U-GDL: A Decentralised Algorithm on DCOPs with Uncertainty[*]

R. Stranders, F. M. Delle Fave, A. Rogers and N. R. Jennings

November 30, 2011

[*]This paper was originally presented at the AAMAS 2011 workshop on Optimisation on Multi-agent systems (Optmas 2011) on the 3 May 2011

U-GDL: A decentralised algorithm for DCOPs with uncertainty

R. Stranders, F. M. Delle Fave, A. Rogers & N. R. Jennings
University Of Southampton
{rs2,fmdf08r,acr,nrj}@ecs.soton.ac.uk

Abstract

In this paper, we introduce DCOPs with uncertainty (U-DCOPs), a novel generalisation of the canonical DCOP framework where the outcomes of local functions are represented by random variables, and the global objective is to maximise the expectation of an arbitrary utility function (that represents the agents' risk-profile) applied over the sum of these local functions. We then develop U-GDL, a novel decentralised algorithm derived from Generalised Distributive Law (GDL) that optimally solves U-DCOPs. A key property of U-GDL that we show is necessary for optimality is that it keeps track of multiple non-dominated alternatives, and only discards those that are dominated (i.e. local partial solutions that can never turn into an expected global maximum regardless of the realisation of the random variables). As a direct consequence, we show that applying a standard DCOP algorithm to U-DCOP can result in arbitrarily poor solutions. We empirically evaluate U-GDL to determine its computational overhead and bandwidth requirements compared to a standard DCOP algorithm.

Introduction

The challenge of coordinating systems composed of large numbers of autonomous agents has become a key focus of current artificial intelligence research. Example application domains include disaster management, where search and rescue robots are deployed to locate and retrieve casualties, and the Smart Grid, where the use of agents to control the flow of power within electricity networks is being explored. In many such settings, it is imperative that this coordination takes place in a decentralised fashion, so as to preclude the existence of a single point of failure. Thus, these coordination problems are often represented as distributed constraint optimisation problems (DCOPs) (Modi et al., 2005) in which the agents' collective goal is to maximise a global objective function that can be factorised into a sum of local functions that represent the local interactions between agents. They can then be efficiently solved by a wide range of algorithms (Modi et al., 2005; Fitzpatrick and Meertens, 2003; Maheswaran, Pearce, and Tambe, 2005), including those based on the Generalised Distributive Law (GDL) (Aji and McEliece, 2000), such as max-sum (Rogers et al., 2011) and DPOP (Petcu and Faltings, 2005).

However, in doing so, it is implicitly assumed that the value of the local functions that compose the global objective function are known with certainty and can be represented as scalar quantities. In reality, this is unlikely to be the case, since uncertainty is endemic within most interesting application domains. For example, within the rescue robots domain, the effectiveness of any coordinated decision will depend on a host of imprecisely known factors such as the exact location of the casualties, their condition, and local environmental conditions. Similarly, the power flow within an electricity grid depends on the aggregate behaviour of millions of consumers and can never be known with certainty. Thus, it is more realistic to consider that the value of these local functions are also uncertain, and hence, must be represented by probability distributions over possible values.

Now, at first sight, this does not appear to present a problem since it might be thought that we could simply represent the expected, or mean, value of these local functions as scalars, and proceed as before. However, in doing so, we would be implicitly assuming that the decision process is risk neutral. In contrast, there is an extensive literature on decision making under uncertainty that indicates that preferences over outcomes should be determined by a utility function which encodes the agents' risk profile (Levy, 2006) — the willingness of agents to expose themselves to uncertain outcomes in the prospect of a higher reward. This utility function is often non-linear, meaning that the decision process is either risk averse or risk seeking. For example, robots tasked with extracting casualties might act cautiously and prefer certain, but possibly less highly valued, outcomes, over uncertain ones. In contrast, a second team with the objective of mapping the area might exhibit a more risk seeking behaviour.

Existing DCOP algorithms fall short of dealing with uncertainty because they are incapable of representing uncertain outcomes (Rogers et al., 2011; Petcu and Faltings, 2005; Modi et al., 2005; Fitzpatrick and Meertens, 2003; Maheswaran, Pearce, and Tambe, 2005) or are only capable of maximising expected outcome without taking into account the specific risk profile of the agents (Léauté and Faltings, 2009; Jain et al., 2009; Taylor et al., 2010; Atlas and Decker, 2010). Indeed, as we demonstrate in this paper, applying existing DCOP algorithms to settings with uncertain outcomes and non-neutral risk profiles can cause them to perform arbitrarily poorly.

