Approximate Stochastic Constraint Solvers over Watts-Strogatz Network Topologies

K. Nathaniel Tucker
Harvard University
tucker@college.harvard.edu

ABSTRACT

Distributed constraint optimization (DCOP) is a powerful framework for representing and solving distributed combinatorial problems. DCOP problems search for the optimal solution, optimizing the total gain or cost of all the agents together.

However, optimally solving these problems can be costly and prohibitively so, thus there are a class of solvers that find approximations to these problems. This paper will be addressing these solution concepts.

The paper will study different properties including: improvement rate, runtime, and complexity under different types of network structures, most importantly: high average clustering coefficient and low characteristic path length.

Findings show tradeoffs between complexity/runtime and overall performance. They also hint at particular solvers, MGM and Max-Sum, running better on particular networks, high and low clustering respectively.

General Terms

DCOP, Constraints, Approximation Techniques, Network Topologies

Keywords

DSA, BR-I, SAP, MGM, Max-Sum, Watts-Strogatz

1. INTRODUCTION

Constraints are elements of situations that limit available solutions to problems. Constraints by limiting the possible range of solutions, thereby encode information into a system. Thus allowing computationally intractable problems at times to be efficiently solved.

This concept has been applied to scheduling, automated reasoning, and decision theory problems. All involve computationally intractable problems that can be made tractable by understanding the underlying constraints imposed on the problems.

The paper will focus on the most tractable form problems and solutions, specifically stochastic optimization on the most generalizable form of distributed multi-agent constraint optimization problems: DCOP problems. And the goal of this paper is to optimize these solutions.

In these DCOP problems a set of agents will work together using only local information and message passing in order to jointly achieve a globally optimal solution.

The most natural way to see an example of this behavior is to consider a scheduling problem, where agents must agree on the time of one or various meetings.

Therefore one can see the global problem of scheduling a company's meeting for the week, the local problem of when to schedule a department's meeting (represented as a neighborhood), and of course the individual problem.

However, despite the tractability of these solution concepts, there remains a dearth literature about them. They have yet to be studied under various conditions.

Thus the motivation behind this paper is to devise a series of heuristics that can make these problems even more tractable, by analyzing the solvers over various network topologies.

2. BACKGROUND AND RELATED WORK

This work focuses on the unique intersection of three domains: Network Topologies, Constraints, and Stochastic Approximation Techniques. All of which come together to form the foundations of the project.

2.1 Network Topology

Here we will give a definition of what network topology is, and a justification for studying it.

Network topology is loosely speaking the arrangement of various elements of a network. And depending on the way that nodes are distributed on a graph, such a graph will have different topologies.

Our paper will consider an undirected graph $G = (E, V)$.

And there will be a set of features that are important for us to understand about the graph. We will let d_{ij} be the shortest distance between nodes $v_i, v_j \in V$. We define n to be the number of nodes in V. We define the neighbors of a specific node as $N_i = \{v_j : e_{ij} \in E\}$, and thus we define $|N_i|$ to be the degree of $v_i \in V$. Finally we define the clustering of a specific node as:

$$c_i = \frac{|\{e_{jk} : v_j, v_k \in N_i, e_{jk} \in E\}}{(|N_i| - 1)|N_i|} \tag{2.1.1}$$

These definitions are necessary for defining the two aspects of graph topology that will interest us: characteristic path length and clustering. The combination of the two will be referred to as having Small World properties. These will be studied on constraint networks.

We will define these properties as:

1. Characteristic path length of $L(G)$

$$L = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} \tag{2.1.2}$$

2. Clustering coefficient $C(G)$

$$C = \frac{1}{n} \sum_{i=1}^{n} c_i \tag{2.1.3}$$

Finally we will define high clustering and low characteristic path length to be having the Small World property. We will now show why we wanted to study the Small World property vs. the many others.

2.1.1 Constraint Network Topologies

In Jin & Liu 2003, they studied naturally occurring constraint networks and the topologies thereof. They discovered that in DCOP problems, the networks could have the Small World topology, depending on how it was formulated. If each variable was controlled by one agent, then the Small World property was observed. If instead the agents were organically laid out over the variables, the Small World property was not. The conclusion was that different topologies can occur via different constructions. [6]

This ability to easily change the topology of the network, allows designers to further optimize the process by first composing the constraint network into the most favorable topology. This shows us why the Small World property, out of many others should be studied.

Moreover in Araujo & Lamb 2008, they discuss creating networks that are optimal for various constraint solvers. Thus where the designers are able to actually construct the network itself, knowledge of where solvers succeed would be very important. [1]

In sum, the Small World property is generally inherent to real social networks, and can be easily constructed and removed artificially into and from a network. Thus this paper will explore the Small World property to see which approximate algorithm preforms best with and without it.

