp decrease of the error rate. This is not because distance estimation is more accurate at

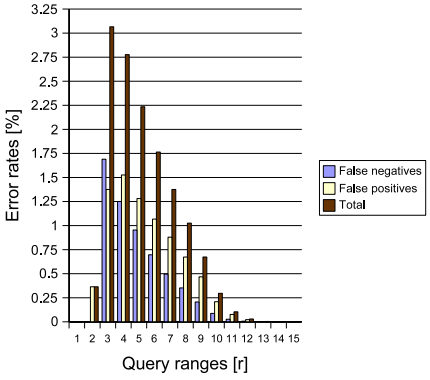

Fig. 5. Error rates in random uniform networks with respect to the number of nodes classified as being inside the range. The measurement range is $2r$.

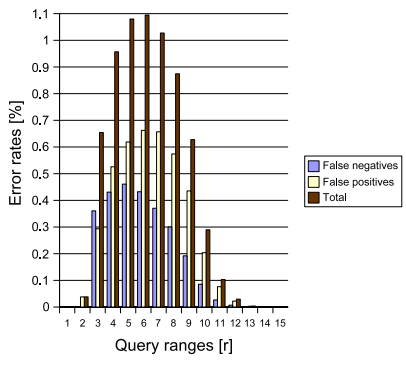

Fig. 6. Error rates in random uniform networks with respect to the total number of nodes. The measurement range is $2r$.

longer distances, but because we display classification errors as *relative* ones. For example, in Fig. 5 we show the error with respect to the total number of nodes classified inside the respective query range. So if the query range is increased and the absolute error remains the same, the relative error would decrease because of a larger number of nodes inside the query range. Similarly, in Fig. 6, for large query radii, most nodes clearly contain most of the other nodes in their range. Thus the probability of a misclassification is smaller. The maximum possible query radius of around $14r$ affects only a few nodes in the corners of the $10r \times 10r$ square area.

The previous test was based on a uniformly distributed random network. We expect our algorithm to perform worse if the shortest path distance between two nodes is long compared to the Euclidean distance. This is the case for nodes in the two wings of a C-shaped network. These networks consist of 160 nodes randomly positioned in C-shape; they are generated analogously to the random C-shaped placement used in experiments by Shang and Ruml [6].

As expected we observe a higher error rate for the C-shaped network. Figures 7 and 8 show the classification error rates. The error rate is significantly higher. We observe a lot of false negative errors; the discrepancy between the Euclidean distance and the graph distance is the main reason for this. The error rate could be reduced by changing the value of p in (7); the optimal p is topology-dependent. But because we do normally not know the topology in advance, we used the same p as in the experiments with uniform networks.

Distance Error. In order compare with other algorithms, we analysed the distance errors. Again we chose $p = 0.98$ as the distance position inside the intervals. Figure 9 shows a cumulative distribution of all the distance errors in the random uniform networks. In total, almost two million distances were represented in our simulations (100 networks consisting of 200 nodes each). All of these distances were compared with the inferred distances. Figure 9 shows,

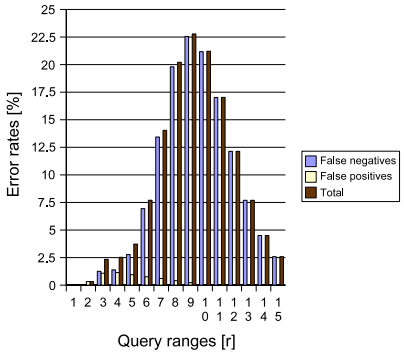

Fig. 7. Error rates in random C-shaped networks with respect to the number of nodes classified as being inside the range. The measurement range is $2r$.

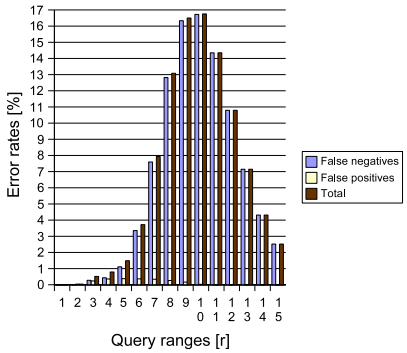

Fig. 8. Error rates in random C-shaped networks with respect to the total number of nodes. The measurement range is $2r$.

