CONTROLLING COMPUTATIONAL COST: STRUCTURE, PHASE TRANSITIONS AND RANDOMIZATION

Cornell University

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. K274

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2004-219 has been reviewed and is approved for publication

APPROVED: /s/

DANIEL E. DASKIEWICH
Project Engineer

FOR THE DIRECTOR: /s/

JAMES A. COLLINS, Acting Chief
Information Technology Division
Information Directorate

REPORT DOCUMENTATION PAGE

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE JULY 2004	3. REPORT TYPE AND DATES COVERED Final Apr 00 – Dec 03

4. TITLE AND SUBTITLE
CONTROLLING COMPUTATIONAL COST: STRUCTURE, PHASE TRANSITIONS AND RANDOMIZATION

5. FUNDING NUMBERS
C - F30602-00-2-0530
PE - 62301E
PR - ANTS
TA - 00
WU - 06

6. AUTHOR(S)
Bart Selman

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Cornell University
Upson Hall
Ithaca New York 14853

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Advanced Research Projects Agency   AFRL/IFTB
3701 North Fairfax Drive   525 Brooks Road
Arlington Virginia 22203-1714   Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-2004-219

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Daniel E. Daskiewich/IFTB/(315) 330-7731/ Daniel.Daskiewich@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 Words)*
This report describes Cornell's contribution to the ability to build information systems that use highly decentralized and autonomous negotiation of tasks for distributed resource allocation. This effort extends phase transition analysis to structured domains and generalized constraint satisfaction tasks. The effort was devoted to connecting frameworks for multi-agent negotiation based systems with the research on analytical and empirical computational complexity. The general goal was to improve the expressiveness and scalability of complex distributed systems by exploiting computational hardness awareness in both the design and operation of the systems.

14. SUBJECT TERMS
Computational Complexity, Autonomous Negotiation, constraint Satisfaction Problems

15. NUMBER OF PAGES
46

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

List of Figures

List of Tables

1. Summary

The mission of our group has been connecting the work on negotiation frameworks with the research on analytical and empirical computational complexity. The general goal has been to use our findings in both the design and operation of complex distributed systems, improving their expressiveness and scalability based on principled controlled hardness awareness. In particular, our goal was to study the *impact of problem structure on the computational complexity of the problem*, and to exploit this connection in both problem modeling and solving.

At various stages of the project, our work has been dedicated to both:

1. ANTS (short for Autonomous Negotiation Teams) Sensor Networks domain and negotiation protocols, and

2. ANTS Autonomic Logistics domain.

In both domains, our work has focused on *identifying the domain's structural features that most affect the complexity of the negotiation*. Starting with the specification of the challenge problem, our work in Sensor Networks domain has been focused on:

1. Generic research on accelerating Constraint Satisfaction Problem (CSP) problem solvers by analysis and exploiting problem's structure.

2. Formal specification of agents' negotiation abilities, and the negotiation tasks. Abstracting the sensor network negotiation tasks as Distributed Constraint Satisfaction Problem (DCSP) and Constraint Optimization (COP).

3. Developing fully-functional discrete-event simulator for simulating negotiation between sensor agents upon a realistically modeled network, and studying three-side connections between (1) structural properties of the negotiation tasks; (2) network properties, and (3) relative advantage of various negotiation strategies.

4. Comparative evaluation of several backtrack-style and local search negotiation strategies on various negotiation tasks.

5. Formal complexity analysis of the negotiation problem in sensor networks, specifying the precise complexity hierarchy of the problem sub-classes as a function of different structural parameters of the problem.

6. Analysis of practical scalability of the negotiation protocols by studying the form of their phase transition behavior, again, as a function of different structural parameters of the problem.

In the second part of the program, the effort of our group has been directed towards ANTs Autonomic Logistics domain. In Sensor Networks domain our focus was mainly on identifying scalable negotiating formations and hardness aware problem solving. In Autonomic Logistics domain we focused on exploiting problem structure right at the level of problem modeling. The conceptual idea was to exploit sophisticated knowledge representation techniques for compact and informative problem encoding. The results achieved by our group show that contribution of such sophisticated encodings can be two-fold. First, principle applicability of the given problem solvers can be extended to a wider range of negotiation tasks. Second, the performance of these problem solvers can be dramatically improved.

Joining other contractors involved in Autonomic Logistics domain, and working closely with the ISI contractors, our work in Autonomic Logistics domain was focused on:

1. Generic research on critical resources that form the core of combinatorics in hard real-world problem instances. Development and computational analysis of algorithms for discovering critical resources in the given problem instances.

2. Study of the CAMERA (Coordination and Management Environments for Responsive Agents) tool modules, a real-time resource management architecture providing a basis for integrating technologies developed by the ISI (Information Sciences Institute) contractors.

3. Analysis of the SNAP (Schedules Negotiated by Agent Planners) problem domain. SNAP is a specialized system for scheduling flight operations developed by the ISI contractors on top of the CAMERA tool, and used as the practical benchmark for Autonomic Logistics domain.

4. Analysis of ATTEND (Analytic Tools to Evaluate Negotiation Difficulty) pseudo-boolean encoding (ISI) for CAMERA resource management problems, focusing on "lessons learnt" from the SNAP domain.

5. Formal development of a pseudo-boolean extension Cornell-1 to the ATTEND encoding. Cornell-1 allowed CAMERA/SNAP to enable resource/task allocation by taking into account the *dynamic* properties of the resources. More formally, Cornell-1 enabled CAMERA to deal not only with "negotiation for scheduling", but also "negotiation for planning" problem, raising significantly the expressivity of the system.

6. Further extension of the pseudo-boolean encoding. The pseudo-boolean extension Cornell-2 enriched its predecessor by addressing resource abstraction due to changing resolution of the negotiation time scale.

7. Implementation, testing, and performance evaluation of Cornell-2, followed by its deployment to the ISI contractors for successful integration with the CAMERA tool.

2. ANTs Sensor Networks Domain and Negotiation Protocols

Our principle approach to analyzing negotiation protocols for Sensor Networks domain has been based on intuitive reduction of this negotiation problem to the general problem of Distributed Constraint Satisfaction (or DisCSP, for short) *[YH00]*. The objective has been to adapt general purpose distributed negotiation protocols, and to analyze the complexity of the corresponding class of problems (which we call SensorDCSP) both analytically and empirically. The main goal has been to study practical scalability of various distributed negotiation strategies as a function of the qualitative structure of the problem.

In a DisCSP problem, 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 $\{A_1, ...,A_n\}$ of agents; (2) A set $\{P_1, ...,P_n\}$ 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 the Sensor Network domain we have multiple sensors $S = \{s_1, ...,s_m\}$ and multiple mobile targets $T = \{t_1, ...,t_n\}$ (or mobiles, for short) which are to be tracked by the sensors. The goal is to allocate three sensors to track each mobile node, such that all these triplets of sensors are pair-wise disjoint and consistent with 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 Figure 1(a). In addition, 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 Figure 1(b). Finally, it is required that each sensor only track at most one mobile. A possible solution is depicted in Figure 1(c), where the set of three sensors assigned to each mobile is indicated by the lighter edges.

(a)   (b)   (c)
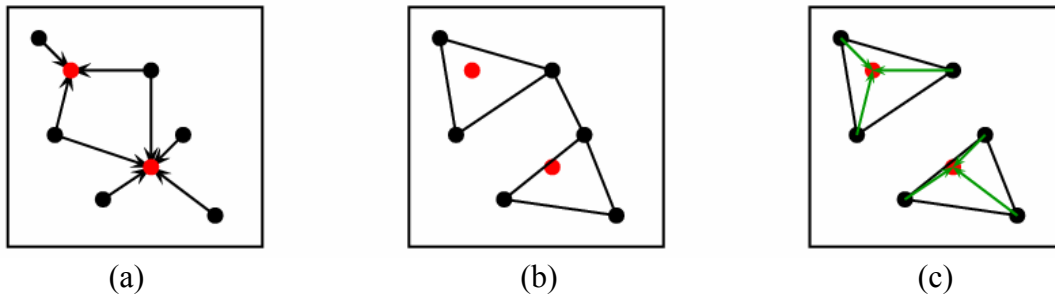
Figure 1: A SensorDCSP problem instance: (a) Visibility graph; (b) Compatibility graph; (c) Feasible sensors/mobiles assignment.

3

Our DisCSP encoding of the Sensor Network problem is called SensorDCSP and is 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 constraint between the variables of one agent is 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.

In our work we consider two probably most popular DisCSP algorithms, *Asynchronous Backtracking (ABT) [YDIK92]*, and *Asynchronous Weak-Commitment (AWC) [Yokoo95]*. We provide a brief overview of these algorithms. In what follows, the *neighbors* of a given agent are the agents with whom it shares constraints.

ABT is a distributed asynchronous version of a classical backtracking negotiation 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, namely the *nogood* messages and *ok* messages. Agents initiate the negotiation 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 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.

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. If an agent's current assignment is inconsistent with that agent's 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 raises its priority by one unit above the maximal priority of its neighbors. Then it finds an assignment that is consistent with the assignments of its higher-priority neighbors and informs its neighbors by sending them *ok* messages. If no new *nogood* can be generated, the agent waits for the next message.

2.1 Phase Transition and Complexity Profiles for SensorDCSP

Our analytical analysis *[BKGS01]* showed that:

1. SensorDCSP is NP-complete, since a known NP-complete problem of partitioning a graph into cliques of size three can be reduced to it.

2. In limiting case in which every pair of sensors is compatible SensorDCSP is solvable in polynomial time as each such problem can be reduced to a feasible flow problem in a bipartite graph.

For our experiments, we have defined a random distribution of instances of SensorDCSP, and developed a parameterized instance generator for this random distribution that generates DisCSP-encoded instances of the sensor network negotiation problem. An instance of this problem is generated from two different random graphs, the visibility graph and the compatibility graph. Apart from the number of mobiles and number of sensors, we also specify parameters controlling edge density of the visibility graph P_v and edge density of compatibility graph P_c. Each of these parameters specifies 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 completely control the number of constraints in the generated instances.

For the problem of negotiation in Sensor Networks, as well as in any other networked environment, 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). 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.

In our experiments 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 for the parameters P_c and P_v (ranging from 0.1 to 0.9), providing us with 81 data

5

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.

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. Here we use fixed-delay links because we assume that a set of virtual agents work inside a private computation node and this allows virtual agents to communicate with each other using dedicated communication links. This scenario could correspond to a heavy-loaded network situation where inter-agent delay fluctuations are due to the queuing process on intermediate systems. The difference of a factor of 1000 between the two delays reflects that intra-agent computation is usually less expensive than inter-agent communication. Later we will show how different delay-distribution models over the inter-agent communication links can impact the performance of the algorithms.
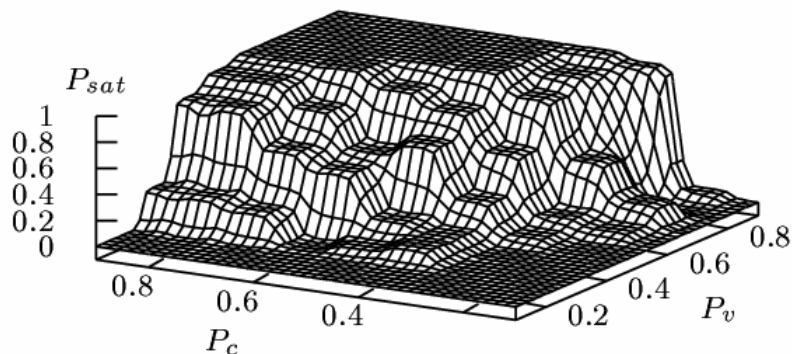
Figure 2: Percentage of satisfiable instances depending on the density parameter for the visibility graph (P_v) and the density parameter for the compatibility graph (P_c).

Figure 2 shows the ratio of satisfiable instances as a function of P_c and P_v. When both probabilities are low, most of the generated instances are 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.

