Advance Access publication on February 25, 2010
doi:10.1093/comjnl/bxq021

A Framework for Cooperative Control Applied to a Distributed Sensor Network

ANTONY WALDOCK* AND DAVID NICHOLSON

Advanced Technology Centre, BAE Systems, Filton, Bristol, UK
**Corresponding author: antony.waldock@baesystems.com*

This paper proposes and evaluates a framework for cooperative control of a multi-agent system. The framework is evaluated on a target-tracking application where a distributed sensor network is tasked to autonomously observe targets within the environment. The problem of cooperative control is defined using two distinct levels of cooperation: implicit and explicit. Implicit cooperation is defined as cooperation through only the exchange of environmental data to compile a common picture over which to reason locally. For example, in this paper, decentralized data fusion algorithms are used to build and update a common picture of the target positions and velocities. Explicit cooperation, which is the main focus of this paper, negotiates the agents explicitly on a joint set of actions to perform. In this paper, the problem of explicit cooperation is formulated as a distributed optimization, and a framework to find the joint set of actions is proposed. The framework utilizes two algorithms, the Max-Sum algorithm, to globally solve a factorizable utility function, and Probability Collectives (PC), to solve the individual factors of the utility function. The paper presents experimental results of the two algorithms using a simulated distributed sensor network when the tracking problem is and is not factorizable. The results show that the proposed framework can efficiently and effectively enable cooperation in a distributed sensor network. The Max-Sum algorithm provides a distributed and flexible approach to solve a factorizable utility function, where the PC algorithm was shown to efficiently solve the individual factors when more than four sensors are required to cooperate.

Keywords: cooperative control; distributed sensor networks; probability collectives; Max-Sum

Received 4 November 2009; revised 27 November 2009
Handling editor: Alex Rogers

1. INTRODUCTION

Cooperation in a multi-agent system (MAS) seeks to achieve some desired collective effect through the execution of individual actions. The requirement for cooperation cuts across a range of civil and military planning and control applications. Constructing a suitable set of individual decision strategies can be difficult when there is any degree of complexity in the required collective behaviour, e.g. coping with a dynamic, uncertain and hostile environment. In most practical systems, the decision strategy is determined centrally using a 'trial and error' approach to define the necessary interactions [1]. This 'bottom-up' approach can be labourious and, though it can result in useful systems, provides no guarantees of system functionality or stability. Consequently, this work adopts a 'top-down' framework for achieving cooperation by posing the problem as distributed optimization.

In the following sections, this paper first outlines the related work in the area of cooperation applied to a distributed sensor network. The example target-tracking problem is outlined in Section 3 with a definition of the varying levels of cooperation in Section 4. The two levels of cooperation detailed in this paper, implicit and explicit, are then defined in Sections 5 and 6, respectively. The remainder of the paper is then divided into two sections, which present the experimental results of the Max-Sum and Probability Collectives (PC) algorithm to find the joint action set for the overall factorizable utility function and then for the individual factors. The conclusions and future work are then discussed in the final two sections.

2. RELATED WORK

Cooperation applied to a distributed sensor network has extensively been studied in two main problem settings: cooperative control of sensor-to-target assignments and cooperative control of mobile sensor trajectories. The focus of the current work is on sensor-to-target assignment for a

multi-target-tracking application, but the techniques described are equally applicable to cooperative control of a variety of assets engaged in an information gathering task.

Sensor-to-target assignment problems are characterized by a number of sensors, each with a restricted view of the world due to physical constraints. For example, if the world is populated by several targets, each sensor may only be able to observe one of the targets in a single time instant. The problem is to assign sensors to targets in order to maximize some measure of system-wide performance, such as the combined quality of target state estimates. A number of studies promote a centralized solution to the problem by formulating it using linear programming with suitable constraints [2]. An example of this approach was reported by Kastella [3], who used mutual information as a global utility function for managing a group of radar sensors engaged in airborne target tracking. The sensor-to-target optimization problem has also been solved in a centralized fashion using simulated annealing [3] to optimize hundreds of Doppler radar sensors. However, centralized solutions in general do not scale very well and are not robust to failures because they rely on a single centralized processing node.

Distributed approaches have also been applied to the sensor-to-target assignment problem. Isler *et al.* [4] refer to it as a focus-of-attention problem and develop approximate solutions that exploit sensor geometry and leverage results from maximal set packing. Horling *et al.* [5] provide a summary of MAS architectures and algorithms applied to sensor resource allocation. Algorithms that exploit the decentralization of gradient-based search are also outlined in [6]. This study also employs decentralized Kalman filters to compose the global utility function. These approaches are highly reliant on the shape and nature of the objective function that cannot be guaranteed because it is highly dependent on the sensor model used.

A particularly interesting study of cooperative sensing is reported by Grocholsky *et al.* [7]. This work is notable for distinguishing cooperative solutions based on explicit negotiation and coordinated solutions based only on observed information exchange. Although the focus of this work was cooperative control of sensor trajectories, an example was also presented for a sensor-to-target assignment scenario [8]. This example used Distributed Data Fusion algorithms and optimized assignments with respect to information-theoretic objective functions. It employs a distributed optimization algorithm known as iterated better response. For continuous optimization problems, this can be realized by solving a nonlinear Jacobi equation using a distributed implementation of sequential quadratic programming [9]. However, this requires a smooth and differentiable objective function. In practice, it is often required to introduce heuristics (such as asynchronous updates, noise terms and randomized update orders) to prevent oscillations and help escape unstable equilibrium or weak local optimum solutions.

In summary, although there are many solutions to the cooperative sensing problem, they often suffer from either poor scaling (centralized), limiting assumptions (smooth and continuous cost functions) or unprincipled update mechanisms (e.g. arbitrary annealing schedules). The prime purpose of this work is to investigate two powerful distributed optimization approaches such as the Max-Sum algorithm and PC that do not suffer from these limitations to enable cooperative decision-making in a group of agents.

3. TARGET TRACKING APPLICATION

To provide a concrete investigation into cooperative behaviour in a MAS, a representative problem is required. Owing to the maturity of the domain, cooperation within a distributed sensor network was used. Although, all the methods of cooperation proposed and investigated within this work are equally applicable to other applications that can be represented as a MAS.

