Communication and Computation in Distributed CSP Algorithms*

Cèsar Fernàndez[1], Ramón Béjar[1], Bhaskar Krishnamachari[2], and Carla Gomes[2]

[1] Departament d'Informàtica i Enginyeria Industrial, Universitat de Lleida
Jaume II, 69, E-25001 Lleida, Spain
{ramon,cesar}@eup.udl.es
[2] Department of Computer Science, Cornell University
Ithaca, NY 14853, USA
{bhaskar,gomes}@cs.cornell.edu

Abstract. We introduce SensorDCSP, a naturally distributed benchmark based on a real-world application that arises in the context of networked distributed systems. In order to study the performance of Distributed CSP (DisCSP) algorithms in a truly distributed setting, we use a discrete-event network simulator, which allows us to model the impact of different network traffic conditions on the performance of the algorithms. We consider two complete DisCSP algorithms: asynchronous backtracking (ABT) and asynchronous weak commitment search (AWC). In our study of different network traffic distributions, we found that, random delays, in some cases combined with a dynamic decentralized restart strategy, can improve the performance of DisCSP algorithms. More interestingly, we also found that the *active introduction of message delays by agents can improve performance and robustness, while reducing the overall network load.* Finally, our work confirms that AWC performs better than ABT on satisfiable instances. However, on unsatisfiable instances, the performance of AWC is considerably worse than ABT.

1 Introduction

In recent years we have seen an increasing interest in Distributed Constraint Satisfaction Problem (DisCSP) formulations to model combinatorial problems arising in distributed, multi-agent environments [2, 14, 16, 17, 18, 20]. There is a rich

* Research supported by AFOSR, grant F49620-01-1-0076 (Intelligent Information Systems Institute) and F49620-01-1-0361 (MURI grant on Cooperative Control of Distributed Autonomous Vehicles in Adversarial Environments), CICYT, TIC2001-1577-C03-03 and DARPA, F30602-00-2-0530 (Controlling Computational Cost: Structure, Phase Transitions and Randomization) and F30602-00-2-0558 (Configuring Wireless Transmission and Decentralized Data Processing for Generic Sensor Networks). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFOSR, DARPA, or the U.S. Government.

set of real-world distributed applications, such as in the area of networked systems, for which the DisCSP paradigm is particularly useful. In such distributed applications, constraints among agents, such as communication bandwidth and privacy issues, preclude the adoption of a centralized approach.

We propose SensorDCSP, a benchmark inspired by one of such distributed applications that arise in networked distributed systems [1, 8]. SensorDCSP is a truly distributed benchmark, a feature not present in many prior benchmark problems used to study the performance of DisCSP algorithms, such as N-Queens and Graph Coloring. SensorDCSP involves a network of distributed sensors simultaneously tracking multiple mobile nodes. The problem underlying SensorDCSP is NP-complete. We show that the SensorDCSP domain undergoes a phase transition in satisfiability, with respect to two control parameters: the level of sensor compatibility and the level of the sensor visibility. Standard DisCSP algorithms on our SensorDCSP domain exhibit the easy-hard-easy profile in complexity, peaking at the phase transition, similarly to the pattern observed in centralized CSP algorithms. More interestingly, the relative strength of standard DisCSP algorithms on SensorDCSP is highly dependent on the satisfiability of the instances. This aspect has been overlooked in the literature due to the fact that, so far, the performance of DisCSP algorithms has been evaluated mainly on satisfiable instances. We study the performance of two well known DisCSP algorithms – asynchronous backtracking (ABT) [18], and asynchronous weak commitment search (AWC) [17]– on SensorDCSP. Both ABT and AWC use agent priority ordering during the search process. While these priorities are static in ABT, AWC allows for dynamic changes in the ordering, and was originally proposed as an improvement over ABT. One of our findings is that although AWC does indeed perform better than ABT on satisfiable instances, its performance is not as good on unsatisfiable problem instances.

Our SensorDCSP benchmark also allows us to study other interesting aspects specific to DisCSPs that are dependent on the physical characteristics of the distributed environment. For example, while the underlying infrastructure or hardware is not critical in studying CSPs, we argue that this is not the case for DisCSPs in communication networks. This is because the traffic patterns and packet-level behavior of networks, which affect the order in which messages from different agents are delivered to each other, can significantly impact the distributed search process. To investigate these kinds of effects, we implemented our DisCSP algorithms using *a fully distributed discrete-event network simulation environment* with a complete set of communication oriented classes. The network simulator allows us to realistically model the message delivery mechanisms of varied distributed communication environments ranging from wide-area computer networks to wireless sensor networks.

We study the impact of communication delays on the performance of DisCSP algorithms. We consider different link delay distributions. Our results show that the presence of a random element due to the delays can improve the performance of AWC. For the basic ABT, even though link delay deteriorates the performance of the standard algorithm, a decentralized restart strategy that we developed for

ABT improves its solution time dramatically, while also increasing the robustness of solutions with respect to the variance of the network link delay distribution. These results are consistent with results on successful randomization techniques developed to improve the performance of CSP algorithms [4]. Another novel aspect of our work is the introduction of a mechanism for *actively* delaying messages. The active delay of messages decreases the communication load of the system, and, somewhat counter-intuitively, can also decrease the overall solution time.

