Robust Distributed Constraint Optimisation

Archie C. Chapman,[1] Alessandro Farinelli[2] and Sarvapali D. Ramchurn[3]

[1] School of Electrical and Information Engineering, University of Sydney, Australia
archie.chapman@sydney.edu.au
[2] School of Computer Science, University of Verona, Italy a.farinelli@uv.edu.it
[3] School of Electronics and Computer Science, University of Southampton, UK
sdr@ecs.soton.ac.uk

Abstract. This paper proposes *minimum p–dominance* as an appropriate measure of solution robustness for distributed systems, when individual components or agents are subject to perturbations in their actions, errors in their perceptions, or mis–specification of their rewards. In particular, applications of p–dominance to distributed constraint optimisation problems (DCOPs) are investigated. Specifically, a local optimum of a DCOP is p–dominant if it is robust to a probability of deviation of at most $1 - p$, which can be interpreted as the probability of a wrong action, perception or reward specification by a subset of the agents. As such, the local optimum with minimum p–dominance is the most robust — it is robust to higher deviation probabilities. This preliminary work develops a *generalised distributive law*–based message–passing algorithm for computing the p–dominance value for each of a DCOP's local optima. This opens a way to directly balance considerations of optimality and robustness when selecting a solution from the set of local optima in DCOPs.

Keywords: Distributed constraint optimisation, robust optimisation, p-dominance, generalised distributive law, Nash equilibrium

1 Introduction

Consider the following collaborative multi–agent scheduling problem: Two users, A and B, share a resource (computation, electricity generation, production etc). Each user has two (un-splittable) jobs to complete; they wich to decide on a schedule to complete all four jobs within two time–slots. The resource can complete jobs at varying speeds, but costs for doing this are quadratic in the sum of size of the jobs scheduled in each slot. For this collaborative optimisation problem, an optimal solution to this problem is one that minimises the global cost. Specifically, let all four jobs be of size 1, and the resource cost function be $(x_t/2)^2$, where x is the total load in time–slot t, as described in Figure 1. Any schedule with total cost of 2 is optimal, namely: A scheduling both of its jobs in time–slot 1 and both of B's in time–slot two, or one job each in each time–slot. In this case, which solution should be selected?

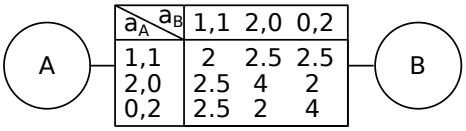

Fig. 1. A two–agent, two–job, two–slot collaborative scheduling problem: actions are the number of jobs per slot; the size of all jobs is 1, and the per-slot cost is $(x/2)^2$.

One proposed *refinement* to the set of solutions is called the *minimum p–dominance* criterion, based on the game–theoretic measure introduced by [14]. Minimum p–dominance is a generalisation of *risk dominance* for games with more than two players and two actions. Specifically, the Nash equilibrium of a game is p–dominant if it is robust to a probability of a correlated, worst–case deviation of at most $1-p$. The minimum p–dominant equilibrium, then, is robust to the greatest chance of such a deviation.

In the example in Figure 1, the minimum p–dominant solution is the one in which each agent submits one job each in each time–slot for processing. To see this, the p–dominance level for each equilibrium is computed, as follows. Since the problem is symmetric, p needs to be computed for only one agent:

- For the solution $(\{0,2\},\{2,0\})$, p is found by solving $2p = 4(1-p)$, where 4 is the cost of the worst–case deviation, giving $p = 2/3$;
- For $(\{2,0\},\{0,2\})$, the solution is also $p = 2/3$;
- For $(\{1,1\},\{1,1\})$, solving $2p = 2.5(1-p)$ gives $p = 4/9 < 2/3$.

Thus $(\{1,1\},\{1,1\})$ is the minimum p–dominant equilibrium, and can be considered the most robust solution to this collaborative scheduling problem.

This paper proposes robustness, as measured by minimum p–dominance, as a selection heuristic for multi–agent optimisation problems with multiple optima. Moreover, in some settings, and in particular distributed constraint optimisation problems (DCOP)s, the robustness of a solution could be used as a measure of desirability itself, to be directly traded-off against reward.

In more detail, the p–dominance criterion is an appropriate way of measuring a solution's robustness in distributed collaborative systems.[4] In these settings, individual components or agents may be subject to perturbations in their actions, errors in their perceptions, or mis–specification of their rewards. Given this, a solution's p–dominance measures the effects on one agent's decision–making of the chance that other agents make mistakes when implementing a solution. Thus, it directly addresses problems that arise in inherently distributed and/or multi–agent interaction.

