Bidding Mechanisms for Data Allocation in Multi-Agent Environments *

Rina Schwartz* and Sarit Kraus*,†

* Department of Mathematics and Computer Science,
Bar-Ilan University, Ramat-Gan, 52900 Israel
{schwart,sarit}@macs.biu.ac.il

† Institute for Advanced Computer Studies,
University of Maryland, College Park, MD 20742

Abstract. We propose a bidding mechanism for data allocation in environments of self-motivated data servers with no common preferences and no central controller. The model considers situations where each server is concerned with the data stored locally, but does not have preferences concerning the exact storage location of data stored in remote servers. We considered situations of complete, as well as incomplete, information, and formally proved that our method is stable and yields honest bids. In the case of complete information, we also proved that the results obtained by the bidding approach are always better than the results obtained by the static allocation policy currently used for data allocation for servers in distributed systems. In the case of incomplete information, we demonstrated, using simulations, that the quality of the bidding mechanism is, on average, better than that of the static policy.

1 Introduction

In this paper, we consider the problem of determining the location of data items in a distributed information system, where the information servers are self-motivated and each one is trying to maximize its own utility.

A specific example of a distributed knowledge system is the Data and Information System component of the Earth Observing System (EOSDIS) of NASA [7]. This distributed system supports archiving 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 specific topics. When new data arrive at a DAAC, the DAAC checks if the data are relevant to one of its topics, and if so, it uses other criteria, such as storage cost, to decide whether or not to accept the data and store them in its database. If the data is not relevant to the DAAC, it may forward it to another DAAC whose topics seem more relevant.

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

In this paper, we propose the use of a bidding mechanism as a solution method for the data allocation problem in environments where the servers are self-motivated and have no common preferences and no central controller, and where the clients are autonomous.[2] In addition, a server is concerned about the data stored locally, but does not have preferences concerning the exact storage location of data stored in remote servers. According to our approach, the location of each data unit will be determined using a bidding mechanism, where the server bidding the higher price for obtaining the data will actually obtain it.[3] This approach yields an efficient and fair solution, its implementation is simple, and the bidders are motivated to offer efficient prices.

Bidding has been used previously in Distributed Artificial Intelligence (DAI) in the contract net framework [11]. Agents in the contract net environment decompose their tasks to subtasks and subcontract them to other agents, using bidding. Extensions of the contract net for environments with self-motivated agents were proposed in [9]. Mullen and Wellman [6] proposed a market price model for decisions about establishing mirror sites in a network. They suggested competitive-market pricing of the transportation price (when no mirror site is established) and the price of establishing mirror sites. However, the competitive approach is not useful in our environment since it is applicable only in environments in which it is possible to produce more than one item of each product type, and in our environment each data item is unique.

A bidding mechanism is suggested by [8] for an automated negotiation environment, where phone companies compete to serve as carriers for long distance phone calls, with dynamic prices. In particular, they propose the use of *Vickrey's sealed bidding scheme* [12]. In this kind of auction, each bidder submits one bid, in ignorance of the other bids, and the highest bidder pays the amount of the second-highest bid and wins. In this paper, we have implemented a sealed bidding scheme for the data allocation problem in distributed knowledge environments. We have specified the rules of the bidding in the case of data allocation and suggested strategies for the servers.

In the following sections, we will describe the data allocation problem and suggest a utility function which characterizes the servers' preferences. Then we will suggest a bidding protocol for data allocations, which is a dominant strategy mechanism [5]. We will also consider the incomplete information case, where each server has information only about past usage frequency of local data, and suggests how it can estimate its utility from other data items (new or remote ones) in order to bid efficiently for them. We will discuss the servers' performance in the complete information case, as well as in the incomplete information case.

[2] Previous work on file (data) allocation in distributed systems (e.g., [3]) considers systems where a central decision maker exists, which tries to maximize the performance of the overall system. This assumption is not valid today in many cases, when the objective is to distribute information among self-motivated servers.

[3] Only one copy of each dataset is allowed, since the datasets considered are extremely large, and it is not efficient to allow multiple copies of a dataset. However, if the datasets are small it is possible to extend the model to allow multiple copies of each dataset.

2 Environment Description

We consider an environment in which there is a set of several (more than two) information servers, denoted by SERVERS, connected by a communications network. The information stored in each information server is clustered in datasets.[4] Each dataset is characterized by a set of keywords and contains a large number of documents.[5] The set of datasets in the system in a given time period is denoted by DS.

