Chapter 12

COMMUNICATION AND COMPUTATION IN DISTRIBUTED CSP ALGORITHMS

César Fernández[1], Ramón Béjar[1], Bhaskar Krishnamachari[2], Carla Gomes[3], Bart Selman[3]

[1]*Departament d'Informàtica i Enginyeria Industrial*
Universitat de Lleida
Jaume II, 69, E-25001
Lleida, Spain
{cesar,ramon}@eup.udl.es

[2]*Department of Electrical Engineering-Systems*
University of Southern California
Los Angeles, CA 90089
bkrishna@almaak.usc.edu

[3] *Department of Computer Science*
Cornell University
Ithaca, NY 14853, USA
{gomes,selman}@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. On unsatis-

fi able instances, however, 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 [Conry et al., 1991; Sycara et al., 1991; Yokoo, 1994; Yokoo, 1995; Yokoo et al., 1992; Yokoo and Hirayama, 2000]. There is a rich 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 particular distributed application that arises in networked distributed systems [Béjar et al., 2001; Krishnamachari et al., 2002]. SensorDCSP is a truly distributed benchmark, a feature that does not fi gure in many of the benchmark problems (such as N-Queens and Graph Coloring)that have been used to study the performance of DisCSP algorithms . SensorDCSP involves a network of distributed sensors that track multiple mobile nodes simultaneously. This problem is an abstraction of the dynamic distributed tracking problem introduced in Chapter 2. We think that this abstraction captures the essential behavior of the real problem, and so it is a good starting model for understanding the full challenge problem.

The problem underlying SensorDCSP is NP-complete . We show that the SensorDCSP domain undergoes a phase transition in satisfi ability, with respect to two control parameters: the level of sensor compatibility and the level of sensor visibility. Standard DisCSP algorithms on our SensorDCSP domain exhibit the easy–hard–easy profi le in complexity, peaking at the phase transition, which is similar to the pattern observed in centralized CSP algorithms. More interestingly, the relative strength of standard DisCSP algorithms on SensorDCSP is highly dependent on the satisfi ability of the instances. This aspect has been overlooked in the literature on account of the fact that, so far, the performance of DisCSP algorithms has been evaluated mainly on satisfi able instances. We study the performance of two well-known DisCSP algorithms— (ABT) [Yokoo et al., 1992] and asynchronous weak-commitment search (AWC) [Yokoo, 1995]—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 fi nd-

ings is that although AWC does indeed perform better than ABT on satisfiable instances, just the opposite is true of unsatisfiable instances.

Our SensorDCSP benchmark also allows us to study other interesting properties that are specific to DisCSPs and 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 a variety of 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. Moreover, though link delay causes the performance of the standard ABT algorithm to deteriorate, a decentralized restart strategy that we have 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 that were developed for the purpose of improving the performance of CSP algorithms [Gomes et al., 1998]. 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 counterintuitively, can also decrease the overall solution time.

The organization of the rest of the chapter is as follows: In Section 12.2 we formalize our model of DisCSP. In Section 12.3 we describe SensorDCSP and model it as a DisCSP. In Section 12.4 we describe two standard DisCSP algorithms and the modifications we have incorporated into the algorithms. In Section 12.5 we present our experimental results on the active introduction of randomization by the agents, and in Section 12.6 we present results on delays caused by different traffic conditions in the communication network. Finally, we present our conclusions in Section 12.7.

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 $\{A_1, A_2, \ldots, A_n\}$ of agents; (2) A set $\{P_1, P_2, \ldots, P_n\}$ of local (private) CSPs, where CSP P_i pertains to agent A_i (and A_i is the only agent that can modify the values assigned to the variables of P_i); (3) A global CSP, each of whose variables is also a variable of one of the local CSPs.

