An Efficient Algorithm For Solving Dynamic Complex DCOP Problems

2009 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Workshops

An Efficient Algorithm For Solving Dynamic Complex DCOP Problems

Sankalp Khanna*†, Abdul Sattar*, David Hansen† and Bela Stantic*
*Institute for Integrated and Intelligent Systems
Griffith University, QLD 4111. Australia
Email: S.Khanna@griffith.edu.au, A.Sattar@griffith.edu.au, B.Stantic@griffith.edu.au
†The Australian e-Health Research Centre
71/918,RBWH, Herston, QLD 4029. Australia
Email: David.Hansen@csiro.au

***Abstract*—Multi Agent Systems and the Distributed Constraint Optimization Problem (DCOP) formalism offer several asynchronous and optimal algorithms for solving naturally distributed optimization problems efficiently. There has been good application of this technology in addressing real world problems in areas like Sensor Networks and Meeting Scheduling. Most of these algorithms however exploit static tree structures and are thus not well suited to modeling and solving problems in rapidly changing domains. Also, while in theory most DCOP algorithms can be extended to handle complex local sub-problems, we argue that this generally results in making their performance sub-optimal, and thus their application less suitable. In this paper we present new measures that emphasize the interconnectedness between each agent's local and inter-agent sub-problems and use these measures to guide dynamic agent ordering during distributed constraint reasoning. The resulting algorithm, DCDCOP, offers a robust, flexible, and efficient mechanism for modeling and solving dynamic complex problems. Experimental evaluation of the algorithm shows that DCDCOP significantly outperforms ADOPT, the gold standard in search-based DCOP algorithms.**

I. Introduction

Despite being a relatively young research area, with the first asynchronous Distributed Constraint Satisfaction Problem (DisCSP) algorithm proposed in 1992 [1], and the first complete Distributed Constraint Optimization Problem (DCOP) algorithm, ADOPT, proposed in 2003 [2], the Distributed Constraint Reasoning formalism has developed rapidly to offer efficient and sophisticated algorithms to model and solve a variety of naturally distributed multi-agent problems. Several notable DCOP approaches employing techniques from search (e.g. ADOPT and its several variants), dynamic programming (e.g. DPOP [3] and its several variants) and cooperative mediation (e.g. APO [4]) have emerged and are being successfully used to model and solve problems in sensor networks, meeting scheduling, etc.

This research was motivated by our effort to model and solve naturally distributed complex optimization problems in the health domain, a typical example being the scheduling of elective surgery in a large public hospital. The problem involves several departments, each with its own complex scheduling problem. The departments need to negotiate with each other to build, and maintain, the elective surgery schedule in the face of a dynamic health landscape. Allocation of airport slots to airlines, or public infrastructure to utilities companies, are examples of similar problems in relatively dynamic environments where several agents, each with complex sub-problems, are negotiating in a privacy preserving manner, to optimize a common cost function.

Working towards this aim, the first natural observation is that most current algorithms are based on tree structures, which are static in nature and would continually need to be rebuilt in dynamic environments. Also, given the nature of the problem domain, partial centralization based strategies would not be a good fit here because of obvious privacy and decision control concerns. We also note that the metrics used to compare algorithms are questioned by most researchers. Silaghi and Yokoo [5] have shown that it is possible to construct problems that can be exploited by algorithms such as ADOPT and DPOP to exhibit their superiority. Also, Maheswaran et al. [6] show that the performance of ADOPT in solving real world problems is significantly worse than in solving similar-sized map coloring problems.

We draw from Zhou's work [7] in the DisCSP field and generalize the novel measures of constraint density (not to be confused with the traditional measure, i.e. ratio of actual number of constraints to possible number of constraints), to introduce new DCOP measures of Dynamic Cost Density (DCD) and Degree of Unsatisfaction (DU), and then use these measures to dynamically guide agent ordering in our new Dynamic Complex Distributed Constraint Optimization Problem (DCDCOP) algorithm. We compare DCDCOP's performance to ADOPT, the current standard in DCOP search, and show that the new algorithm offers a significant improvement over ADOPT.

The rest of this paper is organized as follows. In section II, we describe our case study of the elective surgery scheduling problem at a large Australian public hospital. Section III presents our proposed Multi-Agent System (MAS) architecture for modeling and solving this class of problems. Section IV follows with a discussion of the DCOP formalism and the shortcomings of the current state of the art in DCOP algorithms in addressing real world dynamic complex problems. In section V, we present the new measures of DCD

978-0-7695-3801-3/09 $26.00 © 2009 IEEE
DOI 10.1109/WI-IAT.2009.175

and DU, and the DCDCOP algorithm, and explain these with a simple example. Section VI reports on the empirical evaluation of the algorithm as compared to ADOPT. Lastly, in section VII, we present our main conclusions and discuss ongoing work.

