Learning agent organisations: studies on collaborative modelling, performance management and learning capacities of networks of collaborative agents
Mulder, W.

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
6 MODELLING LEARNING AGENT ORGANISATIONS

We introduce a formal paradigm to study global adaptive behaviour of organisations 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 into a workload dispatcher and the student is replaced by an organisation of workers. The jobs that the dispatcher creates consist of sequences of tasks that can be modelled as sentences of a language. The workers in the organisation 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 organisation can adapt itself to structural pressure from an environment. We show that local learning capabilities contribute to global performance improvements. The theoretical framework is implemented 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-organisation and performance optimisation of collaborative agent organisations.

6.1 A formal descriptive model of learning organisations

The notion of an organisation as a network of collaborative agents is almost as general as the idea of a system. In this chapter we study a formal model of learning organisations. We build on earlier work done in the domain of grammar induction, specifically the work of learning DFAs using MDL (Adriaans & Jacobs [6]) and (Adriaans & Vitányi [7]). We base our approach on the broadly accepted theory of learnability, the notion of learning by identification (Gold [58], Angluin [11]), 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 organisation of students/workers with a certain learning capacity.
The work described in this chapter is motivated by research questions concerning the management of grid environments (Mulder & Jacobs [102]) and collaborative network organisations (Mulder & Meijer [99]), but it also touches on issues studied in ant colony behaviour (Sim et al. [121]) and deep belief networks (Hinton [69]). In hindsight, planning problems like the ones studied in (Heijer et al. [63]) and (Adriaans [4]) belong to the same domain, but in that work genetic algorithms were used to analyse the structure of the workload. At that time the techniques for learning DFA's 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 optimisation (Anderson [10]) in the sense that we study variants with highly structured workloads.

The paradigm: Consider an organisation of specialized worker-agents. Each agent can perform only one type of task and can work at only one task at a time. An agent has limited overview of the rest of the organisation and can delegate work to colleagues in his immediate environment, but he does not know the whole organisation. A job description consists of a sequence of typed tasks. Workloads consisting of series of jobs are submitted to the organisation by an agent or a dispatcher in the environment outside the organisation. This dispatcher generates and sends workloads with a certain structure to the organisation of agents. 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 are involved with the execution of a particular job report back to each other when the job has been processed successfully and also report which individual agents have executed which task of that job. Each agent keeps track of such information and uses it to learn which type of job is 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 as there is enough data to learn a model of the successful tracks of jobs through the organisation the agent will use this model to route the work. In this sense one can say that the organisation adapts its global behaviour on the basis of local learning capabilities. We are interested in this kind of global adaptation as a result of local learning.

31 In our approach however, the agents are situated at fixed locations, which differs from the literature about agent based path optimisations, as e.g. used in Ant Colony Optimisation (ACO) studies. Though, in our situation one may regard the jobs as being the mobile entities that leave trails like ant-pheromones. To the best of our knowledge, there have not been any ACO studies in which grammar models are used.
6.2 Regular Task Learning (RTL) framework

**Definition:** A job $<ID, AID, [<t_1,d_1>, ..., <t_n,d_n>]> \text{consists of a job index with a finite sequence of tasks. Here } ID \in \mathbb{N} \text{ is a job index, and } AID \in \mathbb{N} \text{ is the index from the agent that sent the job. A task consists of a tuple } <t,d> \text{ with } t \in T \text{ is a type in a finite set of types } T \text{ and } d \in \mathbb{R} \text{ 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, ...,t]> \text{. A workload consists of a finite sequence of jobs.} $

