Iterative Query-Based Approach to Efficient Task Decomposition and Resource Allocation

Michal Pěchouček[1], Ondřej Lerch[2], and Jiří Bíba[1]

[1] Department of Cybernetics, Faculty of Electrical Engineering
Czech Technical University in Prague
Technická 2, Prague, 166 27, The Czech Republic
{pechouc,biba}@labe.felk.cvut.cz
[2] Department of Software Engineering, Faculty of Nuclear Engineering
Czech Technical University in Prague
Břehová 7, Prague, 115 19, The Czech Republic
ondrej.lerch@gmail.com

Abstract. Intelligent coordination in complex multi-agent environments requires sophisticated mechanisms for suboptimal *task decomposition* and efficient *resource allocation* provided by the the agents. Besides the quality of coordination (i.e. efficiency of decomposition and resource allocation) we need to handle also computational efficiency restriction such as fast response time and limited communication traffic among the agents as well as optimization of the amount of private knowledge disclosure, during collaboration patterns negotiation among the semi-collaborative agents. We present a novel contracting mechanism based on the use of the approximated acquaintance model, a structure where the agents store the information about the states, capabilities and resources of possible collaborators. We suggest an approach of iterative construction of the partially-linear acquaintance models that is beneficial mainly in complex agent communities.

1 Introduction

This paper presents a novel contracting mechanism based on the use of the *acquaintance model*, a knowledge structure representing agents' mutual awareness. An appropriate use of the acquaintance models reduces the communication traffic requirements and reduces the computational requirements on agent's decision making. This approach provides the most significant benefits in system with larger amount of agents (tens to hundred of agents) and with very high number of operational alternatives (i.e. ways how a single request can be handled). Besides operational efficiency improvements, the use of acquaintance models also provides an elegant mechanism for avoiding private knowledge disclosure.

The presented approach has got its applicability potential in the logistics and supply chain management domains, where complex business interaction needs to be optimized. Especially the knowledge disclosure aspect is very important in this domain. The acquaintance model based contracting has been deployed in the OOTW (operations other then war) humanitarian relief provision scenarios [1]. Besides, this efficient coordination mechanism can be used in manufacturing domains (illustrated in [2]), resource

leverage in larger (possibly ad-hoc) networking environment [3], telecommunication, and others.

We will be working with a community of agents where each agent can act as a **service provider** (agent that provide some service or product) or a **service requester** (agent that request services or products). The algorithm will be illustrated on a single contract case, where there is only one requester and n providers. However, the algorithm is designed so that each agent can be a provider and a requester at the same time and the agents can also contract each other (similar to [4]). The problem is to find best decomposition of a specific task into granular services and the most optimal contracting among the community of possible service providers.

As mentioned earlier, the presented problem integrates two deliberation activities (i) *decomposition* and (ii) *contracting* (delegation). The main difficulty is that efficiency of decomposition depends on how well the subtasks can be contracted, while contracting depends on how the task is decomposed. Both deliberation processes are deeply interlinked.

1.1 Existing Contracting Mechanisms

Contracting is one of the most important problems that has been studied in the field of multi-agent system. The most widely used approaches to contracting are based on the *contract-net-protocol* [5]. Here the requester broadcasts the call for proposals within a community of potential providers. The providers reply with a collaboration bids. The requester selects the most optimal bid and sets a contract. The contract-net-protocol can be also multi-staged, that is understood as an auction. There are several auctioning mechanisms available in the community: *English auction*, *Dutch auction*, *seal-bid auction*, *Vickery auction* [4].

The problem with the classical contract-net-protocol is that it may get stuck in local optimum. This happens primarily if we are trying to balance a load in a nontrivial network of n requester and n providers. That is why several variants of the original O-contract-net-protocol have been suggested. C-contract-net-protocol works similarly to the classical protocol while the subject of negotiation is not a single task but a collection of tasks. The S-contract-net-protocol works with agents who are not selling and buying tasks but they swap the tasks between two agents instead. Similarly, the M-contract-net-protocol arrange swapping among several (more than 2) agents. It has been proved that with a finite number of agents and tasks, the $OSCM$-contract-net-protocol protocol that combines all these variants can always find a globally optimal solution in a finite number of steps [6].

While the concept of delegation, task decomposition and commitments have been also widely studied in the multi-agent community [7], [8] [9] our focus will be mainly centered around linking the contracting and decomposition activities within the community of collaborating agents.

1.2 Acquaintance Models

As mentioned previously, use of the acquaintance models is going to be a central concept of the presented contracting mechanism. The *acquaintance model* is computational

model of agents' mutual awareness, built by each of the agents. The acquaintance model is a collection of agent's social knowledge [10] available from previous interaction or provided by independent monitoring mechanisms. The acquaintance model may contain both the inevitable information for setting any kind of collaboration (such as the *white-page* information – IP addresses, ACL, port-number) and an optional information that can improve the quality of collaboration (such as information about the services available, free capacities, overall load of the agents and others). The acquaintance model can also contain non-public information such as trust models or models of the other agent's internal mental states (e.g. commitments, intentions).