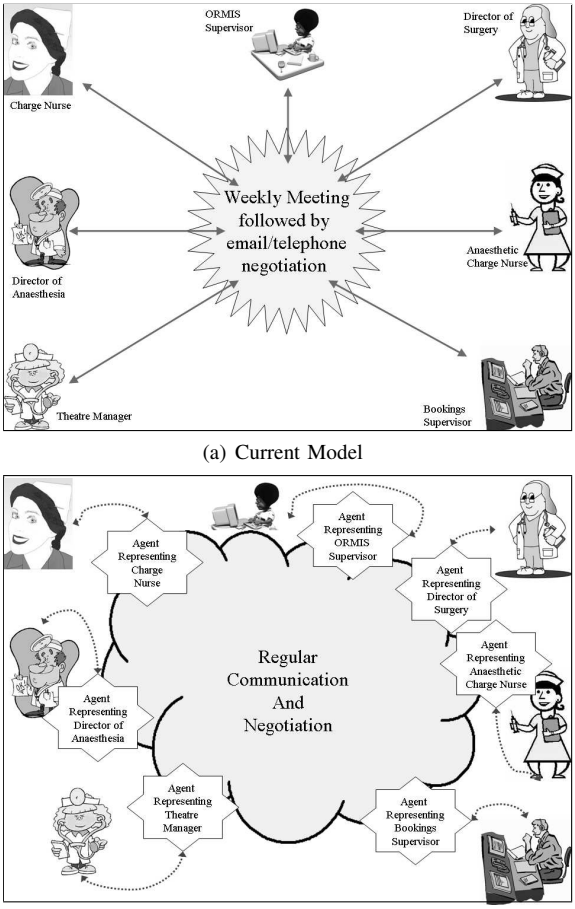
II. Scheduling Elective Surgery at the Princess Alexandra Hospital - A Case Study

Elective surgery is a planned, non-emergency surgical procedure, which can be scheduled at the patient's and surgeon's convenience. The escalating demand for elective surgery is however compounded by a shortage of trained surgeons, anaesthetists and nurses. Recent Queensland statistics show that, as of 1 April 2009, 33,993 patients were waiting for elective surgery of which almost 18% had waited longer than a clinically desirable time [8].

We conducted an extensive study of scheduling processes followed at the Princess Alexandra Hospital (PAH), a large public hospital in Queensland's capital city of Brisbane. PAH offers 21 operating theaters that can be utilized by various departments for elective surgery. For the process of scheduling, the theater schedule is divided into AM and PM slots of 3.5 hours each. These slots are allocated to various doctors or departments but can be utilized for trauma or emergency if urgently required.

Each department connected (i.e. allocating staff or other resources) to the surgery carries out their individual scheduling activity. The bookings department books patients into the *Bookings Schedule* in consultation with the relevant surgical teams, and records these bookings into the *Operating Room Management Information System* (ORMIS). The different departments can access this information by looking into ORMIS or accessing the latest *Bookings Schedule* on the shared drive, where it is updated everyday at 3PM.

Every Thursday, the managers of the different departments meet and review bookings for the week ahead (Figure II). Each session is discussed and conflicts in the departmental schedules are worked out by negotiation. Unexpected emergencies, variation in patients' health state, sudden perturbations in staffing, and surgeon availability etc. lead to further changes being often required. However, all changes made to the system after this meeting are dealt with individually by the departments, and resolved on a case-by-case basis using conventional communication such as telephone and emails, or even by face-to-face meetings. In keeping with the dynamics of the domain, the schedule needs to be updated quickly and efficiently. This is often not possible, because of delays in inter-departmental communication, and this leads to the adoption of an easy but inefficient solution resulting in a compromised schedule. For example, if a procedure is canceled at the last minute, because new medical reports say it is no longer required, the bookings department would want to offer the slot to another procedure. They may however be unable to confirm availability of specialist staff or equipment, because the charge nurse or theater manager were temporarily unavailable, which would lead to the slot being unused.



(a) Current Model

(b) Proposed Model

Figure 1. Scheduling Elective Surgery at the Princess Alexandra Hospital

III. Proposed MAS Architecture

Given the naturally distributed nature of problems like elective surgery scheduling, an intelligent agent based approach is a promising paradigm for modeling the environment. We propose a methodology where intelligent agents, trained with the constraints, preferences, priorities etc. of the administrators, optimize schedules for their respective departments (Figure II). They then negotiate in a privacy-preserving manner (i.e. without sharing more information than is essential) to resolve inter-agent constraints. The architecture of each agent (see Figure 2) incorporates an interface module to handle internal and external communication, an intelligence module to handle decision making and learning, and a DCOP engine to drive the optimization.