Thus, to address this shortcoming, in this paper we first introduce DCOPs with uncertainty (U-DCOPs), a novel generalisation of the canonical DCOP framework that represents the outcomes of the local functions as random variables (distributed according to a given probability density function), where the global objective is to maximise the expectation of an arbitary utility function (that represents the agents' risk-profile) applied over the sum of these local functions. We then develop U-GDL, a novel, decentralised, algorithm that optimally solves U-DCOPs derived from the Generalised Distributive Law (GDL) (Aji and McEliece, 2000) by extending the semantics of its $\max$ and $+$ operators to random variables rather than scalars. In doing so, the $+$ operator becomes the convolution operator, while the $\max$ operator now ranks random variables based on their expected utility. A key property of U-GDL that we show is necessary for optimality, is that it keeps track of multiple non-dominated alternatives, and only discards those that are dominated (i.e. local partial solutions that can never turn into an expected global maximum regardless of the realisation of the random variables). As a direct consequence, applying a standard DCOP algorithm to U-DCOP is shown to be provably sub-optimal. We empirically evaluate our algorithm and demonstrate that maintaining and communicating these multiple non-dominated alternatives results in an increase in computational overhead and bandwidth requirements of at most a factor of 5 compared to the standard GDL algorithm, which can perform arbitrarily poorly.

In more detail, we advance the state of the art as follows:

- We formally describe U-DCOPs, a novel formalism that subsumes canonical DCOPs and generalises them to the setting where the value of local functions are non-deterministic, and the agents' objective is to maximise an arbitrary utility function that ranks uncertain global outcomes.
- We present a formalism for deriving dominance conditions for discarding partial solutions that can never achieve global optimality. In the absence of a closed form condition, we introduce two types of conditions for dominance: sufficient and necessary. The former is guaranteed to preserve the optimal solutions, while possibly also maintaining sub-optimal ones. The latter might discard non-dominated solutions but is guaranteed to eliminate dominated ones. We demonstrate that applying a standard DCOP to solve a U-DCOP is equivalent to using a necessary rule.
- We develop U-GDL, the first decentralised algorithm to solve U-DCOPs. We demonstrate that, in conjunction with a sufficient (i.e. weaker) condition, U-GDL achieves optimality with an additional cost in terms of computation and communication overhead, while a necessary condition has no optimality guarantee.
- We empirically evaluate U-GDL and show that it achieves optimality at a cost that is at most a factor 5 greater in terms of computational overhead and bandwidth respectively compared to applying a standard DCOP algorithm (which is incapable of optimally solving U-DCOPs).

The remainder of this paper is organised as follows. First, we formally define DCOP with uncertainty. Then we discuss the GDL algorithm and present our U-GDL algorithm. Finally, we empirically evaluate U-GDL and conclude.

DCOPs with Uncertainty

A DCOP with uncertainty (U-DCOP) extends the (canonical) DCOP. We first give a formalisation of the canonical problem. A DCOP is a tuple $\langle \mathbf{x}, \mathcal{D}, \mathcal{F} \rangle$, where $\mathbf{x} = \{x_1, \dots, x_n\}$ is the set of variables, $\mathcal{D} = \{D_1, \dots, D_n\}$ is a set of discrete and finite domains from which variables with corresponding indices take their values, and $\mathcal{F} = \{f_1, \dots, f_m\}$ is a set of *value functions* defined over *local scope* $\mathbf{x}_i \subseteq \mathbf{x}$, which assigns a value to each assignment of variables in $\mathbf{x}_i$. Thus if $\mathbf{x}_i = \{x_{(1)}, \dots, x_{(r_i)}\}$, the arity of f_i is r_i and its signature is $f_i : D_{(1)} \times \dots \times D_{(r_i)} \to \mathbb{R}$. The objective of a DCOP is to find a variable assignment $\mathbf{x}^*$ that maximises the global value:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \sum_{i=1}^{m} f_i(\mathbf{x}_i) \tag{1}$$

A U-DCOP is a tuple $\langle \mathbf{x}, \mathcal{D}, \mathcal{F}, U \rangle$, where the value functions $\mathcal{F}$ are non-deterministic. As a result, instead of being scalar, the local value of an assignment of $\mathbf{x}_i$ returned by f_i is a random variable V_i. That is to say, $V_i \sim f_i(\mathbf{x}_i)$, or, equivalently, the probability distribution function (pdf) of V_i is given by $p(V_i) = f_i(V_i|\mathbf{x}_i)$. Thus, the global value $V = \sum_{i=1}^{m} V_i$ is now also a random variable, whose pdf is the convolution of the pdfs of the individual V_i. Thus, each global assignment to $\mathbf{x}$ results in a (possibly) different distribution of V.