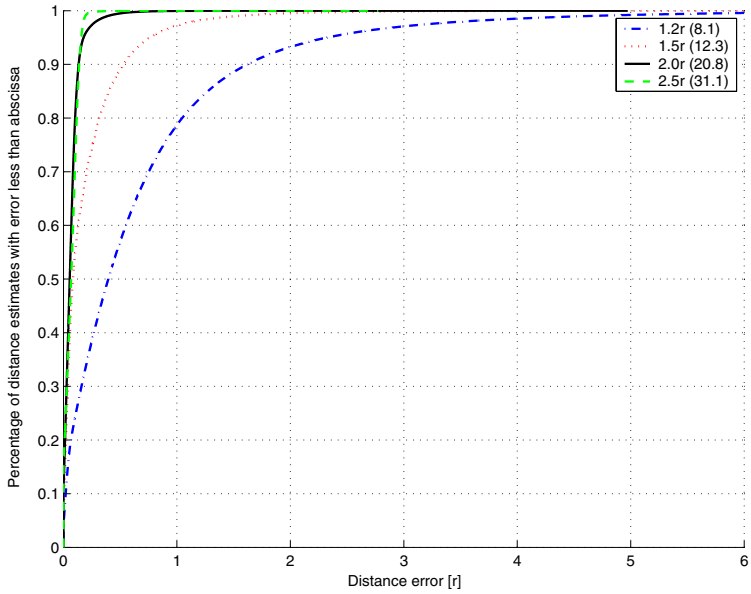

Fig. 9. Cumulative distribution of distance errors for $p = 0.98$ in random uniform topology. The distribution is shown for different measurement ranges (connectivities).

for example, that around 90% of the distances are less than $1.5r$ away from the real distance if we take $1.2r$ as our measurement radius. It is expected that the accuracy is improved if the measurement range (and connectivity) is increased. In the evaluation of the missclassification error, we chose a measurement radius of $2r$. For this radius we expect the distance error of 90% of the nodes to be less then $0.15r$.

Figure 10 shows the cumulative distribution of distance errors in the random C-shaped networks. Although a large portion of the inferred distances are accurate, there is also a significant number of inferred distances that are not. Again the discrepancy between the graph distance and the Euclidean distance explains this observation. However, we have this extreme discrepancy between some pairs of nodes only. Shang and Ruml [6] only report the error medians, rather than their entire error distributions. However, our distance error median is comparable to their position error median.

4.2 Feasibility

The algorithm has been implemented on the Particle Smart-Its. The arteFACT component listens to measurement broadcasts by Relate nodes and maintains its own spatial model. The location model is updated by each node independently whenever a Relate node broadcasts its measurement updates.

Before we update the model, inferred intervals that have not been replaced by new measurements are reset to $[0, \infty]$. In a similar way we reset old measurement

