A Game Theoretic Model for Management of Mobile Sensors

L. Ronnie M. Johansson* Ning Xiong Henrik I. Christensen
Centre for autonomous systems,†
Department of Numerical Analysis and Computer Science,
The Royal Institute of Technology (KTH),
SE-100 44 Stockholm, Sweden

25th June 2003

Abstract

This report suggests a framework for multi-sensor multi-target tracking with mobile sensors. Sensors negotiate over which targets to track (possibly sharing targets to benefit from data fusion technology) using a game theory based algorithm. Sensors' preferences over negotiation offers are articulated with individual utility functions which encompass both information gain and directional derivative. An approach to consider terrain effects on mobile sensors is also explained. Simulation results show that the negotiation algorithm has interesting advantages compared to a greedy algorithm that seeks to optimise information gain without consideration to derivatives. We notice that the negotiation procedure forces sensors to share targets, while improving robustness to sensor failure. Sensors also tend to proactively reconsider their target assignments for long-term improved information gain.

*Tel: +46-8-790-6724, Fax: +46-8-723-0302, E-mail: rjo@nada.kth.se

†This work was financially supported by the Swedish Defence Research Agency, Division of Command and Control.

Contents

1 Introduction

Mobile sensing resources (or *mobile sensors* for short) provide a flexible aid to decision support systems for decision-making in dynamic, extensive environments. Their sensing capabilities contribute with observations to the decision support system and their mobility allow them to adapt to changing information needs and altered mission requirements.

Sensors are a limited resource and to achieve good performance in a system with mobile sensors, allocation and use of sensor is a key aspect to consider. *Sensor management* is the process that aims at controlling sensors to improve overall system performance [NN00]. Typical factors of concern for a practical sensor management system are probability of target detection, track/identification accuracy, probability of loss-of-track, probability of survival, probability of target kill, etc [XS02].

One aspect of managing mobile sensors is coordination of their actions. Choosing a *centralised* approach to coordinate the system promises to provide the system with optimal coordination. However, such a system is both vulnerable (e.g., if the centralised control node is destroyed, the whole system will fail) and slow (e.g., sensors have to await orders from the centralised control). These two factors are essential for systems operating in civilian applications, and even more so in a military application since the environment is expected to be hostile and willing to exploit the two drawbacks (e.g., by jamming communication or targeting the centralised control). Decentralised control, on the other hand, assumes that the system is mainly controlled by its components (e.g., mobile sensors), allowing it to "degrade gracefully" if some of its components fails. However, achieving good performance with decentralised control is a, by far, greater problem.

Distributed artificial intelligence (DAI) is a research field that concerns itself with coordinated interaction among distributed entities, known as *agents* [Wei99]. *Game theory*, constituting a toolbox of methods for analysing interactions between decision makers [OR94], has attracted a lot of attention from the DAI community. Interestingly, game theory concepts seem to apply better to automated agents than to the real-life human decision-makers for which it was originally intended [ZR96]. The reason is simply that the agents are normally both rational and obedient, qualities which rarely apply to their human counterparts.

Game theory offers models for distributed allocation of resources and provides at the same time mechanisms to handle uncertainty. An important subtopic of game theory is *negotiation*. As part of negotiation there are ways to generate multi-objective optimisation results that are at least Pareto optimal. At the same time, these methods allow for robust handling of game/agent configurations which makes it robust to jamming and use of sensors with limited availability.

Works in DAI seldom consider uncertainties [CFK97] such as those imposed by the physical world (e.g., estimation errors) which are inherent to target tracking applications. Noteworthy recent exceptions concerning target tracking include [DN01] and [CLOHd$^+$01]. In [DN01], stationary sensors form coalitions (groups) where each coalition track a certain target. The members of a coalition fuse their measurements to improve target state estimation. In [CLOHd$^+$01], mobile sensors form coalitions to track targets, each sensor capable of sensing one target at a time. Movements of sensors are decided by a hierarchy of coalition leaders, each responsible for a certain geographical area.

The management, in our approach, is performed using negotiation models from game theory. We utilise an algorithm for agent negotiation which we have previously

developed and evaluated [XCS03]. In the previous work, sensor agents negotiated about which targets to track, dividing the set of targets among themselves. Sensors were stationary, but now we apply the same algorithm to the case with mobile sensors and allow sensors to share targets. Our work constitutes a framework for future studies of management of distributed mobile sensors in the face of uncertainties and sensor failures.

In future work, we want to show the advantages of the game theoretic approach when faced with uncertainties and possible sensor failures.

The next section will place this work in the context of prior work. Section 3 presents the primary objectives of this work, including management of mobile sensors and sharing of targets. Section 4 explains the negotiation procedure and its utility functions. Section 5 presents some results of using the negotiation strategy and, finally, Section 7 concludes and suggests future research.

2 Target allocation using sensor agent negotiation

The work that precedes the work described in this paper considered a sensor network consisting of geographically dispersed, non-mobile sensing resources [XCS03]. The sensing resources (sensors) were expected to cooperatively monitor some geographical area to keep track of all targets known to be within that region. A solution to the problem would be a partition of the target set into disjoint subsets, and an assignment of subsets to sensors so that every subset was assigned to exactly one sensor. In other words, a valid solution required that every target be tracked by exactly one sensor. However, sensors were allowed to track more than one target each.

A solution algorithm based on the game theory concept of negotiation was proposed and the utilities of negotiation offers were calculated from the information gain (explained in Section 2.2) the corresponding target division was expected to bring.

2.1 Agent negotiation

There are two kinds of consequences of an agent negotiation: agreement and disagreement. Disagreement means that no solution acceptable for all agents can be reached. In the other case, an agreement between the agents can be reached.

Every agent has its own preference relation over possible agreements and times of agreements. We assume that a sensor agent $i \in S$, S being the set of sensor agents, has a utility function, U_i, which represents its preference relation. The utility function assign values to all possible outcomes of a negotiation: $\{O \times \{0,1,\ldots,K\} \cup \{Disagreement\}\}$, where O is the set of possible offers and K refers to the final step of negotiations.

As agents negotiate in order to realize cooperative behaviours among themselves in multi-target tracking, reaching an agreement is in line with the interests of all agents and no one can benefit from disagreement (i.e., non-coordinated behaviour[1]).

A formal description of the negotiation game we study is the 5-tuple $< S, O, H, P(H), (U_i) >$, where

S is the set of sensor agents (called players in game theory terminology)

[1] Clearly, non-coordinated behaviour (i.e., sensors tracking whatever target they like, disregarding the allocations of the other sensors) would be unlikely to meet the system requirements, such as ensuring that all targets are tracked.

O is the set of *negotiation offers*, i.e., the possible sensor to target allocations. A member of O, o, is an allocation function that maps sensors to subsets of the set of targets, T, i.e., $o : S \rightarrow 2^T$.

H is the set of sequences of offers and responses (called *histories* in game theory) in a negotiation. A non-terminal history is a sequence, for instance (o_0, R, o_1, R), which ends with a rejection, R, preceded by a series of consecutive offers (o_t is the offer a step t in the negotiation). A terminal history, on the other hand, ends with an agreement, e.g., (o_0, R, o_1, R, o_2, A).

$P(H)$ is a function that determines which agent has the turn to make an offer after a non-terminal history h.

(U_i) are utility functions of sensor agents $i \in S$ over outcomes, which describes how the agents value every allocated group of targets

It is assumed that at a particular step of a negotiation, one of the agents makes an offer and the other agents respond to it by acceptance or rejection. The order in which the agents make their proposals is specified before the negotiation begins. The first action in the game occurs in step zero when one agent makes the first offer and the other agents accept or reject it. Acceptance by all other agents ends the game with agreement while rejection by at least one other agent forces the game to continue with another step. Subsequently, another agent proposes something in the next step which is then accepted or rejected by the others. The game continues in this manner until an agreement has been reached or until the final step K. If no agreement is reached at step K, we say that the game ends with disagreement.

A *negotiation strategy* for an agent is essentially a function that specifies what the agent has to do after every possible history.[2] Concretely, the strategy prescribes what offer to make when it is the turn of the agent to make an offer, and whether to accept or reject an offer in steps when the agent is to respond to a proposal made by another agent. A *strategy profile* is a collection of strategies for all involved agents. We would like to find a strategy profile leading to an outcome that is profitable for all participants and that no agent can benefit from using a strategy not belonging to the profile.

A fundamental concept for analysing behaviours of rational agents is the *Nash Equilibrium* [Nas53]. A strategy profile of a game of alternating offers is a Nash Equilibrium if no agent can profit by deviation given that all other agents use the strategies specified for them in the profile. Unfortunately, simple Nash Equilibrium seems not sufficient in extensive games[3] in the sense that it ensures the equilibrium of its strategies only from the beginning of the negotiation, but may be unstable in the intermediate stages.

A stronger notion for extensive games is that of *subgame perfect equilibrium* (SPE) [OR94] that requires that the strategy profile included in every subgame is a Nash Equilibrium of that subgame. This is a comprehensive concept implying that agents are rational at any stage of the negotiation process: no one can be better off by using another strategy regardless of what happens. It was shown in the preceding paper that if all agents honour SPE strategies, there is an offer made in the first step which is preferred by all parties over all possible future outcomes.

[2] Thus, *strategy* is similar to the concept of *policy* in AI-literature

[3] In extensive games, agent actions are performed in sequence as opposed to strategic games, or one-shot games, where actions are performed simultaneously.

When reasoning about strategies, we start at the final step of negotiations, K. If no agreement has been reached yet, one will certainly be reached in the final step since disagreement (which would be the outcome of a rejection of the final offer) is the worst outcome for all agents and will be avoided. If it is agent i's turn to make an offer at step K, it will choose the offer that is best for its own payoff, and the other agents accept this offer, since disagreement is the only alternative.

At all steps before K, the agent, whose turn it is to make the offer, will consider the agreement that would be reached at the next step and proposes something that is better or at least as good (in terms of utility) for all agents than what they can expect to attain in future steps.

Suppose $o^*(t+1)$ represents the agreement that will be reached at step $t+1$. Given that the agents are rational, then for all parties at time t the $super(t)$-set defines the set of acceptable solutions:

$$Super(t) = \{o \in O \mid \forall i \; U_i(o,t) \geq U_i(o^*(t+1),t+1)\} \tag{1}$$

If an agent selects its offer from the $Super$-set in Equation 1, then all other (rational) agents will accept it since no offers with higher utility will be offered in future steps.