2.1.2 Small World

The Small World property itself is actually quite important. Many natural and man-made systems exist in the form of networks, and research has found that many such networks exhibit a these characteristics in their topology, which means that the shortest distance between any two nodes is quite small and that the nodes are highly clustered.

These properties bring about efficiency for a network. In Jin & Liu 2003, they argued that a network with these two properties would support efficient communication both globally and locally. They defined ϵ_{ij} to be the communication efficiency between to vertices v_i, v_j. They suggest $\epsilon_{ij} = \frac{1}{d_{ij}}$ where d_{ij} is the shortest path between the two. Based on this they defined the communication efficiency of the entire network G to be as follows:

$$E_{globe} = E(G) = \frac{1}{n(n-1)} \sum_{i \neq j \in G} \epsilon_{ij} = \frac{1}{n(n-1)} \sum_{i \neq j \in G} \frac{1}{d_{ij}} \tag{2.1.4}$$

Then assuming that G_i is a sub-network of G composed only of the neighbors of v_i (including itself) and the edges between them. Then the local communication efficiency of G is as follows:

$$E_{loc} = \frac{1}{N} \sum_{i \in G} E(G_i) \tag{2.1.5}$$

It was found that E_{globe} would be high in networks with the low characteristic path length, and that E_{loc} would be high in networks with high clustering. Thus one would predict that networks with high communication would be able to solve global problems more easily. [7]

However such a prediction may often turn to be the opposite. Walsh 1999 proved that such a topology: high clustering and low characteristic path length, can make search problems very difficult. The search cost actually increased as the graph's characteristic path length decreases. [12]

Thus the crux of our project is to show empirically where each claim holds.

2.2 Constraints

Here we will form a definition of constraints and some of the assumptions we will make about them.

The concept of a constraint is often central to human activities. Most generally, a constraint limits the possibilities of a certain situation or a certain universe.

The simplest example of a system of constraints is a meeting schedule for a company. Natural constraints for the above example are: each person can only attend one meeting simultaneously, meetings must be held on weekdays, etc.

And such an example can give us intuition on different types of constraints. In general one can have hard constraints and soft constraints. In the above example, a person being in two meetings at once is a hard constraint. It is impossible. Meanwhile meetings not being on the weekend can be a soft constraint. It would obviously have high cost to do so, but it is not impossible.

Constraint programming then is a general framework for providing algorithms for solving these types of problems.

The process to solving these problems comes in two steps:

1. Represent the problem, generally in a constraint network
2. Solve the constraint network

This paper will not focus on the first process, but rather the second. The second is generally solved with a constraint solver. And the solution is a value for each variable. These constraints will be represented in constraint networks.

2.2.1 Constrain Network

A constraint network is defined by the three tuple: (X, D, C), where $X = \{x_1, ...x_n\}$ is a discrete set of variables; these will correspond to our $V \in G$. $D = \{D^1, ...D^n\}$ is a unique set of variable domains enumerating all possible values of the corresponding variables. And $C = \{C_1, ...C_m\}$ is a set of constraints; these will correspond to our $E \in G$. Constraints can be of two types: hard and soft.

Soft and Hard Constraints

Hard constraints enumerate all valid joint assignments of all variables in the scope of the constraint. A soft constraint, C_i^s is a function F_i, that is defined on the variables $S_i \subseteq X$ which comprise the scope of the function. Thus one can write $F_i = D_i^1 \times ... \times D_i^s \to \mathbb{R}$ where $s = |S_i|$. Thus one can imagine how one can encode hard constraints in soft constraints, simply by imposing higher costs to certain functions. Thus one should simply care about soft constraints.

Domains in Constraint Networks

As for domains, while one can represent any number of domain in our network, one can focus on binary domains and thus binary constraint networks. The is fully generalizable seeing that every constraint network can be mapped to a binary constraint network.

In the general case DCOP problems are NP-Hard, and generally can only be solved in exponential time. Thus this paper's focus on approximation techniques to avoid this deficit.

2.3 Stochastic Approximation Techniques

Stochastic optimization techniques are used in a plethora of areas, including aerospace, medicine, and finance. Stochastic algorithms allow one to efficiently find optimal solutions.

Stochastic approximation methods attempt to find maxima of functions that may not be able to be computed directly. These are optimization methods that then generate and use random variables.

The randomness injected into the optimization can enable it to avoid local maxima. In general these techniques tend to avoid high computation and memory usage. This paper will focus on these techniques on constraint networks over various network topologies.

