A Hybrid Continuous Max-Sum Algorithm for Decentralised Coordination

Thomas Voice and **Ruben Stranders** and **Alex Rogers** and **Nicholas R. Jennings** [1]

Abstract. In this paper we tackle the problem of coordinating multiple decentralised agents with continuous state variables. Specifically we propose a hybrid approach, which combines the max-sum algorithm with continuous non-linear optimisation methods. We show that, for problems with acyclic factor graph representations, for suitable parameter choices and sufficiently fine state space discretisations, our proposed algorithm converges to a state with utility close to the global optimum. We empirically evaluate our approach for cyclic constraint graphs in a multi-sensor target classification problem, and compare its performance to the discrete max-sum algorithm, as well as a non-coordinated approach and the distributed stochastic algorithm (DSA). We show that our hybrid max-sum algorithm outperforms the non-coordinated algorithm, DSA and discrete max-sum by up to 40% in this problem domain. Furthermore, the improvements in outcome over discrete max-sum come without significant increases in running time nor communication cost.

1 INTRODUCTION

There has been much recent interest in decentralised coordination problems for multi-agent systems, where multiple physically distributed devices must communicate with each other in order to collaboratively achieve some objectives. For example, these problems include coordinating systems of low power, locally communicating devices, such as sensor networks for wide-area security surveillance [9] and environmental monitoring [12], and multi-agent rescue robotics [14]. Typically, these sorts of problems can be represented as distributed constraint optimisation problems (DCOPs), in which interactions between the agents' states is modelled by a constraint graph. A number of algorithms have been developed for solving general DCOPs. However, many of these algorithms, like OptAPO [8], ADOPT [10] and DPOP [13] are designed to find globally optimal solutions, but at a cost of either exponential computation or communication requirements. Other algorithms, such as the distributed stochastic algorithm (DSA) [4], are designed to operate on large scale applications, but can often converge to poor quality solutions, due to the simplicity of the communication between agents.

Recently, the max-sum algorithm has been proposed as a middle ground between these two approaches [3], with the intention being that it converges to good quality solutions whilst remaining fully decentralised and scalable. These qualities are highly desirable for the class of applications mentioned above, where robustness is important, there are large numbers of agents, and a low communication or computational overhead is required. However, despite its advantages, the max-sum algorithm (and indeed, all the aforementioned approaches) is limited by the fact that it requires agents to have discrete state spaces. There are many multi-agent applications where devices must coordinate actions which are best represented by continuous state variables. Examples include controlling the orientation of sensors during target tracking [4], controlling mobile sensor trajectories during exploration [5] and coordinating sense sleep cycles to maximise coverage [6]. If agent states are continuous, this continuous space must be discretised before the any of the aforementioned algorithms can be applied. Furthermore, through the course of operation of these algorithms, each agent conducts repeated searches over the state spaces of itself and its interacting neighbours. Thus, care must be taken to discretise the continuous space with sufficiently few discrete states for such searches to be computationally tractable, which in turn limits the expressiveness and efficiency of the algorithms themselves.

Accordingly, we identify a need for decentralised coordination algorithms that have scalable computational and communication costs, and can seek good quality solutions for problems with continuous control spaces which interact in complex ways. Some work has already been done in this direction, however it has relied on objective functions being piece-wise linear, and suffers from unfavourable increases in complexity as the number of agents grows [16]. Furthermore, there also exist decentralised non-linear optimisation methods capable of accurately finding local optima over multi-dimensional continuous state spaces. In particular, many complex resource allocation algorithms in the field of flow control can be seen as distributed multi-dimensional gradient based optimisation methods (see [7] and [15] for an overview). However, these methods are most useful for finding optima of convex global objective functions. They are not designed to navigate complex interactions between local constraints in order to find globally optimal solutions, and would be likely to converge to a sub-optimal local maximum if applied to a non-convex DCOP. Thus, these algorithms would not be suitable for the above mentioned applications.

Thus, against this background, in this paper we seek to extend the functionality of discrete max-sum to situations where the accuracy of control options is important (and thus a prohibitively high level of discretisation is required), there is uncertainty about what control range or level of discretisation is appropriate and the agents' utility functions can not be decomposed into piece-wise linear components.

In particular, to address this shortcoming, we propose the hybrid continuous max-sum (HCMS) algorithm, which combines the discrete max-sum algorithm with continuous non-linear optimisation methods. Informally, the intention is to improve on continuous optimisation methods by using the max-sum process to escape undesirable basins of attraction and improve on the max-sum algorithm by using continuous optimisation methods to evolve state space discretisations over time so as to make the initial choice less critical.

In more detail, we make the following contributions.

