Bounded Decentralised Coordination over Multiple Objectives

Francesco M. Delle Fave, Ruben Stranders, Alex Rogers & Nicholas R. Jennings
University Of Southampton
{fmdf08r,rs2,acr,nrj}@ecs.soton.ac.uk

ABSTRACT

We propose the bounded multi-objective max-sum algorithm (B-MOMS), the first decentralised coordination algorithm for multi-objective optimisation problems. B-MOMS extends the max-sum message-passing algorithm for decentralised coordination to compute bounded approximate solutions to multi-objective decentralised constraint optimisation problems (MO-DCOPs). Specifically, we prove the optimality of B-MOMS in acyclic constraint graphs, and derive problem dependent bounds on its approximation ratio when these graphs contain cycles. Furthermore, we empirically evaluate its performance on a multi-objective extension of the canonical graph colouring problem. In so doing, we demonstrate that, for the settings we consider, the approximation ratio never exceeds 2, and is typically less than 1.5 for less-constrained graphs. Moreover, the runtime required by B-MOMS on the problem instances we considered never exceeds 30 minutes, even for maximally constrained graphs with 100 agents. Thus, B-MOMS brings the problem of multi-objective optimisation well within the boundaries of the limited capabilities of embedded agents.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems

General Terms

Algorithms, Theory, Experimentation

Keywords

Coordination, Distributed Problem Solving

1. INTRODUCTION

Many real world problems involve the optimisation of multiple, possibly conflicting, objectives. Examples of bi-objective problems include the use of unmanned aerial vehicles (UAVs) for searching for survivors, while simultaneously establishing a wireless communication network between them [17], and controlling the motion of mobile robots to minimise travel distance while preventing collisions [14]. In both cases, independently maximising one objective results in detrimental performance in terms of the other. Now, due to the potential life-critical nature of these problems, centralised control of these UAVs or ground robots is not desirable, since it creates a single point of failure. Consequently, the use of multi-agent technology has been advocated to achieve reliable, robust and scalable control within these sensitive scenarios [7, 9]. Specifically, many decentralised constraint optimisation algorithms have been proposed to allow multiple agents to coordinate their actions in an attempt to achieve their collective goals [4, 5, 10, 13, 15].

Cite as: Bounded Decentralised Coordination over Multiple Objectives, F. M. Delle Fave, R. Stranders, A. Rogers and N.R. Jennings, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Yolum, Tumer, Stone and Sonenberg (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 371-378.

However, whilst both multi-objective constraint optimisation and decentralised coordination has generated considerable interest, no decentralised multi-objective optimisation algorithms have been proposed in previous work.

On the one hand, the field of decentralised optimisation has focused exclusively on problems involving a single objective, which are often represented as *decentralised constraint optimisation problems* (DCOPs). A number of algorithms have been proposed for solving general DCOPs, which can be divided in three broad classes. The first contains algorithms that are designed to find optimal solutions, such as ADOPT [13] and DPOP [15], but have a computational or communication complexity that is exponential in the number of agents. The second is composed of algorithms, such as the distributed stochastic algorithm (DSA) [5] or Maximum Gain Message [10], designed for large multi-agent systems, but which often converge to poor quality solutions. However, there exists a third class of algorithms usually referred to as Generalised Distributive Law (GDL) [1], which constitutes a compromise between the extremes represented by the first two classes, and can be used to compute good quality approximate solutions. In particular, one GDL algorithm, the max-sum algorithm, has been shown to generate solutions closer to the optimum than (for example) DSA, while being robust against message loss and exhibiting a scalable computational and communication cost [4]. An inherent shortcoming of the max-sum algorithm, however, is that it is not guaranteed to converge on cyclic constraint graphs, in which case it can perform arbitrarily poorly. This limits its applicability in safety-critical domains. The *bounded* max-sum algorithm, an extension to the standard max-sum algorithm, addresses this shortcoming, by pruning the constraint graph to a tree. In so doing, it is capable of providing performance guarantees on the computed solutions [3].

On the other hand, research on multi-objective constraint optimisation has yielded two extensions to well known single-objective algorithms, such as bucket elimination and branch and bound [16, 11], to solve general multi-objective optimisation problems. However, such approaches are centralised, and therefore lack the robustness required in the aforementioned scenarios. Moreover, since these algorithms exhibit a computation cost that is exponential in the number of agents, they are capable of solving only the smallest of problem instances. Another line of research has investigated extensions to standard DCOP algorithms for solving single-objective DCOPs with resource constraints [2], such as a capacity constraints in power distribution networks [8]. However, these resource constraints are not expressive enough to represent general multi-objective problems.

Thus, against this background, we identify a need for a decentralised coordination algorithm that has scalable computational and communication costs, and can provide good quality solutions for problems involving multiple objectives. To address this requirement, we propose the first decentralised coordination algorithm for multi-objective optimisation problems; the bounded multi-objective max-sum algorithm (B-MOMS). B-MOMS extends the bounded max-sum algorithm to solve multi-objective DCOPs (MO-DCOPs), a novel extension to the DCOP framework. These MO-DCOPS are encoded into a bipartite factor graph, in which vertices represent variables (owned by agents) and multi-objective functions, and edges represent the dependencies between the two. The B-MOMS algorithm then operates by exchanging messages between the variables and functions to compute approximate solutions to an MO-DCOP, whilst providing quality guarantees.

In more detail, this paper makes the following contributions to the state of the art:

- We propose the MO-DCOP problem, a general formalism for multi-objective coordination problems, which generalises the well known DCOP framework to the multi-objective setting.

- We develop B-MOMS, the first bounded decentralised algorithm for solving multi-objective optimisation problems. The operation of B-MOMS consists of three phases:

  - The first extends the bounded max-sum algorithm [3] to compute a cycle-free sub-graph of the multi-objective factor graph. To achieve this, we generalise the maximum spanning tree problem solved by the previous algorithm to handle the vector weights that we face in MO-DCOPs.
  - The second generalises the key mathematical operators required by max-sum to optimally solve the multi-objective problem encoded in this cycle-free factor graph.
  - Since there might be multiple Pareto optimal assignments to the cycle-free problem, the third and final phase enables agents to reach consensus on which global assignment to choose.