The agents thus incorporate learning from domain-expert interaction, and have the ability to use intelligent reasoning to translate environmental changes and negotiation requests

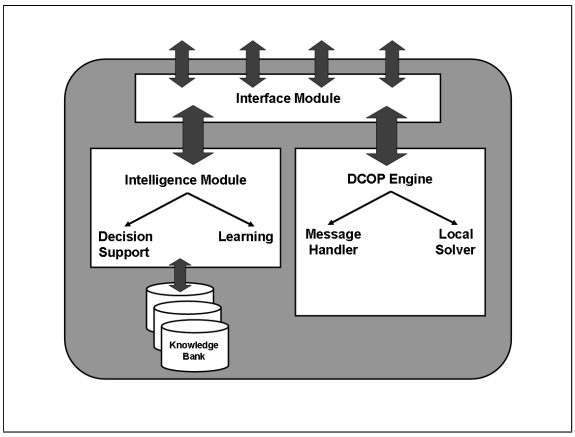


Figure 2. Proposed Architecture for Agent

to constraints for the DCOP engine. Using an appropriate DCOP algorithm the agents optimize their local solution, and then collaborate with each other to resolve inter-agent constraints and align themselves.

The DCOP algorithm thus needs to be robust enough to handle the complexity of each agent's sub-problems and perform inter-agent negotiation in a truly distributed manner, efficient enough to solve the problem in a timely and optimal manner, and flexible enough to adapt to the dynamics of the environment. Additionally, by shielding the local solver mechanism from the inter-agent negotiation process, each agent can use a local solver strategy to suit its own need.

IV. Related Work

We start with discussing the DCOP formalism and some of the current approaches to solving DCOPs. We then revisit real world dynamic complex problems and discuss the shortcomings of current approaches in addressing these.

A. The DCOP formalism

In solving a DCOP, the goal for each agent is to assign values to its variables such that a given global objective function is minimized. The cost functions in DCOP are analogous to constraints in DisCSP, and DCOP is thus regarded as a generalization of the DisCSP formalism. For simplicity, we use the term constraints and cost functions interchangeably. Formally, we can define a DCOP as consisting of:

1) A finite ordered set of Agents $A = \{A_1, A_2, A_3, ..., A_n | n \in Z^+\}$, where, for each Agent A there exists :
   a) A finite ordered set of variables $V = \{V_1, V_2, ..., V_n | n \in Z^+\}$,
   b) A domain set $D = \{D_1, D_2, ..., D_n\}$, containing a finite and discrete domain D_i for each V_i,
   c) A constraint set $C = \{C_1, C_2, ..., C_m\}, m \in Z^+$, where each $C_j, \forall j \in [1, m]$ is defined as a cost function on a pair of variables, $f_{i,i'} : D_i D_{i'} \rightarrow N, \forall V_i, V_{i'} \in V$, and
   d) An ordered solution set $S = \{v_1, v_2, v_3, ..., v_n | v_i \in D_i, \forall i \in [1, n]\}$ where the aggregate cost $F(A) = \sum_{(x_i, x_{i'} \in V)} f_{ii'}(d_i, d_{i'}), x_i \leftarrow d_i, x_{i'} \leftarrow d_{i'} \in A$.
2) The solution set of the DCOP $S^\star$ is defined as the set of the solution sets of each agent.

In keeping with the norm, it is assumed that all constraints are binary, and optimization functions are associative, commutative, and monotonic [2]. In dealing with complex DCOPs however, we do not make the general assumption of one variable per agent.

B. Current State of the Art

Several DCOP algorithms and their variants have been introduced by recent research. Due to space constraints, we focus on the following key algorithms, each representing a significantly different approach. Also, we do not critique each variant as they all still have the same shortcomings when applied to dynamic complex problems.

Asynchronous Distributed OPTimization, or ADOPT [2], is a complete and asynchronous DCOP algorithm. In ADOPT, agents are first prioritized into a Depth First Search (DFS) tree, whereby each agent maintains lower and upper bounds for the subtree rooted at their node. The agents then use opportunistic best-first search to assign their variables such that the lower bound is minimized. *Cost* messages propagate up the tree and *threshold* and *value* messages are sent down the tree, iteratively tightening the lower and upper bounds until the lower bound of the minimum cost solution is equal to its upper bound. If an agent detects this condition, and its parent has terminated, then an optimal solution is found and it may also terminate. The other key idea in ADOPT is to store lower bounds as a threshold and discard partial solutions before they are proven to be definitely suboptimal, thus maintaining linear space complexity at each agent. In the worst case, ADOPT may require an exponential number of messages to arrive at a solution.

Distributed Pseudotree Optimization Procedure, or DPOP [3], is a complete dynamic programming algorithm that involves a three phase process. Similar to ADOPT, the first phase involves the formation of the DFS tree. Phase two involves calculating and propagating the utility (cost) bottom-up, i.e. from the leaves upwards to the root. Phase three involves a downward value propagation, initiated by the root node. Each agent then calculates its optimal value based on the utility message received from its subtree and the value message received from its parent. DPOP thus generates only a linear number of messages, but the message size grows with every traversal up the tree and the algorithm thus requires large amounts of memory, up to space exponential in the induced width of the problem.

