Using Grammar Induction to Model Adaptive Behavior of Networks of Collaborative Agents

Wico Mulder and Pieter Adriaans

Department of Computer Science
University of Amsterdam,
Science Park 107
1098 XG Amsterdam,
The Netherlands
wico.mulder@logica.com, pietera@science.uva.nl

Abstract. We introduce a formal paradigm to study global adaptive behavior of organizations of collaborative agents with local learning capabilities. Our model is based on an extension of the classical language learning setting in which a teacher provides examples to a student that must guess a correct grammar. In our model the teacher is transformed in to a workload dispatcher and the student is replaced by an organization of worker-agents. The jobs that the dispatcher creates consist of sequences of tasks that can be modeled as sentences of a language. The agents in the organization have language learning capabilities that can be used to learn local work-distribution strategies. In this context one can study the conditions under which the organization can adapt itself to structural pressure from an environment. We show that local learning capabilities contribute to global performance improvements. We have implemented our theoretical framework in a workbench that can be used to run simulations. We discuss some results of these simulations. We believe that this approach provides a viable framework to study processes of self-organization and optimization of collaborative agent networks.

Keywords: collaborative agents, learning, grammar induction, self-organization.

1 Introduction

The notion of an organization, as a network of collaborative agents, is almost as general as the idea of a system. In this paper[1] we study a formal model of learning organizations. We build on earlier work done in the domain of grammar induction, specifically the work of learning Deterministic Finite Automata (DFA) using the principle of Minimum Description Length (MDL), reported in

[1] Our work is supported by a BSIK grant from the Dutch Ministry of Education, Culture and Science (OC&W) and is part of the ICT innovation program of the Ministry of Economic Affairs (EZ) and a grant from the Casimir program from NWO.

J.M. Sempere and P. García (Eds.): ICGI 2010, LNAI 6339, pp. 163–177, 2010.

[2] and [3]. We base our approach on the broadly accepted theory of learnability, the notion of learning by identification [5], which deals with linguistic structures and the learnability of these structures. We replace the classical teacher student model with one in which a teacher/dispatcher presents a structured workload to an organization of students/agents with a certain learning capacity. Our work is motivated by research questions concerning the management of grid environments [11] and collaborative network organizations [10], but it also touches issues studied in ant colony behavior [12] and deep belief networks [7]. In hindsight, planning problems like the ones studied in [6] and [1] belong to the same domain, but there we used genetic algorithms to analyze the structure of the workload because at that time the techniques for learning DFA models from positive examples were not yet developed. Our work can also be seen as a more specific version of the problems studied in scheduling using local optimization [4], in the sense that we study variants with highly structured workloads.

The paradigm: Consider an organization of specialized worker-agents. Each agent can perform only one type of task and can work at only one task at a time. He has limited overview of the rest of the organization. He can delegate work to colleagues in his immediate environment, but he does not know the whole organization. A job description consists of a sequence of typed tasks. Workloads that consist of jobs are submitted to the organization by an agent or a dispatcher in the environment outside the organization. This dispatcher generates workloads with a certain structure. An agent accepts a job when the first task of this job matches the type of work he is specialized in. After acceptance, and after other pending work is finished, the agent executes the task and sends the rest of the job to one of his colleagues. The agents that were involved with the execution of a particular job report back to each other when the job was processed successfully and which individual agents executed the tasks. The agent keeps track of this information and uses these data to learn which type of jobs are to be routed to which of his direct colleagues. In the absence of (sufficient) data the agent will dispatch the jobs to his close colleagues at random, but as soon there is enough data to learn a model of the successful tracks of jobs through the organization the agent will use this model to route the work. In this sense one can say that the organization adapts its global behavior on the basis of local learning capabilities. We are interested in this kind of global adaptation as a result of local learning.

2 Formal Definitions

Definition 1. *A* ***job*** $< ID, AID, [< t_1, d_1 >, ..., < t_i, d_i >] >$ *consists of a job index with finite sequence of tasks. Here* $ID \in \mathbb{N}$ *is a job index, and AID is the index from the agent that sent the job. A* ***task*** *consists of a tuple* $< t_i, d_i >$ *with* $t_i \in T$ *is a type in a finite set of type* T *and* $d_i \in \mathbb{R}$ *is a number indicating a duration. We will consider tasks with duration* 1 *only, consequently the job notation can be simplified to* $< ID, AID, [t_1, ..., t_i] >$ *. A* ***workload*** *consists of a finite sequence of jobs.*