There have been several specific architectures of the acquaintance model designed in the past – *tri-base* (3bA) acquaintance model [11], *twin-based model* [12], acquaintance model in ARCHON [13]. In the remaining parts of the article the specific architecture of the acquaintance model is unimportant, as we will be discussing how the knowledge is maintained and exploited on an abstract level. However, there is one clear distinction between the existing acquaintance models and the abstract acquaintance model we will be using in our experiments. Unlike the listed models, we will be using the concept of acquaintance models for representing an *approximate social knowledge*. Such knowledge is an estimated information about the other agents and may be constructed by various approximation mechanisms and limited interaction among the agents.

1.3 Task Decomposition and Resource Allocation Approaches

A task decomposition and a resource allocation is one of the most crucial points of inter-agent cooperation. There have been proposed various approaches based on concepts like market mechanisms, task/resource clustering, load balancing, etc. Both the problems of the task decomposition and the resource allocation are often substantially mutually dependent and generally belong to class of NP-complete or even NP-hard tasks. This is caused by complex dependences between subtasks that particular tasks consist of (e.g. precedence constraints) as well as by a need for sharing different special resources among more tasks.

Interesting results are obtained by means of combining the agent approach with non-deterministic optimization approaches like genetic algorithms in order to obtain solutions of complex problems in a reasonable time and quality. An approach combining the agent paradigm with genetic algorithms (GA) was presented in [14]. Resource allocation in a computational grid is done in three steps: (i) rough task clustering negotiated by the agents "possessing" (representing) particular resources – the tasks are clustered so that tasks with a more intensive messaging are assigned to nodes inter-connected by means of better communication lines , (ii) optimal mapping of task clusters to resource clusters by means of GA and (iii) recursive distribution of the task clusters to appropriate resource clusters. The tasks could require specific resources and were assumed to be inter-connected.

Another approach exploiting hierarchical grid topology and execution time prediction was presented in [15]. In a task farming problem (i.e. several independent tasks are to be executed in parallel so that the execution time is minimized) an execution time prediction is used to choose resources/nodes for their execution and the job scheduling is done by means of GA. The tasks may be generally of different kinds (i.e. re-

quire different types of resources/services). When considering only one-service tasks, the scheduling is done by means of a Roulette Wheel Selection according to predicted execution times of the jobs at particular nodes.

An agent-oriented solution exploiting market mechanisms was introduced in [16]. The system TRACE consists of multi-agent organizations that are allowed to allocate dynamically task and resources in order to efficiently process incoming stream of tasks. Each agent may be assigned only a task that is capable to handle (i.e. the agent is equipped with resources/abilities required for the particular task). There are distinguished permanent and marketable agents and each organisation has own resource manager agent. The tasks arrive arbitrarily to permanent agents who become responsible for the tasks. If an agent determines that the task cannot be proceeded within a given deadline, the task is decommitted within the organization. The resource manager collects an information about decommitted tasks and handles renting or offering marketable agents from or to other organizations. The decommitted tasks are then sent to permanent agents again. The determination of need for marketable agents is carried out by means of a market mechanism. Together with the decommitted task the resource manager is paid a decommitment penalty by the permanent agent. The higher is the contribution, the higher priority has the decommitted task. The resource manager calculates an equilibrium task allocation and participates in a market competition for available marketable agents. As soon as the market approaches an equilibrium (number of all offered marketable agents at given price is equal to number of required ones), it hires the agents and provides them with a necessary amount of domain information concerning the tasks they are hired for.

Another market mechanism was presented in [17]. There was provided an explicit formal description of an agent utility which differs in the subject of minimization when allocating the tasks on the resources, i.e. either (i) the total-duration or (ii) the total-price. While the one objective function is minimized the other is considered as a constraint. The *grid task agents* compete for resources handled by *grid resource agents*. The grid resource agents aim to maximize their utility as well as the grid task agents do – thus the grid resource agents update the prices with respect to a demand and the grid task agents offer prices with respect to demanded amount of resources and available budget. The bargaining process is finished when an equilibrium between demanded and offered resources is achieved. The negotiation between grid task agents and grid resource agents is held indirectly by means of a grid market that gathers information of all the actual negotiations (a global market view) and provides it to negotiating parties while multiple independent negotiations are facilitated.

In special cases the task decomposition and resource allocation problems may be solved polynomially or pseudo-polynomially provided the properties and requirements of the tasks are restricted. A simplification consists e.g. in reducing the number of available resources (e.g. scheduling only for two dedicated processors) or in relaxation of task inter-dependences (e.g. there are missing precedence constraints among the subtasks). An interesting simplification consists in reducing the number of resource types required to an only type while multiple resources are available for processing. In this paper we focus on an ”as-soon-as-possible” task decomposition and resource allocation of tasks requiring only one type of resource/service while minimizing the total duration.

2 Problem Definition

In this section we will formally define the coupled problem of decomposition and contracting. We will denote R as a requester agent, S as a type of service that requester R request in order to complete its task, a_t as a total amount of service S that requester R requests, n as a number of providers offering the service type S, P_j as a provider agent offering service S, where $j = \{1, ..., n\}$ and $d_j(a)$ as a duration for the agent P_j to deliver the amount a of service S.

