Peer-to-Peer Coordination of Autonomous Sensors in High-Latency Networks using Distributed Scheduling and Data Fusion[*]

Lambert Meertens and Stephen Fitzpatrick

December 2001

Kestrel Institute Technical Report KES.U.01.09

Kestrel Institute, 3260 Hillview Avenue,
Palo Alto, CA 94304, USA
meertens@kestrel.edu & fitzpatrick@kestrel.edu

Project: e-Merge-ANT
http://ants.kestrel.edu/

Abstract

This report details an approach to the real-time coordination of large networks of short-range sensors that communicate over short-range, low-bandwidth, high-latency radio channels.

Each sensor is limited in the distance over which it can scan and in the type of data that it can acquire, so nearby sensors must collaborate to acquire complementary data for accomplishing such tasks as multiple-target detection and tracking.

Operational limitations on sensors and real-time requirements on tasks restrict the number of tasks in which a given sensor can collaborate. The quality with which a given task is accomplished and the cost incurred are affected by which particular sensors collaborate in achieving the task. Consequently, a coordination mechanism is required to optimize collaboration.

The coordination mechanism reported is fully distributed — each sensor decides for itself which measurements it should take to achieve optimal collaboration with nearby sensors, based what it knows about their intended measurements, and informs those sensors of its own intentions so that they can likewise optimize their own measurements.

In order to determine which measurements are optimal, a sensor must have knowledge about the targets that are the intended subjects of the measurements. To this end, each sensor exchanges measurement data with nearby sensors, and operates a data fusion process to maintain local target estimates.

The reported coordination mechanism is claimed to be scalable, low-cost, adaptive and robust.

Keywords: distributed constraint optimization, distributed scheduling, distributed resource management

[*]This work is sponsored in part by DARPA through the 'Autonomous Negotiating Teams' program under contract #F30602-00-C-0014, monitored by the Air Force Research Laboratory. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

Contents

List of Figures

1 Introduction

There is currently interest in using large networks of simple sensors to perform traditional sensing *tasks* (such as target detection and tracking) in a way that has low set-up and maintenance costs. For example, one scenario envisions the deployment of tens of thousands of small, cheap, battery-powered, short-range sensors that use low-power radios for communicating measurements and to coordination inter-sensor collaboration on target detection and tracking.

Inter-sensor *collaboration* is required because an individual sensor may be capable of scanning only a limited geographical area and acquiring only partial information about a target; for example, it may be capable of determining the distance from itself to the target, but not the direction. Consequently, several sensors need to collaborate to acquire complementary data that is *fused* to produce complete target estimates. To ensure high-quality data fusion, the sensors may need to take their measurements approximately simultaneously.

The objective of the network's *coordination mechanism* is to determine which sensors should collaborate and which measurements each should take — typically, there are many possible combinations of sensors that could achieve a given task (such as tracking a target) and each sensor could usefully participate in several tasks but it is limited in the number of tasks in which it can participate at any given time. The *quality* of the estimates produced by the data fusion process and the *cost* of achieving the given tasks (measured, e.g., in terms of the energy used) can be significantly affected by the coordination decisions.

The tasks that are to be achieved by the sensor network are dynamic:

- As targets move, the set of sensors that can scan the targets changes.
- New targets are detected by background scans, giving rise to new tracking tasks.
- Tracked targets move beyond the range of the sensor network, causing a tracking task to become obsolete.

Moreover, the status of the sensor network is typically dynamic; e.g., hardware may fail.

As circumstances change, the sensors must re-coordinate to adapt, and coordination itself must be robust against hardware failure. Because there are temporal requirements on the tasks (e.g., a given target will be within range of a given sensor for only a limited period), coordination must be achieved in real-time. Moreover, the number of sensors is expected to be high (e.g., 10^5) so coordination must be scalable.

Given these requirements, a highly-distributed coordination mechanism seems appropriate:

- The distance over which communication can take place in a single 'hop' is limited by the power of the radio transmitter. A distributed coordination mechanism allows most decisions to be made locally, using only single-hop communication, which in turn allows the network to adapt quickly.
- For non-trivial sensors, coordination is likely to be combinatorially hard. A distributed coordination mechanism allows the computational costs to be spread over many computational units. The objective is to have a scalable coordination mechanism, in which, e.g., the number of computational units grows linearly with the number of sensors and the load on each computational unit remains fixed.
- A distributed coordination mechanism is inherently more robust against hardware failure. For example, the failure of a few computational units should not cause the failure of the entire coordination network, and hence the entire sensor network.

Of course, distributing the coordination mechanism raises it own potential problems: each component of the distributed mechanism needs to adapt the behavior of the sensors under its control quickly enough to keep apace with changes in target number and/or behavior, but each component must also have regard for the actions being performed by nearby sensors (under the control of other coordination components) in order to ensure high quality collaboration between the sensors. That is, there is a need to balance speed of adaptation against ‘coherence’ of decisions made by autonomous but interacting components of the coordination mechanism.

The remainder of this report describes a highly-distributed coordination mechanism in which each sensor has its own coordination component, which interacts with other coordination components using a peer-to-peer, stochastic, hill-climbing optimization algorithm. An example involving simple radar sensors is given.

The structure of the remainder of this report is as follows:

- The radar sensor hardware is described along with some details of the communication infrastructure.
- The problem to be solved is defined in terms of the quality of target tracking that is expected to arise from a proposed vector of measurements, given current target tracks, models of target behavior and sensor models. It is argued that coordinating sensors based directly on this criterion is too computationally expensive for real-time systems.
- An alternative problem is introduced: optimizing the expected quality of sensing. Although sensing quality is, in this report, only heuristically related to track quality, it is conjectured that attaining high-quality sensing should lead to high-quality target tracks.
- A distributed coordination mechanism is then introduced that uses local information and local sensor interaction to approximately solve the problem of optimizing sensing quality.
- The main details of implementing the coordination mechanism for the radar example are given.

2 Example Application: Target Tracking using Simple Radars & Local Broadcast Communication

The distributed coordination mechanism reported here is intended to be mostly generic or at least readily adaptable to various types of sensors. Nevertheless, some design decisions are motivated by a particular application, involving simple radar sensors detecting and tracking standardized targets moving in a plane. This section describes the hardware and communication infrastructure, and some preliminary details about what measurements are needed for target tracking.

2.1 The Sensors: Simple Active Radars

The sensors are fixed-frequency, continuous-wave radars. A single sensor is comprised of three emitter-detector pairs and one sampler (an analogue-to-digital converter). Each detector can receive only the signal from its corresponding emitter after reflection off a target. At any given time, the sampler can measure the strength of the radar signal received by one and only one of the detectors. For a single measurement, the sampler is used in one of two possible modes:

- in *amplitude-only mode*, the sampler reports the maximum signal amplitude measured over a period of approximately 0.6 seconds;
- in *amplitude-and-frequency mode*, the sampler reports both the maximum signal amplitude measured and the Doppler shift between the emitted and reflected signals; the time required for this mode depends on the Doppler frequency and varies between approximately 0.6 and 1.8 seconds.

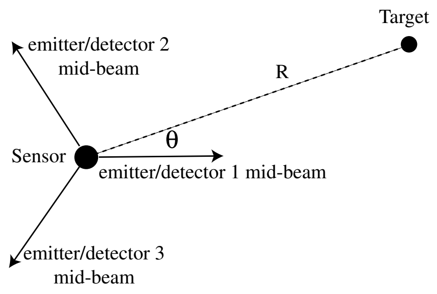


Figure 1: Signal strength detected by a radar depends on emitter-target distance and angle

The signal's amplitude corresponds to a set of target positions, as follows. The emitter-detector pairs are oriented at 120^o intervals in the plane (see Figure 1). The signal produced by an emitter is predominantly directed forward, along the emitter's *mid-beam*: the signal strength expected to be detected from a *standard target* is modeled by the equation

$$m(R, \theta) = Ke^{-(\theta/A)^2}/R^2 \tag{1}$$

where R is the sensor-target distance, θ is the angle between the emitter's mid-beam and the target ($0^o \leq \theta \leq 180^o$), and K and A are constants.

The emitters, detectors and sampler are subject to random noise (both internal and environmental) that limits the effective range of the sensors. The constant A is about 40^o so that, for example, a target that is at mid-beam will produce a signal that is $e^{9/4} \approx 9.5$ times larger than a target that is at the same distance from the sensor but that is 60^o off mid-beam.