The organization of the rest of the paper is as follows. In Section 2 we formalize our model of DisCSP. In Section 3 we describe SensorDCSP and model it as a DisCSP. In Section 4 we describe two standard DisCSP algorithms and the modifications we have incorporated into the algorithms. In Section 5 we present our experimental results on the active introduction of randomization by the agents and, in Section 6, we present results on delays caused by different traffic conditions in the communication network. Finally, we present our conclusions in Section 7.

2 Distributed CSPs

In a distributed CSP, variables and constraints are distributed among the different autonomous agents that have to solve the problem. A DisCSP is defined as follows: (1) A finite set of agents $A_1, A_2, \cdots, A_n$; (2) A set of local (private) CSPs $P_1, P_2, \cdots, P_n$, where the CSP P_i belongs to agent A_i; A_i is the only agent that can modify the value assigned to the variables of P_i; (3) A global CSP defined among variables that belong to different agents.

In general in DisCSP algorithms each agent only controls one variable. We extended the single-variable approach by making every agent consist of multiple virtual agents, each corresponding to one local variable. In order to distinguish between communication and computation costs in our discrete event simulator, we use different delay distributions to distinguish between messages exchanged between virtual agents of a single real agent (intra-agent messages) and those between virtual agents of different real agents (inter-agent messages).

3 SensorDCSP – A Benchmark for DisCSP Algorithms

The availability of a realistic benchmark of satisfiable and unsatisfiable instances, with tunable complexity, is critical for the study and development of new search algorithms. In the DisCSP literature one cannot find such a benchmark. SensorDCSP, the sensor-mobile problem, is inspired by a real distributed resource allocation problem [13] and offers such desirable characteristics.

In SensorDCSP we have multiple sensors $(s_1, \ldots s_m)$ and multiple mobiles $(t_1, \ldots t_n)$ which are to be tracked by the sensors. The goal is to allocate three distinct sensors to track each mobile node, subject to two sets of constraints: visibility constraints and compatibility constraints. Figure 1 shows an example with six sensors and two mobiles.

Each mobile has a set of sensors that can possibly detect it, as depicted by the bipartite visibility graph in the leftmost panel of Figure 1. Also, it is required that each mobile be assigned three sensors that satisfy a compatibility relation with each other; this compatibility relation is depicted by the graph in the middle panel of Figure 1. Finally, it is required that each sensor only track at most one mobile. A possible solution is shown in the right panel, where the set of three sensors assigned to every mobile is indicated by connecting them to the mobile with the light edges of the figure.
This problem is NP-complete since we can reduce it from the problem of partitioning a graph into cliques of size three [1, 6]. However, the boundary case where every pair of sensors is compatible, is polynomially solvable, since we can reduce that case to a feasible flow problem in a bipartite graph [7].

We define a random distribution of instances of SensorDCSP. An instance of the problem is generated from two different random graphs, the visibility graph and the compatibility graph. Apart from the parameters number of mobiles and number of sensors, we also specify a parameter that controls the edge density of the visibility graph (P_v) and a second one that controls the edge density for the compatibility graph (P_c). These parameters specify the independent probability of including a particular edge in the corresponding graph. As these two graphs model the resources available to solve the problem, P_v and P_c control the number of constraints in the generated instances.

We have developed an instance generator for these random distributions that generates DisCSP-encoded instances. We believe that SensorDCSP is a good benchmark problem because of the simplicity of the generator, and because, as we shall show, one can easily generate easy/hard, unsatisfiable/satisfiable instances by tuning the parameters P_v and P_c appropriately.

We encoded SensorDCSP as a DisCSP as follows: each mobile is associated with a different agent. There are three different variables per agent, one for each sensor that we need to allocate to the corresponding mobile. The value domain of each variable is the set of sensors that can detect the corresponding mobile. The intra-agent constraints between the variables of one agent are that the three sensors assigned to the mobile must be different and must be pair-wise compatible. The inter-agent constraints between the variables of different agents are that a given sensor can be selected by at most one agent. In our implementation

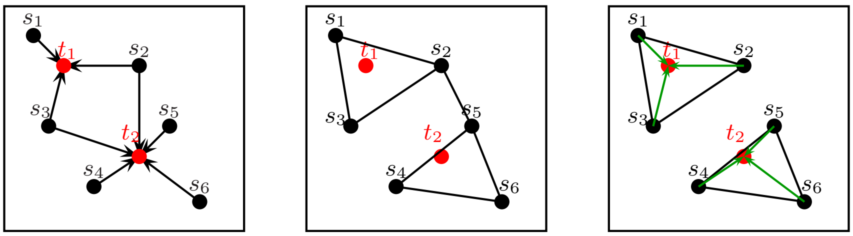


Fig. 1. A SensorDCSP problem instance

of the DisCSP algorithms this encoding is translated to an equivalent formulation where we have three virtual agents for every real agent, each virtual agent handling a single variable.