This paper focus on problems that can be formulated and solved as DCOPs, which then allows the p–dominance criterion to be formulated as a c–semiring

[4] Although p–dominance is defined using game–theoretic reasoning about the optimisation problem, it does not require treating the problem as a non-cooperative game with selfish agents.

and solved using a (distributed) message–passing scheme. Specifically, the method developed here draws on existing results that link the local optima of a DCOP and the pure Nash equilibria of a game–theoretic reformulation of the DCOP framework [11, 4], to apply the p–dominance criterion. Using the DCOP game reformulation, the standard optimality criterion for DCOPs can be interleaved with p–dominance, employing a c–semiring formulation of both, which allows for efficient computation in problems with sparse structure [6, 3].

The paper is structured as follows: The next section introduces the requisite background on c–semirings and DCOPs, including standard approaches to solving them. Then, in Section 3 the two criteria of local optimality and robustness are introduced. Section 4 describes how to encoded local optimality and robustness as a single c–semiring so that it is amenable to distributed computation, and derives an algorithm to compute the robust local optimum of a DCOP. Finally, Section 5 reviews some related work on algorithms for DCOPs and graphical representations of games, and robustness in DCOPs. Section 6 concludes and outlines future directions.

2 Review of DCOPs

A constraint optimisation problem is formally represented by a set of variables $X = \{X_1, \ldots, X_m\}$, each of which may take one of a finite number of values, $x_j \in X_j$, a set of constraints $C = \{c_1, c_2, \ldots\}$, and a global utility function, $V(x)$, that specifies preferences over configurations of states of variables in the system. A constraint $c = \langle X_c, v_c \rangle$ is defined over a set of variables $X_c \subset X$ and a function, v_c, that specifies the value of different configuration of states of the variables X_{c_k}. The objective is then to find a global configuration of variable, x^*, that optimises the global value, which is an aggregation the constraint values:

$$V(x)^* = \max_{x \in X} \left[\sum_{c_k \in C} v_{c_k}(x_{c_k}) \right]. \tag{1}$$

Given this, a DCOP is produced when a set of autonomous agents each independently controls the state of a subset of the variables, called the agent's *action*, but share the goal of maximising the rewards for satisfying constraints. Without loss of generality, only the case where each agent controls one variable is considered. As such, the terms 'state of a variable' and 'action of an agent' can be used interchangeably. The set of agents involved in a constraint is denoted N_c, the set of constraints that i is involved is C_i, and the set of agents that i shares constraints with, i's *neighbours*, is $\nu(i)$.

2.1 *c*–semirings and the generalised distributive law

A standard approach to optimally solving DCOPs relies on formulating them as *c–semirings* and using message–passing to compute a solution in a distributed fashion.

Definition 1. *A* **c–semiring** *is an algebraic structure given by a tuple* $\langle K, \oplus, \otimes \rangle$ *defining a* carrier*,* K*, and two binary operations* $\oplus$ *and* $\otimes$*, for which, for any* $x, y, z \in K$*: (i) both possess identities in* K*; (ii)* $\otimes$ *distributes over* $\oplus$ *(i.e.* $x \otimes (y \oplus z) = x \otimes y \oplus x \otimes z$*); and (iii) both are commutative (e.g.* $x \otimes y = y \otimes x$*).*

Given this, the problem of finding the globally-optimal solution to a DCOP — the solution with maximum value, as defined in Equation (1) — is characterised by a valuation algebra on the semiring $\langle \mathbb{R}, \max, + \rangle$. This formulation can then be solved using one of a variety of message–passing algorithms, all of which exploit the *generalised distributive law* (GDL) to solve the problem using dynamic programming (for a unifying overview of these techniques, see [19]).

In particular, if the factor graph forms a tree, the GDL approach is guaranteed to converge to the optimal solution with predefined number of message exchange (i.e. two messages for each edge).

Building on this, some methods use tree-decompositions to reformulate a loopy factor graph as a tree, and solve this optimally, but the reformulation is itself a hard problem and the new, tree–structured problem is almost always of a higher degree of complexity to solve [19]. Moreover, although guarantees are lacking for loopy factor graphs, experience with GDL algorithms using loopy value propagation is positive, with actions, if not values, converging to high quality solutions with high probability.

Beyond this, one very useful property of c–semirings is that, in many cases, two of them can be transformed and superimposed to produce another c–semiring. This property will be exploited later in order to derive a robust DCOP algorithm.

