# Effective Variants of Max-Sum Algorithm to Radar Coordination and Scheduling

Yoonheui Kim
University of Massachusetts at Amherst, MA 01003, USA
ykim@cs.umass.edu

Michael Krainin
University of Washington, Seattle, WA 98195
mkrainin@cs.washington.edu

Victor Lesser
University of Massachusetts at Amherst, MA 01003, USA
lesser@cs.umass.edu

## ABSTRACT

This work proposes new techniques for saving communication and computational resources when solving distributed constraint optimization problems using the Max-Sum algorithm in an environment where system hardware resources are clustered. Solving a coordination problem in a decentralized environment requires a large amount of resources and thus exploiting the innate system structure and external information as much as possible is necessary for such a problem to be solved in a computationally effective manner. These techniques facilitate effective problem solving through the use of a pre-computed policy and two phase propagation on Max-Sum algorithm, one inside the clustered resources and one among clustered resources. This approach shows equivalent quality to the standard Max-Sum algorithm while reducing communication requirements on average by 50% and computation resources by 5 to 30% depending on the specific problem instance. These experiments were performed in a realistic setting involving the scheduling of a network of as many as 192 radars in 48 clusters.

## Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Coherence and coordination

## General Terms

Algorithms, Performance

## Keywords

DCOP, Max-Sum, semi-centralized

## 1. INTRODUCTION

This paper focuses on utilizing semi-centralized (clustered) hardware system structure to solve the agent coordination problem in large-scale networks of agents. It proposes modifications on message-passing algorithms for reducing the required computation and communication resources for efficient solving. Decentralized coordination in large-scale systems of cooperative agents requires large computational and communication resources and often is not feasible in realistic problems requiring real-time deadlines: The decentralized computation of an optimal coordination policy often requires control overhead involving the construction of the structure for solving the problem, e.g. establishing a psuedo-tree for DPOP [1]. In a realistic decentralized setting of multiple agents, the establishment of a constraint optimization problem often occurs online and needs to be solved as quickly as possible and thus control overhead should be minimized. Also, the requirement for quick delivery of the solution limits the amount of communication or computation. Even without these problems, as the network gets very large, communication and computation burden still can be extensive given the time and bandwidth constraints and thus minimizing these resource requirements in these settings is valuable.

We consider the real-time sensor system NetRad, which is designed for detecting and monitoring hazardous weather phenomena in real time [2]. The NetRad system at the highest level is organized as a collection of controllers, each responsible for scheduling a cluster of radars based on the evolving weather scenario. Because one radar cannot scan effectively its entire region within the given time limit and some phenomena need to be scanned by more than one radar for velocity measurement, radar scheduling is vital for maximum system performance. Given a new map of phenomena, the policy generation should be done within the limited time (e.g. in this system 60 seconds).

We model the distributed scheduling problem as a constraint optimization problem and solve it approximately using the Max-Sum algorithm [3]. Although the Max-Sum algorithm is not guaranteed to be optimal with cyclic constraint graphs, it has been shown to work very fast and efficiently even in an approximate mode with little control overhead. This work proposes new extension of the Max-Sum algorithm for saving communication and computational resources by exploiting the innate semi-distributed system structure of NetRad. These techniques facilitate effective problem solving through the use of pre-computed policy and two phase propagation, one inside the clustered resources and one among clustered resources. This approach shows equivalent quality to the standard Max-Sum algorithm while reducing extensive communication requirements on average by 50% and computation resources by 5 to 30% depending on the specific problem instance. These experiments were performed in a realistic setting involving the scheduling of a network of as many as 192 radars in 48 clusters.

The rest of the paper is structured as follows: In Section 2 we review related work, while in Section 3 we model the scheduling problem as DCOP. In Section 4 we describe the individual techniques employed, while in Section 5 we present the experimental results. Finally, we conclude the paper and points to future work in Section 6.

## 2. RELATED WORK

As already described, we model the radar scheduling problem in NetRad system as a distributed constraint optimization problem (DCOP) solved with the approximate Max-Sum algorithm. DCOPs are applied on various problems on real world multiagent systems such as meeting room scheduling [1] and sensor network problems(traffic light synchronization [4] and smarthome devices [5]). Although such DCOP applications indicate that the exact algorithms work on such system, solving real world problems of 50 or more nodes often requires extensive communication on many problems [4].

There has been work done on speeding up the Max-Sum algorithm and the fast Max-Sum [6] avoids sending unnecessary information which will not be used in computing the maximum value. However, this work does not directly apply to the NetRad application where the utility of agents' choices are dependent on each other. Also, given the semi-centralized system structure, our techniques also can be applied on top of this extension of Max-Sum.

There are many approaches from a systems perspective for exploiting a semi-centralized system structure [7], but to our knowledge, there has not been extensive work involving DCOP algorithm. There is some work in the DCOP communities that utilizes partial centralization [8, 9] but that does not directly exploit the structure of hardware resources and the centralization decisions are made given the constraint graph. The most similar work is the iterative decoding algorithm in compound code networks [10] where the network consists of fixed-size subnetworks. The algorithm works on constituent codes which do not have cycles and then on the connections between constituent codes, therefore iterating between two different levels (within each subnetwork and across the sub-networks).

