Using privacy loss to guide decisions in distributed CSP search *

Richard J. Wallace and Marius C. Silaghi
Cork Constraint Computation Center, University College Cork, Cork, Ireland
Department of Computer Sciences, Florida Institute of Technology, Melbourne, FL
email: r.wallace@4c.ucc.ie,msilaghi@cs.fit.edu

Abstract

In cooperative problem solving, the communication necessary for solution search can also lead to privacy loss on the part of the agents involved. Such loss can be assessed either by directly tallying the number and importance of specific items of information revealed or by tracking reductions in the set of possible values associated with a particular item of information. In both cases information loss can occur either because of direct communication or by inferences that other agents make from one's communications. The results of these inferences are stored in the "views" that agents have of other agents. In the present work on distributed constraint solving, such views are organized as extensions of normal CSP representations that model information about possible values in unknown CSPs of other agents. Here we show how this approach can be extended so that agents also maintain views of other agents' views of themselves; the latter are called "mirror views". Mirror views can in turn be used to monitor one's own privacy loss, and can support strategies designed to reduce the loss of particular kinds of private information. Experiments with a simulated meeting scheduling system show that it is possible to reduce privacy loss with strategies based on mirror views.

Introduction

When agents collaborate on a problem solving task, the assumption is often made that any requisite information will be shared. In many settings, however, agents may want to maintain their privacy as much as possible while still engaging in collaborative problem solving. This raises the question of how to meet the added requirement of privacy maintenance while still trying to solve problems efficiently.

Privacy issues were, in fact, a driving force behind the development of distributed algorithms for constraint solving (Yokoo 1998). However, only in recent years has this concern been taken into account in algorithm design and evaluation (Freuder, Minca, & Wallace 2001)(Silaghi, Sam-Haroud, & Faltings 2000) (Yokoo, Suzuki, & Hirayama 2002).

The majority of existing algorithms for solving distributed constraint satisfaction problems are based on constructive search. Despite the fact that they claim to support privacy requirements, none of them offers support for strategies that would attempt to reveal less important data in favor of saving sensitive information. Only after such support for privacy-related strategies is developed, will existing algorithms be able to exploit their potential to reduce privacy loss (Bella & Bistarelli 2001) (Silaghi & Faltings 2002).

Privacy issues are complicated by the fact that agents may make deductions about aspects of another agent's problem even under conditions of limited communication. This can be done by reasoning in terms of sets of possibilities regarding problem elements, some of which can be ruled out depending on the information communicated. For example, in a meeting scheduling scenario, if a proposed meeting is accepted by another agent, this rules out certain possibilities regarding other meetings that might have been in the communicating agent's schedule. More complex reasoning can be carried out with specialized data structures to represent various forms of possibilistic information, using both arc consistency reasoning and inferences based on set inclusion relations among different kinds of possibilities (Wallace 2002).

Building on these earlier ideas, the present paper introduces new techniques that allow an agent to assess the amount of privacy loss that may occur when different items of information are communicated. This allows an agent to select items to minimize privacy loss. The basic idea is to allow agents to develop views of themselves as they are viewed by other agents; for this reason we call them "mirror views". Mirror views can be used to track one's own privacy loss during the course of problem solving and to estimate additional loss due to one or another communication. With this information, agents can make informed decisions about communications in order to minimize such loss.

Using a meeting scheduling testbed described in earlier papers (Freuder, Minca, & Wallace 2001) (Wallace, Freuder, & Minca 2004), we examine the effects of such knowledge on efficiency and privacy loss when agents collaborate to solve a problem of mutual interest. As earlier, each agent has a pre-existing schedule of meetings that is known only to it. The problem is to schedule k further meetings that are suitable for all agents, while limiting the amount of information revealed during problem solving.

In this initial investigation on mirror views, we show that

*This work was supported in part by Science Foundation Ireland Grant No. 00/PI.1/C075.

agents can choose meetings to reduce either the amount of information revealed concerning their possible schedules, or places in their schedule where no meeting has been scheduled ("open slots"). In some cases, this is accomplished with a marked loss in efficiency, but in other cases the efficiency/privacy tradeoff can be managed successfully.