Furthermore, it is important to notice that the $Super$ set is non-empty for all steps before K. This is induced from the characteristic that the utilities of offers decrease over time. Particularly, the offer $o^*(t+1)$ is included in $Super(t)$ since we have $U_i(o^*(t+1),t) > U_i(o^*(t+1),t+1)$ for all agents i.

The $Super$ set of acceptable offers is very useful to establish SPE strategies at steps before K. The non-emptiness of this set ensures that the agent whose turn it is to make an offer has enough choices to make its proposal acceptable to the other agents. The strategy we use is that the agent i whose turn it is to make an offer at step t will propose the offer $o^* \in Super(t)$ that maximises $U_i(o,t)$. If several candidate offers maximise U_i, (this set was denoted $Compet(i,t)$, in the preceding work) we let agent i propose the offer that not only maximises $U_i(o,t)$ but also the sum of the utilities of the other agents, i.e., $o^*(t) = \arg\max_{o \in Compet(i,t)} \sum_{k \in S \setminus i} U_k(o,t)$.

Finally, the fact that $U_i(o(t),t) \geq U_i(o(t+1),t+1)$ for all agents causes the game to end already in the first step with agreement $o^*(0)$.

2.2 Sensor performance utilities

We assume that the sensors track targets using a Kalman filter and let the utility functions of the agents, U_i, depend on the decrease in uncertainty that is estimated in the Kalman calculations.

We will not present the entire Kalman filter method here (instead see, e.g., [BSF88]), just simply point out what part of the method was used in the previous work to derive a utility measure for target tracking sensors.

In the Kalman filter method, we let $x_j(k)$ represent the system model of target j at time k, and $y_j(k)$ the corresponding measurement model. The following familiar equations are used:

$$\begin{aligned} x_j(k) &= F_j x_j(k-1) + w_j(k-1) \\ y_j(k) &= H_{ij} x_j(k) + v_{ij}(k) \end{aligned} \tag{2}$$

In Equation 2, $w_j(k)$ and $v_{ij}(k)$ are system and measurement noise, respectively.

An expression for the update of the target estimation error covariance reveals the measure of performance [BSL93] which we use:

$$P_{ij}^{-1}(k|k) = P_{ij}^{-1}(k|k-1) + H_{ij}^T R_{ij}^{-1} H_{ij} \tag{3}$$

Here, $P_{ij}(k|k)$ is the updated state estimation error covariance and $P_{ij}(k|k-1)$ is the predicted covariance. The matrix $H_{ij}^T R_{ij}^{-1} H_{ij}$ is derived from sensor characteristics, where R_{ij} is the measurement noise covariance. Note that a decrease in the measurement noise covariance also leads to a decrease in the state covariance, i.e., a reduction of uncertainty about the target state. In view of this, we define the norm of the matrix $H_{ij}^T R_{ij}^{-1} H_{ij}$ as *sensor information gain*, $g(i,j)$, contributed by sensor i on target j.

$$g(i,j) \triangleq \begin{cases} \|H_{ij}^T R_{ij}^{-1} H_{ij}\|, & \text{if sensor } i \text{ tracks target } j \\ 0, & \text{if not} \end{cases} \tag{4}$$

By means of sensor information gain, we establish the measure of performance of a sensor estimating properties of all targets assigned to it. Suppose sensor i is in charge of a group of targets, D_i, then its contribution to the global picture is accrued by measuring all assigned targets. Hence, the performance of sensor i, P_i, is defined to be the sum of these information gains for state estimates of targets in D_i,

$$P_i(D_i) \triangleq \sum_{j \in D_i} g(i,j) = \sum_{j \in D_i} \|H_{ij}^T R_{ij}^{-1} H_{ij}\| \tag{5}$$

We call a value given by the expression $P_i(D_i)$ for the *sensor performance* of sensor i when tracking a group of targets D_i.

We note that an offer is a distribution D of targets among sensors, $D = \bigcup_i D_i$, i.e., each sensor i gets a subset of targets D_i to track. For every sensor, the acquired set of targets corresponds to a value of sensor performance using the definition in Equation 5, and we will now explain how we use the sensor performance value, for a sensor and a set of targets, to calculate the corresponding utility value.

An agent is assumed to receive a *reward* not more than unity in terms of its contributed performance. The purpose of doing so is to normalise the sensor performance value for easy handling and to allow for non-zero rewards for sensors that accept no work (i.e., do not track any targets). The reason to allow sensors to be "lazy" is to encourage them not to reveal themselves (by use of active sensors) too often, i.e., to be *quiescent*. The reward r_i of sensor i, appointed target group D_i, is given by

$$r_i(D_i) \triangleq \alpha + (1-\alpha)(1 - e^{-\beta \cdot P_i(D_i)}), \quad 0 \leq \alpha < 1 \text{ and } \beta > 0 \tag{6}$$

such that $P_i \in R^+$ is converted into a regular interval $[\alpha, 1)$. Here, β is a parameter which decides how eager the sensor is to acquire more information about a target. A high value on β means that the sensor agent is satisfied with less certain state estimates (cf α_j in Figure 2). The other parameter, α, controls the agent's willingness to differentiate between the offers. For instance, $\alpha = 1$ means complete indifference, i.e., all offers have the same value ($r_i(D_i) = 1$).

As explained in Section 2.1, utility is expected to decrease over time in the negotiation, and we therefore define the time-dependent utility function in this way:

$$U_i(D,t) \triangleq (K - t + 1) r_i(D_i). \tag{7}$$

3 Primary objectives

In this paper, we extend the previous work discussed in Section 2 considering the following three aspects:

Mobile sensors We allow sensors to move to increase sensor performance. We further allow the characteristics of the terrain to affect the *preferred direction of motion*.

Shared targets We extend the previous work by allowing sensors to track the same targets (previously, the targets were divided between the sensors). Through use of multiple sensors tracking the same target, it is possible to improve the performance on state estimates as typically found in the multi-sensor tracking and multi-sensor fusion literature (e.g., [BSF88]). Here, this problem is studied in the context of target assignment and performance optimisation. We model that the value of tracking a target, which is already being tracked by other sensors, is less than if none tracks the target.

Performance loss when tracking many targets We model that the measurement performance on each target tracked by a sensor decreases with the number of targets tracked by the same sensor. The reason is of course that the sensor has limited time and resources for its measurements and if it has to track more targets and divide its resources among the targets, then also the measurement error covariance will increase for every target (and sensor gain decrease).

In order to allow the mobility of sensors to have any effect, we further assume that sensor platforms have the ability to move at a speed that is comparable to the speed of the targets.

4 Utility and Negotiation for Mobile sensors

There are only small differences between the game considered in the previous work (defined in Section 2.1) and the one we consider here. Sensor agents negotiate by making offers that the other agents might accept or reject. As in the previous work, an offer, o, is a specification of allocations, that assigns groups of targets to sensors. Unlike the previous work, the target groups may overlap, significantly increasing the number of valid offers. The other difference is in the utility functions, which, we shall see in the next section, has a two-dimensional values.

For negotiation about target allocation of mobile sensors, we consider both the reward for each sensor as well as its *directional derivative* in the preferred direction of motion. The reward, as we will see, is calculated somewhat differently than in the previous work and does not immediately yield the negotiation utility. The preferred direction of a sensor platform is the spatial direction in which the sensor would like to travel. When we do not consider terrain characteristics, the preferred direction will simply coincide with the gradient of the reward function.

In the next section, we will first present an approach to consider mobility in the negotiation. In the subsequent sections 4.2-4.4, we will discuss how to calculate both the reward for the novel considerations of overlapping target groups and decreased tracking performance, and the preferred direction. We also address the resulting multi-objective optimisation problem.

4.1 Negotiation

Before we start to discuss the details about reward and directional derivative, we will, for this work, assume that every sensor agent has the required information and is cap-

able of calculating both objectives for all sensors.[4] Hence, given a sensor agent $i \in S$ and an offer of allocations of sensors to targets $o \in O$, we can calculate reward $r_i \in \mathbb{R}$ and directional derivative $r'_{i,\delta} \in \mathbb{R}$, i.e., a sensor and an offer yields a reward and preferred direction, $S \times O \rightarrow \mathbb{R} \times \mathbb{R}$. Here $\delta \in \Delta$ is the preferred direction, and Δ the set of unit vectors.

We want to consider both factors, reward and derivative (measured as change in reward per length unit), simultaneously to acquire a combined utility metric. A valid but tentative approach is to assert a utility function $U = U(f(r), g(d))$ that analytically combines the two. However, the factors are incommensurable, and, hence, such a function is sensitive both to the application in question and the choice of measurement unit. E.g., we might propose $U(r,d) = r + d$. While this utility function might yield satisfying results for some applications, it will certainly not do so in general. Rather, the appropriate functions (f and g) have to be found for every specific application or class of applications.

The problem we are facing is that of multi-objective optimisation. Whereas elaborate approaches to this problem has been proposed (such as [FF98]), in this work we prefer to study the results of an approach that does not suggest a preference of one factor over the other. (Hence, it might not work optimally for every application, but is expected to work well for every application.) We order the offers only according to *dominance*.

A sensor agent, i, will prefer an offer o_1 to another offer o_2, $o_1 \succ_i o_2$, if and only if o_1 dominates o_2. An offer o_1 can only dominate another offer o_2 if one of the reward and directional derivative values of o_1 is greater than the corresponding value of o_2 and the other one at least as great as its counterpart, i.e., $r_i(o_1) \geq r_i(o_2)$ and $r'_{i,\delta}(o_1) \geq r'_{i,\delta}(o_2)$ and at least one of the inequalities should be strict. If neither o_1 nor o_2 dominates the other, we write $o_1 \sim_i o_2$.

We elaborate further on the topic of dominance. Figure 1 shows twenty offers, here depicted with circles, plotted in a graph according to the reward and derivative in the preferred direction of a certain agent (certainly, the plot would look different for another agent). We find that there are, in this example, five offers that are not dominated by any other offer. We conclude that these are the "best" offers the agent could get. We call the set (or *class*) of these the *offers of the first order*. We iteratively classify the rest of the offers, knowing that an offer o_k, which is dominated by an offer o_l of order l, will be a member of order $l+1$ or greater. Each offer in Figure 1 belongs to one of five orders and the members of each order are connected to each other with dashed lines for illustration.

A more formal definition of class of offers for a particular sensor is as follows.

Definition: Class of offers All pairs of offers (o_1, o_2), $o_1, o_2 \in O$, that fulfill the condition that $o_1 \sim o_2 \wedge \neg \exists o_m \in O\, [(o_m \sim o_j \wedge o_m \succ o_k)]$ for $j \neq k$ and $j,k \in \{1,2\}$ are said to belong to the same class of offers.

