Distributed Fault Detection For Multi-Agent Systems Based On Vertebrate Foraging

Sebastian Schmid
Friedrich-Alexander-Universität Erlangen-Nürnberg
Nuremberg, Germany
sebastian.schmid@fau.de

Andreas Harth
Friedrich-Alexander-Universität Erlangen-Nürnberg
Fraunhofer IIS, Fraunhofer Institute for Integrated Circuits
Nuremberg, Germany
andreas.harth@iis.fraunhofer.de

ABSTRACT

We present our distributed algorithm to detect malfunctioning units in multi-agent groups based on how social vertebrates, like chimpanzees, forage in groups for food. Agents with our algorithm use only the cooperation's outcome and limited communication range to form adaptive groups for a given task and to identify a malfunctioning member. We evaluate our algorithm with simulated experiments in several setups where distributed agents have to form groups to achieve a given task. One agent of the agent population is malfunctioning and shall be identified. We measure the time for the agent population to form successful groups as well as the detection rate of the malfunctioning member. We conclude that our algorithm can detect the error on average with 98% in our scenarios.

KEYWORDS

Distributed Fault Detection, Adaption, Resilience, Stigmergy

1 INTRODUCTION

Distributed agent-based systems offer scalability, graceful degradation, and adaptivity and do not need a central communication medium [5]. Agent-based systems, in form of autonomous machines, are already used in the industrial context because of their adaptivity and resilience e.g. for manufacturing or transportation [25]. In terms of scalability, multiple agents may combine their power to fulfill a shared task, e.g. lifting a heavy box together, where we say that cooperation among agents happens. Jennings [15] defines such collective structures of agents as subsystems and "nothing more than a team of components working together to achieve a collective goal". Such cooperations can also be solved in a distributed fashion with partial global planning [8].

Cooperation in industrial scenarios between several agents of a multi-agent system offers performances that orient on the actual problem according to the possibilities of the agents, as only as many resources are used as needed [9]. But to achieve the promised performance of cooperation in form of groups, all involved agents have to be able to participate without any excessive performance loss, or else the cooperating group cannot achieve the expected outcome. Wear and internal faults may lead to performance loss as natural processes over time, which threatens the performance of the agent population system if its parts do not work as intended.

To detect failing agents, the literature favors time-intensive, regular checks or omniscient systems that evaluate the status of all agents [24], but such centralized systems suffer from the heavy computational load with more agents to evaluate, require detailed measures and models, and are prone to single points of failure [4, 5]. Such solutions contradict the original idea of distributed, autonomous agents, which raises the question: how can agents detect malfunctioning members in a distributed fashion when cooperating on tasks?

Three main points make the detection of malfunctioning agents difficult if the distributed aspects of the agent-based system shall be preserved:

(1) Fixed assumptions over the agent population have to be reduced such that agents are independent of each other and the population is flexible to be increased or reduced. When agents use only their own, **local observations** to find malfunctioning agents, agents avoid relying on pre-shared, fixed data and use their own judgment.
(2) The assumption of immediate availability to communicate with arbitrary agents via a network for information exchange at will is a fallacy [27]. We consider restricted, **local communication** for agents where only subsets of agents may be reached, but others not. This avoids the requirement to communicate with all agents.
(3) Malfunctioning agents shall be identifiable for other agents. To avoid a blackboard-like, centralized directory, agents themselves shall hold and expose information on their possible failure. These marks act as indirect communication, called **stigmergy** [13]. Stigmergy gives scalability and resilience: every agent saves only its own information. Even if the population changes, no additional memory is needed to save other agents' performance.

We argue that it is important to avoid centralized assumptions and knowledge over the agent population as these would defy the whole purpose of a distributed agent-based systems [15]. We avoid the usage of pre-calculated models that would tell agents exactly how particular members should behave, thus we put an emphasis on using only observations of the outcome of group cooperation. Instead, we used nature as inspiration and implemented an algorithm for distributed agents that fulfills the demands from above: the food foraging of social vertebrates e.g. chimpanzees [3]. Like animals that group together to forage efficiently and reshape their ties to other members of their population to maximize survival chances, similarly, our agents shall maximize the chances to fulfill their task.

