Learning-based diagnosis and repair

Nico Roos

Data Science and Knowledge Engineering
Maastricht University
roos@maastrichtuniversity.nl

Abstract. This paper proposes a new form of diagnosis and repair based on reinforcement learning. Self-interested agents learn locally which agents may provide a low quality of service for a task. The correctness of learned assessments of other agents is proved under conditions on exploration versus exploitation of the learned assessments.
Compared to collaborative multi-agent diagnosis, the proposed learning-based approach is not very efficient. However, it does not depend on collaboration with other agents. The proposed learning based diagnosis approach may therefore provide an incentive to collaborate in the execution of tasks, and in diagnosis if tasks are executed in a suboptimal way.

1 Introduction

Diagnosis is an important aspect of systems consisting of autonomous and possibly self-interested agents that need to collaborate [4–7, 10, 8, 9, 11, 12, 14–18, 20, 19, 29, 30, 21–23, 25, 24, 26–28, 32–34, 37]. Collaboration between agents may fail because of malfunctioning agents, environmental circumstances, or malicious agents. Diagnosis may identify the cause of the problem and the agents responsible [31]. Efficient multi-agent diagnosis of collaboration failures also requires collaboration and requires sharing of information. Agents responsible for collaboration failures may be reluctant in providing the correct information. Therefore it is important to have an incentive to provide the right information. The ability to learn an assessments of other agents without the need to exchange information, may provide such an incentive.

This paper addresses the learning of a diagnosis in a network of distributed services. In such a network, tasks are executed by multiple agents where each agent does a part of the whole task. The execution of a part of a task will be called a service.

The more than 2000 year old silk route is an example of a distributed network of services. Local traders transported silk and other goods over a small part of the route between China and Europe before passing the goods on to other traders. A modern version of the silk route is a multi modal transport, which can consists of planes, trains, trucks and ships. Another example of distributed services is the computational services on a computer network. Here, the processing of data are the distributed services. In smart energy networks, consumers of energy may also

be producers of energy. The energy flows have to be routed dynamically through the network. A last example of a distributed service is Industry 4.0. In Industry 4.0, the traditional sequential production process is replaced by products that know which production steps (services) are required in their production. Each product selects the appropriate machine for the next production step and tells the machine what is should do.

To describe a network of distributed services such that diagnosis can be performed, we propose a directed graph representation. An arc of the graph represents the provision of a service by some agent. The nodes are the points where a task[1] is transferred from one agent to another. Incorrect task executions are modeled as transitions to special nodes.

The assumption that agents are self-interested and no agent has a global view of the network, limits the possibility of diagnosis and repair. We will demonstrate that it is still possible to learn which agents are reliable w.r.t. the quality of service that they provide.

The remainder of the paper is organized as follows. In the next section, we will present our graph-based model of a network of distributed services. Section 3 presents an algorithm for locally learning the reliability of agents providing services. Section 4 presents the experimental results and Section 5 concludes the paper.

2 The model

We wish to model a network of services provided by a set of agents. The services provided by the agents contribute to the executions of tasks. The order of the services needed for a task need not be fixed, nor the agents providing the services. This suggests that we need a model in which services cause state transitions, and in each state there may be a choice between several agent-service combinations that can provide the next service. The service that is provided by an agent may be of different quality levels. We can model this at an abstract level by different state transitions. If we also abstract from the actual service descriptions, then we can use a graph based representation.

We model a network of services provided by a set of agents Ag using a graph $G = (N, A)$, where the N represents a set of nodes and $A = \{(n_i, n_i', ag_i) \mid \{n_i, n_i'\} \subseteq N, ag_i \in Ag\}_{i=1}^{|A|}$ set of arcs. Each arc $(n, n', ag) \in A$ represents a service (n, n') that is provided by an agent $ag \in Ag$. We allow for multiple services between two nodes provided that the associated agents are different; i.e., several agents may provide the same service.

A set of tasks T is defined by pairs of nodes $(s, d) \in T$. Any path between the source s and the destination d of a task $(s, d) \in T$; i.e., a path $(a_1, \ldots, a_k)$ with $a_i = (n_i, n_{i+1}, ag_i)$, $n_1 = s$ and $n_{k+1} = d$, represents a correct execution of the task.

An incorrect execution of a task $(s, d) \in T$ is represented by a path that ends in a node d' not equal to d; i.e., a path $(a_1, \ldots, a_k)$ with $a_i = (n_i, n_{i+1}, ag_i)$,

[1] In smart energy networks the tasks are the directions in which energy must flow.

$n_1 = s$ and $n_{k+1} = d' \neq d$. A special node f is used to denote the complete failure of a service provided by an agent. No recovery from f is possible and no information about this failure is made available.

To describe a sub-optimal execution of a task $(s, d) \in T$, we associate a set of special nodes with each destination node d. These nodes indicate that something went wrong during the realization of the task. For instance, goods may be damaged during the execution of a transport task. The function $D : N \to 2^N$ will be used for this purpose. Beside the nodes denoting suboptimal executions, we also include the normal execution; i.e., $d \in D(d)$. Moreover, $f \in D(d)$.