Definition 2. *A **collaborative learning agent** is a tuple* $< AID, t, WL, H, M, L, A, S, F >$ *where:*

- $AID \in \mathbb{N}$ *is an index.*
- $t \in T$ *is the type.*
- WL *is the **work-list**, a list of (partial) jobs to be executed by the agent. The first task in each job must be of type* t*.*
- H *is the **history**, which consists of a list of **reduced jobs** and a list of **processed job paths**. The list of reduced jobs contains jobs that are waiting or have been sent through by the agent after finishing a task. Jobs in the history can have three statuses: **waiting** (if the job still needs to be accepted by an other agent) or **sent** (if the job has been accepted by another agent) or **finished**. If the total job is finished the (partial) processed path is stored in the history. A processed path of a job that has been processed successfully by the organization has the form* $< ID, AID, [< t_1, AID_1, d_1 >, ..., < t_i, AID_i, d_i >] >$*, where* $AID_1, ..., AID_i$ *are the indexes of the agents that have actually executed the tasks.*
- M *is a learned **model**. The framework allows to use various types of models. In our work we study the use of DFA models.*
- A *is a **learning algorithm**. The learning algorithm takes a list of processing paths* P *as input and produces a model* M*.*
- L *is a **learning strategy**, that defines how and when* A *is invoked. One could consider batch learning, continuous learning, interval learning, learning with depreciation etc.*
- S *is a **job acceptation and distribution strategy**. This regulates how the learned model* M *is used to dispatch jobs to other agents.*
- F *is a **status** of the agent which can be either* free *or* busy*, depending on whether the agent is working on a job or not.*

Definition 3. *A collaborative learning agent is **resource-bounded** if one or more of its resources are limited. Limits to be considered could be: the size of the work-list, the size of the history, the size of the list of processed paths, the size of the model and the amount of processing time allowed to learn the model. An interesting boundary case is the situation in which the size of the work-list is* 1*, i.e. every agent can only handle one job at the time.*

Definition 4. *An **organization of collaborative agents** is a network (or digraph) of agents* $O =< \Gamma, r >$*, where* Γ *is the set of collaborative learning agents and* $r \in \Gamma \times \Gamma$ *is a directed cooperation relation. A **direct controlled neighborhood** of an agent* i *in an organization* O *is* $\{x| < i, x >\in r\}$*, i.e. the set of all agents that have a direct relation starting in* i*. A **direct supervised neighborhood** of an agent* i *in an organization* O *is* $\{x| < x, i >\in r\}$*, i.e. the set of all agents that have a direct relation ending in* i*. Agent* j *can be reached from agent* i *iff there is a path from* i *to* j*.*

Definition 5. *A teacher or **workload dispatcher** is an agent outside the organization that generates and submits workloads to the organization according to*

a certain submission strategy. This can be either a batch or a continuous stream of jobs.

Based on these definitions the learning process takes the following form:

1. Given are a workload dispatcher W and a finite number of agents Γ of various types T.
2. The workload dispatcher W and the agents agree on a class of workload descriptions from which W may select one to generate workloads. This step is analogous to the selection of a class of languages to be learned in the Gold model ([5]).
3. The agents Γ select an initial organization form, i.e. they select r.
4. The workload dispatcher starts to generate and submit jobs.

The whole process is discrete and regulated by a central timer. At each time-step a two phase process takes place:

1. Communication: The teacher submits a job $< ID, dispatcher, [< t_1, d_1 > , ..., < t_i, d_i >] >$ to an agent of the organization. The agents submit, using their distribution strategy S, reduced jobs $< ID, AID, [< t_k, d_k >, ..., < t_i, d_i >] >$ to their colleagues, where AID is the index of the dispatching agent. The agents accept a job from the dispatcher or from one of their colleagues and put it on their work-list. If a reduced job cannot be submitted it is kept in the history with **waiting** status. As soon as it is accepted the job gets the status **sent**. When an agent AID_n finished the last task of a job $< ID, AID_m, [< t_i, d_i >] >$ he sends a message $< ID, [< t_i, AID_n, d_i >] >$ to his supervising agent AID_m. This agent AID_m updates his history and sends the enriched description $< ID, [< t_l, AID_m, d_l >, < t_i, AID_n, d_i >] >$ to his supervising agent AID_l etc.
2. Execution: The agents perform a task of a job and put the reduced job in the history with the **waiting** status. Each agent selects a new job from his local work list. If there is no new job the status of the agent is **free**. If there is a task beeing carried out the status of the agent is **busy**.