A class may not be empty, but may contain a single offer o_s iff
$\forall o_j \exists o_k\, [o_j \sim o_s \wedge o_j \sim o_k \rightarrow o_k \succ o_s \vee o_k \prec o_s]$. In order to strictly define class order, we first define the notion of *class dominance*.

Definition: Class dominance A class of offers C_a is said to dominate another class C_b, $C_a \succ C_b$, iff $\exists o_a \exists o_b\, [o_a \succ o_b]$, $o_a \in C_a, o_b \in C_b$.

[4] In a practical application, the complete knowledge is not going to be available to all sensors, but for an initial study it is convenient to make this assumption.

We use the following recursive definition to define order of class.

Definition: Class of first order A class C of offers is said to be of the first order iff none of its offers are dominated by another offer, $\neg\exists o_m \in O \setminus C\,[o_m \succ o_j]$ for all $o_j \in C$.

Definition: Class of kth order A class of offers C is said to be of order k iff its members are dominated only by members of classes of order k and less.

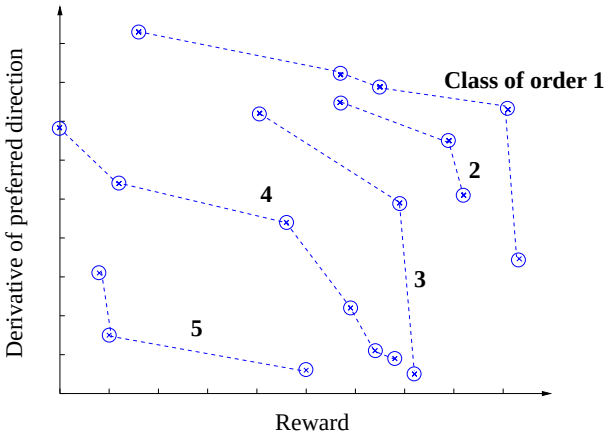


Figure 1: Twenty offers plotted according to derivative of preferred direction and reward. Offers which belong to the same class are connected with lines.

Now, we will express the utility function for sensors. Using our notion of orders, we can assign a utility value to the offers for all agents. Furthermore, according to the negotiation procedure described in Section 2.1, the utility of an offer accepted at time $t+1$ is always less valuable than the same offer accepted at time t. Therefore we need to construct a utility function that is dependent on the time step of the negotiation:

$$U_i(o,t) \stackrel{\Delta}{=} \alpha_U(K-t) - order_i(o), \quad \text{integer } \alpha_U > 0, \tag{8}$$

where $order_i(o)$ is a function that maps offers to its order for sensor agent i.

The interpretation of the utility function in Equation 8 is that the agents will accept less offers the longer the negotiation continues. In fact, for every step the negotiation continues, α_U number of orders of offers will become unacceptable for every sensor agent. Thus, offers of low order (which are desired by the agent) yield a high utility value.

The negotiation procedure in the preceding work described in Section 2 is virtually unaffected by the extensions we make in this work. The reason for this is that all the novelties have been encapsulated in the calculations of the utility function. However, the result of the negotiation will of course be quite different.

An agent that have several "best" offers to choose from should select one according to some second criterion. This could for instance entail minimising the sum of orders, i.e., if the set of best offers is O', then the offer to select should be the one o^* that satisfies $o^* = \arg\min_{o \in O'} \sum_i order_i(o)$. Another suitable criterion could be to select the offer in O' that minimises maximum order for any sensor, i.e., $o^* =$

$\arg\min_{o \in O} \max_i order_i(o)$. If there are still more than one offer that fulfills the criterion, then one offer could be selected randomly.

4.2 Workload effect on tracking performance

A sensor is expected to make less certain measurements for each target if it tracks many targets than if it tracks only a few. Let us assume that sensors have some sort of resource (e.g., time, energy, money, samplings) that they can spend utilise to make measurements. The maximum amount of this resource available to the sensor for a time unit is $\rho_{i,max}$ and the amount it chooses to use to track some target j is denoted ρ_{ij}. We model that the measurement noise v_{ij} in Equation 2 is dependent on the dedicated resource amount ρ_{ij}. The measurement noise is Gaussian, i.e., $v_{ij} \sim N(\mathbf{0}, R)$, with zero mean and the measurement error covariance matrix R which we encountered in the sensor performance formula in Equation 5.

Dissecting R, we notice that it is a diagonal matrix (since a prerequisite for the Kalman filtering is that the, say k, measurement noise components of v_{ij} are independent)

$$R = \begin{bmatrix} \sigma_1^2 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \sigma_k^2 \end{bmatrix} \tag{9}$$

with the variances σ_l^2, $l \in \{1, \ldots, k\}$, as diagonal elements.

The standard deviation functions, $\sigma_l(\rho)$, will take a minimum, $\sigma_{min,l} \geq 0$, for $\rho = \rho_{max}$ and will increase towards infinity when the dedicated resource decreases towards ρ_{min}, $\lim_{\rho \to \rho_{min}(j)} \sigma_l(\rho) \to \infty$, where $\rho_{min}(j)$ is the minimum resource amount necessary to track target j.

Hence, a varying workload on a sensor will affect the standard deviation and the measurement error covariance matrix, which in turn will have effect on the refined sensor gain expression which we will discuss in the next section.

Note that using this model, we allow sensors to allocate different amounts of resources to different targets.

4.3 Target allocation

In the preceding work, the specific task was studied where every target was tracked by exactly one sensor. In this work, we relax that restriction and allow sensors to "share" targets. Thus, we are able to reduce uncertainty by fusing measurements from different sensors and get a higher grade of sensor usage than in the disjoint case.

Our approach here to determine the reward for every sensor, r_i, is to divide the *total reward* on every target, $\sum_j r_j(S_j)$ (S_j being the set of sensors tracking target j), among the sensors in S_j proportionally to their individual contribution.

We define the reward on every target to be

$$r_j(S_j) \triangleq 1 - e^{-\alpha_j g_j(S_j)}, \quad \alpha_j > 0 \tag{10}$$

where

$$g_j(S_j) = \sum_{k \in S_j} \| H_{kj}^T R_{kj}^{-1} H_{kj} \|, \tag{11}$$

i.e., the total information gain on target j. Here, we might want to replace g_j with some measure from information theory when a Kalman filter is not applicable. Previously,

we used sensor information gain (Equation 4), but now, as we allow multi-sensor fusion, we define $g_j(S_j)$ as above and notice that whenever S_j contains a single sensor i $g_j(S_j) = g_j(i) = g(i,j)$.

Now, the *net reward* for every sensor (similarly expressed as in the previous work) is

$$r_i^{net}(D_i) \stackrel{\Delta}{=} \alpha_i + (1-\alpha_i) r_i^m(D_i) \quad 0 \leq \alpha_i \leq 1 \tag{12}$$

where D_i is the group of targets tracked by sensor i, α_i reflects the willingness of a sensor agent to compromise about offers, and the measurement reward is

$$r_i^m(D_i) \stackrel{\Delta}{=} \sum_{j \in D_i} \gamma_{ij} r_j(S_j) \tag{13}$$

and

$$\gamma_{ij} \stackrel{\Delta}{=} \frac{g_j(i)}{g_j(S_j)} = \frac{\|H_{ij}^T R_{ij}^{-1} H_{ij}\|}{\sum_{i \in S_j} \|H_i^T R_{ij}^{-1} H_i\|}, \tag{14}$$

i.e., the relative contribution of sensor i to the state estimate of target j.

This definition of sensor reward, $r_i^m(D_i)$, has the effect that the same gain from a sensor on a target will yield different rewards depending what other sensors track the same target. This makes sense since the target reward does not improve linearly with the information gain (e.g., in a target tracking application, tracking airborne targets at high speeds, to go from metre to centimetre precision in position estimates should not yield much extra reward since the improved precision can not be efficiently utilised).

To prove that $r_i^m(D_i)$ is actually a disbursement of the total reward on targets we need to show that $\sum_i r_i^m(D_i) = \sum_j r_j(S_j)$.

$$\begin{aligned} &\textstyle\sum_i r_i^m(D_i) = \sum_i \sum_{j \in D_i} \gamma_{ij} r_j(S_j) = \big\{\text{Group all } r_j\text{-terms in the sum}\big\} = \\ &\textstyle\sum_j r_j(S_j) \sum_{i \in S_j} \gamma_{ij} = \big\{ \sum_{i \in S_j} \gamma_{ij} \stackrel{\Delta}{=} 1 \big\} = \sum_j r_j(S_j) \end{aligned} \tag{15}$$

■

Typical appearances of net reward functions, r_i^{net}, are shown in Figure 2. In the figure, we use $\alpha_i = 0.3$ and plot r_i^{net} for $\alpha_i \in \{1, 10, 100\}$. We show the results for a single sensor tracking a single target. The curves have similar shape for other values on α_i. For these curves, we have used the covariance matrix in Section 5.1. From the curves, we can see that that an increase in the value of α_j implies that the sensor is satisfied with less certain target state estimates.

4.4 Preferred direction

Given the measurement reward function, $r_i^m(D_i)$, for each sensor, the gradient can be calculated in this way:

$$grad\ r_i^m \equiv \nabla r_i^m \equiv \Big(\frac{\partial r_i^m}{\partial x_i}, \frac{\partial r_i^m}{\partial y_i}\Big), \tag{16}$$

where x_i and y_i are the spatial coordinates of sensor i's position.

The gradient vector points in the direction in which the reward for sensor i will increase the most.[5] This model makes the subtle (and incorrect) assumption that the

[5] Note that we are, in this work, only considering the current target states when calculating the gradient. Prediction of future target states to further improve the performance of the mobile sensors is left for future work.