To measure the quality of the execution of a task $(s, d) \in T$, we associate a utility with every possible outcome of the task execution: $U(d', d)$ for every $d' \in D(d)$. Here, $U(f, d) \leq U(d', d) < U(d, d)$ for every $d' \in D(d) \backslash d$.

The possible results of a service provided by agent ag in node n for a task $t = (s, d)$ with destination d, will be specified by the function $E(n, d, ag)$. This function $E : N \times N \times Ag \to 2^N$ specifies all nodes that may be reached by the provided service. The function must satisfy the following requirements:

- $E(n, d, ag) \subseteq \{n'' \mid (n, n'', ag) \in A\}$

We also define a probability distribution $e : N \times N \times Ag \times N \to [0, 1]$ over $E(n, d, ag)$, describing the probability of every possible outcome of the provided service; i.e.,

- $e(n, d, ag, n') = P(n' \mid n, d, ag)$
  where $n' \in E(n, d, ag)$ and $\sum_{n' \in E(n,d,ag)} e(n, d, ag, n') = 1$.

There may be several agents in a node n that can provide the next service for a task $t = (s, d)$ with destination d. The function $succ : N \times N \to 2^{Ag}$ will be used to denote the set of agents $succ(n, d) = \{ag_1, \ldots, ag_k\}$ that can provide the next service.

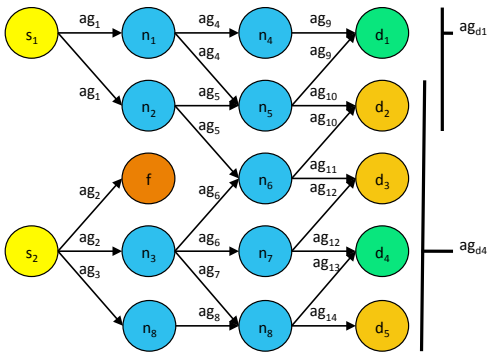

Fig. 1. An example network.

Figure 1 gives an illustration of a network of services represented as a graph. The network shows two starting nodes for tasks, s_1 and s_2, two successful desti-

nation nodes for tasks, d_1 and d_4, two unsuccessful destination nodes for tasks, d_2 and d_3, the failure node f and seven intermediate nodes.

3 Distributed learning of agent reliability

Agents may learn locally diagnostic information using feedback about the result of a task execution. The diagnostic information learned by each agent may enable it to pass on a task in a node to a next agent such that the task is completed in the best possible way. So, an agent must learn the reputation of the agents to which it passes on tasks. This reputation may depend on the node in which the coordination with the next agent takes place as well as on all future agents that will provide services for the task.

We could view our model of a network of services provided by agents as a Markov Decision Process (MDP) [1, 13]. In this markov decision process the nodes in $D(d)$ given the task (s, d), are absorbing states. Only when reaching a node in $D(d)$ a reward is received. All other rewards are 0. The transition probabilities are given by $e(n, d, ag, n')$. If these probabilities do not depend on the destination; i.e., $e(n, d, ag, n') = P(n' \mid n, ag)$, then we have a standard markov decision process for which the optimal policy can be learned using Q-learning [35, 36]. However, Q-learning requires that an agent providing a service knows the Q-values of the services the next agent may provide. This implies that we have a Decentralized MDP [2, 3] in which collaboration is needed to learn the optimal Q-values of services. If agents are willing to collaborate, it is, however, more efficient to use the traditional forms of diagnosis [31]. Therefore, in this section, we assume the agents are self-interested and do not collaborate.

To enable local learning of the agents' reputations, we assume that for every task $t = (s, d) \in T$ one and the same agent ag_d is associated with all nodes in $D(d) \backslash f$. Moreover, we assume that each agent that provided a service for the task execution, has added its signature to the task. The incentive for adding a signature is the payment for the provided service. The agent ag_d uses these signatures to make the payments and to inform the agents that provided a service about the success of the whole task execution. The latter information enables each service agent to assess the quality of the agents to which it passes on tasks. If the payments depend on the quality of service of the whole chain, the agents providing services will have an incentive to provide the best possible service and to pass on a task to a next agent such that the quality of the next services is maximized.

An agent ag that provided a service must pass on task $t = (s, d) \in T$ to the next agent if the task is not yet finished. There may be k agents that can provide the next service: $ag_1, \ldots, ag_k$. Assuming that agent ag can identify the current node n and thereby the quality of its own service, ag would like to learn which of the k agents is the most suited to provide the next service for the task.

The agent ag_d associated with the destination d of task $t = (s, d) \in T$ will inform agent ag about the actual quality $d' \in D(d)$ that is realized for the task. This feedback enables agent ag to evaluate the quality of the whole chain of

services starting with a next agent ag_i. So, even if agent ag_i is providing a high quality service, it may not be a good choice if subsequent agents are failing.