Apart from this two-stage process agents can independently start learning sessions to update their model. This can simply be performed in a step after the communication or execution step.

One can study research questions of the following form in this setting:

- Does the organization **accept** a job with a certain structure at a certain moment in time? We say that an organization accepts a job when it is capable of processing this job, i.e. the job travels through the organization and ends in a situation in which each task of the job has been handled by an agent and finished.
- Is the organization **adequate** for a certain class of workloads, i.e. will all possible sequences of tasks be accepted?
- Is the structure of the organization **optimal** for a certain class of workloads, i.e. will all possible tasks be accepted in the shortest possible time?

The setting that we will study in this paper is the one in which the structural descriptions of the jobs match a regular language. Here the teacher selects a DFA to generate a workload. The agents use a learning algorithm based on MDL to learn DFA models on the basis of positive examples. The intuition is that an optimal organization for such a workload would be a model that is isomorphic to a parallel nondeterministic automaton (NFA) equivalent to the original DFA selected by the workload dispatcher. In order to analyze this we need a result from language learning theory.

Let a **theoretically optimal compression algorithm** be an algorithm that always finds the optimal compression of a data-set in terms of its Kolmogorov complexity. We know that such an algorithm does not exist, but also that it can be approximated in the limit ([9]). We also know that an MDL algorithm using such a compression algorithm is optimal, in the sense that it always find the best (or 'a' best, if there are more) theory in terms of randomness deficiency ([3]). Let's call such an MDL algorithm optimal. Such an optimal MDL algorithm does not exist, but it can be approximated in the limit. This insight allows us to use the notion of an optimal DFA-learner in some of the proofs below. The results represent limit cases that can be approximated empirically using practical implementations of MDL. Of course the observation that it might in practice be impossible to implement an effective coding scheme for the model and the data remains.

We can now turn our attention to organizational learning issues. We distinguish two types of learning:

Definition 6. *Given a set processing paths jobs of the form* $< ID, AID, [< t_1, AID_1, d_1 >, ..., < t_i, AID_i, d_i >] >$ *one can make three sets of sentences. Sentences in the first set have the form* $[t_1, ..., t_i]$*. Learning a DFA structure of this set amounts to learning the workload language. We call this* ***environmental learning****. Sentences in the second set have the form* $[AID_1, ..., AID_i]$*. Learning a DFA structure of this second set amounts to learning the structure of the organization given the workload language. We call this* ***organizational learning****.*

Efficient adaptive behavior depends on an intertwining of these two forms of learning.

3 Some Theoretical Results and an Open Problem

In this section we present some theoretical results. It is useful to consider some boundary cases.

Definition 7. *A* ***minimal unbounded clique*** *is an organization in which there is exactly one agent of each type with unbounded resources and in which each agent is connected to every other agent (including the reflexive connection).*

A clique is the organizational counterpart of a universal automaton that accepts any language. The corresponding theorem is:

Theorem 1. *A minimal unbounded clique is adequate for any finite workload.*

Proof: each agent can locally maintain a work-list of any length. Therefore the dispatcher can simply dispatch the whole workload to the relevant agents at once. After performing a task of a job the agent has always a neighbor of the right type to dispatch the task. Such an agent will always accept the task since there are no bounds for the work-list. Therefore, at any moment in time, as long as there are jobs in the system, at least one agent will perform at least one task. The total amount of work is reduced with each time step. Since the workload is finite the organization will finish all the work in a finite amount of time. The theorem also holds for cliques that are not minimal.

That the unboundedness is essential is clear from the following result:

Lemma 1. *A resource bounded clique can not accept every workload.*

Proof: Suppose we have a workload of size l containing jobs with similar tasks $< ID, AID, [t_1, ..., t_i] > | i > 1, t_i = t$. These jobs are accepted by an agent of type t having a bounded memory of size k. For each job, this agent forwards a reduced job to itself. Now, if $l > k$, then after a finite number of steps the memory of this agents gets fully occupied and both the dispatch as well as the agent will keep on waiting.

Since each agent acts as a dispatcher of reduced jobs, this can happen for any number of agents of type t in the clique. Such a situation can be explained using the notion of gridlock, commonly used to describe congestion due to traffic that blocks itself.