We illustrate the problem by introducing the running example of a shop floor, where groups of transporter units need to transport a heavy steel motor housing together.

Example 1.1. *MH1, a motor housing weighing 300kg, needs to be transported. Unit5, a transporter unit nearby, has the task to move*

Proc. of the Adaptive and Learning Agents Workshop (ALA 2023), Cruz, Hayes, Wang, Yates (eds.), May 29-30, 2023, London, UK, https://alaworkshop2023.github.io/. 2023.

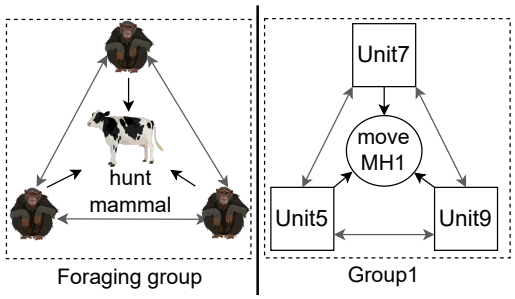

Figure 1: Chimpanzees form a group to hunt a mammal (a resource that cannot be accessed by an individual), and like social vertebrates the transportation units of Ex. 1.1 have to form a group to achieve the task of moving motor housing MH1, which cannot be achieved alone. Chimpanzee and cow image from TogoTV (© 2016 DBCLS TogoTV, CC-BY-4.0).

MH1 and is looking for other units for cooperation. Unit5 has a limited communication range and contacts other units therein. Unit5 perceives units Unit7, Unit9, and Unit13 inside the range and tries to form a group via ties with other units. Unit5 asks Unit7, Unit9, and Unit13 if ties among them exist, and Unit7 and Unit13 confirm, such that now Unit5, Unit7, and Unit13 form a group. The units assert each other that according to their specifications, the group can lift exactly 300kg, which is sufficient to move MH1. Unit5, Unit7, and Unit13 meet at MH1 and start to lift. But MH1 one does not move a bit! Unit5 cannot be sure if it malfunctioned or Unit7 or Unit13 as only the outcome of the cooperation is visible: failure. By random decision, Unit5 puts a mark of failure on Unit13 that this cooperation failed and tries to form another group to solve the task, e.g. Group1 together with Unit7 and Unit9. And indeed, over time more failure marks appear on Unit13 from other attempts to lift motor housings - while all other units formed groups to solve the respective tasks, every time Unit13 was involved, the task failed. Marked by many other units as malfunctioning, a human technician inspects Unit13 and sees that a spring was broken which made moving the motor housing impossible.

Ex. 1.1 shows how units form a group and identify the malfunctioning unit in a distributed fashion, as discussed above:

- Unit5 relies only on **local observation**, the failed outcome of lifting MH1, to judge to outcome of Group1
- Unit5 uses **local communication** and interacts only with units inside the limited communication range
- Unit5 leaves marks of failure as **stigmergy** at the other units that failed to achieve the task. Here, the marks started to accumulate at Unit13 over time

The parallel can be easily seen, if the three units that move a motor housing of Ex. 1.1 are compared to three chimpanzees that hunt a mammal, see Fig. 1: alone, each participant is not strong enough, but as a group they are. Cantor and Farine describe the foraging behavior of social vertebrates in [3] including further influences like birth and death over generations. We extend and adjust their model with a focus on distributed agents that want to achieve their task over several groups and also detect specific agents that are malfunctioning.

We investigate our approach by simulating a population of distributed agents on a shop floor, based on our use case, that shall use our solution to find groups and identify a malfunctioning agent. Furthermore, we restrict the communication among agents, and use direct, limited communication to form groups among agents, and stigmergy [14] (that is communication via placing information in the environment) to pass information about the performance of specific agents coming from observations. Finally, we measure the success rate to detect the malfunctioning agent and the time for agents to find suitable groups.