Now, unlike the scalar values in Equation 1, the uncertain outcomes modelled by these distributions are not readily comparable. However, there exists a vast literature (Levy, 2006) about selecting between uncertain outcomes, and in many settings it is recognised that simply maximising expected value is not prudent. Rather, the actual shape of the distributions must be taken into account. For instance, in many application domains, as described in the introduction, it is common that decision makers are risk averse; they prefer certain outcomes over uncertain ones. Therefore, a *utility function* $U : \mathbb{R} \to \mathbb{R}$ is needed that ranks different outcomes based on the decision makers' preference. In general, these utility functions are non-decreasing, i.e. more value is preferred over less. For example, in disaster response, the utility increases with the number of civilians saved, but disaster responders might prefer a smaller (in expectation) but highly certain number of saved civilians over a better (in expectation) but highly uncertain (thus risky) outcome. In light of this, the objective of a U-DCOP is to maximise expected utility rather than expected value:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} E\left[U\left(\sum_{i=1}^{m} V_i\right)\right] \tag{2}$$

It is important to note that, since Equation 2 cannot be expressed as a sum of factors, a U-DCOP cannot be expressed as a DCOP, because in general:

$$\sum_{i=1}^{m} E[U(V_i)] \neq E\left[U\left(\sum_{i=1}^{m} V_i\right)\right] \tag{3}$$

x_1	x_2	f_1		f_2		f_1+f_2		SEU	EUS
		μ	σ^2	μ	σ^2	μ	σ^2		
0	0	9	8^2	10	15^2	19	17^2	-4	**2**
0	1	3	5^2	10	12^2	13	13^2	-4	0
1	0	15	7^2	5	24^2	20	25^2	-11	-5
1	1	2	4^2	2	3^2	4	5^2	$\mathbf{-3}$	-1

Table 1: The function payoffs of Example 1.

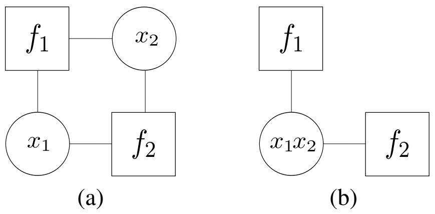


Figure 1: (a) The constraint network of Example 1 (b) the constraint network made acyclic by merging variables.

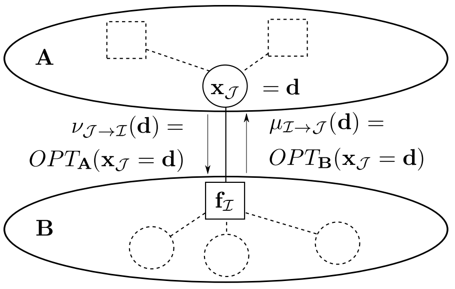
Hence, a U-DCOP cannot be solved by standard DCOP methods by simply defining $f_i(\mathbf{x}_i) = E[U(V_i)]$. The following example illustrates this important fact.

Example 1 *Consider a U-DCOP with* $\mathbf{x} = \langle\{x_1, x_2\}, \mathcal{D} = \{\{0,1\},\{0,1\}\}$*, and* $\mathcal{F} = \{f_1(x_1,x_2), f_2(x_1,x_2)\}$*. Furthermore let* $U(v)$ *be a function such that* $E[U(V)] = \mu_V - \sigma_V$*, where* μ_V *and* σ_V^2 *are the mean and standard deviation of* V*. The payoffs of functions* $\mathcal{F}$ *are shown in Table 1. The last two columns represent the left-hand and right-hand side of Equation 3, i.e. the sum of the expected utilities (SEU) and the expectation of the utility of the sum of the variables (EUS).*

As can be concluded from this table, the solution to this U-DCOP is $x_1 = x_2 = 0$ *(see last column). Note that simply maximising SEU, rather than EUS, results in an a suboptimal expected utility (*-3 *instead of* 2*). Thus, using a canonical DCOP algorithm results in sub-optimality.*

The Generalised Distributive Law

A canonical DCOP can be solved in a decentralised fashion by passing messages across the vertices of a *constraint network* representation of the DCOP. In such a network, value functions and variables are represented as vertices, and an edge exists between f_i and x_j iff $x_j \in \mathbf{x}_i$, i.e. variable x_j is a parameter of function f_i. Figure 1(a) shows the constraint network representation of the DCOP from Example 1. Many such message passing algorithms exist (Modi et al., 2005; Fitzpatrick and Meertens, 2003; Maheswaran, Pearce, and Tambe, 2005), of which the Generalised Distributive Law (GDL) algorithm is the most general (Aji and McEliece, 2000). GDL uses two operators, $\oplus$ for combining values and $\otimes$ for selecting values from a set, and exploits the distributive property of $\oplus$ over $\otimes$ to reduce the amount of computation required to solve a DCOP. For the standard DCOP, these operators are $\max$ and $+$ respectively (see Equation 1). Therefore, in the remainder of this paper, we use $\max$ and $+$ instead of $\oplus$ and $\otimes$.