If a radar beam reflects off several targets simultaneously, the reading produced by the detector is essentially a random combination of what the individual targets would produce separately, and is effectively useless.

The measured signal's frequency is Doppler shifted by an amount that is directly proportional to the target's radial speed.

Each emitter can be independently activated or deactivated. While an emitter is active, it consumes power, regardless of whether or not its detector is being sampled. If a deactivated emitter is activated, the emitted beam is unstable and does not give reliable measurements for approximately 2 seconds.

2.2 Target Tracking

The measurements taken by a single sensor are sufficient to allow target detection (the presence of a target nearby is indicated by a strong signal) but they are not sufficient to allow a target to be tracked: for example, a given signal amplitude indicates only that a target is at some point along a contour in the plane — it does not indicate where on the contour.

To narrow down a target's possible positions, more-or-less simultaneous measurements from several, nearby sensors must be combined using *trilateralization*, as shown in Figure 2. Of course, readings from sensors are subject to noise so new measurements for a target are combined with the target's *track* to achieve the best compromise between where the target is expected to be (based on its history) and where the sensors report it to be.

This report does not go into the details of the tracking/data fusion algorithm. Rather, it is assumed that one of the objectives of the coordination mechanism is to achieve simultaneous scans of each target and that estimates of target positions can be obtained by extrapolating tracks.

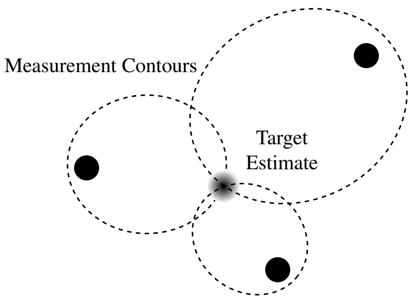


Figure 2: Determining a target's position using trilateralization

2.3 Broadcast Communication

Each radar sensor has a radio transmitter and receiver that can be used to communicate measurements and arbitrary data in short bursts (message lengths should preferably be kept below 100 bytes). In the demonstration application, communication operates on a fixed frequency in broadcast mode on a fixed, cyclic schedule: that is, in every cycle of the schedule, each sensor has a fixed time period during which it can transmit without interfering with other transmissions. The communication range is limited by the effective physical transmission range, so 'broadcast' mode is really a dense, local, multicast mode.

The number of sensors that are within communication range (and thus that are capable of interfering with each other's transmissions) is likely to be high relative to the number of messages that can be sent per second (the minimum duration of a message is determined by the likely size of messages and the communication bandwidth and latency). Consequently, the period of the communication schedule, and thus the average time between a sensor determining that it has information to send and being able to send it, is likely to be large.

In other words, communication is likely to have high *effective* latency even if the underlying communication fabric's latency is only moderate. This is expected to be true even if other, scalable communication schemes are used. High-latency communication is an underlying assumption of the rationale for the coordination mechanism reported here.

Two final issues should be noted in passing: (1) transmission interferes with sampling, so the communication and sensing schedules need to be compatible; (2) execution of the communication schedule is achieved autonomously by each sensor based on its local clock (which is synchronized with other sensors' clocks using a simple, distributed protocol whose communication piggy-backs on transmissions for measurements and other data).

3 World Estimates and Quality Metrics

The objective of a sensor network is to cost-effectively maintain an estimate of certain properties of the real world. For example, for a single target, a network's estimate may include the target's position, velocity, acceleration, facing (i.e., orientation relative to velocity), etc. The cost of operating the network may be measured as the amount of battery energy consumed, for example.

The following assumption is made:

Assumption 1 (Separable targets) *A world state is a finite map from some uninterpreted domain of* targets *to* single-target information.
In other words, the world is assumed to be made up of separately identifiable targets, for each of which

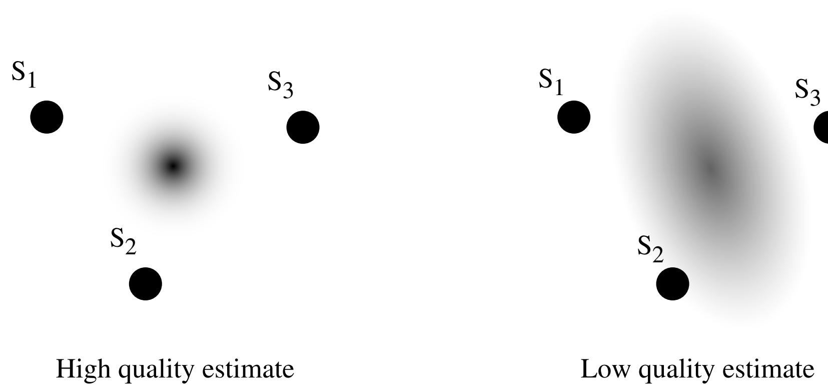


Figure 3: Quality of a position estimate reflects how accurately the position is known

independent information is represented. The set of targets for which information is represented by world state w is denoted by $\mathrm{dom}(w)$ and the information for target g is denoted by $w(g)$.[1]

The notation $e \equiv \{x \in X \rightarrow y\}$ is used to indicate that e is a finite map whose domain is X and that each x in the domain has corresponding value y. Thus, $w \equiv \{g \in \mathrm{dom}(w) \rightarrow w(g)\}$.

Typically, the state of the real world is not known precisely. A *world estimate* may be mathematically expressed by viewing the "true" real-world state as a random variable with a probability distribution over the space of world states. For convenience we assume that world states consist of continuous components (e.g., position and velocity), and that an estimate is continuous and may be given by a probability *density* function. (In general, world states may be discrete or hybrid, and the probability distribution could be any *measure* whose total is one, but note that such distributions may be approximated or may be represented by densities like Dirac functions.) If ω is the density function for some probability distribution over the space of world states W, then $\omega(w)$ is the probability density that the real world has state w (where $w \in W$).

The *quality* of an estimate quantifies, in some way, the accuracy of the knowledge conveyed, where high quality corresponds to high accuracy. For example, the quality of an estimate of a single target's position might be stated in terms of the variance of the position's probability density (see Figure 3). In general, assume that the quality of an estimate is quantified as a real number and let it be denoted by $\zeta(\omega)$.

Assumption 2 (Model of world evolution is given)
It is assumed that an estimate ω of the real world's state at some time t_0 can be *projected* to give an estimate $\epsilon_{t_0 t}\omega$ at another time $t > t_0$ according to a *model of world evolution* ϵ.

For simplicity of notation, the subscript 't_0' is omitted from such expressions as $\epsilon_{t_0 t}\omega$ since evolution is always relative to the time for which the estimate holds (which can be assumed to be represented as part of the estimate itself).

For example, suppose that a target is modeled as having a constant velocity and a constant velocity probability distribution, and that a world state represents the target's position and velocity: $w \equiv \langle p, v\rangle$. Then an estimate can be projected as follows.

- Let $\mathrm{vel}(\omega)$ denote the density function of the probability distribution for velocity, regardless of position: $\mathrm{vel}(\omega) \equiv \int \omega\langle p, v\rangle\, dp$. This distribution does not change: $\mathrm{vel}(\epsilon_t\omega) = \mathrm{vel}(\omega)$.

[1]This assumption is not strictly necessary for the abstract formulation developed in this section, but it is useful for pedagogic purposes.

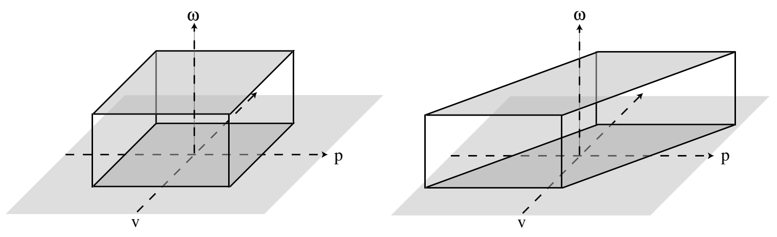


Figure 4: Projection of a uniform probability distribution over positions and velocities

- The mean of the position changes according to the usual linear model: $\overline{p}_t = \overline{p}_{t_0} + \overline{v}_{t_0}(t - t0)$, where $\overline{p} \equiv \iint p.\omega\langle p, v\rangle \; dp \; dv$ and $\overline{v} \equiv \iint v.\omega\langle p, v\rangle \; dp \; dv$.