- We prove B-MOMS is optimal in acyclic factor graphs, and derive problem dependent approximation bounds in general graphs that do contain cycles.

- We present an extensive empirical evaluation of B-MOMS by benchmarking against a centralised optimal algorithm on a multi-objective extension of the graph colouring problem. We demonstrate that the approximation ratio never exceeds 2, even for extremely constrained problems (i.e. fully connected graphs), and is less than 1.5 for graphs where constraints exists between 20% of all pairs of agents for 14 variables. Moreover, the results indicate that the runtime required by B-MOMS never exceeds 30 minutes, even for maximally constrained graphs with 100 agents, positioning it well within the confines of many real-life applications.

The remainder of this paper is organised as follows. In Section 2 we discuss the theoretical background of B-MOMS. In Section 3, we formalise the MO-DCOP framework. In Section 4, we develop the algorithm, describing each of its three phases, and describe its theoretical properties in Section 5. Finally, we empirically evaluate B-MOMS in Section 6. Section 7 concludes.

2. PRELIMINARIES

In this section we review the theoretical background of our algorithm. In particular, in Section 2.1, we introduce fundamental concepts related to multi-objective optimisation and in Sections 2.2 and 2.3 we discuss the max-sum algorithm and the bounded max-sum algorithm respectively.

2.1 Multi-Objective Optimisation

A multi-objective optimisation problem (MOOP) is defined as the problem of simultaneously maximising k incommensurable *objective functions*, defined over a set $\mathbf{x} = \{x_1, \ldots, x_M\}$ of M discrete variables, where each x_j takes values in a discrete domain $D_{x_j} = \{d_j^1, \ldots, d_j^{|D_{x_j}|}\}$. Thus, a solution to a MOOP is an assignment $\mathbf{a}^* = \{(x_1 = d_1^{(1)}), \ldots, (x_M = d_M^{(M)})\}$ of values to variables, such that:

$$\mathbf{a}^* = \arg\max_{\mathbf{a} \in D_{\mathbf{x}}} \mathbf{U}(\mathbf{x}) = [U^1(\mathbf{x}), \ldots, U^k(\mathbf{x})]^T \tag{1}$$

where $D_{\mathbf{x}} = \times_{j=1}^{M} D_{x_j}$ is the domain of variables $\mathbf{x}$. Here, each objective function U^i can be defined over a subset $\mathbf{x}_i \subseteq \mathbf{x}$ of the variables of the problem. However, for ease of exposition, we assume each function is defined over the same set of variables.

Now, since the functions are incommensurable, it is possible that multiple assignments satisfy Equation 1. For example, consider three assignments $\mathbf{a}_1$, $\mathbf{a}_2$ and $\mathbf{a}_3$, such that $\mathbf{U}(\mathbf{a}_1) = [4, 5]$, $\mathbf{U}(\mathbf{a}_2) = [4, 3]$, and $\mathbf{U}(\mathbf{a}_3) = [6, 3]$. Clearly we have that $[4, 5]$ and $[6, 3]$ are larger than $[4, 3]$. However, $[4, 5]$ and $[6, 3]$ are not comparable. Thus, $\mathbf{a}_2$ does not satisfy Equation 1. Indeed, Equation 1 involves the optimisation of sets of *partially-ordered* assignments. Thus, to characterise the optimal solutions of a multi-objective optimisation problem, we use the well known concept of *Pareto optimality*:

Definition 1 (Pareto Optimality [12]). *An assignment* $\mathbf{a}^* \in D_{\mathbf{x}}$ *is* Pareto optimal *iff there does not exist another assignment* $\mathbf{a} \in D_{\mathbf{x}}$*, such that* $\mathbf{U}(\mathbf{a}) \geq \mathbf{U}(\mathbf{a}^*)$*, and* $U^i(\mathbf{a}) > U^i(\mathbf{a}^*)$ *for at least one objective function.*

The utility vector $\mathbf{U}(\mathbf{a}^*)$ corresponding to a Pareto optimal assignment $\mathbf{a}^*$ is referred to as a *non-dominated* vector. We define the notion of *non-dominance* as follows:

Definition 2 (Non-dominance). *A vector* $\mathbf{U}(\mathbf{a}^*) \in D_{\mathbf{x}}$ *is* non-dominated *iff there does not exist an assignment* $\mathbf{a} \in D_{\mathbf{x}}$*, such that* $\mathbf{U}(\mathbf{a}) \geq \mathbf{U}(\mathbf{a}^*)$*, with at least one* $U^i(\mathbf{a}) > U^i(\mathbf{a}^*)$*. Otherwise,* $\mathbf{U}(\mathbf{a}^*)$ *is said to be* dominated.

Thus, since a multi-objective problem involves the optimisation over partially ordered assignments, its solution is a *set* of Pareto optimal assignments. In the remainder of this paper, we will refer to the set of Pareto optimal assignments as **PO**.

2.2 The Max-Sum Algorithm

The max-sum algorithm is a decentralised message-passing optimisation algorithm belonging to the generalised distributive law (GDL) framework [1]. GDL algorithms exploit the factorisability of many optimisation problems, to solve them in an effective and efficient manner. Particularly, such problems are characterised as optimisation problems where the valuation algebra of the global constraint function is a *commutative semi-ring* (i.e. where the distributive law holds). All standard DCOP problems exhibit this characteristic [4].

Now, the most general characterisation of a DCOP involves M agents, each controlling a single discrete variable x_j, $j \in [1, M]$. Constraints between agents are encoded as functions $U_i(\mathbf{x}_i)$ $(i \in [1, N])$ over these variables. The scope $\mathbf{x}_i \subseteq \mathbf{x}$ of constraint function U_i contains the variables of the agents over which the constraint is defined. The aim of the coordination problem is then to choose variable assignments that maximise the sum of the constraint functions:

$$U(\mathbf{x}) = \sum_{i=1}^{N} U_i(\mathbf{x}_i) \tag{2}$$

In order to use the max-sum algorithm the problem is encoded as a special bipartite graph called a factor graph, in which vertices represent variables and functions, and edges the dependencies between them.