A
$\mathbf{x}_{\mathcal{J}}$
$= \mathbf{d}$
$\nu_{\mathcal{J}\to\mathcal{I}}(\mathbf{d}) =$
$OPT_{\mathbf{A}}(\mathbf{x}_{\mathcal{J}} = \mathbf{d})$
$\mu_{\mathcal{I}\to\mathcal{J}}(\mathbf{d}) =$
$OPT_{\mathbf{B}}(\mathbf{x}_{\mathcal{J}} = \mathbf{d})$
$\mathbf{f}_{\mathcal{I}}$
B

Figure 2: The values exchanged between $\mathbf{x}_{\mathcal{J}}$ and $\mathbf{f}_{\mathcal{I}}$ in messages ν and μ equals the maximum summed function value in components $\mathbf{A}$ and $\mathbf{B}$

Now, it is well known that applying the GDL algorithm to acyclic constraint networks results in optimal solutions (Aji and McEliece, 2000). For solving DCOPs that are represented by cyclic networks there exists a variety of approaches (Rogers et al., 2011; Petcu and Faltings, 2005; Dechter, 2003). In this paper, we employ a simple method to transform a constraint network into an acyclic graph that is easy to decentralise. It is important to note that this choice does not impact the generality of our algorithm, and that U-GDL can be used in conjunction with any of the aforementioned (more sophisticated) methods.

In more detail, this method proceeds by iteratively merging two variables x_i, x_j until the network is acyclic. Merging variables involves substituting them by a new variable $x_{\{i,j\}}$, whose domain is the Cartesian product of D_i and D_j, that is connected to all functions f adjacent to x_i and x_j. Merging can occur recursively, i.e. merged variables can also be merged. Functions connected to the same pair of (merged) variables are added to preserve acyclicity. Figure 1b shows the constraint network obtained by merging variables in the cyclic network in Figure 1a. More formally, we denote a merged variable as $\mathbf{x}_{\mathcal{J}}$, where $\mathcal{J} = \{j_1, \ldots, j_k\}$ is the set of indices of variables contained in $\mathbf{x}_{\mathcal{J}}$. The domain of each merged variable $\mathbf{x}_{\mathcal{J}}$ is then defined as $D_{\mathcal{J}} = D_{j_1} \times \cdots \times D_{j_k}$. In a similar fashion we define a merged function as $\mathbf{f}_{\mathcal{I}}$, where $\mathcal{I} = \{i_1, \ldots, i_r\}$ represents the set of indices of functions that were merged into $\mathbf{f}_{\mathcal{I}}$. Each merged function $\mathbf{f}_{\mathcal{I}}$ is then defined as the sum of all such functions (i.e. $\mathbf{f}_{\mathcal{I}} = \sum_{i\in\mathcal{I}} f_i$). Finally, the scope merged function $\mathbf{f}_{\mathcal{I}}$ is the union of the scopes of functions $f_i : i \in \mathcal{I}$.

Now, in order to optimally solve a DCOP, these two types of nodes exchange messages based on the following equations (Aji and McEliece, 2000):

Message from variable $\mathbf{x}_{\mathcal{J}}$ to function $\mathbf{f}_{\mathcal{I}}$:

$$\mu_{\mathcal{J}\to\mathcal{I}}(\mathbf{x}_{\mathcal{J}\cap\mathcal{I}}) = \max_{\mathbf{x}_{\mathcal{I}\setminus\mathcal{J}}} \left[\sum_{\mathcal{M}\in\mathrm{adj}(\mathcal{J})\setminus\mathcal{I}} \nu_{\mathcal{M}\to\mathcal{J}}(\mathbf{x}_{\mathcal{M}\cap\mathcal{J}}) \right] \quad (4)$$

Message from function $\mathbf{f}_{\mathcal{I}}$ to variable $\mathbf{x}_{\mathcal{J}}$:

$$\nu_{\mathcal{I}\to\mathcal{J}}(\mathbf{x}_{\mathcal{I}\cap\mathcal{J}}) = \max_{\mathbf{x}_{\mathcal{J}\setminus\mathcal{I}}} \left[\mathbf{f}_{\mathcal{I}}(\mathbf{x}_{\mathcal{I}}) + \sum_{\mathcal{N}\in\mathrm{adj}(\mathcal{I})\setminus\mathcal{J}} \mu_{\mathcal{N}\to\mathcal{I}}(\mathbf{x}_{\mathcal{N}\cap\mathcal{I}}) \right] \quad (5)$$

Here, $\mathrm{adj}(n)$ returns the vertices adjacent to n. When the algorithm has converged, these messages are functions of a single merged variable $\mathbf{x}_{\mathcal{J}}$ that represent the maximum aggregate function values $OPT_{\mathbf{A}}$ and $OPT_{\mathbf{B}}$ possible over components $\mathbf{A}$ and $\mathbf{B}$ of the constraint network formed by removing the edge between $\mathbf{f}_{\mathcal{I}}$ and $\mathbf{x}_{\mathcal{J}}$ (see Figure 2). This is a direct consequence of the following key principle of GDL:

Principle 1 *Because* $\max$ *distributes over* $+$*, such that* $\max(x, y)+z = \max(x+z, y+z)$*, the maximum of the sum of the two components* $\mathbf{A}$ *and* $\mathbf{B}$ *is the global maximum. Formally, let* $\mathcal{F}_{\mathbf{A}} = \{f_{a_1}, \ldots, f_{a_m}\}$ *and* $\mathcal{F}_{\mathbf{B}} = \{f_{b_1}, \ldots, f_{b_m}\}$ *be the functions in components* $\mathbf{A}$ *and* $\mathbf{B}$ *respectively. Then:*

$$\max_{\mathbf{x}} \sum_{i=1}^{m} f_i(\mathbf{x}_i) = \max_{\mathbf{x}} \sum_{i=a_1}^{a_m} f_i(\mathbf{x}_i) + \max_{\mathbf{x}} \sum_{i=b_1}^{b_m} f_i(\mathbf{x}_i) \quad (6)$$

Therefore, when the GDL algorithm has converged[1], each variable can compute its *marginal function* $U_{\mathcal{J}}$ as follows:

$$U_{\mathcal{J}}(\mathbf{x}_{\mathcal{J}}) = \sum_{\mathcal{M}\in\mathrm{adj}(\mathcal{J})} \nu_{\mathcal{M}\to\mathcal{J}}(\mathbf{x}_{\mathcal{M}\cap\mathcal{J}}) = \max_{\mathbf{x}\setminus\mathbf{x}_{\mathcal{J}}} \sum_{i=1}^{m} f_i(\mathbf{x}_i) \quad (7)$$

For each state $d \in D_{\mathcal{J}}$, $U_{\mathcal{J}}(d)$ is equal to the maximum value that the global objective function can attain if $\mathbf{x}_{\mathcal{J}} = d$.

It is important to note that the GDL algorithm cannot readily be used to solve U-DCOPS (i.e. by defining the functions such that $f_i(\mathbf{x}_i) = E[U(V_i)]$), as Example 1 shows. Therefore, we need a new algorithm.

The U-GDL Algorithm

Having formulated the GDL algorithm, we now extend it to solve U-DCOPs. To do this, we need to define the $\max$ and $+$ operators for random variables, rather than scalar variables. Thus, the $+$ operator should now perform the addition of two random variables. For two random variables X and Y distributed according to pdfs f and g, the pdf of $X + Y$ is given by the *convolution* of f and g:

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x-a)g(a)da \quad (8)$$

Not all classes of pdfs are closed under convolution. However, the convolution of two Gaussians always results in another Gaussian, and thus in order to keep the exposition clear, we use them in the remainder of the paper. We note that approximate methods may be used (such as sampling) to represent arbitrary pdfs.

Defining the $\max$ operator is less trivial. This is because if we simply define $\max(X, Y) = X$ iff $E[U(X)] \geq E[U(Y)]$, Principle 1 no longer holds because $+$ does not distribute over $\max$, i.e. $E[U(X)]+E[U(Z)] > E[U(Y)]+E[U(Z)] \not\Leftrightarrow E[U(X+Z)] > E[U(Y+Z)]$, unless U is linear. Therefore, to solve U-DCOPs, the $\max$ operator should distribute over convolution, and select the maximal elements from a set of random variables based on their expected utility (Equation 2). More formally, given a binary dominance relation $\succeq$ that ranks random variables, and a set of random variables P, $X \in \max(P)$ iff $\forall Y \in P : X \succeq Y$. Now, relation $\succeq$ needs to be chosen to ensure Principle 1 holds.[2]

In order for these requirements to be satisfied, the $\succeq$ needs to exhibit the following property:

Property 1 *For any random variable* Z*:*

$$\begin{aligned} X \succeq Y &\Leftrightarrow X + Z \succeq Y + Z \\ &\Leftrightarrow E[U(X+Z)] \geq E[U(Y+Z)] \end{aligned} \quad (9)$$

It can be shown that it is impossible to determine whether $X \succeq Y$ without knowing the shape of the pdf of Z and $U(v)$, but specific conditions can be derived for classes of pdfs and utility functions. In general, however, it can be shown that $\succeq$ is a partial order, as the following example illustrates:

Example 2 *Let* $U(x)$ *be the utility function from Example 1. Then,* $X \succeq Y$ *iff* $\mu_X + \mu_Z - \sqrt{\sigma_X^2 + \sigma_Z^2} \geq \mu_Y + \mu_Z - \sqrt{\sigma_Y^2 + \sigma_Z^2}$ *for any* $\mu_Z, \sigma_Z \in \mathbb{R}$*. Algebraic manipulation shows that this inequality holds iff:*

$$\mu_X - \mu_Y \geq \max(0, \sigma_X - \sigma_Y) \quad (10)$$

To see why $\succeq$ *is a partial order, consider* $X \sim \mathcal{N}(0, 5^2)$ *and* $Y \sim \mathcal{N}(27, 35^2)$*. Clearly, the inequality in Equation 10 does not hold, so* $X \not\succeq Y$ *(or vice versa). To see why this is true, consider* $Z_1 \sim \mathcal{N}(0, 0)$*,* $E[U(X+Z_1)] = -5 > -8 = E[U(Y+Z_1)]$*, but for* $Z_2 \sim \mathcal{N}(0, 12^2)$*,* $E[U(X+Z_2)] = -13 < -10 = E[U(Y+Z_2)]$*. Thus, discarding* Y *because* $E[U(X)] > E[U(Y)]$ *can result in sub-optimality.*