- For problems with acyclic factor graphs, we derive theoretical optimality results for our algorithm. In particular, we can show that,

[1] School of Electronics and Computer Science, Southampton University, UK email: tdv@soton.ac.uk

for suitable parameter choices, the HCMS algorithm outperforms the discrete max-sum algorithm operating over the same discretisation of the state space and, for sufficiently fine discretisations, the HCMS algorithm converges to a near optimal state.

- We empirically evaluate our HCMS approach over a multi-sensor target classification problem, and compare its performance to the discrete max-sum algorithm, as well as a non-coordinated approach and the distributed stochastic algorithm (DSA). In so doing, we show that HCMS can outperform the DSA and discrete max-sum by over 40%, with reference to the non-coordinated algorithm performance.
- We further show that the improvements in outcome the HCMS algorithm achieves over discrete max-sum come with neither significant increases in running time nor communication cost.

The rest of the paper is organised as follows. Section 2 contains a formal description of the hybrid continuous max-sum algorithm, and statement and proof of our theoretical results. The results of our empirical evaluation of the performance of our algorithm are in Section 3. We conclude in Section 4.

2 ALGORITHM DESCRIPTION

The hybrid continuous max-sum algorithm is intended to solve general coordination problems between multiple agents. More formally, we consider the case where there are N cooperative agents and M utility functions $U_1, U_2, \ldots U_M$, where each agent i has a continuous state variable x_i and a set of utility functions $\mathbf{U}_i$. We assume each utility function U_j has a unique agent i such that $U_j \in \mathbf{U}_i$, and U_j only depends on a subset of the set of agents which are in direct communication with i. We write $U_j = U_j(\mathbf{x}_j)$ where $\mathbf{x}_j$ is the appropriate set of variables. The factor graph representation of this problem is a bipartite graph with a vertex for each variable and each utility function, and an edge between variable x_i and function U_j if and only if $x_i \in \mathbf{x}_j$. We do not assume anything more about the nature of the individual utility functions, and it is not required that they are known to other agents. The DCOP we consider is to find a set of states $\mathbf{x}^*$ such that social welfare of the whole system (i.e. the sum of the individual agents' utilities) is maximised:

$$\mathbf{x}^* = \arg\max \sum_{i=1}^{M} U_i(\mathbf{x}_i).$$

To ensure low communication cost, and a fully decentralised solution, it should be expected that an agent will only directly communicate with agents whose states affect its own utility functions. Thus in most applications, the computation and communication cost each agent experiences would depend on its number of neighbours rather than the size of the network.

The HCMS algorithm operates by combining the discrete max-sum algorithm with non-linear optimisation techniques. Given a state space discretisation for each variable x_i, that is, a set of values $x_i(1), x_i(2), \ldots, x_i(k_i)$ taken from its state space, the discrete max-sum algorithm finds an approximately optimal set of states within this discretisation. The HCMS algorithm improves on this by adjusting this state space discretisation to seek better quality solutions. We now describe the operation of the HCMS algorithm in more detail, beginning with a description of the discrete max-sum algorithm.

2.1 Discrete Max-Sum

The max-sum algorithm proceeds by exchanging information between functions and variables along the edges of the factor graph.[2] Each variable x_i communicates to each utility function $U_j \in \mathbf{U}_j$ the function $q_{i\to j}(\cdot)$, where, for $z = 1, 2, \ldots, k_i$,

$$q_{i\to j}(z) = \alpha_{ij} + \sum_{k\in\mathcal{M}_i\setminus j} r_{k\to i}(z), \qquad (1)$$

where $\mathcal{M}_i$ is the set of function indexes, indicating which functions depend on x_i and the normalising constants α_{ij} are chosen so that the sum of $q_{i\to j}(\cdot)$ over its domain is zero.

The functions $r_{i\to j}(\cdot)$ are communicated from utility functions U_i to state variables $x_j \in \mathbf{x}_i$, where for $z = 1, 2, \ldots, k_j$

$$r_{i\to j}(z) = \max_{\mathbf{y}: y_j = z} \Big(U_i\big(\{x_m(y_m)\}_{m\in\mathcal{N}_i}\big) + \sum_{k\in\mathcal{N}_i\setminus j} q_{k\to i}(y_k) \Big), \qquad (2)$$

where $\mathcal{N}_j$ is the set of agents whose states are in $\mathbf{x}_j$.

After a fixed number of iterations, the resulting solution is given by each variable x_i taking the state $x_i(z_i)$ where z_i maximises $\sum_{j\in\mathcal{M}_i} r_{i\to j}(\cdot)$. The motivation here is that this sum is an approximation to $h_i(\cdot)$, which is the *marginal function* of x_i, where for any state y, $h_i(y)$ is equal to the maximum value the global objective function can attain if $x_i = y$.