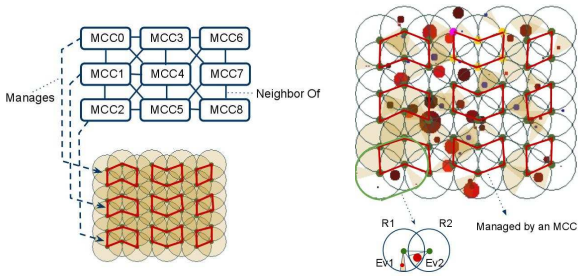
Our approach to exploit this clustered structure is to utilize a pre-computed policy and modify the message passing schedule of the algorithm for using semi-centralized system structure without involving additional control overhead. Optimization algorithms on graphical models can be considered as solving the multiple factorized local functions involving a subset of variables [11]. To our knowledge, the benefit of the prior knowledge on values of such local functions have not been studied.

Additionally, in the Max-Sum algorithm, we modified the so called synchronized flooding schedule [10], with which messages are exchanged between every neighbor each clock tick, to work in two phases given limited global dependencies in the utility structure of the graph. There are many approaches that work in two levels in DCOP literature[koptimal,ls-optimal] to save computation or to provide solution quality guarantees, however none works by modifying or delaying the message schedules.

# 3. RADAR SCHEDULING PROBLEM

## 3.1 NetRad System

The NetRad System is a system of radars specially designed for the purpose of quick detection of low-lying meteorological phenomena such as tornadoes. They are short-range radars used in dense networks, thereby alleviating blind-spots caused by the curvature of the Earth. NetRad radars additionally do not just use the traditional sit-and-spin strategy; rather, they can be focused to scan in a particular volume of space. By exploiting the collected information on weather phenomena, scanning strategies can be dynamically created for specific weather phenomena in the current environment.




(a) The system structure for 48 radars (b) 48 radars with 96 phenomena

**Figure 1: System structure for radars (a), Example configuration of radars with example scenario (b). All radar ranges and phenomena are assumed circular shaped. All phenomena locations and sizes are randomly selected. In (b), Radar 1 (R1) can choose to scan Event 1 (Ev1), Event 2 (Ev2) or to scan both depending on the utility. Scanning all phenomena in range with sufficient quality may not be possible given the time limit to scan.**

The NetRad system consists of multiple MCCs[1] each of which controls a set of radars. The MCC system is a closed loop control system in that it responds to the emerging weather events based on detected features in the radar data and end-user concerns that may vary over time. End-users such as forecasters, emergency managers, and researchers can provide information as to what sort of data they are looking for and how frequently. Consequently, the MCC ranks the importance of tasks dynamically so as to give preference to what the data users want currently.

The MCC gathers moment data from the radars and then runs detection algorithms on this weather data. The result of this analysis leads to a set of weather-scanning tasks of interest for the next radar scanning cycle. The MCC then determines the best scanning strategy for the available radars that will maximize the sum of the utility associated with the chosen tasks according to a utility function based on the end-user priorities. This scan strategy is used by the NetRad radars on the next cycle.

Tasks may also be either pinpointing or non-pinpointing, meaning either there is, or is not, a significant gain by scanning the associated volume of space with multiple radars at once. The utility gained from scanning a pinpointing task increases up to a certain limit with the number of radars scanning the task; the utility for a non-pinpointing task is the maximum utility among the individual radars that scan the same phenomenon. The global utility is simply the sum of utilities of all tasks. For a more in-depth description of the MCC system, see [2, 12] for more details on the utility function.

## 3.2 Radar Scheduling Problem Formulation

The NetRad system simulator consists of controllers of radars which are put in a grid structure and can communicate with all of its neighbors with equal cost. Each controller agent $A_i$ controls and schedules a set of radars $\mathbf{R}_i$. At each

[1]meteorological command and control

heartbeat the system shares a map of phenomena and the maximum utility assigned to each phenomena.

Given the map of phenomena, each radar selects discretized scanning ranges by choosing a subset of phenomena in its range. For efficiency, only the scans that tightly cover a phenomena are created and then merged with other such scans to create new combined wider sweeps. Therefore it limits unnecessary scans that do not cover any phenomena and minimize the domain size of each radar to simplify the scheduling problem while still having at least one scan that covers each phenomena.

For each phenomena $p_j$, the weight $w_j$ is a constant determined by the requested user or the weather pattern. The utility (factorized local function) for each phenomenon $p_j$ is defined as,

$$u_j : p_j \times \mathbf{r}^{p_j} \rightarrow c_j \tag{1}$$

where $c_j$ denotes coverage where $0 \leq c_j < 4$ and $\mathbf{r}^{p_j}$ denotes the scanning policy of radars which have $p_j$ in range.

The goal of the system is to find a radar configuration $r_1, \ldots, r_n$ which maximize the sum $U$ of the utilities for all phenomena and represented as,

$$U = \sum_j u_j(p_j, \mathbf{r}^{p_j}) \times w_j = \sum_j c_j \times w_j \tag{2}$$

Each radar with a set of possible scanning policy can be thought as a variable with a finite discrete domain and the local utility function $u_j$ works as constraint function with parameter $\mathbf{r}^{p_j}$. With these variables with finite domains and constraints, the radar scheduling problem can be naturally modeled as distributed constraint optimization problem with local functions involving a subset of variables.

