Solving DCSP problems in highly degraded communication environments

Saeid Samadidana
University of Tulsa
800 S Tucker Dr
Tulsa, Oklahoma, USA 74104
saeid-samadidana@utulsa.edu

Roger Mailler
University of Tulsa
800 S Tucker Dr
Tulsa, Oklahoma, USA 74104
mailler@utulsa.edu

ABSTRACT

Although there have been tremendous gains in network communication reliability, many real world applications of distributed systems still face message loss, limitations, delay, and corruption. Yet despite this fact, most Distributed Constraint Satisfaction (DCSP) protocols assume that communication is perfect (messages that are sent will be received) although not ideal (not in a timely manner). As a result, many protocols are designed to exploit this assumption and are severely impacted when applied to real world conditions. One notable exception is the Distributed Probabilistic Protocol (DPP), which was originally designed to solve DCSP problems in communication restricted environments. The original DPP paper, however, only showed that the protocol works well on distributed graph coloring problems and was not tested using communication degraded conditions. In this study, we expand upon the modified version of DPP(mDPP) to show that the protocol is very effective at solving general DCSP problems in highly degraded communication environments. The new version of DPP brings into account the fact that an agent who has not sent a message for a while will collect the evidence to change the probability of changing its value. To test the impact of message degradation on the performance of distributed algorithms, we simulated a real environment using probabilistic message loss and delay. Under these conditions, MGM shows poor results while DPP and DSA are still able to operate efficiently. DBA in some cases shows acceptable results however like MGM displays poor results under message delay issues and in some cases of message loss.

CCS CONCEPTS

• **Computing methodologies** → **Multi-agent systems**;

KEYWORDS

Constraint Satisfaction, Dynamic, Robustness, Probability, Communication

WI 2017, Leipzig, Germany

DOI: 10.475/123_4

ACM Reference format:
Saeid Samadidana and Roger Mailler. 2017. Solving DCSP problems in highly degraded communication environments. In *Proceedings of Web Intelligence 2017 conference, Leipzig, Germany, 23-26 August 2017 (WI 2017),* 9 pages.
DOI: 10.475/123_4

1 INTRODUCTION

Many distributed algorithms like DSA [10], DBA [11], MGM [8] and Max-Sum [1] are designed for solving problems where centralized algorithms cannot be applied because there are memory and processing limitations or privacy concerns. A central complaint about distributed approaches stems from their reliance on the network layer to provide perfect communications. However, despite tremendous progress in communication technology there are still numerous problems such as information loss, data corruption, and delivery delay that cannot be avoided. Oddly, this important issue is largely ignored in designing distributed algorithms.

Some notable exceptions to this trend can be found in the work of Modi and Ali [6], Wahbi and Brown [7]. In Modi and Ali, the authors modify the ADOPT protocol to work with the network layer to reduce network load. This was done by tagging each message to indicate its importance. When an unimportant message failed to be communicated properly it was just ignored. This significantly reduced the number of messages that needed to be resent, which reduced overall load. In the work of Wahbi and Brown [7] the authors tested the ABT [5] and AFC-ng [3] protocol using a multi-hop wireless network simulator. They were able to show that the performance of both protocols is profoundly affected by the network's topology.

This work builds on the work of Mailler [4] in which the author develops a protocol called the Distributed Probabilistic Protocol (DPP). DPP utilizes prior probability distributions of agent's ability to reduce constraint violations combined with Bayesian reasoning to significantly reduce the need to coordinate. The end result, as shown in the paper, is a dramatic reduction in network load while still providing fast, high-quality solutions. However, the original DPP work failed to address some important issues with computing the prior probability distributions, the protocol was only tested on graph coloring problems, and like many other studies, the paper did not address communication degradation. In paper, it is shown that the mDPP works on general DCSP problems and it compared to to MGM, DBA, and DSA algorithms.

Then the performance of the new version of DPP, MGM, DBA, and DSA in communication degraded environments is analysed.

The structure of this paper is as follows: First, in section two we provided a brief formal definition of a DCSP problem and the DSA, DBA, and MGM algorithms. The original version of DPP and mDPP are explained in section three. In section four we provided the test results in different scenarios and compared the algorithms' performance. Section three is divided to three subsections dedicated to no message loss, message loss and message delay cases. In the last section, we conclude the study and mention some areas for future studies.