Max-sum defines two types of messages that are exchanged between variables and functions:

From variable x_j to function U_i:

$$q_{j\to i}(x_j) = \sum_{k\in M(j)\setminus i} r_{k\to j}(x_j) \tag{3}$$

where $M(j)$ represents the set of indices of the functions connected to variable x_j (i.e. the functions in which x_j occurs as an argument).

From function U_i to variable x_j:

$$r_{i\to j}(x_j) = \max_{\mathbf{x}_i\setminus x_j}\left(U_i(\mathbf{x}_i) + \sum_{k\in N(i)\setminus j} q_{k\to i}(x_k)\right) \tag{4}$$

where $N(i)$ represents the set of indices of the variables connected to function U_i.

Note that both $q_{j\to i}(x_j)$ and $r_{i\to j}(x_j)$ are scalar functions of variable x_j. When the factor graph is acyclic, these messages represent the maximum aggregate utility possible over the respective components of the graph formed by removing the dependency between U_i and x_j, for each value $d \in D_{x_j}$ in the domain of variable x_j. Thus, in this case, the marginal function of each variable is calculated by:

$$z_j(x_j) = \sum_{i\in M(j)} r_{i\to j}(x_j) = \arg\max_{\mathbf{x}\setminus x_j} \sum_{i=1}^{N} U_i(\mathbf{x}_i) \tag{5}$$

after which the optimal assignment of x_j is found by:

$$a_j = \arg\max_{x_j} z_j(x_j)$$

2.3 The Bounded Max-Sum Algorithm

When the factor graph is cyclic, the straightforward application of max-sum is not guaranteed to converge. However, by pruning edges such that an acyclic sub-graph of the factor graph is obtained, a bounded approximation can be derived [3]. More specifically, here, the goal is to compute a variable assignment $\tilde{\mathbf{a}}$ in the acyclic factor graph, such that:

$$V^* = \sum_{i=1}^{N} U_i(\mathbf{a}_i^*) \le \rho \sum_{i=1}^{N} U_i(\tilde{\mathbf{a}}_i) = \rho \tilde{V}$$

where $\mathbf{a}^*$ is the optimal solution of the cyclic factor graph, and ρ is the approximation ratio. To ensure this approximation ratio is as small as possible, the algorithm prunes those edges that have the least impact on solution quality. The impact of an edge between x_j and U_i is defined as its weight w_{ij}, which is computed by:

$$w_{ij} = \max_{\mathbf{x}_i\setminus x_j}\left[\max_{x_j} U_i(\mathbf{x}_i) - \min_{x_j} U_i(\mathbf{x}_i)\right] \tag{6}$$

Once all the weights are computed, the GHS algorithm [6] is used to compute a maximum spanning tree of the factor graph in a decentralised fashion. The newly obtained acyclic factor graph is then used in the second phase, in which the max-sum algorithm is used to compute $\tilde{\mathbf{a}}$, which is the optimal variable assignment to the modified problem:

$$\tilde{V}_m = \sum_i \min_{\mathbf{x}_i^c} U_i(\tilde{\mathbf{a}}_i)$$

where $\mathbf{x}_i^c$ is the set of variables that were eliminated from the scope of function U_i, corresponding to the edges that were pruned from the factor graph.

The approximation ratio ρ is now given by:

$$\rho = 1 + (\tilde{V}_m + W - \tilde{V})/\tilde{V} \tag{7}$$

where W is the sum of the weights of the pruned edges. Thus, an upper bound on the optimal solution can be computed as follows:

$$\tilde{V}_m + W \ge V^*$$

3. MO-DCOP FORMALISATION

We formalise the general problem by extending the DCOP framework to the setting of multiple objective functions. More formally, we consider the multi-objective DCOP (MO-DCOP) problem, which involves the simultaneous optimisation of k DCOPs, where each DCOP is defined as in Section 2.2. Specifically, we consider the problem of maximising the following vector of objective functions:

$$\mathbf{U}(\mathbf{x}) = \left[U^1(\mathbf{x}), \ldots, U^k(\mathbf{x})\right]^T \tag{8}$$

Since each component of $U(\mathbf{x})$ is a DCOP, this global objective functions is decomposable into N factors, each of which is a multi-objective constraint function $\mathbf{U}_i$:

$$\mathbf{U}(\mathbf{x}) = \sum_{i=1}^{M} \mathbf{U}_i(\mathbf{x}_i)$$

where, again, each $\mathbf{x}_i \subseteq \mathbf{x}$ is the subset of variables representing the scope of multi-objective constraint function $\mathbf{U}_i$, which are defined as:

$$\mathbf{U}_i(\mathbf{x}_i) = \left[U_i^1(\mathbf{x}_i), \ldots, U_i^k(\mathbf{x}_i)\right]^T$$

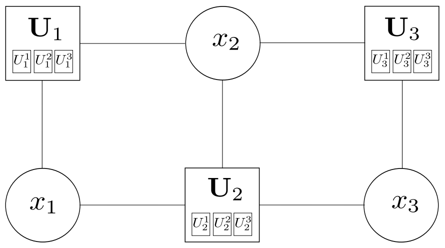


Figure 1: An example of a multi-objective factor graph, involving three variables, three objectives and three contraint functions.

Note that, for ease of exposition, we assume that all the local constraint functions U_i^k are defined over the same set of variables $\mathbf{x}_i$. However, this is not an essential requirement for the application of our algorithm.

Within this setting, the different solutions of a MO-DCOP are characterised using the solution concepts of Pareto optimality and non-dominance introduced in Section 2. Following these definitions, the solutions of a MO-DCOP are characterised as a set of Pareto optimal assignments PO corresponding to a set of non-dominated utilities vectors ND. Note that $\prod_{i=1}^{M} |D_i|$ is an upper bound of the number of possible solutions, which is equal to the size of the Cartesian product of the domains of all variables.