2.3.1 Constraint Optimization Problems

In soft constraint networks, one faces the problem of constraint optimization problems. The goal is to find the best solution, or to optimize a global function $F(\bar{a}) = \sum_i F_i(\bar{a}_i)$ where $\bar{a} = \{a^1, ...a^n\}$ given that $a^j \in D^j$. Thus a general constraint optimization problem will seek to maximize:

$$\bar{a}^* = argmax_{\bar{a}} \sum_i F_i(\bar{a}_i) \tag{2.3.1}$$

This is generally the case. We however will be dealing with DCOP problems. Our setting will involve agents that control only one variable, again without loss of generality. However, we will make the assumption that agents are assumed not to be self interested, and their one goal is to optimize the global function.

Optimal DCOP Algorithms

Finding an optimal solution for a DCOP problem is NP-Hard problem. This is most easily seen by reducing the problem to the problem of deciding 3-colorability of a graph. However solutions do exist.

The solutions are either search based or dynamic programing Based. An example search based algorithm would be ADOPT (Asynchronous Distributed OPTimization). And an example of a dynamic solution would be DPOP (Dynamic Programming Optimization Protocol). [16]

The characteristics for optimal DCOP algorithms are well understood, and the properties of the entire framework of local message passing algorithms have been extensively analyzed, with guarantees placed on their solutions under a range of assumptions, for example tree structure or single loop graphs. In contrast, no such analysis has been completed for approximate DCOP algorithms.

One reason for this deficit, is that these algorithms arise from very different literatures. They either are from adaptive processes of game theory, or from distributed versions of centralized procedure developed for traditional constraint solving systems.

To better understand, consider how traditional constraint optimization problems evolved. The traditional computer science approaches to these problems, like the breakout algorithm, arc consistency, dynamic programing, and stochastic optimization techniques, became suddenly distributed. For example distributed breakout and maximum-gain messaging were two DCOP solvers descended from the breakout algorithm.

Now consider game theoretic approaches for DCOP. Here each autonomous agent's aim is to maximize its own private utility function, through its independent choice of strategy. And such an approach can easily lead to concepts such as distributed stochastic algorithm. [3]

For the most part, optimal solutions are computationally and memory intensive, and for the purposes of this paper we will not focus on them.

Approximate DCOP Algorithms

Again, solving a constraint network is an NP-Hard problem, and often the worst-case of the complete methods are prohibitive for practical applications. In these cases, approximate algorithms are preferred.

Here one is often sacrificing the optimality of a given outcome for computational and memory efficiency. These will require very little local computation and communication. These are the algorithms that we will focus on.

2.4 Summation

We have explained the reasons for studying the Small World property, our assumptions when dealing with a constraint network, and our rationale for studying approximate DCOP solvers and assumptions thereof. We will now show how we formulated our experiment.

3. EXPERIMENTAL DESIGN

We will be designing and using: a distributed constraint network, constraint solvers, and a network generator (particularly important is the Watts-Strogatz model) for this experiment.

3.1 Small World Topologies

There are two properties that we will model over our topologies:

1. Characteristic path length of $L(G)$: 2.1.2
2. Clustering coefficient $C(G)$: 2.1.3

The characteristic path length measures the typical separation between two vertices. The average clustering coefficients measures cliquishness of a typical neighborhood on the graph. The paper will be using the Watts-Strogatz model to achieve both.

3.1.1 *Watts-Strogatz*

The Watts-Strogatz model utilizes two graph structures: a ring lattice and an Erdos-Renyi random graph. The ring lattice has high clustering coefficients, and the Erdos-Renyi has low shortest path.

The Watts-Strogatz model then interpolates the two. Starting with a regular ring lattice, the model rewires each edge at random with some probability p, where if $p = 0$ it will remain a regular ring lattice, and if $p = 1$, it will become an Erdos-Renyi.

The paper will examine behaviors over this model with probabilities ranging from zero to one. We will then overlay a distributed constraint network atop these created graphs.

3.2 Distributed Constraint Networks

A Distributed Constraint Network will be the underlying model of this paper. It can be formally defined by the four tuple (X, D, C, A). Where each refer previously to a constraint network in its simpler form, except A. This is the set of agents $A = \{A_1, ..., A_k\}$ where each agent can only control a subset $X_i \subseteq X$, and each variable is assigned to one agent (injective).

There are in particular a few more stipulations that we have for the agents. Agents can control only the variables that were assigned to them, meaning they can only see these variables and only change these variables. Furthermore, agents can only see the constraints that involve variables that they control. And finally, we will coin the term neighbors, which will be any two agents for which there exists a constraint that depends on variables that they both control. Only neighbors can communicate with each other. This is a distributed constraint network.