An agent ag can learn for each combination of a task destination (the node d) a next agent ag' and the current node n, the probability that the remainder of the task execution will result in the quality $d' \in D(d) \backslash f$. The probability estimate is defined as:

$$pe(d' \mid d, ag', n, i) = \frac{C_{d' \mid i}}{i}$$

where i is the number of times that a task t with destination d is passed on to agent ag' in the node n, and $C_{d' \mid i}$ is the number of times that agent ag_d gives the feedback of d' for task t with destination d.

Agent ag may not receive any feedback if the execution of task t ended in a complete failure, unless agent ag_d knows about the execution of t. In the absence of feedback, agent ag can still learn the probability estimate of a complete failure:

$$pe(f \mid d, ag', n, i) = \frac{C_{f \mid i}}{i}$$

where $C_{f \mid i}$ is the number of times that no feedback is received from agent ag_d. An underlying assumption is that agent ag_d always gives feedback when a task is completed, and that the communication channels are failure free.

Estimating the probability is not enough. The behavior of future agents may change over time thereby influencing the probability estimates $pe(d' \mid d, ag', n, i)$. Assuming that the transition probabilities $e(n, n', ag, n'')$ of provided services do not change over time, the coordination between agents when passing on tasks is the only factor influencing the probability estimate $pe(d' \mid d, ag', n, i)$. Since agents have an incentive to select the best possible next agent when passing on a task, we need to address the effect of this incentive on the probability estimates. *First, however, we will investigate the question whether there exist an optimal policy for passing on a task to a next agent and a corresponding probability* $P(d' \mid d, ag', n, i)$.

To answer the above question, utilities of task executions are important. With more than two possible outcomes for a task execution, i.e., $|D(d)| > 2$, the expected utility of a task execution needs to be considered. Therefore, we need to know the utility $U(d', d)$ of all outcomes $d' \in D(d)$. We assume that either this information is global knowledge or that agent ag_d provides this information in its feedback.

Using the utilities of task outcomes, we can prove that there exists an optimal policy for the agents, and corresponding probabilities.

Proposition 1. *Let ag be an agent that has to choose a next agent ag' to provide a service for the task $t = (s, d) \in T$ in node n. Moreover, let $P(d' \mid ag', d, n)$ be the probability of reaching d' given the policies of the succeeding agents.*

The utility $U(d, ag, n)$ an agent ag can realize in node n for a task t with destination d, is maximal *if* every *agent ag chooses a next agent ag^* in* every *node n in which it can provide a service, such that the term $\sum_{d' \in D(d)} P(d' \mid ag^*, d, n) \cdot U(d', d)$ is maximal.*

Proof. Given a task $t = (s, d) \in T$ we wish to maximize the expected utility agent ag can realize in node n by choosing the proper next agent to provide a service for the task.

$$\begin{aligned} U(d, ag, n) &= \sum_{d' \in D(d)} P(d' \mid d, n) \cdot U(d', d) \\ &= \sum_{d' \in D(d)} \sum_{ag'} P(d' \mid ag', d, n) \cdot P(ag' \mid d, n) \cdot U(d', d) \\ &= \sum_{ag'} P(ag' \mid d, n) \cdot \sum_{d' \in D(d)} P(d' \mid ag', d, n) \cdot U(d', d) \end{aligned}$$

Here $P(ag' \mid d, n)$ is the probability that agent ag chooses ag' to be the next agent.

Suppose that the term $\sum_{d' \in D(d)} P(d' \mid ag', d, n) \cdot U(d', d)$ is maximal for $ag' = ag^*$. Then $U(d, ag, n)$ is maximal if agent ag chooses ag^* to be the next agent with probability 1; i.e., $P(ag^* \mid d, n) = 1$. Therefore,

$$U(d, ag, n) = \sum_{d' \in D(d)} P(d' \mid ag^*, d, n) \cdot U(d', d)$$

We can rewrite this equation as:

$$\begin{aligned} U(d, ag, n) &= \sum_{d' \in D(d)} P(d' \mid ag^*, d, n) \cdot U(d', d) \\ &= \sum_{d' \in D(d)} \sum_{n' \in E(n, d, ag^*)} P(d' \mid d, n') \cdot P(n' \mid ag^*, d, n) \cdot U(d', d) \\ &= \sum_{n' \in E(n, d, ag^*)} P(n' \mid ag^*, d, n) \cdot \sum_{d' \in D(d)} P(d' \mid d, n') \cdot U(d', d) \\ &= \sum_{n' \in E(n, d, ag^*)} e(n, d, ag^*, n') \cdot U(d, ag', n') \end{aligned}$$

Here $P(n' \mid ag^*, d, n)$ is the transition probability of the service provided by agent ag^*, and $U(d, ag^*, n') = \sum_{d' \in D(d)} P(d' \mid d, n') \cdot U(d', d)$ is the expected utility agent ag^* can realize in node n' by choosing the proper next agent to provide a service.