The agents of interest are sensors that are networked together and engaged in tracking multiple targets in their environment. Each sensor can take a resource-constrained action (orientate towards a particular target in the environment), which results in measuring the position of a single target. In this scenario, two or more sensors are allowed to form a coalition and measure the same target. The overall aim is to select a set of joint actions (control parameters for each sensor), which reduces the total amount of uncertainty associated with position and velocity estimates of the targets across the entire sensor network. An example of a target-tracking scenario is illustrated in Fig. 1, which shows three static sensors and three moving targets, but the framework developed in this paper is not limited to static sensors. The next section introduces the problem of cooperative control and defines the varying levels of cooperation.

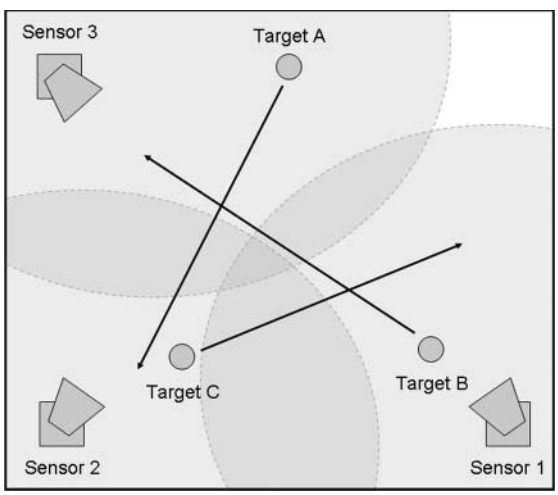

FIGURE 1. An example tracking scenario with three sensors (shown with a limited range by the grey shaded areas) and three targets.

4. LEVELS OF COOPERATION

Cooperative control in a MAS could be performed using a multitude of different mechanisms. Figure 2 shows the high-level processes required by an agent: sensing, inference, control and actuation. Cooperation could occur at any of these levels. For example, an agent could directly stream the output of its sensor to another agent for inference. This method of cooperation is typically undesirable because of the communication bandwidth required. This work focuses on two levels of cooperation, which have been defined as implicit and explicit cooperation.

For implicit cooperation, the agents share/fuse information about the environment to compile a common picture. The agents then reason independently at the control level using the same common picture of the environment. In this paper, standard decentralized data fusion (DDF) algorithms (briefly outlined in Section 5) are used to compile this common picture of the positional and velocity estimates of the targets in the environment. As stated later in this work, reasoning over a common picture can result in the selection of cooperative actions, but in some situations agents can select conflicting plans, i.e. both agents try to achieve the same goal ignoring another.

For explicit cooperation, which is the main focus of the investigations, the common picture is used to synthesize a global utility function, which can be used to select a joint set of actions (control parameters for each sensor). Hence, each agent will select the action that maximizes the entire system performance not just their individual performance. Before a framework to enable explicit cooperation can be discussed, the next section briefly describes implicit cooperation and the picture compilation algorithms used in this paper.

5. IMPLICIT COOPERATION

This section defines implicit cooperation including the picture compilation algorithms used and the local decision mechanisms. DDF is a robust, modular and scalable solution to the problem of obtaining common and consistent state estimates (e.g. target types and positions) across a sensor network [10]. DDF imposes architectural constraints on the sensor network, which eliminate the conventional notion of a fusion centre as well as access to full knowledge of the global network topology by each sensor node. DDF also defines probabilistic information update algorithms that map a variety of sensor network architectures. The algorithms are implemented at each sensor node, to filter and fuse their local data and to assimilate processed data from the other nodes.

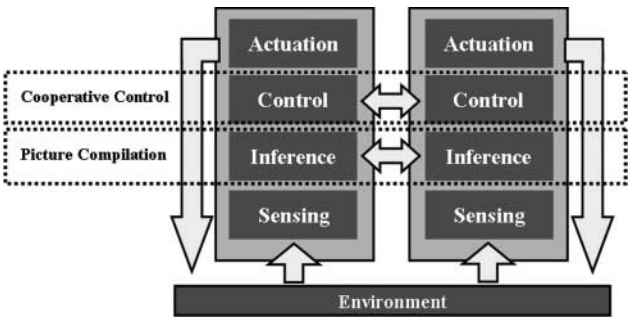

FIGURE 2. Levels of cooperation in a distributed sensor network.

In this paper, the sensor network comprises N stationary sensors engaged in tracking M mobile targets in their environment. The sensors implement DDF algorithms to estimate the dynamic states (position, velocity) of the targets. Interleaved within each node's DDF process is a target assignment algorithm that informs the sensor nodes about which target to observe, given the constraint they can only observe one-out-of-N targets at each sensing opportunity. However, two or more sensors may simultaneously observe the same target. The DDF algorithm steps are as follows:

Algorithm 1 DDF Algorithm

1. Predict the new target state (x,y,z,vx,vy,vz)
2. Select the control parameters for the sensor (pan angle θ)
3. Measure target state (x,y,z)
4. Communicate target measure to other sensors
5. Wait for communicated messages for X milliseconds
6. Update target states with the received measurement(s)

The DDF algorithms maintain *information* states about the targets for computational and communication efficiency. However, information also provides a direct normative basis on which to select the control parameters. The key quantity is the Fisher information matrix, $\mathbf{Y}(k|k)$, which is calculated directly by the information form of the Kalman filter. The notation $(k|l)$ refers to an estimate at time k conditioned on all observations up to and including time l.

First in Step 1, each sensor's information filter predicts the target state $\mathbf{Y}(k|k-1)$ using a motion model for the specific target under track. The experiments in this paper assume a linear motion model with additive Gaussian process noise.