2 BACKGROUND

2.1 Problem description

Generally, a Constraint Satisfaction Problem(CSP) is defined as [5]:

- a set of n variables: $V = \{v_1, \ldots, v_n\}$.
- A set of g agents: $A = \{a_1, \ldots, a_g\}$.
- Discrete, finite domains for each variable: $D = \{D_1, \ldots, D_n\}$.
- A set of constraints $C = \{c_1, \ldots, c_m\}$ where each $c_i(d_{i,1}, \ldots, d_{i,j})$ is a function $c_i : D_{i,1} \times \cdots \times D_{i,j} \rightarrow \{true, false\}$.

With the definition above the goal is to find an assignment $A = \{d_1, \ldots, dn_n\}$ of domain values to variables such that all constraints of R are evaluated to true. However, it is not always possible to find such a solution. It is shown that CSP problem is NP-complete. In DCSP problems, usually some agents are defined such that each agent is responsible for one or more variables and their goals are satisfying their variables' constraints. For each agent to satisfy its variables' constraints, however it must communicate with other agents. Neighbors are defined as two directly connected variables. It is possible, without loss of generality, to assume that every agent has only one variable assigned to it. In this paper, only binary constraints are used and result can easily be extended to general cases.

Two important factors in DCSP problems are Density and Tightness that are known by $p1$ and $p2$ respectively and defined as:

- $p_1 = 2c/n(n-1)$
- $p_2 = falseassignments/possibleassignment$

Another useful measure is degree that is degree that is defined as $d = c/n$ and c is the number of constraints. In this paper, all problems have 3 domain values and tightness is set to 0.33 that means 3 out of 9 possible constraint assignment are randomly set to false.

2.2 Distributed Stochastic Algorithm

DSA [10] is a stochastic algorithm for solving the DCSP problems that uses some constant probabilities. It is shown that DSA is very effective in solving graph coloring problems and it converges to a solution very fast with a reasonable number of messages that each agent sends. However as it is mentioned in the review by Leite et. al [2] DSA is not guaranteed to find the global optimal solution. Maheswaran [8] provided an example in which DSA does not yield the global optimum solution. They showed that DSA is not effective in DCOP problems. It means that when the utility values for satisfying or violating the constraints are different DSA cannot find the global optimum solution.

DSA starts with setting a change probability p value before running the algorithm and each agent changes its value based on that probability. It is shown that different values of the probability p for the same problem, impact the performance and convergence rate of the algorithm. A small value of p leads to slower convergence that promotes the stability of the solution and large value of p cause a faster convergence rate. In many problems, therefore, p is set to 0.3. Having a fixed probability, has problems. In dynamic and changing environments the topology of the network changes periodically so setting a value of p can be difficult. DSA has five different variants and in this paper we used the DSA-B variant. In this variant, values change based on the probability p when number of constraint violations is decreased or when there is no change in the number of improves. A complete description of all variants of DSA can be found in work of Zhang [10].

2.3 DBA and MGM

The Distributed Breakout Algorithm(DBA) and Max-Gain Messages(MGM) algorithms are two other popular distributed algorithms in which agents send their current values periodically. The DBA procedure is simple. First, all agents send their initial values to its neighbors. After receiving all values from neighbors, agent sends its improvement value and waits to get neighbors' improve messages. After getting all improve values from neighbors, agent will find if it is allowed to change its value or not. If its improvement is better than all others, it will change its current value to new one and will send it to all neighbors otherwise it will just sends its current value. DBA has a mechanism to escape from local minimum, therefore, it is expected to show better result in comparison to MGM and it shown this is true.

One problem that prevents the use of DBA is the huge number of messages agents send. It does not matter if one agent wants to change its value it must send its current value to all its neighbors. In the case that communication cost is high or fewer messages is preferred DBA can be problematic. As we show in case of communication delay the efficiency of sharply drops. As well there are some probabilities of losing messages, performance of DBA depends highly on the timeout parameter. As it is shown, DBA works quite well when timeout is set to 2 cycles but when timeout is set to 4 cycles the performance decreases sharply.