**Definition:** 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 a type in a finite set of types.
- $WL$ is a 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 a 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 another agent) or sent (if the job has been accepted by another agent) or finished. If a job is finished the processed path, or a part of it, is stored in the history. A processed path of a job that has been processed successfully by the organisation has the form $<ID, AID, [<t_1,AID_1,d_1>, ..., <t_n,AID_n,d_n>]>$, where $AID_1, ..., AID_n$ are the indexes of the agents that have actually executed the tasks.
- $M$ is a learned model. The RTL-framework allows us 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 acceptance 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:** 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:** An organisation of collaborative agents is a network (or digraph) of agents \( O = < \Gamma, r > \), where \( \Gamma \) is the set of collaborative learning agents and \( r \in \Gamma \times \Gamma \) is a directed cooperation relation. A direct controlled neighbourhood of an agent \( i \) in an organisation \( O \) is \( \{ x \mid < i, x > \in r \} \), i.e. the set of all agents that have a direct relation starting in \( i \). A direct supervised neighbourhood of an agent \( i \) in an organisation \( O \) is \( \{ x \mid < x, i > \in r \} \), 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:** A teacher or workload dispatcher is an agent outside the organisation that generates and submits workloads to the organisation 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:
- Given are a workload dispatcher \( W \) and a finite number of agents \( \Gamma \) of various types \( T \)
- The workload dispatcher \( W \) and the agents agree on a class of workload structures 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 (Gold [58])
- The agents \( \Gamma \) select an initial organisation form, i.e. they select \( r \)
- 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:

- Communication: The teacher submits a job \(<ID, \text{dispatcher}, [<t_1,d_1>, ..., <t_i,d_i>]\) to an agent of the organisation. 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. An agent accepts a job from the dispatcher or from one of his colleagues and puts it on his work-list. If a reduced job cannot be submitted, i.e. if there is no agent that can handle the next task of the job, 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 \(<ID, [<t_i, AID_m, d_i>]\) to his supervising agent \(AID_m\). This agent \(AID_m\) updates his history and sends the enriched description \(<ID, [<t_i AID_m, d_i>, <t_i AID_m, d_i>]\) to his supervising agent \(AID_l\) etc.

- Execution: The agents perform a task of a job and put the reduced job in the agents local 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 being carried out the status of the agent is busy.

In parallel to this two-stage process the agents can have their own process of updating or generalizing their models.

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

- Does the organisation accept a job with a certain structure at a certain moment in time? We say that an organisation accepts a job when it is capable of processing this job, i.e. the job travels through the organisation and ends in a situation in which each task of the job has been handled by an agent and finished.

- Is the organisation adequate for a certain class of workloads, i.e. will all possible sequences of tasks be accepted?

- Is the structure of the organisation optimal for a certain class of workloads, i.e. will all possible tasks be accepted in the shortest possible time?

The setting that we study in this work is the one in which the structural descriptions of the jobs match a regular language. Here the teacher selects a DFA to generate workloads. The worker-agents use a learning algorithm based on MDL to learn DFA models on the basis of positive examples. The intuition is that an optimal organisation 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 analyse 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 (Li & Vitánji [85]). 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 (Adriaans & Vitánji [7]). 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 limited 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 organisational learning issues.

We distinguish two types of learning:

**Definition:** Given a set of *processing paths* of jobs of the form \(<ID, AID, [ <t_1, AID_1, d_1>, ..., <t_i, AID_i, d_i> ] >\) one can make two 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 **environment 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 organisation given the workload language. We call this **organisation learning**.
6.3 Some theoretical results and an open problem

As it is useful to consider some boundary cases, we present some theoretical results.

**Definition:** A *minimal unbounded clique* is an organisation 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 minimal unbounded clique is the organisational 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 correspondingly typed agents at once. After performing a task of a job the agent has always a neighbour 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 organisation 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:** A resource bounded clique cannot accept every workload.

**Proof:** Suppose we have a workload of size $l$ containing jobs with similar tasks $\langle ID, AID, [t_1, \ldots, t_i] \rangle \mid i > 1, t_i = t$. 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\(^{32}\), commonly used to describe congestion due to traffic that blocks itself.