We summarize our contributions as follows:

- We present a distributed algorithm for agents to form cooperative groups on a given task and identify malfunctioning agents, based on the foraging of vertebrates
- We evaluate the performance of our algorithm in several simulated scenarios with a given task that has to be achieved by groups of agents where a malfunctioning agent is present that hinders the group formation

2 BACKGROUND AND RELATED WORK

2.1 Vertebrate foraging

Cantor and Farine [3] model the emergence of foraging groups in vertebrate societies. They analyze how groups between members of a society evolve when cooperation is necessary to obtain food. They state that groups tend to evolve in a direction that leads to the optimum food supply for all participating members, where group and individual shares are maximized, and resources used ideally. Their model takes several animal-related specialties into account, like mortality, birth, and kinship. We give the most important part of their baseline model here: An agent population of size N wants to access a resource of size R that cannot be accessed by individuals. Ties can be created between agents which form the willingness to forage together, and if the ties are symmetric, the chain leads to a group [26]. All groups compete around access to R. Division among h groups is calculated by $s_k = R \times (n_k^2 / \sum_{j=1}^{h} n_j^2)$ as the group share s of group k with n_k members. An individual i's share r_i is calculated by $r_i = s_k / n_k$. The ties of i are modified according to r_i, building random new ties if $r_i > 1$ during abundance, deleting random old ties if $r_i < 1$ during a shortage, or keeping all ties for optimal outcome $r_i = 1$. New ties among members may be created by chance (0.01% per simulation cycle). Cantor and Farine conclude that self-organization and specialization among members lead quickly to a stable cooperative group of size R, a stepping stone for more sophisticated processes like social learning.

2.2 Intelligent agents and model-based agents

We follow the structures of Russell and Norvig [23] for intelligent agents. We use model-based agents (MBA) as agents that possess internal states, called models, and memorize parts of their environment that cannot be perceived at the moment. Agents may update their model as the world evolves. MBAs decide according to a set of condition-action rules with the help of the model's information and their current perception. The extension to a goal-based agent built on an MBA is straightforward when goals are defined that describe situations that are desirable for the agent.

2.3 Stigmergy and indirect communication

Stigmergy is the use of asynchronous interaction and information exchange between agents exclusively through changes in their environment, but not directly with each other [14]. It is inspired by the indirect communication of insects like termites [10]. Stigmergy is the base for algorithms coming from ants [7] and has widespread usage [28, 29]. The self-organization and coordination of agents are discussed in e.g. [12]. Our agents use indirect communication to talk about the performance of a third one, by attaching information to the third agent. An agent can query this information without the need to contact any of the other agents, but can just see how the performance of an agent is judged and decide on this, which leads to self-organized identification of malfunctioning agents.

2.4 Distributed fault detection and isolation

A succinct overview of the extensive literature on fault detection and isolation can e.g. be found in [18]. Literature focuses mostly on centralized approaches where all data can be freely evaluated to uncover faults in the overall system, which leads to outstanding results with neural networks or model-based approaches [19]. However, for distributed multi-agent systems, centralized approaches can become infeasible [6]. Model-based fault detection for mobile agents can lead to detailed detection and isolation of agents, but reliable, valid physical models of agents and systems are needed, otherwise, estimations have to be used [17]. Bossens et al. [2] discuss several fault detection mechanisms focused on robotic teams.

Shames et al. [24] apply their approach for interconnected second-order systems as an example of a multi-agent robot setting for synchronized formation. They can successfully detect and isolate a malfunctioning node, but as each node needs a separate observer with a state to their neighbors, the approach puts a computational burden on each of the nodes. Davoodi et al. [4] build a homogeneous multi-agent network and apply detector and controller units that evaluate the system state in an observer-controller-based way. Agents have a linear dynamical model they use to achieve simultaneous consensus and fault detection, based on the residual. Despite the distributed nature of the approach, they rely on a model evaluation and a connected network graph. Guo et al. [11] propose algorithms based on communication and sensing to detect faulty and malicious agents. Their approaches are based on the local calculation of deviations and comparison with neighbors by transmitting the agent's own state and together with the neighbors' states. Similarly, Boem et al. [1] rely on decentralized estimators in a heterogeneous multi-agent system, where independent nodes lead dependent nodes. Independent nodes do not communicate with each other but can still detect faulty dependent agents by estimation of fault and control input.

