Negotiation On Data Allocation in Multi-Agent Environments*

Rina Schwartz[1] **Sarit Kraus**[1,2]
[1]Department of Mathematics and Computer Science
Bar-Ilan University, Ramat-Gan, 52900 Israel
{schwart, sarit}@macs.biu.ac.il
[2]Institute for Advanced Computer Studies
University of Maryland, College Park, MD 20742

Abstract

We propose a strategic negotiation model that takes into account the passage of time during the negotiation process itself in order to solve the problem of data allocation in environments with self-motivated servers which have no common interest and no central controller. The model considers situations characterized by complete, as well as incomplete, information. Using this negotiation mechanism, the servers have simple and stable negotiation strategies that result in efficient agreements without delays. We provide heuristics for finding the details of the strategies which depend on the specific settings of the environment, and demonstrate the quality of the heuristics, using simulations. We prove that our methods yield better results than the static allocation policy currently used for data allocation for servers in distributed systems.

Negotiation is proposed in Distributed Artificial Intelligence (DAI) as a means for agents to communicate and compromise to reach mutually beneficial agreements. Negotiation is especially beneficial in multi-agent systems (MA), where the agents are self-motivated, and where there is no central controller (Rosenschein & Zlotkin 1994). In (Kraus, Wilkenfeld, & Zlotkin 1995), a strategic model of negotiation (Osborne & Rubinstein 1990) was developed that takes the passage of time during the negotiation itself into consideration. It provides simple, efficient, and stable strategies for automated negotiators in a broad range of situations. In this paper, we apply and extend the strategic model of negotiation to the problem of data allocation in multi-agent environments. In our application, there are several information servers which are self-motivated but share data, and they negotiate on the allocation of data items.

A specific example of a distributed knowledge system is the Data and Information System component of the Earth Observing System (EOSDIS) of NASA (NASA 1997). It is a distributed system which supports archival data and distribution of data at multiple and independent data centers (called DAACs). The current policy for data allocation in NASA is static: each DAAC specializes in some topics. When new data arrives at a DAAC, the DAAC checks if the data is relevant to one of its topics, and, if so, it uses other criteria, such as storage cost, to decide whether to accept the data and store it in its database. If the data is not relevant to the DAAC, it may forward it to another DAAC whose topics seem more relevant.

Previous work on file (data) allocation in distributed systems (e.g., (Dowdy & Foster 1982; Du & Maryanski 1988)) considers systems in which a central decision maker exists, who tries to maximize the performance of the overall system. This assumption is not valid in many cases today, when trying to distribute information among self-motivated servers. We propose the use of negotiation as a solution method for environments with self-motivated servers which have no common interest and no central controller. We applied the strategic model of negotiation to such environments and provided utility functions for the agents. As it turns out, some of the assumptions made in previous negotiation models, e.g., (Kraus, Wilkenfeld, & Zlotkin 1995), are not valid in our situations. This required the adaptation of the negotiation model and finding new stable strategies. Using these strategies, the agents reach efficient agreements without delay. We provide heuristics for finding the details of the strategies, which depend on the specific setting of the environment, and we also demonstrate the heuristics' quality, using simulations. We deal with cases of complete information systems, as well as that of incomplete information. We proved that our methods yield better results than the static allocation policy currently used in such systems. Using negotiation efficiency by self-motivated servers demonstrates the benefits of using a negotiation mechanism in real world, multi-agent environments.

Environment Description

In the environment which we consider, there is a set of several (more than two) information servers, denoted

 This material is based upon work supported in part by the NSF under Grant No. IRI-9423967. Rina Schwartz is supported by the Israeli Ministry of Science.

by SERV, connected with a communication network. Each server is located in a different geographical area and receives queries from clients in its area. In response to a client's query, a server sends back information stored locally or information stored in another server, which it retrieves from that server. In our model, each server has its own interests and wants to maximize its own utility, which is described below.