In this paper, we consider an information system where each client may retrieve information directly from the server in which it is stored, and s/he pays this server for the retrieved information. Therefore, in our environment, a server is concerned only with the datasets stored in its local databases, and is indifferent to the exact location of datasets not stored locally. The environment we consider changes dynamically. New datasets arrive frequently, and usage frequency of old datasets changes over time. The servers must determine the location in which each new dataset will be stored and they are also able to change the location of old datasets. Since the datasets considered are very large, each dataset can be stored only in one server, and it is forbidden for a server to store a dataset unless it has become its legal "owner." Conflicts among the servers may arise when two or more servers would like to store the same dataset locally.

In a centralized system, a solution for such a problem is simple: the location of each dataset will be determined so as to maximize the profits of the entire system. But a dataset allocation which is beneficial for the entire system may be non-beneficial for some of the servers. In our case, where each server has its own interests, the servers will follow the centralized solution only if it is beneficial to them. Thus, any proposed protocol must be fair and consider the preferences of all the servers in order to be accepted by the designers of the servers. Moreover, if there is incomplete information about the usage of datasets, then the centralized solution is not applicable, since nobody has enough information in order to compute this solution.

In all the situations which we considered, the servers are uncertain about the future usage of the datasets. First, we have considered a symmetric environment with *complete information*, where all the servers have the same knowledge about the past, and they have the same expectations about the future usage of each dataset by clients located in each area, but they do not know the actual future usage. Then, we have considered an asymmetric environment with *incomplete information*, where each server knows the past usage only of the datasets stored locally, but can only partially estimate the past usage of datasets stored by other servers.

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

[5] A document corresponds to a 'granule' in EOSDIS.

3 Utility Functions

In this section, we will describe the components of the utility derived by a server from storing a dataset. Recall that each server receives queries from clients and answers them by sending back documents which belong to the datasets located in its databases. The clients pay the server a *query_price* per document that is retrieved as an answer to a query. We assume that there is a monetary system in the environment which is used for this payment, as well as for other payments described below.

The cost of sending a document to a client depends on the virtual distance between the client and the server. It is measured in terms of delivery time, which plays an important role in loaded systems in which the documents are very large (e.g., images). For simplicity, we assume that each client is located in a geographical area of one of the servers, and in order to compute the distance between server i and a client which is located in the geographical area of server j, we use the distance between servers i and j. The function *distance* specifies the virtual distance between any two servers. The term *answer_cost* specifies the cost for a server providing a client from another area with one document over one unit of distance.

An important factor that plays a role in the utility function of a server from a specific dataset located in its database is the expected usage of this dataset by clients. $Usage : SERVERS \times DS \mapsto R^+$ is a function which associates with each server and a dataset the number of documents belonging to this dataset which will be requested by clients located in the geographical area of that server, during one time period (e.g., one week, one month, etc.). In addition, consider the storage cost. We denote by *storage_cost* the cost of storing one data unit of a dataset in a server for one time period,[6] and the function *dataset_size* specifies the size of each dataset in data units. Each server calculates the utility it obtains from a dataset location, given its estimation of the expected query flows related to this dataset. Note that *storage_cost* and *answer_cost* are common to all the servers. The following attribute defines the utility (or loss) for a server in one time period from one dataset stored in it, when its usage is known.

Attribute 3.1 *The profits which server* s *expects to obtain from storing dataset* ds *locally for one time period are as follows:*

$$V_s(ds) = -storage\_cost * dataset\_size(ds) + \sum_{s' \in SERVERS}(usage(s', ds) * (answer\_price - distance(s, s') * answer\_cost)).$$

V_s considers the costs and benefits due to queries which are obtained at one time period, and also the storage cost which is expected to be paid at each time period (e.g., using the disk space).

The following attribute defines the profits P_s which server s expects to obtain from storing one dataset over time. We assume that there is a monetary system in which each server is able to borrow any required amount of money at the current

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

interest rate r. Using this interest rate, P_s is evaluated as the net present value (NPV) of future income from queries related to the dataset, computed w.r.t. the interest rate r. The NPV is used in financing systems in order to find the value of an investment. It is computed by discounting the cash flows at the firm's "opportunity cost" of capital [2]. We will use the same term for finding the value of a dataset storage, considering the dataset as a possible investment.

The function below specifies the net present value of the profits flow accepted by server s related to dataset ds, assuming that ds will be stored in loc indefinitely. In subsection 5.5 we will discuss the expected profits for situations where old datasets can be reallocated.