An adaptive organization needs to find a balance between two forces, 1) the structure of the workloads and 2) its internal structure. Separating these two issues is not always possible or necessary on the basis of local learning capabilities. For example suppose that a teacher/dispatcher is not a good informer, in the sense of Gold, for the workload language, i.e. there are parts of the language that are never produced. In that case there might be parts of the organization that are never used, but an agent with only local knowledge of the organization might never know this. We therefore introduce the notion of a universal dispatcher:

Definition 8. *Given a type set T, a **universal dispatcher** U is one which creates workload on the basis of the universal language of T, i.e. any finite subset of T^* can be a valid work load. We demand that the universal dispatcher also is a text for this language, i.e. every string in T^* will be produced by U in the limit. A universal dispatcher randomly chooses for each job an agent of the right type, i.e. an agent that can perform the first task.*

One could view a universal dispatcher as an environment that creates maximally noisy messages. Such noise gives a possibility for the local agents to explore the organization. The following situation illustrates this. Let us define the notion of a mixed-clique organization. This will be an organization that consists of two or more cliques that are mixed over the individual agents :

Definition 9. *$O_{1,2} = O_1 \cup O_2$ is a **two-clique organization** iff the following conditions hold: we have type sets T_1 and T_2 such that $T_1 \cap T_2 \neq \emptyset$ and $(T_1 - T_2) \cup$*

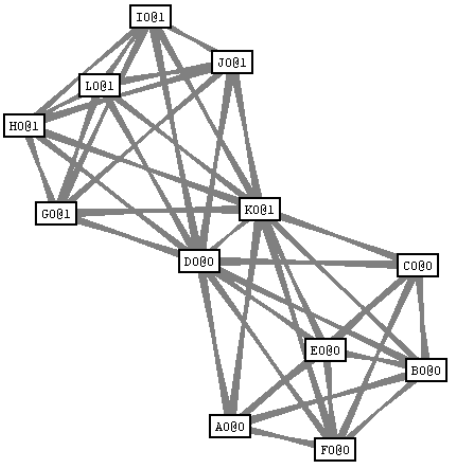

Fig. 1. an example of a two-clique organization

$(T_2 - T_1) \neq \emptyset$, *i.e. they overlap but are mutually different. The two organizations* $O_1 =< \Gamma_1, r_1 >$ *and* $O_2 =< \Gamma_2, r_2 >$ *are such that* O_1 *contains a finite non-empty set of agents for each type* $t \in T_1$ *and* O_2 *contains a finite non-empty set of agents for each type* $t \in T_2$. *Moreover* O_1 *and* O_2 *are non-minimal cliques that overlap in the sense that there are agents that belong to* O_1 *as well as* O_2 *but for some types* $t \in T_1 \cap T_2$ *there are agents* $< a, t, WL, H, M, L, A, S, F >\in O_1$ *but* $< a, t, WL, H, M, L, A, S, F >\notin O_2$, *i.e. the organizations share types but not all agents of a certain type belong to both organizations.*

The problem for agents in a two-clique organization is that they do not know to which part of the organization they themselves or their direct controlled neighborhood belong. A fundamental question is whether the agents still can learn optimal routing in such a confused setting. We can prove the following lemma:

Lemma 2. *Given a universal dispatcher for* $T_1 \cup T_2$ *and a corresponding two-clique organization* $O_{1,2}$ *with agents that use an optimal DFA induction strategy, the agents will in the limit create a maximally adequate organization, in the sense that any workload that can be processed by the organization will be processed.*

Proof: (Sketch) Note that the two cliques in $O_{1,2}$ only cooperate with each other over the agents that they share. The corresponding workload-language is one in which arbitrary fragments of T_1^* can via shared types be mixed with arbitrary fragments of T_2*. The universal dispatcher will create four types of jobs: 1) jobs that only contain tasks of types in T_1, 2) jobs that only contain tasks of types in T_2, 3) jobs that contain tasks that belong to $T_1 \cup (T_2 - T_1)$ or 4) to $T_2 \cup (T_1 - T_2)$. The first two types of jobs can be processed by the organization, the others not necessarily since they contain tasks that can only be performed in different parts of the organization that not necessarily have direct

communication. The universal dispatcher will distribute the jobs randomly over the appropriate agents. These agents will perform their task and then select an appropriate agent for the reduced task in their direct controlled environment. The job-descriptions will remain in the histories of the individual agents. In the limit these histories will contain sublists of successful jobs and jobs that apparently never were processed by the rest of the organization. Note that the histories of the successful jobs are tagged with the ID's of the individual agents. Now by performing two learning algorithms an agent can learn two models: by performing an MDL learning algorithm on the sequences of types that correspond to successful jobs he can learn which parts of the organization accept which type of jobs, this is environmental learning. This model will be a DFA over the alphabet of types. By performing a DFA learning algorithm in the sequences of ID's of agents he can learn a model of the organization. this is organizational learning. This will be a DFA over the alphabet of agent ID's. The optimality of the DFA induction guarantees that in the limit these models will be correct. This ensures that any local agent in the organization will only dispatch jobs to agents of which he is certain that they can handle them.