The information is clustered in datasets.[1] Each dataset is characterized by a set of keywords and contains documents, corresponding to a 'granule' in EOSDIS. A negotiation session is initiated when a set of new datasets, denoted by DS, arrive, and each new dataset has to be allocated to one of the servers by mutual agreement among all of them. The new datasets are stored in a temporary buffer until a decision is made for their allocation. Negotiation is a process that may include several equidistant iterations. We assume that agents can take actions in the negotiation only at certain times in the set $Time = \{0, 1, 2...\}$ that are fixed in advance. An *allocation* is an assignment of each dataset to one server, i.e., a function Allocation : DS $\mapsto$ SERV, indicating the server where each dataset will be stored. We denote the set of all possible allocations by *Alloc*.

Utility Function

Each server, i, receives queries from clients in its area and answers them by sending back documents which may belong to a dataset located in i, or to a dataset located in another server, $j \neq i$. The clients pay server i *query_price* per document which they received. If an answer document is located in a remote server, j, server i needs to retrieve it from j. The cost for i to retrieve a document from server j depends on the virtual distance between i and j, which is specified by the function *distance*. The virtual distance is measured in terms of delivery time, which plays an important role in loaded systems where the documents are very large (e.g., images). We denote by *retrieve_cost* the cost for server i of retrieving one document for one unit of distance.[2] Providing documents to other servers, when they need it for answering their queries, is also costly. We denote by *answer_cost* the cost for server j of providing another server with one document over one unit of distance.

An important factor that plays a role in the utility function of a server from an allocation of a specific dataset is the expected usage of this dataset by the server and other servers. $Usage : SERV \times DS \mapsto R^+$ is a function which associates with each server and dataset the expected number of documents of this dataset which will be requested by clients in the area of this server. Finally, consider the storage cost. We denote by *storage_cost* the cost of storing one data unit of a dataset in a server.[3] The function *dataset_size* specifies the size of each dataset in data units.

Utility from one dataset The utility function of an agent from a given allocation of a set of datasets is the combination of its utility from the assignment of each dataset. Taking the above parameters into consideration, the utility for a server from the assignment of one dataset, (ds), to a certain location, (loc), is as follows:

Attribute 1

$$V_{server}(ds, loc) = \begin{cases} local(ds) & loc = server \\ remote(ds, loc) & otherwise \end{cases}$$

where

$$\begin{aligned} local(ds) = {} & usage(server, ds) * query\_price - \\ & storage\_cost * dataset\_size(ds) - \\ & \textstyle\sum_{d \in \mathrm{SERV}} usage(d, ds) * \\ & \quad distance(d, loc) * answer\_cost \end{aligned}$$

and

$$\begin{aligned} remote(ds, loc) = {} & usage(server, ds) * query\_price - \\ & usage(server, ds) * distance(server, loc) * \\ & retrieve\_cost \end{aligned}$$

The utility of a server from an allocation consists of the utility from the assignment of each dataset.

Definition 1 *The function $U_{server} : Alloc \times Time \mapsto R$ specifies for this server the utility from each possible allocation at each time period. It consists of the utility from the assignment of each dataset and the cost of negotiation delay.*

We will first introduce U_{server}, ignoring the cost of negotiation delay.

Attribute 2 *For each server $\in$ SERV, and $S \in$ Alloc:*

$$U_{server}(S, 0) = \sum_{x \in \mathrm{DS}} V_{server}(x, S(x))$$

According to the above definition, the utility from the assignment of one dataset is independent of the assignment of the other datasets. This property reflects that we do not take into consideration the overall load of the system, which may cause delays in transferring information and which yields more transfer costs. However, a severe load on one server is prevented, since, as we show later, the servers reach fair agreements.

Effect of negotiation time on the utility function For a server participating in the negotiation process, the time when an agreement is reached is very important. There are two reasons for that. First, there is the cost of communication and computation time spent on

[1] A dataset corresponds to a *cluster* in information retrieval, and to a *file* in the file allocation problem.