The second step is to select the control parameters θ_i to reconfigure the sensor i. A common strategy is to select the control parameters to maximize the predicted information $I_{i,j}(k)$ for sensor i observing target j at time k. Each sensor i uses its observation model to predict the amount of information, $I_{i,j}(k)$, associated with observing each target j at time k. In this paper, Equation (1) is used to evaluate the sum of the information (or entropy) over all the targets for a single sensor i using control parameter θ_i.

$$I = \sum_{j=1}^{M} \left(\frac{1}{2} \log(2\pi e)^9 (|\mathbf{Y}_{i,j}(k|k-1) + I_{\theta_i,j}(k)|) \right). \quad (1)$$

When sensor i has been manipulated using the control parameter θ_i and observed one or all of the targets (in Step 3), the observed information $\mathbf{I}_{i,j}(k)$ is sent to all other sensors in the sensor network. The target measurements can be communicated through a globally broadcast message or propagated across the network between sensors through a point-to-point protocol. Each sensor then assimilates its own information about the target with the information it receives about the target from its communication channels (it is assumed that the information can be associated without error). The assimilation equation has the advantage of being additive in DDF:

$$\mathbf{Y}_{i,j}(k|k) = \mathbf{Y}_{i,j}(k|k-1) + \sum_{i=1}^{N} \mathbf{I}_{i,j}(k). \tag{2}$$

The performance of the sensor network is evaluated by measuring the uncertainty in the positional accuracy, as in the following equation:

$$I = \sum_{j=1}^{M} \left(\frac{1}{2} \log(2\pi e)^9 (|\mathbf{Y}_{i,j}(k|k)|) \right). \tag{3}$$

In the experiments outlined later in this paper, the DDF messages are given sufficient time to fully propagate around the network. Therefore, the DDF layer provides a common state estimate for the targets at each node in the sensor network. Notice that the exchange and assimilation of observation information in DDF couples future sensor-to-target assignment decisions, leading to coordinated decisions and was defined earlier as a method of implicit cooperation.

Consequently, the implicit cooperation strategy (commonly referred to in sensor management as a 'selfish' or local strategy) is simply to assign sensors to targets without having the sensors explicitly negotiate to account for each others' preferences. However, negotiation is expected to improve performance at the expense of additional communication between the sensors. The following section defines the problem of explicit cooperation and how it relates to the common picture built using DDF.

6. EXPLICIT COOPERATION

The previous section presented an approach to building and maintaining a common distributed picture of the position and velocities of targets within the environment. This section addresses the problem of selecting the control parameters to control or reconfigure the sensors to maximize the information gathered. Typically, the selection of the control parameters is performed locally such that each sensor maximizes the global information, given the local measurements possible by this sensor alone (Equation (1)). In this paper, the aim is to select the control parameters for all sensors that maximize the sum of information for all targets across the entire sensor network and not just a single sensor.

The problem of cooperative control can be defined as a distributed optimization problem where the utility function U is defined as the sum of the predicted information over all targets:

$$U = \sum_{j=1}^{M} \left(\frac{1}{2} \log(2\pi e)^9 (|\mathbf{Y}_{i,j}(k|k-1) + \sum_{i=1}^{N} I_{\theta_i,j}(k)|) \right), \tag{4}$$

where M is the number of targets, N is the number of sensors in the network and $I_{\theta_i,j}$ is the predicted information for target j given that sensor i is controlled or configured using control parameter θ_i. With regards to the distributed optimization required for explicit cooperation, there are two aspects of the utility function that are important.

Equation (4) states that given a set of control parameters (θ_i), the agent predicts the information gain (I_θ) for all the measurements that could be taken. This utility function maximizes the positional and velocity estimates of the targets, but in reality it is likely to include further terms that incorporate power requirements for sensing, time to execute the action or probability of acquisition etc.

The addition of further terms are likely to increase the complexity of the shape and nature of the utility function and decrease the ability to use standard optimization techniques, which, for example, rely on a convex structure. The existing techniques (outlined in Section 2) that address explicit cooperation as a distributed optimization are either centralized or rely on a smooth and differentiable utility function. The form of the utility function is highly dependent on the sensor model used to generate the predictions of the information gain (I) and is likely to include discontinuities and irregularities. The framework proposed in this paper utilizes two algorithms to optimize the utility function: Max-Sum for the overall utility function in Section 7 and PC for the individual factors in Section 8.

7. EXPLOITING FACTORIZATION OF THE UTILITY FUNCTION

This section outlines a method of efficiently solving the distributed optimization, to enable cooperation, when the utility function (as defined in the previous section) is factorizable. First, a method of factorizing the utility function is discussed and a method of building a factor graph that utilizes information from the DDF layer is proposed. This is followed by an introduction to the Max-Sum algorithm that is used to find the set of control parameters that maximize the utility function and the experimental results of applying the technique to the problem outlined in Fig. 1 are presented.

7.1. Factorizing the utility function

In regard to the distributed optimization required for explicit cooperation, there are two aspects of the structure of the

utility function that are important. First, the utility function U for this application is a summation over the predicted global information for all targets. In the general case, this is likely to be true for agents where the result of performing an action does not directly affect the performance of another agent. Hence, the utility function can be divided into a smaller set of factors that could be maximized individually. Factorizing the utility function to a smaller set of functions reduces the overall complexity of the optimization process and enables a distributed approach to be efficiently utilized. Although, it is worth remembering that this factorization step may result in an approximation of the actual problem. This is highlighted in the following example.

Figure 3 shows a sensor network with five sensors (represented using squares) and three targets (shown with circles). Each sensor has a limited range (which is shown using the dotted lines in the figure) and therefore each sensor can only observe a subset of the targets in the environment. In reality, the observational range of a sensor is unlikely to be such a sharp discontinuity in all directions and this process will be harder in practice than this simple example. Therefore, the resultant factorized utility function is likely to be an approximation of the actual utility function but the degree of approximation error introduced is dependent on the application.

In the example above, the utility function $U(\theta_1, \theta_2, \ldots \theta_N)$ can be defined in terms of a summation of the target factors U_j over only the sensors that can observe that target (j). In the example, the utility function can be factorized in the following way:

$$U(\theta_1, \theta_2, \ldots, \theta_N) = U_A(\theta_1, \theta_2, \theta_3) + U_B(\theta_2, \theta_4) + U_C(\theta_5), \tag{5}$$

where U_B is defined as

$$\frac{1}{2}\log(2\pi e)^9(|\mathbf{Y}_{i,j}(k|k-1) + I_{\theta_2,B}(k) + I_{\theta_4,B}(k)|) \tag{6}$$

when the utility function is to maximize the predicted global information. This factorization is represented in Fig. 4, where

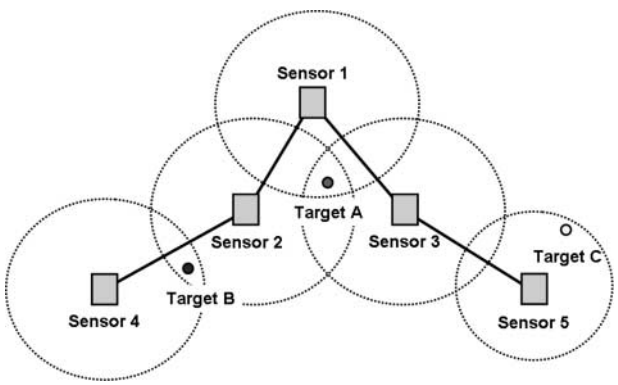

FIGURE 3. Example sensor network with targets.

