Chapter 1

# ANALYSIS OF NEGOTIATION PROTOCOLS BY DISTRIBUTED SEARCH

Guandong Wang 1, Weixiong Zhang 1, Roger Mailler 2, Victor Lesser 2
1Department of Computer Science and Engineering
Washington University in St. Louis
St. Louis, MO 63130, USA
{gw2,zhang}@cse.wustl.edu
2Department of Computer Science
University of Massachusetts
Amherst, MA 01003, USA
{mailler,lesser}@cs.umass.edu

**Abstract**

Negotiation is one of the main mechanisms for coordination and cooperation in multiagent systems. However, most negotiation protocols are complex and their features are difficult to characterize. In this paper, we propose a general experimental approach to analyzing negotiation strategies using distributed search. In this approach we first formulate the problems that negotiation protocols intend to solve as distributed constraint satisfaction/optimization problems, and then capture the negotiation protocols as distributed search algorithms. By analyzing the derived search algorithms, we can characterize many important properties of the negotiation protocols. In this paper, we are particularly interested in the properties of a newly developed negotiation protocol, which is motivated by distributed sensor network applications, including its completeness, complexity, convergence rate, and scalability. Although the idea of viewing negotiation as distributed search is not completely new, in this research we not only view negotiation as distributed search, but directly apply a search algorithm to reveal the essential features of a negotiation protocol and analyze its performance.

## 1. Introduction and Overview

Negotiation is an important mechanism for coordination and collaboration in multi-agent systems. It is particularly effective for systems consisting of

self-interested agents, each of which may have different objectives to achieve and different restrictions to abide to. It is perhaps also prevalent in applications where privacy of individual agents needs to be protected. Due to its importance, negotiation has been studied for quite some time, and many different negotiation strategies and protocols have been proposed and developed [Cammarata et al., 1983; Adler et al., 1989; Conry et al., 1988; Durfee and Lesser, 1989; Durfee and Montgomery, 1990; Lander and Lesser, 1992]. Cooperative negotiation has been studied to solve difficult distributed problems such as distributed conflict resolution [Adler et al., 1989; Lander and Lesser, 1992], distributed task allocation [Cammarata et al., 1983; Durfee and Montgomery, 1990], and distributed resource allocation [Conry et al., 1988]. In all these domains, multiple agents share some common resources, e.g., communication channels and CPU times, and can mutually benefit from one another by cooperatively scheduling the resources. For these problems, agents only have the information about their local tasks and resources. They gradually become aware of the global information through negotiation and solve the global problem by individually solving sub-problems and integrating the solutions to sub-problems into a globally consistent solution [Lesser, 1990]. For some applications, agents may also need to assess and refine the global solutions into global optimal solutions to make the best use of common resources.

However, global optimal solutions are very hard to achieve through cooperative negotiation in a distributed setting, due to its computational and communicational complexity. Furthermore, most existing negotiation protocols are complex and their features are difficult to characterize. To our knowledge, we have not seen a published work that analyzes a negotiation protocol in depth to understand important issues of a negotiation method, such as its completeness, complexity and scalability.

In this paper, we propose an experimental approach to analyzing negotiation methods. Our strategy consists of two steps. In the first step, we formulate the distributed problems solved by a negotiation method by distributed constraint satisfaction/optimization problems, and capture the negotiation protocol as a distributed constraint search. In the second step, we study the properties of the negotiation protocol by analyzing the derived search algorithm. Note that to a larger extent, the idea of viewing negotiation as distributed search is not completely new. Indeed, it has been suggested that distributed AI can be viewed as distributed search [Lesser, 1990]. Nevertheless, we not only view negotiation as distributed search in this research, but take this view one step further and directly apply a search algorithm to capture the essential features of a negotiation protocol and analyze its properties and performance.

In this research, we specifically focus on a resource allocation problem underlying a mobile object tracking problem using distributed sensors and a recently developed negotiation protocol, Scalable Protocol for Anytime Multi-

level negotiation (SPAM) [Mailler et al., 2001; **?**], for this problem. The SPAM negotiation protocol has been successfully used to manage a set of distributed sensors to solve the problem of tracking multiple targets. Specifically, a set of networked sensors cooperatively detect and localize a set of moving targets by taking local measurements and exchanging information. Since a sensor is only capable of measuring the distance from a target to the sensor as well as the speed of the target, in order to estimate the location of the target, multiple sensors have to detect at the same time and combine their measurements. Having measurements from more sensors at the same time or getting the measurements more frequently will produce a higher quality tracking.

The primary objective of this multiple target tracking problem is to allocate the sensors to the targets so as to maximize the tracking quality. The SPAM protocol is designed for solving this problem with distributed negotiating agents. Each sensor is associated and managed by an agent. Whenever a new target is detected, an agent may also play the role of managing the task of keeping track of the target, i.e., determining which sensors to use for tracking and making schedules for the available sensors. Such an agent is also called a track manager in the protocol. When more than one target enters the system, conflicts on the demand of sensors exist, and track managers may need to negotiate with one another to resolve the conflicts on their local schedules and/or cooperatively produce a global schedule to attempt to maximize the overall tracking quality. The key idea of the SPAM protocol is to select one of the conflicting track managers as a mediator to resolve the conflicts and generate partial solutions for the conflicting managers involved. This conflict resolution process may propagate to multiple negotiation threads. An experimental study showed that the SPAM protocol works very well for this real-time sensor tracking problem [Mailler et al., 2001; **?**]. However, the SPAM protocol is too complex to be amenable to a thorough theoretical analysis, and almost all of its important features, such as completeness and convergency, have not been analyzed.

In addition to developing a general experimental approach of analyzing negotiation strategies, another objective of this research is to elucidate the properties of SPAM, including its completeness, convergence, complexity and robustness. To this end, we view the resource (sensor) allocation problem for target tracking as a distributed constraint satisfaction/optimization problem and transform the SPAM protocol into distributed search algorithms. We then characterize many important features of the SPAM protocol by analyzing the derived search algorithms.

The paper is organized as follows. We briefly describe the sensor tracking problem and the SPAM negotiation protocol in section 2. We formulate the cooperative negotiation problem as a multi-agent constraint problem in section 3. In section 4, we describe our strategy to characterizing the negotiation as

distributed constraint search, and further generalizing and extending the SPAM protocol as two distributed constraint search algorithms. In section 5 we apply the derived search algorithms to experimentally analyze some essential features of the negotiation protocol. Finally we conclude and summarize our results in section 6.

## 2. Target Tracking and the SPAM Protocol

The application we consider is the tracking of multiple moving targets using a network of loosely coupled sensors. This application has also motivated the SPAM negotiation protocol. We will briefly describe the problem in section 2.1 and the protocol in section 2.2.