In the Max Gain Message algorithm(MGM), agents start with sending their initial value. Then after getting all initial values from neighbors they calculate their maximum possible gain and send it to the neighbors. Then each agent waits for its neighbors Max-Gain message. The agent with higher max

gain will change its value. The gain value can be anything defined by the problem, like the minimum conflicts or maximum score of constraints pairs. This algorithm, like DBA, suffers from sending a huge number of messages in each cycle. In each cycle agents send their value they have changed or not. MGM like DBA suffers from the problem of deadlocking due to message delay or loss. Therefore, it needs a time-out mechanism to prevent from deadlocking. In addition, MGM unlike DBA does not have a mechanism to escape from local minimum therefore we should expect poor results compared to DBA. Our results confirm this idea. Indeed, results show MGM has the weaker results compared to the algorithms discussed here.

3 DISTRIBUTED PROBABILISTIC PROTOCOL (DPP)

DPP is a distributed protocol designed by Mailler [4] that uses prior knowledge to decide about each agent's next move. The original idea of designing DPP was to create an algorithm that is robust to communication problems and is not dependent on other agents information. In this algorithm, each node sends its initial value and a list of improve probabilities(PDF) to its neighbors. Then in each cycle, every agent calculates its improve and change probabilities. This is done by using the improve probabilities sent by its neighbors and its own PDFs and its previous improve values that it had sent to neighbors before. Mailler has shown in his paper that by having *a priori* knowledge of neighbors probability of change and improvement values, problem can be solved using little communication and fast convergence rate. Figure 1 shows the initial and main procedures of this protocol. Figures 2 and 3 show the procedures for calculating the improve values and sending Ok? and Improve? Messages respectively.

The original version of DPP was only tested on graph coloring problems. Usually in graph coloring problems, constraints are simply structured. Although, this assumption is not necessary, it makes problem less complicated. One of the big advantage of the DPP is that it does not require an initial setting by user. It simply works with some initial estimates of each agent improve probabilities(PDF) and updates change and improve probabilities in each cycle. DPP works based on some probabilities to change or keep current value. DPP unlike some other algorithms like the DSA does not need the probability to be set by user and tries to update the probabilities in each cycle based on prior information. This feature of DPP can be very effective in dynamic problems in which constraint distribution, density and tightness of problem may change. The change probability of each agent in DPP is calculated by taking the product over the probability that agent X has the highest improve among all neighbors Y. This is shown in equation 1. In the next section we provide a modified version of this equation. In this paper the new version of DPP is used to compare with the other algorithms. We show that the mDPP is a good candidate for highly degraded problems.

procedure **main**
initialize;
while (not terminated) **do**
update *agent_view* with ok? (x_j, d_j) messages;
update *agent_view* with improve $(x_j, improve_j)$ messages;
calculate_improve;
send_ok;
send_improve;
end do;
end main;
procedure **initialize**
$pdf_i \leftarrow$ **calc_improve_pdf**;
send ((**init**, (x_i, d_i, pdf_i)) to all $x_j \in neighbors$;
while (not received **init** from $x_j \in neighbors$) **do**
update *agent_view* with incoming init (x_j, d_j, pdf_j) messages;
end initialize;

Figure 1: The main and initialize procedures of the DPP algorithm.

procedure **calculate_improve**
$v \leftarrow$ the value with the least conflict $(v \neq d_i)$;
if d_i has no conflicts **or** v has more conflicts than d_i **do**
$new\_value = d_i$;
$improve_i \leftarrow 0$;
else
$new\_value = v$;
$improve_i \leftarrow$ difference in conflicts between d_i and v;
end if;
end calculate_improve;

Figure 2: The calculate_improve procedure of the DPP algorithm.

$$\prod_{Y_i=Y_1}^{Y_m} P(Y_i \leq x | improve_x^t = i) \quad (1)$$

3.1 Modified Version Of DPP(mDPP)

Before going through to the explanation of the new version of the DPP lets see what was the problem with change probability equation. Suppose an agent has 2 neighbors and in last cycle it had 1 improvement and in this cycle it has 2 improvement.