This lemma can be generalized to the following general theorem:

Theorem 2. *Given a universal dispatcher for T and a corresponding organization O of any structure with agents that use an optimal DFA induction strategy, the agents will in the limit create a maximally adequate organization (given its structure), in the sense that any workload that can be processed by the organization will be processed.*

Proof: (Sketch) A job that has to be handled by the an organization O can end up in three ways: 1) it gets stuck when there is no connection to a colleague agent that can handle the next task, 2) it gets stuck for similar reasons, but due to an agent that made a wrong routing decision, 3) it gets accepted. Every agent in O maintains a history. On the basis of this history an agent can in the limit learn which agents in his direct controlled environment can process which types of sentences. He can use this information to route workloads. If the learned models are adequate, then no routing decision of any agent diminishes the processing capacity of the total organization and no jobs get unnecessary stuck. While learning the number of jobs that get stuck due to a wrong routing decision diminishes: i.e. the organization is maximally adequate.

Note that if there are multiple entry agents with different levels of adequacy, the successful processing of a job might depend on the first agent that is selected by the workload dispatcher, but this is obviously also a problem of the original organization, so this does not depreciate the value of this proof. If the original organization could process the job starting from a certain agent, then so can the optimized organization.

Unfortunately nice general proofs such as the one presented above are not available for organizational learning. Even if the agents have optimal DFA learning algorithms, issues of local versus global organization come in to play. It might be the case that a local optimization of one agent prevents other agents from

performing more efficient optimization. Since there is a timing issue local adaptations might oscillate or sweep through the organization in a chaotic way. We conclude this section with the formulation of an open problem:

Definition 10. ***Organizational Learning problem:*** *Given a work dispatcher that uses a regular job language, can an adequately rich clique of agents with optimal learners always converge to an optimal organization?*

If this problem is unsolvable in general, we would be interested in the particular constraints that make it viable. We will leave this for future research.

4 Experiments

4.1 Simulator

We developed a software workbench to run simulations of learning agent organizations. The workbench, called "Workbench for Intelligent Collaborative Organizations" (WICO) can be used to create various workloads, organizations and experimental setups.

Components of the workbench are:

- a **DFA editor** that can be used to build a pre-defined probabilistic DFA structure for possible workloads
- a **workload generator** that generates jobs form a given probabilistic DFA
- a **work dispatcher** that can be configured to send jobs to an organization of agents
- an **organization factory** for creating agent organizations with different topologies
- a **DFA learning algorithm** based on **MDL** which is used by the agents to learn DFA models
- a number of **visual components** to visualize experiments, organizations, workloads and the DFA models
- an **experiment controller unit** that can be used to define experiments and capture results

4.2 Learning Capacity

In a first series of experiments we used the workbench to study the **environmental learning** capacity of an unbounded network. The workload was generated using a probabilistic DFA as shown in figure 4. Figure 3 shows an example of a minimal unbounded clique aimed to learn the task structure of the workload. Using this DFA a workload was generated containing strings such as e.g. *BDEFE, ACEFE, BDEFE, BDEFEFEFEF, BDEFEF, ACEFEFE, BDEFEFE, BDEFEF, BDEFEFE, ACEFEF, ...* . Figure 5 shows the DFA of agent A as it has learned the successful jobs in which it was involved. This grammar also reflects the organization 'below' him, i.e. those agents that were involved with the reduced jobs.