### 2.1 Tracking multiple targets

In this problem, a set of Doppler sensors are scattered with varying orientations throughout a tracking area. Each sensor is able to detect an object within a fixed radius. However, the overall detection area of a sensor is divided into three equal sectors, and the sensor can only operate in one sector at any given time. The Doppler sensor is only capable of detecting the distance between a target and the sensor by measuring signal amplitude as well as the speed of the target based on signal frequency. In order to track a target, at least three sensors are required to sense at the same time and triangulate the location of the target. Having more sensors tracking at the same time or taking measurements more frequently will produce a higher quality tracking.

The problem can thus be described as follows. Given $n$ sensors and $m$ targets, develop a schedule for all the sensors to track the maximal number of targets as accurately and frequently as possible over a period of time. The complication of the problem stems in at least two factors. First, multiple targets can exist and conflicts on allocating a sensor to multiple targets may be unavoidable, especially when the number of sensors is not sufficient. Second, an agent has partial knowledge of the overall problem but needs to act using local information and information from its neighboring agents.

From utility theoretical point of view, the problem can be considered as a problem of maximizing a utility function of the targets being tracked over a period of time. If the number of sensors allocated to a particular target is less than three, no reliable estimation to the location of the target can be obtained. On the other hand, even though having more than three sensors allocated to a target will give rise to better tracking result, the tracking quality is not linear of the number of sensors used and the quality improvement will be negligibly small after a specific number of sensors. For example, the utility function of tracking a set of $m$ targets can be defined as the total sum of the utility of tracking individual targets, each of which can be defined as $s \times \frac{1}{2}\sqrt{3l-5}$ if $l$ sensors

are allocated to a target during a period time $s$. This utility function could be more complicated if it takes into account the location and the orientation of each individual sensor with respect to the target being tracked.

## 2.2 The SPAM protocol

The SPAM protocol is designed with many considerations on real-time performance and dynamic issues. The protocol is divided into three stages or abstraction levels to cope with three levels of real-time constraints. In the lowest or sensor level, the system is required to respond immediately so that the problem is solved without any information on each local sensor. At the end of this stage, many conflicts over the allocation of the sensors may exist. At the second abstraction level, the problem is solved with knowing the schedule on each local sensor. At the end of this stage, all the local conflicts can be solved, but some non-local conflicts may be introduced due to missing the global information. At the third or the resource level, the conflicting agents negotiate over their local schedules through a mediator who generates partial solutions for all these agents and tries to solve non-local conflicts. As long as conflicts exist in the system, some agent will become a mediator and propagate the negotiation. A brief description of these three stages are given below.

**Protocol 1** SPAM Protocol Stage 0

```
Evaluate and order the usable resources
Decides an initial objective level
if (have more time) then
   Go to Stage 1
else
   Choose a solution maximizing the local utility
   Bind the solution and exit
end if
```

Both simulation and hardware experiments show that the SPAM protocol works very well for the real-time moving target tracking problem. However, since the protocol itself is too complex to analyze, some important features of the protocol are still unclear. We are especially interested in the following properties of the protocol: completeness, time complexity, rate of convergence, and scalability. In order to capture these properties, we will transform the protocol into succinct distributed constraint search algorithms. By analyzing these search algorithms, we can characterize many important features of the protocol. Before we launch onto search algorithms, we first describe the constraint formulation of the tracking problem, the topic of the next section.

**Protocol 2** SPAM Protocol Stage 1

```
Collect the local information from all usable resources
Generate the set of local solutions
if (have solution without conflicts) then
   Choose the solution with the maximum local utility
   Bind the solution
else
   Choose a solution which both minimizes the conflicts and maximizes the
   local utility
   Bind the solution and exit
   if (have more time) then
      Go to Stage 2
   end if
end if
```

**Protocol 3** SPAM Protocol Stage 2

```
Mediator detects the oscillation by checking the history of negotiation
if (have oscillation) then
   Lower the objective level
end if
Request meta-level information from the conflicting agents
while (have no solution) do
   Generate partial solutions for all these agents
   if (all agents are at their lowest objective level) then
      Choose a solution at the lowest objective level with min-conflicts
      Exit
   else
      Lower the objective level of one agent
   end if
end while
Mediator sends all the partial solutions to the other agents
The other agents evaluate and rank the solutions
Mediator chooses a consistent solution according to the feedback
All the agents bind their solutions
Some agent propagates the negotiation if having conflicts
```

## 3. Constraint Problems in Cooperative Negotiation

A distributed problem that can be solved by cooperative negotiation normally involves a set of agents, each of which has some tasks to be scheduled

using a set of shared resources. Given a set of $n$ agents over $m$ resources, a cooperative negotiation problem can be formally represented as $(A, R)$, where $A = \{a_1, a_2, \cdots, a_n\}$ is the set of $n$ agents and $R = \{s_1, s_2, \cdots, s_m\}$ the set of $m$ resources. An agent, $a_i$, is represented by a tuple:

$$a_i = (R_i, C_i, w_i),$$

where $R_i$, $R_i \subseteq R$, is a set of resources that can be used by agent $a_i$, $C_i$ represents a task to be scheduled by $a_i$, and $w_i$ is the weight assigned to task $C_i$.

The task of $C_i$ may also consists of a set of sub-tasks $T_i = \{T_1^{(i)}, T_2^{(i)}, \cdots, T_{t_i}^{(i)}\}$, and $(T_i, R_i)$ constitutes a local sub-problem that can be internally solved by agent $a_i$. In other words, $(T_i, R_i)$ is local to $a_i$ and is unknown to the other agents. The overall task $C_i$ requires a certain number of resources, and this number, represented by $o_i$, is called the objective level. Given the set of usable resources $R_i$ and the objective level $o_i$ of the agent $a_i$, there are $l_i = \binom{|R_i|}{o_i}$ alternative solutions. Then task $C_i$ can be represented by

$$C_i = \bigvee_{j=1}^{l_i} S_{ij}$$

where $S_{ij}$ represents one possible solution for the agent $a_i$ to the task $C_i$. And $S_i = \{S_{i1}, S_{i2}, \cdots, S_{il_i}\}$ constitutes the solution space for the sub-problem $C_i$ on agent $a_i$. Given the objective level $o_i$, $S_{ij}$ can be further represented by

$$S_{ij} = \bigwedge_{m=1}^{o_i} (r_m^{ij} = i)$$

where $r_m^{ij} \in R_i$. Here $r_m^{ij} = i$ means that the resource $r_m^{ij}$ is allocated to task $C_i$. Moreover, in order to make $C_i$ true, at lease one solution $sol_i = \bigwedge_{m=1}^{o_i} (r_m^i = i)$, $sol_i \in S_i$, must be true; therefore, all the resource $r_m^i$ should be allocated to $C_i$. We call $sol_i$ the local sub-solution for the sub-problem $C_i$ on agent $a_i$.