Finally, we encode the MO-DCOP as a multi-objective factor graph by representing each variable x_i of the MO-DCOP as a variable node, while each function node now represents a vector function $\mathbf{U}_i$. The following example illustrates such a multi-objective factor graph.

EXAMPLE 1. *Consider the factor graph in Figure 1, with three variables x_1, x_2, and x_3, which are controlled by agents $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$. There are three constraints between the agents, represented by functions $\mathbf{U}_1$, $\mathbf{U}_2$, and $\mathbf{U}_3$, each defined over three objectives U^1, U^2, and U^3.*

4. THE B-MOMS ALGORITHM

In this section we present the bounded multi-objective max-sum (B-MOMS) algorithm. B-MOMS extends the max-sum algorithm to compute solutions to MO-DCOPs. In detail, our algorithm proceeds in three phases:

- The *bounding* (B) phase, which extends the bounded max-sum algorithm discussed in Section 2.3, in order to provide quality guarantees. To achieve this, we generalise the maximum spanning tree algorithm to vector weights.
- The *max-sum* (MS) phase, during which the agents coordinate to find the Pareto optimal set of solutions to the cycle-free factor graph computed in the first phase. This requires a redefinition of the two key operations required by the max-sum algorithm (addition and marginal maximisation) for multiple objectives.
- The *value-propagation* (VP) phase, in which agents select a consistent variable assignment. This extends the standard VP phase [15] to the case where multiple non-commensurable alternatives exist.

In what follows, we discuss each phase in more detail.

4.1 The Bounding Phase

This phase builds upon the bounded max-sum algorithm described in Section 2.3 by extending the edge weights w_{ij} to the multi-objective vector weights. To this end, we first compute the impact of each variable x_j in the scope of each local multi-objective function $\mathbf{U}_i$ over all k objectives:

$$\mathbf{w}_{ij} = \left[w_{ij}^1, \ldots, w_{ij}^k\right]^T$$

Moreover, we define each scalar weight w_{ij}^o $(1 \leq o \leq k)$ as in the single-objective case:

$$w_{ij}^o = \max_{\mathbf{x}_i \setminus x_j} \left[\max_{x_j} U_i^o(\mathbf{x}_i) - \min_{x_j} U_i^o(\mathbf{x}_i)\right]$$

Since the problem of finding a maximum spanning tree is defined on instances with scalar edge weights, it is necessary to rank the vector weights. This procedure must ensure that the resulting ordering favours deletion of dominated vectors over non-dominated ones. One way of doing this is to assign a scalar weight w_{ij} to each vector $\mathbf{w}_{ij}$, which is proportional to the number of edge weights it dominates. More formally:

$$w_{ij} = -|\{\mathbf{w}_{mn} \mid \mathbf{w}_{mn} > \mathbf{w}_{ij}, (i,j) \neq (m,n)\}|$$

Thus, using this scalarisation, non-dominated weight vectors are assigned a value of 0, vectors dominated by a single elements are assigned a value of -1, and so on. With these scalar edge weights, the GHS algorithm can be used to compute a maximum spanning tree as discussed in Section 2.3.

4.2 The Max-Sum Phase

The second phase executes max-sum on the acyclic factor graph resulting from the previous phase. In order to apply max-sum to the multi-objective setting, however, the messages exchanged between functions and variables (Equations 3 and 4) need to be generalised to vectors of constraint functions.

Recall from Section 2.2 that, if the factor graph is acyclic, the messages r and q exchanged between U_i and x_j represent the maximum aggregate utility over the two components of the graph obtained by removing the dependency between U_i and x_j. The same holds in the multi-objective case. However, instead of $q_{j\to i}(x_j)$ and $r_{i\to j}(x_j)$ being scalar functions of x_j, these messages now map the domain of x_j to a *set* of utility vectors. To see why this is true, note that in the multi-objective domain, maximum utility is now defined in terms of the dominance relation from Definition 2. Since this relation induces a partial ordering, more than one such maximum might exist. To illustrate this, consider the following example:

EXAMPLE 2. *Suppose the following message $r_{i\to j}(x_j)$ is sent by function $\mathbf{U}_i$ to variable x_j with domain $D_j = \{Red, Green, Blue\}$:*

$$r_{i\to j}(x_j) = \begin{cases} \{[0,0,0]\} & \textit{if } x_j = Red \\ \{[0,1,0]\} & \textit{if } x_j = Green \\ \{[-1,2,-1],[2,1,-1]\} & \textit{if } x_j = Blue \end{cases}$$

This message conveys the fact that, if x_j is assigned the value Red, the maximum possible utility obtained within the sub-graph connected to $\mathbf{U}_i$ after removing the dependency on x_j is equal to $[0,0,0]$. *Similarly, if x_j = Blue, there are two incomparable maxima (since neither dominates the other), namely* $[-1,2,-1]$ *and* $[2,1,-1]$.

To compute these messages, the two key mathematical operators required by max-sum—the addition of two vector functions (Equations 3 and 4) and the marginal maximisation with respect to a single variable ($\max_{\mathbf{x}_i \setminus x_j}$ in Equation 4)— need to be defined for this domain. These operators were previously defined in the context of the multi-objective bucket elimination algorithm [16]. Here, however, we redefine these to compute the messages exchanged in the max-sum algorithm.

In more detail, to add two functions f and g defined over scope $\mathbf{x}$:

$$(f+g)(\mathbf{x}) = ND(\{\mathbf{v}+\mathbf{w} | \mathbf{v} \in f(\mathbf{x}); \mathbf{w} \in g(\mathbf{x})\})$$

where function $ND(A)$ filters out the dominated vectors from input set A. Using the + operator we can compute the sum of the messages r (which are univariate functions of x_j) in Equation 3, as well as the addition of multi-variate function $\mathbf{U}_j$ to the sum of univariate functions of *different* variables x_k in Equation 4.