2.2 Hybrid Continuous Max-Sum

The HCMS algorithm involves implementing the same message passing as described above for the discrete max-sum, using the current discretisations of the variable state spaces. There is also additional information communicated between variables and functions. Firstly, each variable x_i must communicate to all $j \in \mathcal{M}_i$ the values of its state space discretisation. Secondly, each utility function U_i communicates to each state variable x_j for $j \in \mathcal{N}_i$ either $f^1_{i\to j}(\cdot)$, or both $f^1_{i\to j}(\cdot)$ and $f^2_{i\to j}(\cdot)$, where for $n = 1, 2$ and $z = 1, 2, \ldots, k_j$, $f^n_{i\to j}(z)$ is given by:

$$\frac{d^n U_i}{dx_j^n}\bigg|\arg\max_{\mathbf{y}: y_j = z} \Big(U_i\big(\{x_m(y_m)\}_{m\in\mathcal{N}_i}\big) + \sum_{k\in\mathcal{N}_i\setminus j} q_{k\to i}(y_k) \Big). \qquad (3)$$

As described above, the key difference between the HCMS and the discrete max-sum algorithm, is that each variable x_i evolves its state space discretisation, $x_i(1), x_i(2), \ldots, x_i(k_i)$ in order to find better quality solutions. To do so, the variable employs continuous non-linear optimisation techniques, which evolve as if maximising an objective function which takes the value $\sum_{j: x_i\in\mathbf{x}_j} r_{j\to i}(z)$ with n^{th} gradient $\sum_{j: x_i\in\mathbf{x}_j} f^n_{j\to i}(z)$ at each point $x_i(z)$ for $z = 1, 2, \ldots, k_i$.

The motivation for this is that, as a result of the discrete max-sum message passing process, for each variable x_i, for all $z = 1 \ldots, k_i$, the received values of $\sum_{j: x_i\in\mathbf{x}_j} r_{j\to i}(z)$ can be used as an approximation to $h_i(x_i(z))$. Furthermore, $\sum_{j: x_i\in\mathbf{x}_j} f^n_{j\to i}(z)$, can be used as an approximation to the value of $d^n h_i / dx_i^n$ evaluated at $x_i(z)$.

2.3 Continuous Non-linear Optimisation

Since each variable is attempting to optimise its marginal function using a series of approximations, it is important that robust optimisation methods are chosen. We choose gradient methods, which are

[2] Information is exchanged between functions and variables belonging to different agents by passing messages through the communication network.

robust to errors and can be implemented in a highly scalable, asynchronous, decentralised fashion using only local information. Indeed, the congestion control mechanism in use on the current Internet may be seen as such an implementation [15]. This leads to the intuition that the different state updates of the variables will not interact in unpredictable or harmful ways.

Accordingly, after each iteration of the HCMS message passing process, every state variable x_i updates its states space discretisation, by adding $\Delta x_i(z)$ to $x_i(z)$ for each $z = 1, \ldots, k_i$, where

$$\Delta x_i(z) = \kappa_i(z) \sum_{j: x_i \in \mathbf{x}_j} f^1_{j \to i}(z).$$

This determines the HCMS algorithm up to a choice of scaling factor $\kappa_i(z)$. In this paper, we consider two schemes for setting this parameter. Firstly, we consider a straightforward gradient method, which has a fixed constant $\kappa_i(z) = \kappa_i$. This is the simplest way to choose a stepsize, and the results from experimenting with this method for different values of κ_i should give intuition as to how sensitive the HCMS algorithm is to stepsize choice. Secondly, we attempt to improve on this simple scheme by making a choice of stepsize based on the Newton method, where a fixed constant κ_i is given so that

$$\kappa_i(z) = \kappa_i \Big(\sum_{j: x_i \in \mathbf{x}_j} f^2_{j \to i}(z) \Big)^{-1},$$

unless this value is negative or above κ_i, in which case we set $\kappa_i(z) = \kappa_i$. This bounding of $\kappa_i(z)$ deviates from normal Newton method behaviour, however it is necessary to prevent the algorithm from converging to minima, or behaving unpredictably around points of inflection.

The choice of these parameters must be, to some extent, fitted to the problem in question. If the values of $\kappa_i(z)$ are too small, then the algorithm will evolve slowly, and may not reach high quality solutions in the specified number of iterations. If the values of $\kappa_i(z)$ are too large, then the algorithm may be limited in how close it can come to converging on a high quality solution, due to continually overshooting the optimal point. We examine how the performance of the fixed stepsize gradient and Newton based methods depend on the parameter κ_i in Section 3.2. We find that there is a wide range of choices which yield good results.