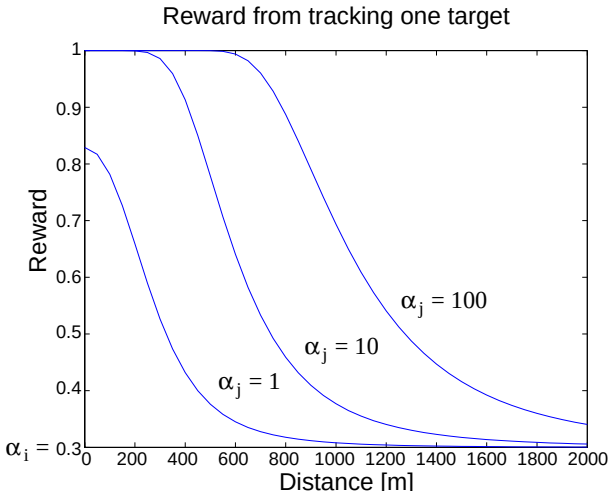


Figure 2: The three curves show a net reward function plotted with $\alpha_i = 0.3$ and $\alpha_j = 1, 10,$ and 100.

targets are stationary. However, it is a fairly good approximation that should be refined in the future; possibly by predicting and exploiting future target states. The gradient would be the preferred direction to move for the sensor if terrain properties were not considered.[6] However, the terrain may make motion in the direction of the gradient difficult or perhaps even impossible, and a more passable path, although less rewarding, might be a better preferred direction.

Now assume we can construct a (possibly rough) terrain dependent function, which discounts the reward change in various directions. Let the *terrain function* be $t(\mathbf{p}, \mathbf{e}_\theta)$, where $\mathbf{p}$ is a two dimensional position in the environment and $\mathbf{e}_\theta$ is a unit vector, $\theta \in [0, 2\pi)$. Furthermore, let the terrain function assume values between 0 and 1, $t \in [0, 1]$. The terrain function $t(\mathbf{p}, \mathbf{e}_\theta)$ takes high values in directions where the sensor platform can easily move (such as in the direction of a good road) and low values in directions where it cannot move very well (zero in the direction of an unpassable obstacle). We assume that the value reflects the passability in the chosen direction in the following time step.

The directional derivative $r'_{\mathbf{e}_\theta}$ in any direction, $\mathbf{e}_\theta$, is simply a projection of the gradient onto $\mathbf{e}_\theta$, i.e., $r'_{\mathbf{e}_\theta} = \mathbf{e}_\theta \bullet \nabla r^m$. The parameter θ is the angle between the gradient and $\mathbf{e}_\theta$, as shown in Figure 3.

Now, we propose that the preferred direction, δ^*, is the unit vector that corresponds to the largest directional derivative discounted by $t(\mathbf{p}, \mathbf{e}_\theta)$, i.e.,

$$\delta^* = \arg\max_{\mathbf{e}_\theta} \left\{ t(\mathbf{p}, \mathbf{e}_\theta) \cdot r'_{\mathbf{e}_\theta} \right\}. \tag{17}$$

Figure 4(a) shows a terrain function in a position $\mathbf{p}$ where terrain has no effect on the mobility, i.e., in all directions, ϕ, $t(\mathbf{p}, \mathbf{e}_\phi) = 1$. Figure 4(b) shows the resulting discounted directional derivatives (which in this case were unaffected by the terrain function) where the gradient is depicted as the solid line with a cross on its end point. The length of a line corresponds to the size of its derivative. Directions with derivatives less than zero (those directions which have more than a 90 degree angle to the

[6] Hence, in the case of airborne sensors, the gradient would suffice as a preferred direction.

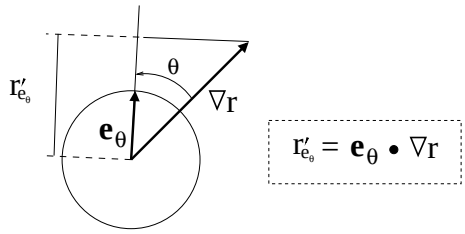
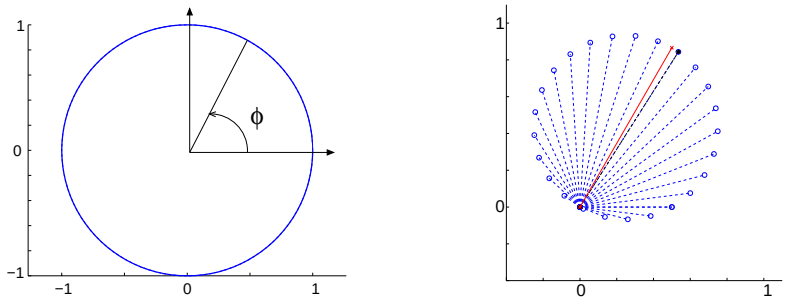


Figure 3: The directional derivative, $r'_{\mathbf{e}_\theta}$, in the direction of the unit vector $\mathbf{e}_\theta$ is calculated as the projection of the gradient, ∇r, on $\mathbf{e}_\theta$.

gradient) are not depicted. Since directional derivatives do not change sign due to the terrain function; they are only discounted with a positive factor so the smallest discounted derivative possible is zero. Hence, the directions with negative derivatives can be ignored.



(a) (b)

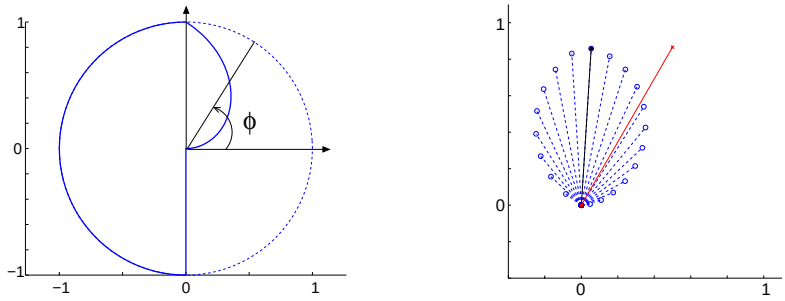
Figure 4: **(a)** shows the terrain function $t(\mathbf{p}, \mathbf{e}_\phi)$. In this case $t = 1$ for all ϕ, so it is the function of the unit circle. **(b)** shows the size of some discounted directional derivatives when the gradient is the solid line pointing upwards and right. In this case, the directional derivatives are discounted with the function in (a) and are thus unaffected.

Figure 5(a) shows the heterogeneous terrain function

$$t(\mathbf{p}, \mathbf{e}_\phi) = \begin{cases} 2\phi/\pi & \phi \in [0, \pi/2] \\ 1 & \phi \in (\pi/2, 3\pi/2] \\ 0 & \phi \in (3\pi/2, 2\pi) \end{cases} \tag{18}$$

Figure 5(b) shows the same gradient as in Figure 4(b), but here the directional derivatives have been discounted with the function in Equation 18. The direction which has the greatest directional derivative is depicted with a solid line, calculated with Equation 17, and would be the preferred direction of an agent. This direction deviates notably from the gradient.

We have now seen how the preferred directions of a sensor platform are calculated for terrain which has no effect on the platform (in Figure 4(b)) and terrain which has (Figure 5(b)). We now expand our field of view to study the preferred directions in



(a) (b)

Figure 5: **(a)** shows the plot of the terrain function in Equation 18. **(b)** shows some of the directional derivatives discounted with the terrain function in Equation 18.

a whole area. Figure 6(a) shows the directional derivatives in various positions in the plane when its preferred direction is unaffected by terrain conditions. A target in position $(400, 350)$ (the small "x") attracts a sensor platform. We note that the derivatives are small in the periphery and close to the target, and large in between. This is the same characteristics we saw in the curves in Figure 2.

In Figure 6(b), an obstacle (representing almost unpassable terrain) has been positioned to the left in figure. The preferred directions direct the sensor platform away from the obstacle, while trying to preserve a course towards the target. For instance, along the upper and lower edges of the obstacle, the preferred directions are along the edge of the obstacle rather than into the obstacle.

Even though the approach with terrain functions presented here looks nice in this example, it is indeed short-sighted. There is a risk that sensor platforms get stuck behind obstacles. However, this does not necessarily mean that the tracking will fail, rather it means that the current allocation has been given a new value which will possibly affect the outcome in the next round of negotiations (i.e., another allocation, with a better preferred direction, might be a more appealing alternative).

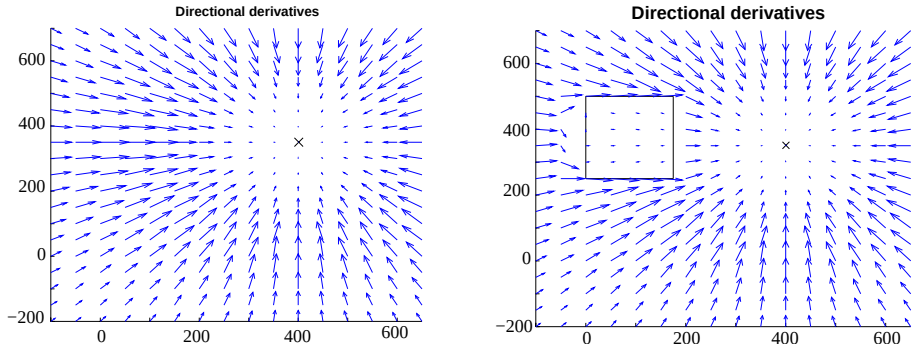
5 Experimental results

First, we verify that the negotiation algorithm is still beneficial for stationary sensors with respect to the new features of the problem (Section 3). Then we move on to verify its suitability to the case with mobile sensors.[7] In Appendix A we select the values of some of the parameters used in the simulation, and in Appendix B we derive an analytical expression for the gradient.

5.1 Selected parameters and requirements

For our simulations, we assume that the standard deviation, σ_{tot}, of the measurement noise covariance R, is equal for every measurement component and tracked target.

[7] In this paper, merely snapshots of simulations are shown. However, full animations are available at this URL: `http://www.nada.kth.se/~rjo/pubs/mobile/anim/`.



(a) (b)

Figure 6: **(a)** The derivatives of the preferred directions in various positions. The target, the "x" in position $(400, 350)$, acts as the force that attracts the sensor platform and the terrain does not affect the preferred directions of the platform. **(b)** Here, an obstacle (representing very rough terrain) is situated in the left part of the figure. The generated preferred directions tries to steer the sensor platform away from the obstacle while preserving a course towards the target.

We, furthermore, assume that it increases inversely linearly with the dedicated relative resource amount, i.e., the standard deviation is scaled by a factor $\left(\frac{\rho}{\rho_{max}}\right)^{-1}$, and quadratically with the Euclidean distance d between target and sensor. If the tracking resource, discussed in Section 4.2, is divided evenly between n tracked targets, the resource amount used to track each of the targets is $\rho(n) = \rho_{max}/n$, yielding the scale factor $\left(\frac{\rho_{max}/n}{\rho_{max}}\right)^{-1} = n$ for the standard deviation. From this discussion, we suggest the following standard deviation expression for our experiments

$$\sigma_{tot} = \sigma_{min} \cdot n \cdot (1 + cd^2). \tag{19}$$

The first two factors are always greater than zero and $d \geq 0$. The coefficient $c > 0$ controls how greatly the distance from sensor to target affects the measurement error covariance.