Optimal Asynchronous Partial Overlay (OptAPO) [4] is an alternative approach to DCOP that utilizes partial centralization to solve difficult portions of a DCOP problem. While partial centralization offers an excellent mechanism to solve DCOPs in several scenarios, it is generally unsuitable in the kind of problems we seek to address. In most such cases, negotiating agents would normally refuse to share information or relinquish decision making control of their private sub-problems.

C. Solving Dynamic Complex Problems

Since both ADOPT and DPOP utilize static DFS tree structures, changes to the constraints would often result in the need for the tree to be rebuilt. Also, since ADOPT discards no-goods to maintain linear space complexity, changes to the constraints would result in bounds being discarded and the search restarted.

Both ADOPT and DPOP also offer variants to deal with dynamic environments. Modi [9] offers a formalism for mapping and solving dynamic resource allocation problems but this is applied in the DisCSP domain. This is extended to map over-constrained problems into DCOP but can handle only static problems as the author concedes to the lack of an effective DCOP algorithm for dynamic problems. Petcu et al. [10][11] propose S-DPOP and RS-DPOP, which utilize self stabilizing DFS trees to guarantees optimal solution stability in distributed continuous-time combinatorial optimization problems. Lass et al. [12] deal with the *complicating factor of dynamism* by wrapping ADOPT in an *Adapter* that receives and handles dynamic event requests.

In dealing with the issue of complex sub-problems, algorithms can theoretically utilize decomposition or compilation. In practice however, decomposition results in failure to exploit the inherent benefit of domain centralization, and also blows the distributed problem size out of proportion. Burke and Brown [13] show that the compilation outperforms decomposition in case of large local sub problems but only small domain size, whereas decomposition is more appropriate when the number of inter-agent constraints and domain size is large but only for small problems. Several ADOPT variants use techniques such as decomposition [9], compilation [14], interleaving and relaxation [15], to name a few, to deal with complex sub-problems. All of the above variants however still suffer from working off a static tree structure that needs rebuilding from time to time.

Further, applying decomposition to DPOP would result in a significant increase in the message sizes, while compilation would need a novel mechanism of calculating the agent utility for different combinations of local variable assignments. It would however be interesting to evaluate the effect of utilizing our measure of DU as the utility metric on DPOP.

We thus conclude that while the above DCOP algorithms are optimal in a static environment, there is need for a more flexible robust algorithm, which can model the complexity and adapt better to a dynamic environment.

V. The DCDCOP Algorithm

We present the new measures of DCD and DU and a new Dynamic Complex Distributed Constraint Optimization Problem (DCDCOP) algorithm and explain these measures, and algorithm execution, with a simple example.

A. Defining DCD and DU

Zhou [7] offers innovative measures of Dynamic Constraint Density (DCD) and Degree of Unsatisfaction (DU) (a measure of how far an agent's instantiation is from reaching a consistent state) and presents algorithms based on these to handle complex and dynamic constraint reasoning problems efficiently. These however are suited only to satisfaction and constraint relaxation approaches as the measure does not take varying constraint costs into account.

We generalize the definitions of Zhou to define the following new static measures of Intra-Agent Cost Density ($IACD$) and Inter-Agent Cost Density (I_ACD) :

$$IACD_i = \begin{cases} 0, & \text{if } |intraV_i|=0 \\ \frac{\sum_{j=1}^{|intraC_i|}\left(\delta_m\left(intraC_i^j\right)\right)}{|intraV_i|}, & \text{otherwise} \end{cases} \quad (1)$$

$$I\_ACD_i = \begin{cases} 0, & \text{if } |interV_i|=0 \\ \frac{\sum_{j=1}^{|interC_i|}\left(\delta_m\left(interC_i^j\right)+\sum_{l=1}^{|\kappa C_i^j|}\left(\delta_m\left(intraC_i^l\right)\right)\right)}{|interV_i|}, & \text{otherwise} \end{cases} \quad (2)$$

where $intraC_i$ is the set of intra-agent constraints for agent i, δ_m represents the maximum cost of the constraint, $intraC_i^j$ is the j^{th} intra-agent constraint of agent i, $intraV_i$ is the set of variables constrained by $intraC_i$, $interC_i$ is the set of inter-agent constraints for agent i, $interC_i^j$ is the j^{th} inter-agent constraint of agent i, κC_i^j is the set of intra-agent constraints belonging to i and connected to $interC_i^j$, and $interV_i$ is the set of variables constrained by $interC_i$ and controlled by agent i.

The measure of I_ACD takes into account the interconnectedness of the variables that are attached to an inter-agent constraint. The higher the cost of intra-agent constraints attached to the variable, the greater the impact of the variable on the cost density. This is justified as changing the value of this variable would attract a much higher effort towards optimizing the problem. Also, the measures of $IACD$ and I_ACD both equate to zero if there are no intra-agent or inter-agent variables connected to constraints respectively. Thus no variables, and a consequent cost density of 0, would imply that this component of the problem does not need to be solved further. Using these measures, we can now define Static Cost Density (SCD).