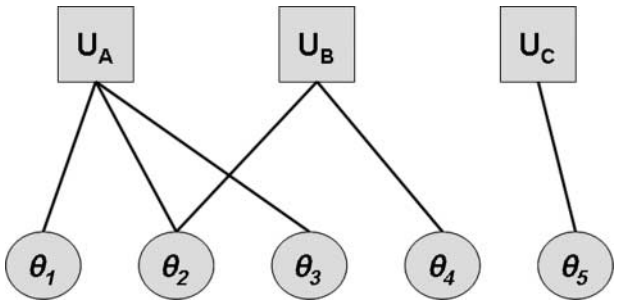

FIGURE 4. Example factor graph.

the sensor control parameters θ_i are represented as circles and the factors of the utility function are shown as squares. This is termed a factor graph where the interconnections between the variables and the edges represent a dependency in the factor.

The definition of U_B in Equation (6) highlights the limiting factor in the decomposition of the utility function. This factor of the utility function appears to be also composed of a summation and hence the same rule as before should apply. Unfortunately, the summation of the predicted track information $(\mathbf{Y}(k|k-1))$ and the predicted measurement information $(I(k))$ is surrounded by a determinant. This dictates that $|I_{\theta_2,B}(k) + I_{\theta_4,B}(k)|$ does not equal $|I_{\theta_2,B}(k)| + |I_{\theta_4,B}(k)|$, therefore, this portion of the objective function cannot be broken down into smaller segments and therefore, must be optimized as a single factor.

Section 8 addresses how to tackle these factors of the utility function (one that cannot be factorized further) using a distributed optimization technique based on PC. The following section addresses how a factor graph of the utility function can be built and maintained for a tracking application and then solved to find a globally optimal set of joint control parameters.

7.2. Building and solving the factorized utility function

This section is divided into two parts, first it addresses the problem of how to build and maintain the factor graph that represents the utility function and secondly, how the Max-Sum algorithm can be used to derive the control parameters that maximize the utility function through a simple message passing scheme. The ability to use a simple message passing scheme enables a decentralized approach to be taken.

7.2.1. Communicating observable tracks

This section outlines how the factor graph of the utility function can be constructed for a tracking application using the underlying DDF process. The constructor of the factor graph occurs in two stages: variable nodes and factor nodes. At initialization, each of the sensors creates a variable node that represents the control parameter (θ_i) to be adjusted.

The second stage is to define a single factor (U_j) for each target (j) observable in the entire sensor network. The sensor

responsible for spawning the original target track is tasked with managing the factor node for that target. At the end of the initialization stage, the factor graph for the utility function exists in the form of both the variable and factor nodes. During the scenario, the edges of the factor graph must continually be updated depending on the position of the targets in the environment.

The key to factorizing the utility function is to identify the subset of sensors capable of observing the individual targets. The edges of the factor graph are built using observation data derived from the underlying DDF layer. The track measurements, exchanged during the distributed picture compilation process, are augmented to include the list of targets that a sensor is capable of observing. This set of observable targets is used by each sensor to build a mapping between the target and observable sensors. Each sensor must maintain this mapping only for those targets (factors) that it is responsible for. Every time a target moves within or out of range of a sensor, the underlying factor graph is updated by adding or removing the corresponding edge. The following section describes how the factor graph can be used with a message passing scheme to find the set of variables that result in the maximum utility.

7.2.2. *The Max-Sum algorithm*

This paper proposes solving the resultant factor graph using the Max-Sum algorithm as defined in [11]. The Max-Sum algorithm, which is a variant of the sum-product algorithm [24], exploits the factorizable form of the utility function to efficiently find the maximum control parameters (for only discrete variables) using a simple message passing scheme. The Max-Sum algorithm and variants have been exploited in 'loopy' belief propagation in Bayesian networks [12]. In this section, the Max-Sum is adapted to optimize a sensor to target assignment in a distributed sensor network.

The Max-Sum algorithm computes the maximization of the utility function by computing summaries at the variable and factor nodes and sending appropriate messages along the edges of the factor graph. These messages are split into two types: messages ($Q_{i\to j}$) from variable nodes to factor nodes and messages ($R_{j\to i}$), in the reverse direction, from factor nodes to variable nodes. At initialization, and each time the factor graph is altered, the values of Q and R are set to an initial small random number, which is used to help break any symmetry within the optimization. As previously stated, the variable nodes represent the control parameter (θ_i) at each sensor (i) and the function nodes represent the utility (U_j) for each target (j). The messages represent summaries of the utility function available at the variable and factor nodes and defined as follows:

From variable to function:

$$Q_{i\to j}(\theta_i) = \alpha_{ij} + \sum_{j'\in M(i)\setminus j} R_{j'\to i}(\theta_i), \tag{7}$$

where α_{ij} is a scalar to normalize the value such that

$$\sum_{\theta_i} Q_{i\to j}(\theta_i) = 0. \tag{8}$$

From function to variable:

$$R_{j\to i}(\theta_i) = \max_{\theta'_j\setminus i}\left(U_j(\theta'_j) + \sum_{i'\in N(j)\setminus i} Q_{i'\to j}(\theta_{i'})\right), \tag{9}$$

where R is an approximation of the utility function, Q is the preferences for each control parameter represented by this variable and U_j is the utility for target j as defined by Equation (6). The function $M(i)$ represents the set of factor nodes that this variable is connected to in the graph and $N(j)$ corresponds to the set of variables nodes that this factor node is connected to. Both $M(i)$ and $N(i)$ are derived from the process in the previous section using the set of observable targets from each sensor. The parameter θ'_j represents the set of all the control parameters from all the sensors involved in observing target j in U_j. Typically, these factors are solved using a brute-force approach (the evaluation of the utility of all combinations for all the control parameters). The scalability of this approach will be highly dependent on the number of variables (control parameters) involved in the factor and a comparison between a new method based on PC and a brute-force approach are presented in Section 8.