As a rough rule of thumb, we would suggest taking κ_i to be at most inversely proportional to an approximate upper bound of

$$\sum_{j: x_i \in \mathbf{x}_j} \Big| \frac{d^2 U_j}{d^2 x_i} \Big|.$$

This is because, when using a gradient method to converge to maximise an objective function f, if the stepsize parameter is always chosen to be less than $2/K$, where K is an upper bound on f'', then each iteration leads to an improved solution. For our problem domain (see Section 3), we found that 99% of evaluated values of $|d^2U_j/d^2x_i|$ were bounded by 2.7. From the number of agents in our experiments we get an order of magnitude for κ_i to be around 0.1 or 0.01. This is born out by our empirical results, where κ_i larger than 0.1 begins to yield poorer results, and $\kappa_i = 0.01$ gives the best results over all.

2.4 Message Passing

There is one last aspect of HCMS algorithm operation which remains to be described. In this subsection we discuss the timing of the message passing. If the factor graph of the problem is a tree, then each variable and utility function can simply send a message every time it has received all messages required to calculate the contents according to Equation (1), (2) or (3). Those messages that do not require any information to calculate could simply be sent at regular intervals. As noted in [2], for these problems under the discrete max-sum algorithm, the messages $r_{i \to j}(z)$ and $q_{j \to i}(z)$ represent the maximum aggregate utility possible over the respective halves of the graph formed by removing the i to j link, if variable x_j is in state $x_j(z)$. This means that under these circumstances the discrete max-sum algorithm quickly converges to the global optimal solution. In the next section we will demonstrate that good theoretical results hold in this case for the HCMS algorithm also.

As with the discrete max-sum algorithm, in the more general setting, if there are cycles in the factor graph, then the above does not hold. It is not possible for all the variables and utility functions to wait for all the necessary information before sending a message, for otherwise, some messages would never be sent. Thus, if the factor graph can not be expected to be acyclic, then we suppose that each variable and utility function simply periodically sends all messages, using the most up to date information available. Any unknown functions are assumed to be zero for initial calculations, until the first set of messages is received.

2.5 Theoretical Results

We now show some theoretical results that apply to the HCMS algorithm over problems with acyclic factor graphs.

Proposition 1 *Suppose we have an HCMS algorithm solving a DCOP with an acyclic factor graph. If the stepsize is decreasing, and is always sufficiently small, then the maximum achievable utility given the set of possible states for each variable strictly increases over time.*

For every iteration, the message passing algorithm acts like the standard max-sum algorithm for the current variable state space discretisations. Thus, once all messages have been sent, by the results in [2] for the standard max-sum algorithm, for each agent i and $j \in \mathcal{M}_i$, for $z = 1, \ldots, k_i$,

$$\sum_{j=1}^{k_i} r_{j \to i}(z) = \max_{y: y_i = z} \sum_{i=1}^{M} U_i\big(\{x_m(y_m)\}_{m \in \mathcal{N}_j}\big).$$

For each variable x_i let z_i be defined as

$$z_i = \arg \max_{z=1,\ldots k_i} \sum_{j: x_i \in \mathbf{x}_j} r_{j \to i}(z).$$

By definition,

$$z = \arg\max_{y} \sum_{i=1}^{M} U_i\big(\{x_m(y_m)\}_{m \in \mathcal{N}_j}\big).$$

Furthermore, for each variable x_i, the value of

$$\sum_{j: x_i \in \mathbf{x}_j} f^1_{j \to i}(z_i)$$

is equal to the partial derivative of the objective function by x_i, evaluated at $\{x_k(z_k)\}_{k=1}^M$.

Hence, each update step moves $\{x_k(z_k)\}_{k=1}^M$ in the direction of the gradient of the objective function at that point. Provided the step sizes are sufficiently small, then utility at this point will strictly increase (see, for example [1] chapters 8, 9). Thus, after each iteration,

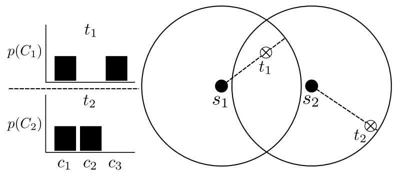

Figure 1. Sensor Configuration 1

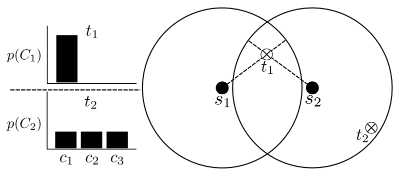

Figure 2. Sensor Configuration 2

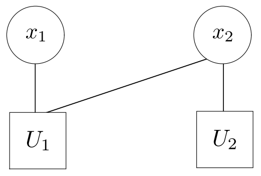

Figure 3. Factor graph for the sensor layout in Figures 1 and 2.

there is a combination of states which gives more utility than the previously maximum possible. □

As a corollary to this proposition, we can deduce that such an HCMS algorithm operating over an acyclic factor graph will converge to a state with higher utility than the discrete max-sum algorithm, provided both algorithms begin with the same state space discretisations. Furthermore, the utility of the solution provided by such an HCMS algorithm can be made to be arbitrarily close to optimal, if the initial state space discretisations are sufficiently fine. This is because with a sufficiently fine discretisation, then there will be at least one combination of initial possible states which is already sufficiently close to the optimal solution, and the progress of the algorithm can only improve upon this.