# 4. MAX-SUM AND MODIFICATIONS

## 4.1 Max-Sum algorithm in NetRad System

Max-Sum is a distributed message-passing optimization algorithm belonging to the class known as Generalized Distributive Law (GDL) [13]. Max-Sum is a variation of Sum-Product algorithm where the global utility function is maximized.

In the Max-Sum algorithm, there is a set of variables $\mathbf{x} = \{x_1, x_2, \ldots, x_m\}$ on which a set of functions $\mathbf{F} = \{F_1, F_2, \ldots, F_n\}$ depend. Each function $F_i = F_i(\mathbf{x}_i)$, $\mathbf{x}_i \subset \mathbf{x}$. The goal is to find $\mathbf{x}^*$ which satisfies the following:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \sum_{i=1}^{n} F_i(\mathbf{x}_i) \tag{3}$$

Therefore, the Max-Sum algorithm can be viewed as a constraint optimization algorithm where the search is for the settings of variables which maximize the sum of a set of local utility functions. To achieve this, the Max-Sum algorithm defines a factor graph by creating a node for each variable and for each function. The graph is bipartite, and a function node is connected to a variable node if the corresponding function is dependent upon that variable. The bulk of the algorithm is in the messages passed between nodes, which are:

The message $q_{i\rightarrow j}$ **from Variable** $i$ **to Function** $j$ **is:**

$$q_{i\rightarrow j}(x_i) = \alpha_{ij} + \sum_{k \in M_i \setminus j} r_{k\rightarrow i}(x_i) \tag{4}$$

Here $\alpha_{ij}$ is a scalar set such that $\sum_{x_i} q_{i\rightarrow j}(x_i) = 0$, and $M_i$ contains the indices of function nodes connected to variable node $i$.

The message $r_{j\rightarrow i}$ **from Function** $j$ **to Variable** $i$**:**

$$r_{j\rightarrow i}(x_i) = \max_{\mathbf{x}_j \setminus i} [F_j(\mathbf{x}_j) + \sum_{k \in N_j \setminus i} q_{k\rightarrow j}(x_k)] \tag{5}$$

where $N_j$ contains the indices of variable nodes connected to function node $j$ in the factor graph.

If the factor graph is cycle-free, then the messages are guaranteed to converge, and the resulting solution will maximize $\sum_{i=1}^{n} F_i(\mathbf{x}_i)$. When the graph contains cycles, the messages may not converge, and even if they do the resulting solution may be sub-optimal. Empirical results show that even in this case, the algorithm frequently provides good solutions [14].

In NetRad domain, radars are mapped to variable nodes where each value of variable is a radar's possible scanning strategy and phenomena to function nodes and the optimization problem is defined as finding the radar configuration that maximizes the sum of utility functions for phenomena in the system. Each MCC controls multiple radars and messages exchanged within the MCCs have no cost. However, communication across MCCs occur for shared phenomena among multiple radars controlled by different MCCs.

## 4.2 Using Organization Structure: Max-Sum Alternating 2-level Hierarchy (MS2L)

We modify Max-Sum to work on two levels for the general graphs with cycles in order to increase the algorithm efficiency in the context of clustered hardware resources. We modified the message passing schedule of Max-Sum to propagate sometimes within a subgraph of the factor graph associated with clustered hardware resources where the computation within each node occurs as regular Max-Sum using the message values obtained in the previous cycle. There is no modification to the algorithm except skipping the computation of outgoing messages to the nodes outside the partition, thus saving inter-processor communication. This modified Max-Sum, which we call MS2L, alternates between a global propagation cycle and a local propagation cycle so as to ensure that the utility values can also travel to other parts of the graph.

In the algorithm, information is shared among nodes through messages and the algorithm converges to a single point when there is no new information flowing in any direction [15]. We conjecture that the communication can be more efficient when this information sharing is delayed until a subset of nodes become closer to consensus. By delaying sending messages outside the local processor until more developed values are constructed within a subgraph, we expect to reduce inter-processor communication without affecting overall performance. Also, in order to avoid getting into local optima, we ensure that the algorithm periodically communicate globally. Therefore we modify Max-Sum to have following message passing schedule.

The 2-level propagation schedule
1. (Initialization) At any vertices, carry out the global flooding.
2. (Local flooding) Both variable and function nodes sends messages only to the neighbors within the same MCC. For each local neighbor, given the newest message on each edge, compute the message values for each local neighbor and send. Let the variable node's neighbors be $N_i$ and the nodes in MCC $k$ $m_k$. In function nodes, it sends the same message to a subset of neighbors $N_i \cap m_k$. In variable nodes, it computes the message using the previous messages from neighbors outside the MCC. At cycle t, the message from the variable to function node is,

$$q^t_{i\to j}(x_i) = \alpha_{ij} + \sum_{k\in N_i \cap m_k \setminus j} r^t_{k\to i}(x_i) + \sum_{k\in N_i \setminus m_k} r^{t-1}_{k\to i}(x_i)$$