The resultant control parameters are derived from the marginal function that is calculated by summing the approximation R. The results from [11] show that, with sufficient iterations, the marginal function will approximate $\sum_{j=1}^{M} U_j$, the utility function. The version of Max-Sum used in this paper uses a normalization factor (α) to cope with loops in the graph structure. Recent research by Stranders *et al.* [13] has shown a new bounded approach that iteratively removes cyclic references while maintaining an estimate of the deviation from the initial (fully connected) graph. The resultant algorithm to derive the control parameters for a single sensor in a distributed sensor network is presented in Algorithm 2.

Algorithm 2 Max-Sum algorithm

1. Update the list of sensors that can observe the assigned track(s)
2. **if** list of sensors has changed **then**
3. Rebuild the Factor Graph
4. Initialise function node for the assigned track(s)
5. **end if**
6. Evaluate the utility function (U_j) for assigned tracks (j)
7. Exchange Function (R) and Variable Messages (Q) for Y milliseconds
8. Run the DDF algorithm 1 using the assigned value of the variable as the control parameter

7.3. Experimental results

In this section, the performance of a local, centralized and decentralized (Max-Sum algorithm) strategy is applied to the target-tracking application. The sensors and targets are arranged as in Fig. 1 where the sensors are stationary and the targets move as indicated. Each of the sensors has a limited observation range, which is shown by the grey areas bounded by the dashed lines. This scenario was selected because it requires the factor graph to be modified as the targets move and selection of the optimal control parameters requires optimization of the utility function rather than purely the local factors.

All the sensors were initialized with a weak prior as to the position of all the targets and were given the targets (factors) they are responsible for maintaining in the factor graph. The performance of the application was evaluated using the global information as defined in Equation (3) from one sensor (the DDF network was given sufficient time to enable all track measurements to be propagated around the network before proceeding—hence the picture is common across all sensors).

The initial results presented in Fig. 5 show the performance of a local, centralized and decentralized assignment strategy on the scenario. The local assignment strategy selects the control parameter (pan angle) that maximizes the total global information given the measurements taken only by this sensor. The centralized assignment strategy chooses the control parameters for all the sensors using a brute-force approach to find the assignment that results in the maximum global information for all combinations of assignment. The decentralized strategy solves each of the factors using a brute-force approach and then uses the Max-Sum algorithm to derive the maximum for the utility function.

Figure 5 shows the total global information for all the targets in the scenario. In the first 10 steps of the scenario, the total global information increases as the estimate of a target's position and velocity improves in accuracy with each new measurement. The performance of all the assignment strategies is equivalent at this stage in the scenario because the targets are initialized sufficiently close to the sensors to make a local assignment strategy globally optimal. As all the targets pass through the centre of the environment, each sensor must handover the tracked target to another sensor. Depending on the noise in the individual scenario, the two handover points occur at approximately time steps 18 and 34. Figure 5 shows that the local strategy is significantly lower than both the centralized and decentralized strategies at these handover points in the scenario. Although the local assignment strategy is based on a common picture, the control parameters selected require a conflict resolution to prevent the selection of the same target and therefore achieve a clean handover.

Given sufficient time (Y in Step 7 of Algorithm 2), the decentralized approach achieves the same level of performance as the centralized approach. The performance of the decentralized approach is then evaluated as the time allowed to exchange variable and factor messages is reduced. Figure 6 shows the performance compared with the centralized approach as the negotiation time (period to exchange variable and factor messages) is adjusted from 50 to 1000 ms (the experiment was conducted 20 times for each negotiation time).

Figure 6 shows that with only 50 ms of negotiation time, the Max-Sum algorithm does not have the time to exchange a sufficient number of messages to find the maximum control parameters. The high variance in this result is driven from the initial random preferences used. With only a small number of messages exchanged, these initial random preferences still have a pronounced effect. Although, it is worth remembering that with only a few messages, the decentralized approach achieved 1089 where the local assignment strategy achieved only 1049. As the negotiation time is increased, the performance of the decentralized strategy achieves that of the centralized approach.

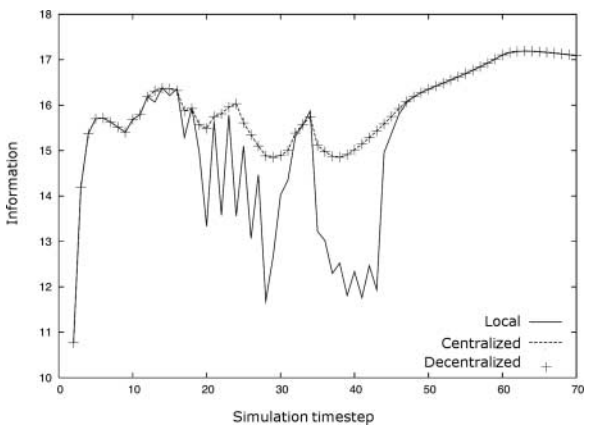

FIGURE 5. Information over all the targets for each time step of the simulation for a Local, Centralized and Decentralized assignment strategy.

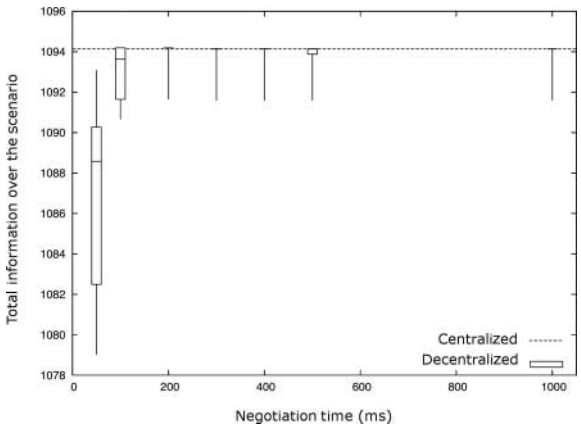

FIGURE 6. Total Information (sum of the information during the entire scenario) for a Decentralized assignment strategy as the negotiation time (Y) is adjusted.