Equation 19 is plotted in Figure 7 for one to four targets. The values on the x-axis denotes the distance to target and the y-axis the relative increase in covariance, with $\sigma_{min} = 1$ as reference. We see, e.g., that the covariance for one target at distance 400 metres, when concurrently tracking four targets, is about ten times the minimum covariance.

We require that the tracking system always tracks all targets (i.e., sensor to target assignments that do not include assignments to all targets will be ignored by all sensor agents).

5.2 Computational issues

The time complexity of the algorithm to implement is heavily dependent on the number of offers to consider, q, which, in turn, is dependent on the number of sensors, s,

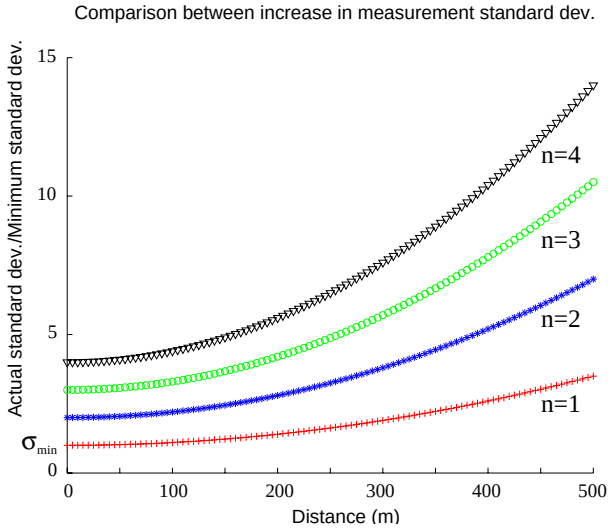


Figure 7: The covariance function in Equation 19 is here plotted for one to four concurrently tracked targets.

and targets, t. In the previous work, offers contained only target distributions where targets were not shared. In that work, the number of offers were $q = O(s^t)$, i.e., it was exponential in the number of targets.

Now, since targets may be shared by sensors, the size of the offer set increases to $O(2^{ts})$ [8] and is, hence, exponential in both the number of targets and sensors. Hence, in our work it is necessary to adopt some heuristic to lower the number of possible offers.

By studying the problem at hand, it is often quite possible to design suitable approximations. In this case, we assume that it is quite unlikely and unwanted that sensor agents change their allocations much from one negotiation to the other. In support of this assumption is the fact that target positions are dependent on kinematic constraints, and the optimal allocation of targets is therefore expected to change slowly over time. Exploiting the expected inertia in the change of the optimal allocation, we construct the set of offers to negotiate about in the following way:

Given the current allocation of targets to sensors, let the set of offers to negotiate about include all combinations for which each sensor, either

- keeps its current allocation,
- drops one target of the current allocation or picks up a new one, or
- exchanges one target for another.

Even though this heuristic reduces the size of the offer set considerably to $O(t^{2s})$, it is still exponential in the number of sensors. For future work, the size of the offer set will have to be decreased even further, but for the experiments in this article it suffices.

[8] Any sensor may track any number of targets, hence, every sensor may allocate 2^t targets, yielding a total number of $(2^t)^s = 2^{ts}$ possible allocations.

5.3 Target tracking with stationary sensors

The first thing we want to verify is that the negotiation algorithm still, with an implementation of the conditions in Section 3, produces satisfying results.

We run a simulation, very much similar to the one used in the preceding work. In the scenario (Figure 8), three stationary sensors, spatially separated and placed on a line in east-west direction, track four targets for some time. The targets approach the sensors in pairs, one pair approaching from the east and the other from the west. Targets τ_3 and τ_4 travel slightly faster than τ_1 and τ_2.

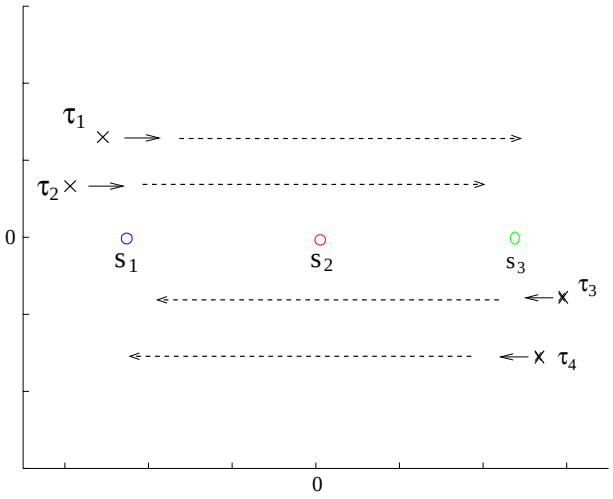


Figure 8: In this scenario, three stationary sensors (s_1, s_2 and s_3) track four targets (τ_1 to τ_4).

In the previous work [XCS03], we compared our negotiation algorithm with an optimal algorithm which optimised the sum of information gain. The results showed that the negotiation algorithm reached a result which was very close (99%) to the optimal algorithm in terms of *average total sensor information gain*.

A second criterion to observe was *concentration degree*, i.e., a measure of how well the targets are divided among the sensors. E.g., a target distribution in which all targets are tracked by a single sensor will yield a high concentration degree. This is an awkward situation since if the sensor that tracks all targets fails or is destroyed all targets will be lost. Thus, a low concentration degree, representing that targets are divided evenly among the sensors, is desired. In the previous work, the concentration degree turned out to be about 10% better for the negotiation algorithm compared to the optimal algorithm in a simulation.

In the current work, for every completed negotiation, we compare the result of the negotiation algorithm (i.e., an assignment of sensors to targets) with the result that the optimal algorithm would have yielded in the same situation. In our experiment, we want to compare the following criteria for our negotiation algorithm and an optimal one,

Total reward This is the value of the sum of the target rewards, i.e., $\sum_j r_j$. Of course, a high value of total reward corresponds to a good overall tracking performance and is desirable.

Redundancy This is the number of targets that are being tracked by one or more sensors. For those targets that are being tracked by one or more sensors, we have redundant measurements which can be fused. This is wanted for that reason, but also because if one sensor fails or is destroyed, the other sensor(s) will still receive measurements. If only one sensor tracks a target, if that sensor is lost so is the target. A high value, while preserving high total reward, is desirable.

Lost targets This is a value of the average number of targets lost if one of the sensors fails or is destroyed. A low value is desired.

In the following simulation, the three stationary sensors track the four targets over five hundred rounds of negotiations. The targets are moving fast and the sensors are re-negotiating their target assignments (i.e., starting a new round of negotiations) in every time step (perhaps every second or so).

Figure 9 shows the result of the negotiation algorithm (`N-tracker`) and Figure 10 the result of the optimal algorithm (`O-tracker`). In each diagram, the x-axis is time and y-axis which targets are being tracked by the sensor corresponding to the diagram. As we can see, the results of the two algorithms appear to be very similar.

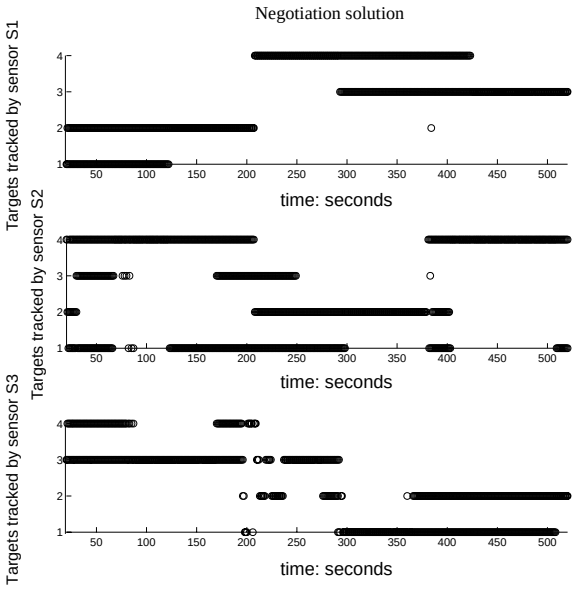


Figure 9: The diagrams shows the target allocations of all three sensors for every time step in the negotiation.

In Figure 11, we see three diagrams. The topmost diagram depicts the relative reward of the `N-tracker` in every time step, i.e., the reward of the `N-tracker` divided by the reward of the `O-tracker`. Of course, the `N-tracker` will never receive as much reward as the `O-tracker`, but its rewards are certainly comparable.

The middle diagram depicts the differences in redundancy between the two algorithms. In every time step, the redundancy of the `O-tracker` is subtracted from the redundancy of the `N-tracker`. As shown, most of the time, the difference is zero, i.e., the two algorithms have the same redundancy. However, quite often the `N-tracker` has a greater redundancy and only during a few time steps the `O-tracker` has a greater redundancy.

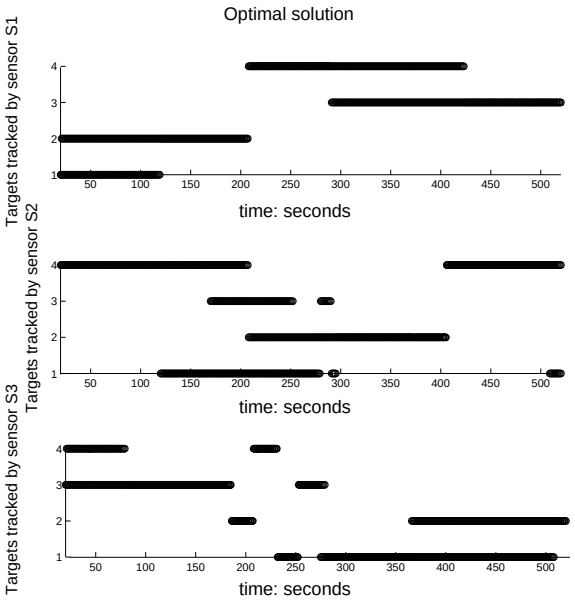


Figure 10: The diagrams shows the optimal target allocations of all three sensors for every time step in the negotiation.

The bottommost diagram shows the average number of lost targets (if one sensor is destroyed) plotted for both algorithms for every time step. The values of the `N-tracker` is plotted with a dotted line and the values of the `O-tracker` is plotted with a dashed line. We see that the results seem to coincide with the redundancy diagram, i.e., the `O-tracker` outperforms the `N-tracker` only in a few time steps.

Our experiments with stationary sensors show that the negotiation algorithm yields near-optimal tracking quality while improving robustness to sensor failure.