Also consistent with other CSP problems is our observation that the hardest instances for these backtracking algorithms generally occur in the region where the phase transition occurs. Figure 3 shows the mean solution time with respect to the parameters P_c and P_v: The hardest instances lie on the diagonal that defines the phase-transition zone, with a

6

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 earlier, the SensorDCSP problem is NP-complete only when not all sensors are pair-wise compatible (*i.e.,* when $P_c < 1$). Therefore, the parameter P_c could separate regions of different mean computational complexity, as in other mixed P/NP-complete problems *[MZKST99,Walsh02]*. This is particularly noticeable in the mean-time distribution for AWC shown in Figure 3.
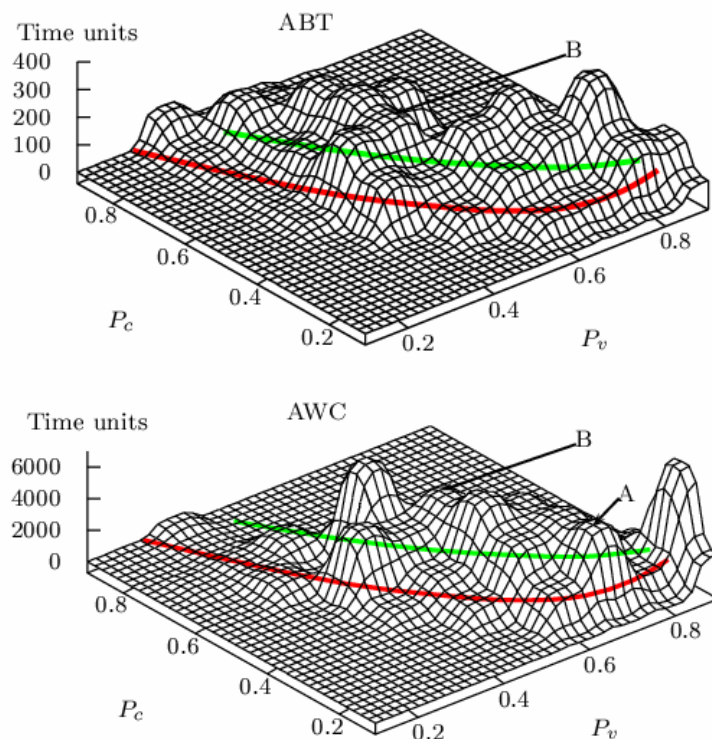
Figure 3: Mean solution time with respect to P_c and P_v for ABT and AWC algorithms.

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 point to better performance for AWC *[YDIK98, YH00]*. 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.
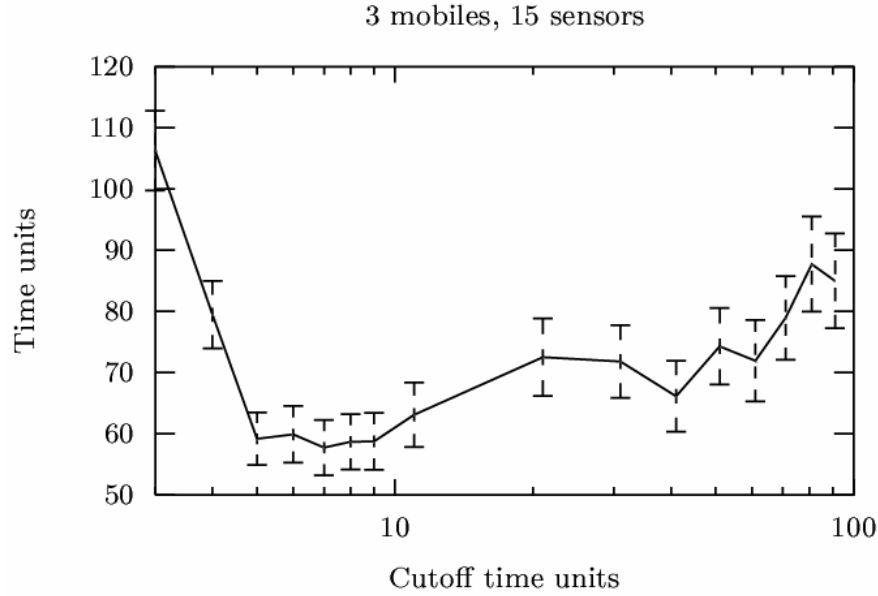
Figure 4: Mean time to solve a hard satisfiable instance by ABT using restarts, plotted with different cutoff times.

2.2 Randomization and Restart Strategies.

As a part of our computational analysis we studied the effect of adding a restart strategy *[GSK98]* to ABT. The introduction of a randomized value selection function was directly assumed in its original formulation *[YDIK98]*. In extensive experiments we performed with our test instances, we found that the randomized selection function is indeed better than a sequential value selection. On the other side, randomization can result in greater variability in performance, and thus ABT should be equipped with a restart strategy. We have not defined a restart strategy for AWC, because 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 4 shows the mean time needed to solve a hard satisfiable instance, together with the corresponding 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 argued later, 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.

8

2.3 Active Delaying of Messages

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 because the order in which messages from different agents reach their destination 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:

with probability p:
$\quad d = D \cdot (1+r)$
else (with probability ($1-p$))
$\quad d = D$

Transmitting message m with delay D means that the agent requires its communication interface to add D seconds to the delivery time currently scheduled for m and all the successors of m in the message queue. The latter preserves the order of transmission and reception for the messages sent from one agent to another agent. The parameter r is the fraction of the communication delay (D) added by the agent.

In this section we present the results of our experiments with the AWC and ABT algorithms, and active delaying of messages. The amount of delay added by the agents is a fraction r 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.
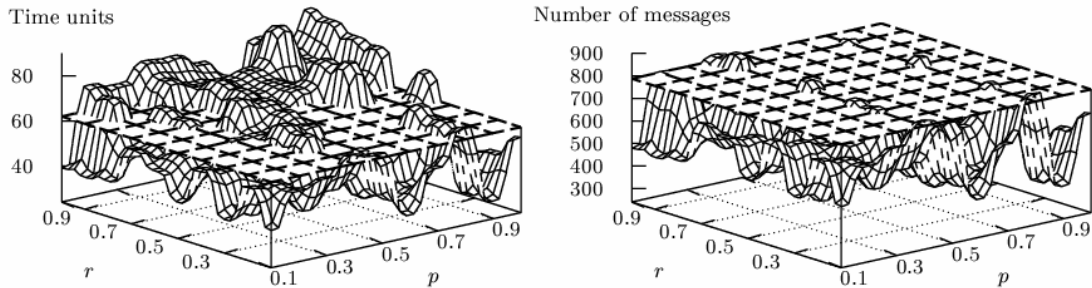
Figure 5: Median time and number of messages needed to solve a hard satisfiable instance (point A in Figure 3) with AWC when agents add random delays in outgoing messages. The horizontal plane represents the median time (or the median number of messages) for the case where no delay is added ($p = 0$).

Figure 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 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 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 always 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 the increase in delay (r) is not too high.

Figure 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 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 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 inherently restarting search strategy.
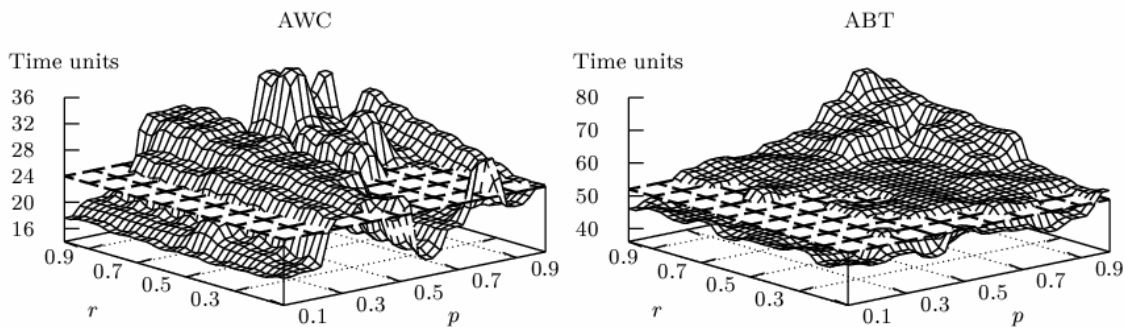
Figure 6: Median time for AWC and ABT to solve a hard satisfiable (point B in Figure 3) instance when agents add random delays in outgoing messages. The horizontal plane represents the median time for the case where no delay is added ($p = 0$).

2.4 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. 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 Section 2.3 we discussed inter-agent communication links with random, exponentially distributed delays. In this section we study the effect produced in the performance of DisCSP algorithms by considering delay distributions corresponding to different traffic conditions. 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 7 shows how the median time and number of messages needed by AWC for solving a hard satisfiable instance with 4 mobiles and 15 sensors vary with 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 affect the performance of the algorithm. An element of randomness in the delay distributions 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.
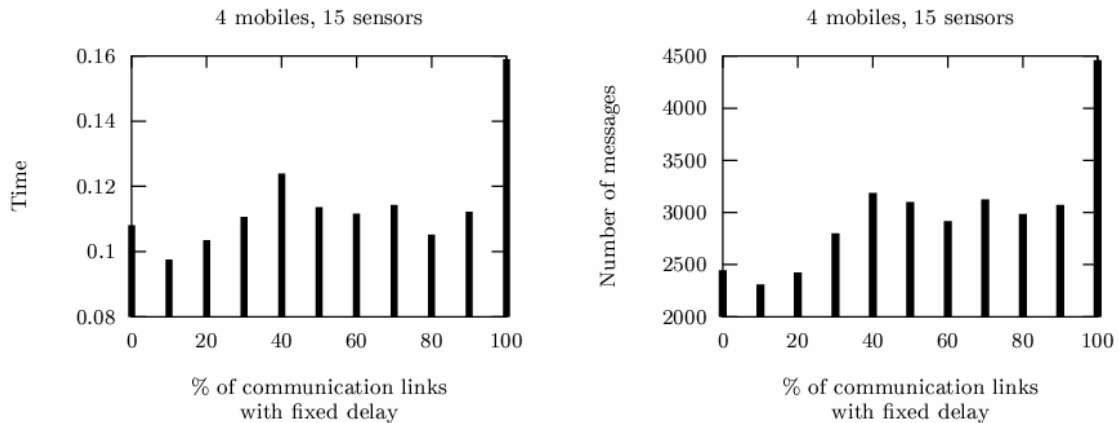
Figure 7: Median time and number of messages needed to solve a hard satisfiable instance with AWC depending on the percentage of fixed-delay inter-agent communication links.

11
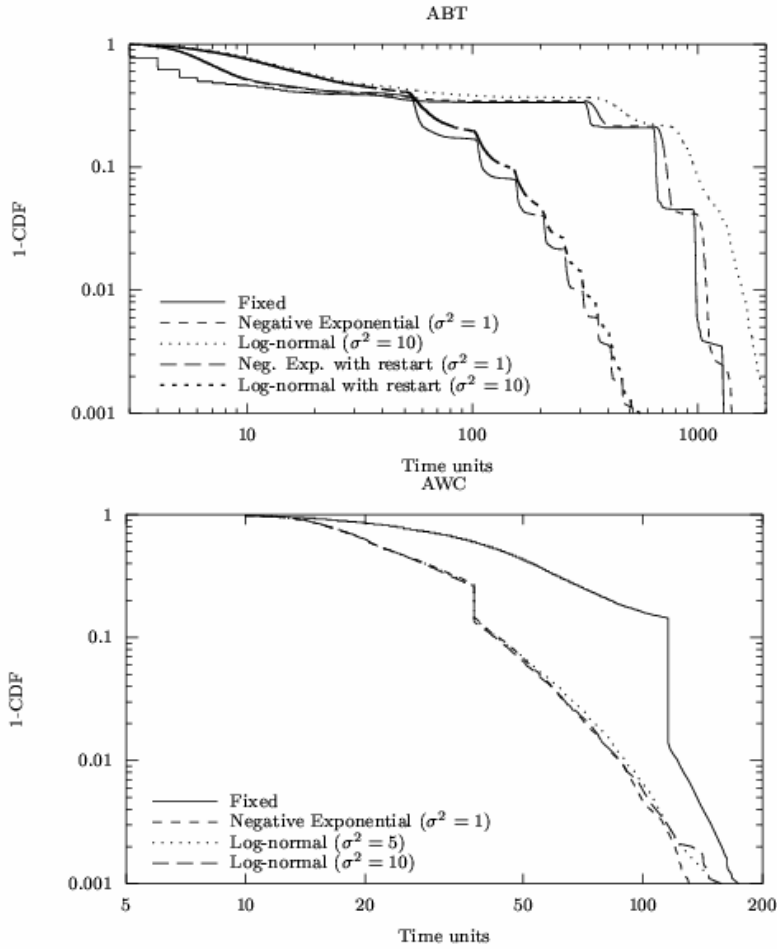
Figure 8: Cumulative Density Functions (CDF) of the time needed to solve hard instances for their respective algorithms, AWC, ABT and ABT with restarts under different link delay models.