Here, an agent is a track manager, and a resource corresponds to a sensor in the SPAM protocol. With the restriction that a resource can only be used by one task at any time, each resource should have a consistent allocation or assignment in the local sub-solution of each task $C_i$, which constitutes a constraint among tasks. Note that each task $C_i$ is distributed among a set of agents. Therefore, we can formulate the cooperative negotiation problem as the following distributed constraint satisfaction problem: *Given a constraint problem $(A, R)$, is there an assignment of resources $R^+ \subseteq R$ such that $\bigwedge_{i=1}^{n} C_i$ is satisfied?*

As discussed in Section 2.1, an agent $a_i$ can have a utility function $U_i : S_i \mapsto \mathbb{R}$ that can be used to discriminate alternative local solutions. Furthermore, the overall goal of the negotiation problem is to find a globally consistent solution with maximal global utility among all the solutions. Therefore, since $C_i$ has a weight $w_i$, the global utility can be simply defined as $Util_{total} = \sum_{i=1}^{n} w_i \cdot Util_i$, where $Util_i = U_i(sol_i)$ is the utility of the agent $a_i$ with the local sub-solution $sol_i$. With the extension, the agent $a_i$ is represented by $(R_i, C_i, w_i, U_i)$, and we can formulate the cooperative negotiation problem as the following constraint optimization problem: *Given a constraint problem* $(A, R)$*, what is the assignment of resources to agents such that the constraint* $\bigwedge_{i=1}^{n} C_i$ *is true and the utility function* $U_{total}$ *is maximized?*

# 4. Negotiation Protocol as Search Algorithms

The problems solved by cooperative negotiation can be formulated as distributed satisfaction or optimization problems. A negotiation protocol can be viewed as a distributed search process. In this section, we propose to characterize the negotiation protocol by a distributed search algorithm.

## 4.1 Negotiation as distributed search

Negotiation and search are fundamentally different. The former is naturally a multi-agent problem solving method in which information may not be shared among agents. Thus, negotiation is generally harder than search, since a global view of a problem may never be constructed. In such a situation, it may become difficult for an agent to even determine whether the current variable assignments are in a better state than the previous ones. Moreover, a negotiation process may be trapped in an infinite negotiation loop in which all the agents revisit some previously encountered global solutions endlessly. Furthermore, negotiation may have more restrictions than search. For example, time or some other parameters can become a factor when evaluating the results of the negotiation. Thus, the negotiation is normally more complicated in the sense of searching for a solution.

Although negotiation and search are different approaches to problem solving, they both search for assignments to variables of a problem which constitute solutions. Search can also be an ingredient of negotiation. In a negotiation problem, a task of an agent can be viewed as a variable. It can be assigned any values (sub-solutions) in its solution space. The sub-solutions of different variables may conflict with one another due to the inter-agent constraints. When an agent assigns a value to its task, it also needs to make it consistent with the assignments of other agents which is similar to what a normal constraint search

algorithm does. An agent's assignment, when communicated to other agents as a proposal in the negotiation, can be rejected by the other agents. Other agents may provide counter-proposals. Thus a negotiation is just a search process, in which the agents try to assign values or revise values to their tasks to satisfy all the inter-agent constraints.

Generally, a negotiation may have two primary goals, to search for consistent solutions and to search for a consistent solution of a maximal utility. The first goal focuses on the conflict resolution, which is a problem solved routinely by constraint satisfaction search algorithms. In this regard, negotiation can thus be viewed as a constraint satisfaction method. The second goal of finding a consistent solution of a maximum utility is simply an optimization problem, a harder search problem. Note that a task of an agent can be a complicated subproblem, and the solution space of the subproblem itself may be large. All agents need to search, cooperatively, to find the best possible global solutions.

In short, negotiation can be viewed as a mechanism for solving distributed constraint satisfaction and constraint optimization problem. Taking this view, we propose to use search algorithms as tools for analyzing negotiation protocols.

## 4.2 SPAM protocol as search algorithms

The original SPAM protocol [Mailler et al., 2001; **?**] has many features to handle real-time and dynamics issues. Although these features are very necessary and important to deal with real-world applications, they are hardly amenable to a thorough analysis. It seems to be very difficult to design an abstract model of such a complicated protocol for a theoretical analysis. Therefore, it is difficult to understand, through an analytical approach, some primary properties of the protocol, such as completeness, rate of convergence, complexity and scalability.

The difficulty for a theoretical analysis suggests that an experimental analysis is in demand. Here, we propose to use search algorithms to capture a negotiation protocol so as to characterize the important features of the protocol through analyzing the search algorithms. Once a negotiation protocol is transformed to search algorithms, whenever a theoretical analysis is possible for the search algorithms, such an analysis can also be translated back to the original negotiation protocol.

Another difficulty for analyzing a distributed negotiation protocol comes from the distributed nature of the applications to which the protocol is applied. First, it is usually difficult to set up distributed experiments with a large number of agents and resources using a sufficient number of hardware, while still being able to collect enough accurate experimental data for an evaluation.

Second, most experiments in a distributed environment are not repeatable. A distributed negotiation protocol is in essence nondeterministic. There are indeed many factors, for instance the synchronization among agents, that can change experimental results from one run to another.

Therefore, caution must be taken in the experimental analysis. In this section, we will first transform the SPAM protocol to a sequential search algorithm, called sequential SPAM. Here, sequential search does not necessarily mean centralized search. The search process in the sequential SPAM can still be distributed among different agents, but in each step only one agent, chosen arbitrarily, is allowed to change its local values. One run of such a sequential algorithm corresponds to one possible execution of the original negotiation protocol. Introducing sequential execution is expected to have little impact on the effects of the protocol while making the analysis easier. The sequential search algorithm is designed to represent the features of the original protocol as close as possible, so that the results from the search algorithm is expected to shed some light on the original protocol.

We will then modify the SPAM protocol to construct a synchronous search algorithm, called synchronous SPAM. In this version, the agents negotiate in a more tightly cooperative manner. The agents are dynamically ordered during the negotiation. The agent that revises its local assignment earlier has a higher priority. This modification is introduced to make the protocol complete, since the original protocol cannot guarantee the completeness of a negotiation process, as we will see in the next section. Note that completeness is not the same as optimization. If a constraint problem is not satisfiable, the synchronous SPAM algorithm is expected to give an answer of 'NO'. Using synchronous SPAM, we are able to evaluate the completeness of the original protocol. Moreover, we also expect to understand how to improve the original protocol by analyzing the synchronous search algorithm.

**4.2.1 Sequential SPAM.** We now characterize the SPAM protocol as a sequential search algorithm in which only one agent is allowed to change its local values. The protocol is viewed as the following search algorithm, shown in Figure 1.1.