The **decomposition** of the service S in amount a_t to agents $P_1, \ldots, P_n$ is an arbitrary vector of non-negative integers $(a_1, \ldots, a_n)$, where

$$a_j \in \mathbb{Z}^+, \quad \sum_{j=1}^{n} a_j = a_t. \tag{1}$$

If a_j equals 0, then the provider P_j is not contracted at all.

The **overall duration** of decomposition $(a_1, \ldots, a_n)$ is defined as

$$d(a_1, \ldots, a_n) = \max_j d_j(a_j) \quad j \in \{1, \ldots, n\}, a_j > 0. \tag{2}$$

The explanation of (2) is as follows: if the requester agent contracts the service S in total amount a_t using the decomposition $(a_1, \ldots, a_n)$, then it means that each agent $P_j \in \{P_1, \ldots, P_n\}$ has to perform S in amount a_j. The whole task is therefore performed when the last agent performs its sub-task. Those agents for which $a_j = 0$ are not contracted at all, and therefore duration $d_j(0)$ is not taken into account.

Our primary effort is to find **optimal decomposition** that minimizes (2), that is to find the vector $(a_1^{min}, \ldots, a_n^{min})$ such that

$$(a_1^{min}, \ldots, a_n^{min}) = \arg\min_{a_1,\ldots,a_n} d(a_1, \ldots, a_n), \tag{3}$$

where $(a_1, \ldots, a_n)$ fulfill (1).

In this work, we focus on the task decomposition to minimize the overall **duration** (2). Another important task is to decompose the task to minimize the overall **price**. Let us assume that the price of service S in amount a equals $p_j(a)$ if the task is performed by the provider P_j.The overall price of decomposition $(a_1, \ldots, a_n)$ is defined as

$$p(a_1, \ldots, a_n) = \sum_j p_j(a_j) \quad j \in \{1, \ldots, n\}, a_j > 0. \tag{4}$$

The difference between the overall duration (2) and overall price (4) is that the overall duration is the maximum of durations $d_1(a_1), \ldots, d_n(a_n)$ while the overall price is the sum of prices $p_1(a_1), \ldots, p_n(a_n)$. In our work, we focused on overall duration minimization since in the real-world scenario, the lowest price decompositions are $(0, \ldots, 0, a_j = a_t, 0, \ldots, 0)$ where P_j is the cheapest provider. This is caused by the fact that providers offer better price per 1 amount if contracted for more amounts.

In this work we consider an only *type* of service to be decomposed (in contrast to decomposing a bunch of different services). While the choice of the overall price as the

only objective function makes the decomposition rather trivial, the choice of the overall duration remains still reasonable in a case of need for meeting a deadline (it is necessary to get the service as soon as possible and the price is less important).

Though, in the real-world scenarios the price is hardly completely ignored – more likely it is set as an optimization constraint and a **constrained decomposition problem** is solved: we minimize the duration $d(a_1, \ldots, a_n)$ under the condition that the price $p(a_1, \ldots, a_n)$ is smaller or equal than predefined constant p_{max}. An even more probable constrained decomposition problem is introduced by the vice versa setting: a deadline is set as the constraint and the subject of optimization is the price: we minimize the price $p(a_1, \ldots, a_n)$ under the condition that the duration $d(a_1, \ldots, a_n)$ is smaller or equal than the predefined constant d_{max}. Finally, an interesting problem is also **combined decomposition problem** that takes into account both decomposition duration and price and uses a joint minimization criterion

$$f_\alpha(a_1, \ldots, a_n) = \alpha \cdot d(a_1, \ldots, a_n) + (1 - \alpha) \cdot p(a_1, \ldots, a_n),$$

$0 \leq \alpha \leq 1$, which includes both the duration and the price. The coefficient α determines if we prefer the minimization of duration or price.

However, in our work, we actually do not cover neither the combined decomposition problem nor the constrained decomposition problem and minimize the overall duration only (i.e. we cover cases when it is necessary to obtain the service as soon as possible).

3 Solution

We have designed a straightforward decomposition mechanism that finds the most optimal decomposition given the right objective function and available data [18]. The decomposition algorithm is polynomial and easy to construct. Its behavior, however, depends strongly on the data stored in the acquaintance models of the agents. In the following we will discuss the decomposition algorithms and two approaches how the acquaintance model can be constructed and maintained – *Batch-Query AM Construction* and *Iterative-Query AM Construction*. The decomposition algorithm and acquaintance model construction are demonstrated on a simple scenario of decomposition a service between two providers (see section 4).

3.1 Decomposition algorithm

For purposes of this paper, we introduce a decomposition algorithm derived from the algorithm presented in [18], however, with significantly smaller complexity. We assume that durations $d_j(a)$ can only take discrete values, that is $d_j(a) \in \mathbb{Z}^+$, and that the functions $d_j(a)$ are extended to the point $a = 0$ by $d_j(0) = 0$. We can now construct pseudo-inverse function $a_j(d)$ as

$$a_j(d) = \max \{a \in \{0, \ldots, a_t\} \mid d_j(a) \leq d\}, \; a \in \mathbb{Z}^+. \tag{5}$$

The meaning of (5) is as follows: if $a_j(d) = \tilde{a}$ then the provider P_j is able to produce $\tilde{a}$ pieces of S in the duration $\tilde{d} \leq d$, but he is not able to produce more pieces than $\tilde{a}$.