3. (Global flooding) For all neighbors, do a regular message calculation using the newest message on each edge. Function nodes compute the messages at cycle $t$ for all neighbors using messages at $t-1$ for neighbors $N_i \setminus m_k$. The function node does not have updated messages for all neighbors due to local propagation in the previous cycle thus it combines previous messages from neighbors outside MCC.

$$r^t_{j\to i}(x_i) = \max_{\mathbf{x}_j \setminus i}[F_j(\mathbf{x}_j) + \textstyle\sum_{k\in(N_j \cap m_k \setminus i)} q^t_{k\to j}(x_k) + \sum_{k\in(N_j \setminus (i\cup m_k))} q^{t-1}_{k\to j}(x_k)]$$

4. Repeat step 2 and 3.

This scheme is different from simplifying the problem by breaking the network into several subgraphs. It only delays the message delivery to make communication more efficient and the computational complexity remains same even in local flooding. This modification explores how Max-Sum can adapt to the system's organizational structure and its associated communication topology as well as the utility structure. In the NetRad domain, as the connection between function nodes and variables nodes are determined based on the spatial location of the corresponding radar and the phenomena, the dependency structure between nodes are simpler and the factor graph constructed following the domain is more easily decomposable. We exploit the property of the graph structure in the domain and modify Max-Sum to reduce the required resource for global level propagation.

Also, in NetRad, the system has an organizational structure where an MCC manages several radars and where MCCs communicate to collaborate with neighboring radars. It is beneficial to simplify the computation and communication occurring across MCCs. Therefore, we adapted Max-Sum to exploit this structure effectively by skipping some outgoing messages from MCCs in alternating cycles.

## 4.3 Starting with Known Policy

In the previous section, we proposed modifications to the message-passing scheme utilizing semi-centralized system structure. In this section, we propose to construct better initial messages incorporating global information to further optimize the efficiency of the algorithm i.e. to start the algorithm with a policy for subgraph contained in the cluster processor.

We speculate that a good known policy can be used to create such starting messages as it incorporates non-local inference if created based on more information than just a local function. In the Max-Sum algorithm, a node's outgoing messages are dependent on the incoming messages it received in the prior cycles and the first initial messages will depend only on the local function. The variable after the first message would take on the value

$$\tilde{x}_i = \arg\max_{x_i} \sum_{j\in N_i} \max_{\mathbf{x}_j \setminus i} F_j(\mathbf{x}_j) \tag{6}$$

This message would be the value assuming the best-case setting of other variables and only incorporates the local preferences. Given a known policy $\hat{x}$, we modify the algorithm for function nodes to send the following messages which do not involve maximization to the connected variable nodes. Function node $j$ to variable node $i$:

$$F_j((\hat{\mathbf{x}}_j \setminus i) \cup x_i) \tag{7}$$

After receiving these messages, if a variable node were to take on a value, it would be:

$$\tilde{x}_i = \arg\max_{x_i} \sum_{j\in N_i} F_j((\hat{\mathbf{x}}_j \setminus i) \cup x_i) \tag{8}$$

PROPOSITION 1. *If the assignment $\hat{\mathbf{x}}$ is such that no* individual *variable can by itself change its value to increase the global utility, then $\hat{\mathbf{x}}$ is a solution to the assignment constraints imposed by Equation 8. If changing any* individual *variable's value will strictly decrease the global utility, then $\hat{\mathbf{x}}$ is the unique solution for Equation 8.*

PROOF. From the perspective of an arbitrary variable node $i$, all other nodes are fixed to the configuration specified by $\hat{\mathbf{x}}$. Maximizing $\sum_{j\in N_i} F_j((\hat{\mathbf{x}}_j \setminus i) \cup x_i)$ leads to maximization of the global utility given the values of other variables. This is because only the functions for nodes $j \in N_i$ are affected by $x_i$.

If $\hat{x}_i$ were not a solution to this, then the algorithm which selected $\hat{x}_i$ to be part of $\hat{\mathbf{x}}$ could have instead selected $\tilde{x}_i$ to receive a higher utility. Since by supposition, no individual variable can change its value to increase the global utility, $\hat{x}_i$ is a solution to Equation 8. If changing any variable's value in $\hat{\mathbf{x}}$ will decrease the global utility, then there can be only one solution to Equation 8. Since $\hat{x}_i$ is a solution, it must be the unique solution. □

Thus, in the sense of the above property, we can insert a variable assignment into a factor graph as a starting solution. The property requires that no single variable can change its value to increase the utility. This is a desirable property for an optimization algorithm to have, and a fairly lax one. Any algorithm which does not satisfy this constraint can be followed by a hill-climbing procedure in order to meet the requirement of Property 1.

In addition to what Property 1 can tell us, Equation 8 by itself looks quite a bit better than Equation 6. While the assignment still only considers directly neighboring function nodes, it does so using better assumptions. For nodes other than itself, it assumes a configuration that is known to exist rather than a separate maximization for each function node. The assumed variable assignments are also known to be consistent with a good global utility, and $x_i$ will fit itself into this assignment.

After the messages from function nodes to variable nodes, we allow the variable nodes to send one set of messages before proceeding with the regular algorithm. This is so the next set of messages from function nodes will have a starting point other than assuming uniform functions in variable node messages.