Similar to the SPAM protocol, this algorithm is divided into three stages. In stage 0, each agent chooses a solution for its local problem without knowing any information about the current schedule of each resource. In stage 1, each agent collects the information of all the resources that can be used. Then one of these agents with conflicts with other agents, chosen randomly, tries to resolve the conflicts locally by searching for a solution in its local solution space. If there exists a local solution that does not conflict with all the other agents, the agent will change to such a non-conflict solution with the best utility.

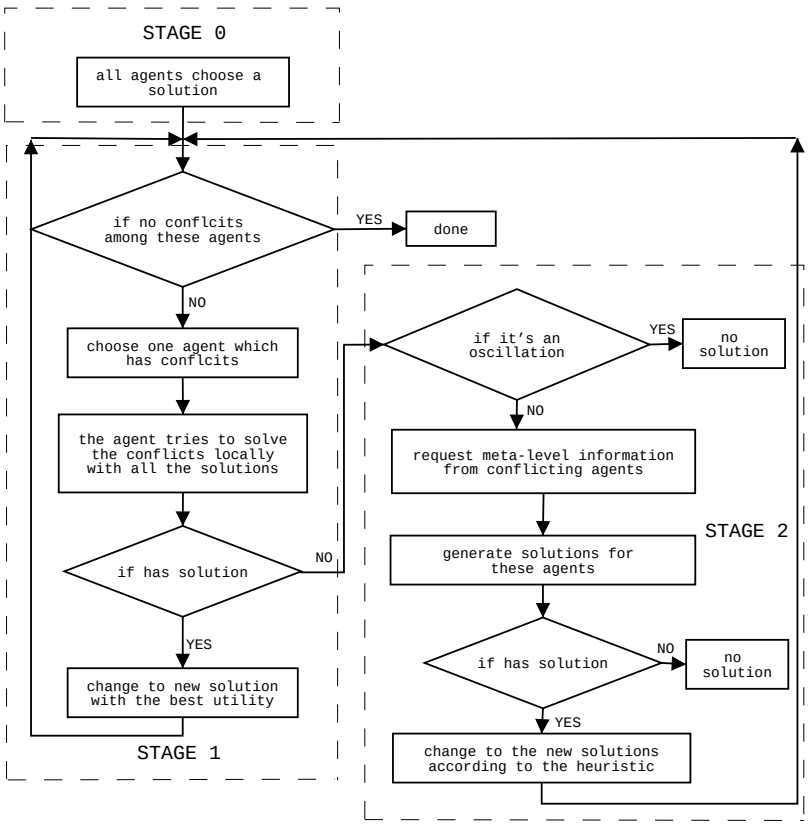



*Figure 1.1.* Sequential SPAM

If there exists no non-conflict solution, the agent will choose a solution which minimizes the conflicts with the other agents and moves on to stage 2. In other words, this agent becomes a mediator to resolve the conflicts in which it is involved. In stage 2, the agent requests meta-level information from the conflicting agents. It then tries to generate solutions for these agents. If there exist non-conflict solutions, these agents will change to non-conflict solutions based on the heuristic that the most constrained agent chooses a solution first.

If there is still a conflict among agents, a neighboring conflicting agent will attempt to resolve the conflict by propagating the negotiation. To avoid an infinite negotiation loop, each agent records the history of its previous negotiations. If an oscillation is detected, the algorithm terminates or the agent's objective level is reduced (meaning that more resources to be used) to lower the constrainedness of the conflicts. However, as we will see later, this termination condition will make the algorithm lose possible solutions or solutions

with higher objective levels. Therefore, the completeness of the algorithm or the protocol can not be guaranteed.

**4.2.2 Synchronous SPAM.** The synchronous SPAM search algorithm is not exactly the same as the original SPAM protocol. It simulates the original protocol as close as possible and ensures completeness. The algorithm is shown in Figure 1.2.

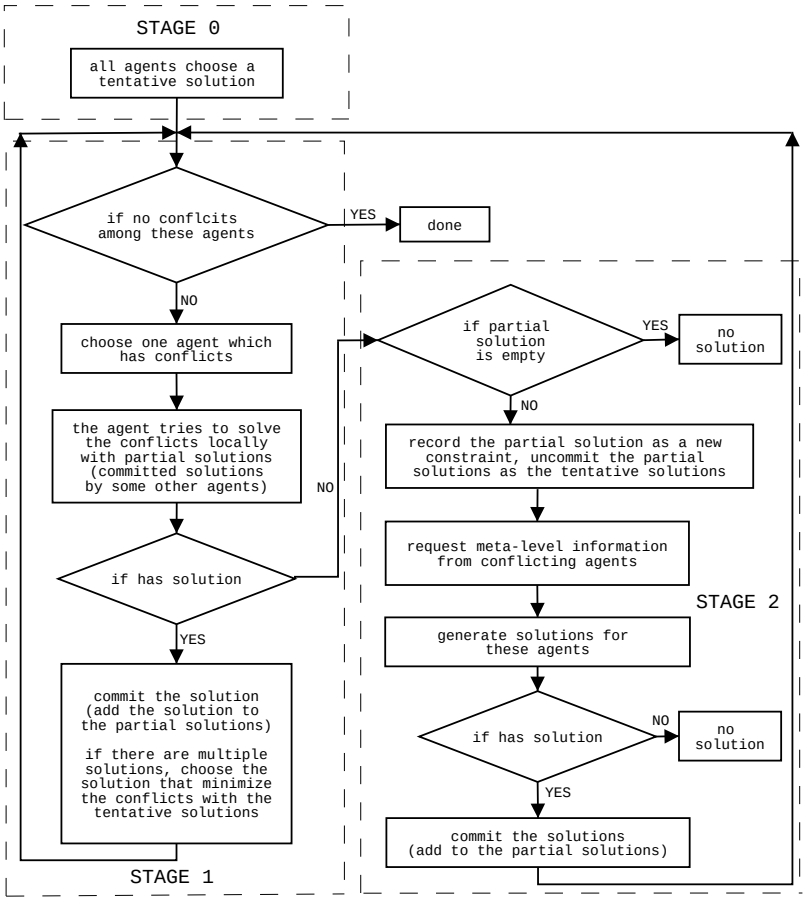



*Figure 1.2.* Synchronous SPAM

In the synchronous SPAM algorithm agents negotiate in a more tightly cooperative manner. The algorithm has additional features, such as priority and commitment, to guarantee the completeness. In this algorithm, each agent has a tentative initial solution for its local problem. The tentative solution is revised when the agent commits its solution. All the committed sub-solutions

constitute a partial solution to the overall problem. The revised solutions must satisfy all the constraints with the sub-solutions. If there exist multiple solutions, a min-conflict heuristic [Minton et al., 1992] is used to minimize as many conflicts with tentative solutions as possible. If no solution exists, the partial solutions will be added as a new constraint, and all the committed solutions will be uncommitted and become tentative solutions again. Here the idea is borrowed from weak-commitment search algorithm [Yokoo, 1998] which has proved to be more efficient than backtracking algorithms in many cases. After the partial solution is uncommitted, the agent will request meta-level information from the other conflicting agents and try to generate solutions for all of these agents. If no solution exists, the whole problem will have no solution at the current objective level. As a result, some agents must lower their objective levels, i.e., reduce some constraints of their local problems. Otherwise, these agents will just commit the solutions. Note that when there is no solution, the algorithm will not try to minimize the number of violated constraints. Thus this algorithm is for solving constraint satisfaction not optimization.