based on the equation 1 the probability of changing to new value is a small number due to production of probabilities therefore even the agent has a higher improvement it is quite unlikely to change to new value. This problem is fixed by the idea that after some cycles that one agent does not have the

procedure **send_ok**
if $new\_value \neq d_i$ **do**
$p \leftarrow$ **calc_change_probability**;
if $random < p$
$d_i \leftarrow new\_value$;
send ((**ok?**, (x_i, d_i)) to all $x_j \in neighbors$;
$improve_i \leftarrow 0$;
end if;
end if;
end send_ok;
procedure **send_improve**
for all $x_j \in neighbors$ **do**
$p \leftarrow$ **calc_improve_probability**(x_j);
if $random < p$
send ((**improve**, $(x_i, improve_i)$) to x_j;
end if;
end do;
end send_improve;

Figure 3: The send_ok, and send_improve procedures of the DPP algorithm.

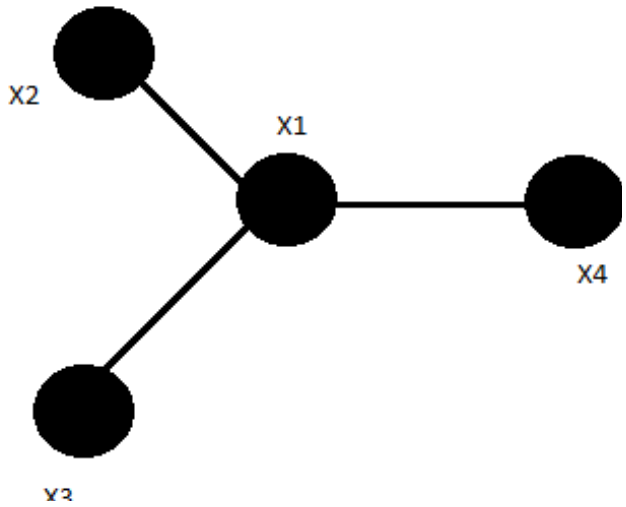

Figure 4: A simple DCSP Problem

chance to change its value and its improvement value is high must have higher chance to change its value. Therefore, after a while all agents that have the highest improve potential to solve the constraints will change their values. Equation 2 shows how the probability for each neighbors is obtained and equation 3 shows the modified version of the equation 1. The t_x is the last time that agent X has sent message to its neighbors. As it is visible if neighbors do not send a message for a while the probability that agent X changes its value will increase. This is logical and prevents the protocol from getting stuck in situation that agents wait for neighbors to move and actually no agent moves. In algorithms like DSA some variables does not have the chance to change their values and other agents may have higher chance all the time. This problem does not exists in the mDPP as an agent that for a long time has not send any message will have the chance to send message provided it has a good improvement value.

$$P(Y \leq x | improve_x^t = i) = \sum_{j=0}^{x} P(Y = j | improve_Y^t = i) \tag{2}$$

$$\prod_{Y_i=Y_1}^{Y_m} 1 - (P(Y_i \leq x | improve_x^t = i))^{t_x} \tag{3}$$

4 TEST RESULTS

In this section, DBA, MGM, DSA and mDPP are tested to solve DCSP problems in three different scenarios and it is shown that mDPP works well. First we tested DCSP problems in which there is no message loss or delay issues. To better understand the performance of algorithms, many problems with different variables and densities including low, medium and high are generated. For example, with 200 variables and a degree of 2, the density will be 0.0201005 and a with degree of 2.3, the density will be 0.02311. The degrees 2.0, 2.3 and 2.7 are used to demonstrate low, medium and high densities respectively.

Second, we simulated a situation in which there is a probability of the message loss. Different probabilities are used to have a better vision of the algorithms' behaviour in highly degraded environments. Third, we examined the algorithms in cases in which there exists some random network delays. In many real problems, the network speed is a big issue because this may cause delays in the communication. We simulated this issue and examined the performance of the algorithms with different network speeds.