The remainder of the paper is organized as follows. The next section describes some situations where both privacy and efficiency can be assessed. Then we discuss the assessment of privacy loss and introduce basic measures. We continue with a section introducing the idea of "shadow CSPs" that can represent an agent's current knowledge about another agent's schedule in terms of existing possibilities, and which is used here to implement the "mirror views". The next section describes experiments based on the meeting scheduling scenario that test the effect of strategies designed to reduce privacy loss, and the last gives conclusions.

Example Scenarios

For expository purposes, we will use some problems from previous work as examples. These problems, in fact, constitute the environment in which our basic ideas concerning privacy loss were originally derived. Our expectation is that, once properly fledged, these ideas can extend their present range to include other kinds of distributed problems and multi-agent systems.

The first example is a simplified meeting scheduling problem. In this scheduling problem, each of k agents has its own calendar, consisting of appointments in different cities at different times of the week. The problem is to find one or more meetings that all agents can attend given their existing schedules and constraints on travel time. Agents communicate on a 1:1 basis; the basic protocol is for one agent to suggest a meeting time in a certain city to each of the other agents, who then tell the first agent whether the choice is acceptable or not given their existing schedules. This continues until a meeting is found that is acceptable to all agents.

To further simplify the problem, we assume a fixed set of cities where meetings can be held: London, Paris, Rome, Moscow and Tbilisi. We also restrict meeting times to be an hour in length and to start on the hour between 9 AM and 6 PM, inclusive, on any day of the week.

We can represent a problem of this type as a CSP, where variables are the 70 time-slots, and values are the cities where a meeting can occur. The basic constraints are the times (in hours) required for travel between meetings in different cities, as shown in Figure 1. In this figure, times between cities within one region (Western Europe or the former Eastern Bloc) are shown beside arcs connecting cities; the arc between the two ellipses represents constraints between any two cities in the different regions.

The second example is a multi-agent graph coloring problem, where agents must color a portion of the graph in common. In one variant of the problem, all agents have the same graph to color, so that for a graph of V nodes, K must be colored in common and $V - K$ can be colored independently by each agent. In another variant, a common subgraph forms part of each agent's graph, while the remaining part is different for each agent. In all cases, agents start with the same

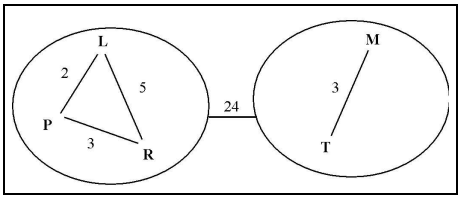

Figure 1: Time constraint graph. Cities are London, Paris, Rome, Moscow and Tbilisi.

set of colors, i.e. the domains are the same for all (common and private) variables. Here, also, the same protocol has been used as for the meeting scheduling problem, so the basic communications are either a proposed assignment (color variable v_i with color c_k) broadcast to other agents, or an acceptance or rejection sent to the proposer.

Characterizing and Measuring Privacy Loss

Basic considerations

In the first example described in the last section, private information pertains to meetings and open slots; in the second, private information pertains to existing assignments in the private portion of one's graph and to colors available for assignment to the common nodes.

In the first case, the most obvious measures of privacy loss involve the number of meetings and open slots that another agent either learns about directly or can deduce, and in the second the number of existing assignments or available colors. Therefore, a viable measure of privacy loss is in terms of information gain. This is supported by examples like the following. Suppose 100 possibilities can be discarded in each of two situations, in the first case from an original set of 400, in the second from an original case of 101. It certainly appears that the latter constitutes a greater loss of privacy.