As we will see in the next section, this algorithm can guarantee completeness. It can also be easily modified into an asynchronous version in which multiple agents can search in parallel but still guarantee the completeness.

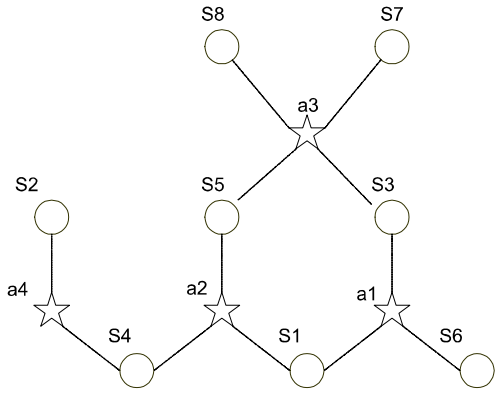
## 5. Experimental Analysis and Results

To reveal the properties of the SPAM protocol, we now analyze the completeness, convergence rate, complexity and scalability of sequential and synchronous SPAM algorithms. Here, the completeness of a protocol is the capability of finding a solution if one exists; the convergence rate concerns the number of negotiation steps required to reach a solution; the complexity measures the total number of steps taken before a protocol terminates; and the scalability considers how the properties of a protocol change as the size of a system increases.

### 5.1 Completeness

The synchronous SPAM protocol can be proved complete. Since the algorithm records the abandoned partial solutions as new constraints, the algorithm will not create the same partial solution twice. Therefore, the completeness of the protocol is guaranteed, because there are a finite number of partial solutions to be enumerated. The worst-case time complexity of this protocol is obviously exponential in the number of agents. Assuming that there are $n$ agents, and each agent has a solution space of size $S$, the worse case time complexity will be $O(S^n)$. Since the problem itself is NP-complete, this result seems inevitable. The worst case space complexity of this protocol is also exponential in the number of agents since if there is no solution for the whole problem, all

the partial solutions will be added as new constraints. However, unlike most of the tree search algorithms, the synchronous SPAM changes the search order flexibly, which makes it more efficient since it avoids exhaustively searching all the bad solutions when previous values are set wrong.




Problem Matrix

| | s1 | s2 | s3 | s4 | s5 | s6 | s7 | s8 |
|---|---|---|---|---|---|---|---|---|
| a1 | 1 | | 1 | | | 1 | | |
| a2 | 1 | | | 1 | 1 | | | |
| a3 | | | 1 | | 1 | | 1 | 1 |
| a4 | | 1 | | 1 | | | | |

Initial Assignment

| | s1 | s2 | s3 | s4 | s5 | s6 | s7 | s8 |
|---|---|---|---|---|---|---|---|---|
| a1 | 1 | | | | | 1 | | |
| a2 | 1 | | | | 1 | | | |
| a3 | | | 1 | | | | 1 | |
| a4 | | 1 | | 1 | | | | |

conflicts

*Figure 1.3.* An example for incompleteness

The sequential SPAM protocol is not complete. This is simply because that the agents in this algorithm do not follow any ordering of assignments during the search. This will cause some agents to cycle through some sub-solutions and fall into an infinite search or negotiation loop. In principle, the algorithm is similar to a local search and can step on a loop in search space or be trapped by local minima. To avoid such an infinite loop or local minimum, the protocol simply gives up the current search by lowering the current objective level if an oscillation is detected. This makes the algorithm incomplete to find some possible solutions.

Figure 1.3 gives a simple example of incompleteness where four agents (track managers) try to utilize eight resources (sensors) for target tracking. The problem has a solution which gives each agent a maximum objective level of two. The synchronous SPAM algorithm can find the solution after several

steps given the initial assignment shown in the Figure 1.3, while the sequential SPAM fails to find such a solution, due to oscillations in search processes.

| | a1 | | a2 | | a3 | | a4 | | notes |
|---|---|---|---|---|---|---|---|---|---|
| Step 0 | s1 | s6 | s1 | s5 | s3 | s7 | s2 | s4 | (a1/a2) conflicts on s1 |
| Step 0 | s1 | s6 | s1 | s5 | s3 | s7 | s2 | s4 | a1 negotiates with a2 |
| Step 1 | s1 | s6 | s4 | s5 | s3 | s7 | s2 | s4 | (a2/a4) conflicts on s4 |
| Step 1 | s1 | s6 | s4 | s5 | s3 | s7 | s2 | s4 | a2 negotiates with a4 |
| Step 2 | s1 | s6 | s1 | s5 | s3 | s7 | s2 | s4 | (a1/a2) conflicts on s1 |
| Step 2 | s1 | s6 | s1 | s5 | s3 | s7 | s2 | s4 | a1 negotiates with a2 |

oscillation

*Figure 1.4.* Execution of sequential SPAM on the example

Figure 1.4 illustrates an example of execution steps of sequential SPAM. For the initial assignments (after stage 0), $a_1$ and $a_2$ have conflicts on the resource $s_1$. $a_1$ searches locally but finds that there is no local solution to resolve the conflicts (after stage 1). It then goes into stage 2 to negotiate with $a_2$. It finally finds a partial solution (assign $(s_4, s_5)$ to $a_2$ and $(s_1, s_6)$ to $a_1$) for both $a_1$ and $a_2$ which resolves conflicts. So in step 2, new values are assigned. Since $a_2$ and $a_4$ still have conflicts on $s_4$, $a_2$ propagates the negotiation. But note that $a_4$ has only two resources available, the only partial solution for $a_2$ and $a_4$ is to assign $(s_2, s_4)$ to $a_4$ and $(s_1, s_5)$ to $a_2$. So in step 2, new values are assigned. Now, if $a_1$ is to propagate the negotiation, it will find that the same situation in step 2 as in step 0. An oscillation is detected which makes the algorithm exit without finding the possible solution.