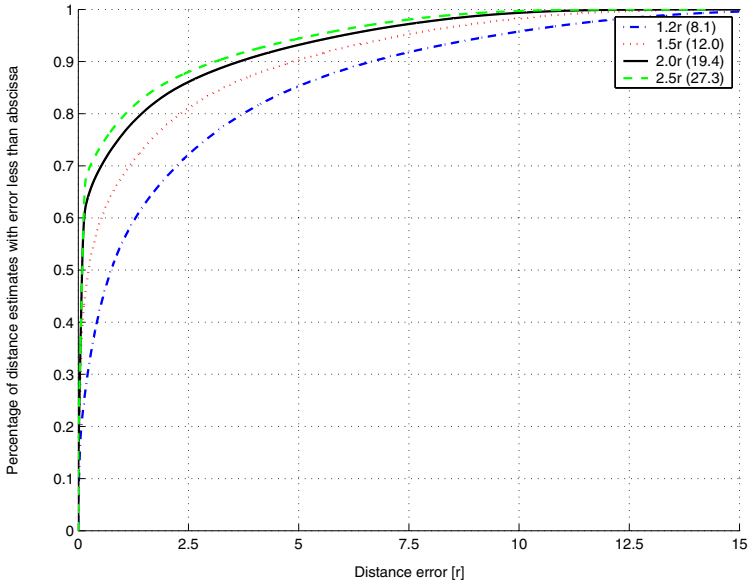

Fig. 10. Cumulative distribution of distance errors for $p = 0.98$ in random C-shaped topology. The distribution is shown for different measurement ranges (connectivities).

intervals to account for the case when devices have moved out of the ultrasound measurement range. This is achieved by timestamping the measurements with the local clock on arrival. In our prototypical implementation we chose a timeout of about 8s. Further experiments will help to choose this parameter according to update requirements. This will also help us to minimize the number of model updates for a given required update rate. Currently we update the model whenever we receive new measurements.

The current implementation supports a network of 10 nodes using 17% of the program memory and 64% of the data memory in the worst case. These are promising results, especially in the light that further optimizations are possible. For example, we use a full $n \times n$ adjacency matrix to represent the graph.

5 Discussion

Our main objective was to find an efficient method for range queries. Thus, false positive and false negative errors are our main concern; distance accuracy is of less importance. Of course, distance accuracy and classification error rate (false negative and false positive errors) are correlated. Using intervals instead of just distances gives us several advantages. First, intervals give us a measure of uncertainty. Second, we can influence the number of false positive errors with respect to the number of false negative errors. Because we have intervals, we know exactly the lower and upper limit of a distance. For example, if we choose $p = 1.0$ in (7), we would not have any false positives but only false negatives; or if we choose $p = 0.0$,

we do not have any false negatives, but only false positives (neglecting raw measurement errors). There is an optimal p that minimises the overall error; the goal is to find a good trade-off between false positives and false negatives. However, if the application is very sensitive to only one kind of error, that kind of error could be minimised. We achieved a relatively low error rate for our range queries by exploiting the non-uniform distribution of distances within the intervals.

Irrespective of the number of false positives and false negatives we also showed that the distance errors are small for most of the node pairs. In Sect. 2.2 we stated that the distance accuracy requirements of our application are around 30-40 cm. We showed for a measurement range of $2r$ that the distance error of 90% of the nodes is less than $0.15r$. So if we choose r to be around 1 m (which corresponds to the Relate measurement range of 2 m ($= 2r$)), we expect to satisfy these requirements.

We first evaluated our algorithm on random uniform networks. We assume that these random networks did not represent the worst or best case. The performance of our algorithm is expected to be worse if the Euclidean distance between two nodes is small but a large number of inference steps are necessary to infer the distance interval, i.e. long graph distance between nodes. We modelled this scenario in random C-shaped networks.

For our scenario we have not exploited all the information. We do not directly make use of the fuzzy information that is given by the intervals. The length of the intervals could be an indicator for accuracy; we expect less accurate results if the intervals are longer.

The main advantage of our approach is its low complexity. We showed that it can be implemented on resource constrained embedded sensor nodes. Based on the requirements of our scenario, the algorithm only computes distances. This is in contrast to other location algorithms for wireless sensor networks which generally compute positions (cf. Sect. 6).

6 Related Work

We are not aware of any practical work on range queries in wireless sensor networks. However, there are several research areas that provide theoretical and practical foundations for our work. Hightower and Borriello provide a good overview of positioning systems for ubiquitous computing [7], and there have been a number of surveys describing and classifying positioning algorithms [8–10].

Most important to our work is the Relate project from which we used the sensing hardware. A USB dongle variant was used in [4] to provide relative positioning support for mobile devices. In contrast to our work, Relate dongles rely on host devices such as laptops or PDAs to compute the spatial model.