5.4 Target tracking with mobile sensors

Now, we introduce mobile negotiating sensors and wish to evaluate their performance. We here use the utility function in Section 4.1, but we will for now assume that the terrain has no effect on the negotiation.

For evaluation, we make two types of comparisons:

- For every sensor to target assignment the negotiation algorithm produces, we compare it to an optimal reward one (just like in the stationary case).
- We design and implement a "greedy" tracker (`G-tracker`) which operates independently of the negotiation based tracker (`N-tracker`).

We let the `G-tracker` reconsider the sensor to target assignment as often as the `N-tracker` does. After having selected the most optimal assignment, the sensors travel, at full speed, in the direction of the gradient. Whereas that seems reasonable, we will see in Section 5.5 what effects such an approach might have in a scenario where speed is dependent on terrain.

In our first simulation with mobile sensors, we want to know whether our reward function makes sensors try to fixate one target or if they tend to locate themselves where

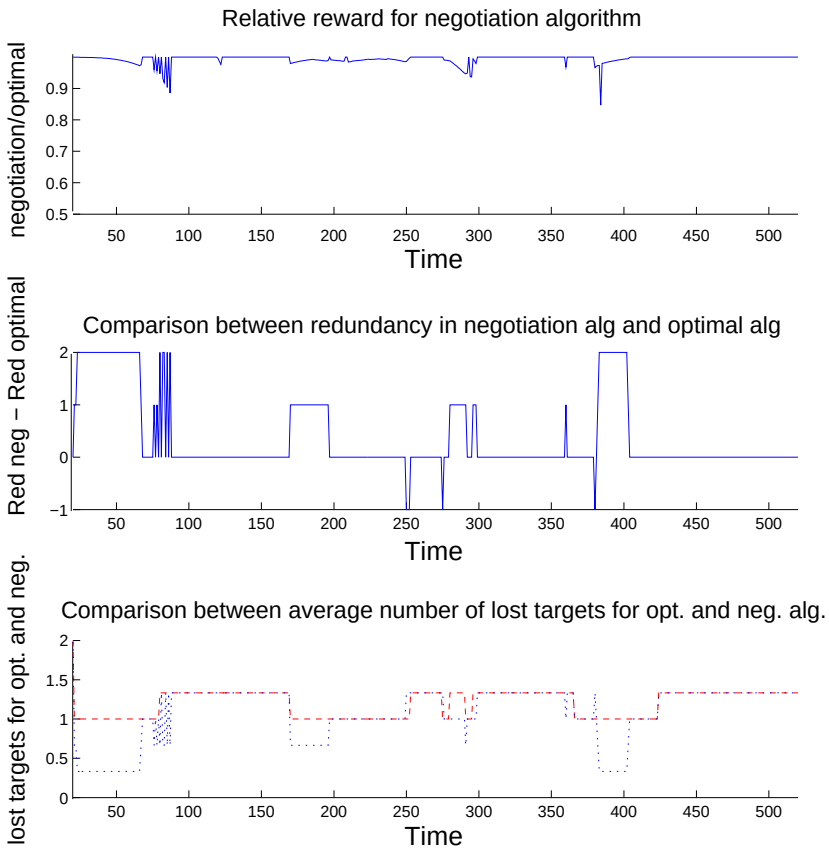


Figure 11: In these diagrams, we compare the results of the `O-tracker` and the `N-tracker`. The topmost diagram shows the relative reward of the `N-tracker` compared to the `O-tracker`. The middle diagram shows the difference in redundancy between both algorithms in every time step. The bottommost diagram plots the average number of lost targets (if one sensor fails) for both algorithms (the dotted line corresponds to the result of the `N-tracker`).

measurement performance on all targets is good. In Figure 12, two sensors track four targets. In this and the following figures that depict snapshots of target tracking with mobile sensors, the crosses are targets, the tiny circles are the mobile sensors, and the line that extends from the centre of each sensor indicates the current direction of motion of the sensor (it does not, however, indicate the speed of the sensor). Additionally, in some of the figures, dotted lines are drawn from sensors to targets. These lines clarify which sensors are tracking which targets.

The simulation starts at time $t = t_1$, and at this time the targets are divided between the two sensors in such a way that the upper sensor is willing to track the two upper targets and the lower sensor is willing to track the two lower targets. The upper targets are moving upwards and the lower targets are moving downwards. We see that the sensors, which in this simulation have the ability to catch up with the targets, prefer to situate themselves in between the targets.

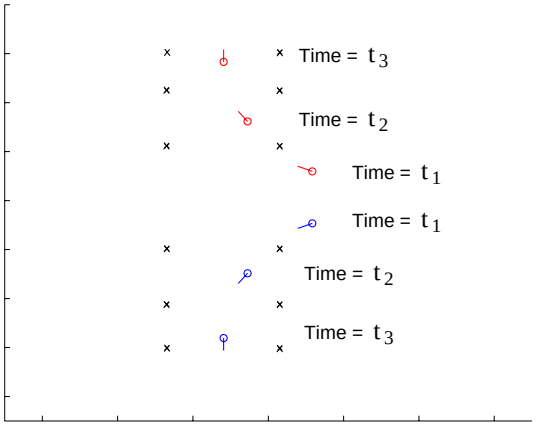


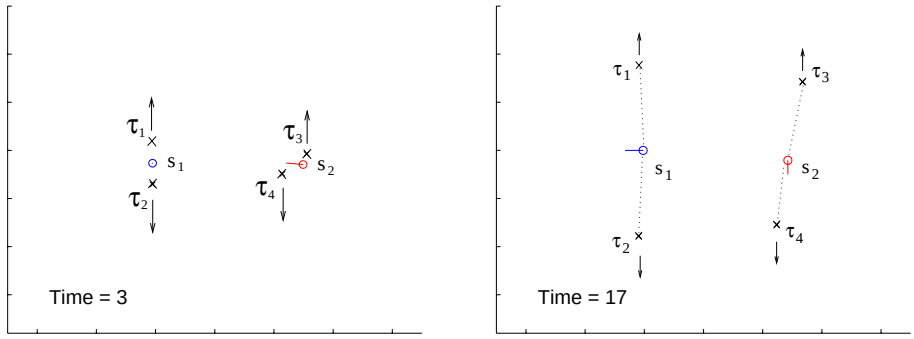
Figure 12: The Figure shows three superimposed snapshots, at times t_1, t_2 and t_3 ($t_1 < t_2 < t_3$), of a scenario where two sensors track two targets each.

In our next experiment, we study a scenario where the `G-tracker` runs into problems. In this case, sensor s_1 (in Figure 13(a)) wants to track the targets τ_1 and τ_2. However, they move in opposite directions, leaving s_1 with a resulting zero gradient, i.e., s_1 gets stuck while the targets move away (as seen in Figure 13(b)). Sensor s_2 on the right has a similar problem since its targets are also moving in opposite directions. After a while, however, the `G-tracker` assigns targets τ_1 and τ_3 to sensor s_1 and the others to s_2, allowing sensor s_1 to escape from its deadlock. If we align targets τ_3 and τ_4 with sensor s_2 and rerun the simulation, we can actually make both sensors get stuck forever.

The `N-tracker`, run on the same scenario, yields a more appealing result. To begin with, we see that the negotiation brings about a somewhat surprising assignment of targets to sensors (Figure 14(a)); s_1 tracks τ_3 and τ_4, and s_2 the other two, contrary to the allocation of the `G-tracker` (see once again Figure 13(a)). The reason is of course that the "greedy" allocation yields very low directional derivatives which allows the `N-tracker` to reach other solutions.

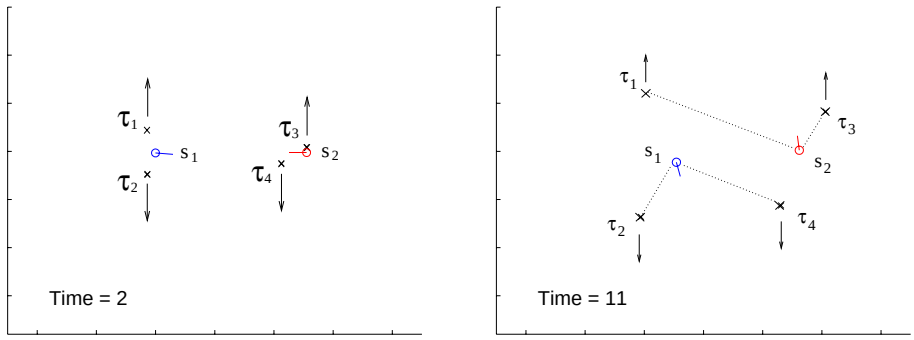
After a short while, sensor s_1 starts to follow the targets τ_2 and τ_4 that are moving downwards, and the other two are followed by sensor s_2 (Figure 14(b)).

In Figure 15, we compare the results of the `N-tracker` and `G-tracker` in terms of reward. At time $t = 10$, the `N-tracker` decides that sensor s_1 should track targets τ_2



(a) (b)

Figure 13: **(a)** In this scenario, two sensors s_1 and s_2 track four targets τ_1 to τ_4. Targets τ_1 and τ_3 are moving upwards and τ_2 and τ_4 downwards. Initially, the `G-tracker` assigns τ_1 and τ_2 to s_1 and τ_3 and τ_4 to s_2. **(b)** After some time, the targets have moved, but due to the "greedy" allocation of targets to sensors, the sensors are stuck between their assigned targets and have hardly moved.



(a) (b)

Figure 14: **(a)** Initially, the negotiation algorithm assigns targets τ_3 and τ_4 to sensor s_1 and the rest to sensor s_2. **(b)** After some time, the negotiation algorithm assigns targets τ_2 and τ_4 to sensor s_1 and the rest to sensor s_2.

and τ_4 and quickly receives a total reward which is greater than that of the `G-tracker`. At time $t = 27$, also the `G-tracker` decides that one sensor should track the targets moving upwards and the other the ones going downwards. However, as we can see from the rewards in the figure, the `G-tracker` is unable to catch up with the `N-tracker`. Since the targets in this scenario are allowed to travel at a higher speed than the sensors, the reward drops rapidly and at time $t = 40$ and beyond, both algorithms receive very low rewards.