4 DisCSP Algorithms

In the work reported here we considered two specific DisCSP algorithms, Asynchronous Backtracking Algorithm (ABT), and Asynchronous Weak-Commitment Search Algorithm (AWC). We provide a brief overview of these algorithms but refer the reader to [20] for a more comprehensive description. We also describe the modifications that we introduced to these algorithms. As mentioned before, we assume that each agent can only handle one variable. The neighbors of an agent A_i refer to the set of agents that share constraints with A_i.

The **Asynchronous Backtracking Algorithm (ABT)** is a distributed asynchronous version of a classical backtracking algorithm. This algorithm needs a static agent ordering that determines an ordering between the variables of the problem. Agents use two kinds of messages for solving the problem – *ok* messages and *nogood* messages. Agents initiate the search by assigning an initial value to their variables. An agent changes its value when it detects that it is not consistent with the assignments of higher priority neighbors, and so it maintains an agent view, which consists of the variable assignments of its higher priority neighbors.

Each time an agent assigns a value to its variable, it issues the *ok* message to inform its set of lower priority neighbors about this new assignment. When an agent is not able to find an assignment consistent with its higher priority neighbors, it sends a *nogood* message to the lowest priority agent among the agents that have variables in the *nogood*. A *nogood* message consists of a subset of the agent view that does not permit the agent to find a consistent assignment for itself. A *nogood* message causes the receiver agent to record the received *nogood* as a new constraint and to try to find an assignment consistent with its higher priority neighbors and with all the recorded constraints. If the top-priority agent is forced to backtrack, because it cannot fix the problem by asking a higher priority neighbor to change its assignment, this means that the problem has no solution. On the other hand, when the system reaches a state where all agents are happy with their current assignments (no *nogood* messages are generated), this means that the agents have found a solution.

The **Asynchronous Weak-Commitment Search Algorithm (AWC)** can be seen as a modification of the ABT algorithm. The primary differences are as follows. A priority value is determined for each variable, and the priority value is communicated using the *ok* message. When the current assignment is not consistent with the agent view, the agent selects a new consistent assignment that minimizes the number of constraint violations with lower priority neighbors. When an agent cannot find a consistent value and generates a new *nogood*, it sends the *nogood* message to all its neighbors, and increases its priority one unit over the maximal priority of its neighbors. Then, it finds a value consistent with higher priority neighbors and informs its neighbors with *ok* messages. If no new *nogood* can be generated, the agent waits for the next message.

The most obvious way of introducing randomization in DisCSP algorithms is by randomizing the value selection strategy used by the agents. In the ABT algorithm this is done by performing a uniform random value selection, among the set of values consistent with the agent view and the *nogood* list, every time the agent is forced to select a new value. In the AWC algorithm, we randomize the selection of the value among the values consistent with the agent view and the nogood list, and that minimize the number of violated constraints. This form of randomization is analogous to the randomization techniques used in backtrack search algorithms.

A novel way of randomizing the search, relevant in the context of DisCSP algorithms, is by introducing forced delays in the delivery of messages. Delays introduce randomization because the order in which messages arrive to the target agents determines the order in which the search space is traversed. More concretely, every time an agent has to send a message, it follows the following procedure:

```
1. With probability $p$:
     $d$ := r;
   else (with probability $(1 - p)$)
     $d$ := 0;
2. deliver the message with delay $d$
```

By delivering a message with delay d we mean that the agent informs its communication interface that it should wait d seconds before delivering the message through the communication network. The parameter r is the fraction of the mean communication delay added by the agent. In our implementation of the algorithms, this strategy is performed by using the services of the discrete event simulator that allow specific delays to be applied selectively in the delivery message queue of each agent.

We have also developed the following decentralized restarting strategy suitable for the ABT algorithm: the highest priority agent uses a timeout mechanism to decide when a restart should be performed. It performs the restart by changing its value at random from the set of values consistent with the *nogoods* learned so far. Then, it sends *ok* messages to its neighbors, thus producing a restart of the search process, but without forgetting the *nogoods* learned. This restart strategy is different from the restart strategy used in centralized procedures, such as rand-satz [4], because the search is not restarted from scratch, but rather benefits from prior mistakes since all agents retain the *nogoods*.

5 Complexity Profiles of DisCSP Algorithms on SensorDCSP

As mentioned earlier, when studying distributed algorithms it is important to factor in the physical characteristics of the distributed environment. For example, the traffic patterns and packet-level behavior of networks can affect the order in which messages from different agents are delivered to each other, significantly impacting the distributed search process. To investigate these kinds of effects, we have developed an implementation of the algorithms ABT and AWC using the Communication Networks Class Library (CNCL) [5]. This library provides a discrete-event network simulation environment with a complete set of communication oriented classes. The network simulator allows us to realistically model the message delivery mechanisms of varied distributed communication environments ranging from wide-area computer networks to wireless sensor networks.