All scenarios are tested 100 times and the average result is used. Then convergence rate and point of each algorithm for each test case is provided and the results summarized in different tables. As shown by Mailler [9] the convergence rate is important in analysing the dynamic problems. In addition, the result of the test cases for 200 variables and density of 0.023100 (degree 2.3) are shown in some figure. Each test case is generated randomly as follows. First, all variables and their domain values are created. Second, two variables are chosen randomly. Third, one random domain values of each of these two variables are selected and the the constraint is set to false(Provided no same constraint exists). This process continues until all constraints are created.

4.1 No message loss result

Mailler [4] has compared the DPP protocol with DSA and DBA for graph coloring problems. He showed that DPP is very competitive with DSA and DBA. However, that comparison is limited to simple graph coloring problems. Many real problems are more complicated than the graph coloring problems. In this section, we show result of solving DCSP

problems with mDPP, DSA, DBA and MGM. The parameters of this scenarios are as follows:

- $n = \{100, 200, 400\}$ that n is number of variables.
- $Degree = \{2.0, 2.3, 2.7\}$.
- $Tightness = \{0.33\}$.

Figure 4 and 5 show the convergence rate and point of these algorithms with degree 2.3 (Medium) and 200 variables. To better understanding of the algorithms performance the message loss result showed with message loss and delay. comparison of convergence rate and point of algorithms for problems with 200 variables and degree 2.3(Medium) . In addition the result of these algorithms for different densities and variables size are summarized in tables 1 and 2. As the results show DPP has the best convergence rate and point in almost all the test cases. In some cases MGM had lower convergence rate but higher convergence points. One reason that MGM has lower convergence rate in comparison to the others is that it does not have a mechanism to scape from the local minimum therefore it cannot progress and converges quickly to a non-optimal solution. DSA was also very effective for DCSP problems in an ideal environment.

4.2 Message Loss

In this section, we talk about the performance of discussing algorithms in case of message loss. The method for simulating message loss is as follows. A random value is generated for every message that each agent sends. If that number is greater than the loss probability the corruption flag of message is set to 1. The simulator router simply drops the messages with this flag. The parameters of this scenario are as follow.

- $n = \{200\}$ that n is number of variables.
- $Degree = \{2.0, 2.3, 2.7\}$.
- $Tightness = \{0.33\}$.
- $Timeout = \{2, 4\}$.

To see the performance of each algorithm different probabilities are used including 10, 25 and 50 percent. These can demonstrate the behavior of the algorithms in the case of small and high message loss probabilities. Based on the nature of DBA and MGM they need all agents to get messages from all their neighbors. Therefore, if any messages get lost they cannot escape form deadlock. Therefore some mechanisms must be applied to prevent this issue. The mechanism used here is to set a timeout for these algorithms in a way that after timeout was reached agents will send new messages although they may have not received all required messages. To see how timeout influences the performance of DBA and MGM two timeouts values are chosen and they are 2 cycles and 4 cycles.

The comparison of convergence rate and point for problems with 200 variables and degree 2.3 are shown in figure 4. In addition, we tested these algorithms with degree 2(low) and 2.7(high) and results are shown in tables 3 and 4. As result shows DPP has smallest convergence points for degrees 2 and 2.3. However DBA has better result for degree 2.7. The performance of MGM decreases sharply with 4 cycles timeout and it has the worst results in this case. However with 2 cycles,

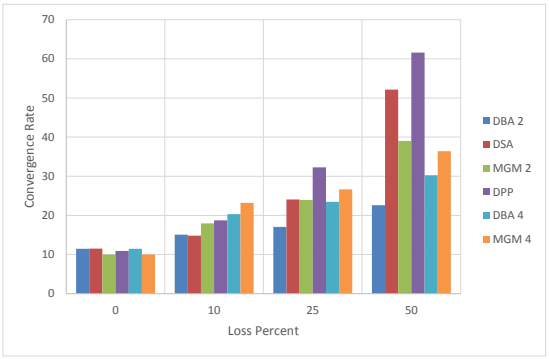

Figure 5: Convergence Rate For 200 Variables With Medium Density and Different Message Loss Probabilities