3.3 Constraint Functions

Constraint functions in their most general form can have nearly ubiquitous use. However changing the constraints of the constraint network itself could potentially have dramatic effect. The paper will examine two scenarios of constraints, in a cooperative and non-cooperative mode.

Our example of an anti-cooperative constraint network is simply 3-coloring on a graph. This will be encoded in soft functions with a reward of 1 for colors being different. That of a cooperation game is classically where nodes derive benefit for being the same color as their neighbors. This will be encoded in soft functions with a reward of 1 for being of the same color.

These two constraint functions will be tried with our constraint solvers.

3.4 Approximate DCOP Algorithms

The experiments of this paper all focus on approximate DCOP algorithms.

There are two classes of approximate algorithms for addressing DCOP's that the paper will focus on: local Greedy algorithms and Generalized Distributed Law (GDL) based algorithms

Greedy Approximate DCOP Algorithms

In such a local greedy algorithm, one begins with a random assignment for all variables. Then based on a series of greedy local moves, the algorithm will seek to optimize the global function. A local move will consist of gathering local information and then changing one or more local variable, local being defined as members of a neighborhood.

The neighborhood will be dependent on a parameter k, where k is the size of coalitions that agents can form. For us they will be either the degree of the agent, or 1, in our greedy algorithms. [16]

Because these are greedy algorithms, they will be searching for some local gain, i.e. a local move that will increase the global function 2.3.1. Thus the search stops when there are no more moves of this kind.

In principal the flaw with these types of applications is just that. Local searches and greedy choices often lead to local maxima. The presences of local maxima mean that a result can be arbitrarily far from the global optimal solution. However, generally the results in practice are quite good.

In previous work Chapman et. al. 2011, greedy stochastic algorithms used in the paper were benchmarked on graph coloring where the mean connectivity of the nodes was held constant and the constraint functions were uniform. The results showed an ordering of SAP > DSA > BR-I > MGM, and in fact SAP, DSA, and BR-I all achieved the highest of the scores. [3]

The paper focuses on these four algorithms of such kind: DSA, BR-I, MGM & SAP.

DSA is the simplest yet still effective way to approach the Greedy algorithm here. DSA, or the Distributed Stochastic Algorithm has been widely applied in many domains. Our implementation of DSA is seen in algorithm 1.

DSA will pick some initial value for all agents. Then DSA will continue to iterate through its steps until termination condition is met. In each step the agent contacts the local graph and randomly chooses to change or keep its old value. [5]

Algorithm 1 DSA: Distributed Stochastic Algorithm

```
 1: Initialize each variable, $x_i \in X$ to random $d \in D_i$
 2: for Each Activation Step do
 3:     for Each Agent $A_i$ do
 4:         Choose activation probability $p_i \in [0, 1]$
 5:         Generate some random number $r_i \in [0, 1)$
 6:         if $r_i < p_i$ then
 7:             Choose new value $a_i$ such that local gain is maximized
 8:             $x_i \leftarrow a_i$
 9:         end if
10:     end for
11: end for
```

DSA can be used synchronously and asynchronously, however this implementation is synchronously.

Through empirical work, the algorithm is effective if the activation probability is generally low, allowing informational coherence in the network. However this is also a huge drawback of the mechanism as the quality of solutions can be very dependent on the chosen activation probability.

The strength of the algorithm lies in the fact that DSA is extremely low overhead in terms of memory, computation, and communication. And generally speaking DSA is monotonic towards a maxima.

BR-I or better reply with inertia. The BR-I uses the immediate payoff for selecting a strategy as its target function, but for its decision rule, it uses randomization instead of just pure optimality. The choices are chosen according to a uniform distribution on all choices satisfying:

$$F(a_i, a_{-i}) - F(a_i^{t-1}, a_{-i}) > 0 \qquad (3.4.1)$$

A description of the solver is shown in algorithm 2

Algorithm 2 BR-I: Better Reply with Inertia

```
 1: Initialize each variable, $x_i \in X$ to random $d \in D_i$
 2: for Each Activation Step do
 3:     for Each Agent $A_i$ do
 4:         Choose activation probability $p_i \in [0, 1]$
 5:         Generate some random number $r_i \in [0, 1)$
 6:         if $r_i < p_i$ then
 7:             Choose uniformly a $a_i$ that satisfies equation 3.4.1
 8:             $x_i \leftarrow a_i$
 9:         end if
10:     end for
11: end for
```

BR-I has similar properties to DSA, but is more able to avoid local maxima.