This partial order implies that instead of containing a single pdf, the messages $\nu_{\mathcal{I}\to\mathcal{J}}(x_j)$ and $\mu_{\mathcal{J}\to\mathcal{I}}(x_j)$ (Equations 4 and 5) need to carry *sets of non-dominated pdfs*. Here, domination is formally defined as:

Definition 1 (Dominance under U**)** *A random variable* X *dominates* Y *(denoted as* $X \succ Y$*) iff for any pdf* $p(Z)$ *of random variable* Z*,* $E[U(X+Z)] \geq E[U(Y+Z)]$*, with strict inequality for at least one* $p(Z)$*.*

To understand dominance in terms of Figure 2, suppose Z is an optimal (but unknown) value in component $\mathbf{B}$, and suppose there are two alternative values X and Y in component $\mathbf{A}$. Then, Y can be safely discarded by variable $\mathbf{x}_{\mathcal{J}}$ if $X \succeq Y$. This is because $Y + Z$ can never lead to a global optimum of Equation 2, regardless of Z.

Unfortunately, there does not always exist a closed form expression for the integrals that need to be evaluated to determine dominance for a given pair of utility function and class of pdf. In such cases, recourse can be taken to a different logical condition, of which two types can be distinguished: sufficient and necessary ones.

Definition 2 (Sufficient Condition for Dominance) *A condition* C *that imposes relation* $\succ_C$ *is called* sufficient *if* $X \succ_C Y \Rightarrow X \succ Y$*.*

[1]Convergence requires a number of messages equal to twice the diameter of the resulting acyclic constraint network.

[2]Compare this to the $\max$ and $+$ operators on $\mathbb{R}$. Here, $\max$ distributes over $+$ because of the implicit $\geq$ relation used by $\max$ to select the maximal element from a set $P \subseteq \mathbb{R}$. However, if $\succeq$ were chosen such that $x \succeq y$ if the decimal part of x is greater than that of y, $\max$ no longer distributes over $+$.

A sufficient condition is guaranteed to keep all non-dominated random variables, but is not necessarily capable of filtering out all dominated random variables.[3] Therefore, by applying a sufficient condition it is guaranteed that the optimal (i.e. non-dominated) random variable is preserved, and consequently, global optimality is guaranteed. However, it might happen that too many pdfs are propagated in messages ν and μ, i.e. those that can never maximise Equation 2. An example of a sufficient condition for the utility function in Example 2 is C: $\mu_X \geq \mu_Y \wedge \sigma_X \leq \sigma_Y$ (with strict inequality in at least one clause). Clearly, this condition is weaker than Equation 10 because for $X \sim \mathcal{N}(0, 5^2)$ and $Y \sim \mathcal{N}(-2, 4^2)$, we have that $X \succ Y$ but not $X \succ_C Y$.

Definition 3 (Necessary Condition for Dominance) *A condition C that imposes relation $\succ_C$ is called* necessary *if $X \succ Y \Rightarrow X \succ_C Y$.*

A necessary condition is guaranteed to discard all dominated random variables, but might filter out non-dominated ones as well. Thus, using a necessary condition, it is possible to lose optimality. An example of a necessary condition is $\mu_X - \sigma_X > \mu_Y - \sigma_Y$, which imposes a total order. Using this condition is equivalent to applying a canonical DCOP method to maximise the left-hand side of Equation 3.

Definition 4 (Optimal Condition for Dominance under U) *A condition C that imposes relation $\succ_C$ is called* optimal *if it is both necessary and sufficient.*

Since we derived the condition in Equation 10 on the premise that $E[U(X+Z)] \geq E[U(Y+Z)]$, it is optimal by definition. An optimal condition propagates the smallest set of pdfs without discarding non-dominated random variables.

The key difference between a sufficient and a necessary rule is that the former is guaranteed to lead to optimal solutions, but also requires larger messages and more computation, while the latter does not guarantee optimality, but is computationally less expensive. We empirically demonstrate this in the next section.