In general, privacy loss seems to involve discriminatng values of personal attributes from other possibilities. This becomes important when private informatin involves predictability, that is, knowing something about an agent that allows one to predict its behavior in some way. In some cases the relation between an item of information and prediction may be indirect. If someone knows your name, he may be able to find your address in a phone book, and thus predict where you are likely to be. In contrast, knowing that your name contains the letter "a" is unlikely to lead to being able to predict anything, so in this case the loss of privacy would be insignificant.

It is therefore reasonable to measure privacy in terms of reductions in sets of possibilities. In this case, we can refer to an "original-relevant-set" and a "resultant-set" of possibilities. Then privacy loss can be defined as follows:

$$\text{privacyloss} = \log_2(|\text{original\_relevant\_set}|) - \log_2(|\text{resultant\_set}|)$$

and can, therefore, be measured in bits of information.

In addition, it may be possible to reduce the set of possibilities so desired information can be known with certainty. In this case, we call the resultant set the "effective-set". If

we can determine the size of the effective-set in a given situation, then we can also describe privacy loss in relation to a maximum value.

Privacy loss can also be assessed more directly, in terms of the number of actual items of information revealed, for example the number of actual meetings and open slots. This tells us how many effective-sets are known with certainty at the end of a session. The latter can be inferred directly from some communications (proposals or acceptances) and also by more elaborate inference. (E.g. if all possible meetings have been ruled out for a time slot, then it must be open.)

Although in the work described in this paper, all items of information are assumed to be equally important with respect to privacy loss, in many situations it may be important to preserve the secrecy of some items over others. Such requirements can be readily accomodated by using soft constraints (e.g. (Bistarelli *et al.* 1996)), for example by associating each item or tuple of items (in a CSP representation) with an evaluation related to its importance as well as the estimation of privacy loss. For example, the basic privacy loss measure could be multiplied by a weight that is nonlinear in the information gain, with different constants for different items of information. In this way, as information gain increases so that the set of possibilities converges to the effective-set, the cost of further loss is also increased, causing the decision maker to avoid communications that reveal more information about these items.

Agent self-assessment of privacy loss

Given a measure of privacy loss over a well-defined set of possibilities, it becomes possible for agents to estimate their own privacy loss. Such assessments can then be used to guide decision making. In this way, privacy considerations can be incorporated into an agent's own utility function. Since earlier research on privacy issues did not have an actual measure of privacy loss, such incorporation was not possible.

We will organize the procedure of estimating one's own privacy loss around the concept of agent views, using the representation for such views found in the work stemming from the paper of (Freuder, Minca, & Wallace 2001). Specifically, these (additional) views represent information that can be deduced about what another agent knows about oneself, given general assumptions about the information in question and the communications, in particular, communications that the agent is considering. To distinguish them from ordinary views we call them "mirror views". (In the present work we assume that agents do not exchange secrets learned about a third party.)

Structures for Representing and Making Inferences about Other Agents' Information

The measure of privacy loss described above depends on each agent knowing a set of possible values, under the assumption that this set contains the actual value of an element in another agent's problem as well as all other values that could be valid. This set will be part of that agent's view (of the other agent). A view can be updated after each communication from another agent, which may involve reducing some sets of possibilities, with resulting privacy loss for the communicating agent.

For CSPs in general, the kinds of possible values can be organized into "shadow CSPs" that represent sets of possible domain values and constraint tuples. For domain values (unary constraints) there are three kinds of shadow values: possible existing assignments, possible conflicting assignments, and possible future assignments. In the meeting scheduling example, the first kind refers to meetings in another agent's schedule, the second to possible meetings responsible for a rejection, the third to city-in-time-slot elements that might be available for a common meeting. A simple example of a shadow system for representing unary constraints is shown in Figure 2.