MGM is an alternative approach clearly addressed the possibility of out of date knowledge, and forces agents in a neighborhood to agree on which agent has the right to move. MGM or the Maximum Gain Message Algorithm, is shown in algorithm 3.

With MGM agents preform a series of steps until some final condition is met. In each step the agent constructs the local neighborhood and finds its maximal gain given that neighborhood. The agent then calculates the gain of each of its neighbors and changes its value only if it has the maximum gain.[5]

Algorithm 3 MGM: Maximum Gain Message Algorithm

```
 1: Initialize each variable, $x_i \in X$ to random $d \in D_i$
 2: for Each Activation Step do
 3:     for Each Agent $A_i$ do
 4:         Collects Gains and Values from neighbors
 5:         Choose a values $a_i^*$ such that local gain, $g_i^*$, based on current neighbor values is maximized
 6:         if $g_i^*$ is greatest in neighborhood then
 7:             $x_i \leftarrow a_i^*$
 8:         end if
 9:     end for
10: end for
```

MGM has some advantages over DSA. MGM has no parameters to tune. MGM also have very low overhead in terms of memory, computation, and communication. And MGM is monotonic towards a local maximum. This is the last Greedy algorithm that this paper will focus on.

SAP or spatial adaptive play, uses the immediate payoff for selecting a strategy as its target function. However, it differs from the previous solvers in its decision rule. SAP uses a multinomial logit decision rule, known as the Boltzmann distribution:

$$P_{a_i}(\eta) = \frac{e^{\eta^{-1} g_i}}{\sum_i e^{\eta^{-1} g_i}} \tag{3.4.2}$$

Thus the strategies are chosen in proportion to their reward. But their relative probability is controlled by η. If $\eta = 0$ then the *argmax* function results. If $\eta = \infty$ then it produces a uniform random distribution across all strategies, thus creating a random walk.

Increased η increases the runtime, but allows the algorithm to escape suboptimal solutions. The η is then decreased over time. The algorithm is written below in algorithm 4

Algorithm 4 SAP

```
1: Initialize each variable, $x_i \in X$ to random $d \in D_i$
2: Initialize $\eta$
3: for Each Activation Step do
4:     for Each Agent $A_i$ do
5:         Find $a_i$ by drawing from our Boltzmann parametized by $\eta$ (the equation 3.4.2)
6:         $x_i \leftarrow a_i$
7:     end for
8:     decrement $\eta$ approaching 0
9: end for
```

SAP closely resembles distributed simulated annealing and preformed well empirically.

3.4.1 GDL based Algorithms

Generalized Distributive Law (GDL) is a unifying framework to preform inference on graph models. It is a generalized message passing algorithm formed from the synthesis of information theory, digital communications, signal processing, and other works. The GDL algorithm that this paper will focus on is Max-Sum.

Max-Sum This is an iterative message passing algorithm. Agents will continually exchange messages to build up local functions that depend only on variables that they control. Again, Max-Sum can handle larger domains and more than one variable per agent, however we will only describe the binary and single variable domain. We will again assume a synchronous model. We have included a description in algorithm 5.

Max-Sum is an algorithm in which agents exchange messages which accumulate that sum of all costs for possible assignments. A message is of the form:

$$m_{i\to j}(x_j) = \alpha_{ij} + \max_{x_i} \left(F_{ij}(x_i, x_j) + \sum_{k \in N_i \neq j} m_{k\to i}(x_i) \right) \tag{3.4.3}$$

Where α_{ij} is a normalization constant.

The algorithm begins by each agent sending to its neighbors messages that include the cost for each of the receiving agents' possible preferences. The agents collect the sum of all such messages and are able to construct local functions $z_i(x_i)$:

$$z_i(x_i) = \sum_{k \in N_i} m_{k\to i}(x_i) \tag{3.4.4}$$

The function then terminates when some stopping function is met. [5]

Algorithm 5 Max-Sum

```
1: Initialize each variable, $x_i \in X$ to random $d \in D_i$
2: Initialize all messages, $m_{i\to j}$, to constant functions
3: for Each Activation Step do
4:     for Each Agent $A_i$ do
5:         for Each Agent $A_j \neq A_i$ do
6:             Compose a message 3.4.3
7:         end for
8:         Compute that local function $z_i(x_i)$ 3.4.4
9:         Find the argument max value given the local function:
               $x_i^* = arg \max_{x_i} z_i(x_i)$
10:    end for
11: end for
```

The Max-Sum technique is guaranteed to solve the DCOP optimally on acyclic structures, however, if applied to general graphs there is no guarantee. However empirically, despite this lack of convergence max-sum does generally give a good approximation on cyclic graphs. In fact on cyclic graphs of various cycle sizes, it does not converge. [16]

Max-Sum messages are small (scale with domain) and the number of messages exchanged generally varies with the number of agents in the graph. Once again, it does not guarantee convergence.