#### 4.3.1 Using the Structure for Policy Generation

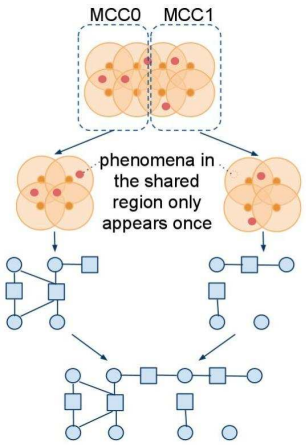



**Figure 2: 2-level Hierarchy Scheme**

We provide a scheme which computes a policy which can be used as in Section 4.3. Instead of generating a policy for the whole problem, we tried to compute the locally optimal policy for subproblems associated with each MCC(See Figure 2). We break the full factor graph into factor subgraphs for each MCC that contains only the radars and phenomena in each MCC and are smaller than the original factor graph. In this way, we first solve a smaller problem within MCCs and then solve a bigger problem using the information from the smaller problem. This is the key difference between local propagation in Section 4.2 as we break down the factor graph into subgraphs each of which has decreased complexity.

In order to accomplish this, we assign each phenomenon to one MCC to avoid redundant utilities for shared phenomena in computing the initial policy. Consequently, the domain of variable nodes and parameter values in the cost function at the function nodes are smaller than the original problem. Thus the computation at each function node $f_j$ that belongs to the set of nodes $m_k$, which belong to $MCC_k$, is done only for each neighbor $v_i \in m_k$.

$$r_{j\to i}(x_i) = \max_{\mathbf{x}_j \setminus i}[F'_j(\mathbf{x}_j \wedge m_k) + \sum_{k \in N_j \wedge m_k \setminus i} q_{k\to j}(x_k)] \quad (9)$$

The message from variable node $v_i$ to function node $f_j$ for $f_j \in m_k$ is,

$$q_{i\to j}(x_i) = \alpha_{ij} + \sum_{k \in M_i \wedge m_k \setminus j} r_{k\to i}(x_i). \quad (10)$$

We assume that function $F'_j$ with a subset of arguments that excludes the variable nodes outside the MCC can be deduced from the original function $F_j$. Additionally, the domain of a variable $v_i$ is a subset of the original problem only relevant to $f_j \in m_k$. The subproblem is used to create the policy used as prior information on local functions.

# 5. EXPERIMENTAL RESULTS

## 5.1 Experimental Setting

We experimented with the Max-Sum algorithm on an abstract simulation environment of the NetRad radar system developed in the Farm simulator framework [16]. In this simulator, weather tasks are abstracted as circular areas as shown in Figure 1(b). Aspects such as the utility function, the effective range of radars, and the separation between radars, however, are the same as in the operational testbed. On the simulation environment, see [17].

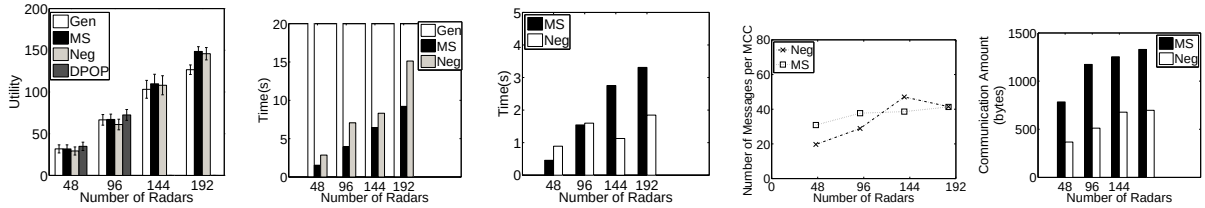
For a statistically meaningful result, we repeated each instance for 100 runs by randomly generating the weather phenomena varying in their size, location and importance. To make the results more easily interpretable, each trial is run for only one radar scan cycle for clear comparison.

The computation time is measured in CPU Time. The experiments were run on a single machine although we assumed that there are several computation units working in parallel in a time step simulation. For experiments involving both the negotiation algorithm and the Max-Sum algorithm, each MCC, when the local computation is completed, waits for other MCCs to finish the computation and then they exchange messages. Therefore, the time complexity in the decentralized setting results from the sum of the longest time taken in the local computation for each round.

We do not account for any communication delay in measuring the completion time. For measurement on communication amount, both the number of messages and the size of messages were measured counting the control messages to construct the network as well including the connectivity establishment between nodes and information sharing on possible values that each variable can take. The total amount of communication is measured in bytes considering one double number as 8 bytes and one integer as 4 bytes.

## 5.2 Performance of Max-Sum on NetRad

In order to evaluate the performance of several alternative optimization algorithms, we varied the number of radars and the number of phenomena. We compare the performance of Max-Sum algorithm to a decentralized negotiation algorithm [17], an exact distributed constraint optimization algorithm [1] and a centralized optimization algorithm based on a genetic algorithm that is currently used for local optimization in the negotiation algorithm in each MCC. The negotiation algorithm, specifically developed for the NetRad problem domain, is an iterative two step process performed concurrently at each MCC. In the first step each MCC performs a local optimization based on its local tasks and knowledge of its neighboring MCCs's proposed scan schedules. In the second step, the MCC negotiates with its neighbors so as to make adjustments to its scheduling based on the strategy of other MCCs. This two step process for performing the distributed optimization tries to maximize the parallelism at the MCC level and to minimize communication among MCCs. In contrast, the standard Max-Sum algorithm does not consider such an organizational structure and is completely decentralized. The Max-Sum algorithm does not explicitly take into account that certain communication links are within an MCC cluster and others are between MCC clusters. The genetic algorithm uses a centralized approach; no communication is required and it only utilizes one processor.