3 Stability and robustness of DCOP solutions

In this section, two alternative criteria for selecting a solution to a DCOP are characterised — *stability*, which corresponds to *local optimality*, and *robustness*. To do this, a game–theoretic reformulation of the original DCOP problem is introduced. These criteria are then encoded as c–semirings, which can be computed by a distributed GDL–based algorithm. This section begins by reformulating DCOPs as games, before introducing a game–theoretic rationale for the two criteria.

3.1 DCOP games

A DCOP game is formulated by assigning control of each variable in a DCOP to an agent, and defining a private utility function for the agent, $u_i(x_i, x_{\nu(i)})$, which is dependent on both its own action and the actions of other agents in the system. There is some flexibility in the choice of utility function, however, it is vital that an agent's utility only increases when the global solution quality is improved. This is done by setting each agent's utility equal to its local effect on the global utility function, which, in a DCOP, is given by the sum of the

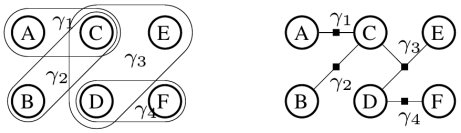

Fig. 2. An acyclic hypergraph (left) and its bipartite vertex–hyperedge incidence tree (right). Circles are agents, hyperedges and squares are local games corresponding to constraints.

constraint values that agent i is involved in:

$$u_i(x_i, x_{\nu(i)}) = \sum_{c_k \in C_i} v_{c_k}(x_i, x_{\nu(i)}). \tag{2}$$

Beyond this, DCOP games can be encoded in a compact graphical representation called *hypergraphical normal form*. In the context of DCOP games, each hyperedge is a team game representation of a DCOP constraint in standard normal form, that is: $\gamma_c = \langle N_c, \{X_i, v_c\}_{i \in N_c} \rangle$, where:

- N_c are the agents involved in constraint c,
- X_i is the variables of agent $i \in N_c$ (nb. x_i is i's global action, which is the same in all of its local games),
- $v_c(x_c)$ is an agent's local utility in a local game, corresponding to the value of the constraint payoff matrix, with $X_c = \times_{j \in N_c} X_j$.

Agents are usually involved in more than one constraint, so let:

- Γ_i be the set of local games in which i is involved,
- $\nu(i) = \cup_{\gamma_c \in \Gamma_i} N_c \setminus i$ be i's neighbours that it shares at least one local game (or constraint) with, and
- $\nu_c(i) = N_c \setminus i$ be i's neighbours in γ_c.

Finally, agent i's payoff, given Equation (2), which can be equivalently expressed as: $u_i(x_i, x_{\nu(i)}) = \sum_{\gamma_c \in \Gamma_i} v_c(x_i, x_{\nu_c(i)})$.

Because of their algorithmic benefits, we are particularly interested in acyclic hypergraphs, or *hypertrees*. The results here rely on classical acyclicity,[5] which is defined in reference to a hypergraph's *vertex–hyperedge incidence graph*: a bipartite graph with one set of nodes containing the nodes of the hypergraph and the second containing the hyperedges. A hypergraph is acyclic if its vertex–hyperedge incidence graph is a tree (as in Fig 2), and the leaves of this graph are called the leaves of the hypergraph.

Now, in a game each agent desires to maximise its private utility, and distributed solutions to the DCOP can be produced by the independent actions of the agents in the system. Specifically, if an agent's goal is to maximise its payoff, then its *best response*, $B_i(x_\nu(i))$, is the set of i's best strategies given its neighbours' strategies:

$$B_i(x_\nu(i)) = \underset{x_i \in X_i}{\operatorname{argmax}}\, u_i(x_i, x_\nu(i)).$$

[5] C.f. the weaker form of α*–acyclicity* [1, 7]

Stable points are characterised by the set of pure Nash equilibria, in which every agent plays a best response:

$$x_i^* \in B_i(x_\nu(i)) \quad \forall i \in N.$$

Importantly, the Nash equilibria of the DCOP game correspond to the local optima of the DCOP, as shown in [4] and discussed in more detail in the context of p–dominance below.

In a game, action sets can be extended to consider *mixed strategies*, $\sigma_i \in \Delta(X_i)$, which are lotteries over the set of actions. Furthermore, agents' strategies can be correlated, such that $\mu \in \Delta(X)$. The best response condition can be directly extended to these strategies: let $\mu_{\nu(i)} \in \Delta(X_{\nu(i)})$ be a correlated mixed strategy played by all of i's neighbours, then $x_i \in B_i(\mu_\nu(i))$ is a best response for i.[6]

Stability is a necessary condition for any useful solution to a DCOP, and in the next section, the concepts of Nash equilibrium and local optimality in DCOP games are linked. Then, in order to formally define robust solutions, a refinement criterion that ranks the stable solutions according to their robustness to other agents' worst–case deviations from a Nash equilibrium profile is introduced.