  However, the uncertainty in the initial position estimate is augmented by the uncertainty in the velocity estimate, magnified over time. More precisely, since the velocity is known to be constant:

  $$\epsilon_t\omega \, \langle p, v\rangle = \omega\langle p - v(t - t_0), v\rangle \; .$$

  For example, Figure 4 (left) shows (the density for) a uniform probability distribution over positions and velocities in some range. Projecting this probability distribution causes each constant-velocity plane to be translated by an amount that is proportional to the velocity (in particular, the plane at $v = 0$ is unmoved). The resulting probability density is shown in Figure 4 (right). The total range of positions with non-zero probability density, and thus the uncertainty in the position, has increased.

In general, models of evolution are more complex; for example, generally no component is known exactly, several of the components may be mutually dependent, and evolution may be non-deterministic.

3.1 Trajectories

A *discrete world trajectory* (or just a *trajectory* for brevity) on an n-vector of times $\vec{t}$ (where $t_1 < t_2 < \cdots < t_n$) is an n-vector of world states $\vec{w}$ and has the meaning that the world has state w_i at time t_i.

For example, a simple trajectory, for a world in which only a single target is present, may be comprised of position and velocity coordinates at various times. In general, each element of a trajectory is a complete world state and may represent information about multiple targets.

The *a priori* probability distribution of a trajectory occurring, according to some initial world estimate ω and some model of evolution ϵ, can be computed as follows:

- The initial world estimate ω is projected to the time t_1 of the first world state: $\omega_1 \equiv \epsilon_{t_1}\omega$. Then the probability density of the first world state w_1 occurring is $p_1 \equiv \omega_1(w_1)$.
- The first world state is then projected to the time of the second world state to determine a conditional estimate ω_2 that gives the probability density of a world state occurring, given that w_1 occurs. A world state w can be projected by first lifting it to a single-point estimate $\widehat{w}$ (which has a Dirac probability density function that is everywhere zero except at some specified point, where the density is infinite) and projecting as usual: $\omega_2 \equiv \epsilon_{t_2}\widehat{w}_1$. Thus, the probability density of w_2 is $p_2 \equiv \omega_2(w_2)$.
- The probability density of the third world state is determined by projecting the single-point estimate of the second world state: $p_3 \equiv \omega_3(w_3)$ where $\omega_3 \equiv \epsilon_{t_3}\widehat{w}_2$. And so on.
- The overall probability density of the trajectory is the product of the individual probability densities.

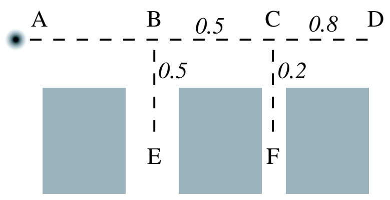


Figure 5: Possible trajectories through an environment of obstacles

Formally, a model of evolution ϵ induces a probability distribution with density function $\Omega_{\omega\epsilon}$ over the space of trajectories $\vec{W} \equiv \prod_{i\in\mathrm{dom}(\vec{t})} W$ on a vector of times $\vec{t}$, from an initial estimate ω over W, where

$$\Omega_{\omega\epsilon}(\vec{w}) \equiv \prod_{i\in\mathrm{dom}(\vec{t})} \omega_i(w_i) \tag{2}$$

where

$$\omega_i \equiv \begin{cases} \epsilon_{t_1}\omega & \text{if } i = 1, \\ \epsilon_{t_i}\widehat{w}_{i-1} & \text{if } i > 1. \end{cases} \tag{3}$$

For simplicity of notation, the subscripts on $\Omega_{\omega\epsilon}$ are omitted — it should be clear from context which estimate and which model of evolution pertain.

Figure 5 illustrates a simple example, in which a single target is moving through a grid of obstacles (for example, a ground vehicle moving along streets). The target starts at point A; at point B, the target has equal probabilities of continuing in a straight line to C, or making a right-angle turn to E; at point C, the target has a probability of 80% of continuing to point D, and 20% of turning to point F.

Assuming that only the target's position is of interest, the possible trajectories and associated probabilities are $p(\mathrm{ABE}) = 0.5$, $p(\mathrm{ABCF}) = 0.5 \times 0.2 = 0.1$, and $p(\mathrm{ABCD}) = 0.5 \times 0.8 = 0.4$. Note that trajectories such as ABEC and ABCDF are infeasible.

3.2 Measurements and Sensor Models

Because sensors are based on physical processes that are subject to noise, there is typically some uncertainty associated with *observations* or *readings*, which are the results of taking measurements. For example, if a target has a weight of 5.000 units, a sensor may report the weight as a random variable with log-normal probability distribution having mean 5.01 and variance 0.1. Let $m(w)$ denote the results of taking a measurement m when the real world has state w.

Assumption 3 (Sensor models given)
A *sensor model* ρ_{wm} is the *a priori* probability density function: $\rho_{wm}(r)$ is the probability density that reading $r \in R$ will be obtained for measurement m when the real world has state w.

It is assumed that there is a quality metric on proposed measurements $\phi_w(m)$ that reflects the expected quality of information obtained. For example, in an informal sense, a radar is expected to produce higher quality measurements for close targets compared with distant targets. A state may represent multiple targets and a measurement may acquire information regarding any subset of the targets, so it is assumed that the quality metric can be represented as a map over targets:

$$\phi_w(m) \equiv \{g \in \mathrm{dom}(w) \to \phi_{wg}(m)\} \ . \tag{4}$$

For the radar example, the basis of the measurement quality metric *for a single target* is Equation 1 which predicts how strong a signal should be obtained from a target. However, the quality metric should probably not be *directly* proportional to the signal strength: an example is given in Equation 13.

When multiple targets are present, interference may occur; i.e., the readings acquired by a sensor may not be a simple union of the readings that would be received for each target if it alone were present. A simple example of interference is one target blocking another target from the view of a line-of-sight sensor.

For *non-interfering targets*, $\phi_{wg}(m)$ can be defined for an individual target without regard to other targets; i.e., $\phi_{wg}(m) \equiv \phi_{w(g)}(m)$ where $\phi_{w(g)}(m)$ denotes the quality of a measurement for a single target in the absence of other targets.

For the radar example, interference can be accounted for by the following formula:

$$\phi_{wg}(m) \equiv \max \left[0, \phi_{w(g)}(m) - \sum_{g' \in \mathrm{dom}(w) \setminus \{g\}} \phi_{w(g')}(m) \right] . \tag{5}$$

- In the case where one signal is much stronger than the others: for the target that produces the strongest signal, the left $\phi_{w(g)}$ term will dominate and the resulting quality will be almost the same as if only this target were present; for the other targets, the sum term will dominate and the overall quality metric will be zero.
- In the case where there are several strong signals, the sum term will always be larger than or comparable to the left $\phi_{w(g)}$ term and the quality metric for every target will be low.

3.3 Data Fusion

The data fusion processor computes estimates of the real world from readings. Informally, an *on-line* or *real-time* data fusion processor D may be characterized by the equation $\omega' = D_\omega \vec{r}$ where ω' is a new estimate computed from an existing estimate ω and a vector of readings $\vec{r}$ acquired by measurements $\vec{m}$ at times $\vec{t}$.

Given a *proposed* vector of n measurements, let $\vec{R}$ denote the space of possible vectors of readings: $\vec{R} \equiv \prod_{i=1,\ldots,n} R_i$ where R_i is the space of possible readings for the i^{th} proposed measurement.

Let $P_{\omega\vec{m}}$ denote the density function of the probability distribution for the vectors of readings. This probability density is determined by the initial world estimate ω, the model of evolution ϵ and the sensor models ρ_i:

- The initial estimate and the model of evolution determine a probability density function Ω over possible trajectories on $\vec{t}$.
- For each trajectory, the probability density for the possible vectors of sensor readings is determined from the sensor models and the individual world states that comprise the trajectory.
- The overall probability density for a vector of readings is a weighted sum of the single-trajectory probability densities.

Given $P_{\omega\vec{m}}$, the *expected* quality of the new estimate is

$$[\![\zeta(\omega')]\!] = \int_{\vec{R}} P_{\omega\vec{m}}(\vec{r}).\zeta(D_\omega \vec{r})\, d\vec{r}\, . \tag{6}$$

3.4 Coordination Mechanism

The following assumption is made:

Assumption 4 (Uncoupled sensor network and targets) *The actions of the sensor network do not affect the evolution of the world state.*

There are two particular implications of this assumption that should be noted:

- The coordination mechanism cannot direct the targets of observation to improve measurement. This implication fits better with sensor networks observing adversaries, rather than, say, a manufacturing plant.
- The targets of observation do not change behavior as a result of being measured. It is in general possible that a target may detect that it is being illuminated by a radar, for example, and change course or attempt to jam the radar. In this report, such possibilities are ignored.