Attribute 3.2 *The profits which server* s *expects to obtain from a dataset* ds *located in* loc *from time 0 until time N, given* V_s*, is as follows:*

$$P_s(ds, loc) = \begin{cases} \sum_{t=0}^{N} \frac{V_s(ds)}{(1+r)^t} & loc = s \\ 0 & otherwise \end{cases}$$

where r is the interest rate, and N is the number of periods during which the environment exists. If the environment is considered to exist forever, then $N = \infty$*. In this case,*

$$P_s(ds, loc) = \frac{V_s(ds) * (1 + r)}{r}.$$

We denote by $U_s(ds, loc)$ the utility which a sever s obtains from a dataset ds which is stored in location loc. In order to evaluate U_s, the server will consider the expected profits as well as the risk involved in obtaining these profits. Such a risk is involved in both the complete and the incomplete environments which we consider in this paper, since in both cases, *usage* is only estimated, and the server is not sure about the value of a dataset, since the queries flow is not certain and thus the payments due to queries are not certain. If s is risk neutral, i.e., its utility is determined only regarding its expected profits [4], then its utility function is the same as its expected profits, and $U_s(ds, loc) = P_s(ds, loc)$. Otherwise, if s is risk averse, then the uncertainty involved in future queries flows will influence the utility it derives from storing the dataset, so U_s will be risk adjusted in order to consider the element of uncertainty involved in its expected profits [2], i.e., $U_s(ds, loc) < P_s(ds, loc)$. Similarly, if s is risk prone, then $U_s(ds, loc) > P_s(ds, loc)$. In the following sections, unless explicitly written, we assume risk neutral servers.

4 The Trading Mechanism

Bidding sessions are carried on during predefined time periods. When new datasets arrive, they are stored in a temporary buffer until the next bidding session, when their location will be decided upon. Each server is represented by an automated agent, which participates in the bidding session. In the rest of the paper we will use a server and its agent interchangeably.

Each server is responsible for the datasets it stores, and the initial responsibility for each new dataset is determined according to a static policy: the server with the areas of interest closest to a new dataset will be responsible for it.[7] At the beginning of a bidding session, each agent broadcasts an announcement for each new dataset it is responsible for, and also for some of its old local datasets. An announcement of the availability of a dataset by agent $s \in$ SERVERS, denoted as the contractor, indicates that agent s would like to *sell* this dataset.

In the next step of the bidding session, for each announcement and for each agent $s' \in$ SERVERS, such that s' is not the contractor which has made the announcement, s' sends a *sealed bid* to the contractor. A bid contains the price, in standard currency, which s' is willing to pay the contractor in order to store the dataset made available in the announcement in s' (i.e., to *buy* the dataset.) If it does not want to buy this dataset, it will bid a negative price, which indicates how much it would like the contractor to pay it, in order for it to agree to store this dataset locally. All the bids must be sent up until a predefined deadline. The contractor of each dataset collects all the bids related to this dataset up to the deadline.

In the third step of the bidding session, which is called the awarding step, the winner of each announcement is determined by its contractor. For each announcement, the winning agent will be the agent with the highest bid, but the price it pays will be determined according to the second highest bid. The price paid is based on *Vickrey's sealed bidding*, and the bidding of true values is a dominant strategy in this protocol. That is, for each dataset, the best bidding strategy for each agent is to bid a price which is equal to its utility from storing this dataset, and this is independent of the other bids [12].

In the awarding step, each agent will broadcast an award message for all announcements it made in the first step. In this message, it will include the "winner" of the dataset, which is the highest bidder, the price it has to pay, as well as the agent that sent the second-highest bid. If the second bid is less than the utility the contractor derives from storing the dataset of the announcement by itself, or if there is only one bidder (which is a rare event, since all servers are supposed to send a bid for each dataset), then the contractor will continue to store the dataset locally (or obtain it, if the dataset is new).

The agents are assumed to be self-interested, and each one tries to maximize its own utility. In order to do so, it is able to borrow money at the current interest rate r. Thus, the agents do not need any initial budget. In our protocol, there is no need to synchronize the announcements and awarding messages of different datasets. Simultaneous announcement of awards is not necessary, since we assume that the utility from one dataset location is independent of the location of other datasets. The simultaneous bidding is also unnecessary, since as we will show below, the dominant strategy for each bidder is to bid its true value, and this holds also if it has information regarding the other bids.

[7] The initial owner can also be determined by the source of the information, which will direct the dataset to the server with the nearest topics

5 Attributes of the Bidding Protocol

In this section we will describe some of the attributes of the bidding protocol. We will prove that it is a dominant strategy mechanism, and we will present a strategy that enables an agent to choose datasets to announce when the announcement process is costly.