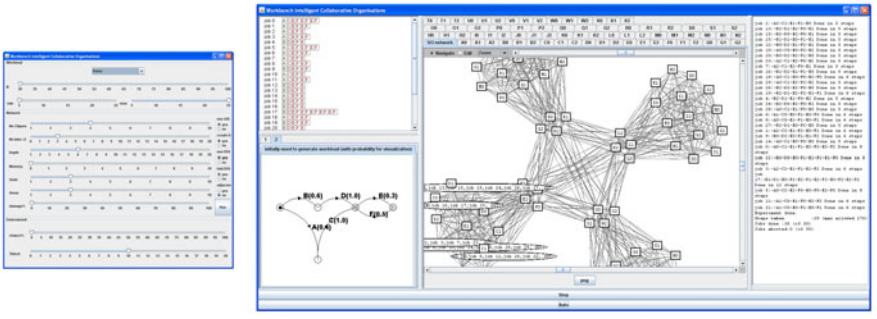

Fig. 2. screen shot of the workbench environment

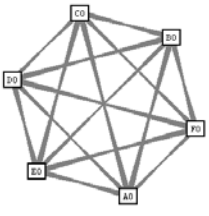

Fig. 3. a minimal-clique

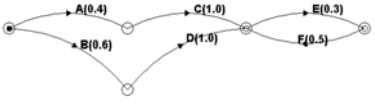

Fig. 4. DFA used for workload generation

Fig. 5. DFA of agent A0

We studied the **organizational learning** capabilities by using a uniform dispatcher sending jobs through various types of organizations. Figure 6 shows a typical result of 1000 dispatched jobs with a task length between 5 and 10. Experiences were that only a fraction of these random jobs got successfully processed, as most of them got stuck in the network because of the lack of a possible connection. The figure shows the DFA models of three agents. Each agent maintains two models; one only containing the task labels and one containing task labels and together with the index of the agent that processed that task. Figure 7 shows another example.

4.3 Network Performance

In a second series of experiments, we looked at the network performance by measuring the proportion of jobs that are successfully handled. The local DFA model is used to determine whether an agent is able to handle the rest of the job. We used a network that consists of two cliques, symbolizing an organization of two departments. One department contains agents that can handle tasks of type A, B, C, D and another department that is specialized in the processing of tasks E and F. Figure 8 shows the organization that was used in this experiment.