We now examine various link-delay distributions that can be used to model communication network traffic. Because of their attractive theoretical properties, negative-exponential distributions of arrival times have traditionally been used to model data traffic. It has been shown, however, 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 *[CB97,LLWW94, Paxson97]*. 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) distribution *[ST94]*.

The log-normal distribution can be used to obtain distributions with any desired variance, whereas FGN processes are able to capture crucial characteristics of the 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 typical parameter values of

H=0.75 and *d=0.4*. Figure 9 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 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 2. The results in Figure 9 and Tables 1 and 2 show that the delay distributions have an algorithm-specific impact on the performance of both AWC and basic ABT.

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 1: Estimated mean and variance, from the empirical distributions, of the number of messages for different algorithms and different inter-agent link delay models when solving a hard satisfiable instance.

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 2: Estimated mean and variance, from the empirical distributions, of the solution time for different algorithms and different inter-agent link delay models when solving a hard satisfiable instance.

For the basic ABT, the solution time on hard instances 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 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. Likewise, 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. In general, we found that on satisfiable instances, AWC always performs significantly better than both basic and restarts-enhanced ABTs. Therefore, AWC appears to be a better candidate in situations where most instances are likely to be satisfiable, and where we cannot avoid random delays in the links.

2.5 Modeling Spatial Structure of the Sensor Networks

Our analysis of Sensor Network problems provides us with the first results on behavior of distributed CSP algorithms in real-world distributed applications. Observe that the very concrete specification of the SensorDCSP problem helps us both to analyze its computational complexity, and to establish coherent experiments for empirical analysis. However, getting closer to the real-world tracking systems, one may have to further specify the properties of the domain. The main information that we believe should be captured in analysis of various tracking systems is the *spatial* properties of both communication between the sensors and visibility of the mobiles. Two reasons make capturing this information essential:

1. Given spatial limitations for both communication between the sensors and visibility of the mobiles, the complexity analysis for general SensorDCSP provides only upper bounds on the complexity of any spatially-limited SensorDCSP. In addition, deriving conclusions on various sub-classes of spatially-limited SensorDCSP from the empirical results on general SensorDCSP is not straightforward whatsoever. In particular, this makes it hard to analyze *scalability* of the negotiation algorithms with respect to real-life tracking systems.

2. The overall goal of any tracking system is to track a set of *moving* objects, and this set is not necessarily constant over time (e.g., some tracked objects run out of the region covered by the sensors, while some new objects are getting into this region). Performance analysis of such a dynamic system is impossible without some realistic assumptions about the *dynamics* of the moving objects, which in turn can be specified only with respect to some concrete spatial model of SensorDCSP.

In addition, spatial nature of the problem instances is likely to lead to inherently decomposable problems, making adopting the general-purpose DisCSP-based negotiation protocols even more attractive. Influenced by the above motivation and the properties of a recently studied challenge problem for distributed tracking systems, we introduce a *grid-based* SensorDCSP problem (or *GridDCSP*, for short), and perform both analytical analysis of this problem and empirical study of DisCSP algorithms on both static and dynamic settings of this problem *[Grid04]*.

The spatial model of the sensor network distributed negotiation in GridDCSP inherits the core properties of the UMass and our test beds: As before, we have multiple sensors, multiple targets which are to be tracked by the sensors subject to visibility and compatibility constraints, and the goal of the negotiation between agents associated controlling the sensors is to allocate three sensors to track each target, while keeping these triplets of sensors pair-wise disjoint. However, now the visibility and compatibility constraints have a close relationship with the physical limitations of the sensors and the properties of the terrain on which the sensors are located.

In GridDCSP, the sensors are located on the nodes of a uniform *grid*. This assumption makes analysis of the problem more coherent, while no generality is lost as some sensors

can be inactive, leading to an arbitrary set of active sensor locations. The targets are located within the surface enclosed by the grid; this way the grid specifies the generally trackable region.
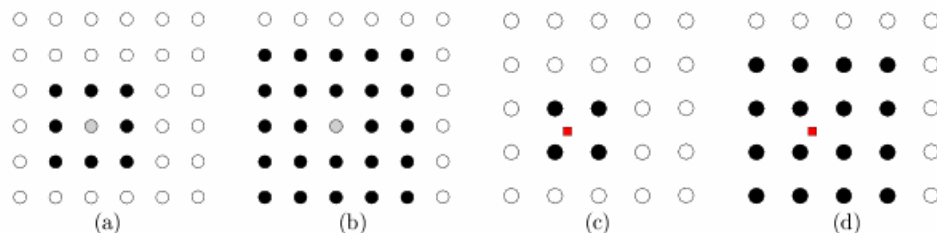
Figure 9: *k*-compatibility and *k*-visibility windows.
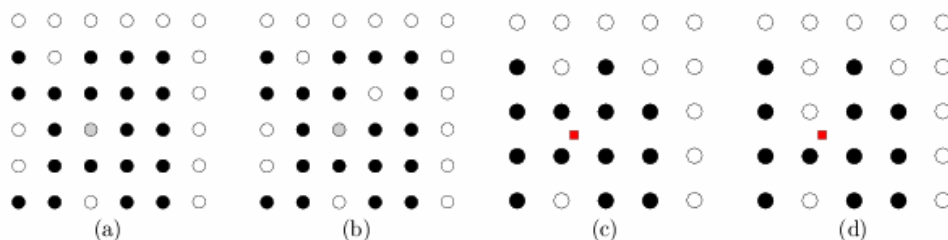
Figure 10: Locality of communication and visibility.

The physical limitations of the sensors are modeled by the notions of k_c-*compatibility* and k_v-*visibility*. The k_c-compatibility window for a sensor *s* corresponds to the set of all sensors that are at most k_c general (rectilinear and/or diagonal) hops from *s*. Similarly, the k_v-visibility window for a target *t* corresponds to the set of all sensors that are at most *k* general hops around *t*. For example, the black sensors in Figures 9(a) and 9(b) correspond to 1-compatibility and 2-compatibility windows for the gray sensor, while the black sensors in Figures 9(c) and 9(d) correspond to 1-visibility and 2-visibility windows for the rectangular sensor, respectively. The compatibility of an GridDCSP-based sensor network is called k_c-restricted if each sensor can communicate only with some sensors within its k_c-compatibility window. The notion of k_v-restricted visibility is defined similarly. For example, thinking of the gray sensor in Figures 10(a) and 10(b) as the only sensor, the compatibility graphs corresponding to Figure 10(a) and 10(b) are 2-restricted. Similarly, thinking of the rectangular mobile in Figures 10(c) and 10(d) as the only mobile, the visibility graphs corresponding to Figure 10(c) and 10(d) are 2-restricted. It is easy to see that higher values of *k* for both compatibility and visibility correspond to more powerful sensors.

While the physical limitations of the sensors in GridDCSP are modeled via the locality windows, the *terrain limitations* are modeled via incomplete compatibility and visibility within the windows. This part of modeling is very similar to our modeling of SensorDCSP: Within a particular class of locality (k_c, k_v), representing problems with k_c-restricted compatibility graphs and k_v-restricted visibility graphs, the problems can be

15

ordered according to the local constrainedness, i.e., the expected number of sensors that a sensor can communicate with and the expected number of sensors that can track a target.

For our experiments, a random distribution of GridDCSP problem instances for a particular pair of locality parameters (k_c, k_v) is defined as follows. An instance of the problem is generated from two different random graphs, the visibility graph and the communication graph. Apart of setting the number of targets and number of sensors, we also specify the parameters P_c, $P_v \in (0,1]$ that control the edge density of visibility and communication graphs, respectively. These parameters specify the independent probability of including a particular edge in the corresponding graph. However, in GridDCSP these parameters have only local effect: For every pair of sensors s and s', the probability $\Pr(s,s')$ for the edge (s,s') to be a part of the communication graph is given by:

$$\Pr(s,s') = \begin{cases} 0 & dist(s,s') > k_c \\ P_c & dist(s,s') \le k_c \end{cases} \qquad (1.1)$$

Similarly, for the visibility graph, we define:

$$\Pr(t,s) = \begin{cases} 0 & dist(t,s) > k_v \\ P_v & dist(t,s) \le k_v \end{cases} \qquad (1.2)$$

Intuitively, higher values for P_c and P_v correspond to less problematic terrain conditions for communication and tracking, respectively.

To conclude, each sensor network problem instance in GridDCSP can be characterized by the following parameters:

- *Order of the problem*, characterized by both the number of sensors and the number of targets.

- *Level of decomposition*, modeled via the locality of compatibility and visibility, using the corresponding notions of window restrictness (k_c, and k_v), and

- *Level of constrainedness*, modeled via the expected fraction of sensors that can communicate with a sensor and the expected fraction of sensors that can track a target, out of the maximally possible such numbers specified by the level of decomposition. These aspects of the problem instances are modeled using the uniform probability distributions P_c and P_v with their corresponding means.

16

2.6 Complexity Analysis for SensorDCSP

After specifying our spatially restricted model for sensor networks, our analysis took several directions:

1. Worst-case analytical complexity analysis of the GridDCSP sensor network problems, with respect to the parameters described above.

2. Average-case empirical complexity analysis of various distributed negotiation protocols with respect to the above parameters and the results obtained for (1).

3. Modification of the previous negotiation algorithms in order to deal with the dynamic characteristics of the problem, and computational analysis of alternative approaches to deal with continuously changing set of moving targets.

First, we have showed analytically that the general GridDCSP-based sensor network problem is NP-complete, thus in worst-case sense it is not easer than its non-spatial original version. In fact, we showed that GridDCSP is NP-complete even for the case of 2-visibility. We identified, however, several tractable special cases of the problem. First, we defined the notion of *locally complete* compatibility graphs, and showed that any problem with such compatibility graphs is solvable in low polynomial time. Informally, the compatibility graph is locally complete if any pair of sensors that are able to track some target can communicate one with another. Second, we have showed that any problem with 1-restricted visibility is solvable in low polynomial time.