After the U-GDL algorithm has converged, each merged variable computes the marginal function using Equation 7, which now contains the set of non-dominated pdfs for every variable state. Then, the optimal variable state can be obtained by computing the expected utility for each of these:

$$\mathbf{x}^*_{\mathcal{J}} = \arg\max_{\mathbf{d} \in D_{\mathcal{J}}} \left[\max_{V \in U_{\mathcal{J}}(\mathbf{d})} E[U(V)] \right]$$

After this, the final stage of the GDL algorithm, value propagation, can proceed as in Petcu and Faltings (2005).

Experiments

As discussed in the previous section, the behaviour of U-GDL varies depending on whether an optimal, sufficient or necessary condition is used. Moreover, while U-GDL is optimal when used with an optimal or sufficient condition, it not possible to theoretically determine the impact on the execution time of the algorithm, or the communication requirements. Thus, we need to resort to empirical evaluation. In addition, we compare the U-GDL algorithm with the standard GDL algorithm that maximises the left-hand side of Equation 3, which is equivalent to solving a U-DCOP with the necessary condition $X \succ_C Y \Leftrightarrow E[U(X)] > E[U(Y)]$. As Example 1 illustrates, this can result in sub-optimality. This allows us to analyse the performance of U-GDL, which explicitly takes uncertainty and the agents' risk profiles into account, against a canonical approach, that does not.

[3]For example $false$ is a sufficient condition.

Experimental Setup

To evaluate the performance of the U-GDL algorithm we apply it to randomly generated (connected) constraint networks, with a varying number of variables n with three states each. The number m of functions is controlled by parameter $\delta \in [0, 1]$. For $\delta = 1$ there exists a pairwise constraint function between all pairs of variables ($m = \binom{n}{2}$), while for $\delta = 0$, the constraint network is a tree (thus, $m = n - 1$).

For each pair of variable assignments, the value assigned by a function is a normally distributed random variable whose μ and σ^2 are drawn from the intervals $[-1, 1]$ and $[0, \sigma^2_{\max}]$ with uniform probability. We varied σ^2_{max} between 1 and 10 to determine the effect of increasing levels of uncertainty on the difference between the optimal solution and the solution found by a canonical DCOP method.

We run the U-GDL algorithm considering the utility function defined in Example 1 and three dominance conditions:

Optimal: $X \succeq Y \Leftrightarrow \mu_X - \mu_Y \geq \max(0, \sigma_X - \sigma_Y)$

Sufficient: $\mu_X \geq \mu_Y \wedge \sigma_X \leq \sigma_Y \Rightarrow X \succeq Y$

Necessary: $X \succeq Y \Rightarrow \mu_X - \sigma_X \geq \mu_Y - \sigma_Y$. This is equivalent to a the standard GDL algorithm (or indeed any canonical DCOP algorithm) that maximises the sum of expected utilities (left-hand side of Equation 3).

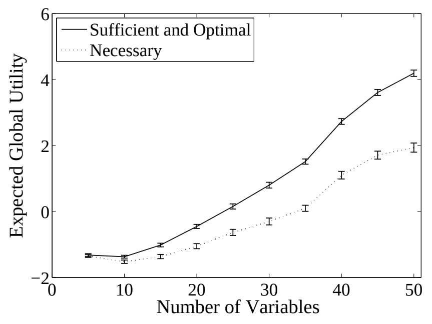
For each of the 200 randomly generated problem instances we measure the solution quality, the average message size, and the required computation time for each dominance condition. The discussion of dominance conditions in the previous section gives rise to the following hypothesis:

Hypothesis 1 *A sufficient condition for dominance results in an optimal solution, but requires larger messages and more computation than an optimal or necessary condition. A standard DCOP algorithm (i.e. a necessary condition) results in suboptimal solution quality.*

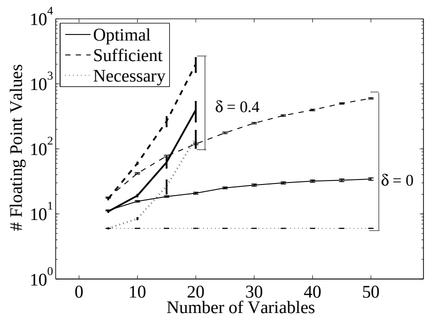
Results

The results are shown in Figure 3.[4] These results support Hypothesis 1: U-GDL in combination with a sufficient and an optimal condition result in optimality (Figure 3a), while the standard GDL algorithm that maximises the sum of expected utilities, does not. We found that the difference between the optimal and necessary conditions increases as

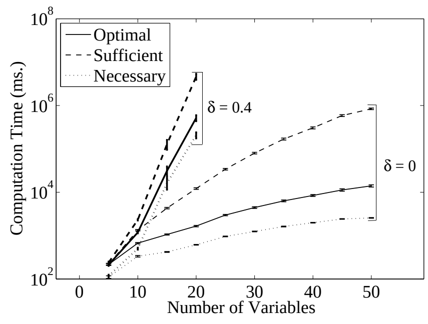
[4]We show results for $\delta = 0$ and $\delta = 0.4$ only, because for $\delta > 0.4$, the constraint graphs often collapsed into a single merged variable, leading to completely centralised problems. This problem can be mitigated by using more a sophisticated method for creating acyclic graphs (which is not our primary aim here). Furthermore, we found no significant difference in solution quality for different levels of δ, so only those for $\delta = 0$ are shown.