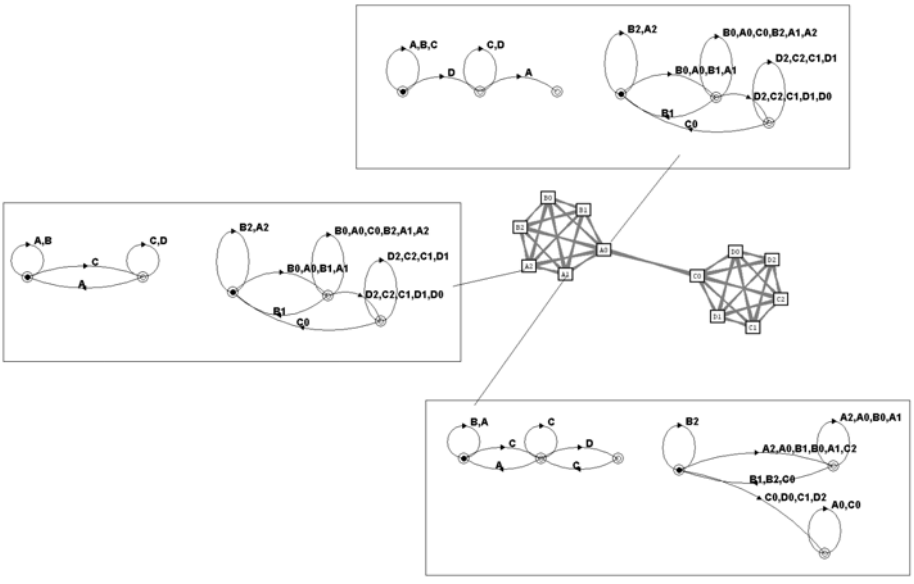

Fig. 6. organizational learning in a simple two-clique organization

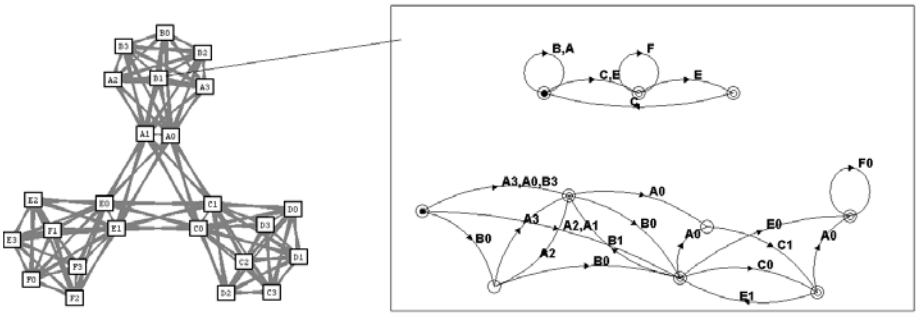

Fig. 7. organizational learning in a three-clique organization

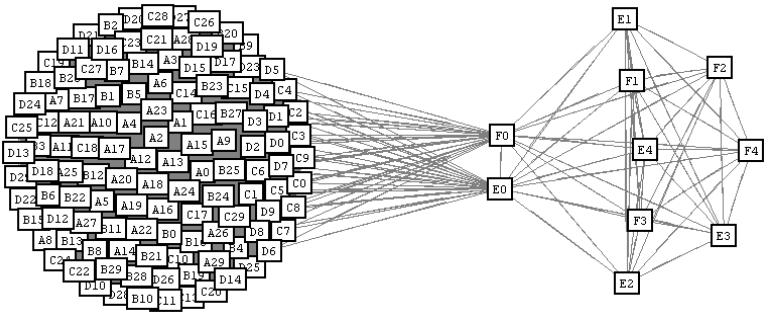

Fig. 8. Graph of a typical organization used in our performance experiments

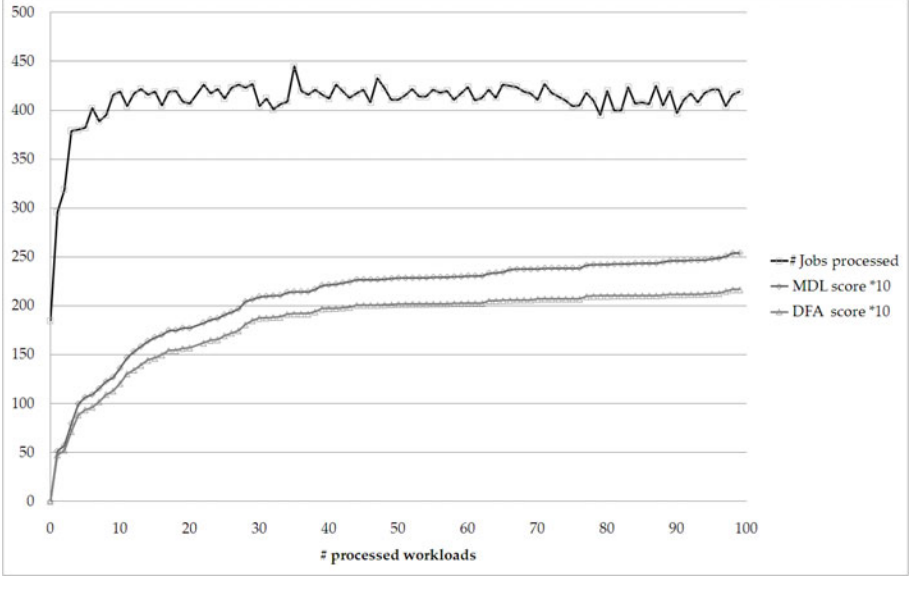

Fig. 9. Results of the experiment on network performance while learning

We looked at the influence of using the locally learned DFA models on the global task handling performance while it processed a series of 100 workloads. To be able to measure the network performance while the network gradually learns, we used a step-by-step approach: we processed a workload of 500 jobs by the network while the agents were instructed not to update their models, after which we processed a small workload of 10 jobs, letting the agents update their model. This was repeated 100 times. The result can been seen in figure 9 (upper curve). One can see that as the network learns, the number of successfully processed jobs increases gradually from roughly 185 to 425. Gradually more C

and D agents updated their model on successful jobs. For the A and B agents the C and D agents that updated their model become more attractive to forward a job. Other C and D agents become less attractive to get a forwarded job. In case there are many agents of one type in an organization, each agent uses its DFA model to decide on which agent gets forwarded a job. The reason for not being able top handle the 500 jobs is because the agents used a strategy based on a probability distribution. Chances are high for agents that can verify that they can handle the rest of a particular forwarded job, but still a low chance of an alternative has been left open. We know that such a strategy can improved.

We looked at the DFA- and MDL complexity score of the individual agent models during the learning process. For the calculation of these scores we refer to earlier work in [11]. The score for the network is calculated as the sum of the DFA and MDL complexity of the individual agent models. Figure 9 (lower two curves) shows the DFA- and MDL scores. Both curves show that both the model-code (DFA score) as well as the model-code including the data (MDL score) gradually evolve until all agents have learned an (almost) complete model for these kind of tasks. The curve of the DFA complexity is expected to behave asymptotically as the structure of the models will converge, the MDL score keeps slowly increasing as long as there are new unique series of jobs send by the workload dispatcher.