Under this assumption, the responsibility of a real-time coordination mechanism may be phrased as follows: determine a vector of measurements, to be taken over some reasonable (short) time span, that optimizes the trade-off between the expected quality of the next estimate and the cost of taking the measurements.

There are two general techniques that might be used to determine an optimal vector of measurements:

- Analysis — Given a model of evolution, sensor models, and a precise definition of the data fusion process, it may be possible to devise a simple algorithm that will directly determine optimal measurements for maximizing the quality of estimates and minimizing the cost of operation.
- Search — A finite approximation of the space of all feasible vectors of measurements can be exhaustively searched. For each vector of measurements, the initial world estimate, the model of evolution and the sensor models can be used to determine a probability distribution for vectors of readings; each vector of readings can be given to the data fusion processor and the quality of the resulting estimate computed; then the expected quality can be computed from the probability density function of readings for this vector of measurements. The coordination mechanism can use these predictions of quality to determine which vector of measurements optimizes the quality-cost trade-off. See Figure 6.

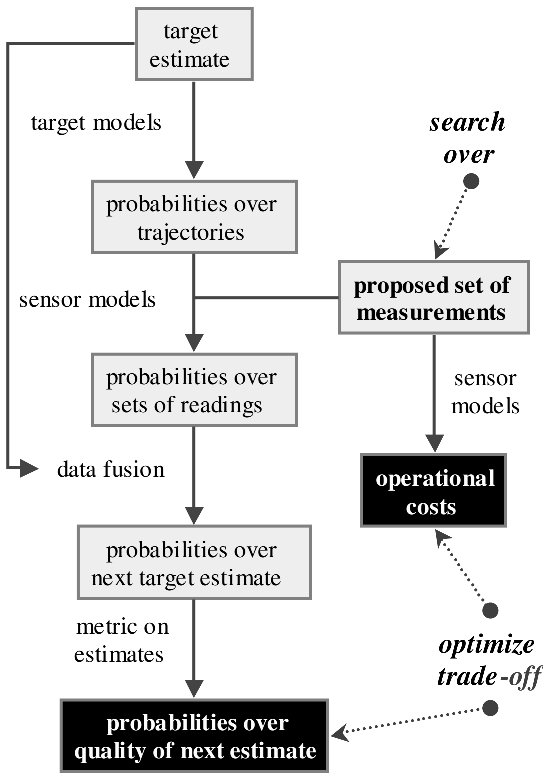
However, analysis is likely to be intractable for non-trivial sensor networks, and while search is theoretically possible for any arbitrary network, its computational costs are almost certainly prohibitive:

1. The space of feasible vectors of measurements is typically large.
2. The space of possible readings for each vector of measurements is typically large.
3. Computing the probability density of vectors of readings for a given vector of measurements is typically computationally expensive when done precisely, as it involves numerous convolutions.
4. Computing a new world estimate for each possible vector of readings is computationally expensive.

The combination of large search spaces and computationally expensive processes at each node of the search space typically make the search approach, in the form stated above, infeasible for real-time systems. An alternative is considered below.

4 Proximate Metric

As explained above, coordination based directly on maximizing the expected quality of world estimates is typically infeasible. However, the computational requirements of the search approach can be significantly



target estimate
target models
probabilities over trajectories
sensor models
probabilities over sets of readings
data fusion
probabilities over next target estimate
metric on estimates
probabilities over quality of next estimate
search over
proposed set of measurements
sensor models
operational costs
optimize trade-off

Figure 6: Optimization of trade-off between expected quality of estimates and operational costs

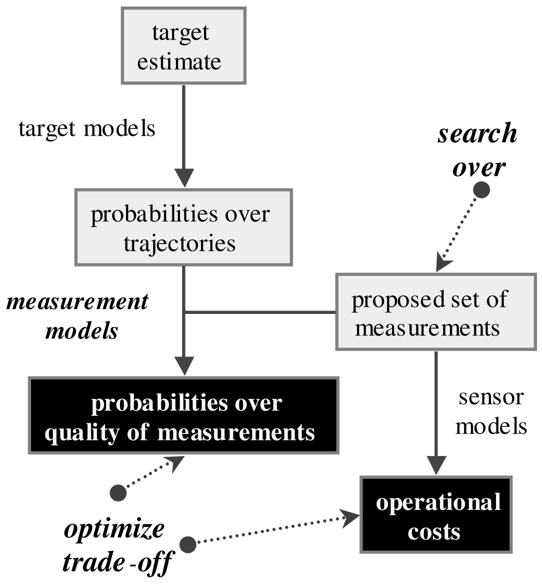


Figure 7: Optimization of trade-off between expected quality of measurements and operational costs

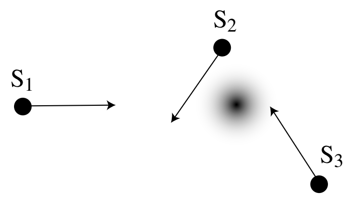


Figure 8: Distance and angle between target and emitter form the basis of a proximate metric

reduced by using a *proximate metric* that directly gauges the quality of measurements without computing new world estimates — see Figure 7. It thus eliminates the costs associated with items 2, 3 and 4 above, although it does introduce other, lower costs.

Conceptually, optimizing with respect to the proximate metric ensures that the sensors acquire data that is *likely* to lead to high quality world estimates. For example, Figure 8 shows three radars scanning a target that has a high-quality position estimate. An (overly-simplified) proximate metric, based on Equation 1 might award high scores to measurements made by sensor S_3 because, using the indicated emitter-detector, it has a good combination of distance and angle. In contrast, measurements from S_1 would achieve low scores because the distance is too large, and measurements from S_2 would achieve moderate scores because, although the distance is reasonable, the angle is high.

A more realistic proximate metric needs to account for how multiple measurements complement each other (e.g., tracking in the radar example requires simultaneous measurements from multiple sensors) and how well complete trajectories (not just single positions/world states) are scanned.

Of course, in many cases, given the difficulty of analyzing data fusion processes and the need for low computational costs, the proximate metric will be heuristic. Even so, it is likely that the coordination mechanism will need to greatly reduce the size of the search space to achieve real-time performance. But since the function of the proximate metric is to guide sensor coordination rather than, say, directly guide counter-measures, a lack of rigorous fidelity is likely to be acceptable.

In the remainder of this section, the proximate metric is considered in abstract terms — no regard is given to computational costs.

4.1 Proximate Metric with respect to Probability Distributions over Trajectories

A proximate metric Ψ measures the quality of a proposed vector of measurements $\vec{m}$, to be taken at times $\vec{t}$, given an initial world estimate ω and a model of evolution ϵ. Conceptually, Ψ is based on considering how well the proposed measurements suit possible trajectories, and forming some cumulative assessment for the probability density function Ω of possible trajectories on $\vec{t}$ induced by ϵ from ω. For example:

- Ψ may be based on a weighted sum over the probability density:

$$\Psi_\Omega(\vec{m}) \equiv \int_{\vec{W}} \Omega(\vec{w})\psi_{\vec{w}}(\vec{m})\, d\vec{w} \tag{7}$$

  where $\psi_{\vec{w}}(\vec{m})$ gauges the quality of a vector of measurements with respect to a single trajectory $\vec{w}$.

- Ψ may be based on how well the proposed measurements suit the most likely trajectory $\vec{w}^*$:

$$\Psi_\Omega(\vec{m}) \equiv \psi_{\vec{w}^*}(\vec{m}).\Omega(\vec{w}^*)\ . \tag{8}$$

While this form of metric is presumably less reliable than the weighted-sum form, it may be useful for simple target models because its computation may be much less expensive. This form will be used below in the radar example. (The quality metric includes a factor, $\Omega(\vec{w}^*)$, for how likely is the most likely trajectory because a trajectory that is highly likely to occur may warrant greater expenditure by the sensor network than a trajectory that is unlikely to occur.)

Regardless of which form is chosen, the basis of the metric is the single-trajectory metric $\psi_{\vec{w}}(\vec{m})$ — this is considered in the next section.

4.2 Quality of Measurements with respect to Single Trajectory

Given a trajectory $\vec{w}$ of world states at times $\vec{t}$, the quality $\psi_{\vec{w}}(\vec{m})$ of a proposed vector of measurements $\vec{m}$ (also at times $\vec{t}$) is a function of two terms: (i) the quality of data acquired by each measurement of its corresponding world state, and (ii) how well the measurements complement each other.