3.5 Evaluation of Solvers

The solvers will be evaluated on three major criteria. Each will be run for n steps where $n = 100$ and we will compare the quality of the solutions, or the total utility of the constraint network.

The solvers will then be evaluated on runtime, and complexity. Runtime being the average runtime over different topologies for 50 trials. As for complexity, the paper will explore the number of messages that are passed in order to judge complexity.

4. RESULTS

The results of the paper are generally empirical, however the paper will try to provide some intuition as to why one sees these results. Then the paper will suggest a set of guidelines for which solvers to use given an underlying knowledge of the network in terms of characteristic path length and clustering coefficients. This can be further used to show the best results given that one can choose topology and solver.

We used both the cooperative game and the anti-cooperative game. The results from the cooperative game showed that all solvers found the optimal solution on average on the second step, thus are not included. Below we have the results for the anti-cooperative game.

Figure 1: Graph Value vs. Step: DSA

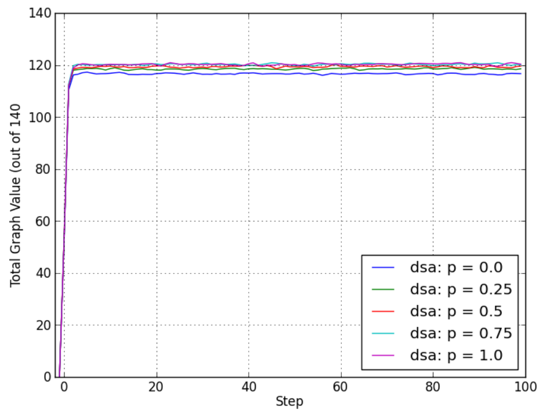
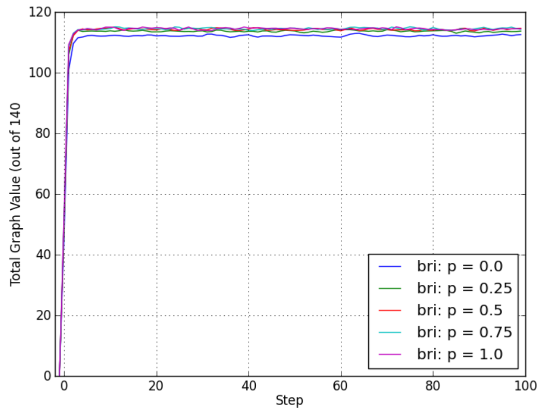
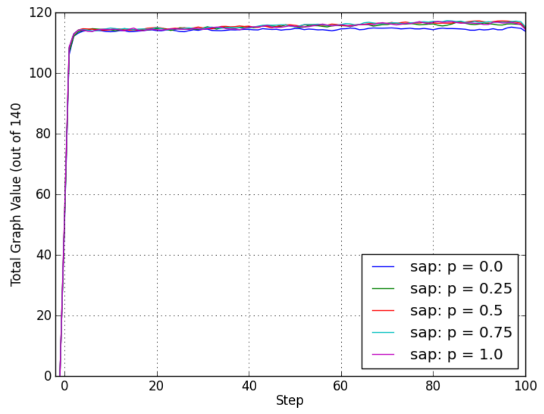


Figure 2: Graph Value vs. Step: BR-I



We ran all five of the solvers on the same Watts-Strogatz generated graphs for 50 trials, keeping track of the graph value at each step. The graphs were 40 nodes of degree 4, meaning a total graph value of 160 being possible (graphs were discarded if not 3-colorable). For each parameter p of the Watts-Strogatz model we created 5 graphs. Given that we evaluated 21 different parameter values (zero through one in increments of .05), this means we ran a total of over 26,000 trials.

Figure 3: Graph Value vs. Step: SAP



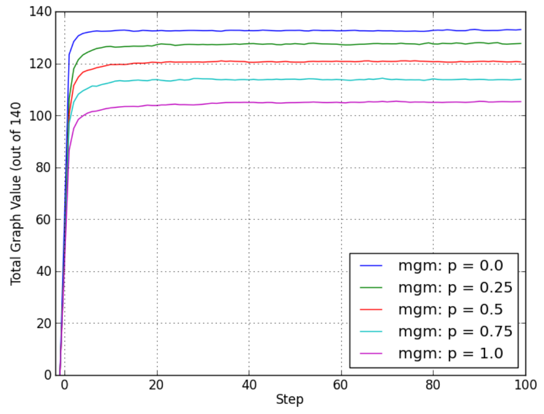
Below we have shown the graph values of select trials (p values of $\{0, .25, .5, .75, 1\}$) in order to better see the trends. Let us first examine the graphs of DSA, BR-I, and SAP.