The results shown in this section have been obtained according to the following scenario. The communication links used for communication between virtual agents of different real agents (inter-agent communication) are modeled as random negative exponential distributed delay links, with a mean delay of 1 time unit. The communication links used by the virtual agents of a real agent (intra-agent communication) are modeled as fixed delay links, with a delay of 10^{-3} time units. We use fixed delay links because we consider that a set of virtual agents work inside a private computation node that allows them to communicate with each other with dedicated communication links. This scenario could correspond to a heavy load network situation where inter-agent delay fluctuations obey to the queuing time process on intermediate systems. The factor of 1000 difference between the two delays reflects that usually intra-agent computation is less expensive that inter-agent communication. In the last section of the paper we will see how different delay distribution models over the inter-agent communication links can impact the performance of the algorithms.
For our experimental results, we considered different sets of instances with 3 mobiles and 15 sensors, with every set generated with different values for the parameters P_c and P_v, ranging from 0.1 to 0.9. Every set contains 19 instances, giving a total number of 81 data points. Each instance has been executed 9 times with different random seeds. The results reported in this section were obtained using a sequential value selection function for the different algorithms.

Figure 2 shows the ratio of satisfiable instances as a function of P_c and P_v. When both probabilities are low, the instances generated are mostly unsatisfiable. On the other hand, for high probabilities most of the instances are satisfiable. The transition between the satisfiable and unsatisfiable regions occurs

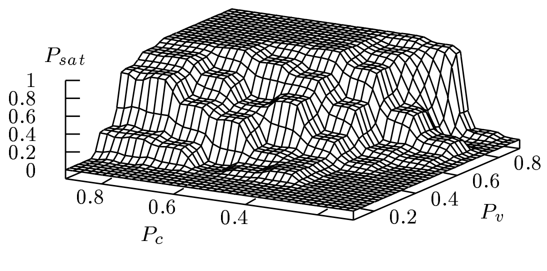


Fig. 2. Ratio of satisfiable instances depending on the density parameter for the visibility graph (P_v) and the density parameter for the compatibility graph (P_c)

within a relatively narrow range of these control parameters, analogous to the phase transition in CSP problems, *e.g.*, in SAT [10].

Also consistent with general CSP problems, we observe that the phase transition coincides with the region where the hardest instances occur. Figure 3 shows the mean solution time with respect to the parameters P_c and P_v. As can be noted, the hardest instances lie on the diagonal that defines the phase transition zone, with a peak for instances with a low P_c value. The dark and light solid lines overlaid on the mesh depict the location of the iso-lines for $P_{sat} = 0.2$ and $P_{sat} = 0.8$, respectively, as per the phase transition surface of Figure 2. As mentioned before, the SensorDCSP problem is NP-complete only when not all the sensors are compatible between them ($P_c < 1$) [7], so the parameter P_c could separate regions of different mean computational complexity, as in other mixed P/NP-complete problems like 2+p-SAT [10] and 2+p-COL [15]. This is particularly visible in the mean time distribution for AWC in Figure 3.
We observe that the mean times to solve an instance appear to be larger by an order of magnitude for AWC than for ABT. At first glance, this is a surprising result considering that the AWC algorithm is a refinement of ABT and results reported for satisfiable instances in the literature [19, 20] conclude on a better performance for AWC. The explanation for such a discrepancy is the fact that our results deal with both satisfiable and unsatisfiable instances. Our further investigations showed that while AWC does indeed outperform ABT on satisfiable

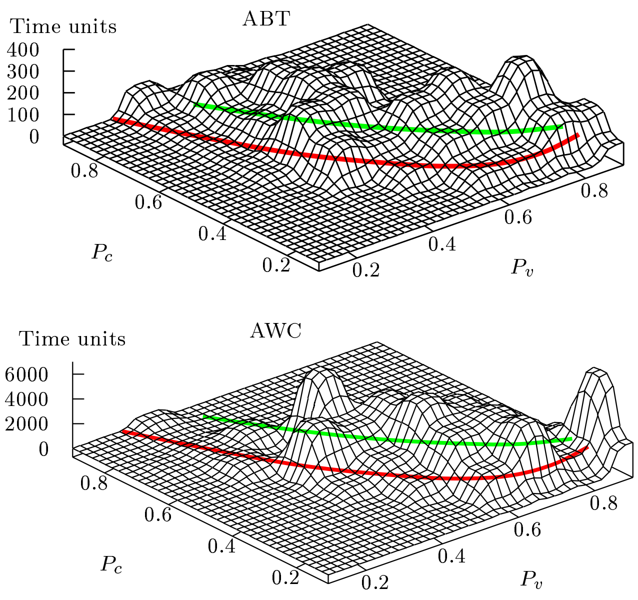


Fig. 3. Mean solution time with respect to P_v and P_c for ABT and AWC algorithms

instances, it is much slower on unsatisfiable instances. This result seems consistent with the fact that the agent hierarchy on ABT is static, while for AWC, such a hierarchy changes during problem solving, taking more time to inspect all the search space when unsatisfiable instances are considered.