In general, each agent in a DisCSP algorithm controls only one variable. We extend the single-variable approach by having every agent consist of multiple virtual agents, one for each 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 [Sanders and Air Force Research Lab, 2000] 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 12.1 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 12.1. Also, it is required that each mobile be assigned three sensors that satisfy a compatibility relation among them; this compatibility relation is depicted by the graph in the middle panel of Figure 12.1. Two compatible sensors have an edge between them in the compatibility graph. A set of three sensors is compatible if they form a triangle in the compatibility graph. Finally, it is required that each sensor tracks at most one mobile. A possible solution is shown in the right panel, where the set of three sensors assigned to each mobile is indicated by the lighter edges.

This problem is NP-complete , since the problem of partitioning a graph into cliques of size three can be reduced to it [Bejar et al., 2001; Kirkpatrick and Hell, 1983]. This is not true, however, on the limiting case in which every

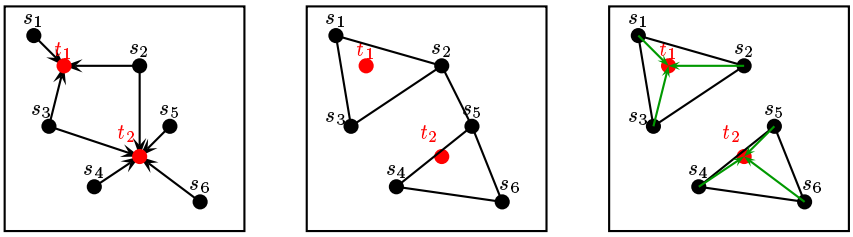

Figure 12.1. A SensorDCSP problem instance

pair of sensors is compatible. That case is polynomially solvable , since it can be reduced to a feasible flow problem in a bipartite graph [Krishnamachari, 2002].

We defi ne 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 parameter (P_v) that controls the edge density of the visibility graph and a parameter (P_c) that controls the edge density of the compatibility graph . 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, unsatisfi able/satisfi able instances by tuning the parameters P_v and P_c appropriately.

We encode 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 distinct and 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 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 specifi c DisCSP algorithms, the Asynchronous Backtracking Algorithm (ABT) and the Asynchronous

Weak-Commitment Search Algorithm (AWC). We provide a brief overview of these algorithms but refer the reader to [Yokoo and Hirayama, 2000] for a more comprehensive description. We also describe the modifications that we have introduced into these algorithms. As mentioned earlier, we assume that each agent can handle only one variable. In what follows, the *neighbors* of a given agent are the agents with whom it shares constraints.

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 of 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 its current value is inconsistent with the assignments of higher-priority neighbors, and so it maintains an agent view consisting 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 lower-priority neighbors of this new assignment. If an agent is unable to find an assignment that is consistent with the assignments of all of its higher-priority neighbors, it sends a *nogood* message, which consists of a subset of that agent's view that makes it impossible for the agent to find a consistent assignment for itself; the *nogood* message is sent to the lowest-priority agent among all the (higher-priority) agents in that particular subset of that agent's view. Receipt of a *nogood* message causes the receiver agent to record the content of that message as a new constraint and then try to find an assignment that is consistent with its higher-priority neighbors and with all of its recorded constraints. If the top-priority agent is forced to backtrack (which implies that its assignment is inconsistent with at least one of its recorded constraints, since there is no higher-priority neighbor with which its assignment could possibly clash), this means that the problem has no solution. If, on the other hand, 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. Agents always select a value for their variable that is consistent with higher priority agents and that minimizes the number of conflicts with lower priority neighbors (min-conflict heuristic). A priority value is determined for each variable, and the priority value is communicated to other agents using the *ok* message. When an agent cannot find a consistent value with its agent view, it generates a new *nogood* and it sends the *nogood* message to all its neighbors. Then, it raises its priority by one unit above the maximal priority of its neighbors, selects a value using the min-conflict heuristic and informs its

neighbors by sending them *ok* messages. If no new *nogood* can be generated the agent simply 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 that are not only consistent with the agent view and the nogood list but also 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 in the context of DisCSP algorithms is to introduce forced delays in the delivery of messages. Delays introduce randomization in that the order in which messages reach their 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:

$d := D \cdot (1 + r)$;

else (with probability $(1 - p)$)

$d := D$;

2 deliver the message with delay d

By transmitting 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 (D) 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 specifi c delays to be applied selectively in the delivery message queue of each agent. It should be noticed that the order of the messages sent by a given agent is preserved even when delays are introduced. However, the order in which messages from different agents arrive to a particular target agent can be altered by the addition of delays.