In the first experiment we considered the AWC protocol on different sets of instances with 25 sensors (grid 5 x 5) and 5 mobiles, with every set generated with different values for the parameters P_c and P_v with respect to Eqs. (1.1) and (1.2). The parameters P_c and P_v range from 0.1 to 1 with an increment of 0.1, giving a total number of 100 data sets, where every set contains 50 instances. Given our analytical complexity results, we consider three hard subclasses of GridDCSP, corresponding to $k_v = 2$ and $k_c \in \{1,2,3\}$. Figure 11(a) shows the percentage of satisfiable instances as a function of P_c and P_v for $k_c = 1,2$ and $k_v = 2$. As in the case of general SensorDCSP, when both probabilities are low, the instances generated are mostly unsatisfiable, while for high probabilities most of the instances are satisfiable. Both for $k_c = 1$ and $k_c = 2$, the transition between the satisfiable and unsatisfiable regions occurs within a narrow range of the density parameters. Observe that, for $k_c = 1$ this range corresponds to significantly higher values of P_c and P_v, comparatively to these for $k_c = 2$. However, the form of the transition for various values of k_c is very similar (see Figure 11(b)), showing a similar phase transition behavior for various subclasses of the GSensorDCSP problem with $k_v = 2$.
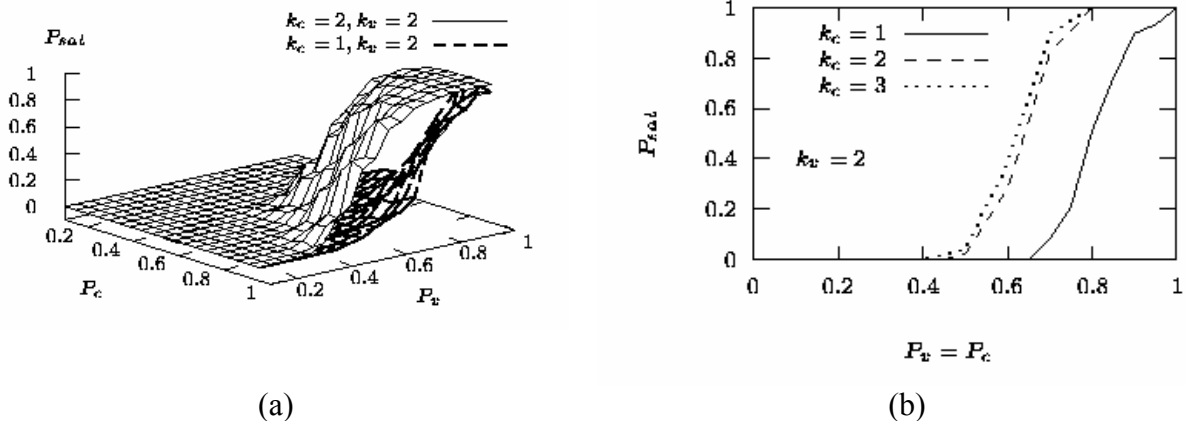
Figure 11: Percentage of satisfiable instances depending on density parameters for the visibility graph (P_v) and the compatibility graph (P_c): (a) Plot for different values of P_c and P_v ; (b) Plot for equal P_c and P_v.
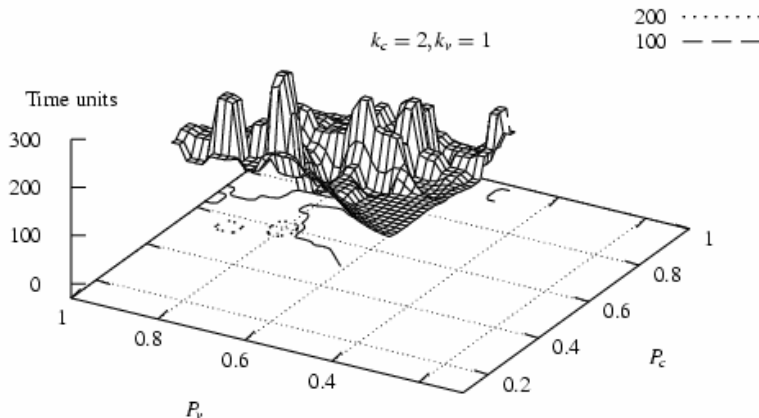
Figure 12: Mean solution time with respect to P_c and P_v for the AWC on instances with 25 sensors, 5 mobiles, $k_c = 1$ and $k_v = 2$.

Consistently with the general SensorDCSP, we observe that the phase transition coincides with the region where the hardest instances occur. For instance, Figure 12 shows the mean solution time with respect to the density parameters P_c and P_v for the problem instances with 25 sensors, 5 mobiles, $k_c = 1$ and $k_v = 2$. Somewhat less expected result is depicted in Figure 13 for the case of $k_v = 1$ (and $k_c = 1$). Recall that we analytically proved this problem class to be polynomial. The actual proof is by a reduction to the problem of feasible integral flow in bipartite graphs. Despite the fact that AWC has no explicit connection with the algorithms for the latter problem, Figure 13(b) shows that these instances are easy for AWC as well.

18
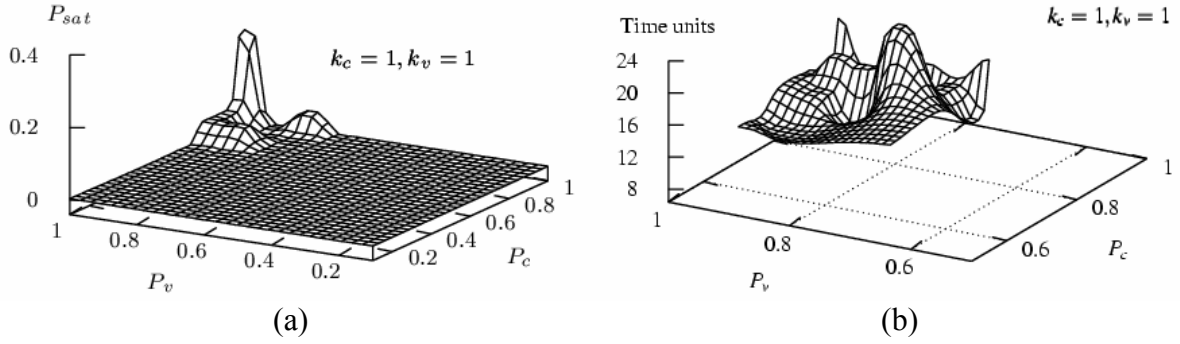
Figure 13: (a) Percentage of satisfiable instances and (b) Mean solution time for the AWC on (polynomial) instances with 25 sensors, 5 mobiles, k_c = 1 and k_v = 2.

For the second experiment with the AWC algorithm, we consider different sets of instances for several orders of the problem (size of the grid), and several levels of decomposition (visibility and compatibility limitations). In particular, we consider grids of 25, 36, 49, 64, 81, and 100 sensors (N = 5, 6, 7, 8, 9, 10), tracking 5, 7, 9, 12, 15 and 18 mobiles, respectively, giving us an approximately constant ratio between the number of mobiles and the number of sensors for each case. Note that N = 10 was the largest problem size we were able to deal with using the CNCL (Communication Networks Class Library) simulator *[JBP96]*. The constrainedness of visibility and compatibility graphs is kept equal (k_c = k_v = k), and different sets correspond to k equal 2, 3, 4, and 5. Each set of problem instances corresponding to a particular pair of values (N, k) contains 30 instances. The important point is that all the problem instances, in all the sets (N, k), have been selected from the corresponding phase transition regions with respect to the density parameters P_c and P_v, representing the regions of the hardest problem instances (as it was shown in Figures 11(a) and 12. (The phase transition regions for every pair (N, k) have been determined in advance.)

The mean solution time for satisfiable instances in this experiment is plotted in Figure 14 as a function of N, where Figures 14(a), (b) and (c) depict this graphs in logarithmic scale for the problem instances with k = 2,3, k = 3,4, and k = 4,5, respectively, while Figure 14(d) presents the whole picture in the linear scale. We observe that the problem scalability with N degrades dramatically as k increases, but it can be considered as reasonable for k = 2 and k = 3. In order to capture the exponential behavior of AWC on these problems, Figures 14(a-c) depicts the obtained measures, showing 95% confidence interval of the samples in logarithmic scale, as well as their corresponding linear regression plots. These plots have been represented in three different interrelated pictures in order to facilitate a pair-wise comparison.

Two conclusions can be drawn from Figure 14. First, it is easy to see that the slopes of the regression lines increase with k. For our set of results, the obtained slopes are 0.03, 0.202, 0.213 and 0.293 for k = {2,3,4,5}, respectively. Second, the exponential dependence of the mean solution time on N seems to fit well according to the

experiments. In particular, the obtained mean square error of the regressions is 0.039 0.04, 0.002, and 0.11 for k equal to 2, 3, 4 and 5, respectively.
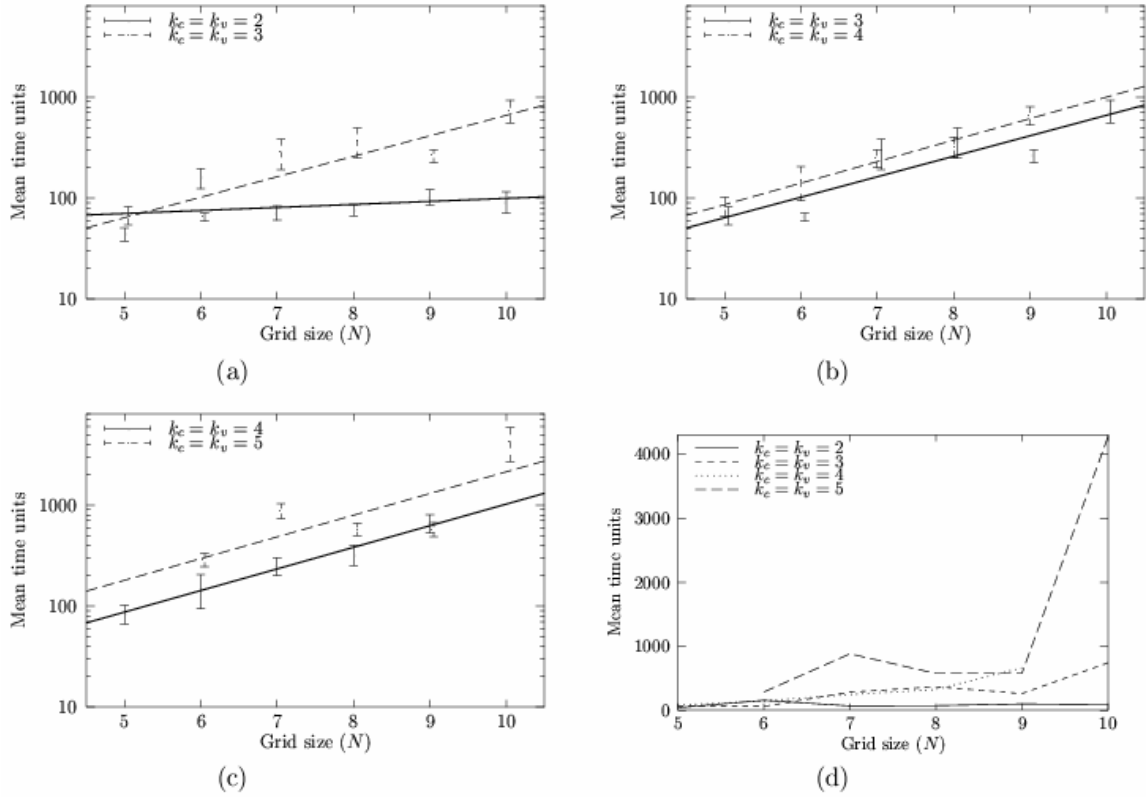
Figure 14: Mean solution time with respect to the order of the problem (size of the grid) for AWC on problem instances from the phase transition regions for $k_c = k_v = 2,3,4,5$.

2.7 Negotiation Techniques for Dynamic Systems

Considering the scalability of generic distributed negotiation protocols, our main concern was about feasibility of striving to optimize in problems with real-life sensor/targets settings, where time deadlines play a crucial role, and the targets being tracked are moving. More formally, the task of a tracking system can be specified as a dynamic GridDCSP-based problem, consisting of an ordered sequence of regular (static) GridDCSP-based problems, which are:

- Defined over the same set of sensors and having the same compatibility graph,
- Possibly differ in their sets of mobile targets (and thus obviously in their visibility graphs), and
- Each problem instance should be solved within a certain time window.