We can now conclude that to maximize $U(d, ag, n)$, agent ag must choose the agent ag^* for which the term $\sum_{d' \in D(d)} P(d' \mid ag', d, n) \cdot U(d', d)$ is maximal, and agent ag^* ensures that $U(d, ag^*, n')$ is maximized. This result enables us to prove by induction to the maximum distance to a node $d' \in D(d)$ that for every agent ag, $U(d, ag, n)$ is *maximal* if every agent ag chooses a next agent ag^* for which the term $\sum_{d' \in D(d)} P(d' \mid ag^*, d, n) \cdot U(d', d)$ is maximal.

- *Initialization step* Let the current node be $d' \in D(d)$. Then the maximum distance is 0 and the current agent is the agent ag_d receiving the result of the task. So, $U(d, ag_d, d') = U(d', d)$.

– *Induction step* Let $U(d, ag_d, n')$ be maximal for all distances less than k. Let n be a node such that the maximum distance to a node in $D(d)$ is k. Then according to the above result, $U(d, ag, n)$ is maximal if agent ag chooses a next agent ag^* for which the term $\sum_{d' \in D(d)} P(d' \mid ag^*, d, n) \cdot U(d', d)$ is maximal, and for every $n' \in E(n, d, ag^*)$, $U(d, ag^*, n')$ is maximal. The former condition holds according to the prerequisites mentioned in the proposition. The latter condition holds according to the *induction hypothesis*. Therefore, the proposition holds.

□

The proposition shows that there exists an optimal policy for the agents, namely choosing the next agent for which the expected utility is maximized. *The next question is whether the agent can learn the information needed to make this choice.* That is, for every possible next agent, the agent must learn the probabilities of every value in $D(d)$ for a task $t = (s, d) \in T$ with destination d. Since these probabilities depend on the following agents that provide services, the optimal probabilities, denoted by the superscript $*$, can only be learned if these agent have learned to make an optimal choice. So, each agent needs to balance *exploration* (choosing every next agent infinitely many times in order to learn the optimal probabilities) and *exploitation* (choosing the best next agent). We therefore propose the following requirements

– Every agent ag uses a probability $P_i(ag' \mid d, n)$ to choose a next agent ag' for the task with destination d. The index i denotes that this probability depends on the number of times this choice has been made till now.
– The probability $P_i(ag' \mid d, n)$ that agent ag will choose agent ag' of which the till now learned expected utility is sub-optimal, approximates 0 if $i \to \infty$.
– $\sum_{i \to \infty} P_i(ag' \mid d, n) = \infty$

The first requirement states that we use a probabilistic exploration. The second requirement ensures that the agent will eventually only exploit what it has learned. The third requirement ensures that the agent will select every possible next agent infinitely many times in order to learn the correct probabilities.

A policy meeting the requirements is the policy in which the agent ag chooses the currently optimal next agent ag' with probability $1 - \frac{1}{(k-1)i}$. Here, k is the number of agents that can perform the next service for a task with destination d, and i is the number of times agent ag has to choose one of these k agents for a task with destination d. The agents that are currently not the optimal choice are chosen with probability $\frac{1}{(k-1)i}$.

We can prove that any approach meeting the above listed requirements will enable agents to learn the optimal policy.

Theorem 1. *Let every agent ag meet the above listed requirements for the probability $P_i(ag' \mid d, n)$ of choosing the next agent. Moreover, let $P^*(d' \mid ag, d, n)$ be the optimal probability of reaching the node $d' \in D(d)$ if every agent chooses a next agent ag^* for which the term $\sum_{d' \in D(d)} P(d' \mid ag^*, d, n) \cdot U(d', d)$ is maximal.*

Then, every agent ag learns $P^*(d' \mid ag', d, n)$ *through* $pe(d' \mid ag', d, n, i)$ *if the number of tasks with destination* d *for which agent* ag *has to choose a next agent* ag'*, denoted by* i*, goes to infinity.*

Proof. We have to prove that: $\lim_{i\to\infty} pe(d' \mid ag', d, n, i) = P^*(d' \mid ag', d, n)$.

We can rewrite $\lim_{i\to\infty} pe(d' \mid ag, d, n, i)$ as:

$$\begin{aligned}
\lim_{i\to\infty} pe(d' \mid ag', d, n, i) &= \lim_{i\to\infty} \frac{C_{d' \mid i}}{i} \\
&= \lim_{i\to\infty} \sum_{n'\in E(n,d,ag')} \frac{C_{n' \mid i}}{i} \cdot \frac{C_{d' \mid C_{n' \mid i}}}{C_{n' \mid i}} \\
&= \lim_{i\to\infty} \sum_{n'\in E(n,d,ag')} pe(n' \mid ag', d, n, i) \cdot \frac{C_{d' \mid C_{n' \mid i}}}{C_{n' \mid i}} \\
&= \sum_{n'\in E(n,d,ag')} P(n' \mid ag', d, n) \cdot \lim_{i\to\infty} \frac{C_{d' \mid C_{n' \mid i}}}{C_{n' \mid i}}
\end{aligned}$$