The Dolphin project developed a 3D peer-to-peer indoor positioning system [11]. They use bidirectional ultrasonic ranging to measure distances between neighbouring nodes. Several anchor nodes have to be manually configured with their position. Then, a distributed iterative multilateration algorithm is used to determine the position of other nodes with respect to the anchor nodes. Com-

pared to other methods the feasibility of their approach was demonstrated by implementing the algorithm on real sensor nodes. However, their technique cannot be directly applied to Cooperative Artefact networks as it relies on anchor nodes which have knowledge of their surveyed locations.

There exist a lot of localisation algorithms for wireless sensor networks. Most algorithms depend on anchor nodes because they compute absolute node positions. Our application cannot depend on anchor nodes. Therefore, we mainly considered relative positioning algorithms: MDS-MAP [12] and the Self-Positioning Algorithm (SPA) [13] are two typical examples, which calculate relative positions using simple connectivity. MDS-MAP uses an all-pairs shortest path algorithm to compute a distance matrix. Then, multi-dimensional scaling (MDS) is applied to find an embedding in the two-dimensional space. In SPA each node forwards distance measurements to neighbouring nodes and calculates node positions using trigonometry in the local neighbourhood. Then, these local coordinate systems are merged into one coordinate system.

MDS-MAP(P) is an improved MDS-MAP algorithm [6]. It is a distributed localisation algorithm which yields median accuracies similar to our algorithm. Table 1 shows the complexity of our algorithm, compared to that of MDS-MAP(P). To compute the complexity of MDS-MAP(P), we used the *Numerical Recipes in C* algorithms for eigenvector decomposition, and Levenberg-Marquardt non-linear regression.[2] The step which merges the local maps, as well as some of the preprocessing steps in the eigenvector decomposition were neglected. In the per node computations which deal only with local maps, it was assumed that distance measurements were available for half the total pairs of neighbours. It was also assumed that the regression takes five iterations to complete.[3]

One of the advantages of the MDS-MAP(P) algorithm is that it is distributed. However, this means that *each* node must first compute its own local map, and then work together with the other nodes to arrive at the global map. Assuming each node has the ten or more neighbours necessary to achieve reasonable accuracy, our interval-based Floyd-Warshall algorithm has lower processing requirements for networks with a total of one hundred nodes or less. It is also important to note that our interval-based Floyd-Warshall algorithm uses only 16-bit integer operations, whereas the MDS-MAP(P) steps mostly rely on floating point operations (such as computing matrix inverses).

[2] Because of the nature of location models where distances are related to (x, y) coordinates using the Euclidean relationship, the gradient matrix used by the regression algorithm holds only four non-zero gradients. Using this knowledge, large computational savings ($O(k^3)$ instead of $O(k^4)$) can be made in the Levenberg-Marquardt algorithm. In our computations, we assumed that this optimisation had been made. To compute the complexity of the matrix inverse operations required by the Levenberg-Marquardt method, we used *Numerical Recipes* Gauss-Jordan elimination with full pivoting.

[3] For this type of problem, non-linear regression normally requires between three and ten iterations [6].

Table 1. Resource requirements of MDS-MAP(P) and interval-based Floyd-Warshall. n is the total number of nodes in the network, and k is the number of neighbours to each node.

	Operations	Storage
MDS-MAP(P) per node		
Compute shortest paths (Floyd-Warshall)	$2k^3 + 6k^2 + 6k + 2$	$k^2 + 2k + 4$
Multidimensional scaling	$36k^3 + 110k^2 + 114k + 39$	$2k^2 + 7k + 3$
Non-linear regression	$276k^3 - 460k^2 + 756k - 287$	$8k^2 + 35k + 52$
Per node total	$314k^3 - 344k^2 + 876k - 246$	$8k^2 + 35k + 52$
MDS-MAP(P) all nodes	$\approx n(314k^3 - 344k^2)$	$\approx n(8k^2 + 35k)$
Interval-based Floyd-Warshall	$14n^3$	$2n^2 + 5$

Thus, the MDS-MAP(P) algorithm trades off higher computational complexity and storage requirements in order to produce *coordinate location* results. By contrast, our algorithm requires less processing and storage, and instead estimates the node-to-node *distances*.