2.6 Communication and Computation Cost

The HCMS algorithm involves a slightly increased communication and computation overhead compared to the discrete max-sum algorithm. Specifically, the differences are as follows:

- Messages passed from function U_i to variable x_j include $f^1_{i\to j}(\cdot)$, and possibly $f^2_{i\to j}(\cdot)$ (Equation 3), instead of just $r_{i\to j}(\cdot)$ (Equation 2). This results in an increase in communication cost of at most a factor of three.
- Messages passed from variable x_i to function U_j include the new space discretisation $x_i(1),\dots,x_i(k_i)$, instead of just $q_{i\to j}(\cdot)$ (Equation 1). This results in an increase in communication cost of a factor of two.
- In terms of additional computation overhead, for each $z = 1\dots k_i$, $f^1_{i\to j}(z)$, and $f^2_{i\to j}(z)$ may be calculated using three evaluations of U_i (if there is no fast closed form expression for these derivatives). However these extra function evaluations do not represent a significant computational cost compared to the optimisation used to calculate the $r_{i\to j}(\cdot)$ functions, in which U_i is evaluated for the entire discrete state space.

So, the increase in communication and computation costs in operating the HCMS algorithm compared to the standard discrete max-sum algorithm is at most a factor of three and does not depend on the number of agents. Thus, the HCMS algorithm would seem to have the same desirable scalability properties as the discrete max-sum.

However, it is worth noting that the above result applies when comparing the two algorithms operating for the same number of iterations. From Proposition 1, we might expect that it is beneficial to operate the HCMS algorithm for more iterations than the discrete max-sum algorithm. In Subsection 3.2, we empirically explore how the performance of the HCMS approach is affected by how many iterations are run, and compare this to the behaviour of the discrete max-sum algorithm.

3 EMPIRICAL EVALUATION

Proposition 1 is in accordance with theoretical results available for the discrete max-sum algorithm. For DCOPs which do not have an acyclic factor graph representation, as with the discrete max-sum algorithm, there are no theoretical guarantees on the performance of the hybrid continuous max-sum algorithm. Thus, to further evaluate our approach, we must turn to empirical data. In this section we evaluate the HCMS algorithm through a series of simulated experiments in a target classification domain. This domain is particularly suitable as a testbed for decentralised coordination algorithms because of the presence of devices with limited communication and computation capabilities, and because the need for robustness and reliability excludes the possibility of using centralised algorithms. Moreover, this domain features an inherently continuous optimisation problem, which is of specific interest for benchmarking the HCMS algorithm against various discrete ones. Thus, we use this domain as an illustrative example, but it should be noted that the HCMS algorithm is more broadly applicable. We begin with a more detailed description of the specific problem scenario under consideration.

3.1 Problem Domain

We consider a network of wireless sensing devices $S = \{s_1,\dots,s_n\}$ that are tasked with classifying targets $T = \{t_1,\dots t_m\}$. Targets are assumed to be stationary, and can be one of $C = \{c_1, c_2, \dots\}$ classes. Sensors are able to take (imprecise) measurements of targets within a fixed sensing range, and are able to rotate to change their viewing direction. When pointing directly towards a target, the probability of a sensor correctly classifying it is maximised, but when rotated away from a target, the sensor acquires less information about the target, and this probability is reduced.

More formally, given a target t whose (unknown) class is modelled with random variable C_t (with domain C), a sensor s obtains a measurement, denoted by random variable M_s (also with domain C), based on its type and viewing direction. For each sensor/target pair, the probability of classifying a target as M_s, given its actual class C_t is given by $p(M_s|C_t,\theta)$, where θ is the angle between the sensor's viewing direction and target t, and ranges between 0 and π. For $\theta = \pi$ (i.e. the sensor looks away from the target) $p(M|C_t,\theta)$ is a uniform probability distribution over C_t, such that no information about the target's class is gained. The following equation has the desired properties:

$$p(M_s|C_t,\theta) = (1 - f(\theta))\, p_s(M_s|C_t) + f(\theta)\frac{1}{|C|} \quad (4)$$

Here, $p_s(M_s|C_t)$ is sensor s's optimal sensing signature, which applies when $\theta = 0$, and $f(\theta)$ is some function of θ with $f(0) = 0$ and $f(\pi) = 1$, such that when $\theta = \pi$, $p(M_s|C_t,\theta)$ is a uniform distribution.