Function a_j is queried pseudo-inverse to d_j since if d_j is injective, then $a_j(d_j(a)) = a$. Let us point out that we do not assume so far that functions d_j are increasing.

Now in the time d, provider P_j produces service S in the amount of $a_j(d)$, therefore all providers can produce amount of

$$a(d) = \sum_{j=1}^{n} a_j(d). \tag{6}$$

If we want to produce service S in the amount $a \geq a_t$ in the shortest possible duration $d = d_{min}$, we simply search for a decomposition $(a_1(d), \ldots, a_n(d))$ which provides the smallest d such that $a(d) \geq a_t$, that is

$$d_{min} = \min \{d \in \{0, 1, 2, \ldots\} \mid a(d) \geq a_t\}. \tag{7}$$

For simplicity, let the amount provided by the provider j is denoted as

$$a_j^{min} = a_j(d_{min}) \qquad j = \{1, \ldots, n\}, \tag{8}$$

i.e. $(a_1^{min}, \ldots, a_n^{min}) = (a_1(d_{min}), \ldots, a_n(d_{min}))$ and the overall provided amount is denoted

$$a_{min} = a(d_{min}). \tag{9}$$

Then the found n-tuple $(a_1^{min}, \ldots, a_n^{min})$ is the optimal decomposition of service S in amount $a_{min} \geq a_t$. Let us prove this simple assertion. First of all, $\sum_{j=1}^{n} a_j^{min}$ has to be equal to a_{min} which holds according to (6), (8) and (9). Second of all, $(a_1^{min}, \ldots, a_n^{min})$ is really optimal for amount $a_{min} \geq a_t$. In any duration $\tilde{d} < d_{min}$ all providers cannot altogether produce $\tilde{a} \geq a_{min}$ amounts as d_{min} is chosen according to (7). Therefore a_t pieces of service S cannot be produced in duration smaller than d_{min}[3].

The problem is that a_{min} may be generally greater than the desired amount a_t. Let us make further assumption that functions $d_j(a)$ are increasing, that is

$$d_j(a) \leq d_j(\tilde{a}) \qquad a < \tilde{a}.$$

According to previous algorithm, for a given a_t we can construct a decomposition $(a_1^{min}, \ldots, a_n^{min})$, which is optimal decomposition of amount $a_{min} \geq a_t$. If $a_{min} > a_t$, we may decrease amounts $a_1^{min}, \ldots, a_n^{min}$ arbitrarily till the decreased amounts $\hat{a}_1, \ldots \hat{a}_n$ fulfill $\sum_{j=1}^{n} \hat{a}_j = a_t$. Of course, $(\hat{a}_1, \ldots \hat{a}_n)$ is the decomposition for a_t, but assuming that functions $d_j(a)$ are increasing, it is also optimal. Explanation is as follows: according to the construction of $(a_1^{min}, \ldots, a_n^{min})$, we cannot produce $\tilde{a} \geq a_t$ pieces of S in the duration $\tilde{d} < d_{min}$. As $\hat{a}_j \leq a_j^{min}$, then also $d(\hat{a}_j) \leq d(a_j^{min})$ and therefore $d(\hat{a}_1, \ldots, \hat{a}_n) \leq d(a_1^{min}, \ldots, a_n^{min})$, which proves that $(\hat{a}_1, \ldots, \hat{a}_n)$ is the optimal decomposition for amount a_t. Duration $d(\hat{a}_1, \ldots, \hat{a}_n)$ cannot in fact be smaller than duration $d(a_1^{min}, \ldots, a_n^{min})$, as it would be in contradiction to the construction of $(a_1^{min}, \ldots, a_n^{min})$, therefore $d(\hat{a}_1, \ldots, \hat{a}_n) = d(a_1^{min}, \ldots, a_n^{min})$.

[3] note that both the amounts and durations are considered to be non-negative integers

Let us summarize the algorithm that finds optimal decomposition $(\hat{a}_1, \ldots, \hat{a}_n)$ of service S in amount a_t. We assume that there are n agents $P_1, \ldots, P_n$ that perform S and that the requester knows durations $d_j(a)$ for all amounts $a \in \{1, \ldots, a_t\}$ and all agents $P_1, \ldots, P_n$. We also assume that the functions $d_j(a)$ are increasing. The algorithm works as follows:

1. We gradually take $d = \{0, 1, 2, \ldots\}$ and calculate $a(d)$ according to (6) till we find $d = d_{min}$ such that $a(d) \geq a_t$. Such smallest d we denote d_{min} and corresponding amount $a(d_{min})$ we denote a_{min}.
2. We put $a_j^{min} = a_j(d_{min})$, if $a_{min} = \sum_{j=1}^{n} a_j^{min}$ is already equal to a_t, then $(a_1^{min}, \ldots, a_n^{min})$ is the optimal decomposition for a_t. Otherwise, we decrease amounts $(a_1^{min}, \ldots, a_n^{min})$ arbitrarily till the decreased amounts $(\hat{a}_1, \ldots, \hat{a}_n)$ fulfill $\sum_{j=1}^{n} \hat{a}_j = a_t$.
3. Amounts $(\hat{a}_1, \ldots, \hat{a}_n)$ create optimal decomposition of S in amount a_t for providers $P_1, \ldots, P_n$.