(a) Solution quality



(b) Message size



(c) Runtime

Figure 3: Empirical results. Errorbars indicate the 95% confidence intervals.

we increased $\sigma^2_{\max}$ (results not shown due to space restrictions), with a mean absolute difference of 0.97 (± 0.12) for $\sigma^2_{\max} = 1$ and 2.82 (± 0.32) for $\sigma^2_{\max} = 10$ (for $n = 50$). Thus, as expected, a standard DCOP algorithm performs worse as uncertainty increases. However, Figure 3b shows that this optimality comes at a cost of a roughly fivefold increase in message size for the optimal condition (at $n = 40$), and a hundredfold increase for the sufficient condition (at $n = 50$). This means that messages carry 5 or 100 times as many pdfs than the standard GDL algorithm, which only carries one pdf per domain element. As a result of keeping track of an increased number of non-dominated alternatives, the runtime of the optimal and sufficient conditions grows by a factor of 5 and 330 respectively (Figure 3c). However, at less than 14 seconds and 15 minutes shared among 50 agents respectively, the runtime of both the optimal and sufficient conditions remains well within the capabilities of even low-powered embedded agents. Consequently, we have demonstrated the viability of the optimal U-GDL algorithm as compared to the state of the art GDL algorithm, which can perform arbitrarily poorly.

Conclusions

We developed U-GDL, a novel algorithm for solving U-DCOPs, a generalisation of canonical DCOPs in which the agents' goal is to maximise the expectation of an arbitrary utility function. U-GDL extends GDL by redefining the $(\max, +)$ algebra to the setting where payoffs are random variables rather than scalars. Within this new setting, the former operator now filters partial solutions that can never achieve global optimality, using so-called dominance conditions. We showed that these conditions can be classified into three categories—optimal, sufficient and necessary. We demonstrate that an optimal condition has the lowest computation and communication overhead for achieving optimality, but does not always have a closed form expression. In such cases, a sufficient or necessary condition can be used. The former achieves optimality at the cost of a higher communication and computation overhead, while the second incurs lower costs but sacrifices optimality. Finally, we benchmarked U-GDL against a standard DCOP algorithm (which we showed is equivalent to using U-GDL with a necessary condition) and demonstrate that the former incurs an increase in computational overhead and bandwidth requirements of at most a factor 5 and guarantees optimality whereas the latter can perform arbitrarily poorly. For future work, we intend to investigate the setting where the probabilities distributions over the outcomes are not known *a priori*. In this setting, which effectively models a multi-armed bandit with factorisable payoffs, the agents have to trade-off exploration against exploitation.

References

Aji, S. M., and McEliece, R. J. 2000. The Generalized Distributive Law. *IEEE Trans. Inf. Theory* 46(2):325–343.

Atlas, J., and Decker, K. 2010. Coordination for uncertain outcomes using distributed neighbor exchange. In *AAMAS 2010*, 1047–1054.

Dechter, R. 2003. *Constraint Processing*. Morgan Kaufmann.

Fitzpatrick, S., and Meertens, L. 2003. Distributed coordination through anarchic optimization. In *Distributed Sensor Networks*. Kluwer Academic Publishers. 257–295.

Jain, M.; Taylor, M.; Tambe, M.; and Yokoo, M. 2009. Dcops meet the realworld: Exploring unknown reward matrices with applications to mobile sensor networks. In *IJCAI 2009*, 181–186.

Léauté, T., and Faltings, B. 2009. E[DPOP]: Distributed constraint optimization under stochastic uncertainty using collaborative sampling. In *IJCAI 2009 Distributed Constraint Reasoning Workshop (DCR'09)*, 87–101.

Levy, H. 2006. *Stochastic Dominance: Investment Decision Making with Uncertainty*. Springer.

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

Modi, P. J.; Shen, W. M.; Tambe, M.; and Yokoo, M. 2005. Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artif. Intell.* 161(1-2):149–180.

Petcu, A., and Faltings, B. 2005. DPOP: A scalable method for multiagent constraint optimization. In *IJCAI 2005*, 266 – 271.

Rogers, A.; Farinelli, A.; Stranders, R.; and Jennings, N. R. 2011. Bounded approximate decentralised coordination via the max-sum algorithm. *Artif. Intell.* 175(2).

Taylor, M.; Jain, M.; Jin, Y.; Yokoo, M.; and Tambe, M. 2010. When should there be a "Me" in "Team"?: Distributed multi-agent optimization under uncertainty. In *AAMAS 2010*, 109–116.