8. EFFICIENTLY SOLVING THE INDIVIDUAL FACTORS

The previous section showed how a factorized utility function could be solved globally. The solution relies on being able to break the problem into small enough factors that can be solved using brute force. In this section, a technique called PC is shown to solve factors that cannot be solved efficiently using brute-force techniques. The section begins with an introduction to PC and then presents the experimental results of PC compared with a brute-force approach on a target-tracking application.

8.1. Probability collectives

PC is a broad framework for analysing and controlling distributed systems, with deep theoretical connections to game theory, statistical physics and optimization [14]. PC algorithms have been applied with success to a range of benchmark optimization problems and have been demonstrated for a distributed flight control application [15].

Typically an optimization problem is solved by manipulating a set of optimization variables, in a deterministic or stochastic fashion (e.g. simulated annealing), until some global cost function of those variables is minimized. PC regards the variables as independent agents playing an iterated game. However, what is manipulated by PC is probability distributions over those variables. The manipulation process seeks to induce a distribution that is highly peaked about the value of the variables that optimize the global objective function. This approach has the following advantages:

(i) it permits a distributed solution in the sense that each agent's update at time t is independent of any other agents' update at that time [16].
(ii) it can be applied in the same general way regardless of whether the variables are continuous, discrete, time-extended, mixtures etc. [17].
(iii) it is robust in the sense that the cost function can be irregular or noisy [18].
(iv) it provides sensitivity information about the problem in the sense that a variable with a peaky distribution is more important to the solution than a variable with a broad distribution

A key result of PC is that the minimum value of the cost function can be found by considering the maxent Lagrangian equation for each agent (variable). This is written as

$$\mathcal{L}_i(q_i) = g_i(q_i) - T \times S(q_i). \tag{10}$$

Here, q_i is agent i's probability distribution over its actions denoted by θ_i; $g_i(q_i)$ is the expected cost evaluated with respect to the distributions of the agents other than i; T is temperature (or beta which is defined as $1/T$); $S(q_i)$ is the entropy associated with the probability distribution q_i. PC algorithms are still being actively researched and matured, but for the purpose of this paper, Algorithm 3 is employed to optimize the Lagrangian.

Algorithm 3 PC optimization

1. beta $\leftarrow beta_{min}$
2. **repeat**
3. iterations $\leftarrow 0$
4. **repeat**
5. Generate a Sample Block using q_i from each agent
6. Evaluate the expected global cost $g_i(q_i)$
7. Update q_i using $g_i(q_i)$
8. iterations = iterations + 1
9. **until** (iterations $> I_{max}$) OR ($\delta q_i < q_{min}$ for all i)
10. beta $\leftarrow$ alpha $\times$ beta
11. **until** beta $> beta_{max}$

The maxent Lagrangian is convex over the set of product distributions over the agent's action space. By operating on q_i in this convex space, we are able to use powerful search methods for finding function extrema developed for continuous domain problems, such as gradient descent. Note that while adding entropy makes the descent easier, it also biases the solution away from extreme solutions. That bias is gradually lowered by annealing the temperature T.

The minimization of the Lagrangian is amenable to solution using gradient descent or Newton updating since both the gradient and the Hessian are obtained in a closed form. Using Newton updating and forcing the constraint on total probability, the following update rule is obtained:

$$q_i(\theta_i) \longrightarrow q_i(\theta_i) - \alpha q_i(\theta_i)\frac{E[G|\theta_i] - E[G]}{T} + S(q_i) + \ln q_i(\theta_i), \tag{11}$$

where θ_i is agent i's action and G is the global cost function. The parameter α plays the role of a step size since the expectations result from the current probability distributions of all the agents. Constraints can be included by augmenting the cost function with Lagrange multipliers and the constraint functions.

Performing the update involves a separate conditional expected utility for each agent. These are estimated either directly if a closed form expression is available or with Monte Carlo sampling if no simple closed form exists. Monte Carlo sampling is the most general approach and it is used for the experimental results described later in the paper.

In Monte Carlo sampling, the agents repeatedly and jointly *i.i.d.* sample their probability distributions to generate joint actions, and the associated costs/utilities are recorded. Since accurate estimates usually require extensive sampling, the global cost G occurring in each agent i's update rule can be replaced with a private cost g_i chosen to ensure that the Monte Carlo estimation of $E(g_i|x_i)$ has both low bias, with respect to estimating $E(G|x_i)$ and low variance. Now that the PC algorithm has been defined, the global cost function G used to enable cooperative behaviour must be identified.

The function (U_j) to optimize within PC is dependent on the factor graph produced in the previous section, but the

optimization will require communication between all sensors involved in that factor. Previous research has shown that the PC algorithm can be implemented using a decentralized approach [19].

8.2. Experimental results

This section presents experimental results of using PC to find the set of joint control parameters for the target-tracking problem outlined in Fig. 1 when the utility function is not factorizable (without significant approximation errors). The essence of the target-tracking scenario is the same, but it was altered from the previous section to allow all the sensors to observe all targets all the time. Hence, the utility function consists of a single factor.

This section reports performance results, metricated in terms of the mean global information (defined in Equation (3)), for three different sensor-to-target assignment strategies: random, selfish and PC. The performance result is the mean of 50 trials where noise is applied to both the sensor measurement and the motion model in the form of process noise. PC uses a block containing 20 samples with a maximum of five iterations, a minimum delta of 0.03 to determine convergence and beta initialized at 0.1, increasing with a rate of 1.5 until beta reaches 100.