The operation of the decomposition algorithm is based on an assumption that the objective function (i.e. delivery times) $d(a_j)$ are known for all the possible amounts of services a_j. The main argument of this article is that these data are not available in the complex or semi-trusted domains. Instead of working with a data-structure representing all possible $d(a_j)$, we will work with approximated knowledge stored in the acquaintance models.

In the following we present two ways how to construct, maintain and use an approximated acquaintance model.

3.2 Batch-Query AM Construction

The simplest approach to building a well informed acquaintance model representing the agent's services is to send several query messages asking '*what if I wanted this amount of that service*':

```
(query
  :requester R
  :provider P
  :content (deadline S a^(k) d(a^(k)) ))
```

where $a^{(k)}$ are amounts corresponding to an uniform division of space of the total required amount a_t into N different amounts $a^{(1)}, \ldots, a^{(N)}$ while $N \in \mathbb{N}$ and to find out about the corresponding delivery times for these amounts. The remaining part of the acquaintance model would be approximated by partially linear function. It is easy to see that the quality of such model would strongly depend on N, the number of query messages sent between the provider and requester.

The batch-query acquaintance model construction algorithm where the requester wants to find an optimal decomposition of service S in the amount a_t works in the following steps:

1. Requester agent finds all agents that provide service S, let us suppose that those agents are $P_1, \ldots, P_n$.
2. Requester agent queries all providers $P_1, \ldots, P_n$ for amounts $a^{(1)}, \ldots, a^{(N)}$, where $a^{(k)}$ is defined by

$$a^{(1)} = 1, \quad a^{(k)} = \lfloor \frac{(k-1) \cdot a_t}{N-1} \rfloor, \tag{10}$$

   where $k \in \{2, \ldots, N\}$ and $N \in \mathbb{N}$ is a predefined constant equal for all providers
3. Requester agent collects all responses from the providers. This allows us to define approximate durations $\tilde{d}_j(a)$, approximate amounts $\tilde{a}_j(d)$ and the approximate total amount $\tilde{a}(d)$. In our case, we approximate the function $d_j(a)$ by partially linear function $\tilde{d}_j(a)$ according to

$$\tilde{d}_j(a) = l(a|a^{(1)}, \ldots, a^{(N)}, d_j(a^{(1)}), \ldots, d_j(a^{(N)})). \tag{11}$$

$$\tilde{a}_j(d) = \max\{a \in \{0, \ldots, a_t\} \mid \tilde{d}_j(a) \leq d\}. \tag{12}$$

$$\tilde{a}(d) = \sum_{j=1}^{n} \tilde{a}_j(d), \tag{13}$$

   provided that function $l(x)$ produces partially liberalization function with parameters $x_1, \ldots, x_k, y_1, \ldots, y_k$ and is defined as

$$l(x|x_1, \ldots, x_k, y_1, \ldots, y_k) = \frac{y_{j+1} - y_j}{x_{j+1} - x_j} \cdot (x - x_j) + y_j. \tag{14}$$

   for $x_j \leq x \leq x_{j+1}$ and $l_0(x) = l(x)$ for $x \neq 0$ and $l_0(x) = 0$ for $x = 0$.
4. We gradually take $d = \{0, 1, 2, \ldots\}$ and calculate $\tilde{a}(d)$ according to (13) till we find $d = \tilde{d}_{min}$ such that $\tilde{a}(d) \geq a_t$.

$$\tilde{d}_{min} = \min\{d \in \{0, 1, 2, \ldots\} \mid \tilde{a}(d) \geq a_t\}.$$

   Such smallest d we denote $\tilde{d}_{min}$ and corresponding amount $\tilde{a}(\tilde{d}_{min})$ we denote $\tilde{a}_{min}$.
5. We put $\tilde{a}_j^{min} = \tilde{a}_j(\tilde{d}_{min})$. We decrease amounts $(\tilde{a}_1^{min}, \ldots, \tilde{a}_n^{min})$ arbitrarily till the decreased amounts $(\hat{a}_1, \ldots, \hat{a}_n)$ fulfill $\sum_{j=1}^{n} \hat{a}_j = a_t$.
6. Amounts $(\hat{a}_1, \ldots, \hat{a}_n)$ create decomposition found by the algorithm that approximates optimal decomposition of service S in amount a_t for requesters $P_1, \ldots, P_n$.

3.3 Iterative-Query AM Construction

The above presented algorithm has proved to be efficient and pragmatic solution for our contracting/decomposition problem. The key difficulty is that its behavior is parameterized by the constant N determining the amount of messages sent in the acquaintance model construction phase. As there is no a priori knowledge about the distribution of the providers' services[4] $d_j(a)$, the appropriate granularity (and hence the parameter N)

[4] by when is which amount of is service available

is unknown before the requests are broad-casted. This property makes the algorithm rather inflexible.

If N is reasonably high the model is very precise, while it also represents substantial amount of unneeded information. If the implementation does not allow parallel querying, this algorithm may also increase the communication traffic substantially. Not only communication traffic matters. We may also assume that every piece of information that provider discloses has to be payed for by the requester (this assumption reflects the fact that a high amount of information disclosure is likely to be just unwanted by the provider).