We will prove that $C_{n' \mid i} \to \infty$ if $i \to \infty$ and $P(n' \mid ag', d, n) > 0$. That is, for every $x \in \mathbb{N}$, $\lim_{i\to\infty} P(C_{n' \mid i} > x) = 1$.

$$\begin{aligned}
\lim_{i\to\infty} P(C_{n' \mid i} > x) &= \lim_{i\to\infty} 1 - P(C_{n' \mid i} \leq x) \\
&= 1 - \lim_{i\to\infty} \sum_{j=0}^{x} (P(n' \mid ag', d, n))^j \cdot (1 - P(n' \mid ag', d, n))^{i-j} \\
&= 1
\end{aligned}$$

So, $C_{n' \mid i} \to \infty$ if $i \to \infty$. Therefore,

$$\lim_{i\to\infty} pe(d' \mid ag', d, n, i) = \sum_{n'\in E(n,d,ag')} P(n' \mid ag', d, n) \cdot \lim_{j\to\infty} pe(d' \mid d, n', j)$$

The estimated probability $pe(d' \mid d, n', j)$ depends on the probability of choosing the next agent. This probability is a function of the j-th time agent ag' must choose a next agent ag'' for a task with destination d in node n'.

$$\lim_{j\to\infty} pe(d' \mid d, n', j) = \lim_{j\to\infty} \sum_{ag''\in succ(n',d)} P_j(ag'' \mid d, n') \cdot \frac{C_{d' \mid ag'',j}}{C_{ag'' \mid j}}$$

where $C_{ag'' \mid j}$ is the number of times that agent ag'' was chosen to be the next agent, and $C_{d' \mid ag'',j}$ is the number of times that subsequently node d' was reached.

We will prove that $C_{ag'' \mid j} \to \infty$ if $j \to \infty$ and $P_j(ag'' \mid d, n) > 0$ for every j. That is, for every $x \in \mathbb{N}$, $\lim_{i\to\infty} P(C_{ag'' \mid j} > x) = 1$. A complicating factor is that $P_j(ag' \mid d, n)$ can be different for every value of j. Let y be the index of

the last time agent ag'' is chosen, and let p_x be the probability of all possible sequences till index y. Then we can formulate:

$$\begin{aligned}\lim_{j\to\infty} P(C_{ag''\mid j} > x) &= \lim_{j\to\infty} 1 - P(C_{ag''\mid j} \leq x) \\ &= 1 - p_x \cdot \lim_{j\to\infty} \prod_{k=y+1}^{j} (1 - P_k(ag'' \mid d, n)) \\ &= 1 - e^{\ln(p_x) + \sum_{k=y+1}^{\infty} \ln(1 - P_k(ag''\ \mid\ d,n))}\end{aligned}$$

According to the Taylor expansion of $\ln(\cdot)$: $\ln(1-P_k(ag'' \mid d, n)) < -P_k(ag' \mid d, n)$. Therefore,

$$\begin{aligned}\lim_{j\to\infty} P(c_{ag''\mid j} > x) &= 1 - e^{\ln(p_x) - \sum_{k=y+1}^{\infty} P_k(ag''\ \mid\ d,n)} \\ &= 1 - e^{\ln(p_x) - \infty} \ = \ 1\end{aligned}$$

The above result implies:

$$\lim_{j\to\infty} pe(d' \mid d, n', j) = \lim_{j\to\infty} \sum_{ag''\in succ(n',d)} P_j(ag'' \mid d, n') \cdot \lim_{k\to\infty} pe(d' \mid ag'', d, n', k)$$

We can now prove the theorem by induction to the maximum distance to a node $d' \in D(d)$.

- *Initialization step* Let the current node be $d' \in D(d)$. The maximum distance is 0 and the current agent is the agent ag_d receiving the result of the task. So, $\lim_{i\to\infty} pe(d' \mid ag_d, d, d', i) = P^*(d' \mid ag_d, d, d') = 1$.
- *Induction step* Let $\lim_{j\to\infty} pe(d' \mid ag', d, n', j) = P^*(d' \mid ag', d, n')$ be maximal for all distances less than k. Moreover, let the maximum distance from n to d' be k.
  Then, the expected utility of agent $ag'' \in succ(n', d)$ is:

$$\begin{aligned}\lim_{j\to\infty} U_j(ag'', d, n') &= \lim_{j\to\infty} \sum_{d'\in D(d)} pe(d' \mid ag'', d, n', j) \cdot U(d', d) \\ &= \sum_{d'\in D(d)} P^*(d' \mid ag', d, n') \cdot U(d', d) = U^*(ag'', d, n')\end{aligned}$$

According to the requirement,

$$\lim_{j\to\infty} P_j(ag_j^* \mid d, n') = 1 \text{ for } ag_j^* = \text{argmax}_{ag''} U_j(ag'', d, n')$$

So,

$$\begin{aligned}ag^* &= \lim_{j\to\infty} ag_j^* \\ &= \lim_{j\to\infty} \text{argmax}_{ag''} U_j(ag'', d, n') \\ &= \text{argmax}_{ag''} U^*(ag'', d, n')\end{aligned}$$