The second operator, marginal maximisation ($\max_{\mathbf{x}_i \setminus x_j}$), takes as input a multi-variate vector function $f(\mathbf{x}_i)$ and outputs a function $f'(x_j)$:

$$f'(x_j = d) = ND\left(\bigcup_{\mathbf{d} \in D_{\mathbf{x}_i \setminus x_j}} f(\{\mathbf{x}_i \setminus x_j\} = \mathbf{d}, x_j = d) \right)$$

where $D_{\mathbf{x}_i \setminus x_j}$ is the Cartesian product of the domains of variables $\mathbf{x}_i \setminus x_j$.

At the end of the max-sum phase, each variable x_j computes its marginal function z_j (Equation 5) to obtain a set of Pareto optimal assignments A_j^*. Since there might be multiple optimal assignments, agents need to reach consensus on which global assignment to choose. Thus, in what follows, we define a value-propagation phase where the agents jointly choose a Pareto optimal solution among the ones computed by the max-sum phase.

4.3 The Value-Propagation Phase

The third and final phase, value-propagation, again operates on the cycle-free factor graph computed by the bounding approach. In this phase, variables and function nodes in this factor graph coordinate to select a consistent variable assignment.

Now, at the end of the MS phase, each variable computes the set A_j^* of marginal Pareto optimal variable assignments for x_j, which is obtained by maximising over the marginal function z_j:

$$A_j^* = \arg\max_{x_j} z_j(x_j)$$

If, for any variable x_j, $|A_j^*| > 1$, then there exists more than a single global optimal assignment in the acyclic factor graph: $|\widetilde{\mathbf{PO}}| > 1$. If this is the case, a variable can select an assignment $a_j^* \in A_j^*$ at random, or one that satisfies some (logical) condition C. For example, a_j^* might be chosen such that objective 1 is maximised, subject to objective 2 being at least 4, or more formally, $a_j^* \in A_j^*$ and $\max z_j(a_j^*)_1$, subject to $z_j(a_j^*)_2 \geq 4$. To select an assignment that satisfies this condition, the value-propagation phase proceeds by passing messages between the variables and functions in the acyclic factor graph, and is thus fully decentralised. First, the variable x_r with the lowest index is chosen as the *root* of the tree, and is responsible for initiating the value propagation phase by selecting a Pareto optimal assignment a_r^* that satisfies C. The variable then sends value-propagation messages ($x_r = a_r^*$) to all the function nodes to which the variable is connected.

The behaviour of all the other nodes in the graph will then depend on their type. More specifically:

Function nodes: Upon receiving a message ($x_j = a_j^*$) from variable x_j, multi-objective constraint function $\mathbf{U}_i$ computes the set $\mathbf{A}^*$ of local Pareto optimal assignments for variables $\mathbf{x}_i \setminus x_j$, conditioned on $x_j = a_j^*$:

$$\mathbf{A}^* = \left\{ \mathbf{a} \mid \mathbf{a} \in D_{\mathbf{x}_i \setminus x_j}, g(\mathbf{a}) \in ND\Big(\bigcup_{\mathbf{a} \in D_{\mathbf{x}_i \setminus x_j}} g(\mathbf{a}) \Big) \right\}$$

where $g(a)$ is defined as:

$$g(\mathbf{a}) = \mathbf{U}_i(\{\mathbf{x}_i \setminus x_j\} = \mathbf{a}, x_j = a_j) + \sum_{k \in N(i) \setminus j} q_{k \to i}(a_k)$$

After computing set $\mathbf{A}$, value propagation selects a Pareto optimal assignment $\mathbf{a}^* \in \mathbf{A}$ that satisfies condition C and sends the message ($x_k = a_k^*$) to every variable x_k ($k \neq j$), where a_k^* is the element of $\mathbf{a}^*$ corresponding to x_k.

Variable nodes: For each non-root variable x_j, once it receives a message ($x_j = a_j^*$) from a function $\mathbf{U}_i$, it sets its value to a_j^* and propagates the message ($x_j = a_j^*$) to all the function nodes $\mathbf{U}_k$, $k \neq i$.

Note that, during value propagation, a single message is sent across each link in the factor graph. Thus, the algorithm terminates once each non-root variable has received a value-propagation message.

5. THEORETICAL ANALYSIS

We now discuss the theoretical properties of the B-MOMS algorithm.

5.1 Optimality of the MS Phase

The first property concerns the performance of the max-sum phase of the B-MOMS. Specifically, we show that the following theorem holds:

THEOREM 1. *The max-sum phase (MS) computes the entire set of Pareto optimal solutions $\widetilde{\mathbf{PO}}$ of the acyclic factor graph that is obtained by pruning edges during the bounding (B) phase of the algorithm.*

PROOF. The proof consists of two steps. Firstly, the valuation algebra defined by the + and max operators discussed in Section 4.2, together with the co-domain of the global multi-objective constraint function $\mathbf{U}$ (Equation 8), is a commutative semi-ring [16]. Secondly, any GDL algorithm optimally solves problems whose valuation algebra is a commutative semi-ring, whenever the underlying constraint graph is acyclic [1]. Thus, since the second phase of B-MOMS is a GDL algorithm, the result holds. □

Now, while solutions $\widetilde{\mathbf{PO}}$ are Pareto optimal in the acyclic sub-graph, they are not necessarily (or likely) Pareto optimal in the original factor graph. However, using Theorem 1, we can derive bounds on the quality of these solutions in the original factor graph.