3.2 Stability, local optimality and Nash equilibrium

In [4], the Nash equilibria of the DCOP game as constructed above were shown to correspond to the local optima of the DCOP. Specifically, by constructing agents' utility functions according to Equation (2), the DCOP is cast as a *potential game*. The defining feature of potential games is that any change in action that increases an agent's private utility also increases the potential function. In the case of DCOP games, the potential function *is* the global value of the system, so all unilateral improvements improve the global solution quality. Furthermore, potential games are guaranteed to have pure strategy Nash equilibria, which arise at the local optima of the global value function [13]. Thus the stable solutions to a DCOP game are locally optimal.

This equivalence is now used to encode c–semiring condition indicating when a variable configuration is stable (in a similar way to [3]). The Nash equilibrium condition can be expressed as the product of a set of indicator functions, $\mathbb{I}\{x_i \in B_i(x_{\nu(i)})\}$, each equal to 1 when an agent plays a best response and ∞ otherwise:

$$\min_{x \in X}[V^{\mathrm{NE}}(x)] = \min_{x \in X}\left[\max_{i \in N} \mathbb{I}\{x_i \in B_i(x_{\nu(i)})\}\right]. \tag{3}$$

Equation (3) defines the c–semiring $\langle\{1, \infty\}, \min, \max\rangle$, which is equal to 1 if x is a local optimum and ∞ elsewhere. Importantly, the carrier value ∞ is *absorbing* under maximisation, making this a variation on the Boolean semiring, with ∞ taking the place of 0.

[6] Such correlated actions lead to a definition of a correlated equilibrium, which generalises Nash equilibrium, but this is beyond our requirements.

This encoding may seem unintuitive, and indeed, the best-response indicator may have been encoded differently, so that the c–semiring had a different carrier and operators. Our choice of these values is driven by our need to combine the stability and robustness criteria, as will be made clear in the next section. Also note that the conventional reward-optimising DCOP solution is one of the local optima, so global reward optimisation also implicitly satifies the stability requirement.[7]

3.3 Robustness of Nash equilibria

In [14], minimum p–*dominance* is developed as a generalisation of the classical notion of risk dominance for games with more than two players and two actions. It is characterised by considering the probability of an agent's opponents' correlated deviations from an equilibrium strategy.

In more detail, let $x^* = \{x_i^*, x_{\nu(i)}^*\}$ be a pure strategy Nash equilibrium, and let $\lambda_{\nu(i)}^p(x^*) \in \{\mu_{\nu(i)} \in \Delta_{\nu(i)} \ : \ \mu(x_{\nu(i)}^*) \geq p\} = \Lambda_{\nu(i)}^p(x^*)$ be any correlated joint mixed strategy of i's opponents that places at least probability p on the playing equilibrium action profile $x_{\nu(i)}^*$.

Definition 2. *An agent's* **p–level** *for a (pure) Nash equilibrium, $p_i^{\min}$, is the lowest value p for which the agent's best response remains its equilibrium action x_i^* in response to any correlated mixed strategy by its opponents $\lambda_{\nu(i)}^p(x^*)$ that places at least probability p on the playing equilibrium action profile $x_{\nu(i)}^*$:*

$$p_i^{\min}(x^*) = \inf \left\{ p \in [0,1] \ : \ x_i \in B_i(\lambda_{\nu(i)}^p(x^*)) \ \forall \Lambda_{\nu(i)}^p(x^*) \right\}, \tag{4}$$

and a Nash equilibrium is called **p–dominant** *if $\forall i \in N$, $p_i^{\min}(x^*) > p(x^*)$.*

As such, a Nash equilibrium is p–dominant down to a level equal to the maximum of any agent's p–level: $p(x^*) = \max_{i \in N} p_i^{\min}(x^*)$. Intuitively, p–dominance is a measure that spans the range of robust solutions to games from non–strict Nash equilibria ($p = 1$), through all strict Nash equilibria ($0 < p < 1$), to dominant–strategy equilibria ($p = 0$). Thus, p–dominance is in effect a robustness measure: A lower p–dominant Nash equilibrium is robust to larger chances of deviations from the equilibrium action profile than those equilibria with a higher value of $p(x^*)$. In turn, maximal robustness of this form can be defined.