5 Conclusion

An organization can learn grammatical models of both workload structures and their own organization while handling sequences of tasks. Using the models, the agents can make early statements about the acceptance of tasks and make decisions on forwarding jobs. We showed that locally learning agents, when using their grammar models in their decision to forward jobs, contribute to the improvement of the global network performance on processing jobs. We believe that our framework is useful for the analysis of problems in the optimization of agent organizations in general. This holds in particular for networks we studied in earlier work; grid infrastructures and collaborative business organizations. The distinction between organizational and environmental learning, and the insight that the strategic impact of both forms of learning is very different, is very important. Every manager working in a complex organization has the experience that extreme efficiency is at times counterproductive. Sometimes it is useful to blow a bit of random noise through the organization to discover where the real bottlenecks are. The theoretical results in the paper (theorem 2) seem to corroborate this insight.

6 Future Work

The theoretical framework and workbench can be used to investigate issues in related fields of research. The notion of agents that make local decisions to handle

and dispatch tasks poses new research questions in the field of **routing and load balancing**. Developing agent strategies for optimal path learning, allows one to investigate how local modeling can lead to robust behavior and global performance optimization. For **continuous streams** of work one can have the following additional criteria: is the organization **insufficient**, **in equilibrium** or **redundant**. The notion of dynamic agent topologies, i.e. the creating and deletion of agents and connections on the fly, allows for research on the handling of continuous streams. We also want to study the influence of **perturbations** on the network of agents, which amounts to the research on the stability and reliability of grid infrastructures. We want to investigate under which conditions an organization in unstable environments can still learn to handle task structures and optimize its behavior.

A **cloud infrastructure**, or shortly cloud, is a scalable and configurable network of ICT resources that are implemented and exploited as services that can be accessed via the internet. The fact that different services might be owned by different organizations imposes many challenges on the management of clouds. In previous work [11] we discussed support for performance management in Grids. We foresee similar problems in the field of cloud computing and think that our approach of self learning agents can contribute there as well.

References

[1] Adriaans, P.W.: Predicting pilot bid behavior with genetic algorithms (Abstract), Symbiosis of Human and Artifact. In: Anzai, Y., et al. (eds.) Proceedings of the Sixth International Conference on Human-Computer Interaction, HCI International 1995, Tokyo, Japan, pp. 1109–1113 (1995)

[2] Adriaans, P.W., Jacobs, C.J.H.: Using MDL for grammar induction. In: Sakakibara, Y., Kobayashi, S., Sato, K., Nishino, T., Tomita, E. (eds.) ICGI 2006. LNCS (LNAI), vol. 4201, pp. 293–306. Springer, Heidelberg (2006)

[3] Adriaans, P.W., Vitányi, P.: Approximation of the Two-Part MDL Code. IEEE Transactions on Information Theory 55(1), 444–457 (2009)

[4] Anderson, E.J., Glass, C.A., Potts, C.N.: Machine scheduling. In: Aarts, E., Lenstra, J.K. (eds.) Local Search in Combinatorial Optimization, pp. 361–414. John Wiley & Sons, Inc., New York (1997)

[5] Gold, E.M.: Language Identification in the Limit. Information and Control, 447–474 (1967)

[6] den Heijer, E., Adriaans, P.W.: The Application of Genetic Algorithms in a Carreer Planning Environment: CAPTAINS. International Journal of Human-Computer Interaction 8, 343–360 (1996)

[7] Hinton, G.E., Osindero, S., Teh, Y.: A fast learning algorithm for deep belief nets. Neural Computation 18, 1527–1554 (2006)

[8] Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation, 2nd edn. Addison-Wesley, Reading (2001)

[9] Li, M., Vitányi, P.: An Introduction to Kolmogorov Complexity and Its Applications, 3rd edn. Springer, Heidelberg (2008)

[10] Mulder, W., Meijer, G.R.: Distributed information services supporting collaborative network management. In: IFIP International Federation for Information Processing, Proceedings PROVE 2006. Network-Centric Collaboration and supporting frameworks, vol. 224, pp. 491–498. Springer, Heidelberg (2006) ISBN 0-387-38266-6

[11] Mulder, W., Jacobs, C.J.H.: Grid Management Support by Means of Collaborative Learning Agents. In: Grids Meet Autonomic Computing, Workshop at the 6th IEEE International Conference on Autonomic Computing (ICAC), Barcelona, pp. 43–50. ACM, New York (2009) ISBN: 978-1-60558-564-2

[12] Sim, K.M., Sun, W.H.: Ant Colony Optimization for Routing and Load-Balancing: Survey and New Directions. IEEE Transaction on system, man and cybernetics 33(5), 560–572 (2003)