This is why we suggest a novel approach to building the acquaintance model based on flexible approximation of the available information. Only one specific amount of the requested service is queried before contracting is initiated. This value is used for very imprecise approximation that is encoded in the acquaintance model. Based on this imprecise model decomposition is computed and appropriate providers are queried for availability their resources. If the bids provided by the providers are close enough to the values approximated by the acquaintance model, the providers are contracted accordingly. If the bids are different then expected, the information provided in the bid is used for refinement of the acquaintance model. New decomposition is computed again and all the process is repeated until the bids are close to the values in the acquaintance model. See below for more detailed specification of the iterative-query acquaintance model construction algorithm:

1. Let us set N equal to 2 and query all providers $P_1, \ldots, P_n$ for $a = \{1, a_t\}$, where a_t is desired total amount. We therefore estimate the real distributions $d_j(a)$ by linear distributions $\tilde{d}_j(a) = \alpha_j a + \beta_j$ which match with $d_j(a)$ for $a = 1, a_t$.
2. For those approximations we find optimal decomposition $a_1^{min}, \ldots, a_n^{min}$ and set $a_1^{(s)}, \ldots, a_n^{(s)}$ equal to $a_1^{min}, \ldots, a_n^{min}$.
3. For the amounts $a_j^{(s)}$ the requester broadcasts appropriate queries to the providers and collects the replies with the values $d_j(a_j^{(s)})$.
4. Provided that $|d_j(a_j^{(s)}) - d_j^{(e)}(a_j^{(s)})| \leq \Delta$, the acquaintance model is regarded as precise enough for the specific contract and the decomposed amounts $(a_1^{(s)}, \ldots, a_n^{(s)})$ can be contracted[5]. The algorithm terminates here.
5. If the algorithm did not terminate in the step 4, the requester inserts new $d_j(a_j^{(s)})$ value into its acquaintance model and carries out new linear approximation and return to step 2.

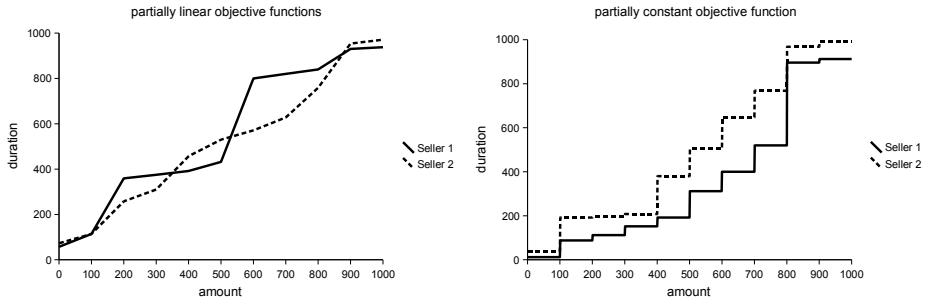
4 Experiments

Properties and efficiency of the listed contracting mechanisms have been empirically tested on a high number of experimental settings. In the following we will show how precise gets the requester's acquaintance model with an increasing number of query messages sent to providers. In other words, how much of providers social knowledge

[5] Δ is an a priori defined error of the admissible acquaintance model precision.

needs to be disclosed for a specific quality of this acquaintance model. We will compare the batch-query AM construction with iterative-query AM construction in a simple scenario where a service requester is building the acquaintance models of two different service providers. We will show how difficult is for the requester to construct the acquaintance model with:

– **piecewise linear objective function**: a real provider's resource availability is represented by a piecewise linear function – such a function may represent duration of processing a certain amount of data in dependence on the tasks already scheduled at a specific node of a computational grid – let say that the tangent of the objective function is given by the computational load of the node in time – thus during the time when the node is less loaded the same amount of data is processed quicker than during a time when the node is more loaded (see Figure 1a) and
– **piecewise constant objective function**: a real provider's resource availability is represented by a piecewise constant function – such a function can represent e.g. total delivery time of a specific order that may be transportees in a number of batches (trucks) and delivery time is identical for several different volumes of the cargo (see Figure 1b).

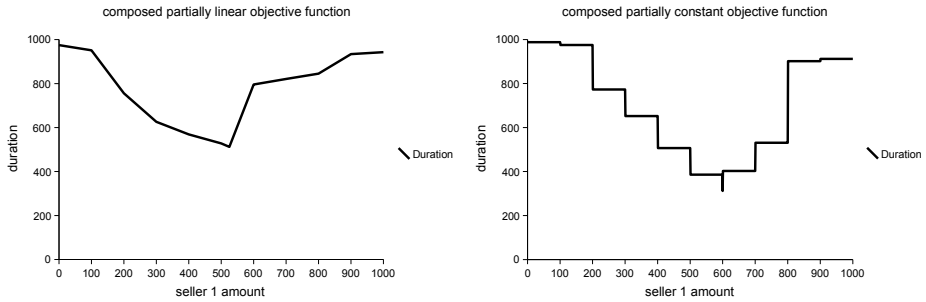


(a) (b)

Fig. 1. (a) Piecewise *linear* objective function — (b) Piecewise *constant* objective function

Let the decomposition and acquaintance model construction be demonstrated on the following simple scenario: let a requester requires 1000 items to be jointly delivered from two providers – `seller1` and `seller2`, whose objective functions are given in Figure 1a. The requester thus needs to approximate in its acquaintance model the functions $\texttt{seller1}(amount)$ and $\texttt{seller2}(amount)$. According to (2) the function to be minimized (i.e. total duration) is given by $\max(\texttt{seller1}(amount), \texttt{seller2}(1000 - amount))$ – see Figure 2a.