Notice first that DSA have preformed the best out of the three of them, with an average graph value of 120 (out of 160). The other solvers preform on average less than 5 percent worse.

The second notable trend to see, is that when $p = 0$, meaning high clustering and high path length, the solvers preform generally worse, though by a small degree.

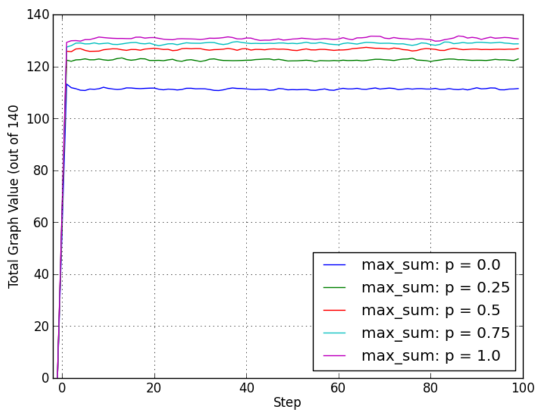
The final trend that is worth pointing out, is that there is very little trend to be seen. In general, regardless of clustering the algorithms preform nearly the same. This should not be surprising that DSA, BR-I, and SAP all preform nearly the same, they all are greedy algorithms with the same update schedule. Now let us look at the two standouts.

Figure 4: Graph Value vs. Step: MGM



These two graphs show completely different trends. First notice that when acting on a favorable topology, both algorithms score better than DSA, BR-I, and SAP. Note also that there is considerable difference in performance between

Figure 5: Graph Value vs. Step: Max-Sum



different graph types.

Let us try to account for this. MGM does much better on graphs with high clustering and high path length. A possible reason for this MGM's update schedule. MGM collects information about its local neighborhood, even more than DSA, BR-I, or SAP. Thus if its local neighborhood is rich and well connected, the information exchange could be more fruitful. In fact it may be that a network's local communication 2.1.5 could determine how well MGM would do.

Max-Sum on the other hand, show the opposite trend. On networks with high characteristic path length ($p = 0$) it preforms very poorly, while on the rest of the topologies preforms quite well. We will try to provide intuition for this. First we could propose that Max-Sum is able to pass messages, thus the more easily that the messages propagate through the entire network the better. Thus Max-Sum's performance could be based on the networks global communication 2.1.4. However there is more likely a simpler explanation for this phenomena. The highly clustered model in Watts-Strogatz is completely cyclical. And we know that Max-Sum preforms worse on these types of networks. Thus as randomness increases, so to does the cyclic nature of the graph degrade.

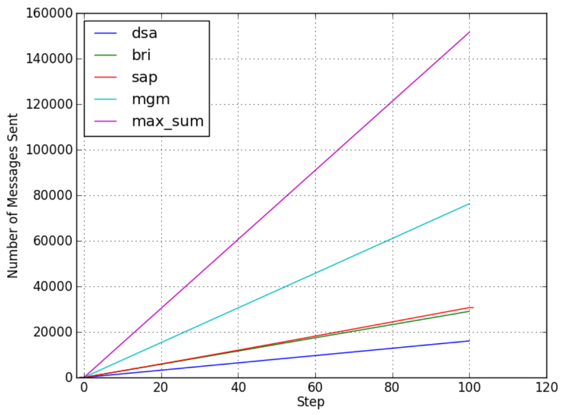
As a final note, we studied the runtime and the complexity of the various solvers. Below we have the average runtime for 50 trials each solver over all of the graphs.

50 Trials:	DSA	BR-I	SAP	MGM	Max-Sum
Seconds	17.65	52.64	49.42	175.74	84.36

One can easily see the tradeoffs made. Max-Sum or MGM take 2 or 4 times as long to complete vs. BR-1 or SAP. While DSA is nearly one third of BR-I and SAP. These timings are not considering code optimization that could have been implemented.

For the complexity of each algorithm we have looked at the number of messages passed. Once again this is not considering optimizations that could have been made. We see below that here Max-Sum uses twice the messages than MGM. While MGM uses twice the messages of SAP/BR-I and DSA, coming in lowest in terms of overhead again, uses half the messages of BR-I/SAP.