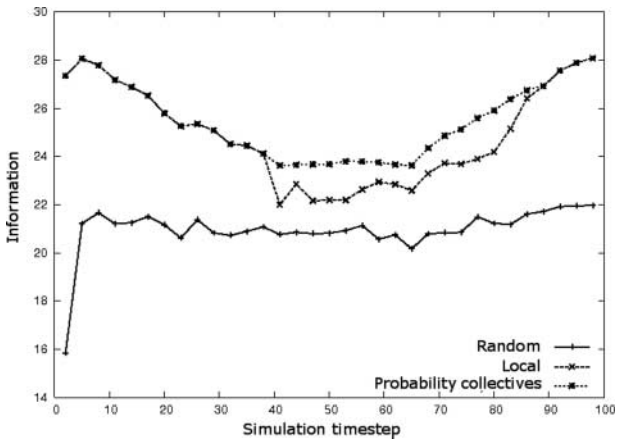

FIGURE 7. Information for a random, selfish (local) and PC strategy during the target-tracking scenario.

Figure 7 shows the performance of the three different assignment strategies. The results show that a cooperative strategy results in similar performance to a local strategy at the beginning and end of the scenario (as seen in the previous experimental results). As before, a cooperative strategy results in higher global information during the middle of the trial when the sensors are required to perform a handover of the targets. A centralized solution was conducted but is not presented in Fig. 7 because the results are the same as the PC assignment. These results indicate that explicit cooperative strategies are important in resolving conflicts in sensor assignment.

Therefore, a PC approach can be used to solve a utility function that cannot be factorized, but how does this compare with a brute-force approach? Figure 8a shows the performance of the strategies as the number of control parameters (sensors) increases and Fig. 8b shows the number of samples required by a brute-force and PC strategy with different numbers of sensors.

Figure 8a shows that the difference in performance of an explicit and implicit cooperation strategy (PC and selfish in this experiment) increases as the number of sensors increases. Although, the exact improvement is likely to be highly dependent on precisely the scenario selected. Figure 8b compares the number of samples required by PC to those used by a brute-force or optimal approach. The complexity of the brute-force approach increases exponentially, as the number of sensors is increased, while the complexity (samples used) of the PC algorithm remains relatively constant. These results show that when relatively few sensors (control parameters) are

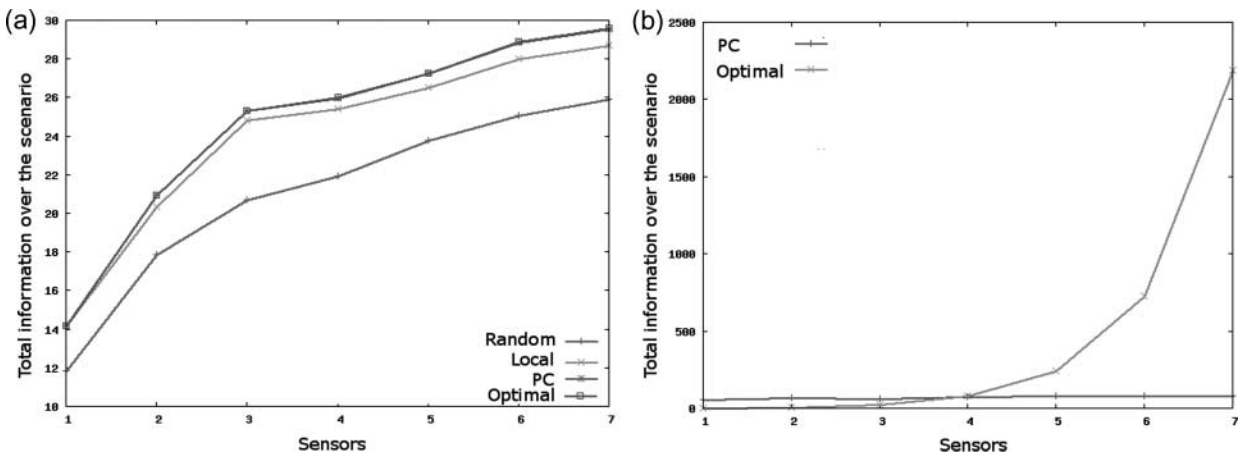

FIGURE 8. (a) The sum of the information during the scenario for different strategies (random, selfish or local, PC, brute-force or optimal) with varying numbers of sensors. (b) Comparison of the total number of samples required for a optimal (brute-force) and PC strategy with varying numbers of sensors.

involved in the factor, then a brute-force approach is suitable but when factors involve more sensors (approximately four and above) then a more scalable approach, such as PC, is essential.

9. CONCLUSIONS

This paper has proposed a framework for cooperative control and the performance evaluated on a simulated target-tracking application. The problem of enabling cooperation in a distributed sensor network was then defined in terms of a distributed optimization and the limitations with the existing approaches discussed. Cooperation between sensors is achieved by solving a distributed optimization problem based on the expected information gain of the predicted target measurements. As a result of these simulations and analysis, the following main conclusions were drawn:

(i) A cooperative assignment strategy was shown to outperform a local (implicit cooperation) strategy for a target-tracking application.
(ii) The Max-Sum algorithm was shown to efficiently and effectively enable cooperation across a distributed sensor network using a simple message passing scheme when the utility function can be factorized without introducing significant approximation errors.
(iii) A PC approach taken was shown to enable cooperation across a distributed sensor network when the utility function cannot be factorized and was shown to be computationally tractable (compared with a brute-force approach) when more than four sensors are involved in a single factor.

10. FUTURE WORK

A natural extension from this current research would be to develop and refine the two existing algorithms in isolation. For example, the Max-Sum algorithm is currently limited to discrete actions and the algorithm has a very basic termination condition (number of iterations performed). Further research could be targeted at extending the Max-Sum algorithm into continuously valued action spaces perhaps building on work recently conducted by Stranders *et al.* [20].

The second avenue for future work leaves behind these particular algorithms and poses a higher level question in regard to the relationship between implicit and explicit cooperation. In practice, given that implicit cooperation is likely to require less communication than explicit cooperation and communication is finite and a valuable resource, would performance be improved by dedicating more resources to implicit rather than explicit cooperation. Furthermore, what is the optimal balance between implicit and explicit cooperation for a given scenario or application?

The third avenue that stems from this research is the possible impact on the timeliness of the decision-making process when cooperation is included. For example, the Max-Sum algorithm may require a preset number of iterations to be completed before decision-making can be performed. Hence, the decision-making process is now limited to the speed of communication between two or more agents. Another approach would be to run the decision-making process at a different rate to the cooperation mechanism (be that Max-Sum or PC). With this approach, the decision strategy could be outdated (if the environment evolved at a faster rate than the communication) and the agent would not respond in a timely manner. A possible approach to this problem would be to estimate the confidence as well as the predicted gain for the cooperation mechanism.