In general, these two terms are highly application-specific — they essentially are heuristics that attempt to characterize the type of sensing that is likely to lead to good tracking. Here, forms appropriate for the radar example are considered.

Section 3.2 refines term (i) into a map from individual targets in the world state to single-measurement, single-target metrics $\phi_w(m) \equiv \{g \rightarrow \phi_{wg}(m)\}$ and gives a form for $\phi_{wg}(m)$ that accounts for the possibility of target interference.

The form of term (ii) is based on the requirement that two or three measurements from different sensors be taken approximately simultaneously, which can be accommodated using a combination of two functions:

- persistence — a function that associates a measurement with the period of time over which it could usefully be combined with other measurements;
- adhesion — a function that, for a given time t, computes the combined quality that arises from all measurements whose persistence functions indicate that they can usefully contribute at time t.

These two functions and their combination are detailed below.

4.2.1 Persistence

A measurement's persistence function relates the measurement's *effective* quality at any arbitrary time to the measurement's peak quality (which occurs at the mid-point of the period over which the measurement was taken). A persistence function can have arbitrary form, but typically it drops off monotonically (to zero) with distance from the measurement's mid-point. For example, Figure 9 shows a 'triangular' persistence function.[2]

Specifically, for a measurement whose mid-point is at time t_0 and whose quality at t_0 is ϕ_0, the *effective* quality at time t is given by

$$\phi_t \equiv \phi_0 \pi(t - t_0) \tag{9}$$

where the persistence function π has range $[0, 1]$. Note that ϕ_0 is a map from targets to single-target quality metrics whereas $\pi(t - t_0)$ is a scalar, so computing their product involves scaling each single-target metric in ϕ_0 uniformly, as detailed in Section 4.2.4.

[2]'Persistence' is somewhat a misnomer since there is a period *before* the measurement is performed for which the function is non-zero.

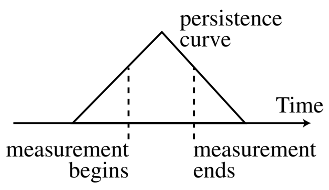


Figure 9: Persistence of a measurement

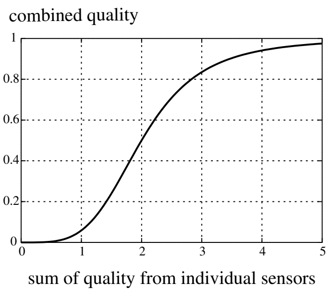


Figure 10: Non-linear adhesion function: $y = x^4/(2^4 + x^4)$

4.2.2 Adhesion

The persistence function effectively smears out a measurement over a non-instantaneous time period. For a given instant, the overall quality of measurement results from the combination of all measurements that have been smeared onto that instant: this is quantified as an adhesion function α that takes as argument a vector $\vec{\phi}$ of simultaneous, single-measurement quality metrics and computes a combined metric.

For the radar example, an appropriate adhesion function *for a single target* is shown in Figure 10:

- When the sum of the metrics for single-measurements is around two or three, the adhesion function awards a high overall quality.
- When the sum of the individual quality metrics is less than 1, the adhesion function awards a low overall quality. One consequence of this is that, if the coordination mechanism can arrange for more sensors to scan the target, it achieves a high *payoff* because the increase in overall quality is high compared with the increase in sensing costs.
- For higher sums, the overall quality awarded is also higher (the function is monotonic) but the rate of improvement rapidly decreases and the function asymptotically approaches 1. One consequence of this is that the coordination mechanism is discouraged from swamping a single target with many sensors, since the increase in quality would be low compared with the increase in sensing costs.

This adhesion function captures the informally-stated preference for two or three simultaneous measurements. However, it should be noted that the requirement is really for multiple measurement from *different* sensors. The persistence functions for measurements from the same sensor may be combined, e.g., by taking their maximum at each instant, as illustrated in Figure 11.

The adhesion function can be applied for every instant (over some appropriate time period), giving an overall time-quality curve, as illustrated in Figure 12.

For *multiple* targets, the adhesion function as described above applies to each single-target metric separately:

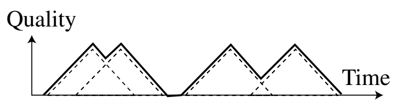


Figure 11: Overlapping measurements from the same sensor

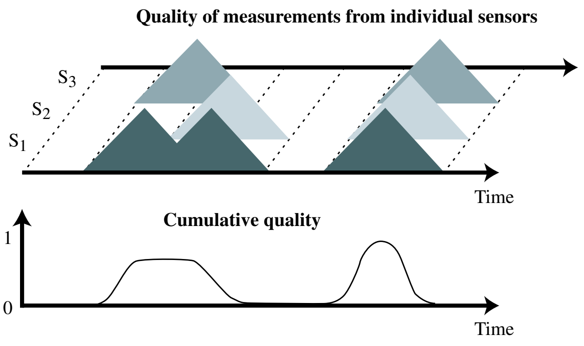


Figure 12: Overlapping measurements from different sensors

for each target and for each instant, the effective metrics (as determined by the persistence function) of all of the measurements are summed and a combined quality computed.

Let $G \equiv \bigcup_{i \in \mathrm{dom}(\vec{w})} \mathrm{dom}(w_i)$ be the set of all targets that are represented in any world state in the trajectory. Then for each $g \in G$:

$$\phi_{\vec{w}g}(\vec{m}, t) \equiv \alpha\left([i \in \mathrm{dom}(\vec{m}) \rightarrow \phi_{w_i g}(m_i).\pi(t - t_i)]\right) \tag{10}$$

where $\phi_{wg}(m)$ denotes the (peak) quality of a measurement m for the given target in the given world state (see Equations 4 & 5), and $\mathrm{dom}(\vec{m}) \equiv \mathrm{dom}(\vec{w}) \equiv \mathrm{dom}(\vec{t})$ by definition. An expression of the form $[i \in D \rightarrow f_i]$ denotes a vector having one element for each $i \in D$, whose value is f_i; the order of the elements is not important[3] because α is not dependent on the order (by assumption).

It is assumed that target information represented in separate measurements can be correlated; i.e., that if one measurement has information regarding some target that it labels A and another measurement has information regarding a target that is also labeled A, then both measurements actually concern the same real-world target.

In this section of the report, all targets arise from the evolution of a common (global) state so target correlation is moot. However, when state information is distributed/localized, correlation may need to be taken into account.

The following assumption, regarding multiple measurements that overlap in time, has so far been made tacitly:

Assumption 5 (Non-interfering sensors) *The results of a measurement made by one sensor are not affected by any measurements being made simultaneously by other sensors.*
That is, although the *usefulness* of a measurement may be affected by what other measurements are being taken simultaneously, the *raw signals* emitted/received by the sensors do not interfere.

In practice, if two sensors are known to interfere, a mutual-exclusion constraint can be imposed — constraints are discussed below.

[3]In other words, it denotes a finite map rather than a vector.

4.2.3 Overall Quality

To allow different proposed vectors of measurements to be compared, a single value can be derived representing the overall quality that is expected to accrue from the vector of measurements. One possible form for the overall quality metric is the sum over targets of the time-average of the instantaneous, single-target metrics:

$$\psi_{\vec{w}}(\vec{m}) \equiv \sum_{g \in G} \int_T \phi_{\vec{w}g}(\vec{m}, t)\, dt \tag{11}$$

where T is some reasonable time period encompassing $\vec{t}$.

4.2.4 Scaling and Adding Mappings

In the preceding sections, it was assumed that multiple-target metrics, which are structured values, can be scaled and added. This section defines these operations.

A single-state metric is a map from targets to single-target metrics. Targets can be represented by uninterpreted symbols (identifiers) and a single-target metric can be taken to be a real number. Thus, the type of a single-state metric is $\mathrm{target} \rightarrow \mathrm{real}$.

To scale a single-state metric, each single-target metric is scaled uniformly:

$$s\,\{g \in G \rightarrow v_k\} \equiv \{g \in G \rightarrow s v_k\}$$

where s is a scalar value.