7 Conclusions

We have presented a lightweight localisation algorithm for ad hoc sensor networks. The algorithm has been designed to satisfy concrete application requirements in terms of the accuracy of location information, spatial queries and the capabilities of the target hardware. In contrast to most previous localisation approaches our algorithm computes constraints on the distance between network nodes in the form of distance intervals. These intervals can be used to represent inaccuracy of distance measurements as well as imprecision as result of inference steps. The length of intervals can be seen as a quality measure for spatial information. In future work we will look at ways to improve the information quality by reducing the interval length. In particular, we are considering combining our approach with Freksa's reasoning method for inferring spatial relations between neighbouring point objects in 2D [14].

References

1. Patwari, N., Ash, J.N., Kyperountas, S., III, A.O.H., Moses, R.L., Correal, N.S.: Locating the nodes: Cooperative localization in wireless sensor networks. IEEE Signal Processing Magazine **22**(4) (2005) 54–69
2. Strohbach, M., Gellersen, H.W., Kortuem, G., Kray, C.: Cooperative artefacts: Assessing real world situations with embedded technology. In: Proceedings of the Sixth International Conference on Ubiquitous Computing (UbiComp). (2004)
3. Strohbach, M., Kortuem, G., Gellersen, H.W., Kray, C.: Using cooperative artefacts as basis for activity recognition. In: Ambient Intelligence: Second European Symposium (EUSAI 2004). (2004)

4. Hazas, M., Kray, C., Gellersen, H., Agbota, H., Kortuem, G., Krohn, A.: A relative positioning system for co-located mobile devices. In: Proceedings of the Third International Conference on Mobile Systems, Applications, and Services (MobiSys). (2005)
5. TecO: Particle webpage. `http://particle.teco.edu/devices/index.html` (2005)
6. Shang, Y., Ruml, W.: Improved MDS-based localization. In: Proceedings of the 23rd Conference of the IEEE Communications Society (Infocom 2004). (2004)
7. Hightower, J., Borriello, G.: A survey and taxonomy of location systems for ubiquitous computing. Extended paper from Computer 34(8) p57-66 (2001)
8. Langendoen, K., Reijers, N.: Distributed localization in wireless sensor networks: A quantitative comparison. Computer Networks **43**(4) (2003) 499–518
9. Muthukrishnan, K., Lijding, M., Havinga, P.: Towards smart surroundings: Enabling techniques and technologies for localization. In: Proceedings of the First International Workshop on Location– and Context-Awareness (LoCA), Springer-Verlag (2005)
10. Niculescu, D.: Positioning in ad hoc sensor networks. IEEE Network **18**(4) (2004) 24–29
11. Minami, M., Fukuju, Y., Hirasawa, K., Yokoyama, S., Mizumachi, M., Morikawa, H., Aoyama, T.: Dolphin: A practical approach for implementing a fully distributed indoor ultrasonic positioning system. In: Proceedings of the Sixth International Conference on Ubiquitous Computing (UbiComp). (2004)
12. Shang, Y., Ruml, W., Zhang, Y., Fromherz, M.P.J.: Localization from mere connectivity. In: Proceedings of the Fourth ACM International Symposium on Mobile Ad hoc Networking and Computing (MobiHoc), ACM Press (2003) 201–212
13. Capkun, S., Hamdi, M., Hubaux, J.: GPS-free positioning in mobile ad-hoc networks. In: Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS), Washington, DC, USA, IEEE Computer Society (2001)
14. Freksa, C.: Using orientation information for qualitative spatial reasoning. In: Proceedings of the International Conference GIS—From Space to Territory: Theories and Methods of Spatio Temporal Reasoning in Geographic Space, London, UK, Springer-Verlag (1992) 162–178