Definition 3. *The* **minimum p–dominant equilibrium** *is the one that minimises the maximum p–level for any agent, that is:*

$$\min_{x \in \mathrm{NE}} p(x) = \min_{x \in \mathrm{NE}} \left[\max_{i \in N} \left(p_i^{min}(x) \right) \right]$$

where $p^{min}(x)$ is the vector containing all agent's p–level, and NE is the set of pure Nash equilibria.

[7] That is, DCOP games have a *price of stability* of one.

Since minimum p–dominance is used to select between Nash equilibria, p–values for non-equilibrium action profiles are not defined. Thus, in order to compute the minimum p–dominant equilibrium, one must first compute the set of Nash equilibria, or interleave the computation of the two. An algorithm for this combined problem is derived in the next section.

4 Computing Robust DCOP solutions

In this section, a distributed algorithm for computing the most robust local optima of a DCOP is described.

First, the task of identifying the set of local optima with that of computing their minimum p-level must be combined; that is Equation (3) will be interleaved with Definition 3. To begin, observe that the scope of $p_i^{min}(x)$ needs to be extended to all joint action profiles. Let:

$$\tilde{p}_i^{min}(x) = \begin{cases} \mathbb{I}\{x_i \in B_i(x_{\nu(i)})\} p_i^{min}(x) = p_i^{min}(x) & \text{if } x_i \in B_i(x_{\nu(i)}), \\ \mathbb{I}\{x_i \in B_i(x_{\nu(i)})\} = \infty & \text{otherwise.} \end{cases} \tag{5}$$

Using this, the minimum p–dominance criterion for all $x \in X$ can be encoded by:

$$\min[V^{pD}(x)] = \min_{x \in X} \left[\max_{i \in N} \left(\tilde{p}_i^{min}(x) \right) \right]. \tag{6}$$

Given this, minimum p–dominant equilibrium is computed using the semiring $\langle \{[0,1] \cup \infty\}, \min, \max \rangle$. Specifically, any action profile that contains a non–best response is assigned a value of ∞; while for each Nash equilibrium, the maximum p_i^{min} value in the system is returned. These are then ranked over all outcomes by finding the minimum p–level attained by any equilibrium; while all non-equilibria are excluded because their values are ∞. This action profile is the most robust to worst–case correlated deviations by any subset of players in the DCOP, in that it is robust to the largest correlated perturbation to the Nash equilibrium action profile.

Next, an efficient method for computing this criterion on sparse problems using a message–passing algorithm is described.

Specifically, a variant of the VNP algorithm [3] is derived and tailored for computing minimum p–dominance in DCOPs. The agents running the algorithm use the function in Equation (5) to produce a new agent-hypergraph, corresponding to a hypergraphical representation of the DCOP game reformulation. Then, as is typical of GDL–type algorithms, messages are then exchanged between nodes according to a two–phase message–passing sequence, called *table–passing* and *assignment–passing*, which directly identifies and ranks the local equilibria of the DCOP. It is important to again note that the agent-hypergraph discussed above is *not* the factor graph of the underlying DCOP.

As computation on loopy topologies is inherently more complicated, the following only describe the hypertree case, for which convergence guarantees can

```
Function vnpTablePassing(Γ_i)
for each hyperedge c : Γ_i,
    if values T_{j→i} have been received from all neighbours outside c, j ∈ ν_{−c}(i),
        compute T_{i→j} for neighbours in c, j ∈ ν_c(i);
        send T_{i→j} to neighbours in c, j ∈ ν_c(i);
    end if
end for
```

```
Function vnpAssignmentPassing(Γ_i, T_{j→i} ∀j ∈ ν(i) )
if an assignment S_{j→i} has been received from any neighbour, j ∈ ν_c(i) ∀c ∈ Γ_i,
    for each hyperedge c : Γ_i,
        if no neighbour k ∈ ν_c(i) has sent a message S_{k→i},
            compute S_{i→k} for neighbours in c, c ∈ ν_c(i);
            send S_{i→k} to neighbours in c, k ∈ ν_c(i);
        end if
    end for
end if
```

Fig. 3. Pseudocode of the two phases of VNP on a hypertree, which is used to compute the minumum p–dominant action.

be obtained; a description of the operation of VNP on loopy hypergraphs can be found in [3]. Specifically, Section 4.1 shows the algorithm is initialised, Section 4.2 describes the table–passing phase, and Section 4.3 the assignment passing phase.