To add two single-state metrics, terms for targets that occur in both metrics are added, and terms that occur in only one of the metrics are carried into the sum unchanged.

$$\{g \in G_1 \rightarrow v_{1g}\} + \{g \in G_2 \rightarrow v_{2g}\} \equiv \{g \in G_1 \cup G_2 \rightarrow v_{12g}\}$$

where v_{12g} is defined by cases:

$$v_{12g} \equiv \begin{cases} v_{1g} + v_{2g} & \text{if } g \in G_1 \wedge g \in G_2 \\ v_{1g} & \text{if } g \in G_1 \wedge g \notin G_2 \\ v_{2g} & \text{if } g \notin G_1 \wedge g \in G_2\ . \end{cases}$$

4.3 Measurement Feasibility

In the preceding sections, the quality of a proposed vector of measurements was considered under the assumption that the network could actually take all of the measurements. However, individual sensors and collections of sensors must observe some *constraints* that restrict the space of *feasible* vectors of measurements. In the radar example, two constraints are: (i) an emitter must be activated at least two seconds before a measurement can be taken; (ii) on a given radar, no more than one detector can be sampled at any time.

The emitter stability constraint relates a measurement to an *action* that is not a measurement, namely turning on an emitter. In general, constraints can involve any type of action the sensor network can execute. Consequently, the coordination mechanism must generate a *schedule* which assigns actions (not just measurements) to the network's resources for execution over specified time periods. Figure 13 shows an example schedule.

Constraints can be modeled in two ways:

Period	Resource	Action
0.00–0.01	radar 1	activate emitters 0 & 1
2.00–2.60	radar 1	sample detector 0
2.00–2.01	radar 2	activate emitter 2
2.60–3.20	radar 1	sample detector 1
3.20–3.21	radar 1	deactivate emitters 0 & 1
4.00–4.60	radar 2	sample detector 2

Figure 13: Example of a schedule

Hard constraints: a constraint is hard if its violation to any degree renders the violating measurements useless. For example, if a measurement is scheduled for an emitter that has been activated for 1.9 seconds, then no quality accrues from the measurement.

Soft constraints: a constraint is soft if its violation results in a *penalty* being assessed against the quality of the vector of measurements — typically, the magnitude of the penalty increases monotonically with the magnitude of the violation. For example, if the emitter activation constraint is modeled as a soft constraint, then a delay of 1.99 seconds before a measurement is taken would incur a small penalty compared with that incurred for a delay of only 1.9 seconds.

Soft constraints are particularly useful when transformational search techniques (such as hill climbing or simulated annealing) are used, in which an existing schedule is manipulated by local transform operations (such as swapping the order of two successive measurements, or transferring a measurement from one sensor to another). For such techniques, soft constraints smooth the search space and allow transformations to take place over schedules that would be forbidden under a hard constraint formulation, which in turn improves convergence properties (by reducing the number of small-scale, non-global optima).

Moreover, soft constraints tend to better capture physical systems, since there is typically some variance in physical processes: for example, it may not take an emitter *exactly* 2.0 seconds to stabilize every time. Consequently, all constraints in this report will be assumed to be soft.

Given a schedule of actions A and a set of soft constraints $\{p_i\}$, the total constraint violation penalty is the sum of the penalties for the individual constraints: $P(A) \equiv \sum_i p_i(A)$. (Note that each constraint can incorporate an appropriate weighting factor.)

4.4 Overall Quality (including Operational Cost)

Given a schedule of actions, it is typically straightforward to assess the cost of executing the schedule. For the radar example, the main cost is the energy consumed by emitters: when an emitter is active, it consumes energy at a constant rate, regardless of whether or not its detector is being sampled. This energy cost can be computed from the times at which the emitters are activated and deactivated.

The details of how to assess costs do tend to be application specific, so they will not be considered further until Section 6 which gives details for the radar example. However, it is assumed that, for a schedule A, the costs $C(A)$ can be assessed on a scale that is commensurate with the measurement quality $M_{\omega\epsilon}(A)$ and the penalty metric $P(A)$ so that an overall quality metric $Q(A)$ can be determined. For this report, a simple linear combination is assumed:

$$Q_{\omega\epsilon}(A) \equiv M_{\omega\epsilon}(A) - P(A) - C(A) \; . \tag{12}$$

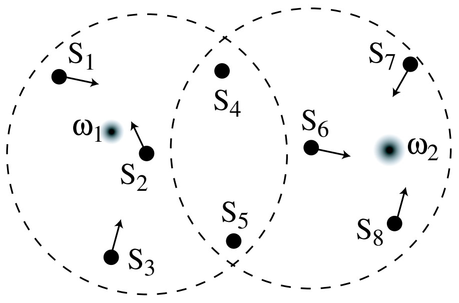


Figure 14: Fusion nodes in different parts of the network receive different measurements

An action schedule contains measurement and non-measurement actions. To determine its measurement quality, the non-measurement actions are expunged to give a vector of measurements $\vec{m}$ and $\Psi_\Omega(\vec{m})$ computed (see Section 4.1).

5 Coordination Mechanism

Equation 12 defines a metric function $Q_{\omega\epsilon}$ for assessing, with respect to an existing world estimate and model of evolution, the quality of a proposed schedule of actions to be executed by a sensor network.

The responsibility of the network's coordination mechanism may be loosely defined as determining a schedule that optimizes $Q_{\omega\epsilon}$. Given that a sensor network is subject to changing circumstances (such as hardware failing and targets not following predictions), a *real-time* coordination mechanism must continually update the network's schedule to maintain optimality as it learns new information about the real world.

In other words, adaptation of the sensor network to changes in the real world occurs through continual rescheduling. It would seem desirable that adaptation occur quickly, but this may not be possible in a distributed environment due to communication latency. How coordination can operate in a distributed environment is considered in detail in the following sections.

5.1 Local World Estimates

The coordination mechanism is based upon knowledge of the real world, as represented by a world estimate. In order to achieve a scalable, distributed coordination mechanism, it is necessary to distribute the world estimate as a collection of local world estimates and to maintain each local world estimate using only local measurements.

In this report, each sensor S_j is uniquely, tightly coupled with a *fusion node* that maintains a local world estimate ω^j that represents information about nearby targets. Each fusion node maintains its estimate based on readings acquired by its own sensor and on readings received from nearby sensors — each fusion node periodically broadcasts its sensor's measurements.

Since any two fusion nodes may be within communication range of different sets of sensors, the readings that they receive may differ, and consequently the local estimates they compute may differ. For example, Figure 14 illustrates two target estimates, ω_1 computed by sensor S_2's fusion node and based on measurements from S_1, S_2 and S_3, and estimate ω_2 computed by sensor S_6's fusion node and based on measurements from S_6, S_7

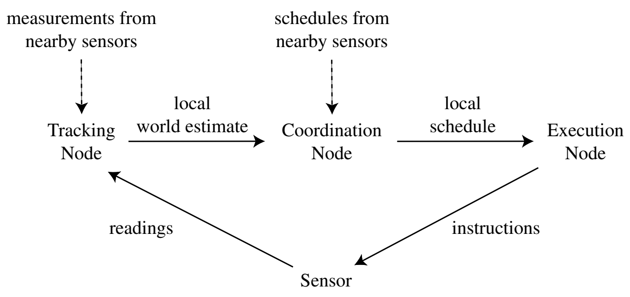


Figure 15: Components coupled with a sensor

and S_8. However, the differences between estimates computed by *nearby* fusion nodes are likely to be small, and not important as regards coordinating the sensors.

Just as the coordination mechanism continually updates sensor schedules, so too each fusion node continually updates its local estimate as it receives measurements from its own sensor and from nearby sensors. It is assumed that the implementation of this process is straightforward and it is not considered further in this report.

5.2 Local Schedules

Each sensor is uniquely, tightly coupled with a *coordination node* that maintains a *local* schedule of actions for the sensor to execute. That is, once a local schedule has been determined, it can be executed without further coordination with other sensors — all inter-sensor collaboration on measurements occurs through each sensor independently executing its own actions at the correct times (which it can accomplish using its local, synchronized clock).

This independent execution is vital to circumventing communication latency: individual sensor actions occur too frequently for them to be initiated by some mechanism that is not resident on the sensor itself. Of course, inter-sensor communication is still required to determine the local schedules, as discussed in the next section.

Figure 15 illustrates the interactions of various components associated with each sensor.

5.3 Local Schedule Quality Metrics

Since each local schedule A^j contains actions for only sensor S_j, the global schedule can, conceptually, be formed as the union of the local schedules: $A \equiv \bigcup_j A^j$. Consequently, the global schedule quality metric (see Equation 12) can be easily redefined in terms of local schedules.

However, the runtime *computation* of the metric in a distributed environment would require extensive communication and is not practical for a real-time application. Instead, coordination nodes compute local metrics that collectively substitute for the global metric.