Systems of shadow CSPs represent two basic kinds of information, related to existing and future variable assignments. In addition, there are other requirements that we can derive from consideration of the basic problem. The most important is that the set inclusion relation should hold between corresponding domains of an actual CSP (that contains actual values revealed by the other agent) and any related shadow CSP, where "corresponding" means being associated with the same variable. In addition, the same relation must hold between corresponding domains of shadow CSPs related to the same actual CSP, here between domains of possible-conflict values, and corresponding domains of possible-existing-assignment values. Generalizing from this, we require that:

1. The entire set of shadow CSPs plus the representation of the actual CSP has a supremum, which we call the "universal" (shadow) CSP and an infimum which may be the null set (cf. Figure 2).
2. For this entire set, the set inclusion relations are reflexive and transitive.
3. There is an additional property of good structure in that if a domain of shadow CSP X is a subset of the corresponding domain of shadow CSP Y, then another domain of X will be a subset of the corresponding domain of Y.

The first two requirements insure that there is a partial order on each set of corresponding domains under the set inclusion relation. Requirement three gives a transitive, reflexive relation for complete shadow CSPs.

The proof that the system remains well-structured shows that, with a given set of communications and rules of inference, as, for example, described for the scenarios in the second section, the set inclusion relation remains achievable at every step in search (Wallace 2002). In particular, this means that we cannot deduce an actual value that has been ruled out as a possible value in any corresponding domain of a superordinate shadow CSP. In other respects, the system is sound because inferences rely on standard logic, plus a closed world assumption.

In the simplest case, at the beginning of a negotiation session, when an agent knows nothing about the other agents' problems, all domains of possible-existing-assignments and possible-future-assignments, as well as

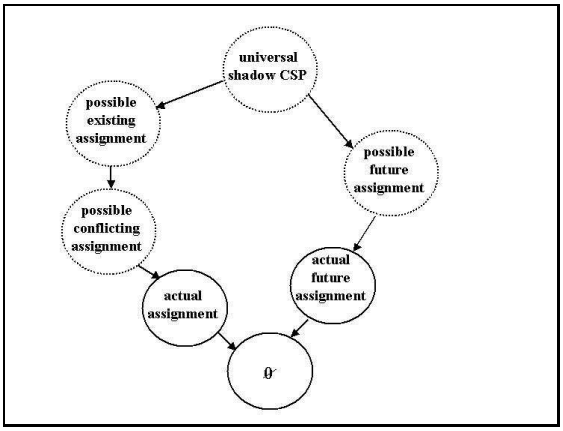

Figure 2: Structure of a basic shadow CSP system for unary constraints. Shadow CSPs are shown as dashed circles. Arrows represent the realization (set inclusion) relation holding between the domains of super- and subordinate CSPs.

the "universal" shadow CSP, have all possible values. In the meeting scheduling situation, for example, each of the 70 domains of the possible-existing- and possible-future-assignment CSPs will have five values, corresponding to five possible cities. This represents the fact that any value may be a value in the other agent's exisiting schedule *or* it may be a value that is available for future assignment, in this case as an additional meeting. The possible-conflicts shadow and the actual CSPs have empty domains, since no conflicts have been deduced, nor any actual existing or future values.

As search proceeds, values are deleted from the domains of the first two CSPs and added to the last two. For example, if an agent proposes a meeting in London at 2 PM on Tuesday, other agents can infer that this agent does not have any meetings in this time-slot; nor can it have meetings in another city at 1 or 3 PM, since the time constraints would prevent it from meeting in London at 2; the latter agents can, therefore, delete these (and other) shadow values from their views of this agent. Similar deductions can be made by the proposing agent if another agent accepts this proposal. On the other hand, if the other agent rejects this proposal, the proposing agent can deduce that the rejection could have been due to an existing meeting at 2 PM in the other agent's schedule, or to a meeting in Paris within two hours of this time, or to a meeting in London within three hours, etc., and can therefore add these shadow values to the possible-conflicts CSP.

In this system, privacy loss is assessed in terms of reductions in the number of possible values for a given domain; since in this sense domains are independent, separate assessments must be made for each domain. In addition, privacy regarding existing assignments must be assessed independently of privacy regarding values available for future assignment. For existing assignments, the original-relevant-set consists of all possible values plus the possibility that there is no assignment at all (if this is possible). For the meeting scheduling example this would be the possibility that the agent has no meeting in a slot; in this case, there are therefore six possibilities for the original-relevant-set *for each slot in each agent view*. And in this case the effective-set is obviously always one. For possible future assignments, the situation is not as straightforward. One approach, which has been used so far in this work is to consider each possible subset of a domain as an information-element. The original-relevant-set is then the power-set of the set of possibilities. For the meeting scheduling example the size of this set is 2^5.