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 [Gomes et al., 1998], because here the search is not

restarted from scratch, but rather benefits from prior mistakes, since all agents retain their *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) [Junius et al.,]. 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 a variety of 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-delay links , with a negative-exponential distribution and a mean delay of 1 time unit. The communication links used by the virtual agents of the same 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 the queuing-time process on intermediate systems. The factor of 1000 difference between the two delays reflects the fact that intra-agent computation is usually less expensive than 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. Every set contained 19 instances and was generated with a different pair of values (ranging from 0.1 to 0.9) for the parameters P_c and P_v, giving 81 data points. Each instance has been executed 9 times, each time with a different random seed. The results reported in this section were obtained using a sequential value selection function for the different algorithms.

Figure 12.2 shows the percentage of satisfiable instances as a function of P_c and P_v. When both probabilities are low, most of the instances generated are

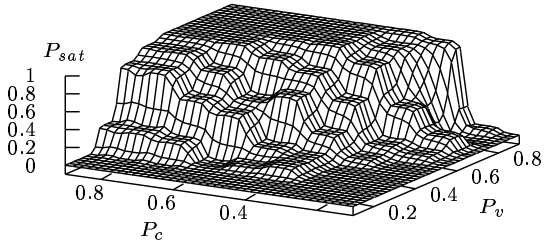

Figure 12.2. Percentage of satisfiable instances with respect to the density parameter for the visibility graph (P_v) and the density parameter for the compatibility graph (P_c)

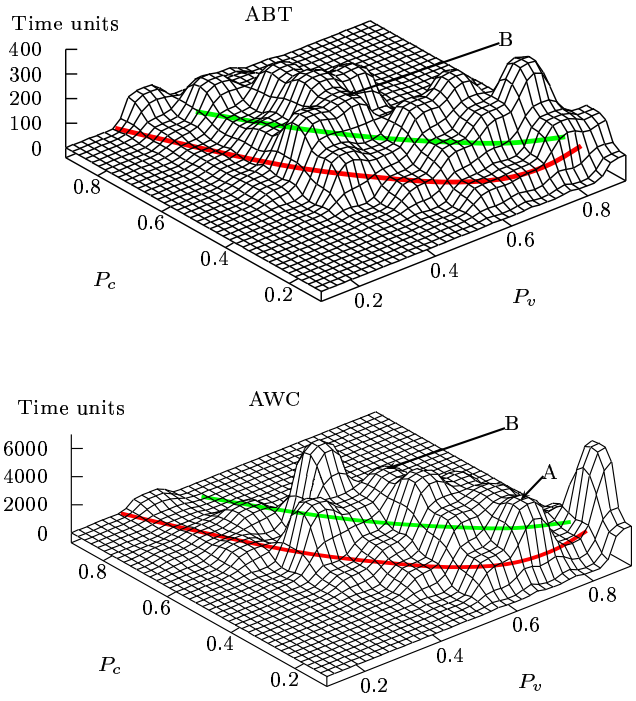

Figure 12.3. Mean solution time with respect to P_v and P_c for ABT and AWC algorithms. Points A and B show the locations of given hard instances analyzed in Subsection 12.5.2

unsatisfiable. For high probabilities, however, most of the instances are satisfiable. The transition between the satisfiable and unsatisfiable regions occurs within a relatively narrow range of these control parameters, analogous to the phase transition in CSP problems, e.g., in SAT [Monasson et al., 1999].