(a) Performance quality (b) Time (c) Time in the decentralized setting (d) Messages (e) Communication

**Figure 3: Gen:Genetic, MS:Max-Sum, Neg:Negotiation. The algorithm is run with the same number of tasks (weather phenomena) as the number of radars. The Genetic algorithm is run with a computation time limit of 10 minutes. We set the time limit to 10 minutes in order to get reasonable optimizations for sake of measuring the performance. Given less than 10 minutes, the utility generated by the centralized optimization were significantly lower than the other approaches**

### *5.2.1 Different Network Sizes*

In order to evaluate the general performance and the scalability of the algorithms, we compare the performance on different sized networks. In these scenarios, there are the same number of phenomena as the number of radars in the network as shown in Figure 3. The performance quality of Negotiation and Max-Sum are approximately the same for all network sizes whereas the performance of the centralized genetic algorithm starts to degrade with more radars. Both the Negotiation and Max-Sum algorithm remain quite close to the optimal solution computed by the DPOP algorithm on 48 and 96 radar cases. We were not able to use DPOP on bigger networks due to memory constraints. The result shows that the Max-Sum algorithm is able to handle the problem well in terms of both quality and computation time on bigger sized problems. We have varied the parameters of the genetic algorithm to improve its performance but it still remains inferior to other methods. This applies similar to an optimization algorithm based on Simulated Annealing not shown in the graph due to general inferior performance on the problem.

In terms of communication, when only messages exchanged across MCCs are counted, Max-Sum needs no more than twice the communication in the negotiation algorithm, but remains significantly lower in comparison to DPOP as seen in Table 1. Communication only between MCCs are measured for both algorithms.

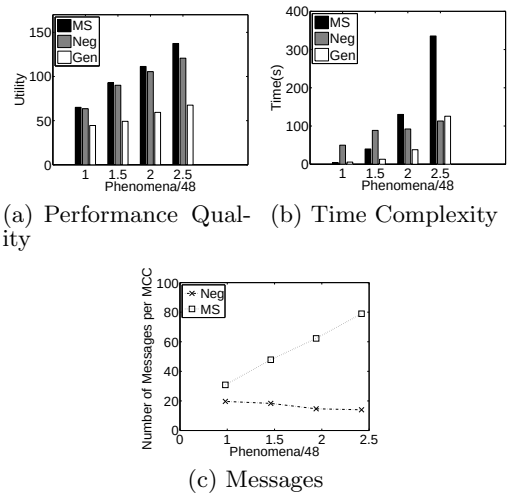
| | 48 | 96 |
|---|---|---|
| Max-Sum | 687.15 | 894.71 |
| DPOP | 3.98M | 35.46M |

**Table 1: Communication Amount in bytes by DPOP and Max-Sum**

### *5.2.2 Different Number of Phenomena*

In the next experiment, we increase the number of phenomena in a 48-radar network, thereby requiring more coordination among radars and studied how the algorithms perform. While the quality of solution of Max-Sum is slightly better, the time complexity of the Max-Sum algorithm sharply increases because the number of neighbor in function nodes in Max-Sum increases as more weather phenomena are added.

Also, the number of messages across MCCs increases as shown in Figure 4(c) as there are more tasks shared by multiple MCCs in the environment. In contrast, the number of




(a) Performance Quality (b) Time Complexity

(c) Messages

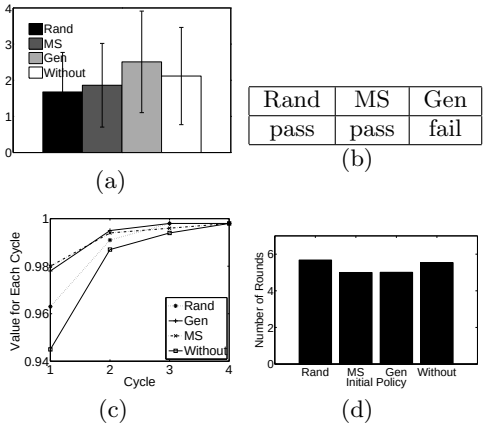
**Figure 4: Run with different number of phenomena. The basis is 48 weather phenomena and this is increased to 120 phenomena(i.e. 2.5).**

messages in the negotiation algorithm decreases because the algorithm quickly converges to a suboptimal solution due to a time limit on local computation inside each MCC subject to time constraints in the system resulting in an early termination within only 1-2 cycles.

## 5.3 Starting with Initial Policy

As described in Section 4.3 and Section 4.3.1, we provided the algorithm with 3 different approaches for computing the initial policy of a 48 radar network with the limit of 10 rounds, where we took the results at the last cycle. The initial policies includes : 1) the solution from the genetic algorithm within MCCs(Init-Gen), 2) the solution of the Max-Sum algorithm within MCCs (MS-Init) following the scheme described in Section 4.3.1, and 3) a randomly generated initial policy (Init-Rand). The initial policy for Init-Rand was generated in each function node which is potential inconsistent among multiple function nodes but these inconsistencies are often quickly resolved.