4.1 Constructing the VNP–graph and initialisation

The minimum p–dominant Nash equilibrium is computed starting with the DCOP game reformulation of the underlying DCOP, encoded in hypergraphical normal form: denote this hypergraph the VNP–graph. As described in Section 3.1, nodes on the VNP–graph represent the agents' actions, and the hyperedges represent undirected utility dependencies between agents. The hyperedges and the agents on them in the VNP—graph are used define the scope and content of the messages passed in VNP.

Each agent also computes their own valued best-response (VBR) functions, as given by Equation (5). Each agent stores the values of its VBR functions in a look-up table, to be used when constructing and processing messages during VNP's operation.

4.2 The Table–Passing Phase

In this phase, agents exchange tables with their neighbours in each hyperedge. These tables contain values which succinctly represent the value of the ranking of each joint strategy of the hyperedge's agents. For reference, pseudocode of VNP's table– and assignment–passing phases on hypertrees is given in Fig 3.

The messages exchanged in the table–passing phase are an array, indexed by the joint strategy space of the agents in the hyperedge common to both sender and recipient. Specifically, i passes to its neighbour j in hyperedge γ_c an array $\mathcal{T}_{i\to j}$ with $\prod_{k\in N_c}|X_k|$ entries, indexed by an ordered $|N_c|$–tuple representing a joint strategy of the agents in γ_c. An element of a message is denoted $\mathcal{T}_{i\to j}(x_i, x_{\nu_c(i)})$. Agents store the messages they receive for use in the second phase.

Using the notation of Equation (5), elements of the message i sends to j are computed by:

$$\mathcal{T}_{i\to j}(x_i, x_{\nu_c(i)}) = \min_{x_{\nu_{-c}(i)}} \left(\max \left[\tilde{p}_i^{min}(x_{\nu_c(i)}, x_{\nu_{-c}(i)}), \left\{ \mathcal{T}_{k\to i}(x_i, x_k, x_{\nu^h(k)\setminus i}) \right\}_{\substack{k\in\nu^h(i) \\ \gamma_h\in\Gamma_i\setminus\gamma_c}} \right] \right) \tag{7}$$

where $\nu_{-c}(i) = \nu(i) \setminus \nu(i)_c$. The operations above perform two processes. First, for a fixed joint strategy $(x_i, x_{\nu(i)_c})$ in γ_c, the max operator *combines* the value of the joint strategy of the agents in all of the hyperedges containing i except γ_c (i.e. $\Gamma_i \setminus \gamma_c$) to give rankings to each joint strategy of i's neighbours outside of γ_c. In this combination, the indicator function sets the entries to ∞ if the joint strategy is not in equilibrium with x_i. Second, the minimisation then *summarises* the information regarding $x_{\nu_{-c}(i)}$ and projects the information onto the fixed joint strategy in γ_c. If there is no equilibrium associated with x_i, then the output is ∞. Now, to complete the minimisation component, an agent needs to find the maximum value message it has received regarding strategies of different agents. To do this, the agent *extends* each incoming message to the joint strategy space of all incoming messages, by combining the relevant message entries to give a value for the respective joint strategy.

When the termination condition for the table–passing phase is met, agent i uses the messages it received from its neighbours to construct the following function:

$$V_i(x_i, x_{\nu(i)}) = \max \left[\tilde{p}_i^{min}(x), \left\{ \mathcal{T}_{j\to i}(x_i, x_{\nu_c(i)}) \right\}_{\substack{\gamma_c\in\Gamma_i \\ j\in\nu_c(i)}} \right], \tag{8}$$

where the second term inside the max operator is the set of messages received by agent i. The above is equal to the equilibrium selection criterion in Equation 6 in games with hypertree interaction structure:

Theorem 1. *The table–passing phase of VNP, using factor functions Equation* (5)*, produces the value of the p dominance criterion for each local pure strategy profile in a hypertree–structured game.*

Proof is identical to that of Theorem 1 of [3] and is by induction, as is standard for GDL–type algorithms.

If the minimum p–dominant solution is unique or doesn't exist, then the table's values show this unambiguously, and each agent selects its strategy without requiring an assignment–passing phase. However, if more than one solution exists, then the agents coordinate on a solution using assignment–passing.

4.3 The Assignment–Passing Phase

Once the termination condition for the table–passing phase has been satisfied, the function in Equation (8) is computed and each agent uses it to decide on a strategy. If a unique solution (or the non–existence of one) is not evident at the

end of the table–passing phase, then the agents coordinate on a single solution using an assignment–passing phase.