\(^{32}\) The term ‘gridlock’ was coined in 1980 by Sam Schwartz ("Gridlock Sam").
An adaptive organisation 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 organisation that are never used, but an agent with only local knowledge of the organisation might never know this. We therefore introduce the notion of a universal dispatcher:

**Definition:** Given a type set $T$, a universal dispatcher $U$ is one which creates workloads on the basis of the universal language of $T$, i.e. any finite subset of $T^*$ can be a valid workload. 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 organisation. The following situation illustrates this. Let us define the notion of a mixed-clique organisation. This will be an organisation that consists of two or more cliques that are mixed over the individual agents:

**Definition:** $O_{1,2} = O_1 \cup O_2$ is a two-clique organisation iff the following conditions hold: we have types sets $T_1$ and $T_2$ such that $T_1 \cap T_2 \neq \emptyset$ and $(T_1 - T_2) \cup (T_2 - T_1) \neq \emptyset$, i.e. they overlap but are mutually different. The two organisations $O_1 = <\Gamma_1, r_1>$ 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 organisations share types but not all agents of a certain type belong to both organisations.
Figure 6.2: Illustration of a two-clique organisation.

The problem for agents in a two-clique organisation is that they do not know to which part of the organisation they themselves or their direct controlled neighbourhood 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 1:** Given a universal dispatcher for $T_1 \cup T_2$ and a corresponding two-clique organisation $O_{1,2}$ with agents that use an optimal DFA induction strategy, the agents will in the limit create a maximally adequate organisation, in the sense that any workload that can be processed by the organisation 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 $T_1 \cup (T_2 - T_1)$. The first two types of jobs can be processed by the organisation, the others cannot necessarily be processed since they contain tasks that can only be performed in different parts of the organisation 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 sub-lists of successful jobs and jobs that apparently never were processed by the rest of the organisation. 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: 1) by performing an DFA learning algorithm on the sequences of types that correspond to successful jobs the agent can learn which parts of the organisation accept which type of jobs. This is environment learning. The model will be a DFA over the alphabet of types. 2) by performing a DFA learning algorithm in the sequences of ID's of agents he can learn a model of the organisation. This is organisation 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 organisation 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 organisation \( O \) of any structure with agents that use an optimal DFA induction strategy, the agents will in the limit create a maximally adequate organisation, in the sense that any workload that can be processed by the organisation will be processed.

**Proof**: (Sketch) A job that has to be handled by an organisation \( O \) 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 organisation and no jobs get unnecessary stuck. i.e. the organisation 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 organisation, so this does not depreciate the value of this proof. If the original organisation could process the job starting from a certain agent, then so can the optimised organisation.

Theorems and proofs such as the one presented above are not available for organisation learning. Even if the agents have optimal DFA learning algorithms, issues of local versus global organisation come into play. It might be the case that a local optimisation of one agent prevents other agents from performing more efficient optimisation. Since there is a timing issue local adaptations might oscillate or sweep through the organisation in a chaotic way. We conclude this section with the formulation of an open problem:

**Definition:** Organisation 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 organisation?

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.

Related to our organisation learning problem is the fact that local optimisation not always leads to global performance improvement.

**Theorem 3:** A local optimisation, i.e. an optimisation made by a single agent in its own model, does not always lead to global performance improvement of the organisation.

**Proof:** (Sketch) Suppose we have an organisation consisting of agents as illustrated below. Agent A can accept tasks of type a, agent B₁ and B₂ accept tasks of type b, agent C accepts tasks of type c and agent D accepts task of type d. Now, a job consisting of a series of task that begins with abd might be handled by agent A and passed to agent B₁. Successful handling of agent D will result in an update of agent A’s model to forward jobs with a next task labelled b always to agent B₁. Suppose a next job accepted by agent A consists of a task series beginning with abc, then, due to the previous update of the model, this job is not forwarded to agent B₂ and agent C, but instead might get forwarded to agent B₁ where is gets stuck. This second job is then not handled by the organisation, which can be interpreted as a decrease in its performance.
Without looking ahead and without performing extra communication, agent $A$ might not even know that agent $C$ exists. A situation like this may happen for any task in a job.