[2] This cost is relevant only in environments in which each client is connected to the nearest server, which answers all his queries. In cases where the clients are autonomous and send their queries directly to the server which has the answer, $retrieve\_cost = 0$.

[3] For simplicity, we assume that storage space is not restricted.

the negotiation. Second, there is the loss of unused information: until an agreement is reached, new documents cannot be used. Thus, the servers wish to reach an agreement as soon as possible, since they receive payment from answering queries. Usage of a stored dataset is considered to decrease over time, since the information becomes less up to date. In our environment, if there is a constant discount ratio of dataset usage and storage cost, then there is also a constant discount ratio of the utility from an allocation. Thus, the utility function of an agreement depends on the details of the agreement and on its time.

Attribute 3 *For $S \in Alloc$ and $t \in Time$, $U_i(S,t) = \delta^t * U_i(S,0) - t * C$, where C is the constant negotiation cost for each time delay, and $\delta < 1$ is the discount rate factor of the allocation.*

The Negotiation Process

One of the attributes of the environment described above is that each server cares about the exact location of each dataset, even if it is stored in another server. Conflicts among the servers may arise with respect to their preferences about dataset allocations. We propose a negotiation process for solving these conflicts, whereby all the servers participate in the negotiation, until an agreement is reached concerning the distribution of the datasets among the servers.

Mechanism for Negotiation

The negotiation protocol is the model of Alternating Offers (Osborne & Rubinstein 1990). As we mentioned above, it is a process that may include several iterations and may even continue forever. The mechanism only provides a framework for the negotiation process and specifies the termination condition, but there is no limit on the offers which are made by the agents.

There are $N \geq 3$ agents, randomly designated 1,2,...,N. Each agent represents one server in the negotiation, and they all need to reach an agreement together on the dataset allocation. In each period $t \in Time$, if the negotiation has not terminated earlier, agent j(t), where j(t)=t mod N, will offer a possible allocation for all the datasets considered, (i.e. $S \in$ Alloc), and each of the other agents may either accept the offer (choose Yes) or reject it (choose No) or opt out of the negotiation (choose Opt).

If an offer is accepted by all the agents, then the negotiation ends, and this offer is implemented. If at least one of the agents opts out of the negotiation, then the *conflict_alloc* is implemented. We consider the conflict allocation to be a static allocation. That is, each new dataset will be allocated to a server with old datasets with topics which are the most similar to the new datasets' topics, where the similarity is measured using any correlation method. If no agent has chosen 'Opt', but at least one of the agents has rejected the offer, the negotiation proceeds to period t+1, agent j(t+1) makes a counter-offer, the other agents respond, and so on.

Note that the protocol specifies that when an agent makes an offer, all the other agents respond simultaneously. Implementing a simultaneous responding protocol can be done by assuring that each agent sends its response before it reads new messages it has received.

An agent's negotiation strategy, in general, is a function of the history of the negotiation to its next move, which can be an offer, if it is its turn to move, or Yes, No, or Opt, if it needs to respond to an opponent's offer.

We assume that each agent prefers any agreement during any given time period over continuation of the negotiation process indefinitely. We also assume that the utility of each server from the conflict allocation, i.e., the static allocation, is never negative. A non-negative utility from a static allocation can be achieved when the price of the queries is high enough. We will show in the next lemma that, in our case, the relation between the utility of an offer and that of the conflict allocation does not change over time. This is different from the attributes of the model presented in (Kraus, Wilkenfeld, & Zlotkin 1995).

Lemma 1 Opting out vs. other offers: *If the model satisfies the assumptions above, then for each $i \in SERV$, $S \in Alloc$, $t1, t2 \in Time$, if $U_i(S, t1) > U_i(conflict\_alloc, t1)$, then $U_i(S, t2) > U_i(conflict\_alloc, t2)$.*

Lemma 2 Opting out costs over time: *If the model satisfies the assumptions above, then if $t1 < t2$, then $U_i(conflict\_alloc, t2) \leq U_i(conflict\_alloc, t1)$.*

The following defines the set of offers which is better for agent i than is opting out.