5.1 Details of the Protocol

In the following, we present the costs and concepts related to the bidding process. The function *contractor* specifies for each dataset its current "owner," which will be its contractor during the bidding session. For an old dataset, this is the agent where it is currently stored, and for a new dataset, this is the agent which is responsible for this dataset (as explained in section 4). The function *move_cost* associates with a dataset, ds, and a server the cost for the server $contractor(ds)$ to move the dataset from it to the specified server. If the dataset is new, then it is stored in the temporary buffer, and $contractor(ds)$ has no costs involved in moving it, so *move_cost* is 0. The function *obtain_cost* associates with a dataset and a server, the cost for the server to move a dataset to its location from the server $contractor(dataset)$, if it is an old dataset, or from the temporary buffer otherwise. Relocating datasets is costly both to the sender and the receiver, since we take into consideration the communication time required on both sides in order to reallocate a dataset. But, allocating a dataset in its initial location causes costs only to the buyer, since it receives the dataset from the temporary buffer, but doesn't use the contractor's resources, since it is not the sender in that case. Finally, we denote by *price_suggested(b,ds)* the price suggested by bidder b for dataset ds.

After a contractor announces a dataset, each server sends a bid concerning this dataset. Sending a bid is free (all the costs related to the process of bidding for a dataset are covered by its contractor). Thus, in general, each agent will send a bid for any dataset in the system. The contractor for each dataset collects all the bids related to this dataset until a predefined deadline occurs. Then it has to decide whether to move the dataset, and if so, where to move it to. The function $move(ds)$ will associate "true" with a dataset ds, if the agent $contractor(ds)$ can beneficially move the dataset to another server, given a set of bids, and "false," otherwise. Further discussion on $move(ds)$ appears below. If the contractor decides to move the dataset, it must abide by the following regulations of our bidding protocol. Deviation from this protocol is revealed immediately and yields a penalty. The following attribute defines the winner of a dataset, as determined by the protocol proposed here.

Attribute 5.1

$$winner(ds) = \begin{cases} argmax_{bidder \in SERVERS} & move(ds) = true \\ price\_suggested(bidder, ds) - & \\ move\_cost(ds, bidder)\} & \\ & \\ none & otherwise \end{cases}$$

If there is more than one bidder with the same maximal value of $price\_suggested(bidder, ds) - move\_cost(ds, bidder)$, then the contractor will select one of them (arbitrarily) to be the *winner*, and the other will be considered to be the bidder with the second highest price.

The price paid by the winning agent (if it exists) is the second best bid, w.r.t. the net suggested price. That is, if the dataset is new, then the final price will be precisely the amount of the second best bid, and if the dataset is old and already stored by the contractor, then the final price will be the second best price, deducting the costs of moving the dataset to the bidder of the second best price, but including the costs of moving the dataset to the winner. Our protocol is different from the basic bidding protocol [12], since we include the payments of the relocation costs in the protocol, and thus the desired properties of the bidding mechanism should be proved for this modified protocol.

Attribute 5.2 *The price which will be paid by the winning agent, if such a winner exists, is:*

$$price(ds) = second\_max_{bidder \neq contractor(ds)}\{price\_suggested(bidder, ds) - move\_cost(ds, bidder) \mid bidder \in SERVERS\} + move\_cost(ds, winner)$$

As mentioned above, the contractor must specify the winner, the second price and the agents which offered the second price in its awarding message, and there is a high penalty for revealed lies. It is easy to show that if there is a penalty for revealed lies, then the contractor would follow the regulation above. It will not be motivated to specify a price higher than the second price in its awarding message, since such a lie can be revealed immediately by the agent that is specified by the contractor as the sender of the second price. It would also not be motivated to specify a lower price, since this would never be beneficial for it.

5.2 Bidding Strategies

Given the above regulations in the bidding protocol, and given a set of bids, the contractor will decide whether or not to move an old dataset, i.e., whether $move(ds)$ is true or false. We suggest that the contractor move a dataset if the utility it is able to derive from selling it is more than the utility it can derive from continuing to store the dataset, (if there is only one bidder, then the dataset will not be moved). Note that this is not part of the regulations, but is the best strategy for a self-motivated contractor if it must choose the price and the winner, as described above.

Attribute 5.3 *Situations in which it is beneficial for the contractor to move a dataset, ds, are:*

$$move(ds) = true \text{ if } |bidders| > 1 \text{ and } second\_max\{price\_suggested(bidder, ds) - move\_cost(ds, bidder) \mid bidder \in SERVERS\} \geq U_{contractor(ds)}(ds, contractor(ds)).$$

In the following lemma we state that the winning agent will derive a nonnegative utility from obtaining the dataset.