it outperforms DSA in general. One possible reason DBA and MGM had acceptable results with 2 cycles is that in each cycle all agents send messages to all of their neighbors even though some messages may not be delivered. In addition, in algorithms like DBA and MGM all messages have the same value therefore loss on some of messages do not impact a large on the solution. However, in DSA and more in DPP, messages may have different values and therefore messages lose may have more impact on the convergence rate. However, as the results show mDPP is a smart protocol in which an agent can predict the behaviour of its neighbor therefore even losing some important messages does not significantly decrease the performance of the algorithm. Another issue in algorithms like DBA and MGM is deciding the timeout value. It needs some analysis on the network to get the proper timeout value. In many real applications , many network do not show a regular behavior therefore a fix timeout value is not suitable. As results show both DBA and MGM have poor results with 4 cycles timeout compared with 2 cycles. It is predictable that with higher timeout the result will be worse. One may suggest a dynamic timeout value but even setting a dynamic value is difficult. In addition to all these problems, some messages may arrive after timeout therefore

Table 1: Convergence Rate (No Communication Loss or Delay)

Algorithm	Density								
	Low			Medium			High		
	Variables Count								
	100	200	400	100	200	400	100	200	400
mDPP	11.01	10.95	10.86	11.29	10.93	11.17	11.64	11.7	11.72
DSA	10.65	10.6	10.84	11.42	11.52	11.60	12.33	12.8	12.35
DBA	11.028	11.44	11.26	11.12	11.5	11.37	11.12	11.37	11.42
MGM	9.26	9.41	9.34	10.06	10.03	10.07	10.9	10.99	10.95

Table 2: Convergence Point (No Communication Loss or Delay)

Algorithm	Density								
	Low			Medium			High		
	Variables Count								
	100	200	400	100	200	400	100	200	400
mDPP	5.06	10.50	21.68	7.22	14.42	28.39	11.55	22.94	45.05
DSA	5.36	10.44	21.27	8.43	16.92	32.90	12.62	25.13	50.51
DBA	5.69	12.59	25.43	11.18	22.35	46.08	19.82	39.34	78.52
MGM	10.54	20.85	41.92	14.77	23.17	58.46	20.8	41.8	82.09

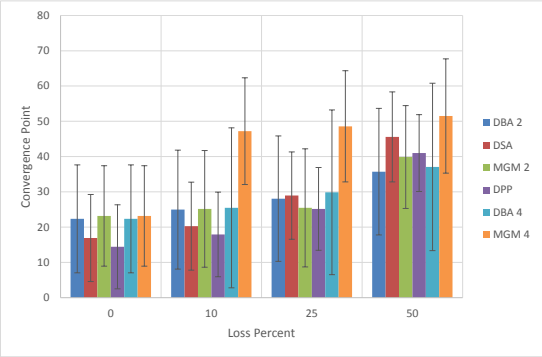

Figure 6: Convergence points For 200 Variables With Medium Density And Different Message Loss Probabilities

agents will not know how to behave with these messages. Simply dropping these messages may result in losing good information. On the other hand, using these messages may conflicts with previous or new messages. This may descrease the performance of DBA and MGM algorithms.

4.3 Communication Delay

In this section, we analysed the performance and behaviour of the algorithms in the case of network delay. In many real problems, it is difficult to provide a high speed network without any issues in delivering messages. We simulated an environment in which maximum number of messages that can be sent in each cycle is determined and in each cycle a random number of messages will be delivered and the the remaining messages will be delivered in the next cycles. For each agent number of messages it can sends is a random value between zero and the maximum number of messages that can be sent. The simulation process is as follows. Based on the size of each message for each algorithm and the average length of each cycle of simulator, we calculated the maximum number of messages can be sent in each cycle. Therefore, for any network speed it is possible to calculate the maximum number of messages that can be sent. Algorithms are tested with network speeds 1 Mbps and 2 Mbps for 200 variables and the different degrees. The parameters of this scenario are as follows:

- $n = \{200\}$ that n is number of variables.
- $Degree = \{2.0, 2.3, 2.7\}$.
- $Tightness = \{0.33\}$.
- $Timeout = \{2\}$.
- $NetworkSpeed = \{1Mb, 2Mb\}$.