In our new approach, each agent also maintains the shadow CSPs assumed to be maintained by other agents about himself. As mentioned before, these shadow CSPs form a "mirror view" and are used to select the next action of the agent so that progress is achieved with minimal privacy loss for himself.

An Experimental Testbed

System and methods

In order to make some assessment of mirror views in practice, a system used earlier for simulating the multi-agent meeting scheduling problem described above was elaborated with mirror views. This system is written in Java, and allows the user to pose the basic problem with different numbers of agents and initial meetings, as well as varying the kinds of information that is gathered, as well as protocols and proposal strategies. The version used here also allows the user to vary the number of common meetings to be found, as well as specifying whether the problem has a solution or not.

In the present elaboration of this system, mirror views were simulated simply by allowing agents to examine other agent's views of themselves before making proposals. This was acceptable because in this situation, general information about constraints and possible variables is the same for all agents, who would therefore make the same deductions about a given communication. Agents could also calculate, for each potential proposal, the number of additional possible existing assignments that could be ruled out and the number of open slots in their schedule that would be revealed. The basic comparisons were between criteria for proposal selection:

- agents simply propose meetings, using possibilistic knowledge inferred about the other agents' schedules, as described in the previous section (cf. (Wallace, Freuder, & Minca 2004))
- each agent proposes a meeting that minimizes the number of possible existing assignments that could be ruled out by an agent receiving this proposal
- each agent proposes a meeting that minimizes the number of open slots that would be discovered on the basis of inferences about this proposal

Experimental tests were run with various numbers of agents, initial meetings in the agents' schedules, number of new assignments required and differences in problem solubility. Each experiment (in which these factors were held constant) consisted of 100 runs. For these experiments, agents followed a "round-robin" procedure in which agents

take turns making proposals following a fixed order (cf. (Freuder, Minca, & Wallace 2001)).

To give an overview of the experimental procedure, an individual test run begins with random generation of schedules followed by a series of proposals which continue until one is found that is acceptable to all agents. Proposals are selected by choosing a time slot and city at random (to ensure unbiased sampling). This is repeated until the candidate meeting fits the proposer's schedule. After that, further tests are made. These include checking that this proposal has not been made before, and then checking against the agent's knowledge, for example, knowledge of other agents' actual meetings, current possible-has-meeting's, etc. The first viable proposal (one meeting all tests specified for the experiment) is communicated to each of the other agents, and the latter reply with an acceptance or rejection to the proposer alone.

The efficiency measure used in these tests was number of proposals per run, averaged over all 100 runs in an experiment. The measures of privacy loss were the entropic measure applied to reductions in possible existing assignments plus the actual number of open slots identified.

Results

In the first experiments to be described, agents searched for one common meeting to add to their schedules. In one set of experiments there were three agents whose initial schedules had 15 meetings. In a second set there were seven agents, each with 10 initial meetings. Some results from the first set, showing efficiency and one direct (non-entropic) indication of privacy loss, are shown in Figure 3. Results for privacy loss related to possible existing assignments are shown in Figure 4.

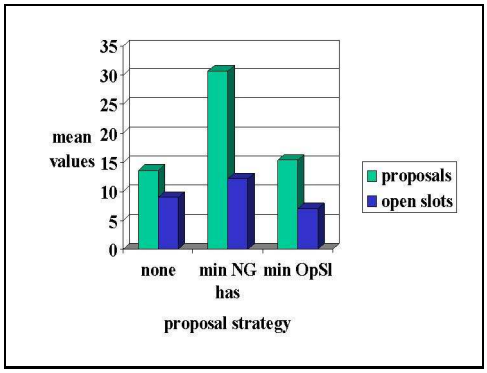

Figure 3: Mean proposals (per run) and open slots revealed (per agent view) in experiments with three agents having 15 pre-existing meetings in their schedules. "Proposal strategies" are based on the criteria described in the text. "min NG has" refers to the strategy that minimizes the number of possible existing assignments that could be ruled out by other agents; "min OpSl" refers to the strategy of minimizing the number of open slots that could be deduced on the basis of a proposal.