5.1 Randomization and Restart Strategies

In this section we describe experimental results that demonstrate the effect of adding a restart strategy to ABT. The introduction of a randomized value selection function was directly assumed in [19]. In extensive experiments we have performed with our test instances, we noticed that the randomized selection function is indeed better than a fixed selection function. However, as the randomization can introduce more variability in the performance, ABT should be equipped with a restart strategy. We have not defined a restart strategy for AWC, because, as we will see in the last section, the dynamic priority strategy of AWC can be viewed as a kind of built-in partial restart strategy. In the results reported in the rest of the paper both ABT and AWC use randomized value selection functions.

To study the benefits of the proposed restart strategy for ABT, we have solved hard satisfiable instances with ABT with restarts, using different cutoff times. Figure 4 shows the mean time needed to solve a hard satisfiable instance with the corresponding 95% confidence intervals for different cutoff times. We observe clearly that there is an optimal restart cutoff time that gives the best performance. As we will discuss in the last section, when considering the delays of real communication networks, the use of restart strategies becomes a requirement, given the high variance in the solution time due to randomness of link delays in the communication network.

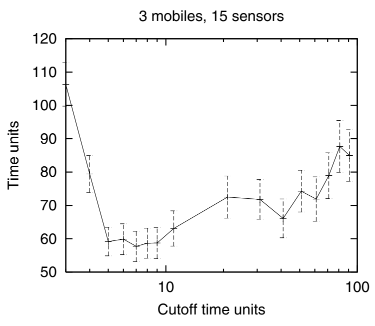


Fig. 4. Mean time to solve a hard satisfiable instance by ABT using restarts with different cutoff times

5.2 Active Delaying of Messages

A novel way of randomizing a DisCSP algorithm corresponds to introducing delays in the delivery of the agents' outgoing messages, as we described in Section 4. In this section we describe our experimental results using AWC, where the amount of delay added by the agents is a fraction r (from 0 to 1) of the fixed delay on the inter-agent communication links. In other words, we consider that all the inter-agent communication links have fixed delays, of 1 time unit, in contrast to what we did in the previous sections, because we want to isolate the effect of the delay added by the agents.

Figure 5 shows the results for a hard satisfiable instance from our SensorDCSP domain, for different values of p, the probability of adding a delay, and r, the fraction of delay added with respect to the delay of the link. We have that the difference in performance in number of messages can be as high as 3 times between the best case and the worst case. The horizontal plane cutting the surface shows the median time needed by the algorithm when we consider no added random delays ($p = 0, r = 0$). We see that agents can indeed improve the performance by actively introducing some additional random delays, when exchanging messages. We also observe that the performance in number of messages is almost always improved when agents add random delays. Perhaps more surprisingly, in terms of the total solution time, the performance can also improve, if the increase in delay r is not too high. The reason could be the ability of AWC to exploit randomization during the search process due to its inherent restarting strategy.

6 The Effect of the Communication Network Data Load

As described in the previous section, when working on a communication network with fixed delays, the performance of AWC can be improved, depending on the amount of random delay addition that the agents introduce into the message delivery system. However, in real networks, the conditions of data load present in the communication links used by the agents cannot always be modeled with

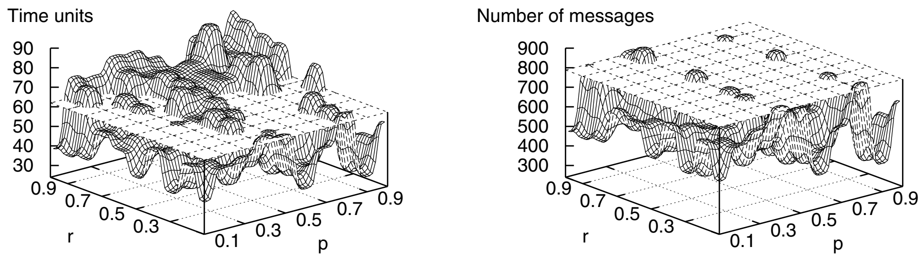


Fig. 5. Median time and number of messages to solve a hard satisfiable instance when agents add random delays in outgoing messages. The horizontal plane represents the median time when no delay is added ($p = 0$)

fixed delay links. It is worthwhile understanding how different communication network environments can impact the performance of the algorithms. In this section we study the effect produced in the performance of DisCSP algorithms by considering delay distributions corresponding to different traffic conditions.

For the results of Section 5.2 we considered inter-agent communication links with random exponentially distributed delays. To study how exponentially distributed delays affect the performance with respect to fixed delays, we can consider intermediate situations in which some of the inter-agent links have a fixed delay and the rest are exponentially distributed.