Lemma 1. *If there is a winner of an announcement, and if it is chosen as specified in attribute 5.1 and is paid price(ds) as specified in attribute 5.2, then if the winner's bid was exactly equal to its utility from obtaining the dataset, it will have a nonnegative utility from "buying" the dataset, deducting the price it should pay.*

Proof. Denote by *winner* the winning agent, by *second* the agent with the second offer, and by *ds* the dataset being considered. Suppose the winner offered exactly the utility it will derive from obtaining the dataset. Then, the utility which the winner will derive obtaining *dataset* is exactly $price\_suggested(winner, ds)$. The price it will have to pay is
$price\_suggested(second, ds) - move\_cost(ds, second) + move\_cost(ds, winner)$.
By definition,
$price\_suggested(second, ds) - move\_cost(ds, second))$
is lower or equal than the chosen
$price\_suggested(winner, ds) - move\_cost(ds, winner))$.
Thus, $price\_suggested(second, ds) - move\_cost(ds, second) +$
$move\_cost(ds, winner) \leq price\_suggested(winner, ds)$. □

Now we will state that each agent's bid will be equal to its utility from obtaining the dataset, for each dataset.

Lemma 2. *In a protocol where the best bidder wins and pays the second price as specified in attributes 5.1 and 5.2, each bidder will bid according to its utility from storing this dataset, deducting the cost of obtaining it. I.e.,*

$$price\_suggested(bidder, ds) = U_{bidder}(ds, bidder) - obtain\_cost(ds, bidder).$$

Bidding the real utility is the dominant strategy, and the proof is similar to that of the Vickrey auction [12]. However, in our case, the winner incurs expenses related to obtaining the dataset, and the contractor incurs expenses related to moving the datasets to the winner (costs of resources needed for the move). Thus, the price paid by the winner is different than the second price, as described in attribute 5.2. Another difference is that, in our case, the contractor itself has its own interests and preferences and has the ability not to move a dataset if the offers it receives are too low, as described in attribute 5.3. These changes cause the proof to be slightly different from the original. Using the above lemmas we have proved the following theorem.

Theorem 3. *If bidding is free, then the allocation reached by the bidding protocol always yields better or equal utility for each server than does using the static policy. The utility function of each server is evaluated according to its expected profits from the allocation.*

In summary, the protocol which we suggest can be implemented in a distributed system in which no central controller exists: its implementation is stable and will ensure satisfactory results. However, even though bidding the true utility of obtaining the dataset is a dominant strategy, it may be beneficial for the first and second bidders to cooperate based on an agreement negotiated between them prior to the bidding, so that the second bidder will bid a lower price, and the first bidder - the winner - will pay a reduced amount. That is, bidding the true utility is not in a strong Nash equilibrium [1], since there may be a subgroup of agents which can gain when they deviate together from the suggested strategies. If communication during the bidding process is forbidden, then the servers will not be able to cooperate. In situations where cooperation may occur, if there is complete information, then should the agents cooperate, the same winner will be chosen per dataset (as in the case of honest reports), so that the bidding process will result in the same allocation, but the gains will be distributed differently among the agents.[8] In situations of incomplete information, the "winner" is not known before the bidding begins, and in order to lower prices cooperation among bidders is not stable, since an agent may agree to bid a low price and then bid a higher one in order to obtain the dataset for itself.

5.3 Estimating Usage

In real-world situations, the agents may have incomplete information about the world. Thus, evaluating their expected profits is problematic. In our environment, we assume that all agents have common knowledge about storage cost and answer cost, but may have asymmetric information about the usage of old datasets and usage of keywords in the past queries, and thus the agents will have an asymmetric and uncertain information about the future usage of datasets. Each agent knows only the past usage of its local datasets, so it will estimate the future usage of these datasets for each geographical area, using its knowledge of the past usage. Estimating the future usage of new datasets and datasets located in remote servers is more difficult, since the agent has no knowledge regarding their past usage. In order to accomplish this, the agent will use information about the datasets' contents. Each dataset is characterized by several keywords, and each query consists of keywords. When a query is handled, the server saves the information about the query, including the keywords which the query contains, i.e., the agent saves the past usage of each area for each keyword and each local dataset.

When an estimation about a new or remote dataset is required, the agent uses the information it has about the keywords' past usage and computes the expected usage of the dataset according to the keywords it is familiar with. The future usage of a new dataset by each geographical area can also be estimated according to the past usage of similar datasets, when their similarity is measured according to the keyword contents.

[8] Note, however, that if there is complete information, a simple protocol which enforces bidding the expected utility can be used.

For simplicity, we assume that the clients form their queries according to keywords and that each query is sent to all the datasets containing that keyword. Under these assumptions, an agent can determine the usage frequency of a new or remote dataset ds for a given area to be the sum of the usage frequency of all the keywords contained in ds. If it does not have data about a keyword which is associated with ds, i.e., no local dataset contains this keyword, then it can use the average usage frequency of all the keywords for the unknown value.