Definition 2 Offers that are preferred over opting out: *For every $t \in Time$ and $i \in SERV$, we define $\overline{OFFERS}^i = \{S | S \in Alloc, U_i(S,t) \geq U_i(conflict\_alloc, t)\}$. We denote by $\overline{OFFERS}$ the set of all offers that are* individual rational, *which means the offers that are preferred by* all *agents over opting out: $\overline{OFFERS} = \bigcap_{i \in SERV} \overline{OFFERS}^i$.*

Equilibrium of the Negotiation

In this section, we present simple strategies for the agents in our environment that we could recommend to *all* agent designers to build into their agents. No designer will benefit by building agents that use any other strategy, and no agent will benefit from deviating from its strategy, given that all the other agents follow them.

Subgame Perfect Equilibria We will use the strong notion of (subgame) perfect equilibrium (P.E.) (Rubinstein 1982). A profile of strategies is a perfect equilibrium if in each stage of the negotiation, assuming that all agents, but one, follow their strategies in the profile, the other agent has no better strategy than to follow its own strategy in the profile.

Definition 3 *A strategy profile is a* subgame perfect equilibrium *of a model of alternating offers if the strategy profile induces in every subgame a Nash equilibrium (Nash 1953) of that subgame.*

Existence of Multiple Equilibria We have proven that for every allocation, S, which is not worse for all agents than is the conflict allocation, there is a profile of strategies in P.E. which results in accepting S in the first period of the negotiation (see also (Haller 1986)). We formally state this result in the next theorem.

Theorem 1 *If the model satisfies the assumptions above, then for each offer $S \in \overline{OFFERS}$, if S is better than is opting out for all agents, then there is a* subgame-perfect equilibrium *of the model of alternating offers, which yields the outcome where S is offered and unanimously accepted in period 0.*

A sketch of the proof: Consider the following strategy, f_i, for agent i: for $t \geq 0$, if $i = j(t)$, then offer S; if $i \neq j(t)$ and you were offered S', then if $S' = S$, respond with Yes, and if $S' \neq S$, opt out.
We proved that the strategy profile $f_1, ..., f_N$ is a P.E. which yields the outcome where S is offered and unanimously accepted in period 0. □

Implementation As was shown, if the agents follow the negotiation protocol described above, any offer S, which is better for all the servers than is the conflict allocation, has a subgame perfect equilibrium, which leads to the acceptance of S during the first period of the negotiation. Since the conflict allocation is the same as the static allocation of datasets, negotiation can always guarantee, at least, the utility from the static allocation.

The problem is how to choose such an offer S. It strongly depends on the exact setting of the negotiation session and cannot be given to the agents in advance. We propose that the designers of the servers agree, in advance, on a joint technique for choosing S. They can decide upon a mechanism which will find an allocation which must, at the very least, give each agent its conflict utility, and, under these constraints, maximize some social-welfare criterion,[4] such as the sum of the servers' utilities, or the generalized Nash product of the servers' utilities,[5] i.e., $\pi(U_i(X) - U_i(conflict\_alloc))$.

Then, the designers will provide their agents, in advance, with that mechanism and with the strategies of Theorem 1, which are in perfect equilibrium and lead to the acceptance of a chosen allocation in the first negotiation period. When a set of new datasets arrive, the agents will use the given mechanism to find an allocation and will negotiate using the equilibrium strategies. If all the designers had agreed, in advance, on the mechanism for finding S, it would not be beneficial for any agent to deviate from using this technique and from following the appropriate strategy, which is in perfect equilibrium, given that the other agents follow them.

In particular, we propose that the designers agree upon classic search methods in order to find a global-optimal solution, which would be better for all the agents than opting out would be. However, we proved, by reduction from the Multiprocessor Scheduling problem (Garey & Johnson 1979), that the problem of finding an allocation maximizing a welfare criterion with restrictions on the servers' utilities is NP-complete. Thus, searching for an optimal solution is not practical in a large system, and the agents should search for suboptimal solutions.