The quality of the final solution rarely changes by more than a fraction of utility and the computation time including policy generation decreases. The policy helps the algorithm




| Rand | MS | Gen |
|---|---|---|
| pass | pass | fail |

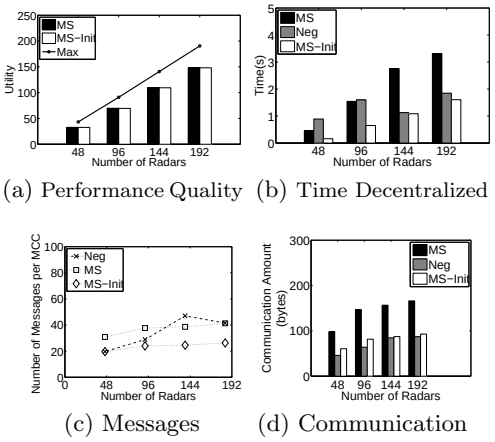
(a) (b) (c) (d)

**Figure 5: (a) The decentralized computation time of Max-Sum including policy generation time (b) T-test result on hypothesis that each policy improves the computation time of regular Max-Sum with $\alpha = 0.05$ (c) Value convergence trend at each round (c) Number of Max-Sum rounds**

save computation time as shown in Figure 5(a) and also improves its anytime performance behavior as in Figure 5(c). Given the policy, Max-Sum can almost instantly produce the initial messages avoiding the maximization step and the policy also reduces the number of rounds taken to converge.

Using this policy, Max-Sum seems to produce a policy with high utility quickly because the messages at the initial round are not biased towards the local functions reaching the final solution quickly as shown in Figure 5(c). It also shows that Max-Sum with a higher-utility-initial-policy starts with the higher utility.

Init-MS provides computation time saving as well as quick convergence speed and stability, although it requires specific domain structure to be effective. Additionally Init-Rand shows less anytime characteristics and some unstable performance showing oscillation between multiple values ending up a higher number of rounds to converge as in Figure 5(d), but it is a quick and easy way to provide an initial policy on any occasion and no need to synchronize policies over multiple nodes is another benefit.




(a) Performance Quality (b) Time Decentralized

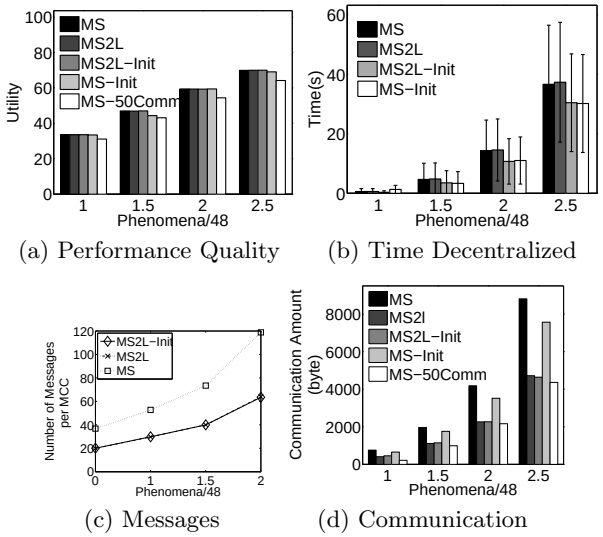
(c) Messages (d) Communication

**Figure 6: Performance of MS-Init. MS:Max-Sum, Neg:Negotiaton, MS-Init: Max-Sum with Init-MS**

### *Policy Generation Using Structure*

We experimented with the algorithm using Init-MS further to show how much we can improve the algorithm using the policy. We ran Max-Sum for 5 cycles where, for Max-Sum with Init-MS, we replace the first two cycles with initial policy computing cycles.

As shown in Figure 6, not only does the performance quality remain similar to Max-Sum, the time complexity decreases by half as well as the communication amount. This computational saving is due to the fact that the computation on the factor graph using only local nodes is much simpler than the computation on the global-level factor graph and also the result of this computation leads to a quicker convergence on the global level. As messages are exchanged only within MCCs to compute the initial policy, the number and size of messages also decreases.




(a) Performance Quality (b) Time Decentralized

(c) Messages (d) Communication

**Figure 7: Performance of MS2L**

## 5.4 Performance of Max-Sum in a Two-Level Hierarchy

We next tried MS2L, the alternating Max-Sum algorithm that uses a repetitive cycles of local and global propagation of messages as in Section 4.2. We experimented with increasing number of phenomena and also with the Init-MS policy replacing the first 2 cycles for generating the policy. We ran experiments with various number of phenomena, as we would like to see the result of MS2L varying the network connectivity which will increase the need for global propagation.

As shown in Figure 7, the utility of MS2L with Init-MS (MS2L-Init) remains similar. Moreover, the communication is reduced by half. It also shows the result of MS-50Comm where we randomly skip the communication 50% of the time and note that the algorithm has not yet converged in the given number of cycles unlike MS2L. Even in the local propagation cycle, the utility is being propagated as effectively as the global propagation cycle and half of the global propagation cycles are enough to reach similar performance. Also by starting with Init-MS, the computation time can be also decreased as in Figure 7(b).