Figures 1 and 2 illustrate this problem domain with two example scenarios. In both scenarios, there are two targets of class c_1 and two sensors. Sensor s_1 is capable of classifying targets of class c_2, but is unable to distinguish between classes c_1 and c_3. Similarly, sensor s_2 detects targets of class c_3, but can not distinguish between c_1 and c_2. In Figure 1, sensor s_1 is directed towards t_1, and s_2 towards t_2. Given this configuration, the posterior probability distribution over the class of t_1 and t_2 is shown on the left in Figure 1. If, however, the sensors are configured as in Figure 2, no information is gained about t_2's class, but t_1's class is correctly determined.

Now, given this, the goal of the sensor network is to minimise the remaining uncertainty in the classification of the targets after the having taken measurements. This is equal to the conditional entropy $H(C_1, \ldots, C_m | M_1, \ldots, M_n)$ of the target's classes given that the measurements of all sensors are known. Since the classes of any two targets (t, t') are assumed to be independent, $H(C_t, C_{t'}|M) = H(C_t|M) + H(C_{t'}|M)$, and the problem is reduced to minimising a sum of conditional entropies of the classification of individual targets. For an individual target t and a set of sensors S that are in range, the conditional entropy of C_t given M_S is given by:

$$\begin{aligned} H(C_t|M_S) &= \sum_{\mathbf{m} \in C^{|S|}} H(C_t|M_S = \mathbf{m}) \\ &= \sum_{\mathbf{m} \in C^{|S|}, c \in C} p(\mathbf{m}, c) \log \frac{p(\mathbf{m}, c)}{p(m)} \\ &= \sum_{\mathbf{m} \in C^{|S|}, c \in C} p(\mathbf{m}|c) p(c) \log \alpha p(\mathbf{m}|c) p(c) \end{aligned} \tag{5}$$

where $C^{|S|}$ denotes the set of all possible measurements that sensors S can collectively make, α is a normalising constant, and $p(c)$ is a prior over the target, which we assume is a uniform distribution.

Since the viewing angle of a sensor is a continuous parameter, taking values from $[0, 2\pi]$, this problem is a DCOP with continuous state spaces. Given this, and the fact that the sensors' actions interact in complex ways, this domain is particularly suitable for benchmarking the HCMS algorithm against existing discrete ones.

In order to use our HCMS algorithm (as well as the discrete max-sum algorithm), we now show how to build a factor graph of this problem. Firstly, we assign a continuous variable x_j to sensor j representing its viewing direction, ranging from 0 to 2π. Secondly, for each *target* i, we define a function $U_i(\mathbf{x}_i)$ with parameters $x_j \in \mathbf{x}_i$ iff target i is in range of sensor j. Thus, U_i is a continuous function of the sensors' viewing directions and is equal to the conditional entropy $H(C_i|\{M_j : x_j \in \mathbf{x}_i\})$ given these viewing directions of sensors in range as in Equation 5. Thirdly and finally, to obtain a truly decentralised approach, we assign the responsibility of computing the outgoing messages for U_i to one of the sensors $i : x_i \in \mathbf{x}_j$ in range, while taking care that the computation load is balanced over these sensors. For the simple scenarios in Figures 1 and 2, the factor graph is shown in Figure 3.

3.2 Results

We benchmark our algorithm against five algorithms:[3]

Discrete Max-Sum This algorithm is run on the same factor graph as used by our approach. By benchmarking against discrete max-sum, we can determine the improvement of coordinating in continuous state spaces using HCMS.

Local Greedy This algorithm selects the angle that minimises entropy on targets within range, regardless of the angles of its neighbours. This algorithm shows the performance that can be achieved without coordination.

Distributed Stochastic Algorithm (DSA) [4] This is an iterative best-response algorithm: sensors are randomly activated and update their angle such that the entropy of targets within range is minimised, fixing the current angle of its neighbours. DSA is an alternative to discrete max-sum, but has more limited information propagation, and has been shown to be outperformed by max-sum in general settings [3].

Random For each sensor, this algorithm selects a viewing angle at random. The random algorithm is included to provide a lower bound on achievable performance.

Centralised Simulated Annealing This is a centralised continuous algorithm for computing a solution that is often optimal. We include this algorithm as an upper bound for achievable performance.

[3] Since the functions in this domain are not piece-wise linear, we were unable to benchmark against continuous max-sum for piece-wise linear functions [16].

For our experimental evaluation, the sensors' viewing domain is discretised into 5 angles for the algorithms with a discrete state space (i.e. discrete max-sum, DSA and Local Greedy). The HCMS algorithm starts with the same initial discretisation as the discrete max-sum algorithm. We considered problem instances in which the sensors are laid out in a square lattice formation, consisting of k^2 sensors, with $k \in [3, 8]$, and the range of each sensor is chosen as $\frac{1}{k}$ to ensure the sensors' ranges are overlapping (but not the extent that the coordination problem becomes so dense that coverage of all targets is trivially ensured). We then randomly generated 100 problem instances (i.e. target locations) for each lattice formation.