Attempting to address this problem, we conducted a simulation with 100 sensors that were supposed to track over time a continuously changing set of moving targets. The parameters used in this experiment have been chosen to represent a network of radars controlling some part of the airspace. As written, we considered a 10×10 uniform grid of sensors, with the distance of 10 miles between any two adjacent sensors, while the tracking area covered by these sensors is defined by the square of 8,100 square miles enclosed by the grid. The compatibility graph and the visibility graphs for all the static sub-problems were constructed to be *4*-restricted $(k_v = k_c = 4)$.
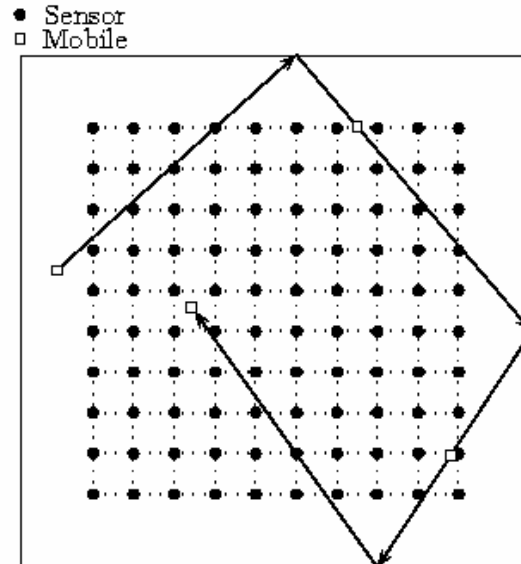
Figure 15: Graphical representation of the dynamic model, showing both the sensor grid, and the extended area in which the mobiles are moving and from the borders of which the movement is reflecting.

The targets in the simulation were moving in the grid according to independently chosen linear trajectories, where the velocity of all the targets was Mach 2 (1,500 miles/hour). Our intention was to keep a controlled, relatively tight ratio between the number of targets and the number of sensors, thus we strived to keep the (now expected) number of 18 targets inside the grid. On the other hand, we wanted to model both targets leaving the grid, and targets entering the grid, while keeping the movement of the targets independent of one another. To achieve this, we extended the number of targets to 36, setting these targets to move in (randomly initialized) linear trajectories inside an area larger than our sensor grid. This area has been modeled by a square of 16,200 square miles (twice as big as the square defined by the grid), and the center of this extended area is exactly the center of the grid (see Figure 15). For the first static sub-problem in the sequence, each target is located at a randomly chosen point inside this extended area, and is annotated with a randomly chosen linear trajectory, that determines the position of this sensor for the next sub-problem and so on. If, at some point, a target reaches the border of the extended area, it reflects from the border at a randomly chosen angle, which determines a new linear trajectory for this target. Such modeling of the target dynamics

provides us with a continuously changing set of targets inside the grid, while the expected size of this set is known (and is 18 targets in our experiment). The time window available to solve each sub-problem has been set to 1.2 seconds, i.e. the minimum time spent by a target inside a cell of the grid (given a speed of Mach 2) provides us at least 20 snapshots of a target during its presence in a particular cell.
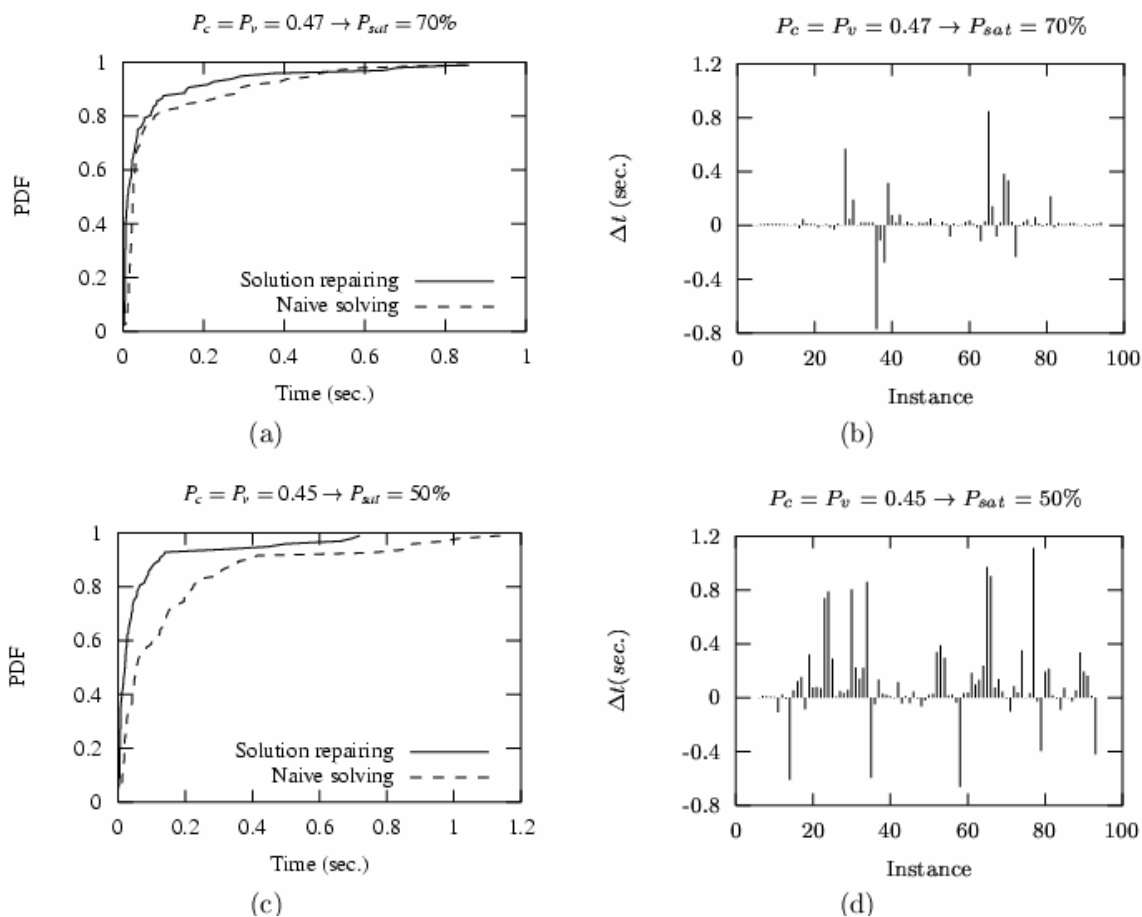
Figure 16: Dynamics of two problems, located at 70% and 50% of satisfiability ratio: (a) and (c) show the cumulative probability distributions for the solution repairing and the naive solving approach; (b) and (d) plot time differences to solve between the two approaches.

Figure 16 depicts the results for two dynamic GSensorDCSP problems Π_1 and Π_2, each consists of 100 static GSensorDCSP sub-problems, where the subproblems for Π_1 (Figure 16(a-b)) and Π_2 (Figure 16(c-d)) were selected from the regions of $P_{sat} \approx 0.7$ ($P_v = P_c = 0.47$) and $P_{sat} \approx 0.5$ ($P_v = P_c = 0.45$), respectively. $P_{sat} \approx 0.5$ corresponds to the region of the hardest instances. The dashed lines in Figures 16(a,c) depict the cumulative probability distributions of solving Π_i within a time window of t seconds. In Π_1, all the solvable sub-problems were solved in less than 0.9 seconds, while in Π_2 all except one sub-problems were solved within the time limit of 1.2 seconds.

22

Observe that, if no assumptions can be made about the connection between the mobiles in two subsequent sub-problems of Π_i, there is no particular reason to treat the sub-problems of Π_i differently than just solving them one by one *independently*, using one of the DisCSP algorithms. In what follows, we refer to this approach as to *naïve* solving of dynamic GridDCSP problems, and the results depicted by the dashed lines in Figures 16(a,c) correspond to this straightforward approach. However, mobile dynamics are typically far from being chaotic (linear trajectories in our experiment), *i.e.* the changes between the subsequent sub-problems are governed by some clear model of mobile dynamics. For instance, consider a network of radars controlling some airspace region. In such an application, it is reasonable to assume that if an aircraft becomes trackable by a sensor, then this aircraft is likely to remain trackable by this sensor in some near future.

One of our hypotheses was that continuity of the mobiles movement can be exploited in improving the performance of the tracking systems. An approach that a priori seems to be promising for dealing with such a problem $\Pi = \{\pi_1, \ldots, \pi_n\}$ is to initialize the search for π_i, $1 < i \leq n$, by the solution already achieved for π_{i-1} (comparatively to starting from an random assignment in AWC used in the naive approach). In what follows, we refer to this approach as to *solution repairing*. Note that in this approach, *nogoods* are not kept and are removed once a solution is obtained, so no additional synchronization is required between agents.

The central question is whether the contribution of solution repairing (versus the naive approach) is expected to be significant in real-life settings of both the mobiles dynamics, and the time available to solve each one of the static sub-problems. One experiment provides positive evidence to this question: The solid lines in Figures 16(a-c) depict the cumulative probability distributions of solving Π_i within a time window of t seconds using the solution repairing approach. It is easy to see that solution repairing clearly outperforms the naive approach, and Figures 16(b-d) illustrate this even better: For each sub-problem π_i, these graphs plot the difference between the times required to solve π_i using AWC from scratch and starting from the solution for π_{i-1}, if this exists (Δt). More interestingly, the results of our experiment show that the *relative attractiveness of solution repairing is higher in the region of harder instances*. For instance, using solution repairing, all the sub-problems of Π_2 were solved in less than 0.75 second. The reason could be that small changes in the problem setting (as the changes between π_i and π_{i+1} are expected to be) usually will not change significantly the placement of the solutions in the search tree. If so, then adopting solution repairing is likely to initialize the search at a node that is close to a solution node in the search tree. Likewise, the contribution of this property is likely to be more significant for sequences of harder problems, *i.e.* problems that a priori have less alternative solutions.

3. ANTS Autonomic Logistics - Resource Enabling in CAMERA

The CAMERA tool, developed by the ISI contractors in the scope of the autonomic logistics ANTS project, is a real-time resource management architecture providing a basis for integrating technologies developed by other contractors in the project. The general problem addressed by CAMERA is this of scheduling a set of *tasks*, conflicting requirements of which demand for various resources essential to fulfill the tasks. The resources can be of various types and nature (e.g. consumable vs. usable, mutually exclusive vs. sharable, etc.), and the tasks can have complex requirements involving various types of resources. The practical benchmark problem that has been used for development and testing for performance various components of the system is this of *scheduling flight operations* of a combat squadron and/or a set of squadrons. This specialized system, called SNAP, has been developed on top of the CAMERA generic tool. In terms of this domain, the tasks represent numerous flight missions and the resources stand for pilots, fuel, available time windows, etc.

The CAMERA tool integrates several alternative solutions for dealing with the core problem, and one of these solutions is based on:

1. Encoding the problem using a Pseudo-Boolean (PB) encoding (performed by the CAMERA sub-tool ATTEND, developed by A. Bugacov from ISI),

2. Solving the generated *pseudo-boolean problem* (*PBP*, for short) using an off-the-shelf PB solver, or a specialized solver developed by the CIRL project contractors. (While the behavior of different solvers is expected to vary on various PBPs, the actual *encoding is solver-independent*).

These two stages of the PBP-based solution are depicted (in the framework of CAMERA) in Figure 17 by the blue and purple frames, respectively.

While the ATTEND encoding addresses the core parts of safe, feasible scheduling and resource allocation between the tasks, the PBP generated by ATTEND is more constrained than the original problem (OP, for short). In this sense, any solution for the generated PBP is a solution for OP, but not vice versa, and thus some of the potentially valuable solutions for OP are sacrificed from the beginning. Below we describe the reasons for such a difference between PBP generated by ATTEND and the original problem that is provided to CAMERA.