To begin, a spanning tree is constructed, emanating from an arbitrarily selected root node, down which parents pass messages containing local strategy profiles to their descendant This tree is built by first connecting the root to all of its neighbours. A particular agent, i, which interacts with the root through γ_c, is either a leaf or an internal node involved in other hyperedges. If it is an internal node, then it is the only agent that appears in both γ_c and any of its other hyperedges (by the definition of acyclicity). As such, the spanning tree is extended by connecting i to all of its neighbours except those in γ_c, and so on.

Given this spanning tree, the root randomly selects a local strategy profile corresponding to one maximum of its version of Equation (8). It then sends messages, $\mathcal{S}_{r\to i}$ to its neighbours in each hyperedge, $i \in N^{\gamma_c}$, containing the local strategy, x_{N_c}, for γ_c. In hyperedges with three or more agents, this is necessary to avoid mis–coordination between two of the root's neighbours, because more than one optimal joint strategy could be associated with the root's strategy. Then i selects a joint strategy for all of its hyperedges *except* γ_c that both maximises its version of Equation (8) and contains the strategy assigned to it by the root, choosing randomly between equivalent strategies. It sends $\mathcal{S}_{i\to j}$ strategy assignments to all of its unassigned neighbours j. This process continues until each leaf of the spanning tree is assigned a strategy by its parent, when the algorithm terminates.

Lemma 1. *The assignment–passing phase of VNP for hypertrees assigns each agent a strategy corresponding to a single criterion–maximising PSNE in a hypertree–structured game.*

The proof is omitted for brevity, but follows the same argument for the max–product algorithm presented in [20]. However it should be clear from the above that, by induction, the root's choice propagates through the width–first spanning tree without any mis–coordination of agents' strategies. Combining Theorem 1 and Lemma 1 gives us the following result.

Theorem 2. *In DCOP games, for the factor functions in Equation* (5)*, VNP computes a p–dominance minimising Nash equilibrium.*

Additionally, it has been shown that if the number of neighbours and the number of hyperedges each agent has is bounded, VNP is polynomial in the total number of agents, because each local computation is polynomially bound [7, 3].

5 Related work

The algorithm presented in this paper is related to the exact DCOP message–passing algorithms DPOP and Action-GDL [19], and to the VNP algorithm for optimising over the set of pure Nash equilibria in a game [3]. VNP itself built on recent work on compact game representations, which yielded several algorithms

that efficiently solve graphical games by exploiting their graphical structure. These include the NashProp algorithm, which computes a Nash equilibria in tree–structured graphical games [8, 15], an adaptation of the max–product algorithm that operates on a reduction of a game to a Markov random field to compute pure strategy Nash equilibria on arbitrary graphs [5], and three algorithms that all work by mapping the Nash equilibrium computation problem to a constraint satisfaction problem (CSP) — PureProp [16], and algorithms by Vickrey and Koller [18] and Gottlob, Greco and Scarcello [7].

The problem considered in this paper is related to previous work applying worst–case robustness to DCOPs, albeit in different contexts and for different purposes. For example, [10] address the problem of maximising the worst-case payoff outcome in an mobile sensor network management problem. In contrast to the setting in this paper, the worst–case outcome is treated a function of the behaviour of a rational adversary, rather than of a part of the team of sensors.

A *quantified* (Q-) DCOP is considered in [12], in which some variables, known as universally quantified variables, are left unassigned by the optimisation routine. Following the standard approach to robust optimisation, a Q-DCOP optimisation is the prolbem of finding an assignment of remaining (non–universally quantified) variables that maximises reward given the worst–case choice of the universally quantified variables. In contrast, this paper adresses robustness to the chance of worst–case purturbations to *any* variable, which is a probabilistic notion, rather than the deterministic worst–case values of onlly those variables in a predefined set (i.e. the universally–quantified variables).

6 Conclusion and future work

This paper represents a first step towards characterising and analysing robust solutions to distributed optimisation problems based on p–dominance, with particular focus on DCOPs. There are a number of directions that we intend to pursue, discussed below.

Application domains: We believe this concept of robustness can be used in a wide variety of important applications. In particular, we aim to investigate p–dominant robust solutions to DCOPs in scenarios where agents should coordinate while considering possible deviations in the strategies of other agents due to lack of knowledge on their behaviors. Such lack of knowledge can be due to several elements such as dynamism of the environment, the fact that agents may enter and leave while the system is running, or to the presence of human operators which could have behaviors that are very hard to model and predict.

In more detail, in the context of robotics there has been a significant interest towards scenarios where robots collaborate without a predefined coordination protocol [17]. In this context, agents might try to learn other agents' policies while collaborating and look for coordinated solutions that are robust to possible deviations.