Figure 6: Messages vs. Step: All Solvers



On one further note, the messages of Max-Sum are nine times (the number of domains) the size of the messages from the other constraint solvers.

With this knowledge of both MGM and Max-Sum, it is easy to see what tradeoffs were made for their high performance.

5. CONCLUSION

We have learned some relatively useful facts. We leaned that MGM and Max-Sum do well on highly clustered and low characteristic path length networks respectively. But that each, MGM and Max-Sum, has tradeoff, most importantly time and message size/memory respectively.

We have also learned that of DSA, BR-I, and SAP, DSA has the lowest overhead, and all preform equally well, regardless of the Small World property being present.

We had stipulated that both Jin & Liu and Walsh et. alia made separate claims about how topology (high clustering and low characteristic path length) affects constraint solvers. Jin & Liu hypothesized that it would help. While Walsh hypothesized that it would hurt. Based on our results we can conclude that both are right and wrong.

High clustering can help MGM, yet hurt Max-Sum. Low characteristic path can help Max-Sum, yet hurt MGM. And DSA, BR-I, and SAP are indifferent.

Though the ultimate conclusion of this paper, is that there is much future work to do.

6. FUTURE WORK

The biggest question for future work, will be to detangle Watts-Strogatz from high global and local communication. Thus we could test our hypothesis: MGM preforms better with high local communication and Max-Sum preforms better with high global communication. If we model other graphs with these properties, then we could detangle any affects from the cyclic nature of Watts-Strogatz etc.

In addition, looking at non-uniform constraint functions might have very interesting effects on DSA, BR-I and SAP. And we would suspect that highly varied constraint functions could hurt DSA, and help the others.

Of course optimization of the code, and making the agents run in parallel could improve our results by leaps and bounds.

7. APPENDIX

For code used to produce results please see https://github.com /knathanieltucker/constraint-network

8. REFERENCES

[1] R. M. Araujo and L. C. Lamb. Memetic networks: Analyzing the effects of network properties in multi-agent performance. In *AAAI*, pages 3–8, 2008.

[2] J. Atlas and K. Decker. Coordination for uncertain outcomes using distributed neighbor exchange. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 1047–1054. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[3] A. C. Chapman, A. Rogers, and N. R. Jennings. Benchmarking hybrid algorithms for distributed constraint optimisation games. *Autonomous Agents and Multi-Agent Systems*, 22(3):385–414, 2011.

[4] R. J. Farinelli, Vinyals. *Distributed Constraint Handling and Optimization.*

[5] A. Grubshtein, R. Zivan, T. Grinshpoun, and A. Meisels. Local search for distributed asymmetric optimization. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 1015–1022. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[6] X. Jin and J. Liu. Agent network topology and complexity. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 1020–1021. ACM, 2003.

[7] X. Jin and J. Liu. Efficiency of emergent constraint satisfaction in small-world and random agent networks. In *Intelligent Agent Technology, 2003. IAT 2003. IEEE/WIC International Conference on*, pages 304–310. IEEE, 2003.

[8] C. Kiekintveld, Z. Yin, A. Kumar, and M. Tambe. Asynchronous algorithms for approximate distributed constraint optimization with quality bounds. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 133–140. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[9] A. Rogers, A. Farinelli, R. Stranders, and N. R. Jennings. Bounded approximate decentralised coordination via the max-sum algorithm. *Artificial Intelligence*, 175(2):730–759, 2011.

[10] J. C. Spall. *Introduction to stochastic search and optimization: estimation, simulation, and control*, volume 65. Wiley. com, 2005.

[11] M. E. Taylor, M. Jain, Y. Jin, M. Yokoo, and M. Tambe. When should there be a me in team?: distributed multi-agent optimization under uncertainty. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 109–116. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[12] T. Walsh et al. Search in a small world. In *IJCAI*, volume 99, pages 1172–1177, 1999.

[13] D. J. Watts and S. H. Strogatz. Collective dynamics of âĂŸsmall-worldâĂŹnetworks. *nature*, 393(6684):440–442, 1998.

[14] H. Yedidsion, R. Zivan, and I. Beer-Sheva. Applying maxsum to dcop mst.

[15] H. L. Zhang, C. H. Leung, and G. K. Raikundalia. Topological analysis of aocd-based agent networks and experimental results. *Journal of Computer and System Sciences*, 74(2):255–278, 2008.

[16] R. Zivan and H. Peled. Max/min-sum distributed constraint optimization through value propagation on an alternating dag. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 265–272. International Foundation for Autonomous Agents and Multiagent Systems, 2012.