Local metrics are based on the following assumption:

Assumption 6 (Encompassing communication) *The collaboration graph is a sub-graph of the communication graph.*

In other words, if two sensors are capable of usefully collaborating on scanning a target, then they are capable of communicating. For flat geographies and for homogeneous sensors, this assumption is satisfied if the communication range is twice the useful sensing range.[4]

Given this assumption, if every coordination node broadcasts its sensor's schedule then each coordination node will know the schedules of all sensors with which direct collaboration may be useful. A local quality metric can be based on this knowledge, as follows.

Let H^j be the set of (other) sensors with which sensor j can communicate. Let A^{jk}(where $k \in H^j$) be j's copy of k's schedule. Then the fragment of the global schedule that is known to j is $\tilde{A}^j \equiv A^j \cup \bigcup_k A^{jk}$ and j's local quality metric is $Q^j_{\omega^j \epsilon} \equiv Q_{\omega^j \epsilon}(\tilde{A}^j)$. Note that the metric is with respect to the local world estimate ω^j.

The global quality metric can, presumably, be expressed as a function of the local quality metrics (with appropriate account taken of the fact that each measurement may be incorporated into multiple local metrics). In general, the relationship between the global and local metrics is not simple. For example, consider a process in which each coordination node simultaneously changes its local schedule so as to optimize its local metric, and assume that a stable set of local schedules is found; i.e., for every j, A^j optimizes Q^j given A^{jk} $(k \in H^j)$. Even though every local schedule is, in a sense, locally optimal, there is no guarantee that the combined schedule optimizes the global metric.

Nevertheless, this is essentially the process that is followed by the distributed coordination mechanism described below. The fact that global optimality is not guaranteed is viewed as a sacrifice that is required to achieve scalability and real-time responsiveness. Whether or not this sacrifice is worthwhile can only be determined in the context of a particular application.

5.4 Distributed Coordination Mechanism

Given the comments of the preceding sections, the objective of a distributed coordination mechanism is to determine a set of local schedules A^j such that each local quality metric Q^j is optimized by A^j given j's copies A^{jk} ($k \in H^j$) of the local schedules of the sensors with which j might collaborate.

To maintain scalability and real-time responsiveness, the local schedules must be determined in parallel. However, doing so raises the danger of *incoherence* in which one node's copies of its neighbors' schedules are being rendered obsolete even while that node is trying to optimize its own schedule with respect to them.

For example, Figure 16 illustrates poor coordination between two sensors scanning two targets: this coordination is poor because, although each sensor scans each target, the scans of each target are not simultaneous. It is possible that the distributed coordination mechanism would cause just sensor S_2 to change its schedule so that it better matches S_1's schedule, resulting in the good coordination illustrated in Figure 17.

However, it is also possible that the coordination mechanism would cause both S_1 and S_2 to change their schedules simultaneously, each trying to match the other, resulting in the poor coordination illustrated in Figure 18. It is even possible that the coordination mechanism would subsequently cause the schedules to simultaneously revert back to those illustrated in Figure 16, and so on in a continual cycle of change without improvement known as *thrashing*.

One way to reduce incoherence to acceptable levels is to introduce a stochastic component to schedule updating: each coordination node periodically decides whether or not it should update its schedule by generating a random number that is compared against some fixed *activation level*; if the random number falls below

[4]To be strictly accurate, the useful sensing range should be extended with the largest distance a target could move over the time span of a schedule, since a target may move from this extended region into the collaborative sensing region.

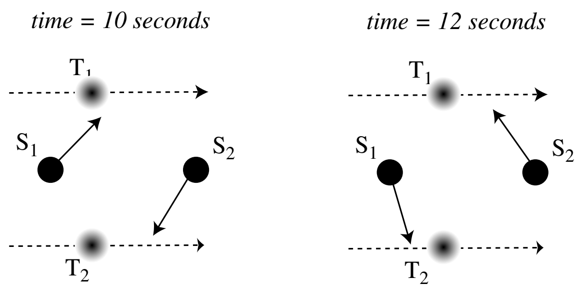


Figure 16: Poor coordination

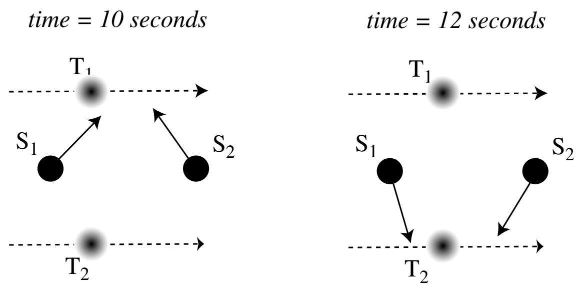


Figure 17: Good coordination

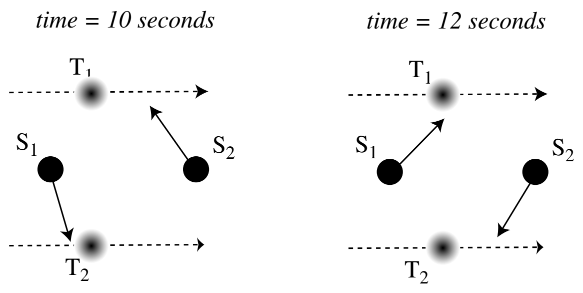


Figure 18: Different, but still poor, coordination

```
Each coordination node $j$ periodically executes the following:
  1. Generate a random number $r_j$.
  2. If $r_j < p$ (where $p$ is a constant):
     (a) Compute a local schedule $A^j$ that is optimal with respect to the local
         target estimate $\omega^j$ and the most recent schedules $A^{jk}$ received from
         nearby coordination nodes $k \in H^j$.
     (b) Broadcast the new schedule $A^j$.
```

Figure 19: Schedule update algorithm executed by coordination nodes

the activation level, the coordination node adjusts its local schedule to optimize it with respect to the last schedules it has received from other nodes.

By varying the activation level, the system designer can adjust the balance between the risk of incoherence and the speed of adaptation. This technique was studied experimentally in [2, 3] in the context of distributed graph coloring (which is also a distributed constraint optimization problem). Those experiments showed the technique to be:

- scalable: the costs for a given node are proportional to the number of sensors with which it collaborates, rather than the total number of sensors in the network;
- adaptive: as circumstances change, the coordination mechanism adapts collaboration accordingly;
- robust: the coordination mechanism was tolerant of infrequent but large-scale faults, and continuous but small-scale faults;
- low cost (in terms of communication): the per-node rate of broadcasts was low.

The distributed coordination mechanism's algorithm is summarized in Figure 19. (Recall that this algorithm is executed simultaneously with the execution of local actions and the updating of the local world estimate.)

6 Radar Example: Further Details

This section presents some details of the implementation of the coordination mechanism for the radar example, to illustrate how the concepts and principles presented in the preceding sections can be concretely applied.

6.1 Target Models

The targets are assumed to be deterministic and to move with constant velocities and, for coordination, it is assumed that the number of targets n is fixed over a single coordination step (this assumption is not made for data fusion). Each world state represents each target's position and velocity: $w \equiv \{g \in \{1, \ldots, n\} \rightarrow \langle p_g, v_g \rangle\}$.

For the proximate metric, the form (Equation 8) based on the most likely world trajectory w^* is used: $\Psi_\Omega(\vec{m}) \equiv \psi_{\vec{w}^*}(\vec{m}).\Omega(\vec{w}^*)$.

For a fixed number of multiple, independent targets, the most likely world trajectory, on a vector of times $\vec{t}$, is the combination of the most likely trajectories for each target. Since the targets are deterministic and have constant velocity, each target's most likely trajectory is given by

$$w_i^* = \{g \in \{1, \ldots, n\} \to \langle \bar{p}_g + \bar{v}_g.(t_i - t_0), \bar{v}_g \rangle\}$$

where $\bar{p}_g, \bar{v}_g$ are the means of target g's position and velocity components of the probability distribution at the start of the coordination step, at time t_0.

6.2 Schedules

A schedule A^j for a single sensor j has two components: a measurement schedule A^j_m and a non-measurement action schedule A^j_a. A schedule is a map from *time slots* to measurements or actions.

- Each time slot begins at an integral multiple of 0.6 seconds (relative to some arbitrary time origin) and lasts for 0.6 seconds.
- The real-world time corresponding to the mid-point of time slot k is denoted as $T(k)$.
- A time slot may contain no information, indicating that nothing happens during the corresponding time period (no measurement quality accrues and no cost is incurred).
- In the measurement schedule, each time slot may contain information for at most one measurement.
- An amplitude-only measurement requires a single time slot; an amplitude-and-frequency measurement requires three successive time slots.
- In the action schedule, each time slot may contain information for multiple actions.
- A single non-measurement action requires a negligible amount of time to execute — all of the non-measurement actions associated with a time slot are thought of as occurring at the start of the time slot.