5.2 Bounded Approximation

To derive these bounds, we follow a procedure similar to that of the bounded max-sum algorithm (Section 2.3). First, we define vector $\mathbf{W} = [W^1, \ldots, W^k]$ as the sum of vector weights $\mathbf{w}_{ij}$ of the edges between U_j and x_i that were pruned in the bounding phase to obtain an acyclic graph (Section 4.1). Furthermore, to characterise the upper bound computed by B-MOMS, we first define the concept of *utopia point*:

Definition 3 (Utopia Point [12]). *The utopia point* $\mathbf{V}^*$ *of a multi-objective optimisation problem characterised by objective function* $\mathbf{U}(\mathbf{x}) = \left[U^1(\mathbf{x}), \ldots, U^k(\mathbf{x})\right]^T$ *is given by:*

$$\mathbf{V}^* = \left[\max_{\mathbf{x}} U^1(\mathbf{x}), \ldots, \max_{\mathbf{x}} U^k(\mathbf{x})\right]$$

Put differently, the utopia point is the vector of values resulting from optimising each k DCOPs independently. Clearly, given any Pareto optimal assignment $\mathbf{a}^* \in \mathbf{PO}$, $\mathbf{U}(\mathbf{a}^*) \leq \mathbf{V}^*$ holds for any MO-DCOP. Thus, the utopia point is an upper bound of the value of a Pareto optimal solution of a MO-DCOP. Given these concepts we can state the following theorem:

Theorem 2. *Given an arbitrary MO-DCOP, for any assignment* $\tilde{\mathbf{a}} \in \widetilde{\mathbf{PO}}$ *computed by B-MOMS, the following bound holds:*

$$\mathbf{U}(\tilde{\mathbf{a}}) + \mathbf{W} \geq \mathbf{V}^* \tag{9}$$

Proof. The theorem follows directly by the fact that we extend the bounded max-sum algorithm for single objective DCOPs to the case of multi-objective problems. In more detail, for each objective o $(1 \leq o \leq k)$ of an MO-DCOP, by using the approach defined in [3], it is easy to see that the following bound holds:

$$U^o(\tilde{\mathbf{a}}) + W^o \geq \max_{\mathbf{x}} U^o(\mathbf{x})$$

thus concluding the proof. □

Similarly, we define the problem dependent approximation ratio $\boldsymbol{\rho} = [\rho_1, \ldots, \rho_k]$ of the solutions computed by B-MOMS, where each ρ_i is given by Equation 7.

5.3 Complexity

Finally, we derive the computation and communication complexity of B-MOMS using properties inherited from the max-sum algorithm. Now, since B-MOMS exploits the factorisability of the problems it is solving, the scope of each constraint function $\mathbf{U}_i(\mathbf{x}_i)$ contains only the variables on which the constraint is defined. Therefore, computing message $r_{i\rightarrow j}$ from function $\mathbf{U}_i$ to variable x_j (Equation 4) requires $O(|D_{max}|^{|\mathbf{x}_i|})$ evaluations of function $\mathbf{U}_i$, where D_{max} is the largest domain among variables $\mathbf{x}$. Hence, the computation is exponential *only* in the number of variables in the scope of $\mathbf{U}_i$, not the total number of variables.

Furthermore, since in the worst case every variable assignment of $\mathbf{x}$ is Pareto optimal, after a certain (but finite) number have been exchanged, these messages contain $O(k \times |D_{max}|^{M+1})$ values: $|D_{max}|$ sets of vectors (one for each value in the variable's domain), each containing at most D_{max}^M non-dominated vectors of size k.

6. EMPIRICAL EVALUATION

Since B-MOMS is not an optimal algorithm, empirical evaluation is required to ascertain the quality of the solutions it computes. To this end, in this section we benchmark the performance of B-MOMS against an optimal algorithm. In the remainder of this section, we present a multi-objective extension to the canonical graph colouring problem used in our experiments, detail the experimental setup, and discuss the results.

6.1 Multi-Objective Graph Colouring

In order to evaluate the performance of our algorithm we consider a multi-objective extension of the graph-colouring problem, which is a well known benchmark problem in the DCOP literature [4]. In this multi-objective graph colouring problem, each agent $\mathcal{A}_j$ owns a single variable x_j, taking values in the domain $D_j = \{Red, Green, Blue\}$. Within this setting, the agents' goals is to maximise the following multi-objective function:

$$\mathbf{U}(\mathbf{x}) = \left[U^1(\mathbf{x}), U^2(\mathbf{x}), U^3(\mathbf{x})\right]^T \tag{10}$$

where U^1, U^2, U^3 are the sum of bi-variate constraint functions $U_i^1(x_j, x_k)$, $U_i^2(x_j, x_k)$, $U_i^3(x_j, x_k)$ that exist among the variables $\mathbf{x}$. These three types of constraint functions are defined as follows:

Chromatic Difference: This objective function represents the common graph colouring conflict function:

$$U_i^1(x_j, x_k) = \begin{cases} 0 & x_j \neq x_k \\ -1 & x_j = x_k \end{cases} \tag{11}$$

Chromatic Ordering: This objective function imposes an ordering among the colours: $Red = 1$, $Green = 2$, and $Blue = 3$. Specifically, given two variables x_j and x_k where $j < k$, the variable with the higher index should have a higher ranked colour:

$$U_i^2(x_j, x_k) = \begin{cases} 0 & \text{if } j < k \text{ and } x_j < x_k \\ -1 & \text{otherwise} \end{cases} \tag{12}$$

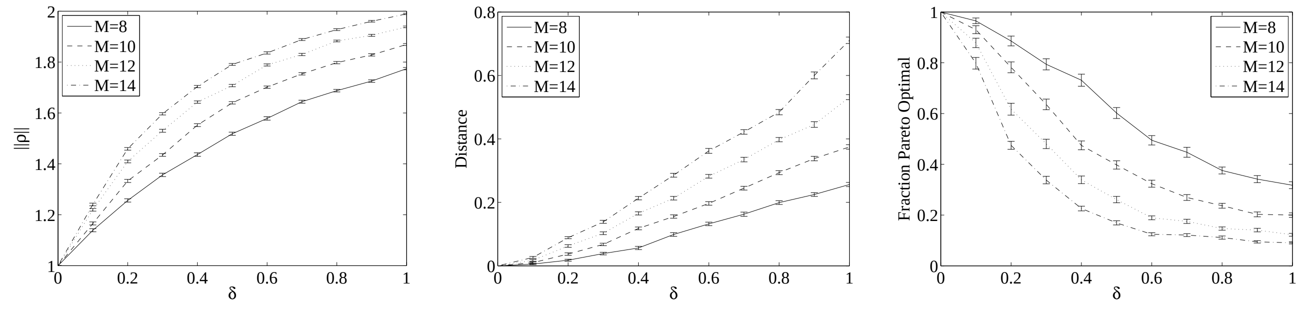
Chromatic Distance: This objective is similar to the chromatic ordering. However, it considers the distance between the colours of different variables. In more detail, given two variables x_j and x_k, the distance of the colours between the two variable should equal one:

$$U_i^3(x_j, x_k) = \begin{cases} 0 & \text{if } |x_j - x_k| = 1 \\ -1 & \text{otherwise} \end{cases} \tag{13}$$

Thus, given an arbitrary graph $G = (V, E)$, we construct a factor graph as follows. Each vertex is represented as a variable x. Furthermore, for each edge (v, v') the factor graph contains a three-objective constraint function $\mathbf{U}_i(x_j, x_k) = [U_i^1(x_j, x_k), U_i^2(x_j, x_k), U_i^3(x_j, x_k)]^T$ where x_j and x_k are the variables corresponding to vertices v and v'.