# 6. CONCLUSION

We applied the Max-Sum approximate constraint optimization algorithm in the NetRad system for coordinating and scheduling weather-sensing radars. In this system, Max-Sum generated policies with high utility but requires more communication and computation than the negotiation algorithm in some settings. We thus modified Max-Sum to start with initial policy to guide and expedite the search process and improves Max-Sum's anytime performance and saves computation. As part of using a policy, we generated a scheme to create a starting policy which works in the two-level hierarchy solving a partial problem within the local processer. Additionally, we developed MS2L, an adapted message passing scheme which alternates between different scopes of the message propagation. This scheme proved the benefit of exploiting the organizational structure by requiring less computation and communication than all other tested algorithms.

This work on the Max-Sum algorithm suggests some directions for future research. One possible direction is a decomposition of factor graph given weak dependency. Temporarily restricting the connections to a subset of variable nodes based on the factor graph will greatly reduce the complexity of the function nodes. This will also reduce the required communication as well as the computational complexity.

Finally, as shown in the experiments, Max-Sum propagates information effectively with delay as in MS2L and absorb new information along with previous messages. In a dynamic environment when tasks are changing, Max-Sum could potentailly profit from using the solution from the previous problems resulting in a saving in resources; we are now exploring this option.

## Acknowledgement

This work was supported in part by the Engineering Research Centers Program of the National Science Foundation under NSF Cooperative Agreement No. EEC-0313747. Any Opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation.

We would like to thank Alessandro Farinelli, Alex Rogers, and Ruben Stranders of the University of Southampton for their assistance in this project.

# 7. REFERENCES

[1] Adrian Petcu and Boi Faltings. A scalable method for multiagent constraint optimization. pages 266–271, 2005.

[2] M. Zink, D. Westbrook, S. Abdallah, B. Horling, E. Lyons, V. Lakamraju, V. Manfredi, J. Kurose, and K. Hondl. Meteorological Command and Control: An End-to-end Architecture for a Hazardous Weather Detection Sensor Network. In *Proc. of the ACM Workshop on End-to-End, Sense-and-Respond Systems, Applications, and Services*, pages 37–42, 2005.

[3] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 639–646, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.

[4] Robert Junges and Ana L. C. Bazzan. Evaluating the performance of dcop algorithms in a real world, dynamic problem. In *AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 599–606, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.

[5] Federico Pecora and Amedeo Cesta. Dcop for smart homes: A case study. *Computational Intelligence*, 23(4):395–419, November 2007.

[6] Sarvapali Ramchurn, Alessandro Farinelli, Kathryn Macarthur, Mariya Polukarov, and Nick Jennings. Decentralised coordination in robocup rescue. *The Computer Journal*, January 2009.

[7] Dipanjan Chakraborty, Anupam Joshi, Yelena Yesha, and Tim Finin. Gsd: A novel group-based service discovery protocol for manets. In *In 4th IEEE Conference on Mobile and Wireless Communications Networks (MWCN*, pages 140–144, 2002.

[8] Adrian Petcu, Boi Faltings, and Roger Mailler. Pc-dpop: a new partial centralization algorithm for distributed optimization. In *IJCAI'07: Proceedings of the 20th international joint conference on Artifical intelligence*, pages 167–172, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[9] Roger Mailler and Victor Lesser. Asynchronous Partial Overlay: A New Algorithm for Solving Distributed Constraint Satisfaction Problems. *Journal of Artificial Intelligence Research*, 25:529–576, April 2006.

[10] Frank R. Kschischang and Brendan J. Frey. Iterative decoding of compound codes by probability propagation in graphical models. *IEEE Journal on Selected Areas in Communications*, 16(2):219–230, 1998.

[11] F. R. Kschischang, B. J. Frey, and H. A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, Feb 2001.

[12] James F. Kurose, Eric Lyons, David McLaughlin, David Pepyne, Brenda Philips, David Westbrook, and Michael Zink. An End-User-Responsive Sensor Network Architecture for Hazardous Weather Detection, Prediction and Response. In *Proceedings of the Second Asian Internet Engineering Conference, AINTEC*, pages 1–15, 2006.

[13] S.M. Aji and R.J. McEliece. The generalized distributive law. *Information Theory, IEEE Transactions on*, 46(2):325–343, Mar 2000.

[14] Ruben Stranders, Alessandro Farinelli, Alex Rogers, and Nick Jennings. Decentralised coordination of mobile sensors using the max-sum algorithm. In *Proc 21st Int. Joint Conf on AI (IJCAI)*, 2009.

[15] Yair Weiss and William T. Freeman. On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, 47(2):736–744, 2001.

[16] Bryan Horling, Roger Mailler, and Victor Lesser. Farm: A Scalable Environment for Multi-Agent Development and Evaluation. In *Advances in Software Engineering for Multi-Agent Systems*, pages 220–237. 2004.

[17] Michael Krainin, Bo An, and Victor Lesser. An Application of Automated Negotiation to Distributed Task Allocation. In *2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2007)*, pages 138–145, Fremont, California, November 2007. IEEE Computer Society Press.