6.3 Local Metrics

The task of a single coordination node is to assign measurements and actions to its schedule's time slots over some reasonable time period, say for the next 12.6 seconds (which corresponds to 21 time slots). The assignments should optimize the overall local quality metric, given the current local world estimate and current copies of other sensors' schedules.

Figure 20 summarizes the main steps in computing (an approximation to) a local measurement quality metric using the most-likely trajectory. To compute the measurement quality, the 'triangular' persistence function shown in Figure 9 and the adhesion function shown in Figure 10 are used. The non-zero region of the persistence function is taken to be 2 seconds, so the equation of the function is $\pi(\Delta t) \equiv \max[0, 1 - |\Delta t|]$. The integral of the quality over time in Equation 11 is approximated by a discrete sum over the time slots. The vector of times $\vec{t}$ on which the most-likely trajectory is founded is given by $t_k \equiv T_m(k)$.

The total local penalty metric can be computed by summing the penalties for each individual constraint. Efficient methods for computing the penalty for a constraint depend on the form of the constraint. For example:

- A cumulative constraint is easily checked simply by iterating through the time slots while maintaining a running total.
- A constraint that applies between successive measurements by the same sensor is easily checked given the time slot representation (since successive measurements occur in successive time slots).

1. **Compute target information.**
   For each target g compute the mean position $\bar{p}_g$ and velocity $\bar{v}_g$ according to the initial estimate ω (for time t_0).
2. **Compute single-measurement metrics.**
   For each time slot k in A_m:
   (a) Determine the expected position of each target: $p_g(k) \equiv \bar{p}_g + \bar{v}_g.(T_m(k) - t_0)$.
   (b) For each target g and each measurement m whose mid-point occurs in this time slot in the local schedule or in any of the schedules of other sensors, determine the single-target quality metric that would be awarded for that sensor if the target were present alone.
   - For an amplitude-only measurement:

     $$\phi_{mg}(k) \equiv \log_2 \left(1 + \frac{\max[0, m(R,\theta) - m_b]}{m_{\max} - m_b} \right) \tag{13}$$

     where R, θ are the distance between the sensor and the target and the angle between the emitter's mid-beam and the target, $m(R,\theta)$ computes the expected signal strength reflected from the target (Equation 1), m_b is a typical background noise level for the sensor that is to take the measurement, and $m_{\max}$ is the largest possible reading for that sensor.
   - For an amplitude-and-frequency measurement, the same formula applies, but the metric is increased by (say) 50% to take account of the additional data acquired.

   (c) Determine the multiple-target metric for the time slot using Equation 5.
3. **Compute multiple-measurement metrics.**
   For each time slot k:
   (a) Determine all measurements whose persistence function have a non-zero value at the time slot's mid-point (i.e., all measurements that occur within 1 second of the mid-point).
   (b) For each of these measurements, compute the effective quality metric at the time slot's mid-point by scaling the measurement's peak metric according to the persistence function (Equation 9).
   (c) Then combine all of the effective metrics according to the adhesion function, to give a single quality metric for each target (Equation 10).
4. **Compute an overall quality metric.**
   Sum the single-time-slot, multiple-target metrics (scaled by 0.6 seconds) over all time slots and targets to give one value that represents the overall quality of the schedule (see Equation 11).

Figure 20: Algorithm for computing a local metric on measurement quality

- A constraint between non-successive actions (such as the emitter stabilization constraint that requires a two second delay between an emitter being activated and a measurement being taken) may be more costly to assess. In general, each action can be interpreted as a transition function on an abstract model of the sensor. For example, if emitter 1 is off when the action 'activate emitter 1' is processed, the action causes the state of emitter 1 to switch to 'on' and causes a record of when 1 was last activated to be updated.

  By processing the actions in order, the effect of the schedule on the sensor can be simulated. Then the emitter stabilization constraint can be checked by determining the difference between each measurement's starting time and the most recent time the appropriate emitter was activated.

- Constraints between measurements on different sensors can be checked because the other sensors' schedules are known.

The cost of executing a schedule can be determined using a simulation of the schedule's effect on the sensor (as discussed above). For example, every time the simulation determines that an emitter is deactivated, an energy cost can be assessed based on when the emitter was last activated.

Note: in order to produce an accurate simulation, the state of the sensor at the start of the coordination step must be known. This can be determined either by querying the hardware for its true state, or querying some model maintained by the execution node.

6.4 Search

The size of the search space for even a local schedule is large. Neglecting everything except amplitude-only measurements, there are four possible choices for each time slot: a measurement one of the three detectors or no measurement. If the schedule extends over 21 time slots, the number of combinations is $4^{21} \approx 4 \times 10^{12}$. This is probably too large for a generative search technique even if pruning is used.

Consequently, a transformational search algorithm is used. An initial schedule is computed by independently choosing for each time slot an amplitude-only measurement that maximizes the single-measurement quality — collaboration, feasibility and operational costs are ignored. This schedule is then iteratively improved using hill-climbing on the full quality metric via transformations that include:

- Removing a measurement from a time slot, so that no measurement is performed.
- Changing the detector that is sampled in a time slot.
- Changing three successive amplitude-only measurements into an amplitude-and-frequency measurement.
- Inserting and removing emitter activation and deactivation actions.

(Of course, finite differencing techniques should be applied to the computation of the quality metric described in the preceding section, so that the quality can be cheaply recomputed after transformation.)

Given that the coordination mechanism is subject to real-time constraints, it may happen that it cannot complete its search in time. Fortunately, a feasible schedule can be quickly extracted for execution:

- Because measurements are assigned to discrete time slots, the constraint that only one detector can be sampled at any given time is automatically satisfied.
- The emitter stabilization constraint can be enforced by finding each emitter activation and removing all measurements involving the just-activated emitter that occur in the next three or four time slots.

The resulting schedule, of course, is probably not optimal (even in a local sense). However, as the execution node begins processing the schedule, the coordination node can continue to improve it. In particular, there is still time to change actions that are not scheduled to begin for another few seconds: consequently, the search should first focus on the initial section of the schedule and improve later sections only when the initial section has attained sufficiently high quality.

If the transformational search algorithm ranks candidate transformations according to potential quality improvement, then a simple way to assign more attention to the initial section of the schedule is to magnify its contribution to the quality metric. This has the advantage of not *entirely* neglecting later parts of the schedule.

Of course, it may happen that even with its ongoing, anytime search approach, the coordination node is simply incapable of producing schedules of sufficiently high quality in the time and with the computational power available. To determine if this is so, a detailed analysis of the probability distribution of schedules and consequent computational costs associated with the search and metric algorithms may be performed. Alternatively, the approach can be validated experimentally.

7 Conclusion

This report details a scalable, distributed, real-time coordination mechanism for local sensors interacting over a local, high-latency communication fabric. The general problem is defined as achieving high-quality estimates of the real world while minimizing operational costs.

Because the general problem is typically too computationally expensive for real-time systems, a simpler problem is introduced — achieving high-quality sensing while minimizing operational costs. It is hoped that solving the simpler problem will lead to world estimates of acceptable quality.

An approximate form of the problem of achieving high-quality sensing is defined for distributed systems in which communication latency requires all interaction to be local. A practical implementation is considered for a network of simple radar sensors.

An experimental assessment of the coordination mechanism for the radar network should be available in a subsequent report.

It may be speculated that the heuristic components of the definition of sensing quality could be made rigorous in information-theoretic terms by considering entropic metrics on world estimates and the amount of information conveyed by measurements, under the assumption that high-quality estimates are those that have high information content. This would be an interesting topic for further research.

References

[1] *The ANTs Challenge Problem*, http://ants.kestrel.edu/challenge-problem/index.html

[2] *Soft, Real-Time, Distributed Graph Coloring using Decentralized, Synchronous, Stochastic, Iterative-Repair, Anytime Algorithms. A Framework*, Stephen Fitzpatrick & Lambert Meertens, Technical Report KES.U.01.05, Kestrel Institute, May 2001

[3] *An Experimental Assessment of a Stochastic, Anytime, Decentralized, Soft Colourer for Sparse Graphs* To appear: Proceedings of SAGA 2001, 1st *Symposium on Stochastic Algorithms, Foundations and Applications*, Berlin, 13-14 December 2001, Springer-Verlag.