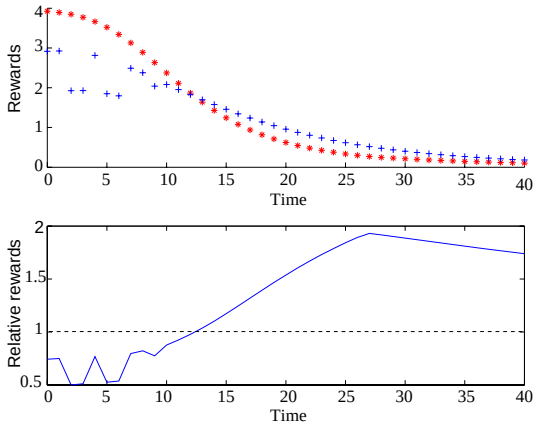


Figure 15: This graph compares the total rewards of the `G-tracker` and the `N-tracker` to each other. The absolute reward has been plotted in the top graph ('*' for `G-tracker`and '+' for `N-tracker`. In the lower graph, the relative reward of the `N-tracker` compared to the `G-tracker` has been plotted. For the first time steps, the `G-tracker` outperforms the negotiation one. At time $t = 10$, the negotiation assigns targets τ_2 and τ_4 to sensor s_1 which results in an increase in performance compared to the `G-tracker`. At time $t = 27$ the `G-tracker` comes to the same conclusion, which explains the negative slope of the curve.

In the final experiment of this section, we once again study the scenario in Figure 8. However, this time the sensors are mobile and both distances to targets and speed of targets have been decreased so that the sensor can take advantage of their mobility (i.e., it is not beneficial to use mobile sensors if their maximum speed is relatively low compared to the targets).

We run both the `G-tracker` and the `N-tracker` and compare the results in Table 1. We see that the `N-tracker` loses in measurement accuracy (its average measurement performance was 90% of that of the `G-tracker`). However, the `N-tracker` instead impresses by its robustness with an average of 1.39 targets being tracked by one or more sensors and average of 0.87 (27% better than the result of the `G-tracker`) of lost targets if one sensor is lost. The reason for this result is that the sensors, through the negotiation, are forced to share targets with each other, and, hence, yield better robustness for the target tracking system as a whole.

5.5 Mobile tracking with terrain considerations

Until now, we have not considered terrain effects on mobile sensors in our experiments. Since it is highly unlikely that the designer of a mobile sensor system can expect a homogeneous environment, we need to consider varying terrain and its effects. In

Table 1: Comparison between `G-tracker` and `N-tracker`

	G-tracker	N-tracker	Relative
Reward	3.7372	3.3765	0.90
Redundancy	0.4510	1.3922	3.09
Lost targets	1.1830	0.8693	0.73

Section 4.4, we discussed how a so-called terrain function can be used to discount the directional derivative generated by a certain assignment.

In the scenario in Figure 16, we have put an *obstacle* into the environment. This obstacle has the property that when a mobile sensor tries to cross it, the maximum speed of the sensor reduces drastically. Such an obstacle represents, for instance, rough terrain or a steep hill. In this example, the speed reduces to 30% of the maximum speed it could achieve in an ideal terrain. Close to the obstacle, the terrain function discounts directional derivatives that lead into the obstacle.

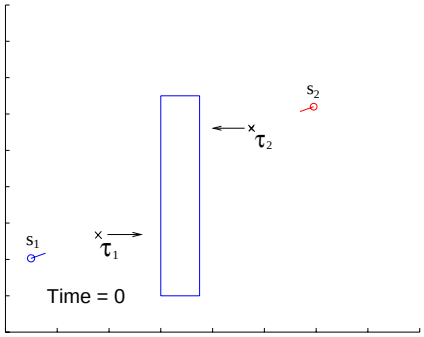


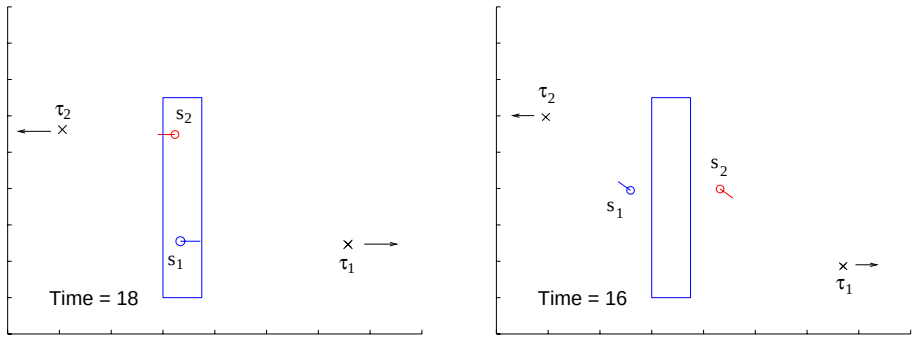
Figure 16: Initially, sensor s_1 tracks target τ_1 and sensor s_2 target τ_2. The rectangle represents an area which slows down mobile sensors that enter it.

We notice that the `G-tracker`, which does not consider terrain, leads the sensors straight into the obstacle, as shown in Figure 17(a). As a result of this, the sensors lose touch with the targets. In the case of the negotiation algorithm, the sensors switch targets close to the border of the obstacle, as shown in Figure 17(b). One reason for this is that offers that give directions that lead into the obstacle get small derivatives and are suppressed.

6 Discussion

There are a number of parameters that can be altered that affect the behaviour of the target tracking system. α_j, in Equation 10, influences the target reward and reflects the value the system assigns to increased accuracy in measurements. As can be seen in Figure 2, by varying the value of α_j, we can customise the value of increase measurement accuracy.

Also included in Figure 2 is α_i which is a parameter in the sensor net reward function (Equation 12). By varying α_i between 0 and 1, we can modify the ability of sensor agents to differentiate between rewards of offers. For $\alpha_i = 0$ the ability of the agent to differentiate between rewards is at its maximum, but for $\alpha_i = 1$ all offers appear to



(a) (b)

Figure 17: **(a)** The `G-tracker` does not consider terrain and leads the sensors into the obstacle, where they are slowed down considerably. **(b)** The negotiation algorithm decides to switch targets between the sensors instead.

the sensor agent to have same reward. This can be understood by once again studying Figure 2 where α_i is set to 0.3 and imagining the effect of increasing the α_i value.

The negotiation procedure can be adjusted in several ways, e.g., by 1) altering the order of offers, by 2) changing the way the number of valid offers is reduced during a negotiation, and by 3) changing the number of steps in the negotiation.

To reiterate, as we explained in Section 2.1, due to the facts that the agents are benign and have complete information about the others, the agent that begins the negotiation can calculate an offer which all other agents will accept. Hence, when we talk about negotiation in the following discussion, we are referring to the search procedure by which the offer, which all agents will accept, is found.

The first way to adjust the negotiation procedure involves deciding on a policy for in which order agents should make their offers (this is the $P(H)$ function of the game definition in Section 2.1). Naturally, the agent who makes the first offer has an advantage. There are several ways to do this, one might for instance want the agent that received the best/worst reward in the previous round of negotiations to start and the others to follow in increasing/decreasing order. In most of our experiments, we used another policy which we considered to be more fair. We let all agents have the advantage of commencing the negotiations about the same amount of times each. In a round-robin fashion, one of the agents, say a_1, started a round of negotiations, another agent, a_2, made the second offer, and a third agent, a_3, made the third offer, etc.

Quite often, the number of steps in the negotiation was larger than the number of agents (allowing every agent to participate in the negotiation), and in those cases, when the last agent had made its first offer, the first agent, a_1, continued. In the next round of negotiations, it was a_2's turn to commence the negotiation, followed by agent a_3, and so on.

The second way to adjust the negotiation procedure is to change how the set of valid offers evolves during a round of negotiations. Valid offers are offers, o, that do not violate the requirements of Equation 1. In the previous work, the composition of the set of offers was heavily dependent on the utility functions of the agents. If the

utility of the offers for all agents, given by Equation 7, differed only slightly, then even for many negotiation steps, the set of valid offers would still be considerable. If so, the agent which commences the negotiation will have a great influence over the outcome. It might be debated whether this is a disadvantage or not. On the one hand, the commencing agent will be very powerful, possibly completely ignoring the wishes of the other agents, which might be undesirable. On the other hand, that the offers have similar values to an agent could be interpreted as indifference on the behalf of the agent. Thus, in that case, the agent simply does not care about the outcome of the negotiation.

In this work, we experimented with another approach that systematically reduces the set of valid offers with each step of the negotiation, guaranteeing that the agent that begins the negotiation will have a more restricted set of offers to choose from. Starting from a larger set of possible offers, $\eta = \frac{\#\text{offers}}{\#\text{negotiation steps}}$ number of offers are excluded from the negotiation (i.e., becomes invalid) with every step in the negotiation, ensuring that there are only η offers left to choose from in the end.

To explain by which criterion offers are removed, consider Figure 18. The figure illustrates the common situation where offers have different utility for different agents. Assume that an offer, o, has been proposed at some step of the negotiation. Now, every other offer o' has a certain distance in utility to o for every agent i, $dist_i(o,o') = |U_i(o) - U_i(o')|$. Let the maximum distance of all agents for an offer, o', be $dist(o,o') = max_i dist_i(o,o')$. Finally, to decrease the set of offers with every step of the negotiation, the η offers with the largest $dist(o,o')$ are removed. Despite the appealing property of this approach, i.e., the strict monotonic decrease of the size of the offer set, we are not yet convinced by its positive effect on the negotiations.

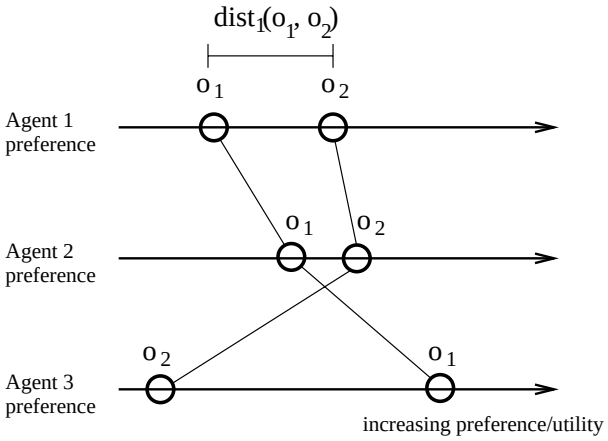


Figure 18: Different offers have different values to different agents. For instance, in this example, $o_1 \succ_1 o_2$ but $o_2 \succ_3 o_1$

The third way to affect the negotiation is to change the number of steps of the negotiation. In our experiments, we used quite lengthy negotiations of at least 30 steps, sometimes many more. The outcome of the negotiation, is under some circumstances, very much dependent on the exact number of negotiation steps. This potential problem is to some extent avoided when the systematic approach to decreasing the set of valid offers, presented in the previous paragraph, is applied. Generally, the more negoti-

ation steps used to reach an agreement, the more "democratic" is the outcome of the negotiation.