Figures 6 and 7 shows the convergence rate and point of discussing algorithms for degree 2.3 and 200 variables.

Table 3: Convergence Rate With Communication Loss

Algorithm	Density								
	Low			Meduim			High		
	Message Loss Probability								
	10	25	50	10	25	50	10	25	50
mDPP	17.09	34.35	50.43	18.73	32.29	61.61	18.41	34.88	60.18
DSA	14.73	21.27	48.77	14.81	24.08	52.13	16.81	26.73	63.28
DBA2	15.34	16.93	22.8	15.09	17.06	22.6	15.03	16.66	20.35
MGM2	16.06	22.4	35.73	17.96	23.94	39.02	19.81	25.99	37.84
DBA4	20.01	22.21	29.30	20.32	23.46	30.28	21.58	23.8	30.9
MGM4	20.32	24.92	34.39	23.22	26.66	36.4	23.14	28.38	39.79

Table 4: Convergence Point With Communication Loss

Algorithm	Density								
	Low			Medium			High		
	Message Loss Probability								
	10	25	50	10	25	50	10	25	50
mDPP	12.23	17.75	31.47	17.93	25.17	40.98	27.57	37.09	57.65
DSA	20.43	21.82	36.54	20.28	28.94	45.59	30.25	38.95	58.84
DBA2	14.76	17.18	22.98	24.96	28.08	35.73	41.25	45.63	56.28
MGM2	15.77	19	31.43	22.18	25.47	39.88	32.1	36.47	50.69
DBA4	15.68	19.37	24.31	25.47	29.88	37.07	39.72	45.44	55.34
MGM4	37.08	39.59	43.61	47.21	48.59	51.51	63.01	62.98	64.98

Tables 5 and 6 provide the convergence rate and point information respectively for mentioned parameters. The results show DPP and DSA have better results but DBA and MGM do not work well. This is reasonable because algorithms like DBA and MGM depend highly on the messages and any delay in sending and receiving messages cause these algorithms to converge late. Therefore, with more delay or lower speed their result become worse. We set the timeout to 2 cycles for DBA and MGM. It is obvious that with higher timeout their performances will be decreased.

5 CONCLUSIONS

In this study, we analysed the performance of some of the most popular distributed algorithms in highly degraded environments. We showed how message loss and delay in communication impact the performance of these algorithms. Algorithms like MGM are shown not to be a good candidate for these situations. DBA was better than MGM in almost all scenarios however, it showed poor results in cases when there were delays in communication. In addition, its convergence rate increases when the timeout is set to 4 cycles. This means that both MGM and DBA are dependent on the timeout setting in this case. The mDPP and DSA were indicated ti be good solutions for distributed problems in cases of degraded environments. The mDPP, with few exceptions outperformed all other algorithms in all scenarios tested making it a very robust protocol for DCSP problems. One important result of this research is that algorithms that depend on frequent messages are not a good choice for degraded environments. Instead algorithms like mDPP that each message may have different value and the chance of sending a message from an agent may be different are better candidates for these problems. Although, the DCOP problems have not been discussed in this study, we expect the mDPP will work well on DCOP problems. In future work, we will analyse the performance of the most popular distributed algorithms especially mDPP in DCOP problems in highly degraded environments.

REFERENCES

[1] A. Petcu A. Farinelli, A. Rogers and N. R. Jennings. 2008. Decentralised coordination of low-power embedded devices using the max-sum algorithm. *in AAMAS-08* (2008), 639–646.

[2] F. Enembreck A. R. Leite and J.-P. A. Barthes. 2014. Distributed constraint optimization problems: Review and perspectives. *Expert Systems with Applications* (2014), 5139–5157.

[3] Wahbi M., Ezzahir R., Bessiere C., Bouyakhf, and E.H. 2013. Nogood-Based Asynchronous Forward-Checking Algorithms. *Constraints 18(3)* (2013), 404433.

[4] Roger Mailler. 2006. Using Prior Knowledge to Improve Distributed Hill Climbing. *Proceedings of the IEEE/WIC/ACM international conference on Intelligent Agent Technology* (December 2006), 514–521.