Static Cost Density of an agent is defined as the sum of the maximum possible Intra-Agent and Inter-Agent Cost Densities.

$$SCD_i = IACD_i + I\_ACD_i \tag{3}$$

In equations 1 and 2 we replace δ_m by δ_c, which gives us the current cost of the constraint, to get our dynamic measures of IntraUnsat (IU), and InterUnsat (I_U), which represent the dynamic intra-agent and inter-agent cost densities respectively. We utilize these to define the measure of Dynamic Cost Density (DCD).

$$IU_i = \begin{cases} 0, & \text{if } |IACD_i|=0 \\ \frac{\sum_{j=1}^{|intraC_i|}\left(\delta_c\left(intraC_i^j\right)\right)}{|intraV_i|}, & \text{otherwise} \end{cases} \tag{4}$$

$$I\_U_i = \begin{cases} 0, & \text{if } |I\_ACD_i|=0 \\ \frac{\sum_{j=1}^{|interC_i|}\left(\delta_c\left(interC_i^j\right)+\sum_{l=1}^{|\kappa C_i^j|}\left(\delta_c\left(intraC_i^l\right)\right)\right)}{|interV_i|}, & \text{otherwise} \end{cases} \tag{5}$$

Dynamic Cost Density of an agent is defined as the sum of the current Intra-Agent and Inter-Agent Cost Densities.

$$DCD_i = IU_i + I\_U_i \tag{6}$$

We can now also redefine the measure of Degree of Unsatisfaction for agent i (DU_i).

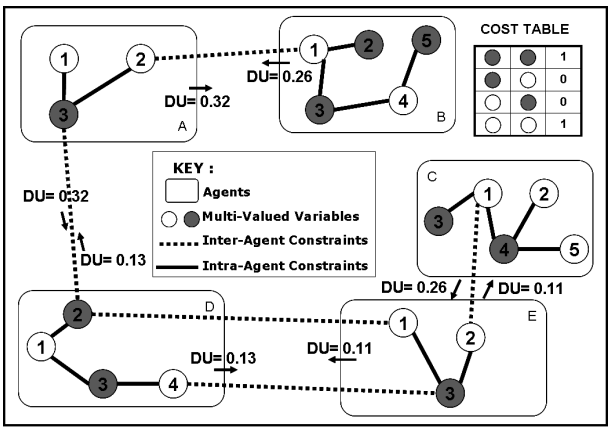
Degree of Unsatisfaction of an agent is defined as the ratio of the Dynamic(current) to Static Cost Densities.

$$DU_i = \frac{DCD_i}{SCD_i} \tag{7}$$

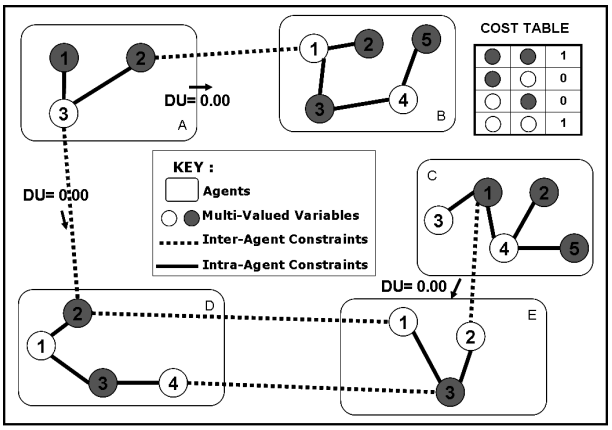
DU provides a measure of how far away an agent's current instantiation is from reaching an optimal state. It does not provide a direct measure to compare the level of complexity to two agents' problems or the time it may take to solve them. However, unlike a simple summation of max cost or current cost, it attaches a higher cost to the changing of more interconnected constraints, thus providing a more realistic measure of the complexity of the solution.

B. An Example of Calculating DU

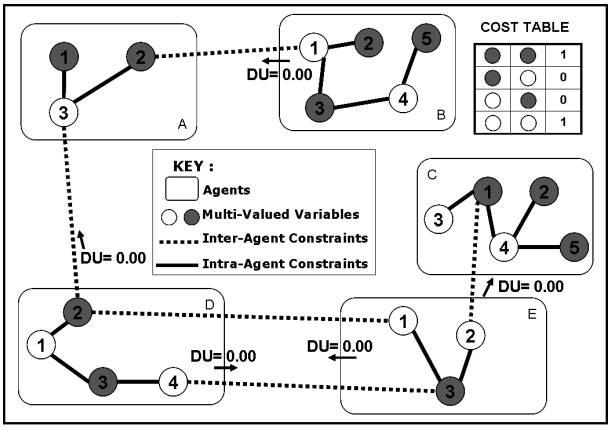
To better understand the above measures, we utilize a simple example as shown in Figure 3(a). Here, we calculate the values of SCD, DCD and DU for agent D, which has four variables, three intra-agent constraints and three inter-agent constraints. The max cost of each constraint (from the cost table shown in the figure) is 1. In calculating the static measures for the problem, we have:



(a) Step 1



(b) Step 2



(c) Step 3

Figure 3. Example of DCDCOP Execution

$$\begin{aligned} IACD_D &= \frac{(1+1+1)}{4} = 0.75 \\ I\_ACD_D &= \frac{((1+(1))+(1+(1))+(1+(1)))}{2} = 3 \\ SCD_D &= 0.75+3 = 3.75 \end{aligned}$$

Now, assuming a snapshot view of the scenario, where each agent knows each other's instantiation, we can calculate the dynamic measures:

$$\begin{aligned} IU_D &= \frac{(0+0+0)}{4} = 0 \\ I\_U_D &= \frac{((1+(0))+(0+(0))+(0+(0)))}{2} = 0.5 \\ DCD_D &= 0 + 0.5 = 0.5 \\ DU_D &= \frac{0.5}{3.75} = 0.13 \end{aligned}$$

Note that the constraint between variables 1 and 2 of agent D is counted twice in the calculation of I_ACD_D and I_U_D. The calculated value of $DU = 0.13$ will be sent to neighbors A and E. Other agents will similarly calculate and send their DU values to neighbors.

C. Details of Algorithm

The DCDCOP algorithm is implemented as follows :

- All agents start by calculating the values of $IACD$, I_ACD and SCD. Then they instantiate their local variables using a Branch and Bound algorithm [16], thus ensuring that the current cost is the minimum allowed by its current context. Each agent then calculates its dynamic measures of DCD and DU and sends DU and the related context to its neighbors (i.e. those agents with whom it shares a constraint).
- All agents start to receive on incoming links. When a message is received, its context is checked to ensure that it is compatible with the agent's $CurrentContext$. If not, the message is discarded and the agent continues to listen on incoming links. If compatible, the $messageContext$ is added to $CurrentContext$ and $messageDU$ is compared with the agent's DU. If DU is higher, the agent will reassign its variables, recalculate its dynamic measures and resend messages on outgoing links. If DU is lower than $messageDU$, it will not reassign its variables, but if relevant, it will recalculate its dynamic measures and resend messages on outgoing links. In the event that $messageDU = DU$, the agent with a higher predefined ordering will reassign its variables, recalculate its dynamic measures and send messages on outgoing links.
- The search stops when each agent has achieved a stable state and no more messages are transacted. In the case of a solvable problem, this happens when the agent, and all its neighbors, arrive at $DU = 0$. In the case of an optimization problem, this happens when the agent with a higher DU no longer changes its local solution as doing so would raise the cost of its solution.

The pseudo code of the algorithm is shown in Figure 4. Each negotiation is handled in one of the following ways: if the values of DU are not identical, the agent with a higher value of DU will change its value; if the values of DU are identical, the agents will follow a fixed predefined ordering between them to decide who changes their value. When an agent with a higher DU finds no better instantiation for its local variables, it will return the same, thus reaching a steady state until it receives a context message from other agents causing it to reevaluate its cost function. Note that the agent with a lower DU can force the agent with a higher DU to negotiate by raising the cost attached to the inter-agent constraint.

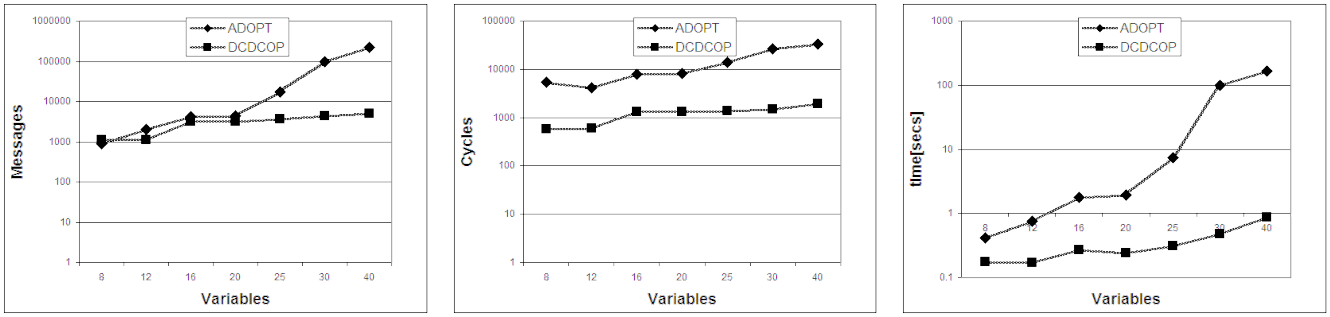
```
Calculate static measures
Solve_local_problem
Calculate dynamic measures
Send message (DU, CurrContext) to all neighbors
Receive messages
when received (messageDU, msgContext) do
    if msgContext and CurrContext are consistent
    then
        add msgContext to CurrContext
        if DU > msgDU then
            Solve_local_problem
        end
        else if DU = msgDU and
        higher_predefined_order then
            Solve_local_problem
        end
        Calculate dynamic measures
        Send message (DU, CurrContext) to all
        neighbors
    end
end do
Procedure : Solve_local_problem
Branch and Bound to solve local problem
```