To overcome unwanted deadlocks, extra communication and feedback about state and decision-making between the agents is necessary. Practical workarounds, such as using interval based communication between neighbouring agents about their current state or using probability distributions in dispatching tasks can be applied, but this gives no guarantees for every possible situation. At the moment we do not know how to ensure that a local optimisation always contributes to global optimisation. We think that possible answers may be obtained from the field of distributed algorithms and concurrent systems. A typical example of the need of a synchronisation mechanism is the ‘Dining Philosophers’ problem\(^{33}\) (Lynch [88]). It demonstrates the need of an extra algorithm to overcome a deadlock situation due to local decisions. There are algorithms that solve the problem, but all of them are based on extra communication about synchronizing individual actions and decisions.

---

\(^{33}\) The ‘Dining Philosophers problem’ is a classic computing synchronisation problem in concurrency. There are five philosophers and five chopsticks. A philosopher must obtain the use of his left and right chopstick concurrently to be able to eat, which means two neighbouring philosophers cannot eat simultaneously since they share the resource, the chopstick. If he is unable to eat, he rests and thinks. The lack of available chopsticks is an analogy to the locking of shared resources in real computer programming.
6.4 Experiments

6.4.1 Workbench Intelligent Collaborating Organisations (WICO)

We developed a software workbench to run simulations of learning agent organisations. The workbench, called "Workbench for Intelligent Collaborative Organisations" (WICO), can be used to create various types of workloads, organisations 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 organisation of agents
- an **organisation factory** for creating agent organisations 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, organisations, workloads and the DFA models
- an **experiment controller unit** that can be used to define experiments and capture results

The workbench allows for the generation of various types of workload structures: An impression of some different types of workload structures is given in table 6.1.
Figure 6.4: Screen shots of the workbench environment
<table>
<thead>
<tr>
<th>Structure of the workload</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABAAA</td>
<td></td>
</tr>
<tr>
<td>Self-loop</td>
<td></td>
</tr>
<tr>
<td>Nested loop</td>
<td></td>
</tr>
<tr>
<td>One loop</td>
<td></td>
</tr>
<tr>
<td>Multi forward</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.1: impression of different structural types of workloads
6.4.2 Environment and organisation learning

In a first series of experiments we used the workbench to study the environment learning capability of an unbounded network.

A workload was generated using a probabilistic DFA as shown in Figure 6.5b. Using this DFA a workload was generated containing strings of the form $AC(EF)^n$, $ACE(FE)^n$, $BD(EF)^n$, $BDE(FE)^n$. $n \geq 0$.

Figure 6.5a shows an example of a minimal unbounded clique aimed to learn task structures of the workload dispatcher. Figure 6.5c shows the DFA of agent A as it has learned from successful jobs in which it was involved. This grammar also reflects the organisation of the colleague agents that were involved with the successful jobs.

![Diagram of minimal clique](image)

![Diagram of DFA](image)

**Figure 6.5**: a) minimal-clique, b) workload generation DFA, c) DFA of agent A
We studied the *organisation learning* capabilities by using a uniform dispatcher sending jobs through various types of organisations. Figure 6.6 gives an impression for typical situations in a two-clique organisation which handled 1000 jobs with random task lengths. 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 some of the agents. Each agent maintains two models[^34]: one containing only the task labels and one containing task labels together with the index of the agent that processed that task.

![DFA models of agents](image)

**Figure 6.6: Illustration of environment- and organisation learning**

[^34]: Note: these are not the local and global models that we used in chapter 3 and 4. In this setting, there is no global model learned or used at all. The agents only use local models, which can represent organisational structures or task structures.
6.4.3 Performance optimisation

In a second series of experiments, we looked at the global performance of agent organisations 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 next task of the job. We used a network that consists of two cliques, symbolizing an organisation of two departments. Figure 6.7 shows an organisation that was used in this experiment. The organisation consists of two departments; one department contains agents that can handle tasks of types A, B, C, D, the other department is specialized in the processing of tasks E and F. Only a few C and D agents have a link with the E and F department.