This implies:

$$\begin{aligned}\lim_{j\to\infty} pe(d' \mid d, n', j) &= \lim_{j\to\infty} \sum_{ag''\in succ(n',d)} P_j(ag'' \mid d, n) \cdot \lim_{k\to\infty} pe(d' \mid ag'', d, n', k) \\ &= \sum_{ag''\in succ(n',d)} P^*(d' \mid ag'', d, n') \cdot \lim_{j\to\infty} P_j(ag'' \mid d, n) \\ &= P^*(d' \mid ag^*, d, n') \;\; = \;\; P^*(d' \mid d, n')\end{aligned}$$

Therefore,

$$\begin{aligned}\lim_{i\to\infty} pe(d' \mid ag', d, n, i) &= \sum_{n'\in E(n,d,ag')} P(n' \mid ag', d, d) \cdot \lim_{j\to\infty} pe(d' \mid d, n', j) \\ &= \sum_{n'\in E(n,d,ag')} e(n, d, ag', n') \cdot P^*(d' \mid d, n') \\ &= P^*(d' \mid ag', d, n)\end{aligned}$$

□

The theorem shows us that each agent can learn which next agent results in an expected high or low quality for the remainder of a task. In order to learn this assessment, the agents must explore all possible choices for a task infinitely many times. At the same time the agents may also exploit what they have learned sofar. In the end the agents will only exploit what they have learned. Hence, the learning-based approach combines *diagnosis and repair*.

An advantage of the learning-based approach is that intermitting faults can be addressed and that no collaboration between service agents is required. A disadvantage is that making diagnosis requires information about many executions of the same task. However, as we will see in the next section, a repair is learned quickly at the price that correctly functioning agents may be ignored.

Agents learn an assessment for each possible destination. In special circumstances, they need not consider the destination, and can focus on the next agent that can provide a service for a task. First, the quality of service provided by an agent does not depend on the destination of the task. Second, we do not use utilities for the result of a task and only identify whether a task execution is successful. If these conditions are met, an agent can learn for every next agent the probability that the task execution will be successful.

4 Experiments

To determine the applicability of the theoretical results of the previous section, we ran several experiments. For the experiments, we used a network of n^2 normal nodes organized in n layers of n nodes. Every normal node in a layer, except the last layer, is connected to two normal nodes in the next layer. Moreover, from every normal node in de first layer, every normal node in the last layer can be reached. With every transition a different agent is associated. To model

that these agents may provide a low quality of service, for every transition from normal node n to normal node n' representing the correct execution of a service by an agent, there is also a transition from n to an abnormal node n'' representing the incorrect execution of the service. Here, the abnormal node n'' is a duplicate of the normal node of n'. For every normal node except the nodes in the first layer, there is a duplicate abnormal node denoting the sub-optimal execution of a service. In this model, no recovery is possible. Figure 2 show a 4 by 4 network. The normal nodes that can be used for a normal execution of tasks are shown in yellow, blue and green. The duplicate abnormal nodes representing a sub-optimal execution are shown in orange. The transitions to the latter nodes and the transitions between the latter nodes are not shown in the figure.

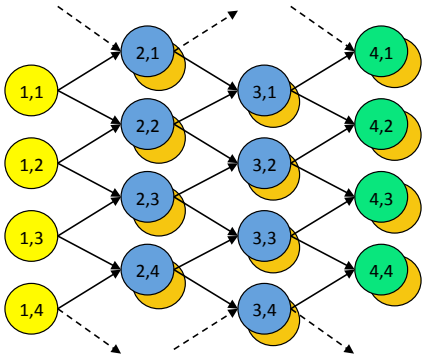

Fig. 2. The network used in the experiments. Note that the dashed arrows denote transitions from nodes (1,4), (2,1) and (3,4) to nodes (2,1), (3,4) and (4,1) respectively.

In our first experiment we determined how often a randomly chosen service is executed in 10000 randomly chosen tasks. We used a network of 10 by 10 nodes in this experiment. Figure 3 shows the cumulative results as a function of the number of processed task. Figure 4 shows in which experiment the service is used.

In the second experiment we used the same network. A fault probability of 0.1 was assigned to the randomly chosen service. Again, we measured how often a service is executed in 10000 randomly chosen tasks. Figure 5 shows the cumulative results as a function of the number of experiments, and Figure 6 shows in which task the service is executed. We clearly see that the agents learn to avoid the agent that provides a low quality of service.

The results show that each agent learns to avoid passing on a task to an agent that may provide a low quality of service. An agent uses the estimated probabilities of a successful completion of a task when passing on the task to the next agent. Nevertheless, as shown in Figure 6, the agents still try the low quality service, but with an increasingly lower probability. This exploration is necessary to learn the correct probabilities.