Also consistent with general CSP problems is our observation that the phase transition coincides with the region where the hardest instances occur. Fig-

ure 12.3 shows the mean solution time with respect to the parameters P_c and P_v. As can be seen there, 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 isolines for $P_{sat} = 0.2$ and $P_{sat} = 0.8$, respectively, as per the phase-transition surface of Figure 12.2. As mentioned earlier, the SensorDCSP problem is NP-complete only when not all the sensors are compatible (equivalently, when $P_c < 1$) [Krishnamachari, 2002], 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 [Monasson et al., 1999] and 2+p-COL [Walsh, 2002] . This is particularly noticeable in the mean-time distribution for AWC shown in Figure 12.3.

We observe that the mean times to solve an instance with AWC appear to exceed those with ABT by an order of magnitude. At first glance, this is a surprising result, considering that the AWC algorithm is a refinement of ABT and that results reported for satisfiable instances in the literature [Yokoo et al., 1998; Yokoo and Hirayama, 2000] point to better performance for AWC. One plausible explanation for the discrepancy is the fact that our results deal with both satisfiable and unsatisfiable instances. On further investigation, we found that while AWC does indeed outperform ABT on satisfiable 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 the hierarchy changes during problem solving; consequently, AWC might be expected to take more time to inspect all the search space when unsatisfiable instances are considered.

5.1 Randomization and restart strategies

In this subsection 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 [Yokoo et al., 1998]. In the extensive experiments that we performed with our test instances, we found that a randomized selection function is indeed better than a fixed selection function. Randomization can result in greater variability in performance, however, so ABT should be equipped with a restart strategy. We have not defined a restart strategy for AWC, because, as will be seen in Section 12.6, 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 used restarts in solving hard satisfiable instances with ABT. Figure 12.4 shows the mean time needed to solve a hard satisfiable instance, together with the corre-

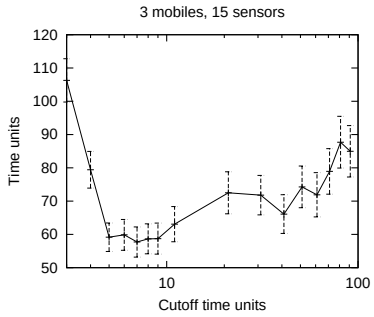

Figure 12.4. Mean time to solve a hard satisfiable instance by ABT using restarts, plotted with respect to cutoff time

sponding 95% confidence intervals, for a number of cutoff times. We observe that there is clearly an optimal restart cutoff time that gives the best performance. As will be discussed in Section 12.6, use of restart strategies is essential when dealing with the delays that occur in real communication networks, given the high variance in the solution time due to randomness of link delays in the communication network.

5.2 Active delaying of messages

One rather novel way of randomizing a DisCSP algorithm is to introduce delays in the delivery of the agents' outgoing messages, as described in Section 12.4. In this subsection we describe our experimental results using AWC and ABT. The amount of delay added by the agents is a fraction r (from 0 to 1) of the delay in the inter-agent communication links. Here, we consider the case where all the inter-agent communication links have fixed delays, of 1 time unit, because we want to isolate the effect of the delay added by the agents. This is in contrast to the experiments described elsewhere in this section, where we report the effects of allowing variable inter-agent delays.

Figure 12.5 shows the results of using AWC to solve a hard satisfiable instance from our SensorDCSP domain (namely, the one that corresponds to point A in Figure 12.3). The solution time and the number of messages are plotted for various values of p, the probability of adding a delay, and r, the fraction of delay added with respect to the delay of the link. The horizontal plane cutting the surface shows the median time (or median number of messages) needed by the algorithm when we consider no added random delays ($p = 0$, $r = 0$). We see that agents can indeed improve the performance of AWC by actively introducing additional, random delays when exchanging messages. The need to send messages during the search process is almost al-