Formally, suppose that there where T time periods before the current bidding session, $key\_usage(area, key)$ indicates the volume of usage of key in different queries of clients located in the area of $server$ in the previous time periods, $dataset\_usage(area, ds)$ indicates the volume of usage of ds by $area$ in the previous time periods, and $exp\_usage_s(area, ds)$ indicates the expectations of agent s about the usage of ds by $area$; then

$$exp\_usage_s(area, ds) = \begin{cases} dataset\_usage(area, ds)/T & \text{contractor(ds)=s} \\ & \text{and is\_old(ds)} \\ \sum_{key \in ds} key\_usage(area, key)/T & otherwise. \end{cases}$$

A more complex learning schema may be considered in order to estimate the future usage of datasets for situations where queries are formed differently, when keywords are related, etc., but we leave this for future work. After estimating the future usage of a dataset, the agent can compute its expected utility from obtaining the dataset, according to the utility function we presented above and w.r.t. the risk involved, as described in Section 3.

5.4 Choosing Datasets

Another issue related to the bidding protocol is how an agent should choose a beneficial set of datasets to announce for bidding. If there is no cost associated with dataset announcement and bidding, then each agent will announce all its datasets, including all its old datasets. However, the announcements can be expensive, due to costs of time and communication. Alternatively, the bidding protocol can limit the number of datasets which an agent can announce, for environments with a large number of datasets. In such cases, each agent will have to select carefully which datasets to announce.

In order to estimate $exp\_announcement\_profit$, which denotes the profits the agent expects to obtain from an announcement, the agent has to estimate which prices it will receive as bids. According to lemma 1, the price which each agent will offer is equal to its utility from storing the dataset, deducting the cost for obtaining the dataset. Thus, in order to decide which dataset to announce, an agent has to estimate the expected utility of the other agents and to compute the expected price it will obtain. The next attribute defines the profit which the agent expects to derive from an announcement. If the profit from moving the dataset is negative, then the contractor will continue to store the dataset, and, in this case, the profit of the announcement is zero. In particular, the expected profit includes the payments which are expected to be obtained for this action

$(expected\_price(ds))$, deducting the utility from continuing to store the dataset and the cost of moving the dataset to the new location.

Attribute 5.4 *The expected profit from announcing a dataset ds:*

$$exp\_announcement\_profit\ (ds)= \\ max\{0, expected\_price(ds) - move\_cost(ds, expected\_winner(ds)) - \\ U_{contractor(ds)}(dataset, contractor(ds))\}.$$

If the contractor is risk-neutral, it will announce only those datasets with an expected profit which exceeds the cost of announcing the dataset. If there is a limit on the number of the announcements, it will announce the datasets with the highest expected profits. If the contractor is risk-averse, then it will consider in its evaluation the risk involved in the expected price and the risk involved in its expected profits from retaining the dataset.

As described above, in order to determine $expected\_price(ds)$ and to be able to compute $exp\_announcement\_profit(ds)$, the agent, s, needs to compute the prices which would be offered by the bidders for ds. The price offered by a bidder $\hat{s}$ depends on its utility function ($U_{\hat{s}}$), which consists of the expected usage of $\hat{s}$ for ds. Thus, in order to evaluate the expected profit of announcing a dataset, the potential contractor needs to estimate the value of $expected\_usage_{\hat{s}}$. If the agent does not have information about $\hat{s}$'s estimation, we propose that it should use its own estimation of the expected usage, which is computed as specified in Section 5.3, as the estimation of $\hat{s}$, i.e., we will assume that $expected\_usage_{\hat{s}} = expected\_usage_s$. If the agent knows which keywords appear in $\hat{s}$'s datasets, it can estimate $expected\_usage_{\hat{s}}$ by using only the keywords that appear both in its own datasets and in $\hat{s}$'s datasets. This may lead to a better estimation of the other agent's beliefs.

5.5 Utility Function -elocation Case

In Section 3 we defined the profits which a server expects to obtain from storing a dataset indefinitely. However, there is a possibility that at some future time, the dataset will be moved to another server. This could happen if its owner announces it, and is offered compensation which is at least equivalent to the expected profit obtained while continuing to store the dataset. In such cases, when evaluating the expected profits of the server for storing a dataset, we have to take into consideration both the costs and benefits associated with this dataset, and the price it expects to receive for this dataset in the future, if it is sold. Formally, the profits that server s expects to derive from obtaining dataset ds, at time t, are as follows:

$$P_s(ds, s, t) = V_s + max\{P_s(ds, s, t+1), \\ expected\_price_{t+1} - move\_cost(ds, expected\_winner(ds)) + \\ P_s(ds, remote \neq s, t+1)\}/(1+r)$$

where r is the interest rate, $expected\_price_{t+1}$ denotes the payments expected to be obtained from selling the dataset at time $t+1$; $move\_cost(ds, expected\_winner(ds))$

denotes the cost of moving the dataset from the winner at time t to $expected\_winner(ds)$; and, $P_s(ds, remote \neq s, t+1)$ specifies the profits the server expects to obtain from not storing the dataset at time $t+1$.