As will be described below, we found that randomized methods may be more beneficial for all the agents than deterministic ones. However, the agents must jointly agree on the same allocation S, and if they use a random mechanism, each will find a different allocation. In order to solve this problem, we suggest dividing the negotiation protocol into two phases in situations where randomized methods are beneficial. In the first phase, each server will search for an allocation, using any (non-deterministic) algorithm and resources it has. All the agents will simultaneously broadcast their findings to the other agents at the end of the phase, and the one with the highest value of the social-welfare criterion which were agreed upon by the designers will be designated as the chosen S. In the second phase, the negotiation will be performed, using the perfect equilibrium strategies with respect to S. We propose that in the first phase all the agents try to maximize the social-welfare criterion.

If finding optimal allocation is possible, then the strategies which maximize the chosen social-welfare criterion are in equilibrium. But, in most of the cases of our problem, when the number of servers and datasets is large enough, finding the optimal allocation within a reasonable time frame is impossible. In such cases, if the servers use a non-deterministic mechanism, then it may be worthwhile for a server which has significantly more computational power than the others to deviate and try to find an offer which is primarily good for itself, and only secondarily for society. However, if the servers have similar power, then its strategies which do not maximize the social-welfare criteria are not stable. If we promise a bonus which is high enough for the

[4] It is possible to reach such results using other negotiation protocols (e.g., (Rosenschein & Zlotkin 1994)). However, our protocol put fewer restrictions on the agents' strategies, which is preferable in environments of autonomous agents.

[5] This solution was proved by Nash to be the unique solution, given some basic axioms and when the set of possible outcomes of the bargaining problem is compact and convex. In our situation, the set of allocation is not convex, but we use the generalized Nash product as a reasonable bargaining outcome.

server whose suggestion has been accepted in the first phase, then maximizing the social-welfare criterion becomes the best strategy for all agents.

Experimental Evaluation

In order to test different techniques for finding an offer which will serve as the basis for equilibrium strategies and to examine the effect of different parameters of the environment and the servers' utility functions on negotiation results, we developed and implemented a simulation of our servers' environment. For lack of space, we do not present the details of our simulations, but simply discuss briefly the results which were obtained.

We tested three methods for finding suboptimal solutions to the problem. The first was a deterministic backtracking algorithm, the second a heuristic repair method (Minton *et al.* 1992) based on a random-restart hill-climbing mechanism, and the third a genetic algorithm (Siegelmann & Frieder 1992). We found that backtracking is the best technique when the number of datasets and servers is relatively small (e.g., 10 datasets and 11 servers). As the number of datasets and servers increases, the performance of hill-climbing and genetic algorithms relative to the backtracking increases. Hill-climbing was the best method for maximizing the sum of the utilities of the servers, and the genetic algorithm was the best for maximizing the generalized Nash product of the servers' utilities.

As we proved, negotiation always leads to better results than does the static approach; however, we would like to identify specific cases in which negotiation is especially beneficial. We compared the performance of our negotiation techniques in different settings, maximizing the sum of the agents' utilities. In the comparison we used a measurement which excluded the gains from queries and the storage costs since their influence on the sum of the servers' utilities does not depend on a specific allocation. In particular, we denote by $vcosts(alloc)$ the variable costs of an allocation which consist of the transportation costs due to the flow of queries. Formally, given an allocation, its variable costs are defined as the following:

$$vcosts(alloc) = \sum_{ds \in \mathrm{DS}} \sum_{s \in \mathrm{SERV}} usage(s, ds) * distance(s, alloc(ds)) * (answer\_cost + retreive\_cost).$$

The actual measurement which we use is denoted by $vcost\_ratio$, the ratio of the variable costs when using negotiation and the variable costs when using the static allocation mechanism. The efficiency of the negotiation technique increases as the $cost\_ratio$ decreases.