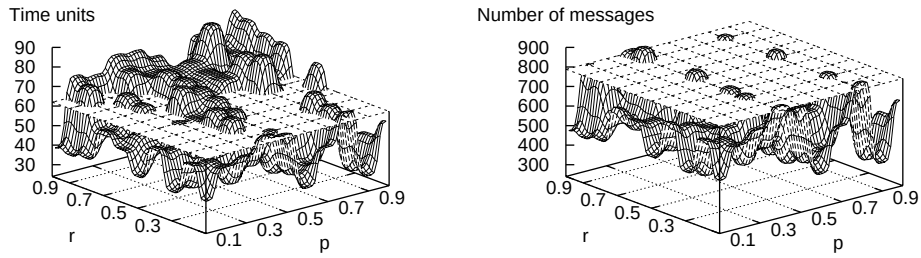

Figure 12.5. Median time and number of messages needed to solve a hard satisfiable instance (point A in Figure 12.3) with AWC when agents add random delays in outgoing messages. The horizontal plane represents the median time for the case where no delay is added ($p = 0$)

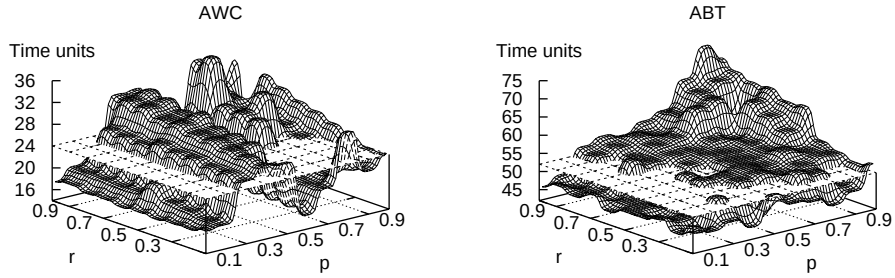

Figure 12.6. Median time for AWC and ABT to solve a hard satisfiable instance (point B in Figure 12.3) when agents add random delays in outgoing messages. The horizontal plane represents the median time for the case where no delay is added ($p = 0$)

ways reduced when agents add random delays; in the best case the number of messages delivered can be as much as a factor of 3 smaller than in the worst case. Perhaps more surprisingly, the solution time can also improve if r or p is not too high. When the values of r and p are too high, we observe a slowdown of the algorithm.

Figure 12.6 shows the results with AWC (left) and ABT (right) for a hard satisfiable instance (namely, the one that corresponds to point B in Figure 12.3). We observe that the performance of AWC is improved in a greater number of cases than that of ABT. Moreover, in the best case the solution time is smaller than that in the worst case by a factor of 2.25 for AWC and 1.63 for ABT. It appears that AWC benefits to a greater extent overall than ABT when

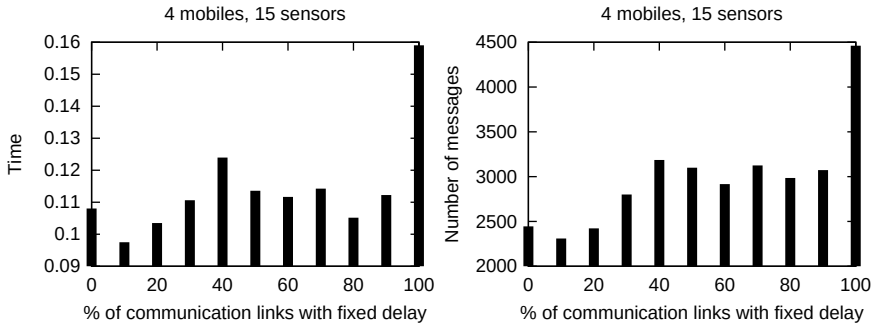

Figure 12.7. Median time and number of messages needed to solve a hard satisfiable instance with AWC depending on the percentage of fixed-delay inter-agent communication links

it comes to the incorporation of delays added by agents. The reason for this could be the ability of AWC to exploit randomization, via its inherent restarting strategy, during the search process.

6. The effect of the communication network data load

As described in the previous section, the performance of AWC applied to a communication network with fixed delays can be improved by the introduction of additional, random delays in message delivery times, with the extent of the improvement depending on the amount of random delay added by the agents. In real networks, however, the conditions of data load present in the communication links used by the agents cannot always be modeled with fixed-delay links. It would thus seem worthwhile to determine how differences in communication network environments can affect the performance of the algorithms. In this section we study the impact of applying DisCSP algorithms to a number of different delay distributions, each one corresponding to a different set of traffic conditions .