Figure 4. The DCDCOP Algorithm

D. Example of DCDCOP Execution

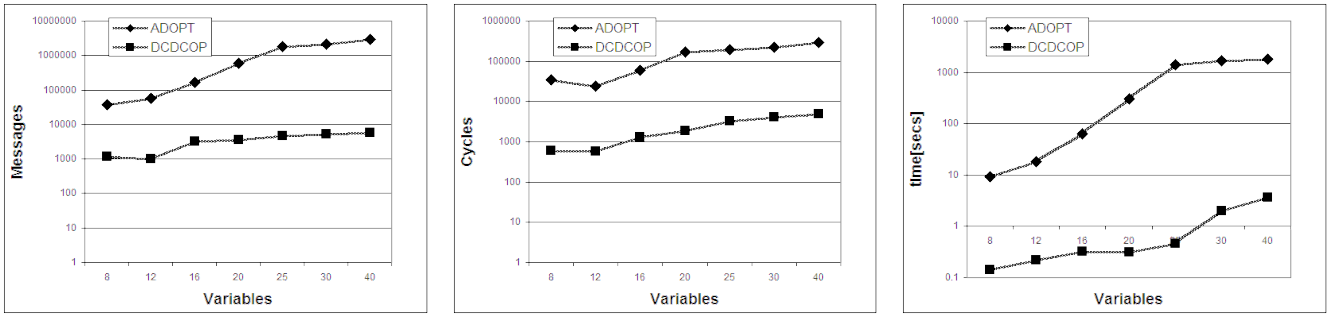
In continuing the example from section V-B, assuming agents A B and C are first to receive their messages, A and C will both reassign their variables and send out new messages, as they have received DU messages less than theirs. This results in the state shown in Figure 3(b).

Now assume D and E receive both messages together. Acting on the newer messages, both D and E will attempt to reassign their variables because their last calculated DU is higher than the $messageDU$ from A and C respectively, but with the updated values from A and C, both will arrive at the same local solution as being the lowest cost. Similarly, B will arrive at the same local solution as being the lowest cost and they will send out their DU messages (Figure 3(c)). Since all agents will detect that they have a DU of 0, the algorithm will terminate.



(a) Log(Number of Messages) (b) Log(Number of Cycles) (c) Log(Time in Seconds)

Figure 5. Performance of ADOPT vs DCDCOP ($LD = 2$)



(a) Log(Number of Messages) (b) Log(Number of Cycles) (c) Log(Time in Seconds)

Figure 6. Performance of ADOPT vs DCDCOP ($LD = 3$)

VI. Experimental Evaluation

Given the need for agents to negotiate in a privacy preserving manner, otherwise optimal approaches like OptAPO are not suitable in this context. And while DPOP and its variants are good DCOP algorithms, we choose ADOPT for our experimental evaluation because, like DCDCOP, it is based on constraint-guided search and thus provides a more realistic comparison. Further, ADOPT's source code can handle multiple variables per agent (using compilation) without any modification to the algorithm, and is thus well suited to a fair and accurate comparison. To further ensure a reasonable comparison, DCDCOP is developed within the original ADOPT source code [17] and evaluated on the same graph coloring problems that were used to report ADOPT's performance in [9] and come bundled with the source code. Also, ADOPT's original messaging and performance evaluation procedures are utilized for the evaluation.

The original graph coloring problem data (8-40 variables) is evenly distributed between 3-5 agents. Also as in the original evaluation, we analyze the performance of both algorithms on problems with link density (LD) of 2 and 3. The performance is compared using three measures: number of messages, number of concurrent cycles, and time(secs). To prevent a large disparity between the results, the algorithms are run with a maximum time, $timeMax$ of 30 mins. Also, given the large performance gain exhibited by DCDCOP, we employ a logarithmic scale for a meaningful display of results (Figures 5 and 6).

We observe that DCDCOP outperforms ADOPT significantly on all three scales of measurement. The speedup can be attributed partly to the algorithm exploiting domain centralization and performing each local reassignment within one cycle, and partly to the novel dynamic measures used to guide the inter-agent negotiation part of the algorithm.

The results allow for some extremely interesting observations. We observe that the difference in DCDCOP's performance for $LD = 2$ and $LD = 3$ is not as significant as in the case of ADOPT. This can be attributed to computational superiority over I/O speeds, the factor also responsible for DPOPs performance gains over ADOPT [3]. We also note that the performance of DCDCOP with 40 variables (in 5 agents), is reasonably similar that of ADOPT with 8 agents (and 1 variable per agent). Further, the performance of DCDCOP does not deteriorate much as we increase the problem size from 8 variables (in 3 agents) to 40 variables (in 5 agents). This further asserts the computational superiority of the centralized optimization algorithms such as Branch and Bound, and reinforces common belief that communication is the bottleneck in distributed problem solving.