By running several simulations, we examined how the change in several parameters of the environment influences $vcost\_ratio$. In each set of simulations, we kept all but one of the parameters fixed. Each set of simulations contained 50-200 runs on randomly generated environments, with 11 servers and 100 datasets. In all the cases, we ran the hill-climbing algorithm on Sparcworks machines, for bounded time, which was linearly dependent on the problem size. We took $answer\_cost$ to be similar to $retrieve\_cost$, and thus, in general, theorized that each server prefers to store datasets in remote areas rather than locally.

We examined the effect of the distribution of the usage of datasets by servers. First, we considered situations where the usage of datasets by the servers has a uniform distribution between 0 and mean*2. We discovered that changing the mean does not significantly influence the variable cost ratio. We also considered a normal distribution of the usage with a given standard deviation (std) and mean. In this case, the variable cost ratio decreases as the std of the usage increases, i.e., negotiation is more beneficial when the usage of datasets is more dispersed.

We studied the effect of the distance between any two servers on the $vcost\_ratio$. We examine a uniform distribution between a given minimal (min) and and maximal (max) distances, and a normal distribution with a given std and mean of (min+max)/2. In the first case, when the max distance (and thus the mean) of the distribution increases, negotiation becomes slightly more beneficial. In the second case, as the std of the distance increases, the variable cost ratio decreases, i.e., negotiation is of greater benefit when there is a greater difference between the distances of the servers.

We also examined the effect of changes in the cost factors on the variable cost ratio. Changing the query price did not influence the results significantly. However, as the $retrieve\_cost$ increases, the benefit of negotiation also increases. On the other hand, increases in $answer\_cost$, as well as increases in the datasets size or storage costs, causes negotiation to be less beneficial. This may be the result of the fact that increases of these values cause the storage of datasets to be less beneficial, and thus it is more difficult to find a good allocation since there are more constraints, and thus the improvement rate is lower.

The number of servers and datasets also influences the results. When the number of servers is kept fixed, as the number of the datasets increases, the negotiation becomes more beneficial. On the other hand, when the number of datasets is fixed, as the number of servers increases, the benefit from the negotiation decreases. We suspect that the reason for the first observation is that dataset allocations are independent of one another. Thus, as the number of datasets increases, there are more possibilities for increasing the benefit to all servers. On the other hand, when the number of servers increases, there are more constraints in finding possible allocations, and it is much more difficult to find a sub-optimal allocation.

We also studied the effect of the choice of the social-welfare criterion on the hill-climbing algorithm's results. We ran 50 experiments with the hill-climbing algorithm on randomly generated environments with 11 servers and 30 datasets. The social welfare criteria

method	vcost ratio	CU	CI
static	1	0.148	*
ave	0.77	0.118	0.94
prod	0.78	0.116	0.87
optimal	0.71	0.171	1.65

Table 1: Comparison of different social criteria.

which we studied were: the sum of the agents' utilities (sum), the generalized Nash product (pred) and the sum of agents' utilities without any constraints on the utility obtained by each agent which leads to the optimal solution of a centralized system (optimal). We compared these results with the servers' outcome from the static allocation, which can be obtained without negotiation (and also served as our opting out outcome). The results are presented in Table 1. The second column specifies the average of the *vcost_ratio*. The third column (CU) indicates the average of the relative dispersion of the utility among the agents (std/mean), and the last one specifies the average of the relative dispersion of the added benefit (CI) of the negotiation among the agents. The results indicate that maximizing the sum achieves a slightly lower vcost ratio vs. maximizing the Nash product and is not far from the optimal. Maximizing Nash product achieves a lower dispersion of the utilities and of the gains due to the negotiation than maximizing the sum, and they both do much better than the optimal w.r.t dispersion of the utility among the servers.