Although sequential SPAM is not complete, our experiments show that the possibility for the algorithm to be complete is very high, especially in under-constrained situations. The experiment is set up with 10 agents and 20 agents. The number of resources are 30 and 60 respectively. Each task has a fixed objective level of 3, which is the best possible in both situations. The availability of resources to agents varies from 0.1 to 0.9. The availability here simply means the probability that any resource can be used by an agent. For example, in our experimental setting of 10 agents and 30 resources, when availability equals 0.5, on average each agent will have 15 resources to use. This implies that when the availability increases, the constrainedness of the problem decreases. Given a set of agents, a set of resources, and the availability $p$, each problem instance in our experiment is generated by randomly adding an edge between an agent and a resource with the probability $p$. An edge between an agent and a resource simply means that the resource is available to the agent.

Figure 1.5 shows the ratio of the problems solved by sequential and synchronous SPAM algorithms over 10,000 problem instances, with different re-

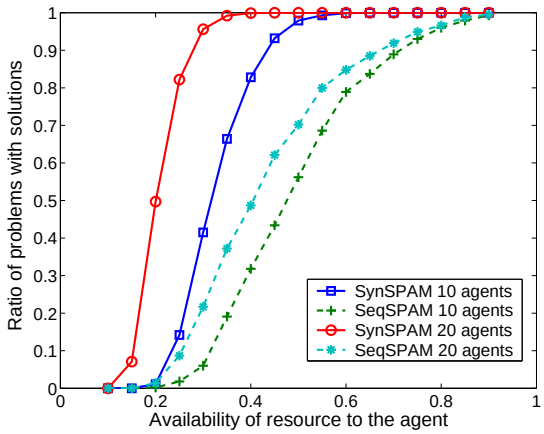



*Figure 1.5.* Number of problems solved by each algorithm

source availability. Since the synchronous SPAM algorithm is complete, the results of synchronous SPAM give a baseline for sequential SPAM. The result for the 10 agents case shows that when the resource availability is bigger than 0.4, most of the problems are solvable; whereas when the resource availability is less than 0.2, most of the problem are unsolvable. As with other CSPs, it is expected that the hardest instances of this negotiation problem are more likely to occur when the resource availability is between 0.2 and 0.4, where around half of the problems are solvable. Notice that when the problem size increases to 20 agents, the transition from the region with most unsolvable problems to the region with most solvable problems is even sharper. We suspect that a phase transition may exist in the negotiation problem.

Figure 1.6 shows the ratio of completeness of the sequential SPAM algorithm as the resource availability of the problem increases or the constrainedness of the problem decreases. This figure comes from the same results plotted in Figure 1.5. Each data point equals to the number of problems solved by the sequential SPAM algorithm divided by the number of problems solved by the synchronous SPAM algorithm. Since synchronous SPAM is a complete algorithm, the ratio directly reflects the ratio of completeness of sequential SPAM. The result shows that as the resource availability increases, the completeness of sequential SPAM increases as well. For the 20 agents problem, when the availability is greater than 0.4, the completeness ratio of sequential SPAM is more than half. When the resource availability increases beyond 0.6, more than 80% of the solvable problem instances are solved by the sequential SPAM algorithm. This simply indicates that the original SPAM protocol is well suited for under-constrained problems, having a very high possibility to be complete.

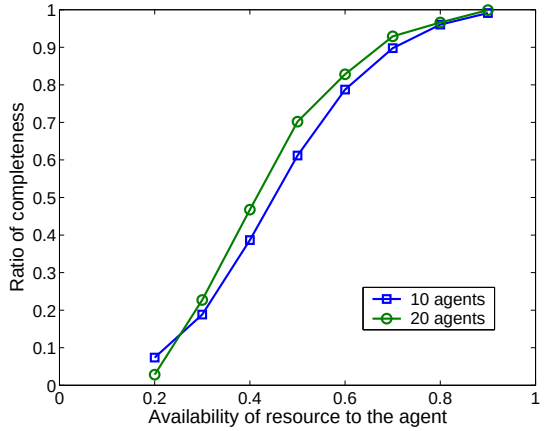



*Figure 1.6.* The rate of completeness for sequential SPAM

## 5.2 Time complexity

The sequential SPAM algorithm sacrifices completeness for computation time, as to be verified by experiments. Again, we run both sequential and synchronous SPAM algorithms on 10,000 problem instances with different resource availabilities. The problem settings are still 10 agents with 30 resources. Figure 1.7 plots the average total CPU time of each algorithm for these 10,000 instances in second. Both experiments were on a linux machine with 756 MB memory and an AMD 1.4GHZ processor.

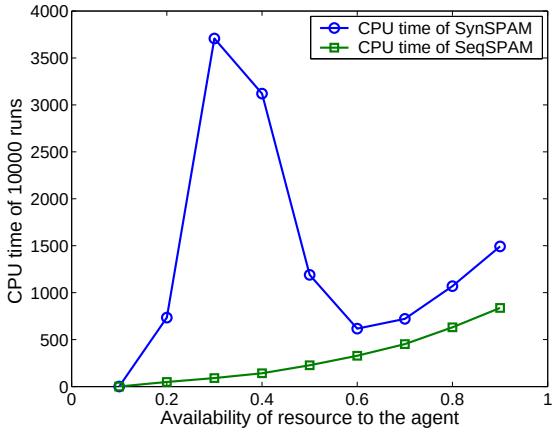



*Figure 1.7.* The CPU time of each algorithm

The result of the synchronous SPAM algorithm shows a phenomenon similar to that of phase transitions. It takes more CPU times on problems with resource availability at 0.3 and 0.4 than the problems in the other resource availability. The results here are consistent with the results in Figure 1.5 which shows that the problems around 0.3 and 0.4 availability are located in the middle of a phase transition on solubility, and thus are harder to determine quickly if they are solvable or not.

However, the result of the sequential SPAM algorithm does not show any phase-transition phenomenon. Its CPU time smoothly increases with the resource availability. Under all the availability settings we considered, the sequential SPAM algorithm takes less CPU time than the synchronous SPAM algorithm. One explanation is that the sequential algorithm is able to give up searching for a solution sooner on hard problems than the synchronous algorithm so that the former finishes faster than the latter. Another, minor reason is that the sequential SPAM algorithm has a lower overhead in each step.

One simple implication of these results is that there exists a tradeoff between the completeness and time complexity of a protocol. Adding additional features such as those we introduced in the synchronous algorithm may make a protocol complete, but meanwhile may decrease the time performance of the protocol. For many soft-constrained problems in practice where the completeness is not crucial, using incomplete algorithms or protocols such as the SPAM protocol seems to be the right choice.