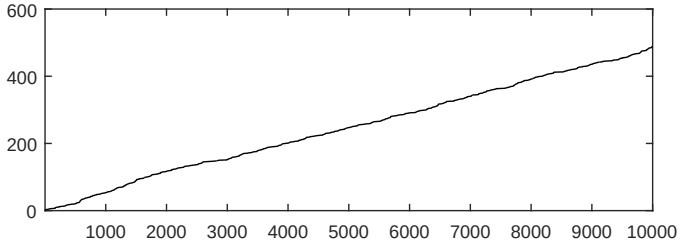

Fig. 3. The number of times a selected service is chosen as a function of the number of processed task.

Fig. 4. The tasks in which a selected service is chosen.

Inspection of the learned probabilities shows that the learning process is slow w.r.t. the total number of executed tasks. Figure 7 shows the learning of the probability that choosing an agent in a node n will result in a good quality of service for a task with a specific destination d. The probability that must be learned is 0.5. The agents only learn when they provided a service for a task with destination d. In Figure 7, the service is executed only 4 times for tasks with destination d of 10000 executions of randomly chosen task. Although the learning process is slow, it is not a problem for the behavior of the network of distributed services. However, it does result in avoiding the services provided by some agents while there is no need for it.

In the third experiment we learned the probability that choosing an agent will result in a good quality of service for a task, independent of the destination of the task. Figure 8 shows the result of the learning process. Again the probability that must be learned is 0.5. The learning process is much faster. However, as

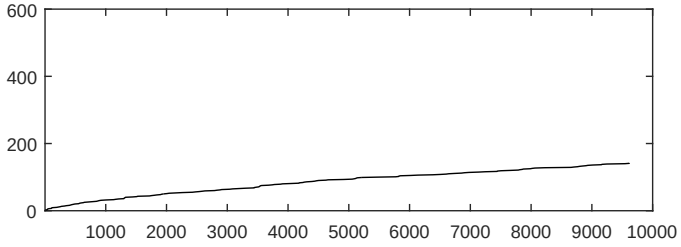

Fig. 5. The number of times a selected service is chosen as a function of the number of processed task.

Fig. 6. The tasks in which a selected service is chosen.

discussed at the end of the previous section, ignoring the destination of a task is only possible if the quality of service does not depend on the destination, and if we only identify whether a task is successful.

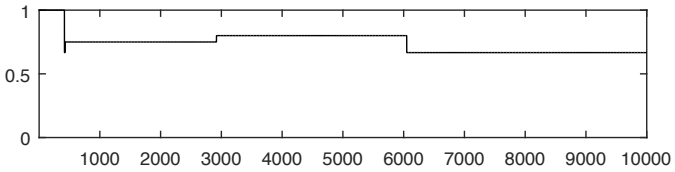

Fig. 7. Learning of the service success probability given a destination.

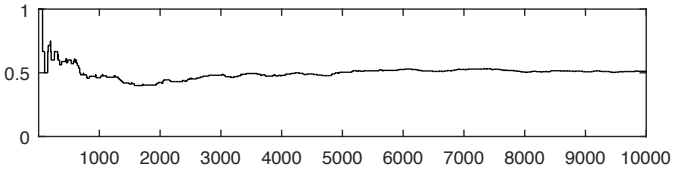

Fig. 8. Learning of the service success probability ignoring the destination.

5 Conclusions

This paper presented a model for describing a network of distributed services for task executions. Each service is provided by an autonomous, possibly self-interested agent. The model also allows for the description of sub-optimal and failed services.

When a task is completed with a low quality, we would like to determine which service was of insufficient quality, which agent was responsible for the provision of this service, and how we can avoid agents that might provide a low quality of service. To answer these questions, the paper investigated an approach for learning in a distributed way an assessment of other agents. The learned information can be exploited to maximize the quality of a task execution. The

correctness of the learned diagnosis an repair approach is proved, and demonstrated through experiments.

An important aspect of the distributed learning approach is that agents do not have to collaborate. Since diagnosis of distributed services is about identifying the agents that are to blame for a low quality of service, this is an important property. It provides an incentive for being honest if agents make a diagnosis in a collaborative setting. Systematic lying will be detected eventually.

This research opens up several lines of further research. First, other policies that balance exploration and exploitation could be investigated. Second, more special cases in which the learning speed can be improved should be investigated. The topology might, for instance, be exploited to improve the learning speed. Third, since agents learn to avoid services of low quality before accurately learning the corresponding probabilities, we may investigate whether we can abstract from the actual probabilities. Fourth, as mentioned in the Introduction and above, the learned assessments provide an incentive for honesty when agents make a collaborative diagnosis. Is this incentive sufficient for agents to collaborate if traditional diagnostic techniques are used?