First, we tuned the scaling factor κ_i for the gradient and Newton method, as discussed in Section 2.3. The results are shown in Figures 4 and 5, where solution quality is expressed as a fraction of the solution computed by simulated annealing. These figures clearly show that the Newton method is much less sensitive to the chosen value of κ_i than the gradient method. However, the gradient method, if properly tuned, gives slightly better results.

Second, we took the best gradient ($\kappa_i = 10^{-1.5}$) and Newton ($\kappa_i = 10^{-1}$) variants of our HCMS algorithm and benchmarked them against the discrete algorithms. The results are shown in Figure 6, and indicate that, using the performance of the random algorithm as a point of reference, our hybrid max-sum algorithm outperforms the discrete coordination algorithms (DSA and discrete max-sum) by roughly 40%.[4] Moreover, and more importantly, the normalised solution quality shows that our algorithm performs comparably to the simulated annealing algorithm.

Finally, we evaluated the speed of convergence of the gradient and Newton variants of HCMS on an 8 by 8 lattice, as compared to the discrete max-sum algorithm. The results are shown in Figure 7. This shows that, while discrete max-sum converges more quickly than HCMS, the solution quality of HCMS variants grows much faster over time. Around 20 iterations, both HCMS variants achieve a solution quality that is 30% better than discrete max-sum. However, since the gradient method exchanges the first derivative, and the Newton method both the first and second derivative (see Section 2.3), this comes at a cost of a twofold and threefold increase in message size respectively. Importantly though, the number of messages remains unchanged.

4 CONCLUSIONS

In this paper we identified a need for a scalable decentralised coordination algorithm for continuous distributed constraint optimisation problems. Such an algorithm would have applications in scenarios where robustness, scalability or low computational or communication overhead is desired. For this setting, we proposed a hybrid approach, combining the max-sum algorithm with continuous non-linear optimisation methods. We showed that, for problems with

[4] In this particular problem domain, discrete max-sum outperformed DSA by an almost negligible amount.

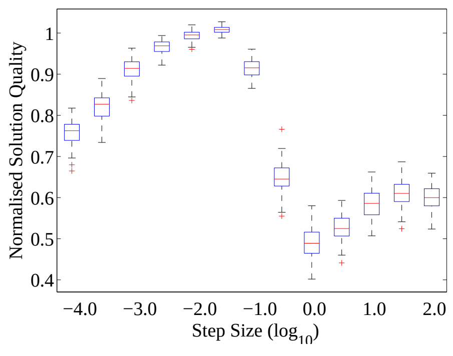

Figure 4. Effect of scaling factor (gradient method)

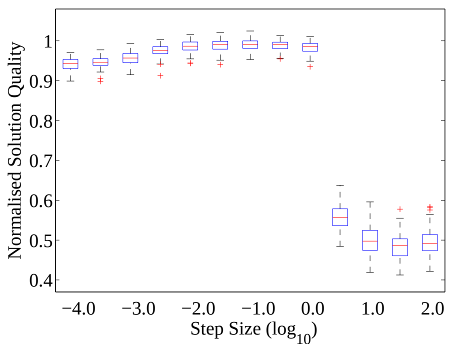

Figure 5. Effect of scaling factor (Newton method)

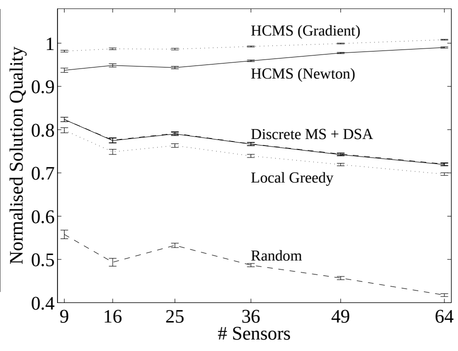

Figure 6. Average performance of evaluated algorithms

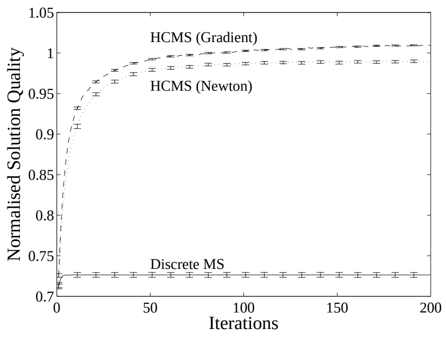

Figure 7. Speed of convergence of the gradient and Newton variants of HCMS compared to discrete max-sum.

acyclic factor graphs, for suitable parameter choices, our proposed algorithm converges to a state with utility close to the global optimum. We also empirically evaluated our approach over a target classification problem, and compared its performance to the discrete max-sum algorithm, as well as DSA, a non-coordinated algorithm and a centralised simulated annealing algorithm. The hybrid continuous max-sum algorithm was found to perform comparably with the centralised simulated annealing algorithm and can outperform DSA and discrete max-sum considerably. Furthermore, the improvements in outcome over discrete max-sum come without significant increases in running time nor communication cost.