## 5.3 Convergency and performance

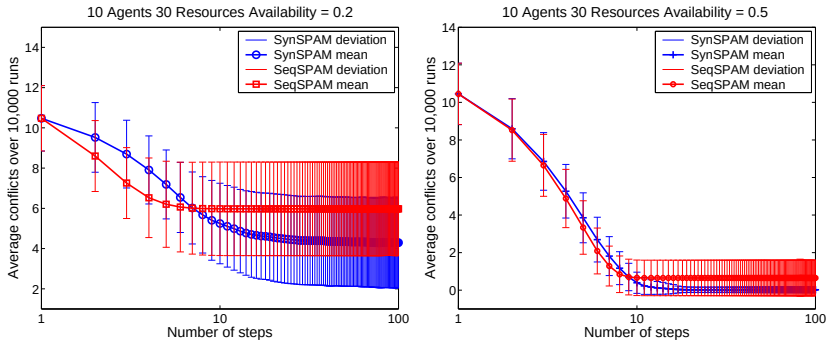



*Figure 1.8.* The convergency speed of two algorithms

The sequential and synchronous algorithms cannot guarantee to always turn the current state into a better one after each step of negotiation. For asynchronous SPAM, during the stage 1, when an agent commits a local solution

that has no conflict with partial solutions, this new solution may introduce more conflicts with tentative solutions than the conflicts it can reduce. Consequently, the overall conflicts are increased after one step of negotiation. Similar example can also be found in stage 2 of the sequential SPAM algorithm. However, both algorithms are able to terminate after a finite number of steps. Therefore, both algorithms will finally converge to some specific values. For the synchronous SPAM algorithm, this value is 0 if the problem is solvable as the algorithm is complete, while for the sequential SPAM algorithm, this value may not necessarily be 0.

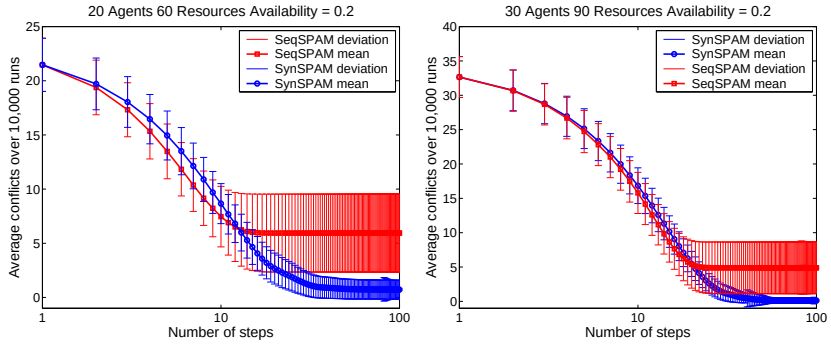



*Figure 1.9.* The convergency speed of two algorithms

Here, we are particularly interested in the rate of convergence of these two algorithms, which basically measures how fast the protocols improve solution quality. Figure 1.8 shows the experimental results with 10 agents and 30 resources averaging over 10,000 problem instances, indicating that the convergency speed of the sequential algorithm is normally better than that of the synchronous algorithm. This result is not surprising. If we compare the two algorithms, we will find that in each step, the synchronous SPAM algorithm tries to resolve the conflicts with only partial solutions, whereas the sequential SPAM tries to resolve the conflicts with all the solutions. This means that in each step, the sequential SPAM is likely to resolve more conflicts than the synchronous SPAM, so that its convergency speed may be faster. Figure 1.9 also shows the results on some larger problems with 20 agents and 30 agents.

Based on these results, we find the following interesting facts. Despite its incompleteness, when the sequential SPAM algorithm terminates, the final solutions are almost always near optimal. Specifically, the average solutions from sequential SPAM are only a few conflicts more than that of synchronous SPAM, especially in the under-constrained situations. This is summarized in Table 1.1, which shows the performance results of two algorithms on tracking problems with different problem sizes and resource availabilities. Each data

*Table 1.1.* Mean Conflicts Over 10,000 problem instances After 100 Steps

| *size* | *algo.* | *resource availability* | | | | | |
|---|---|---|---|---|---|---|---|
| | | *0.2* | *0.3* | *0.4* | *0.5* | *0.6* | *0.8* |
| *10 agents* | *Syn. SPAM* | 4.2956 | 0.8491 | 0.1761 | 0.0173 | 0.0016 | 0 |
| | *Seq. SPAM* | 5.9764 | 3.0456 | 1.3994 | 0.6950 | 0.2799 | 0.0409 |
| *20 agents* | *Syn. SPAM* | 0.7201 | 0.0377 | 0.0016 | 0 | 0 | 0 |
| | *Seq. SPAM* | 5.9953 | 1.9827 | 0.8805 | 0.4009 | 0.1965 | 0.0342 |

point in the table shows the average number of conflicts unsolved after 100 step execution of the algorithms, averaged over 10,000 random problem instances. Based on the table, most of the results of the sequential algorithm are comparative to the synchronous algorithm, especially in the under-constrained cases. For 10-agent problems, the biggest performance gap occurs when the resource availability is equal to 0.3, which is located near the middle of a phase transition on solubility. For 20 agent problems, the phase transition point shifts to the availability of 0.2, and correspondingly the biggest performance gap shifts as well. In all other regions apart from phase transition, the performance differences between the two algorithms are trivial.

In light of all the above results and the results on the completeness, we can conclude that for many real-time applications, it is reasonable to give up the total completeness of SPAM protocol in favor of a faster convergency to a good enough solution.

## 5.4 Scalability

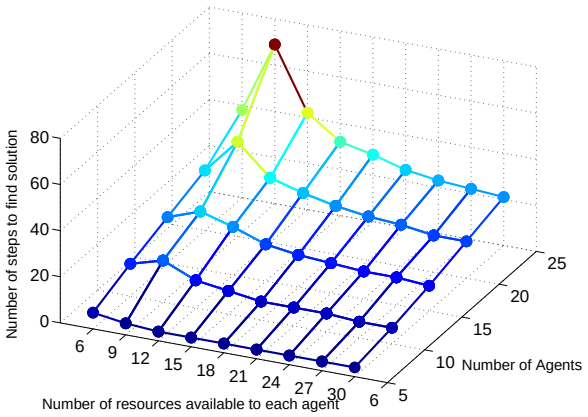



*Figure 1.10.* Synchronous SPAM scalability

Another important aspect of performance analysis of a protocol is the study of how the performance varies as problem size increases, i.e., how the protocol scales. This feature is particularly important in applications such as sensor networks, since such systems can easily have hundreds or thousands of sensors. A good scalability means that system performance does not degrade disproportionally with the size of the system or does not degenerate at all when the number of components increases. Here we are interested in answering the following question: will the number of negotiation steps that SPAM protocol takes to find a solution dramatically increase as the number of agents increases?