In Section 12.5 we considered situations where all the inter-agent communication link delays were either random, exponentially distributed, or fixed . To compare the effects of exponentially distributed delays to those of 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 12.7 shows how the time and the median number of messages needed for AWC to solve a hard satisfiable instance with 4 mobiles and 15 sensors vary with the percentage of inter-agent communication links that have a fixed delay. The remaining 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 affect the performance of the algorithm. An element of randomness in the delay distribution clearly improves the performance of AWC. In addition, observe that there is a fairly good correlation between the number of messages and the time needed, which suggests that an increase or decrease in the solution time is mainly due to a change in the number of messages exchanged.

We now examine various link-delay distributions that can be used to model communication networks. Because of their attractive theoretical properties, negative-exponential distributions of arrival times have traditionally been used to model data traffic. In the past decade, however, it has been shown that although these models are able to capture single-user-session properties, they are not suitable for modeling aggregate data links in local- or wide-area network scenarios [Crovella and Bestavros, 1997; Leland et al., 1994; Paxson and Floyd, 1995]. In view of this, we have simulated network delays according to three different models for the inter-arrival time distribution: the aforementioned negative-exponential distribution, the log-normal distribution, and the Fractional Gaussian Noise (FGN) [Samorodnitsky and Taqqu, 1994] distribution.

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

Figure 12.8 shows the cumulative density functions (CDF) of the time required for three algorithms (AWC, ABT, and ABT with restarts) to solve hard instances when all the inter-agent communication links have delays modeled as fixed, negative exponential, and log-normal. The means were nearly identical, but the variances were quite different.

Table 12.1 presents the estimated mean and variance of the number of messages exchanged when using each of the three aforementioned algorithms, together with several different inter-agent link-delay distributions, to solve the same hard instance. The estimated mean and variance of the solution time for the same scenarios are given in Table 12.2.

The results in Figure 12.8 and Tables 12.1 and 12.2 show that the delay distributions have an algorithm-specific impact on the performance of both AWC and basic ABT. On hard instances, the solution time for the basic ABT algorithm is worse when channel delays are modeled by random distributions than it is in 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 the performance of ABT. Furthermore, ABT with restarts is fairly robust and insensitive to the variance in the link delays.

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

Table 12.1. Estimated mean and variance of the number of messages for different algorithms and 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

Table 12.2. Estimated mean and variance of the solution time for different algorithms and different inter-agent link-delay models