Figure 6 shows the median number of messages and time needed by AWC for solving a hard satisfiable instance with 4 mobiles and 15 sensors, when we vary the percentage of inter-agent communication links with a fixed delay. The rest of the inter-agent communication links are assumed to have random exponentially distributed delays.
The performance of AWC is worst when 100% of the links have a fixed delay, indicating that the conditions of the network clearly affect the performance of the algorithm: a element of randomness in the delay distributions clearly improves the performance of AWC. Observe that we have a clear correlation between the number of messages and time needed, meaning that the increase or decrease in the time needed is mainly because of the change in the number of messages exchanged.

We now examine various link delay distributions that can be used to model communication network traffic. Traditionally, exponential negative distributed inter-arrival times have been used to model data traffic due to their attractive theoretical properties, but in the past decade it has been shown that, although these models are able to capture single user sessions properties, they are no longer suitable for modeling aggregate data links in local or wide area network scenarios[3, 9, 11]. Facing this fact, we simulate network delays according to

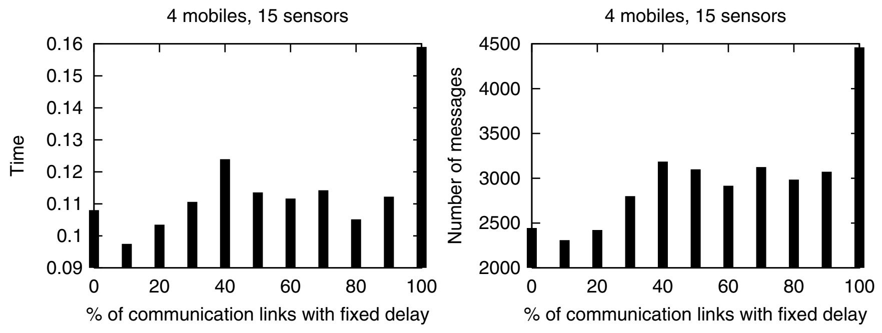


Fig. 6. Median number of messages and time exchanged to solve a hard satisfiable instance by AWC when the data load is not homogeneous among all the inter-agent communication links

three different models for the inter-arrival time distribution; the above mentioned exponential negative distribution, the log-normal distribution and the Fractional Gaussian Noise (FGN)[12].

The log-normal distribution is useful to obtain distributions with any desired variance, whereas FGN processes are able to capture crucial characteristics of the Internet traffic as long-range dependence and self-similarity that are not suited by other models. We synthesize FGN from α-stable distributions with parameters $H = 0.75$ and $d = 0.4$,

Figure 7 shows the cumulative density functions (CDF) of time required to solve hard instances for AWC, ABT, and ABT with restarts, when all the inter-agent communication links have delays modeled as fixed, negative exponential, and log-normal, with identical mean and different variances.

Table 1 and 2 show the estimated mean and variance of the number of messages exchanged as well as the solution time for the different cases when the same instance is used for three algorithms.
The results in Figure 7 and Tables 1 and 2 show that the delay distributions have an algorithm-specific impact on the performance of the basic ABT and on AWC.

For the basic ABT, on hard instances, the solution time becomes worse when channel delays are modeled by random distributions as opposed to the fixed delay case. The greater the variance of the link delay, the worse ABT performs. However, introducing the restart strategy has the desirable effect of improving

Table 1. Statistics estimated from the distributions of number of messages with different inter-agent link delay models

Delay distribution	Mean			Variance		
	ABT	ABT-rst	AWC	ABT	ABT-rst	AWC
Fixed	$1.8 \cdot 10^5$	$1.2 \cdot 10^5$	$8.2 \cdot 10^2$	$3.6 \cdot 10^{10}$	$1.3 \cdot 10^{10}$	$3 \cdot 10^5$
Negative expon. ($\sigma^2 = 1$)	$1.7 \cdot 10^5$	$1.5 \cdot 10^5$	$3.5 \cdot 10^2$	$2.8 \cdot 10^{10}$	$0.9 \cdot 10^{10}$	$4.5 \cdot 10^5$
Log-normal ($\sigma^2 = 5$)	$2.2 \cdot 10^5$	$1.3 \cdot 10^5$	$3.5 \cdot 10^2$	$5.0 \cdot 10^{10}$	$1.7 \cdot 10^{10}$	$4.8 \cdot 10^5$
Log-normal ($\sigma^2 = 10$)	$2.6 \cdot 10^5$	$1.6 \cdot 10^5$	$3.5 \cdot 10^2$	$7.1 \cdot 10^{10}$	$2.4 \cdot 10^{10}$	$4.9 \cdot 10^5$

Table 2. Statistics estimated from the distributions of time to solve in time units with different inter-agent link delay models

Delay distribution	Mean			Variance		
	ABT	ABT-rst	AWC	ABT	ABT-rst	AWC
Fixed	98	69	53	8562	3600	1230
Negative expon. ($\sigma^2 = 1$)	111	71	28	10945	3947	266
Log-normal ($\sigma^2 = 5$)	157	103	28	21601	8438	288
Log-normal ($\sigma^2 = 10$)	188	131	28	30472	13423	402

the performance of ABT. Furthermore, ABT with restarts is fairly robust and insensitive to the variance in the link delays.