6.2 Experimental Setup

We analyse the performance of B-MOMS by executing it on several instances of the multi-objective graph colouring problem defined previously. Specifically, we randomly generate *connected* graphs $G = (V, E)$ with a varying number $M = |V|$ of vertices, and varying graph density $\delta \in$



(a) Approximation ratio $||\boldsymbol{\rho}||$ (b) Minimum distance of computed solutions to a Pareto optimal solution (c) Fraction n_{PO} of Pareto optimal solutions found

Figure 2: Empirical results for $M \leq 14$. Errorbars indicate the standard error of the mean.

$[0, 1]$. This latter parameter defines the constrainedness of the problem by controlling the number of edges; when $\delta = 0$ the number of edges $|E| = M - 1$, and the resulting graph G is a tree. Conversely, when $\delta = 1$, G is a complete graph, and $|E| = \frac{1}{2}M(M-1)$.

We generated problem instances with $M = \{8, 10, 12, 14, 20, 40, 60, 80, 100\}$ and $\delta = \{0.0, 0.1, \ldots, 1.0\}$. Moreover, for each combination of values for M and δ we ran B-MOMS 120 times to achieve statistical significance. We measure the performance of B-MOMS using the following four metrics:

1. The runtime (in milliseconds) required by B-MOMS to compute the set of solutions.

2. The average approximation ratio of the solutions $\widetilde{\mathbf{PO}}$ computed by B-MOMS. More specifically, we defined this as the norm of the approximation ratio vector $||\boldsymbol{\rho}||$ defined in Section 5.2.

3. The minimum Euclidean distance between solutions $\mathbf{a} \in \widetilde{\mathbf{PO}}$ computed by B-MOMS and an optimal solution $\mathbf{a}^* \in \mathbf{PO}$. More formally, we calculate this distance as follows:

$$d(\mathbf{a}) = \left\| \min_{\mathbf{a}^* \in PO} \Big[\mathbf{U}(\mathbf{a}) - \mathbf{U}(\mathbf{a}^*) \Big] \right\|$$

4. Finally, we measure the fraction Pareto optimal solutions found by B-MOMS:

$$n_{PO} = \frac{|\widetilde{\mathbf{PO}} \cap \mathbf{PO}|}{|\mathbf{PO}|}$$

Note that metrics 2 and 3 aggregate the objectives by using the notions of norm and Euclidean distance, which implies commensurability of the objectives. However, these metrics have been widely used in multi-objective literature [18]. Moreover, metrics 3 and 4 require the availability of the set of Pareto optimal solutions $\mathbf{PO}$. To compute these, we used a centralised brute-force optimal algorithm. This optimal algorithm exhaustively enumerates the Cartesian product of the domains of $\mathbf{x}$, and thus, has a computational complexity that is exponential in the number of variables. As a result, we do not report metrics 3 and 4 for $M > 14$, since we were unable to run a sufficient number of experiments to obtain statistically significant results. However, we do report metrics 1 and 2 for M up to 100.

6.3 Results and Discussion

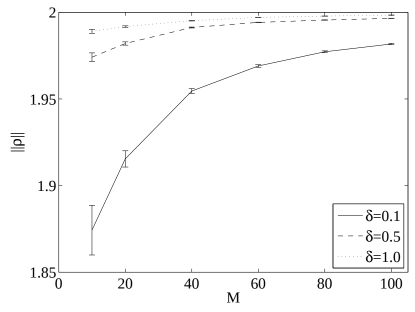
For $M \leq 14$, results are shown in Figure 2. First of all, all three plots confirm that the algorithm is optimal for acyclic graphs ($\delta = 0$), as proved by Theorem 1. Moreover, by increasing the number of constraints of the problem (i.e. by increasing δ), we can observe that the performance of B-MOMS degrades gracefully in terms of the approximation ratio, distance, as well as the fraction of optimal solutions found. Moreover, the approximation ratio never exceeds 2 (i.e. the value of the computed solution is greater than half of that of the optimal solution) even for extremely constrained problems, and is close to the value of 1.27 reported for the single-objective graph colouring problem by the single-objective bounded max-sum algorithm [3] when each variable is involved in three constraints (corresponding to $\delta = 0.3$ for $M = 14$). Most importantly, for relatively sparse graphs ($\delta \approx 0.2$), which are often found in real-life multi-agent applications, B-MOMS recovers roughly 50% of the optimal solutions for $M = 14$.

Figures 3(a) and 3(b) report the approximation ratio and the runtime for larger problem instances. Specifically, Figure 3(a) clearly shows that, even for large instances, the approximation ratio again never exceeds 2, demonstrating the effectiveness of the bounding approach. Furthermore, 3(b) gives strong empirical evidence of the practical applicability of the algorithm. Despite the exponential relation between the number of variables and the time required by B-MOMS[1], for $M = 100$ and a maximally constrained problem, this time does not exceed 30 minutes. Moreover, it is important to note that these experiments were run on a single processor, while the computational load in a multi-agent system is shared among multiple computational entities. This brings B-MOMS well within the realm of the limited computational capacities of embedded agents found in many real-life applications.