Servers with Incomplete Information

In the previous sections, we assumed that all the servers have complete information about each other. In particular, they all know the expectations about future usage of each dataset by the clients located near each server. In real world situations, if there is neither a central statistical unit nor the ability to enforce true reporting, this assumption is not valid. Thus, we will present a distributed allocation mechanism for environments with agents with private information about the usage level of the datasets: each server knows only the usage level of the clients associated with it and the usage level of the datasets stored in its database.

There are several approaches that can be considered for reaching agreements on dataset allocation in an incomplete information environment, for example, a voting protocol, bidding mechanisms (Sandholm 1993), revelation principle mechanisms (Myerson 1979), and negotiation models without revelation steps. None of these models are applicable in our environment, for reasons which cannot be specified here, due to constraints of space. We propose to use a negotiation protocol with a revelation phase, as presented below.

The revelation mechanism includes the following steps. In the first step, all the agents are asked to report, simultaneously, all of their statistical information about past usage of datasets. Given this information, each server calculates the expected usage function, i.e., for each server the expected usage of each dataset for the clients around it. After this step, the negotiation will proceed as in the complete information case. In order to avoid manipulative reports in the first step, we have to ensure that, when using such a revelation process, no server will have an incentive to lie.

In presenting the revealing protocol and discussing its properties, we will use the notion of *local dataset* w.r.t server i to denote a dataset stored in server i. We will use the concept, *remote dataset* w.r.t. server i, to denote a dataset stored in another server. Furthermore, *local users* of server i are the users located in its geographical area. We propose that the servers use the following protocol:

(i) Each server i will broadcast the following:
(a) for each dataset ds the past usage of ds by server i
(b) for each server, j, and for each local dataset ds w.r.t. i, the past usage of ds by j.
The servers send this data simultaneously.
(ii) Each server will process the information obtained from the other agents, and then negotiation will take place, as in the case of complete information. No communication is allowed between the servers before step (i). The following defines situations in which two servers give different reports concerning the same fact.

Definition 4 Conflicting reports: *reports of server i and server j are in* conflict *if there is a local dataset ds w.r.t server i, such that i's and j's reports on the usage of dataset ds by server j are different.*

Conflicting reports by servers i and j indicate that at least one of the servers is lying. In such cases, a high penalty for both servers provides an incentive for truth-telling in most of the reports. The penalty imposed on agents i and j should be distributed evenly among the other servers. The following lemma shows that an agent will tell the truth about its own usage of remote datasets and about the usage of other servers of its local datasets.

Lemma 3 *If the agents follow the pre-negotiation protocol described above, and there is a high penalty exacted for servers with conflicting reports, then there is a Nash equilibrium where each server is telling the truth about its usage level of remote datasets and about other servers' usage of its local datasets.*

The problematic case is that of a server's reports on its usage of its own local datasets, which are motivated by queries of its local users. The server is able to lie about its usage of local datasets, since there are no other reports on the same facts, and thus a lie will not be immediately revealed. One possibility is to ignore all reports of any server about its own usage of a local dataset. This yields inefficiency in the negotiation outcome, but it prevents manipulations by the servers.

If the servers do use the data about self usage of local datasets, a server may change its reports if it expects a higher utility when doing so. However, if it reports a lower usage of a dataset ds than the real one, then the other servers will believe that its retrieval costs of documents of datasets similar to ds are lower than are the real retrieval costs. When using an allocation mechanism which maximizes the sum or the product of the servers' utilities, such a lie may lead to an allocation in which those datasets will be stored far from the liar (since it reported a low usage), and thus the liar's utility will be lower than its utility from the allocation that might have been chosen had it told the truth.

If a server reports heavier usage of a dataset ds than the real one, then it may cause datasets similar to ds to be stored by the liar. This may cause the liar's storage and answer costs to be higher than the retrieval costs if it had been stored elsewhere (based on an honest report.) Moreover, in such cases, the other servers may believe that its utility from opting out is lower than is the real one (since they believe that it has high retrieval costs of datasets similar to ds, which are allocated remotely according to the static allocation, which is implemented if an opting out event occurs). Thus, the agreed upon allocation may be worse for the liar than is opting out, and again, worse than the situation reached when reporting the truth.