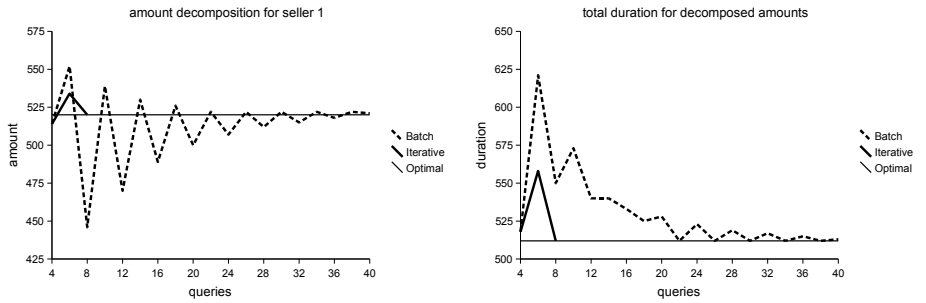
If this function is supposed to be minimized the optimal solution would be to contract `seller1` for 520 units and `seller2` for 480 units. This contract can be delivered within 512 time units. In Figure 3a the straight solid line shows the optimum decomposition. The thick dashed line gives the solution suggested by the batch-query AM



(a) (b)

Fig. 2. (a) Composed piecewise *linear* objective function — (b) Composed piecewise *constant* objective function

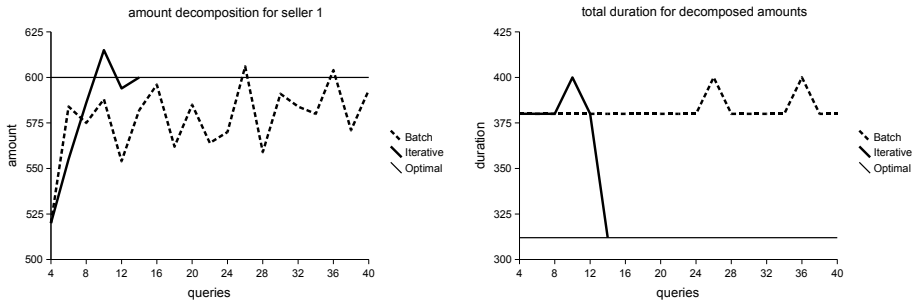
construction mechanism. In the horizontal axis there is N – the number of queries sent to the providers. It is seen that the batch-query AM construction mechanism provides results close to optimum with N around the value 40. The thick solid line gives the solution suggested by the iterative-query AM construction mechanism. The iterative-query AM construction mechanism provides optimal solution after 8 iteration of the algorithm. Similar argument is demonstrated on the graph in Figure 3b. Here the acquaintance model provides an estimation of the delivery due time of the optimal contract as defined above.



(a) (b)

Fig. 3. Approximation of piecewise *linear* objective function in acquaintance model: (a) decomposition — (b) delivery time

The optimal delivery time 512 unites is estimated by the batch-query AM constructed model after sending 40 queries (i.e. 20 iterations – in each iteration queries are sent to both the providers) while iterative-query AM construction mechanism requires only 8 queries.



(a) (b)

Fig. 4. Approximation of piecewise *constant* objective function in acquaintance model: (a) decomposition — (b) delivery time

An interesting result has been obtained when working with the piecewise constant functions. Here the optimal decomposition - `seller1` providing 600 items and `seller2` 400 items – is hard to detect by the batch-query algorithms. The graph in Figure 2b (that is again a composition of the partially constant objective functions of the two sellers) shows that this optimal solution lies on the specific pike.

The graphs on Figures 4a and 4b show that batch-query algorithms have not find an optimal decomposition even after 40 queries and delivery due date has been estimated 380, while the optimal delivery is 312 units. The iterative-query algorithm managed to find the optimal solution by means of 14 queries.

5 Conclusions and Future Work

This partially theoretical and in parts experimental paper introduces a novel mechanism for efficient task decomposition and subcontracting in non-trivial communities of agents. The key novel idea is centered around an assumption that the quality of the acquaintance model can be iteratively improved by learning from obtaining unsatisfactory bids. Efficiency of the presented mechanism have been illustrated on two different examples – partially linear and partially constant distribution of services. The whole formal model and substantially richer set of experiments is available in [18].