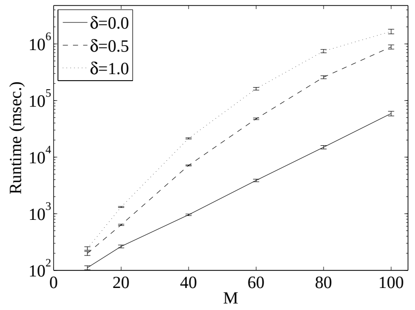
7. CONCLUSIONS

In this paper, we proposed the bounded multi-objective max-sum algorithm (B-MOMS), the first decentralised coordination algorithm for multi-objective optimisation problems. B-MOMS extends the bounded max-sum algorithm for decentralised coordination [3] to compute bounded approxi-

[1]This is partially due to our implementation of the value-propagation phase, which for the purpose of these experiments, recovers an exponential number of (approximately) Pareto optimal solutions, instead of just a single one.



(a) The approximation ratio $||\boldsymbol{\rho}||$ for varying number of vertices



(b) The runtime for varying number of vertices

Figure 3: Empirical results for $10 \leq M \leq 100$. Errorbars indicate the standard error of the mean.

mate solutions to multi-objective DCOPs (MO-DCOPs), a novel extension to the DCOP framework. It consists of three phases. The first phase extends the bounded max-sum algorithm developed for max-sum [3] to compute a cycle-free sub-graph of the multi-objective factor graph, which involves a generalisation of the maximum spanning tree problem to vector weights. The second phase generalises the key mathematical operators required by max-sum to optimally solve the multi-objective problem encoded in this cycle-free factor graph. Since there might be multiple Pareto optimal assignments to the cycle-free problem, the third and final phase enables agents to reach consensus on which global assignment to choose. We proved the optimality of B-MOMS in acyclic constraint graphs, and derived bounds on the approximation ratio. Furthermore, benchmarked B-MOMS against an optimal centralised algorithm on a multi-objective extension of the graph colouring problem. We demonstrate that the approximation ratio never exceeds 2, and is typically less than 1.5 for graphs in which dependencies exist between 20% of all pairs of agents. Moreover, the runtime required by B-MOMS never exceeds 30 minutes, even for maximally constrained graphs with 100 agents, positioning it well within the confines of real-life applications.

For future work, we intend to apply our approach on challenging real-world problems, such as the coordination of multiple mobile sensors, and power distribution networks [8]. Moreover, we would like to extend B-MOMS to the setting where there exist uncertainty about the constraint functions. This is a non-trivial extension, since it requires the exchange of vectors of probability distributions, instead of scalars, and a further generalisation of the two key mathematical operators required by max-sum.

8. ACKNOWLEDGMENTS

The work reported in this paper was funded as part of the ALADDIN (Autonomous Learning Agents for Decentralised Data and Information Systems) Project and is jointly funded by a BAE Systems and EPSRC (Engineering and Physical Research Council) strategic partnership (EP/C548051/1), and also by the Systems Engineering for Autonomous Systems (SEAS) Defence Technology Centre established by the UK Ministry of Defence.

9. REFERENCES

[1] S. M. Aji and R. J. McEliece. The Generalized Distributive Law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000.

[2] E. Bowring, M. Tambe, and M. Yokoo. Multiply-constrained distributed constraint optimization. In *Proc. of the 5th Int. Conf. on Autonomous Agents and Multiagent Systems*, pages 1413–1420, 2006.

[3] A. Farinelli, A. Rogers, and N.R. Jennings. Bounded approximate decentralised coordination using the max-sum algorithm. In *IJCAI-09 Workshop on Distributed Constraint Reasoning*, pages 46–59, 2009.

[4] 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*, pages 639–646, 2008.

[5] S. Fitzpatrick and L. Meertens. Distributed coordination through anarchic optimization. In Victor Lesser, Charles L. Ortiz, Jr., and Milind Tambe, editors, *Distributed Sensor Networks*, chapter 11, pages 257–295. Kluwer Academic Publishers, 2003.

[6] R.G. Gallager, P.A. Humblet, and P.M. Spira. A distributed algorithm for minimum-weight spanning trees. In *ACM Transactions on Programming Languages and Systems*, volume 5, pages 66–77, 1983.

[7] N. R. Jennings. An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4):35–41, 2001.

[8] A. Kumar, B. Faltings, and A. Petcu. Distributed constraint optimisation with structured resource constraints. In *Proc. of the 8th Int. Conf. on Autonomous Agents and Multi-Agents Systems*, pages 923–930, 2009.

[9] V. Lesser, C. L. Ortiz, and M. Tambe. *Distributed Sensor Networks, A Multiagent Perspective*. Kluwer Academic Publishers, 2003.

[10] R. J. Maheswaran, J. Pearce, and M. Tambe. A family of graphical-game-based algorithms for distributed constraint optimization problems. In *Coordination of Large-Scale Multiagent Systems*, pages 127–146. Springer-Verlag, Heidelberg Germany, 2005.

[11] R. Marinescu. Exploiting problem decomposition in multi-objective constraint optimization. In *Proc. of the 15th Int. Conf. on Principles and Practice of Constraint Programming*, pages 592–607. 2009.

[12] R.T. Marler and J.S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26:369–395, 2004.

[13] 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.

[14] A. Mouaddib, M. Boussard, and M. Bouzid. Towards a formal framework for multi-objective multi-agent planning. In *Proc. of the 6th Int. Conf. on Autonomous Agents and Multiagent Systems*, pages 801–808, 2007.

[15] A. Petcu and B. Faltings. DPOP: A scalable method for multiagent constraint optimization. *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence*, pages 266 – 271, 2005.

[16] E. Rollón. *Multi-Objective Optimization in Graphical Models.* PhD thesis, Universitat Politecnicá de Catalunya, 2008.

[17] A. Savikumar and C. Keng-Yan Tan. UAV swarm coordination using cooperative control for establishing a wireless communications backbone. In *Proc. of the 9th Int. Conf. on Autonomous Agents and Multiagent Systems*, pages 1157–1164, 2010.

[18] D. A. Van Veldhuizen and G.B. Lamont. Multi-objective evolutionary algorithm research: A history and analysis. Technical report, Dept. Elec Comp. Eng. Graduate School of Eng., 1998.