There are several questions left open to tackle in future work. An interesting question is whether or not the HCMS algorithm can be improved by employing different non-linear optimisation techniques. For example, variables could use exploratory techniques, such as Bayesian Gaussian process regression based optimisation [11]. Under such a method, each variable would test out different state space discretisations, using the received information to update a model of its marginal function. Such an approach would avoid the possibility of converging to a local rather than global maximum, which could happen under our gradient method based HCMS if the initial state space discretisations were too coarse. A further question would be on whether algorithm performance could be improved by altering the nature of the communications between agents. Our algorithm provides agents with information on the effects of their choices on the global objective function by directly informing them of their effects on local utility functions. However, this is not the only way to express such information. Methods to compress transmissions could reduce communication overhead, while more expressive communication protocols could be developed to facilitate more complex optimisation techniques. A final, further reaching open problem, along a similar vein would be to attempt to form methods for automatically constructing optimal protocols, either through prior problem set training before deployment, or on-the-fly during operation.

REFERENCES

[1] M. Avriel, *Nonlinear Programming: Analysis and Methods*, Dover Publishing, 1999.

[2] A. Farinelli, A. Rogers, and N. R. Jennings, ‘Bounded approximate decentralised coordination using the max-sum algorithm’, in *Proc. of the 21st Int. Joint Conf. on AI*, (2009).

[3] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings, ‘Decentralised coordination of low-power embedded devices using the max-sum algorithm’, in *Proc. of the 7th Int. Conf. on Autonomous Agents and Multi-Agent Systems*, pp. 639–646, (May 2008).

[4] S. Fitzpatrick and L. Meetrens, *Distributed Sensor Networks: A multiagent perspective*, chapter Distributed Coordination through Anarchic Optimization, 257–293, Kluwer Academic, 2003.

[5] B. Grocholsky, J. Keller, V. Kumar, and G. Pappas, ‘Cooperative air and ground surveillance’, *IEEE Robotics and Automation Magazine*, **13**(3), 16–25, (2006).

[6] C. Hsin and M. Liu, ‘Network coverage using low duty-cycled sensors: Random and coordinated sleep algorithm’, in *Proc. of the 3rd Int. Symposium on Information Processing in Sensor Networks*, pp. 433–443, (2004).

[7] F. P. Kelly, ‘Fairness and stability of end-to-end congestion control’, *European Journal of Control*, **9**, 159–176, (2003).

[8] R. Mailler and V. Lesser, ‘Solving distributed constraint optimization problems using cooperative mediation’, in *Proc. of the 3rd Int. Joint Conf. on Autonomous Agents and MultiAgent Systems*, pp. 438–445, (2004).

[9] A. Makarenko and H. Durrant-Whyte, ‘Decentralized data fusion and control algorithms in active sensor networks’, in *Proc. of 7th Int. Conf. on Information Fusion*, pp. 479–486, (2004).

[10] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo, ‘ADOPT: Asynchronous distributed constraint optimization with quality guarantees’, *Artificial Intelligence Journal*, **161**, 149–180, (2005).

[11] M. A. Osborne, R. Garnett, and S. J. Roberts, ‘Gaussian processes for global optimization’, in *Proc. of the 3rd Int. Conference on Learning and Intelligent Optimization*, (2009).

[12] P. Padhy, R. K. Dash, K. Martinez, and N. R. Jennings, ‘A utility-based sensing and communication model for a glacial sensor network’, in *In Proc. of the 5th Int. Conf. on Autonomous Agents and Multi-Agent Systems*, pp. 1353–1360, (2006).

[13] A. Petcu and B. Faltings, ‘DPOP: A scalable method for multiagent constraint optimization’, in *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence*, pp. 266–271, (2005).

[14] P. Rybski, S. Stoeter, M. Gini, D. Hougen, and N. Papanikolopoulos, ‘Effects of limited bandwidth communications channels on the control of multiple robots’, in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 369–374, (2001).

[15] R. Srikant, *The Mathematics of Internet Congestion Control*, Birkhauser, 2004.

[16] R. Stranders, A. Farinelli, A. Rogers, and N. R. Jennings, ‘Decentralised coordination of continuously valued control parameters using the max-sum algorithm’, in *Proc. of the 8th Int. Joint Conf. on Autonomous Agents and MultiAgent Systems*, pp. 601–608, (2009).