![Graph of a typical organisation used in our performance experiments. The left clique shows a department that can handle A, B, C, and D tasks. The right clique is a department that handles E and F tasks. Only a few C and D agents know how to communicate with the E and F department.](image)

Figure 6.7: Graph of a typical organisation used in our performance experiments. The left clique shows a department that can handle A, B, C, and D tasks. The right clique is a department that handles E and F tasks. Only a few C and D agents know how to communicate with the E and F department.

Figure 6.8 (page 106) shows the results of a typical experiment which shows how the global performance handling of the organisation gets improved while learning local DFA models. The experiment illustrates the basic ideas and workings of an organisation that learns using agents that learn locally.

Initially all agents in the organisation have no model at all. Jobs are just forwarded to a colleague agent that can handle the next task. A workload that has to be handled by this group of agents might not get fully handled, since some jobs might get stuck at an agent that has no connection to the E and F agents. However, jobs that get handled cause the DFA model of the involved agents to be filled with successfull paths. At a next moment, when a new workload has to be handled, the
agents use their model to prioritize their decision on forwarding jobs to their colleagues. Those agents who previously were successfully involved, will have a higher chance of getting the rest of the job than those agents who were not.

To be able to actually see the influence of the locally learned DFA models on the global network performance we processed a series of workloads. For this experiment we generated and dispatched 100 workloads, each consisting of 500 jobs. The problem for such an experiment is that during the handling of the jobs, the network already learns. To see how it 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, during which we allowed the agents to update their model. In other words during those 10 jobs, the organisation learns, while the 500 jobs just provide us a means to reasonably count the number of jobs that came through. This was repeated for 100 workloads.

Figure 6.8 shows the resulting learning curve (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 those C and D agents become more attractive to forward a job. Other C and D agents become less attractive to get a forwarded job. The reason that the total number of handled jobs does not reach 500 is a side-effect of a simple stochastic job dispatching strategy we used here.

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 chapter 3 and earlier work in (Mulder & Jacobs [101]). The score for the network is calculated as the sum of the complexity of the individual agent models. Figure 6.8 (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 sent by the workload dispatcher.
Figure 6.8: Results of the experiment on network performance while learning
6.5 Conclusions and future work

Collaborative agents are able to learn grammatical models of both workload structures and their own organisation while handling sequences of tasks. Using the models, the agents can make early statements about the acceptance of tasks and make proper decisions on forwarding jobs. Using a local modelling mechanism and communicating successful jobs, the agents contribute to the improvement of the global network performance.

The distinction between organisational and environmental learning, and the fact that the strategic impact of both forms of learning might be different, is very important: if an organisation wants to learn the handling of tasks as defined by an external environment, its complexity will increase. Instead, if the organisation learns its own structure, the complexity might decrease.

We believe that our framework is useful for the analysis of problems in the optimisation of agent organisations. Experiments have shown that extreme efficiency is at times counterproductive. Sometimes it is useful to blow a bit of random noise through the organisation to discover where the real bottlenecks are. The theoretical results in this chapter (theorem 2) seem to corroborate this insight.

The workbench can be used to investigate issues in related fields of research: Agents that make local decisions to handle and dispatch reduced task series pose new research questions in the field of routing and load balancing. Developing different agent strategies and using them in simulations running in the workbench, allows one to study how local modelling can lead to robust behaviour and global performance optimisation. With the framework and workbench we hope to set some first steps in a new research direction.

Future work will be done on continuous workload streams, studying scenarios in which organisations are insufficient, in equilibrium or redundant. We also want to extend the cases with dynamic organisation topologies. This means that we have to think about strategies to create and delete agents and connections on the fly. Furthermore we 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 what conditions an organisation in unstable environments can still learn to handle task structures and optimise its behaviour.