Thus, without knowing exactly the reports of the other servers concerning their usage of their local datasets, lying may harm the server. Even if it has an estimate about the other servers' usage, it needs unreasonable computation time in order to compute which lie is beneficial, since the problem of finding an optimal allocation itself is NP complete, and the problem of finding which lie will lead to an allocation which is more beneficial than if it had told the truth is, in general, much more complicated. Thus finding a beneficial lie is intractable.

Conclusion

This paper presents a negotiation protocol for the data allocation problem in multi-agent environments. We proved that negotiation is beneficial and that agreement will be reached in the first time period. For situations of agents with incomplete information, a revelation process was added to the protocol, after which a negotiation takes place, as in the complete information case. Due to constraints of space, we discussed only one variation of the multi-server environment. However, we showed that our approach is beneficial in other related situations. For example, we examined situations in which other utility functions for the servers are more appropriate. For example, we studied situations where the assumptions which lead to a discount factor of time delay in the negotiation are not valid, e.g., cases where storage cost is constant and does not change over time. In these cases, too, we proved that every allocation which is preferred by all the servers over opting out, has a perfect equilibrium.

We also considered situations in which it is not possible to implement the negotiation protocol presented in this paper, in particular, situations where simultaneous responses in the negotiation protocol cannot be guaranteed. In the case of sequential responses of the agents in the negotiation, negotiations with bounded length are of special interest, for example, situations with a deadline, or situations where there is a time period during which no allocation is better than is opting out. In these cases, we proved that an efficient solution will be reached during the first time period, using backward induction (Kraus, Wilkenfeld, & Zlotkin 1995). As mentioned above, in our environment, a bidding mechanism is not applicable. However, when $retrieve\_cost = 0$, then bidding mechanisms may be preferred over negotiation mechanisms. Another possible extension is to allow allocation of data in more than one site, which we leave for future research.

References

Dowdy, L. W., and Foster, D. V. 1982. Comparative models of the file assignment problem. *Computing Survey* 14 (2):289–313.

Du, X., and Maryanski, F. J. 1988. Data allocation in a dynamically reconfigurable environment. In *Proc. of the IEEE Fourth Int. Conf. Data Engineering*, 74–81.

Garey, M. R., and Johnson, D. S. 1979. *Computers and Intractability: a Guide to the Theory of NP-completeness.* New York: Freedman and Comp.

Haller, H. 1986. Non-cooperative bargaining of $n \geq 3$ players. *Economics Letters* 22:11–13.

Kraus, S.; Wilkenfeld, J.; and Zlotkin, G. 1995. Multiagent negotiation under time constraints. *Artificial Intelligence* 75(2):297–345.

Minton, S.; Johnston, M. D.; Philips, A. B.; and Laird, P. 1992. Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligece* 58:161–205.

Myerson, R. 1979. Incentive compatibility and the bargaining problem. *Econometrica* 47(1):61–73.

NASA. 1997. EOSDIS Home Page. http://www-v0ims.gsfc.nasa.gov/v0ims/index.html.

Nash, J. F. 1953. Two-person cooperative games. *Econometrica* 21:128–140.

Osborne, M. J., and Rubinstein, A. 1990. *Bargaining and Markets.* California: Academic Press.

Rosenschein, J. S., and Zlotkin, G. 1994. *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers.* Boston: MIT Press.

Rubinstein, A. 1982. Perfect equilibrium in a bargaining model. *Econometrica* 50(1):97–109.

Sandholm, T. 1993. An implementation of the contract net protocol based on marginal cost calculations. In *Proc. of AAAI-93*, 256–262.

Siegelmann, H., and Frieder, O. 1992. Document allocation in multiprocessor information retrieval systems. Technical Report IA-92-1, George Mason Univ.