References

1. R. Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, 6:679–684, 1957.
2. D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. Sequential optimality and coordination in multiagent systems. *Mathematics of Operations Research*, 6(4):819–840, 2002.
3. C. Boutilier. Sequential optimality and coordination in multiagent systems. In *IJCAI*, pages 478–485, 1999.
4. L. Console and P. Torasso. Hypothetical reasoning in causal models. *International Journal of Intelligence Systems*, 5:83–124, 1990.
5. L. Console and P. Torasso. A spectrum of logical definitions of model-based diagnosis. *Computational Intelligence*, 7:133–141, 1991.
6. R. Davis. Diagnostic reasoning based on structure and behaviour. *Artificial Intelligence*, 24:347–410, 1984.
7. F. de Jonge and N. Roos. Plan-execution health repair in a multi-agent system. In *PlanSIG 2004*, 2004.
8. F. de Jonge, N. Roos, and H. Aldewereld. Multiagent system technologies. In *Multiagent System Technologies*, 2007.
9. F. de Jonge, N. Roos, and H. Aldewereld. Temporal diagnosis of multi-agent plan execution without an explicit representation of time. In *BNAIC-07*, 2007.
10. F. de Jonge, N. Roos, and H.J. van den Herik. Keeping plan execution healthy. In *Multi-Agent Systems and Applications IV: CEEMAS 2005, LNCS 3690*, pages 377–387, 2005.
11. F. de Jonge, N. Roos, and C. Witteveen. Diagnosis of multi-agent plan execution. In *Multiagent System Technologies: MATES 2006, LNCS 4196*, pages 86–97, 2006.
12. F. de Jonge, N. Roos, and C. Witteveen. Primary and secondary plan diagnosis. In *The International Workshop on Principles of Diagnosis, DX-06*, 2006.
13. R. A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, 1960.

14. M. Kalech and G. A. Kaminka. On the design of social diagnosis algorithms for multi-agent teams. In *IJCAI-03*, pages 370–375, 2003.
15. M. Kalech and G. A. Kaminka. Diagnosing a team of agents: Scaling-up. In *AAMAS 2005*, pages 249–255, 2005.
16. M. Kalech and G. A. Kaminka. Towards model-based diagnosis of coordination failures. In *AAAI 2005*, pages 102–107, 2005.
17. M. Kalech and G. A. Kaminka. On the design of coordination diagnosis algorithms for teams of situated agents. *Artificial Intelligence*, 171:491–513, 2007.
18. M. Kalech and G. A. Kaminka. Coordination diagnostic algorithms for teams of situated agents: Scaling up. *Computational Intelligence*, 27(3):393–421, 2011.
19. J. de Kleer, A.K. Mackworth, and R. Reiter. Characterizing diagnoses and systems. *Artificial Intelligence*, 56:197–222, 1992.
20. J. de Kleer and B. C. Williams. Diagnosing with behaviour modes. In *IJCAI 89*, pages 104–109, 1989.
21. R. Micalizio. A distributed control loop for autonomous recovery in a multi-agent plan. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence, (IJCAI-09)*, pages 1760–1765, 2009.
22. R. Micalizio. Action failure recovery via model-based diagnosis and conformant planning. *Computational Intelligence*, 29(2):233–280, 2013.
23. R. Micalizio and P. Torasso. On-line monitoring of plan execution: A distributed approach. *Knowledge-Based Systems*, 20:134–142, 2007.
24. R. Micalizio and P. Torasso. *Plan Diagnosis and Agent Diagnosis in Multi-agent Systems*, pages 434–446. Springer, 2007.
25. R. Micalizio and P. Torasso. Team cooperation for plan recovery in multi-agent systems. In *Multiagent System Technologies, LNCS 4687*, pages 170–181, 2007.
26. R. Micalizio and P. Torasso. Monitoring the execution of a multi-agent plan: Dealing with partial observability. In *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI-08)*, pages 408–412. IOS Press, 2008.
27. R. Micalizio and P. Torasso. Cooperative monitoring to diagnose multiagent plans. *Journal of Artificial Intelligence Research*, 51:1–70, 2014.
28. R. Micalizio and G. Torta. Explaining interdependent action delays in multiagent plans execution. *Autonomous Agents and Multi-Agent Systems*, 30(4):601–639, 2016.
29. O. Raiman, J. de Kleer, V. Saraswat, and M. Shirley. Characterizing non-intermittent faults. In *AAAI 91*, pages 849–854, 1991.
30. R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.
31. N. Roos, A. ten Teije, and C. Witteveen. Reaching diagnostic agreement in multi-agent diagnosis. In *AAMAS 2004*, pages 1254–1255, 2004.
32. N. Roos and C. Witteveen. Diagnosis of plan execution and the executing agent. In *Advances in Artificial Intelligence (KI 2005), LNCS 3698*, pages 161–175, 2005.
33. N. Roos and C. Witteveen. Diagnosis of plan structure violations. In *Multiagent System Technologies*, 2007.
34. N. Roos and C. Witteveen. Models and methods for plan diagnosis. *Journal of Autonomous Agents and Multi-Agent Systems*, 19:30–52, 2008.
35. C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, 1989.
36. C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.
37. C. Witteveen, N. Roos, R. van der Krogt, and M. de Weerdt. Diagnosis of single and multi-agent plans. In *AAMAS 2005*, pages 805–812, 2005.