Bridging this gap between the original problem and its pseudo-boolean encoding has been defined to be of highest interest of the SNAP/CAMERA users, and thus this problem was of our main focus.
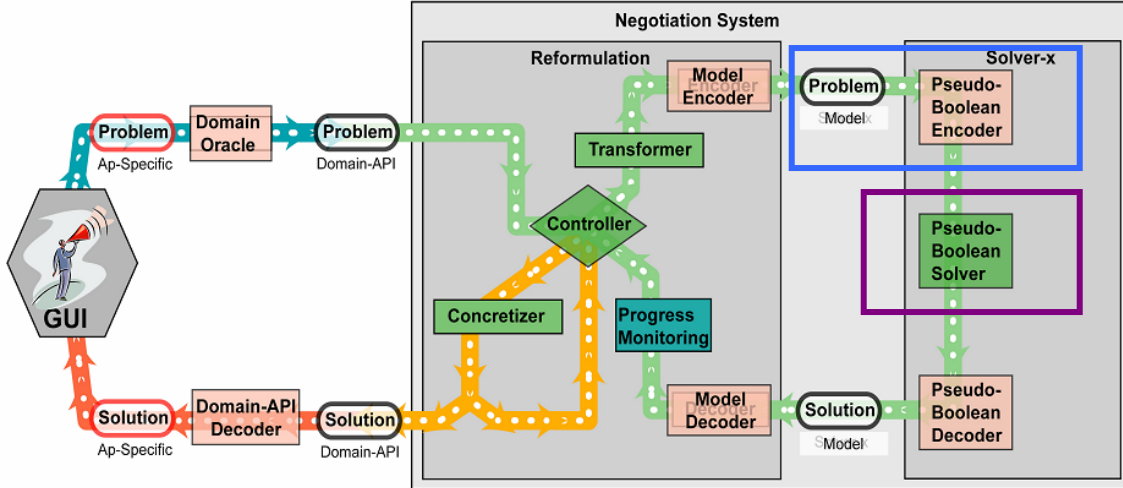
Figure 17: CAMERA/ATTEND architecture.

3.1 Resource Enabling – Formal Problem Statement

As we already mentioned, the flight tasks provided to SNAP have several requirements that require *pilots* as their main resources. Unlike other resources, not every pilot can be assigned to a requirement of a given task, but only pilots that fulfill *essential skills*. These essential skills are a static property of the task requirements and they are drawn from a static set of different skills $Q = \{Q_1, \ldots, Q_m\}$. More specifically, the essential skills of a task T (in what follows, by task we mean a requirement of a task) is a (possibly empty) set $\mathsf{Pre}(T)$ which is an *subset of* Q, representing a set of skills that a pilot should have in order to be counted as qualified for the task T.

In turn, in order to reason about pilots' qualifications, each pilot X is annotated with a (similarly defined) set $\mathsf{Have}(X) \subseteq Q$ describing the qualification history of X. In the SNAP flight operations scheduling problem, *no pilot can be assigned to a task without having the skills required by the task.* More formally, a pilot X can be assigned to a task T if and only if $\mathsf{Pre}(T) \subseteq \mathsf{Have}(X)$.

The crucial part of the flight operations scheduling problem is that the *tasks performed by the pilots provide them with additional qualifications*. Note that the generalization of this property is not unique to SNAP, but appears in many real-life scheduling problems (e.g., long-term students/courses allocation). More specifically, each task T is associated with a (possibly empty) set of skills $\mathsf{Provides}(T) \subseteq Q$, such that if a pilot X is assigned to perform T, then after performing T the qualification history of X is updated as follows:

$$\mathsf{Pre}(X) := \mathsf{Pre}(X) \cup \mathsf{Provides}(T)$$

25

The ATTEND PB encoder was taking into account only the qualification history of the pilots that exists *prior* to the scheduled set of tasks. Therefore, the pseudo-boolean satisfability problem generated by ATTEND ignored the skills that can be *dynamically* obtained by the pilots as a result of participating in tasks to which they are getting assigned.

Our group worked on overcoming the above limitation by taking advantage of the skills dynamically obtained by the pilots within the scheduled set of tasks. This extension turns out to be extremely non-trivial as it extends the general CAMERA/SNAP problem from the scheduling to a mixed *scheduling & planning [ST94]*. In addition, for several software legacy and research reasons, we have attempted to overcome the above limitation of CAMERA *without affecting the framework of ATTEND pseudo-boolean encoding*.

3.2 Resource Enabling via Extended Pseudo-Boolean Encoding

The *resource enabling pseudo-boolean encoding* that we have developed properly extends the ATTEND encoding currently used in CAMERA. Below we list the central axioms of our extension, omitting several technical details that ease the presentation.

First, let us introduce the variables used in the encoding. and for each skill $Q \in Q$:

Variable	Semantics
$X_{i,k}$	For each qualifiable resource X (i.e., pilot in case of SNAP), for each task T_i and for each time slot k, the variable $X_{i,k}$ represents assigning X to T_i at time slot k. Note that the variables $X_{i,k}$ are already used by the current ATTEND pseudo-boolean encoding.
$[\![X,Q]\!]_k$	For each qualifiable resource X, for each skill Q, and for each time slot k, the variable $[\![X,Q]\!]_k$ encodes the proposition that at the time slot k, X has the skill Q in his qualification history.

Our first axiom conditions assignment of resources to tasks on the skills that are required by the latter. More formally, for each task T_i, for each resource X, and for each time slot k, we have:

$$\bigcup_{Q \in \mathsf{Pre}(T_i)} \overline{[\![X,Q]\!]_k} \quad \rightarrow \quad \overline{X_{i,k}}$$

encoded [1] as:

$$\alpha_i \overline{X_{i,k}} + \sum_{Q \in \mathsf{Pre}(T_i)} [\![X,Q]\!]_k \geq \alpha_i \qquad (1.3)$$

where $\alpha_i = |\mathsf{Pre}(T_i)|$. Informally, Eq. 1.3 encodes an intuitive axiom that if X does not have the skills essential to perform T_i, then X should not be assigned to T_i.

The subsequent two axioms correspond to the planning part of the problem, and in particular they embed the planning "frame axioms". The first planning axiom is that for each resource X, for each skill Q, and for each time slot k, we have:

$$[\![X,Q]\!]_k \vee \bigcup_{T_i \in \mathsf{Provide}(Q)} X_{i,k} \rightarrow [\![X,Q]\!]_{k+1}$$

where $\mathsf{Provide}(Q)$ stands for the set of all tasks T such that $Q \in \mathsf{Provides}(T)$. This axiom is encoded as:

$$\beta_Q [\![X,Q]\!]_{k+1} + \overline{[\![X,Q]\!]_k} + \sum_{T_i \in \mathsf{Provide}(Q)} \overline{X_{i,k}} \geq \beta_Q \qquad (1.4)$$

where $\beta_Q = |\mathsf{Provide}(Q)| + 1$. Informally, this axiom states that "if you already have a certain skill, or now you are doing something that provides this skill, then at the next time slot you will have this skill".

The second planning axiom accomplishes the first planning axiom, and it states that for each resource X, for each skill Q, and for each time slot k, we have:

$$\overline{[\![X,Q]\!]_k} \wedge \bigcap_{T_i \in \mathsf{Provides}(Q)} \overline{X_{i,k}} \rightarrow \overline{[\![X,Q]\!]_{k+1}}$$

encoded as:

$$[\![X,Q]\!]_{k+1} + \overline{[\![X,Q]\!]_k} + \sum_{T_i \in \mathsf{Provide}(Q)} X_{i,k} \geq 1 \qquad (1.5)$$

---

[1] In what follows, by $\bigcup$ or $\vee$, and by $\bigcap$ or $\wedge$, we denote *disjunction* and *conjunction*, respectively.

Informally, this axiom states that "if you don't have a certain skill, and you are not doing something that will provide you this skill, then at the next time slot you still will not have this skill".

Together with some additional technical details, this set of axioms (added to the current system of ATTEND's pseudo-boolean constraints) provides necessary and sufficient conditions for correct resource enabling within the CAMERA tool and the SNAP system.

3.3 Extended Pseudo-Boolean Encoding with Changing Time Resolution

Above we described the model of pilot qualifications, allowing the pilots to perform these tasks. The flight tasks provided to SNAP have several requirements that in particular require *pilots* as their main resources. Unlike other resources, not every pilot can be assigned to a requirement of a given task, but only pilots that fulfill some *essential skills*. Prior to having extended encoding E1 described in Section 3.2, the ATTEND PB encoder took into account only the qualification history of the pilots that exists *prior* to the scheduled set of tasks. Overcoming the above limitation by taking advantage of the skills *dynamically* obtained by the pilots within the scheduled set of the tasks has been addressed within the extended encoding E1. Here we extend this encoding to deal with changing time resolution, and henceforth this enhanced encoding is denoted E2.

The problem of a deep time horizon is solved in ATTEND via abstraction of the time resolution. Before the abstraction, for each time slot and for each pilot, we have information whether this pilot is available within this time slot. As a result of time abstraction, the availability of a pilot within a certain time slot can be replicated, schematically resulting in having *several instances of the same pilot within a particular time slot*. The problematic aspect of this problem representation is that:

1. From the scheduling point of view, each pilot instance is treated as an independent pilot, but

2. From the resource enabling point of view, all the instances of the same pilot should be treated as the same pilot (i.e. skills obtained by one pilot instance should be propagated to all other instances).

Because of this asymmetry, the naive approach of using the extended encoding E1 at the level of pilot instances (instead of at the level of pilots) is incomplete, and the quality of the solutions provided by this approach is expected to be extremely low. Therefore, we have developed a new formal model for resource enabling (E2) that successfully deals with anytime abstraction. Likewise, for several software legacy and research reasons, E2 *has no affect on the framework of ATTEND pseudo-boolean encoding*, leaving the latter completely resource enabling independent.

Below we list the central axioms of the E2 resource enabling model, leaving out several technical details to ease the presentation.

First, let us introduce the variables used in the encoding. The main difference here comparatively to E1 is the variables $X_{i,k}^{(j)}$, corresponding to the instance of the pilots. Formally, for each pilot X and for each skill $Q \in Q$:

Variable	Semantics
T_i	Task fulfillment variable: *true* if T_i is successfully scheduled and *false* otherwise.
$X_{i,k}^{(j)}$	For each qualifiable resource X (i.e., pilot in case of SNAP), for each task T_i, for each time slot k, and for each instance j of X available in time slot k, the variable $X_{i,k}^{(j)}$ represents assigning the instance j of X to T_i at time slot k.
$[\![X,Q]\!]_k$	For each qualifiable resource X, for each skill Q, and for each time slot k, the variable $[\![X,Q]\!]_k$ encodes the proposition that at the time slot k, X has the skill Q in his qualification history.

Now, let us introduce the axioms forming the formal model of E2, together with pseudo-boolean realization of those axioms. Note that the principles behind the ideas of the E2 axioms bare some similarity with these of E1. However, the axioms themselves and their PB realization are completely different in E2 compared to E1.

The first axiom that we introduce into the problem conditions assignment of resource instances to tasks on the skills that are required by the latter. More formally, let $I_{X,k}$ be the number of instance of X available in time slot k. Now, for each resource X and time slot k, if $I_{X,k} > 0$, then, for each task T_i, we have

$$
\bigcup_{Q \in \mathsf{Pre}(T_i)} \overline{[\![X,Q]\!]_k} \ \rightarrow \ \bigcap_{j=1}^{I_{X,k}} \overline{X_{i,k}^{(j)}}
$$

This axiom is encoded as a set of $|\mathsf{Pre}(T_i)|$ constraints. Namely, for each $Q \in |\mathsf{Pre}(T_i)|$, we have:

$$
I_{X,k}\,[\![X,Q]\!]_k \ + \ \sum_{j=1}^{I_{X,k}} \overline{X_{i,k}^{(j)}} \ \geq \ I_{X,k} \tag{1.6}
$$

Informally, Eq. 1.6 encodes the requirement that if X does not have the skills essential to perform T_i, then no instance of X can be assigned to T_i.

The subsequent two axioms correspond to the planning part of the problem, and in particular they embed the planning "frame axioms". Informally, the first planning axiom states that "*if the resource already have a certain skill, or currently one of this resource instances is involved in a task that provides this skill, then at the next time slot the resource will have this skill*".