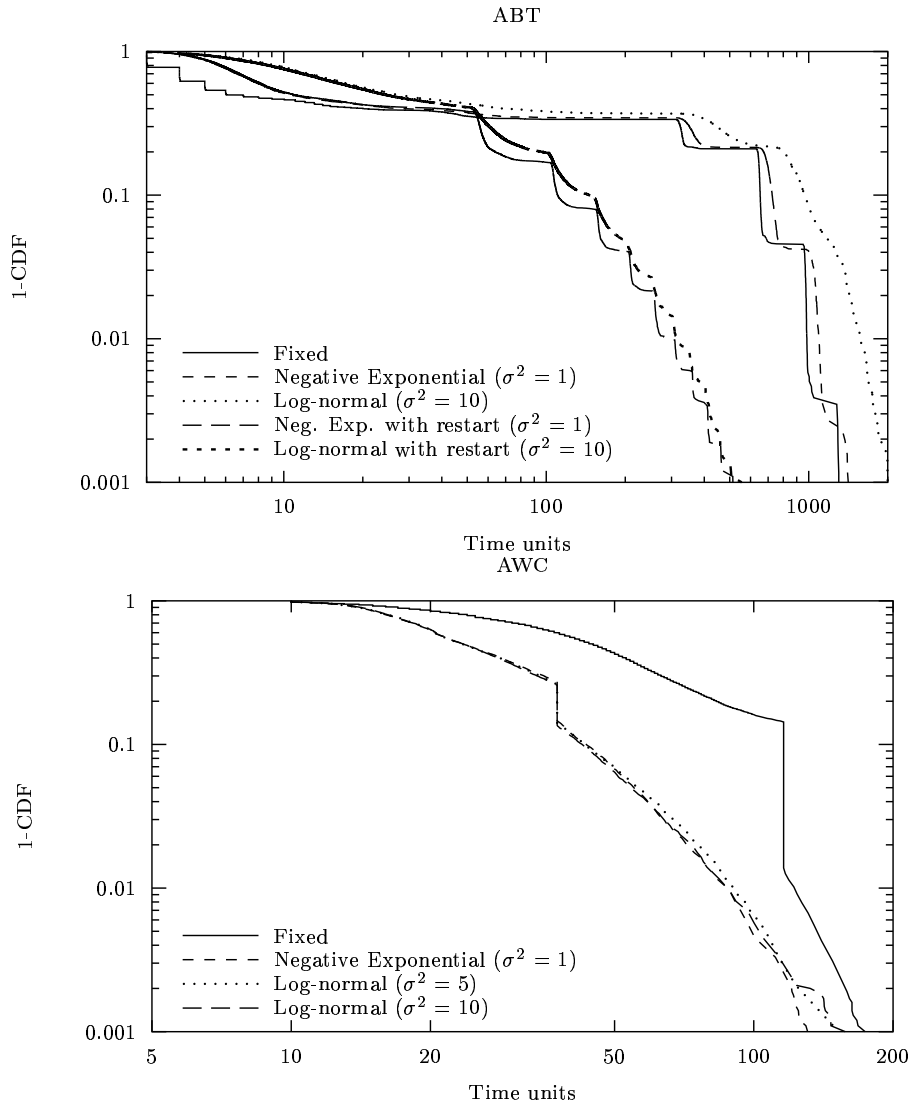

Figure 12.8. Cumulative density functions (CDF) of the time needed to solve hard instances for their respective algorithms (AWC, ABT, and ABT with restarts) using different link-delay models

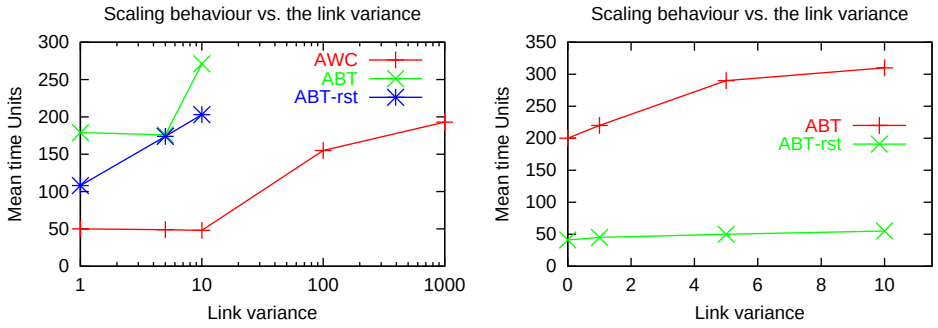

Figure 12.9. Mean time (for AWC, ABT, and ABT with restarts) to solve a hard satisfiable instance, plotted against the link-delay variance (left); same comparison, but on a different instance and only for ABT and ABT with restarts (right)

The behavior of AWC is quite different from that of basic ABT. On hard instances, having randomization in the link delays improves the solution time (compared to the fixed-delay channel). Furthermore, the mean solution time for AWC is extremely robust with respect to the variance in communication link delays, although the variance of the solution time is slightly affected by this.

The left part of Figure 12.9 shows how the mean time needed by each of the three algorithms (AWC, ABT, and ABT with restarts) in solving a hard satisfiable instance scales with an increase in the variance of the exponentially distributed random link delays. Observe that AWC clearly dominates the other two approaches. The right part of Figure 12.9 compares the scaling behavior on a different hard satisfiable instance, but only for ABT and ABT with restarts. The figure clearly shows that although in both cases the scaling appears to be linear, the scaling of the restarting strategy is clearly superior to that of the basic ABT algorithm.