The profits that a server expects to obtain from not storing the dataset at time t are composed of a profit of 0 at the current time period, but it has to take into consideration the possibility of obtaining this dataset in the future. Formally, we state that

$$P_s(ds, remote \neq s, t) = max\{P_s(ds, s, t+1) - expected\_price_{t+1} - obtain\_cost(ds, s), P_s(ds, remote \neq s, t+1)\}/(1+r)$$

However, in general there is only a low probability that the dataset will be sold in the future, and that it will be beneficial for agent s to buy it. Thus, it is reasonable to assume, for simplicity, that the utility for s is 0 from storing a dataset on a remote server.

If the agents believe at time t that in time $t+1$ their expectations of the usage at time $t+1$ will be the same as their current expectations of the usage at time $t+1$, they also believe that no reallocation will be done at time $t+1$, i.e., $U_s(ds, s, t+1) > expected\_price_{t+1} - move\_cost(ds, expected\_winner(ds)) + U_s(ds, remote \neq s, t+1)$ and then we get $U_s(ds, s, t) = V_s + U_s(ds, s, t+1)/(1+r)$. Expanding this formula, we get the $U_s(ds, s, t) = V_s + \frac{V_s}{1+r} + \frac{V_s}{(1+r)^2} + ... + \frac{V_s}{(1+r)^N}$, which is the utility function defined in Section 3. We leave for future research the formulation of the explicit formulas of the general case, where an agent at time t may believe that some of the agents will have a different expectation at time t+1.

6 Experimental Evaluation

In order to test the bidding techniques and compare them with other approaches, we designed and implemented a simulation of our servers' environment. In comparing the performance of the approaches, we used a measurement which excluded the payments of users for their queries and the storage costs, since the total values of these costs do not depend on a specific allocation. So 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 consists of the transportation costs due to the flow of queries. Formally, given an allocation, its variable cost is defined as follows:

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

The actual measurement we use is denoted by $vcost\_ratio$ – the ratio of the variable cost of the bidding mechanism (or another mechanism, as specified below) and the variable cost of the static allocation. The efficiency of the bidding technique increases as the $cost\_ratio$ decreases.

	vcost ratio	CU	CI
static	*	0.22266	*
bidding	0.7375	0.22255	1.0683
optimal	0.6798	0.45966	43.289

Table 1. Bidding in Complete Information Situations

First we tested the bidding mechanism where the agents have complete information about each other and about the environment. In particular, all the agents have the same estimation of future usage of datasets, but are still uncertain about the actual future usage. In such cases, a first bid protocol can be used too, since no server can lie. However, we check the results of the second bid protocol, in order to evaluate the loss of using a second bid protocol w.r.t. using the first bid protocol. We compared three different methods: static allocation, an optimal allocation using a central algorithm which maximizes the sum of all the servers' utilities, and our bidding mechanism.

In Table 1 we present the results of 50 runs of randomly generated environments, with 200 old datasets and 20 new ones, in environments where the relocation of old datasets is seldom beneficial since the size of the datasets is very large. The second column (vcost ratio) in Table 1 specifies the average of *vcost_ratio* of the new datasets and the datasets which were reallocated. This measurement excludes the costs of old datasets not moved in that environment. The third column (CU) indicates the average of the relative dispersion of the utility due to the new datasets and the ones that were moved among the agents (std util/mean util). The last column (CI) specifies the average of the relative dispersion of the added benefit of the new datasets and the old ones that were moved. The variable costs obtained via the bidding mechanism were better than the static policy results, but were not as good as the results obtained by the central optimization algorithm.

We observed that the only case in which the bidding mechanism and the central optimization algorithm located datasets differently were for datasets in which their contractor's utility of storing them locally was higher than its utility of selling them according to the second price, but lower than its utility of selling them according to the first price, causing the contractor to prefer not to sell them. This effect is caused by the use of the second-price bidding. However, the bidding mechanism has an advantage since, in any situation, it guarantees each server a utility which is at least the utility it could obtain via the static policy. We notice that the bidding mechanism yields a lower dispersion of the utility among the agents, w.r.t. the central algorithm. That is, maximizing the sum of utilities by the central algorithm yields a higher dispersion, with some agents unsatisfied with the results. This was prevented, however, by the monetary system of the bidding mechanism, which causes the dispersion of the utilities while using the bidding mechanism to be lower and similar to that of the static allocation.