AWC behaves differently from the basic ABT. On hard instances, having randomization in the link delays improves the solution time compared to the fixed delay channel. Further, the mean solution time for AWC is extremely robust to the variance in communication link delays, although the variance of solution time is slightly affected by this.

Experiments run with FGN delay models show no significant differences in performance for the three algorithms in relation to other traffic models with the same variance.

In general, we found that on satisfiable instances, AWC always performs significantly better than ABT, even ABT with restart. Thus AWC appears to be a better candidate in situations when most instances are likely to be satisfiable.

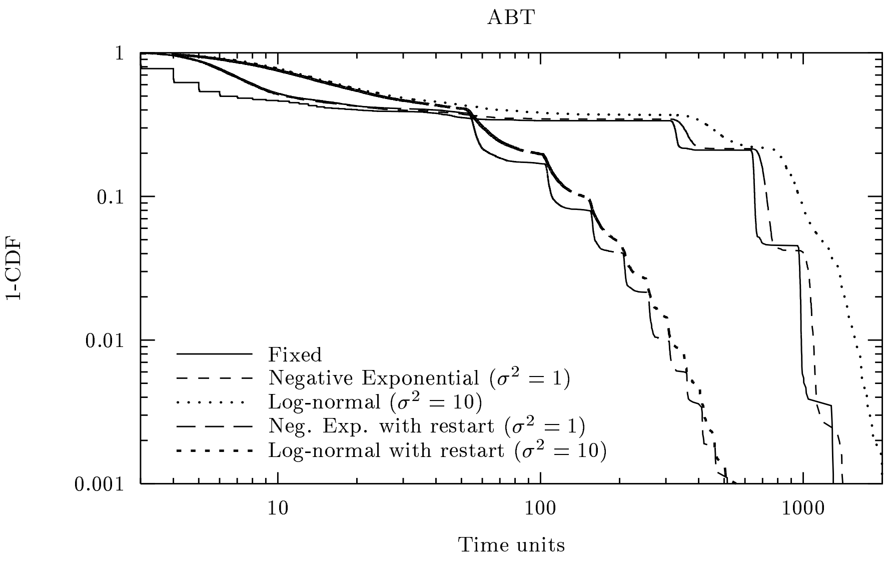
7 Conclusions

We introduce SensorDCSP, a benchmark that captures some of the characteristics of real-world distributed applications that arise in the context of distributed networked systems. The two control parameters of our SensorDCSP generator, sensor compatibility (P_c) and sensor visibility (P_v), result in a zero-one phase transition in satisfiability.

We tested two complete DisCSP algorithms, synchronous backtracking (ABT) and asynchronous weak commitment search (AWC). We show that the phase transition region of SensorDCSP induces an easy-hard-easy profile in the solution time, both for ABT and AWC, which is consistent with CSPs. We found that AWC performs much better than ABT on satisfiable instances, but worse on unsatisfiable instances. This differential in performance is most likely due to the fact that on unsatisfiable instances, the dynamic priority ordering of AWC slows the completion of the search process.

In order to study the impact of different network traffic conditions on the performance of the algorithms, we used a discrete-event network simulator. We found that random delays can improve the performance and robustness of AWC. In contrast, on hard satisfiable instances, the performance of the basic ABT deteriorates dramatically when subject to random link delays. However, we developed a decentralized dynamic restart strategy for ABT, which results in an improvement and shows robustness with respect to the variance in link delays. *Most interestingly, our results also show that the active introduction of message delays by agents can improve performance and robustness, while reducing the overall network load.*

These results validate our thesis that when considering networking applications of DisCSP, one cannot afford to neglect the characteristics of the underlying network conditions. The network-level behavior can have an important, algorithm-specific, impact on solution time. Our study makes it clear that DisCSP algorithms are best tested and validated on benchmarks based on real-world problems, using network simulators. We hope our benchmark domain will be of use for the further analysis and development of DisCSP methods.