It needs to be noted that besides obvious advantages, there also several disadvantages of the iterative-query algorithm. Mainly, the iterative-query algorithm is for the same quantity of message exchange generally slower. It is caused by the fact that in the case of batch-query algorithm, messages are sent to every provider in parallel (or perhaps in a single message). In the case of iterative-query algorithm, new queries are generated on the ground of previously received proposals. Therefore, we think that the iterative-query algorithm is particularly suitable in the domains, where:

– (i) the service amount granularity is very fine and it is technically impossible to enumerate all the amounts and (ii) the N parameter is not known a priori

– where the providers are motivated to minimize the amount of disclosed information (or the requester needs to pay for every information it receives when building the acquaintance model)
– getting the right $d_j(a_j)$ values takes the providers specific amount of time (e.g. given by measurement or non-trivial computation)

In real-life the $d_j(a_j)$ values in competitive environments also often depend on various other aspects such as past contracting track record, other providers providing to the same requester, providers not providing the respective requester, trust, etc.

A major disadvantage of the presented decomposition method is the fact that its results substantially depend on the chosen accuracy of the acquaintance model (i.e. the choice Δ for the iterative-query AM construction mechanism). Both the underestimation and overestimation of the objective functions may cause a deviation of the achieved decomposition from the optimum that possibly result in an increase of the real delivery time with respect to the expected $d_j(a_j)$.

The goal of the introduced contracting mechanism was a minimization of the overall duration of the service delivery. While a complementary problem may be a minimization of the overall price, in real-world environments both the duration and price are rather mutually inter-dependent and balance each other. In our future work we would like to explore such settings in which the price and duration are mutually dependent – this results in solving either a *constraint decomposition problem*, i.e. (i) minimization of duration under a maximum price constraint or (ii) minimization of price under maximum duration constraint or a *combined decomposition problem* when (iii) both the price and duration are minimized. In our future work we would like to carry out also experiments on scalability and to run the system populated with a substantially greater number (tens to hundreds) of negotiating agents.

Acknowledgement

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic under the grant no. MSM6840770013.

References

1. Pěchouček, M., Mařík, V., Bárta, J.: A knowledge-based approach to coalition formation. IEEE Intelligent Systems **17**(3) (2002) 17–25
2. Pěchouček, M., Tožička, J., Mařík, V.: Meta-reasoning methods for agent's intention modelling. In: Autonomous Intelligent Systems: Agents and Data Mining. Berlin: Springer (2005) 134–148
3. Rehák, M., Pěchouček, M., Tožička, J., Šišlák, D.: Using stand-in agents in partially accessible multi-agent environment. In: Proceedings of Engineering Societies in the Agents World V, Toulouse, October 2004. Number 3451 in LNAI, Springer-Verlag, Heidelberg (2005) 277–291
4. Sandholm, T.: Distributed Rational Decision Making. In: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press, Cambridge, MA. (1999) 201–258

5. Smith, R.G.: The contract net protocol: High level communication and control in a distributed problem solver. In IEEE Transactions on Computers **C-29**(12) (1980) 1104–1113
6. Sandholm, T., Lesser, V.: Coalitions among computationally bounded agents. Artificial Intelligence **94**(1-2) (1997) 99–137
7. Tambe, M.: Towards flexible teamwork. Journal of Artificial Intelligence Research **7** (1997) 83–124
8. Sycara, K., Decker, K., Pannu, A., Williamson, M., Zeng, D.: Distributed intelligent agents. IEEE Expert **11**(6) (1996) 36–46
9. Grosz, B., Kraus, S.: Collaborative plans for complex group action. Artificial Intelligence **86**(2) (1996) 269–357
10. Mařík, V., Pěchouček, M., Štěpánková, O.: Social knowledge in multi-agent systems. In Luck, M., Mařík, V., Štěpánková, O., eds.: Multi-Agent Systems and Applications. LNAI, Springer-Verlag, Heidelberg (2001)
11. Pěchouček, M., Mařík, V., Štěpánková, O.: Role of acquaintance models in agent-based production planning systems. In Klusch, M., Kerschberg, L., eds.: Cooperative Infromation Agents IV - LNAI No. 1860, Heidelberg, Springer Verlag (2000) 179–190
12. Cao, W., Bian, C.G., Hartvigsen, G.: Achieving efficient cooperation in a multi-agent system: The twin-base modeling. In Kandzia, P., Klusch, M., eds.: Cooperative Information Agents. Number 1202 in LNAI, Springer-Verlag, Heidelberg (1997) 210–221
13. T., W.: ARCHON: An Architecture for Multi-agent System. Ellis Horwood, Chichester (1992)
14. Sanyal, S., Jain, A., Das, S., Biswas, R.: A hierarchical and distributed approach for mapping large applications to heterogeneous grids using genetic algorithms. In: Proceedings. IEEE International Conference on Cluster Computing, Los Alamitos, CA, USA, IEEE Comput. Society (2003) 496–9
15. Gao, Y., Rong, H., Huang, J.Z.: Adaptive grid job scheduling with genetic algorithms. Future Generation Computer Systems **21**(1) (2005) 151–61
16. Fatima, S.S., Wooldridge, M.: Adaptive task and resource allocation in multi-agent systems. In: Proceedings of the Fifth International Conference on Autonomous Agents, New York, NY, USA, ACM (2001) 537–44
17. Li, C., Li, L.: Competitive proportional resource allocation policy for computational grid. Future Generation Computer Systems **20**(6) (2004) 1041–54
18. Lerch, O.: Effcient contraction mechanisms for virtual enterprises operation. Master's thesis, Czech Technical University (2005)