The fourth avenue, and most general in nature, arises from the observation that although a cooperation mechanism can yield significant improvements, we fundamentally have a limited understanding of when a cooperation mechanism is required. For example, the only time when we know cooperation is beneficial is when the mechanism has finished computing the joint set of actions and the utility function is evaluated. Future work in this area should focus on predicting when cooperation mechanisms may improve performance and not just assume that they will run all the time. The most promising approaches to tackle this are through the application of machine-learning algorithms to predict when to run a cooperation mechanism or by estimating a confidence-bound which is associated with the cooperation mechanism.

ACKNOWLEDGEMENTS

The authors would like to thank David Wolpert (NASA Ames) and Dev Rajnarayan (Stanford University) for helpful discussions relating to Probability Collectives algorithms. Also, the authors would like to thank Alex Rogers, Nick Jennings and the research team at the University of Southampton for helpful discussions on the Max-Sum Algorithm.

FUNDING

This research was funded by the System Engineering for Autonomous Systems (SEAS) Defence Technology Centre (DTC) under the Algorithms and Architectures theme. Aspects of the material presented in this paper are protected by the European Patent Application No. 09250985.0 and Published International Patent Application No. WO 2009/063182.

REFERENCES

[1] Busetta, P., Bailey, J. and Ramamohanarao, K. (2003) A Reliable Computational Model for bdi Agents. *Proc. 1st Int. Workshop on Safe Agents Held in Conjunction with AAMAS2003*, Melbourne, Australia.

[2] Nash, J.M. (1977) Optimal Allocation of Tracking Resources. *Decision and Control Including the 16th Symp. Adaptive Processes and a Special Symp. Fuzzy Set Theory and Applications, 1977 IEEE Conf.*, New Orleans, LA, USA, December, pp. 1177–1180.

[3] Kastella, K. (1997) Discrimination gain to optimize detection and classification. *IEEE Trans. Syst. Man Cybern. Part A*, **27**, 112–116.

[4] Isler, V., Khanna, S., Spletzer, J. and Taylor, C. (2005) Target tracking with distributed sensors: the focus of attention approach. *J. Comput. Vis. Image Underst.*, **100**, 225–247.

[5] Horling, B., Vincent, R., Mailler, R., Shen, J., Becker, R., Rawlins, K. and Lesser, V. (2001) Distributed Sensor Network for Real Time Tracking. *5th Int. Conf. Autonomous Agents*, Montreal, June, pp. 417–424. ACM Press.

[6] Chung, T., Burdick, J. and Murray, R. (2006) A Decentralized Motion Coordination Strategy for Dynamic Target Tracking. *Robotics and Automation, 2006. ICRA 2006. Proc. 2006 IEEE Int. Conf.*, Orlando, FL, USA, May, pp. 2416–2422.

[7] Grocholsky, B., Makarenko, A., Kaupp, T. and Durrant-Whyte, H. (2003) Scalable Control of Decentralised Sensor Platforms. *Information Processing in Sensor Networks: 2nd Int. Workshop, IPSN03*, Palo Alto, CA, USA, pp. 96–112.

[8] Grocholsky, B. (2002) Information-theoretic control of multiple sensor platforms. PhD dissertation, University of Sydney, Australia.

[9] Barclay, A., Gill, P. and Rosen, J. (1997) Sqp methods and their application to numerical optimal control. *Numer. Anal. Rep.*, **97**, 207–222.

[10] Manyika, J. and Durrant-Whyte, H. (1994) *Data Fusion and Sensor Management: A Decentralised Information-Theoretic Approach*. Prentice Hall.

[11] Farinelli, A., Rogers, A., Petcu, A. and Jennings, N. (2008) Decentralised Coordination of Low-Power Embedded Devices Using the Max-Sum Algorithm. *7th Int. Conf. Autonomous Agents and Multi-Agent Systems (AAMAS-08)*, Estoril, Portugal, May, pp. 639–646.

[12] Murphy, K., Weiss, Y. and Jordan, M. (1999) Loopy Belief Propagation for Approximate Inference:An Empirical Study. *Proceedings of Uncertainty in AI*, Stockholm, Sweden, pp. 467–475.

[13] Stranders, R., Farinelli, A., Rogers, A. and Jennings, N. (2009) Decentralised Coordination of Mobile Sensors Using the Max-Sum Algorithm. *21st Int. Joint Conf. AI (IJCAI)*, Pasadena, USA.

[14] Wolpert, D. (2001) Collective Intelligence. In Fogel, D. and Robinson, D. (eds.), *Computational Intelligence Beyond 2001: Real and Imagined*. Wiley.

[15] Bieniawski, S. (2005) Distributed Optimization and Flight Control Using Collectives. Thesis, Stanford University.

[16] Macready, W. and Wolpert, D.H. (2004) Distributed Constrained Optimization. In Bar-Yam, Y. (ed) *Int. Conf. Complex Systems*, Boston, MA, May. Perseus books.

[17] Bieniawski, S., Kroo, I. and Wolpert, D. (2004) Discrete, Continuous, and Constrained Optimization Using Collectives. *Proc. 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conf.*, Albany, New York.

[18] Huang, C., Bieniawski, S., Wolpert, D.H. and Strauss, C.E.M. (2005) A Comparative Study of Probability Collectives Based Multi-agent Systems and Genetic Algorithms. *GECCO*, Washington DC, USA, pp. 751–752.

[19] Waldock, A. and Nicholson, D. (2007) Cooperative Decentralised Data Fusion Using Probability Collectives. *1st Int. Workshop on Agent Technology for Sensor Networks (ATSN-07). A Workshop at the 6th Int. Joint Conf. Autonomous Agents and Multiagent systems (AAMAS-07)*, Honolulu, HI, USA, pp. 47–54.

[20] Stranders, R., Farinelli, A., Rogers, A. and Jennings, N. (2009) Decentralised Control of Continuously Valued Control Parameters Using the Max-Sum Algorithm. *8th Int. Conf. Autonomous Agents and Multiagent Systems*, Budapest, May, pp. 601–608.