In the second set of experiments we introduced incomplete information con-

	vcost ratio	CU	CI
static	*	0.218161	*
bidding	0.791596	0.217939	1.56416
centralized	0.68908	0.441369	41.1896
optimal	0.685007	0.45565	42.3611

Table 2. Bidding in Incomplete Information Situation

cerning the future usage of the datasets, (i.e., the agents didn't know the mean usage by clients in each area of each dataset) and asymmetric information about past usage. To simulate such situations, we implemented a system in which queries are sent according to keyword frequency to the servers. First, the mean usage of each keyword by each geographical area is randomly generated. Then the queries generator sends queries, such that the number of queries concerning a given keyword sent from a specified geographical area is generated using Poisson distribution, with the specified mean usage. Each server receives queries related to its own datasets and maintains the statistical information for estimating the future use. However, it has no knowledge about the queries which were sent to the other servers.

Before the bidding process starts, each agent estimates the usage frequency of the old and the new datasets, as described in section 5.3. In particular, each agent knows the keywords of the datasets located in the other servers. Based on the queries sent to it w.r.t. these keywords, the agent estimates the usage of the other agents. Thus, as the number of keywords in the system increases while the number of datasets is kept fixed, each server has less information about the usage frequency of datasets (since there are keywords that the agent does not have in its datasets) and incomplete information in the system increases.

In Table 2 we present the results obtained from a simulation of 50 randomly created environments where there was some incomplete information in the system (the mean error of the expectations was 13%). We compared the results obtained by the static allocation, the bidding allocation, the centralized allocation which is obtained when maximizing the sum of servers' utilities using all the information stored by all the servers, and the optimal allocation found by a centralized algorithm which also maximizes the sum of servers utilities but has the real usage frequency. We see that the bidding allocation succeeds in reducing the average variable costs of a server, although there is a gap between its performance and the performance of the centralized allocations. This gap was caused since each server had only partial information about the future usage of datasets. However, the bidding mechanism obtained a much lower standard deviation than the centralized alternatives. We also carried out a set of simulations to test the effect of the amount of incomplete information in the system on the bidding performance, by varying the number of keywords while keeping the number of datasets fixed. As would be expected, we found that *vcost_ratio* decreases (i.e., the level of improvement w.r.t. the static allocation increases) as

there is more information.

7 Conclusion

This paper presents a bidding protocol for the data allocation problem in multi-agent environments. For complete information, we have formally proved that bidding yields efficient and fair results. For situations in which the agents have incomplete information, we ran simulations, and the results of the bidding approach were, on the average, better than those of the static policy.

In environments in which each server cares about the exact location of each dataset, even when such a location is remote, a bidding protocol is not beneficial, since the server's agent cannot influence the location of such a dataset. For such environments, we suggest elsewhere [10] using the strategic model of alternating offers as a solution method, enabling each agent to influence the decision of the exact location of each dataset, even without storing it locally. We showed, however, that bidding is better than strategic negotiations in the environments considered in this paper.

References

1. B.D. Bernheim, B. Peleg, and M.D. Whinston. Coalition-Proof Nash Equilibria I: Concepts. *J. of Economic Theory*, 42, 1:1–12, 1987.
2. T. E. Copeland and J. F. Weston. *Financial Theory and Corporate Policy*. Addison-Wesley publishing company, 1992.
3. X. Du and Fred J. Maryanski. Data allocation in a dynamically reconfigurable environment. In *Proc. of the IEEE Fourth Int. Conf. Data Engineering*, pages 74–81, Los Angeles, 1988.
4. S. French. *Decision Theory: An Introduction to the Mathematics of Rationality*. Ellis Horwood Limited, 1986.
5. D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, Cambridge, Ma, 1991.
6. T. Mullen and M. Wellman. A simple computational market for network information services. In *Proc. of the First International Conference on Multiagent Systems*, pages 283–289, California, USA, 1995.
7. NASA. EOSDIS Home Page. http://www-v0ims.gsfc.nasa.gov/v0ims/index.html, 1997.
8. J. S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers*. MIT Press, Boston, 1994.
9. T. Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proc. of AAAI-93*, pages 256–262, 1993.
10. R. Schwartz and S. Kraus. Negotiation on data allocation in multi-agent environments. In *Proc. of AAAI-97*, Providence, Rhode Island, 1997. (to appear).
11. R.G. Smith and R. Davis. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20:63–109, 1983.
12. William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *J. of Finance*, 16:8–37, 1961.

This article was processed using the LaTeX macro package with LLNCS style