Another interesting field is *Human Robot Interaction*, and specifically application scenarios where few operators should control a team of robots [2]. In this context, robots should have a significant degree of autonomy to alleviate the workload of the operator, but at the same time they should involve the humans in critical decisions and most provide relevant info to precisely assess the current situation. One way to approach this problem is for robots to build an estimate of what actions humans typically perform in certain situations and try to compute strategies that are robust to possible deviations.

Analysis of approximate solvers and local search heuristics: At AAMAS 2014, Lesser and Corkill [9] posed a blue-sky challenge, regarding the ability of heuristic DCOP (and dec–MDP) solvers to produce high–reward solutions with additional, desirable characteristics. These characteristics regard, roughly–speaking, the robustnesss of the solutions they produce to perturbations in its implementation and mis–specifications of reward function. In their paper, Lesser and Knight identify this as a key area where understanding of both the characteristics of the solutions produced, and the methods by which heuristic solvers find these solutions, are lacking. We seek to address a part of this challenge, by using our characterisation of robust solutions to DCOPs to analyse the trade–off between robustness and optimality that are produced by effective local search heuristics.

References

1. C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of acyclic database schemes. *J. ACM*, 30(3):479–513, 1983.
2. N. Brooks, E. de Visser, T. Chabuk, E. Freedy, and P. Scerri. An approach to team programming with markup for operator interaction. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 1133–1134, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems.
3. A. C. Chapman, A. Farinelli, E. M. de Cote, A. Rogers, and N. R. Jennings. A distributed algorithm for optimising over pure strategy Nash equilibria. In *AAAI '11*. AAAI Press, 2011.
4. A. C. Chapman, A. Rogers, N. R. Jennings, and D. S. Leslie. A unifying framework for iterative approximate best response algorithms for distributed constraint optimisation problems. *Knowledge Engineering Review*, 2009.
5. C. Daskalakis and C. H. Papadimitriou. Computing pure Nash equilibria via Markov random fields. In *ACM EC '06*, San Diego, California, USA, 2006.
6. E. Elkind, L. A. Goldberg, and P. W. Goldberg. Computing good Nash equilibria in graphical games. In *EC '07*, pages 162–171, 2007.
7. G. Gottlob, G. Greco, and F. Scarcello. Pure Nash equilibria: Hard and easy games. *JAIR*, 24:357–406, 2005.
8. M. Kearns, M. L. Littman, and S. Singh. Graphical models for game theory. In *UAI '01*, pages 253–260, 2001.
9. V. Lesser and D. Corkill. Challenges for multi-agent coordination theory based on empirical observations. In *Proc. AAMAS 2014*, 2014.

10. V. Lisý, R. Zivan, K. P. Sycara, and M. Pechoucek. Deception in networks of mobile sensing agents. In *Proceedings of the 2010 International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2010)*, pages 1031–1038, 2010.
11. R. T. Maheswaran, J. P. Pearce, and M. Tambe. Distributed algorithms for DCOP: A graphical-game-based approach. In *Proceedings of PDCS '04*, pages 432–439, San Francisco, CA, 2004.
12. T. Matsui, H. Matsuo, M.-C. Silaghi, K. Hirayama, M. Yokoo, and S. Baba. A quantified distributed constraint optimization problem. In *Proceedings of the 2010 International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2010)*, pages 1023–1030, 2010.
13. D. Monderer and L. S. Shapley. Potential Games. *Games and Economic Behavior*, 14:124–143, 1996.
14. S. Morris, R. Rob, and H. S. Shin. p-dominance and belief potential. *Econometrica*, 63:145–157, 1995.
15. L. E. Ortiz and M. J. Kearns. Nash propagation for loopy graphical games. In S. Becker, S. Thrun, and K. Obermayer, editors, *NIPS*, volume 15, pages 793–800, 2003.
16. V. Soni, S. Singh, and M. P. Wellman. Constraint satisfaction algorithms for graphical games. In *AAMAS '07*, pages 423–430, 2007.
17. P. Stone, G. A. Kaminka, S. Kraus, and J. S. Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, 2010.
18. D. Vickrey and D. Koller. Multi-agent algorithms for solving graphical games. In *AAAI '02*, pages 345–351, 2002.
19. M. Vinyals, J. A. Rodriguez-Aguilar, and J. Cerquides. Constructing a unifying theory of dynamic programming DCOP algorithms via the generalized distributive law. *Autonomous Agents and Multi-Agent Systems*, 2011.
20. M. Wainwright, T. Jaakkola, and A. Willsky. Tree consistency and bounds on the performance of the max-product algorithm and its generalizations. *Stat. and Comp.*, 14(2):143–166, 2004.