[5] Toru Ishida Makoto Yokoo, Edmund H. Durfee and Kazuhiro Kuwabara. 1992. Distributed constraint satisfaction for formalizing distributed problem solving. *In International Conference on Distributed Computing Systems* (1992), 614–621.

[6] P.J. Modi, S.M. Ali, W. Shen, and M. Tambe. 2003. Distributed constraint reasoning under unreliable communication. *Proceedings of Distributed Constraint Reasoning Workshop at Second International Joint Conference on Autonomous Agents and MultiAgent Systems* (2003).

[7] Kenneth N. Brown Mohamed Wahbi. 2014. The Impact of Wireless Communication on Distributed Constraint Satisfaction. *In Proceedings of the 20th International Conference on Principles and Practice of Constraint Programming* (2014), 738–754.

Table 5: Convergence Rate With Communication Delay

Algorithm	Density					
	0.0201005		0.02311		0.02713	
	Network Speed					
	1 Mb	2 Mb	1 Mb	2 Mb	1 Mb	2 Mb
mDPP	30.39	13.93	30.58	13.84	34.10	14.45
DSA	13.56	11.2	14.77	12.55	17.54	13.52
DBA	16.54	14.90	16.77	15.25	16.23	14.81
MGM	21.77	15.21	25.45	17.48	30.03	20.93

Table 6: Convergence Point With Communication Delay

Algorithm	Density					
	Low		Medium		High	
	Network Speed					
	1 Mb	2 Mb	1 Mb	2 Mb	1 Mb	2 Mb
mDPP	16.12	11.15	22.82	15.46	36.45	24.2
DSA	10.11	10.52	23.46	16.37	24.48	24.82
DBA	17.72	14.61	29.22	24.7	47.88	42.06
MGM	19.99	18.55	27.25	26.34	39.85	36.36

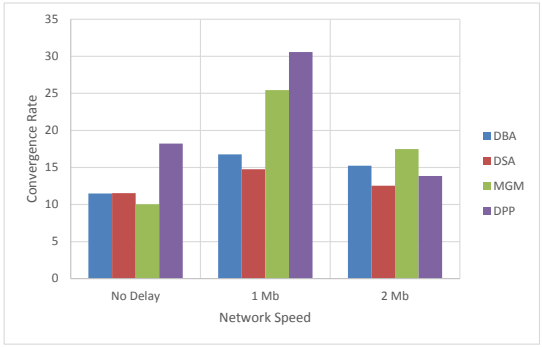

Figure 7: Convergence Rate For 200 Variables With Medium Density In Case of Message Delay

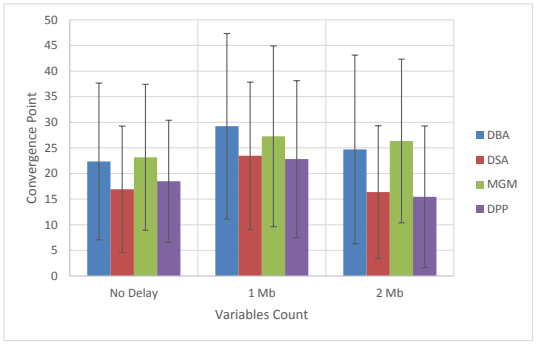

Figure 8: Convergence points For 200 Variables With Medium Density In Case of Message Delay

[8] J. P. Pearce R. T. Maheswaran and M. Tambe. D. 2004. Distributed algorithms for DCOP: A graphical-game-based approach. *In Proc. Parallel and Distributed Computing Systems PDCS* (September 2004), 432439.

[9] Anton Ridgway and Roger Mailler. 2015. Dynamic Theoretical Analysis of the Distributed Stochastic and Distributed Breakout Algorithms. *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent System* (May 2015).

[10] G. Wang W. Zhang and L.Wittenburg. 2002. Distributed stochastic search for constraint satisfaction and optimization:Parallelism, phase transitions and performance. *In proceedings of the AAAI workshop on probabilistic Approaches in Search* (2002), 53–59.

[11] M. Yokoo and K.Hirayama. 1996. Distributed Breakout algorithm for solving distributed constraint satisfaction problems. *International Conference on Multi-Agent Systems(ICMAS)* (1996).