We thus observe how problems that cannot be solved efficiently by DCOP algorithms can benefit greatly if there is

a component of domain centralization that they can exploit. Further, a flexible robust algorithm like DCDCOP can help model the departmental centralization structure of large real world optimization problems, not only offering a natural mapping from real world problem solving structure to DCOP problem solving structure, but also exploiting this to provide an order of magnitude improvement in performance.

VII. Conclusion

Several real world optimization problems translate to agents with complex sub-problems in a dynamic environment, that need to negotiate in a manner where privacy and decision making authority is preserved. The current state of the art in DCOP deals with such problems by offering extensions to algorithms best suited for optimization of single-variable agents in static environments and this often lead to sub-optimality. We present a flexible, robust algorithm, DCDCOP, that is capable of exploiting the inherent domain centralization found in such problems, and uses a novel measure, DU, to dynamically guide agent ordering during optimization. Experimental evaluation shows that DCDCOP significantly outperforms ADOPT, the current state of the art DCOP search algorithm.

We are currently developing realistic and significant benchmark problems based on scheduling process information and data collected during our extensive interviews and interactions with schedulers and domain experts at the PAH. Further evaluation of DCDCOP and other DCOP algorithms on these benchmarks, and investigating the use of the measure of DU to guide agent based negotiation in various DCOP algorithms, are proposed as future work.

Acknowledgment

We wish to thank Dr. Peter Moran and his colleagues at the PAH for allowing us into their world, and sharing their invaluable expertise with us over the last three years.

References

[1] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara, "Distributed constraint satisfaction for formalizing distributed problem solving," in *International Conference on Distributed Computing Systems*, 1992, pp. 614–621.

[2] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo, "An asynchronous complete method for distributed constraint optimization," in *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, Melbourne, 2003, pp. 161–168.

[3] A. Petcu and B. Faltings, "A scalable method for multiagent constraint optimization," in *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, Edinburgh, Scotland, Aug. 2005, pp. 266–271.

[4] R. Mailler and V. Lesser, "Solving Distributed Constraint Optimization Problems Using Cooperative Mediation," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*, New York, 2004, pp. 438–445.

[5] M. C. Silaghi and M. Yokoo, "DFS Ordering in Nogood-based Asynchronous Distributed Optimization (ADOPT-ng)," in *Sixth International Workshop on Distributed Constraint Reasoning*, Italy, 2006.

[6] R. T. Maheswaran, M. Tambe, E. Bowring, J. P. Pearce, and P. Varakantham, "Taking DCOP to the real world: Efficient complete solutions for distributed Multi-Event scheduling," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*, New York, 2004, pp. 310–317.

[7] L. Zhou, J. Thornton, and A. Sattar, "Dynamic Agent Ordering in Distributed Constraint Satisfaction Problems," in *Australian Conference on Artificial Intelligence*, 2003, pp. 427–439.

[8] Queensland Health, "Quarterly Public Hospitals Performance Report March Quarter 2009," http://www.health.qld.gov.au/surgical_access, 2009.

[9] P. J. Modi, "Distributed constraint optimization for multiagent systems," Ph.D. dissertation, University of Southern California, USA, 2003.

[10] A. Petcu and B. Faltings, "S-DPOP: superstabilizing, fault-containing multiagent combinatorial optimization," in *Proceedings of the Twentieth National Conference on Artificial Intelligence, AAAI-05*, Pittsburgh, Pennsylvania, USA, 2005, p. 449–454.

[11] A. Petcu and B. Faltings, "Optimal solution stability in dynamic, distributed constraint optimization," in *Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*. IEEE Computer Society, 2007, pp. 321–327.

[12] R. N. Lass, E. A. Sultanik, and W. C. Regli, "Dynamic distributed constraint reasoning," in *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, Chicago, 2008, pp. 1466–1469.

[13] D. A. Burke and K. N. Brown, "A comparison of approaches to handling complex local problems in DCOP," in *Sixth International Workshop on Distributed Constraint Reasoning*, Italy, 2006, p. 27–33.

[14] J. Davin and P. J. Modi, "Hierarchical variable ordering for distributed constraint optimization," in *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, Hakodate, Japan, 2006, pp. 1433–1435.

[15] D. A. Burke, "Exploiting problem structure in distributed constraint optimisation with complex local problems," Ph.D. dissertation, Department of Computer Science, University College Cork, Ireland, 2008.

[16] E. C. Freuder, "Partial constraint satisfaction," in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, USA*, 1989, p. 278–283.

[17] C. P. Portway, "USC DCOP Repository," http://teamcore.usc.edu/dcop, 2008.