These results show it is possible for agents to reduce privacy loss by specific proposal strategies based on their mirror views, although in these experiments the effects are not large. However, for the strategy that involves tallying prospective no-good possible existing assignments there is a considerable loss in efficiency, both in terms of the standard measure (proposals per run) and runtime. As a result, more open slots were discovered than in the reference condition (cf. Figure 3). (In fact, it is somewhat surprising that this proposal strategy was still effective with respect to the targeted quantity, given the marked increase in number of proposals communicated before finding a solution.) A similar pattern of results was found in the second set of experiments.

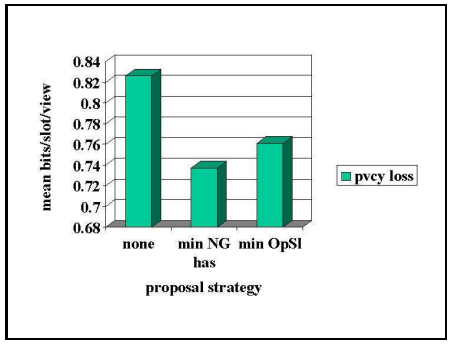

Figure 4: Privacy loss with respect to existing assignments in experiments with three agents having 15 pre-existing meetings in their schedules.

A third series of experiments was based on a harder task that required backtrack search. Here, there were five agents each with eight meetings in their initial schedules. The task was to find three additional meetings that all agents could attend; in fact, the problems in this series were constructed to be insoluble, with at least one viable instance of two additional common meetings. Agents used a form of backtrack search based on successive assignments (variable/value pairs) rather than variables; each agent also used forward checking to prune its own search space. In this case, the validity of some of the knowledge gathered, specifically previous proposals and possible future assignments ruled out, depended on the state of search, so that retracting a proposal was accompanied by discarding the information that depended upon it.

Because the task in these experiments required a full tree search, selecting proposals to minimize privacy loss did not reduce efficiency as in the first experiments. (About 290 proposals were required in all conditions.) Again, there was a modest but noticeable reduction in privacy loss that corresponded to the quantity targeted by the proposal strategy.

Further results: Dynamic versus static (presearch) assessment

The proposal strategies tested in the previous experiments evaluate privacy loss dynamically, i.e. in terms of expected *further* loss based on the current view. Hence, they are responsive to the actual, effective loss of privacy at any point in search. This approach can be compared with one based

on more static assessments, in particular, those made prior to actual search. The latter will not be able to assess actual loss at a particular stage of search, but it is not clear how much difference this will make. In addition, dynamic assessment is much more expensive, so that in some situations, where computational opportunities or resources are limited, it may be more realistic to use static assessments.

An initial evaluation of this difference was carried out, using a variation on the strategy that minimizes the number of possible existing assignments. In this case, assessments were carried out for each viable meeting proposals prior to search; at each step of search, candidate proposals were chosen that had the lowest tallies.

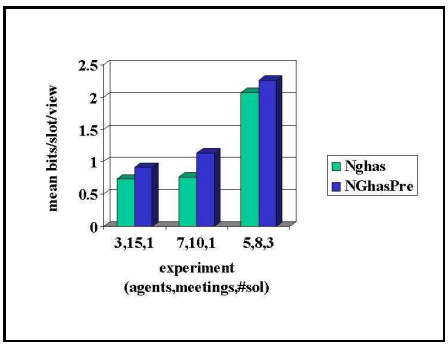

Figure 5: Privacy loss with respect to existing assignments in the three kinds of experiments. Dynamic versus static assessment of privacy loss.