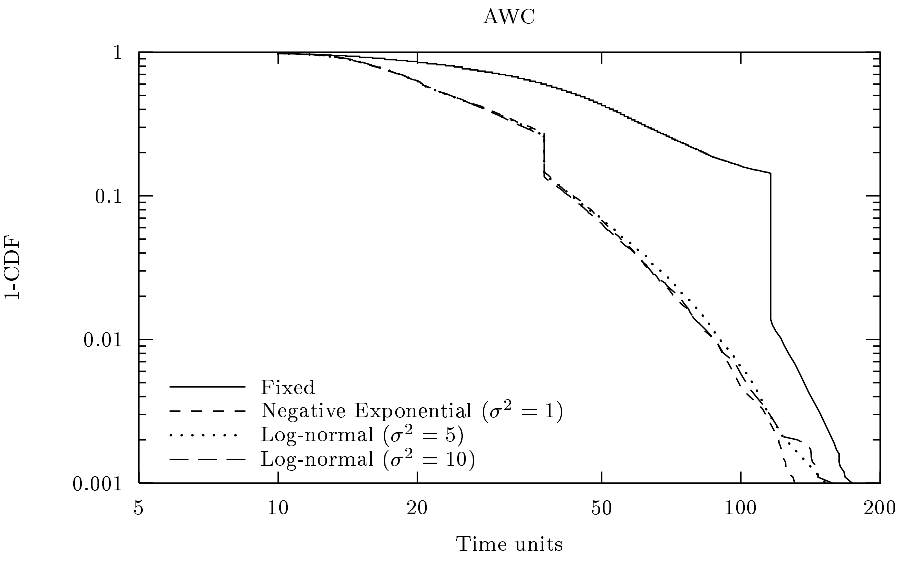


Fig. 7. Cumulative density functions (CDF) of time to solve hard instances for their respective algorithms, AWC, ABT and ABT with restarts under different link delay models

References

[1] R. Béjar, B. Krishnamachari, C. Gomes, and B. Selman. Distributed constraint satisfaction in a wireless sensor tracking system. In *Workshop on Distributed Constraint Reasoning, International Joint Conference on Artificial Intelligence*, Seattle, Washington, August 2001. http://liawww.epfl.ch/ silaghi/proc_wsijcai01.html. 665, 667

[2] S. E. Conry, K. Kuwabara, V. R. Lesser, and R. A. Meyer. Multistage negotiation for distributed constraint satisfaction. *IEEE Transactions on Systems, Man, and Cybernetics (Special Section on DAI)*, 21(6):1462–1477, 1991. 664

[3] M. Crovella and A. Bestavros. Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes. *IEEE Transactions on Networking*, 5(6):835–846, December 1997. 674

[4] C. P. Gomes, B. Selman, and H. A. Kautz. Boosting combinatorial search through randomization. In *AAAI/IAAI*, pages 431–437, 1998. 666, 669

[5] M. Junius, M. Büter, D. Pesch, et al. CNCL. Communication Networks Class Library. Aachen University of Technology. 1996. 670

[6] D. Kirkpatrick and P. Hell. On the complexity of general graph factor problems. *SIAM Journal of Computing*, 12(3):601–608, 1983. 667

[7] B. Krishnamachari. *Phase Transitions, Structure, and Complexity in Wireless Networks*. PhD thesis, Electrical Engineering, Cornell University, Ithaca, NY, May 2002. 667, 671

[8] B. Krishnamachari, R. Béjar, and S. B. Wicker. Distributed problem solving and the boundaries of self-configuration in multi-hop wireless networks. In *Hawaii International Conference on System Sciences (HICSS-35)*, January 2002. 665

[9] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the Self-Similar Nature of Ethernet Traffic (Extended Version). *IEEE Transactions on Networking*, 2(1):1–15, February 1994. 674

[10] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky. Determining computational complexity from characteristic 'phase transitions'. *Nature*, 400:133–137, July 1999. 671

[11] V. Paxson and S. Floyd. Wide area traffic: the failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, 1995. 674

[12] G. Samorodnitsky and M. S. Taqqu. *Stable Non-Gaussian Random Processes*. Chapman & Hall, 1994. 675

[13] Sanders and Air Force Research Lab. ANTs challenge problem. *http://www.sanders.com/ants/overview-05-09.pdf*, 2000. 666

[14] K. Sycara, S. Roth, N.Sadeh, and M. Fox. Distributed constrained heuristic search. *IEEE Transactions on Systems, Man and Cybernetics*, 21(6):1446–1461, 1991. 664

[15] T. Walsh. The interface between P and NP: COL, XOR, NAE, 1-in-k, and Horn SAT. *APES Report*, APES-37-2002, 2002. 671

[16] M. Yokoo. Weak-commitment search for solving constraint satisfaction problems. In *Proceedings of the 12th Conference on Artificial Intelligence (AAAI-94)*, pages 313–318, 1994. 664

[17] M. Yokoo. Asynchronous weak-commiment search for solving distributed constraint satisfaction problems. In *Proccedings of the First International Conference on Principles and Practice of Constraint Programming (CP-95)*, pages 88–102, 1995. 664, 665

[18] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. Distributed constraint satisfaction for formalizing distributed problem solving. In *Proccedings of the Twelfth IEEE International Conference on Distributed Computing Systems*, pages 614–621, 1992. 664, 665

[19] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE Transactions on Knowledge Data Engineering*, 10(5):673–685, 1998. 671, 672

[20] M. Yokoo and K. Hirayama. Algorithms for distributed constraint satisfaction: A review. *Autonomous Agents and Multi-Agent Systems*, 3(2):198–212, 2000. 664, 668, 671