Formally, for each resource X, and for each time slot k, the axiom is as follows.
If $I_{X,k} > 0$, then, for each skill Q, we have:

$$ [\![X,Q]\!]_k \ \lor \ \bigcup_{j=1}^{I_{X,k}} \ \bigcup_{T_i \in \mathsf{Provide}(Q)} X_{i,k}^{(j)} \ \rightarrow \ [\![X,Q]\!]_{k+1} $$

where $\mathsf{Provide}(Q)$ stands for the set of all tasks T such that $Q \in \mathsf{Provides}(T)$. This axiom is encoded as:

$$ \beta_{Q,X} [\![X,Q]\!]_{k+1} + \overline{[\![X,Q]\!]_k} + \sum_{T_i \in \mathsf{Provide}(Q)} \sum_{j=1}^{I_{X,k}} \overline{X_{i,k}^{(j)}} \ \geq \ \beta_{Q,X} \tag{1.7} $$

where

$$ \beta_{Q,X} = I_{X,k} \cdot |\mathsf{Provide}(Q)| + 1 $$

Otherwise, if $I_{X,k} = 0$, then, for each skill Q, we have:

$$ [\![X,Q]\!]_k \ \rightarrow \ [\![X,Q]\!]_{k+1} $$

encoded as:

$$ [\![X,Q]\!]_{k+1} + \overline{[\![X,Q]\!]_k} \ \geq \ 1 $$

The second planning axiom informally states that "*if the resource don't have a certain skill, and non of its instances is involved in a task that provides this skill, then at the next time slot the resource still will not have this skill*". This axiom accomplishes the first

planning axiom, and, formally, for each resource X, and for each time slot k, the axiom is as follows.

If $I_{X,k} > 0$, then, for each skill Q, we have:

$$\overline{[\![X,Q]\!]}_k \ \wedge \ \bigcap_{T_i \in \mathsf{Provide}(Q)} \bigcap_{j=1}^{I_{X,k}} \overline{X_{i,k}^{(j)}} \ \rightarrow \ \overline{[\![X,Q]\!]}_{k+1}$$

encoded as:

$$[\![X,Q]\!]_k + \overline{[\![X,Q]\!]}_{k+1} + \sum_{T_i \in \mathsf{Provide}(Q)} \sum_{j=1}^{I_{X,k}} X_{i,k}^{(j)} \ \geq \ 1 \qquad (1.8)$$

Otherwise, if $I_{X,k} = 0$, then, for each skill Q, we have:

$$\overline{[\![X,Q]\!]}_k \ \rightarrow \ \overline{[\![X,Q]\!]}_{k+1}$$

encoded as

$$[\![X,Q]\!]_k + \overline{[\![X,Q]\!]}_{k+1} \ \geq \ 1$$

Together with some additional technical details, this set of axioms (added to the current system of ATTEND's pseudo-boolean constraints) provides necessary and sufficient condition for correct resource enabling with time abstraction within the CAMERA tool and the SNAP system. However, for the proper integration with the core ATTEND pseudo-boolean encoding, we extend our system with two additional sets of auxiliary constraints. These sets of constraints ensure that the solution to the PB problem will contain no redundant assignments of resource instances to tasks. This requirement is not necessary for the basic ATTEND/CAMERA encoding, but it is necessary for resource enabling with time abstraction.

The first set of auxiliary constraints is specified as follows. For each task T_i, we have:

$$-\Im \cdot \overline{T_i} + \sum_{X} \sum_{k:I_{X,k}>0} \sum_{j=1}^{I_{X,k}} X_{i,k}^{(j)} \leq \alpha_i \qquad (1.9)$$

where $\Im$ is the total number of elements in the internal summation, and

$$\alpha_i = \eta_i \cdot \delta_i$$

where η_i is the number of resources required by T_i, and δ_i is the length of T_i (in time slots).

The second set of auxiliary constraints accomplishes the first one by requiring:

$$-\Im \cdot T_i + \sum_{X} \sum_{k:I_{X,k}>0} \sum_{j=1}^{I_{X,k}} X_{i,k}^{(j)} \leq 0 \qquad (1.10)$$

where $\Im$ is defined as above.

3.5 Complexity, Implementation, and Integration

Taking advantage of the skills dynamically obtained by the pilots within the scheduled set of the tasks, especially in face of abstracting the time axis, turns to be extremely non-trivial as it extends the general CAMERA/SNAP problem from the scheduling to a mixed *scheduling & planning [SFJ00]*. The conceptual difference between planning and scheduling can be formalized as follows:

- In scheduling, all the interdependencies between the scheduled entities, and the properties of the resources are completely static. Therefore, assigning some tasks can only make other tasks more constrained.

- In planning, the opposite is true, thus constrainedness of the problem is not monotonic anymore, and assigning tasks can make other tasks less/differently constrained.

From the worst-case complexity point of view, planning is harder than scheduling (PSPACE-complete *[Byl94]* vs. NP-complete *[GJ78]*). By enhancing the core CAMERA/ATTEND problem with resource enabling we create hybrid planning/scheduling problem, thus careful exploiting of the problem structure should be done in order to avoid dramatic performance degradation. Since the enhanced problem is represented similarly to the original problem in pseudo-boolean encoding, the complexity blow-up can appear in the size of the problem description (i.e. number of constraints required to encode the problem). Below we show that this is not the case with the E2 encoding.

Table 3 summarizes the parameters of the problem, and Table 4 presents the description complexity of the (1) original problem, (2) extension, and (3) combination of (1) and (2).

n	# tasks
m	# resources
k	maximal # instances for a resource
t	# time slots (after abstraction)
q	# dynamically obtainable skills

Table 3: Parameters of the Autonomous Logistic problems.

	ATTEND	E2	ATTEND + E2
Variables	$O(nmkt)$	$O(qmkt)$	$O(mkt(n+q))$
Constraints	$O(nmkt)$	$O(qnmkt)$	$O(qnmkt)$

Table 4: Description complexity of the encodings.

surge19-with-enablement.snap* v2.1(Unofficial build)

File Edit Mission Commands Reports

VMA-513 (2003-05-27T10:56:27-07:00) [Run SNAP] Done running the Serial Solver in 6 seconds

Framework | Pilots | Aircraft | Operational Planning | Missions | Missions Timeline | Metrics | Schedule Editor

Search: Show all items ▼   17 items.

F S V Rul P	Origin	Ty	Takeoff	Stage	Pilots	Locations	External Sup	Cl
01	User	↕	Mon 9 Jun 20:05	NS, FAM	Duncan: 284 pit 203, CarlsonF: 285 pit 203	base	FAC pit	1.0, 0.
	User	↕	Mon 9 Jun 05:45	OAS, FA	Cameresi: 313 pit 203, Edwards: 240 pit 203	base	FAC pit	0.0, 0.
	User	↕	Mon 9 Jun 07:00	OAS, FA	Ritterby: 314 pit 203, Dowell: 241 pit 203	base	FAC pit	0.9, 0.
	User	↔	Mon 9 Jun 10:15	OAS	Parkhurst: 314, Smith: 241	base	FAC	1.6, 0.
	User	↔	Mon 9 Jun 13:15	OAS	Duncan: 313, Edwards: 240	base	FAC	0.3, 0
	User	↕	Mon 9 Jun	AA, FAM	—: 264 pit 203, Bergrud: 264 pit 203	base	pit	—, 0.3
02	User	↔	Tue 10 Jun 06:00	OAS	Cameresi: 313, Edwards: 240	base	FAC	0.0, 0
	User	↔	Tue 10 Jun 07:00	OAS	Ritterby: 314, Dowell: 241	base	FAC	0.0, 0
	User	↕	Tue 10 Jun 20:06	NS, FAM	Duncan: 284 pit 203, CarlsonF: 285 pit 203	base	FAC pit	0.0, 0
	User	↕	Tue 10 Jun 09:00	OAS, FA	Parkhurst: 313 pit 203, Knotts: 240 pit 203	base	FAC pit	0.0, 0.
	User	↕	Tue 10 Jun 13:30	OAS, FA	Duncan: 314 pit 203, Dowell: 240 pit 203	base	FAC pit	0.9, 0
	User	↕	Tue 10 Jun	AA, FAM	—: 265 pit 203, Bergrud: 265 pit 203	base	pit	—, 0.6
03	User	↕	Wed 11 Jun 06:00	OAS, FA	Cameresi: 314 pit 203, Edwards: 240 pit 203	base	FAC pit	0.0, 0
	User	↕	Wed 11 Jun 20:06	NS, FAM	Duncan: 284 pit 203, CarlsonF: 285 pit 203	base	FAC pit	0.0, 0
	User	↕	Wed 11 Jun 07:00	AA, FAM	Ritterby: 264 pit 203, Rountree: 264 pit 203	base	pit	0.0, 0.
	User	↕	Wed 11 Jun 10:30	AA, FAM	Parkhurst: 264 pit 203, Dowell: 264 pit 203	base	pit	0.3, 0.
	User	↕	Wed 11 Jun	AA, FAM	—: 331 pit 203, Bergrud: 266 pit 203	base	GCI/Adversary,Adv	—, 0.9

(a)

surge19-with-enablement.snap* v2.1(Unofficial build)

File Edit Mission Commands Reports

VMA-513 (2003-05-27T11:07:05-07:00) [Run SNAP] Done running the Combined Pseudo-Boolean-Then-Serial Solver in 168 seconds

Framework | Pilots | Aircraft | Operational Planning | Missions | Missions Timeline | Metrics | Schedule Editor

Search: Show all items ▼   17 items.

F S V Rul P	Origin	Ty	Takeoff	Stage	Pilots	Locations	External Sup	Cl
01	User	↕	Mon 9 Jun 21:00	NS, FAM	Parkhurst: 284 pit 203, Smith: 285 pit 203	base	FAC pit	1.3, 0.
	User	↕	Mon 9 Jun 07:30	OAS, FA	Ritterby: 313 pit 203, Rountree: 240 pit 203	base	FAC pit	0.0, 0.
	User	↕	Mon 9 Jun 06:00	OAS, FA	Duncan: 314 pit 203, Dowell: 241 pit 203	base	FAC pit	1.2, 0.
	User	↔	Mon 9 Jun 10:30	OAS	Cameresi: 314, CarlsonF: 241	base	FAC	0.0, 0.
	User	↔	Mon 9 Jun 18:00	OAS	Cameresi: 313, Ward: 240	base	FAC	0.0, 0.
	User	↕	Mon 9 Jun 13:30	AA, FAM	Ritterby: 264 pit 203, Bergrud: 264 pit 203	base	pit	0.0, 0.
02	User	↔	Tue 10 Jun 10:30	OAS	Cameresi: 313, Ward: 240	base	FAC	0.0, 0
	User	↔	Tue 10 Jun 13:30	OAS	Ritterby: 314, Duncan: 241	base	FAC	0.9, 0
	User	↕	Tue 10 Jun 21:30	NS, FAM	Parkhurst: 284 pit 203, Smith: 285 pit 203	base	FAC pit	0.0, 0
	User	↕	Tue 10 Jun 06:00	OAS, FA	Ritterby: 313 pit 203, Knotts: 240 pit 203	base	FAC pit	0.0, 0.
	User	↕	Tue 10 Jun 07:00	OAS, FA	Duncan: 314 pit 203, Dowell: 240 pit 203	base	FAC pit	0.0, 0
	User	↕	Tue 10 Jun 17:00	AA, FAM	Cameresi: 265 pit 203, Bergrud: 265 pit 203	base	pit	0.3, 0.
03	User	↕	Wed 11 Jun 07:30	OAS, FA	Duncan: 314 pit 203, Ward: 240 pit 203	base	FAC pit	0.0, 0
	User	↕	Wed 11 Jun 21:30	NS, FAM	Parkhurst: 284 pit 203, Smith: 285 pit 203	base	FAC pit	0.0, 0
	User	↕	Wed 11 Jun 16:30	AA, FAM	Parkhurst: 264 pit 203, Rountree: 264 pit 203	base	pit	0.3, 0.
	User	↕	Wed 11 Jun 06:00	AA, FAM	Ritterby: 264 pit 203, Dowell: 264 pit 203	base	pit	0.0, 0.
	User	↕	Wed 11 Jun 12:00	AA, FAM	Ritterby: 331 pit 203, Bergrud: 266 pit 203	base	GCI/Adversary,Adv	2.7, 0.

(b)

Figure 18: Snapshots of the CAMERA tool presenting results of negotiations without (a) and with (b) extended encoding E2.

Table 4 shows that, from ATTEND to ATTEND+E2 the size of the problem description grows linearly in the number of different dynamically obtainable skills. The evaluation of the extended encoding within the CAMERA tool shows that the real-life enhanced problems are solvable without exponential blow-up in time complexity.