The main results of these experiments are, (i) the static strategy impairs performance uniformly in relation to its dynamic counterpart (and, therefore, in relation to the baseline, no-strategy condition), (ii) partly because of (i), privacy loss with respect to possible existing assignments is greater than with the dynamic strategy (Figure 5), as well as the baseline condition.

The results suggest that dynamic assessment is required to accurately estimate and control privacy loss. In this connection, it should be noted that such assessment will usually require some kind of view that is constantly updated, such as the shadow system used in this work. The only exceptions would involve stringent independence conditions, in which possibilities cannot be discarded unless a particular item is communicated.

Conclusions and Future Directions

By quantifying privacy loss, we enable agents to estimate their own potential privacy loss and to act in ways that can modify these quantities. This can be done if agents can reason about the effects of communications on the basis of valid general knowledge about sets of possible values for elements of the problem. In other words, agents must have articulated views of other agents and some means of updating these views based on sound inferences from communications. The present work shows that with these capacities agents can modify their communications in order to reduce privacy loss. However, for these problems improvements were modest and in some cases there was an attendant loss of efficiency that, of course, works against the goal of maintaining privacy. There is some indication in this work that trying to minimize actual information lost is more effective than trying to minimize information loss according to entropic measures.

We have also begun to evaluate the difference between dynamic and static strategies for assessment. Our initial results suggest that dynamic assessment will be necessary for effective modulation of privacy loss in most situations. Obviously, however, the whole issue of static versus dynamic assessment of privacy loss is an important area for futher research.

As indicated in the section on privacy measures, it should be possible to refine assessments of privacy loss according to the 'value of secrecy'. Based on these initial studies, it seems reasonable to expect that this should lead to successful strategies along the lines of those developed here, especially when there are only a few elements of high value.

References

Bella, G., and Bistarelli, S. 2001. Soft constraints for security protocol analysis: Confidentiality. In *Proc., Third International Symposium on Practical Aspects of Declarative Languages LNCS 1990*. Berlin: Springer.

Bistarelli, S.; Fargier, H.; Montanari, U.; Rossi, F.; Schiex, T.; and Verfaillie, G. 1996. Semiring-based csps and valued csps: Basic properties and comparison. In Jampel, M.; Freuder, E.; and Maher, M., eds., *Over-Constrained Systems*, volume 1106. Berlin: Springer. 111–150.

Freuder, E. C.; Minca, M.; and Wallace, R. J. 2001. Privacy/efficiency tradeoffs in distributed meeting scheduling by constraint-based agents. In *IJCAI-01 Workshop on Distributed Constraint Reasoning*.

Silaghi, M. C., and Faltings, B. 2002. A comparison of discsp algorithms with respect to privacy. In Yokoo, M., ed., *Third International Workshop on Distributed Constraint Reasoning, DCR-02*, 147–155.

Silaghi, M. C.; Sam-Haroud, D.; and Faltings, B. 2000. Asynchronous search with private constraints. In *Proceedings of AA2000*, 177–178.

Wallace, R. J.; Freuder, E. C.; and Minca, M. 2004. Possibilistic reasoning and privacy/efficiency tradeoffs in multiagent systems. In *Mexican International Conference on Artificial Intelligence - MICAI'04*.

Wallace, R. J. 2002. Representing possibilities in relation to constraints and agents. In *Logics in Artificial Intelligence, JELIA 2002, LNCS 2424*, 407–418. Berlin: Springer.

Yokoo, M.; Suzuki, K.; and Hirayama, K. 2002. Secure distributed constraint satisfaction: Reaching agreement without revealing private information. In Hentenryck, P. V., ed., *Principles and Practice of Constraint Programming - CP2002, LNCS 2470*. Berlin: Springer. 387–401.

Yokoo, M. 1998. *Distributed Constraint Satisfaction. Foundations of Cooperation in Multi-Agent Systems*. Berlin: Springer.