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 performs significantly better than ABT, even ABT with restart. Thus AWC appears to be a better candidate in situations where most instances are likely to be satisfiable, even if we cannot avoid allowing for random delays in the links.

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, asynchronous backtracking (ABT) and asynchronous weak-commitment search (AWC). We showed that the phase-transition region of SensorDCSP induces an easy–hard–easy profile in the solution time for both 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. On hard satisfiable instances, however, 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 a significant, 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.

Acknowledgements

This research was supported by AFOSR, grants 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 grant TIC2001-1577-C03-03; and DARPA grants 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. This paper is an extended version of [Fernandez et al., 2002].

References

Bejar, Ramon, Krishnamachari, Bhaskar, Gomes, Carla, and Selman, Bart (2001). Distributed constraint satisfaction in a wireless sensor tracking system. In *Workshop on Distributed Constraint Reasoning, International Joint Conference on Artificial Intelligence*, Seattle, Washington. http://liawww.epfl.ch/ silaghi/proc_wsijcai01.html.

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

Crovella, Marc and Bestavros, Azer (1997). Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes. *IEEE Transactions on Networking*, 5(6):835–846.

Fernandez, Cesar, Bejar, Ramon, Krishnamachari, Bhaskar, and Gomes, Carla P. (2002). Communication and computation in distributed CSP algorithms. In *8th International Conference on Principles and Practice of Constraint Programming, CP-2002, Ithaca, NY*, pages 664–679.

Gomes, Carla P., Selman, Bart, and Kautz, Henry A. (1998). Boosting combinatorial search through randomization. In *AAAI/IAAI*, pages 431–437.

Junius, M., Buter, M., Pesch, D., et al. CNCL. Communication Networks Class Library. Aachen University of Technology. 1996.

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

Krishnamachari, Bhaskar (2002). *Phase Transitions, Structure, and Complexity in Wireless Networks*. PhD thesis, Electrical Engineering, Cornell University, Ithaca, NY.

Krishnamachari, Bhaskar, Bejar, Ramon, and Wicker, Stephen B. (2002). Distributed problem solving and the boundaries of self-configuration in multihop wireless networks. In *Hawaii International Conference on System Sciences (HICSS-35)*.

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

Monasson, Rami, Zecchina, Riccardo, Kirkpatrick, Scott, Selman, Bart, and Troyansky, Lidror (1999). Determining computational complexity from characteristic 'phase transitions'. *Nature*, 400:133–137.

Paxson, Vern and Floyd, Sally (1995). Wide area traffi c: the failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244.

Samorodnitsky, Gennady and Taqqu, Murad S. (1994). *Stable Non-Gaussian Random Processes*. Chapman & Hall.

Sanders and Air Force Research Lab (2000). ANTs challenge problem. *http://www.sanders.com/ants/overview-05-09.pdf*.

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

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

Yokoo, Makoto (1994). Weak-commitment search for solving constraint satisfaction problems. In *Proceedings of the 12th Conference on Artifi cial Intelligence (AAAI-94)*, pages 313–318.

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

Yokoo, Makoto, Durfee, E. H., Ishida, Toru, and Kuwabara, Kazuhiro (1992). Distributed constraint satisfaction for formalizing distributed problem solving. In *Proceedings of the Twelfth IEEE International Conference on Distributed Computing Systems*, pages 614–621.

Yokoo, Makoto, Durfee, Edmund H., Ishida, Toru, and Kuwabara, Kazuhiro (1998). The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE Transactions on Knowledge Data Engineering*, 10(5):673–685.

Yokoo, Makoto and Hirayama, Katsutoshi (2000). Algorithms for distributed constraint satisfaction: A review. *Autonomous Agents and Multi-Agent Systems*, 3(2):198–212.

DISTRIBUTED SENSOR NETWORKS
A Multiagent Perspective

Edited by
VICTOR LESSER
University of Massachusetts

CHARLES L. ORTIZ, JR.
SRI International

MILIND TAMBE
University of Southern California

Kluwer Academic Publishers
Boston/Dordrecht/London