Finally, we find our solution to the multiple-objective optimisation problem, that arose in Section 4.1, intriguing and it encourages to further investigation.

7 Conclusion and future work

In this report, we have presented a game theoretic model for allocating targets to mobile sensors. Sensor agents negotiate by proposing offers of allocations that involve all sensors. Each agent can evaluate each offer to decide its individual utility.

The utility is composed of two objectives: sensor reward and directional derivative. The first objective, sensor reward, is dependent on the distance between sensor and targets, the number of targets the sensor is concurrently tracking, and whether other sensors track the same target. The other part, directional derivative, is directly calculated from the allocation of the offer, or, when terrain conditions are considered, by discounting derivatives in inconvenient directions.

We showed, in the experiments in Section 5, two interesting properties of our negotiation algorithm: first, the negotiation forces sensors to share targets, improving robustness to the target tracking system (e.g., the scenario in Figure 8). Secondly, considering directional derivatives allow sensors to proactively reconsider target assignments, possibly improving long-term information gain (e.g., as in Figures 14(b) and 17(b)).

Further studies should investigate under what circumstances these properties imply advantages to the target tracking system. With the support of these early results, we anticipate interesting discoveries in our future exploration of negotiation-based, distributed sensor management.

Some of the most salient, concrete directions for future studies are:

- introduction of uncertainty (e.g., in target or sensor state) into the negotiations,
- prediction of (near) future target and sensor states to improve tracking performance,
- to explore and devise a policy to select negotiation strategy depending on the state of the environment.

References

[BSF88] Yaakov Bar-Shalom and Thomas E. Fortmann. *Tracking and Data Association*. Academic Press, 1988.

[BSL93] Yaakov Bar-Shalom and Xiao-Rong Li. *Estimation and Tracking: Principles, Techniques and Software*. Artech House, Boston, 1993.

[CFK97] Y. Uny Cao, Alex S. Fukunaga, and Andrew B. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4:7–27, 1997.

[CLOHd+01] Jr. Charles L. Ortiz, Eric Hsu, Marie desJardins, Timothy Rauenbusch, Barbara Grosz, Osher Yadgar, and Sarit Kraus. Incremental negotiation and coalition formation for fesource-bounded agents. Preliminary report, 2001. AAAI Fall Symposium on Negotiation Methods for Autonomous Cooperative Systems.

[DN01] Pierre Dodin and Vincent Nimier. Distributed resource allocation under communication constraints. In *Proceedings, 4th International Conference on Information Fusion*. International Society of Information Fusion, 2001.

[FF98] Carlos M. Fonseca and Peter J. Fleming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms-part i: A unified formulation. *IEEE Transactions on Systems, Man, and Cybernetics*, 28(1):473–509, January 1998.

[Nas53] John F. Nash. Two-person cooperative games. *Econometrica*, 21:128–140, 1953.

[NN00] G. W. Ng and K. H. Ng. Sensor management - what, why and how. *Information Fusion*, 1:67–75, 2000.

[OR94] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

[Wei99] Gerhard I. Weiss, editor. *Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999.

[XCS03] Ning Xiong, Henrik I. Christensen, and Per Svensson. Negotiation of target distribution enhancing system survivability. Report CVAP-274, Royal Institute of Technology, Numerical Analysis and Computer Science, Computer Vision and Active Perception Laboratory, 100 44 Stockholm, SWEDEN, 2003. http://www.nada.kth.se/cvap/cvaplop/lop-cvap.html, submitted to IEEE Transactions on Systems, Man, and Cybernetics.

[XS02] Ning Xiong and Per Svensson. Sensor management for information fusion - issues and approaches. *Information Fusion*, 3:163–186, 2002.

[ZR96] Gilad Zlotkin and Jeffrey S. Rosenschein. Mechanism design for automated negotiation, and its application to task oriented domains. *AI Journal*, 86(2):195–244, October 1996.

A Simulation parameters

The H_{ij} matrices in the Kalman measurement equation (Equation 2) used here are all equal:

$$H_{ij} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

The measurement error covariance matrices R_{ij} are also assumed to be identical:

$$R_{ij} = \begin{bmatrix} \sigma_{tot}^2 & 0 \\ 0 & \sigma_{tot}^2 \end{bmatrix},$$

We here assume that the standard deviations for measurements, σ_{tot}, for both measurement components, e.g., x- and y- coordinates, are equal and independent.

B Gradient derivation

Recall that the components of the gradient, depicted in Equation 16, are the partial derivatives of the measurement reward function in Equation 13. Important to notice is that both factors of the measurement reward function, γ_{ij} and $r_j(S_j)$, are dependent on the derivation variables, x_i and y_i.

The partial derivatives, $\frac{\partial r_i^m}{\partial x_i}$ and $\frac{\partial r_i^m}{\partial y_i}$, are similar, and, thus, we show only the detailed calculation of $\frac{\partial r_i^m}{\partial x_i}$ and claim that the calculation of other derivative is almost identical.

$$\frac{\partial}{\partial x_i}(r_i^m) = \quad \{\text{since } D(fg) = fg' + f'g\} = \sum_j \gamma_{ij} \frac{\partial}{\partial x_i}(r_j) + \frac{\partial}{\partial x_i}(\gamma_{ij}) r_j \tag{20}$$

We know r_j and γ_{ij} from Equation 14 and Equation 10, respectively. However, $\frac{\partial}{\partial x_i}(r_j)$ and $\frac{\partial}{\partial x_i}(\gamma_{ij})$ have yet to be determined.

$$\begin{aligned} \frac{\partial}{\partial x_i}(r_j) = \quad & \alpha_j \frac{\partial}{\partial x_i}(g_j(S_j)) e^{\alpha_j g_j(S_j)} = \left\{\text{since } e^{\alpha_j g_j(S_j)} = 1 - r_j\right\} = \\ & \alpha_j \frac{\partial}{\partial x_i}(g_j(S_j))(1 - r_j) \end{aligned} \tag{21}$$

$$\begin{aligned} \frac{\partial}{\partial x_i}(g_j(S_j)) = \quad & \left\{g_j(S_j) = \sum_{k \in S_j} \|H_{kj}^T R_{kj}^{-1} H_{kj}\| \text{ from Equation 14}\right\} = \\ & \frac{\partial}{\partial x_i}\left(\sum_{k \in S_j} \|H_{kj}^T R_{kj}^{-1} H_{kj}\|\right) = \\ & \left\{\frac{\partial}{\partial x_i}\left(\|H_{kj}^T R_{kj}^{-1} H_{kj}\|\right) = 0, \forall k \neq i\right\} = \frac{\partial}{\partial x_i}\left(\|H_{ij}^T R_{ij}^{-1} H_{ij}\|\right) \end{aligned} \tag{22}$$

The matrices H_{ij} and R_{ij} used are described in Appendix A, but in order to calculate the partial derivative $\frac{\partial}{\partial x_i}\left(\|H_{ij}^T R_{ij}^{-1} H_{ij}\|\right)$ we also have to decide which matrix norm to use. In this work we use the Frobenius norm, $\|\cdot\|_F$, which considers every element of the matrix.[9]

$$\|H_{ij}^T R_{ij}^{-1} H_{ij}\|_F = \left\| \begin{bmatrix} \sigma_{tot}^{-2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{tot}^{-2} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \right\|_F = 2^{1/2} \sigma_{tot}^{-2} \tag{23}$$

Thus, the partial derivative then becomes

$$\begin{aligned} \frac{\partial}{\partial x_i}\left(\|H_i^T R_{ij}^{-1} H_i\|_F\right) = \quad & -2^{3/2} \sigma_{tot}^{-3} \frac{\partial}{\partial x_i}(\sigma_{tot}) = \{\text{using } \sigma_{tot} \text{ in Eq. 19}\} = \\ & -2^{5/2} c (\sigma_{min} n)^{-2} (1 + cd^2)^{-3} (x_i - x_j) \end{aligned} \tag{24}$$

[9] The Frobenius norm of a $m \times n$ matrix A with cell elements a_{ij} is $\|A\|_F \triangleq \sqrt{\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2}$.

With the result achieved in Equation 24 the derivative in Equation 21 can finally be calculated,

$$\frac{\partial}{\partial x_i}(r_j) = \alpha_j \frac{\partial}{\partial x_i}(g_j(S_j))(1-r_j) = -2^{5/2}\alpha_j c(\sigma_{min} n)^{-2}(1+cd^2)^{-3}(x_i - x_j)(1-r_j) \tag{25}$$

Now only $\frac{\partial}{\partial x_i}(\gamma_{ij})$ is missing to complete the calculation of Equation 20.

$$\begin{aligned}\frac{\partial}{\partial x_i}(\gamma_{ij}) = \quad & \frac{\partial}{\partial x_i}\left(\frac{g_j(i)}{g_j(S_j)}\right) = \left\{\text{since } D(\tfrac{f}{g}) = \tfrac{f'g - fg'}{g^2}\right\} = \\ & \frac{\frac{\partial}{\partial x_i}(g_j(i))g_j(S_j) - g_j(i)\frac{\partial}{\partial x_i}(g_j(S_j))}{g_j(S_j)^2} = \left\{\text{since } \tfrac{\partial}{\partial x_i}(g_j(S_j)) \equiv \tfrac{\partial}{\partial x_i}(g_j(i))\right\} = \\ & \frac{\frac{\partial}{\partial x_i}(g_j(i))}{g_j(S_j)}\left(1 - \frac{g_j(i)}{g_j(S_j)}\right) = \frac{\frac{\partial}{\partial x_i}(g_j(i))}{g_j(S_j)}(1-\gamma_{ij})\end{aligned} \tag{26}$$

The partial derivative $\frac{\partial}{\partial x_i}(\gamma_{ij})$ is now completely known since γ_{ij} and $g_j(S_j)$ are known from Equation 14 and $\frac{\partial}{\partial x_i}(g_j(i))$ from Equation 22.

Now, the partial derivative of the measurement reward function, $\frac{\partial}{\partial x_i}(r_i^m)$, is completely determined by inserting Equations 21 and 26 into Equation 20.

The other partial derivative, $\frac{\partial}{\partial y_i}(r_i^m)$, and is equivalent except for only the factor $(x_0 - x_j)$ in Equation 25 which should be replaced with $(y_i - y_j)$.