The extended encoding E2 has been completely implemented, integrated and tested within the CAMERA tool, and evaluated on the real-life problems of scheduling flight operations of a combat squadron and/or a set of squadrons. *The results of the evaluation (demonstrated during the ANTs demo on June, 3, 2003) clearly showed that the extended encoding enhanced the qualitative capabilities of the CAMERA tool.* For instance, consider Figure 18 that presents screen snapshots of the CAMERA tool. Both figures show the optimal schedules produced by a pseudo-boolean solver for the same flight operations scheduling problem on a squadron level. However, Figure 18(a) shows the resulting schedule for the plain ATTEND encoding of the problem, while Figure 18(b) shows the resulting schedule for the E2-enhanced encoding. In the first case, the optimal schedule left 3 (out of 17) tasks unscheduled, while in the second case all 19 tasks have been scheduled successfully.

4. Identifying and Exploiting Critical Resources

Our collaboration with ISI on extending CAMERA/ATTEND system (specifically, extending the pseudo-boolean encodings) inspired a novel research devoted to *identifying and exploiting critical resources.*

Most interesting AI formalisms for reasoning, planning, and learning have been shown to be worst-case intractable. Such negative complexity results led to an extensive search for tractable subclasses of the general formalisms (e.g. see the work of our group on GridDCSP discussed in Section 2.5). Unfortunately, these tractable subclasses are often too restrictive for real-world applications, thus we saw the emergence of a more practical approach to computationally hard problems in AI, with the introduction of fast satisfiability solvers and fast constraint based reasoning methods. Somewhat surprisingly, on practical problem instances these methods scale well beyond what one might expect based on a formal complexity analysis. In fact, current state-of-the-art SAT solvers can handle problem instances, as they arise in scheduling, planning, and finite model-checking, with up to a million variables and five million clauses *[Chaff01].* The success of these methods appears to hinge on a combination of two factors:

1. Practical combinatorial problem instances generally have a substantial amount of (hidden) tractable sub-structure, and

2. New algorithmic techniques exploit such tractable structure, through, *e.g.*, randomization and constraint learning.

These developments suggest that a standard worst-case complexity analysis does not capture well the true complexity of typical problem instances encountered in practical applications. Theoretical computer scientists have been well-aware of the limitations of worst-case complexity results and have explored alternatives, such as average-case complexity and smoothed analysis *[ST01]*. In average-case analysis, one studies the computational cost of solving problem instances drawn from a predefined problem distribution. Such an analysis clearly can provide valuable insights. However, the relatively basic distributions for which one can obtain average-complexity results appear to be quite far removed from the instance distributions one encounters in practice. In fact, formally defining the distribution of real-world problem instances is generally an open problem in itself. Smoothed analysis attempts to unify worst-case and average-case, but suffers from limited applicability: it works well on algorithms for problems defined over dense fields such as the simplex algorithm, but the applicability of smoothed analysis on discrete problem domains is unclear.

In our work, we pursued an alternative approach of identifying special structural properties common to known problem instances and showing how clever algorithms can exploit such properties. Informal insights about what such special structure might be are currently already used in the design of, for example, branching and variable choice heuristics in combinatorial search methods. A common feature of these techniques is an understanding that *different groups of variables in a problem encoding often play quite distinct roles*. For example, at the highest level, one can distinguish between dependent and independent variables. The dependent or auxiliary variables are needed to obtain compact problem encodings but the true combinatorics arises from the independent variables; *e.g.*, the independent variables in an encoding of a planning domain represent the various actions applicable in a given state of the world, whereas the dependent variables encode the consequences of selecting a particular action. Another powerful intuition in the design of search methods is that one wants to select variables that *simplify the problem instance as much as possible* when these variables are assigned values. This intuition leads to the common heuristic of branching on the most-constrained-variable first.

These general insights have been incorporated in state-of-the-art constraint solvers, and their effectiveness has been demonstrated empirically on a significant number of benchmark problems. However, a more formal underpinning explaining the practical success of these strategies has been lacking. In our work *[WGS03a,WGS03a]*, we introduced a formal framework directly inspired by these techniques and present rigorous complexity results that support their effectiveness.

1. We formalized the notion of *critical resources*, introducing the notion of *backdoor variables*. This is a set of variables for which there is a value assignment such that the simplified problem can be solved by a poly-time algorithm, called the ``sub-solver''. The sub-solver captures any form of poly-time simplification procedure as used in current CSP solvers. We also consider the notion of a *strong backdoor* where any setting of the backdoor variables leads to a poly-time solvable sub-problem. The set of all problem variables forms a trivial

backdoor set, but many interesting practical problem instances possess much smaller backdoors and stronger backdoors.

2. We studied backdoors in several practical problem instances, and identify backdoors that contain only a fraction of the total number of variables. For example, we showed that the CSP encoding of a standard benchmark logistic planning problem contains a backdoor with only 12 variables out of a total of nearly 7,000 variables. When given a set of backdoor variables of a problem instance, one can restrict the combinatorial search by branching only on the backdoor variables and thus search a drastically reduced space.

3. In general, finding the set of backdoor variables for a problem instance is itself a computationally hard problem. The key contribution of our work is that we formally showed that *backdoor variables still provide a concrete computational advantage, even when taking into account the cost of searching for such variables.* We analyze three scenarios:

    a. A <u>deterministic scenario</u> with an exhaustive search of backdoor sets. We show that one obtains provably better search complexity when the backdoor contains up to a certain fraction of all variables.
    b. A <u>randomized</u> search technique, which in effect repeatedly guesses backdoor sets. We showed that this technique provably outperforms a deterministic search.
    c. <u>Variable selection heuristic</u>, which provides guidance towards the backdoor set. This strategy can yet further reduce the search space.

B(n)	deterministic	randomized	heuristic
n/k	small *exp(n)*	smaller *exp(n)*	tiny *exp(n)*
$O(\log(n))$	$\left(\dfrac{n}{\sqrt{\log(n)}}\right)^{O(\log(n))}$	$\left(\dfrac{n}{\log(n)}\right)^{O(\log(n))}$	*poly(n)*
$O(1)$	*poly(n)*	*poly(n)*	*poly(n)*

Table 5: Time bounds for solving CSPs in the various scenarios considered in this work.

Table 5 gives a high-level summary of the results, showing time bounds for solving CSPs in the various scenarios considered in this work. *B(n)* is an upper bound on the size of the smallest backdoor, where *n* is the number of variables in the problem. *k* is a fixed constant. Our empirical results suggest that for practical instances the backdoor is often a relatively small fraction of *n*, e.g., *n/100*, or even of size *log(n)*. By exploiting restart strategies, we can identify a polynomially solvable case when the backdoor contains at most *log(N)* variables. We believe that this final scenario is closest to the behavior of current effective constraint solvers. Our formal analysis also suggests several novel algorithmic strategies that warrant further empirical exploration.

5. Conclusions

The work of our group in the ANTS program has been devoted to connecting frameworks for multi-agent negotiation-based systems with the research on analytical and empirical computational complexity. Our general goal was to improve the expressiveness and scalability of complex distributed systems by exploiting computational hardness awareness in both the design and operation of these systems. In particular, our goal was to study the impact of problem structure on the computational complexity of the problem, and to exploit this connection in both problem modeling and solving.

Considering our work in both ANTS Sensor Networks and Autonomic Logistics domains, we believe that our goals were successfully achieved. On the one hand, our formal and empirical results were motivated and shown to be directly applicable to the benchmark domains of the program. For instance, our formal modeling techniques improved the robustness and scalability of the prototype multi-agent systems for autonomic logistics, and our insights on "initiated message delays" affected the negotiation protocols developed by other contractors in the program. On the other hand, our domain specific results provided us with a platform for further generalization, leading to several generic research contributions that have been already used in the wide scientific community.

References

[Grid04]   Béjar, R., Fernández, C., Domshlak, C., Gomes, C. P., Krishnamachari, B., Selman, B., and Valls, M., Sensor networks and distributed CSP: Communication, computation and complexity", *Artificial Intelligence*, 2004. (in print).

[BKGS01]   Bejar, R., Krishnamachari, B., Gomes, C., and Selman, B, Distributed constraint satisfaction in a wireless sensor tracking system, In *Proceedings of the IJCAI-01 Workshop on Distributed Constraint Reasoning*, 2001.

[Byl94]   Bylander, T., The computational complexity of propositional STRIPS planning, *Artificial Intelligence*, 69(1-2). 165-204, 1994.

[CB97]   Crovella, M. and Bestavros, A., Self-similarity in World Wide Web traffic: Evidence and possible causes, *IEEE Transactions on Networking*, 5(6), 835-846, 1997.

[GJ78]   Garey, M. R., and Johnson, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, 1978.

[GSK98]   Gomes, C. P., Selman, B., and Kautz, H. A., Boosting combinatorial search through randomization, In *Proceedings of the National Conference on Artificial Intelligence*, 431-437, 1998.

[JBP96]   Junius, M., Buter, M., Pesch, D, CNCL - Communication Networks Class Library. Aachen University of Technology. 1996.

[LTWW94]   Leland, W. E., Taqqu, M. S., Willinger, W. and Wilson, D. V., On the self-similar nature of Ethernet traffic, *IEEE Transactions on Networking*, 2(1), 1-15, 1994.

[MZKST99]   Monasson, R., Zecchina, R., Kirkpatrick, S., Selman, B.,and Troyansky, L., Determining Computational Complexity from Characteristic Phase Transitions, *Nature*, 400, 133-137, 1999.

[Chaff01]   Moskewitcz, C., Madigam, C., Zhao, Y., Zhang, L., and Malik, S., Chaff: Engineering an Efficient SAT Solver, In *Proceedings of 41st Design Automation Conference*, 2001.

[Paxson97] Paxson, V., Fast, approximate synthesis of fractional Gaussian noise for generating self-similar network traffic, *Computer Communication Review*, 27(5), 5-18, 1997.

[ST94]   Samorodnitsky, G. and Taqqu, M. S., *Stable Non-Gaussian Random Processes*, Chapman & Hall, 1994.

[SFJ00]   Smith, D., Frank, J., and Jonsson, A., Bridging the gap between planning and scheduling, *Knowledge Engineering Review*, 15(1), 2000.

[ST01]   Spielman, D., and Teng, S.-H., Smoothed analysis: Why the simplex algorithm usually takes polynomial time, In *Proceedings of the 33rd ACM Symposium on Theory of Computing*, 296-305, 2001.

[Walsh02]   Walsh, T., From P to NP: COL, XOR, NAE, 1-in-k, and Horn SAT, In *Proceedings of The Eighteenth National Conference on Artificial Intelligence*, 695-700, 2002.

[WGS03a]   Williams, R., Gomes, and C., Selman, B., Backdoors to typical case complexity*, In Proceedings of International Joint Conference on Artificial Intelligence*, 2003.

[WGS03b]   Williams, R., Gomes, C., and Selman, B., On the connections between backdoors, restarts, and heavy-tailedness in combinatorial search, In *Proceedings of the Sixth International Conference on Theory and Applications of Satisfiability Testing,* 2003.

[Yokoo95]   Yokoo, M., Asynchronous weak-commitment search for solving distributed constraint satisfaction problems, In *Proceedings of the First International Conference on Principles and Practice of Constraint Programming*, 88-102, 1995.

[YDIK92]   Yokoo, M., Durfee, E. H., Ishida, T., and Kuwabara, K., Distributed constraint satisfaction for formalizing distributed problem solving, In *Proceedings of the Twelfth IEEE International Conference on Distributed Computing Systems*, 614-621, 1992.

[YDIK98]   Yokoo, M., Durfee, E. H., Ishida, T. and Kuwabara, K., The distributed constraint satisfaction problem: Formalization and algorithms, *IEEE Transactions on Knowledge Data Engineering*, 10(5), 673-685, 1998.

[YH00]   Yokoo, M. and Hirayama, K., Algorithms for distributed constraint satisfaction: A review, *J. Autonomous Agents and Multi-Agent Systems*, 3(2), 198-212, 2000.