Figure 1.10 shows the experimental results for the synchronous SPAM protocol, in which the number of agents varies from 5 to 25, and the average number of resources available to each agent varies from 6 to 30. The total number of resources is three times of the number of agents in each of 1,000 random problem instances. In other words, the objective level for each agent is set to 3. Note that for the 5-agent problems, the maximum number of resources available to each agent is 15. We simply extend the data on 15 to 30 in the 5-agent case to make the results easy to show. In Figure 1.10, each data point represents the average number of steps over the solvable problems of 1,000 runs. From the results, we can see that the scalability of synchronous SPAM is super linear. Particularly around the phase transition area, the performance of the protocol substantially degrades.

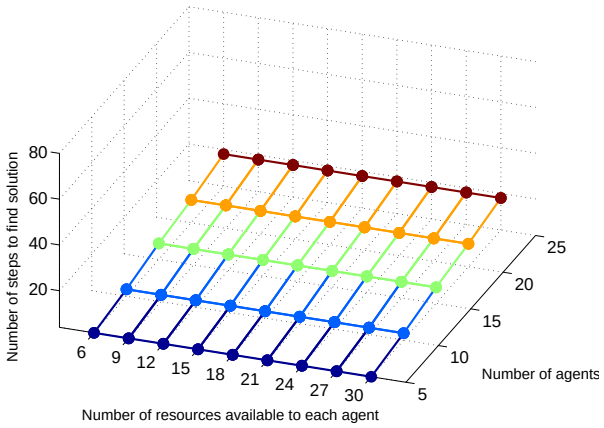



*Figure 1.11.* Sequential SPAM scalability

Figure 1.11 shows the experimental results on the sequential SPAM protocol with the same experiment setup as above. Again each data point represents the average result over the 1,000 solvable problem instances. Comparing to the results in Figure 1.10, Figure 1.11 shows that the scalability of sequential SPAM

is better than the scalability of synchronous SPAM. However, based on these results, we are unable to affirm that the scalability of sequential SPAM is linear. Nevertheless, the results indicate that the sequential SPAM algorithm does not seems to have a substantial degrading region around the phase-transition area similar to the synchronous SPAM algorithm. We should also note that at each point in Figure 1.11, fewer problems are completely solved than the corresponding point in Figure 1.10.

## 5.5 Summary

In this section, we experimentally analyze the properties and performance of two distributed search algorithms derived from a recently developed cooperative negotiation protocol for resource allocation in networks of distributed sensors. The sequential version simulates the protocol as close as possible, while the synchronous version adds some additional features to ensure the completeness. The experimental results on these algorithms help us understand the properties of the original protocol. Specifically, our results show that the SPAM protocol is not complete in terms of finding a solution. However it has a high probability of finding a solution if one exists, especially in under-constrained situations. The SPAM protocol is able to give up sooner on the hard problems so as to finish faster than the complete protocols. The protocol converges well in the sense that it can find good low-conflict solutions within a small number of steps. The solution quality is comparative to the complete protocol in most of the cases. The protocol seems to scale very well as the number of agents and the number of resources increase.

With all these results above and the fact that the SPAM protocol is normally used in a dynamic environment, it is reasonable to give up the completeness of the protocol in favor of computational complexity, convergency speed, and scalability. Note that in a dynamic environment, even if the protocol finally finds the optimal solution with a complete search, the problem could have already changed and the optimal solution may no longer be relevant. Therefore, making the right tradeoff between solution quality and computational time is critical, and the SPAM negotiation protocol seems to make such a tradeoff very well for real-time moving target tracking.

# 6. Conclusion and Discussions

Motivated to understand the properties of a recently developed cooperative negotiation protocol, we proposed an approach to analyzing distributed negotiation methods. In this approach, we view and formulate a negotiation protocol as a distributed search, and then experimentally investigate the properties and performance of the search algorithms to help draw conclusions on the original protocol. We demonstrated this approach on the Scalable Protocol for Any-

time Multi-level (SPAM) negotiation protocol for allocating resources among a set of cooperative distributed sensors. In addition to the contribution on a negotiation protocol itself, we substantially extended the well known notion of distributed AI as distributed search [Lesser, 1990]. We not only viewed a distributed cooperative negotiation as a distributed search, we also proposed to use search as a simulation tool to analyze negotiation protocols. As we demonstrated in this paper, this approach can overcome many difficulties inherent to a negotiation protocol that is hard to investigate analytically. We believe that this approach is general and can be carried over to analyzing other distributed problems and strategies.

In reflecting on the experimental results on the performance of the SPAM negotiation protocol obtained in this paper, it seems that high-quality global solutions to a distributed problem can be achieved without a global control but with a negotiation protocol in which agents rely on information restricted to their small neighborhoods. With a small locality of information sources, anytime performance of a system can also be significantly improved. The results in this paper and those on distributed stochastic search presented in [**?**] suggest that propagating information among neighboring agents and using information in a small neighborhood vicinity may be a good general strategy for distributed problem solving.

## Acknowledgments

This research was supported in part by NSF grants IIS-0196057 and ITR/ EIA-0113618, and in part by DARPA Cooperative Agreements F30602-00-2-0531 and F33615-01-C-1897. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government. Thanks to Sharlee Climer for reading a draft of this paper and Milind Tambe for suggestions to an early version.

# References


Adler, M. R., Davis, A. B., Weihmayer, R., and Worrest, R. W. (1989). Conflict-resolution strategies for nonhierarchical distributed agents. In *Distributed Artifical Intelligence*, volume 2, pages 139–162.

Cammarata, S., McArthur, D., and Steeb, R. (1983). Strategies of cooperation in distributed problem solving. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 767–770.

Conry, S. E., Meyer, R. A., and Lesser, V. R. (1988). Multistage negotiation in distributed planning. In *Readings in Distributed Artifical Intelligence*, pages 367–384.

Durfee, E. H. and Lesser, V. R. (1989). Negotiation task decomposition and allocation using partial global planning. In *Distributed Artifical Intelligence*, volume 2, pages 229–244.

Durfee, E. H. and Montgomery, T. A. (1990). A hierarchical protocol for coordinating multiagent behaviors. In *Proceedings of the Eight National Conference of Artificial Intelligence*, pages 86–93.

Lander, S. E. and Lesser, V. R. (1992). Customizing distributed search among agents with heterogeneous knowledge. In *Proceedings of the First International Conference on Information and Knowledge Management*, pages 335–344.

Lesser, V. R. (1990). An overview of dai: Viewing distributed ai as distributed search. *Journal of Janpanese Society for Artificial Intelligence-Special Issue on Distributed Artificial Intelligence*, 5(4):392–400.

Mailler, R., Vincent, R., lesser, V., Shen, J., and Middlekoop., T. (2001). Soft-real time, cooperative negotiation for distributed resource allocation. In *AAAI Fall Symposium on Negotiation Methods for Autonomous Cooperative Systems*.

Minton, S., Philips, A., Johnston, M. D., and Laird, P. (1992). Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58(1-3):161–205.

Yokoo, M. (1998). *Distributed Constraint Satisfaction*. Springer.