3 APPROACH AND REALIZATION

3.1 Assumptions for our approach

We base our algorithm on Cantor and Farine [3], but while they study the accessibility of a single resource with equally working agents, we modify the approach with the following assumptions:

- Not all agents are assumed to work perfectly in using the resource but fail miserably. Agents cannot directly observe which agent malfunctions but can measure the outcome of their own group that tries to fulfill a shared task.
- All agents intend to fulfill a task R, which cannot be solved alone. Agents have to form a group where each group tries to get as many members such that R is exactly fulfilled.
- Agents limit communication to close neighbors and only extend their communication range if no group can be found.

3.2 Environment and agents

We give the definition of our setup and the individual components:

Definition 3.1 (Floor). The floor F is a space of $m \times n$ floor tiles that represent positions where units are located, where $F = \{f_{(0,0)}, ..., f_{(m,n)}\}$ is connected in Moore neighborhood by $N^M_{f_{(x_0,y_0)}} = \{f_{(x,y)} \in F : |x - x_0| \leq 1, |y - y_0| \leq 1\}$. The Euclidean distance between $f_a, f_b \in F$ is $d(f_a, f_b)$.

Definition 3.2 (Artifacts). Artifacts U are passive, reactive entities that are used by agents as defined in [20]. Artifacts $U = \{u_1, ..., u_N\}$ are defined by the properties $u_i = \langle f \in F, size \in \mathbb{N}^+ \rangle, u_i \in U$. We assume $size < R$ for functioning agents such that multiple agents are necessary to achieve R.

In our running example, artifacts are implemented by transport units. The function of artifacts is given by groups of associated agents (cf. Def. 3.3) in the environment as *performTask* (cf. Def. 3.4).

Definition 3.3 (Foraging Agent). We define our agents as proactive components that make decisions on their own [20]. Agents implement a perception-thought-action cycle to influence their environment, based on defined condition-action rules. We define an agent $a \in \mathcal{A}$ as $a = \langle a_0, \mathcal{M}, perceive, appl, update, act \rangle$ with

$$a_0, \text{ the initial agent state,}$$
$$\mathcal{M} = \langle u \in U, ties \subset \mathcal{A}, R \in \mathbb{N}^+, faultCounter, c \in \mathbb{R} \rangle,$$
$$perceive : \mathcal{N} \times \mathcal{M} \to \mathcal{M}, \mathcal{N} = \{a \in \mathcal{A} \setminus ties | d(f_{u_a}, f_u) \leq c\},$$
$$appl : \mathcal{M} \to \mathcal{M}, \text{derivation of statemens,}$$
$$update : O \times \mathcal{M} \to \mathcal{M}, \text{apply operations to own model,}$$
$$act : Outcome \times \mathcal{M} \to \mathcal{M} \times O, \text{ update model and perform action,}$$

where $\mathcal{M}$ is the agent's model holding relations to the controlled unit u, the ties among agents, its $faultCounter$, and the communication radius c. *Outcome* is the observed performance of groups according to the agent's ties (cf. Def. 3.4). Based on the communication radius, *perceive* uses reachable, neighboring agents to form new ties *act* realizes updates to the agent's own model (e.g. to change the communication range), and actions to influence agents in the *ties* via operations O, where we use the single operation $O = \{increaseFaultCounter\}$. Cooperation among agents can only be formed by reciprocal, symmetric ties, where the connected components of more than one agent define groups, as in [3].

Definition 3.4 (Environment). We define the environment $E = \langle state_0, State, performTask \rangle$ as all scenario components with

$$state_0, \text{ the environment's initial state,}$$
$$State = F \times \mathcal{A} \times U, \text{ the set of all possible component states,}$$
$$performTask : U \times \mathcal{A} \to Outcome, \text{ the groups' performances.}$$

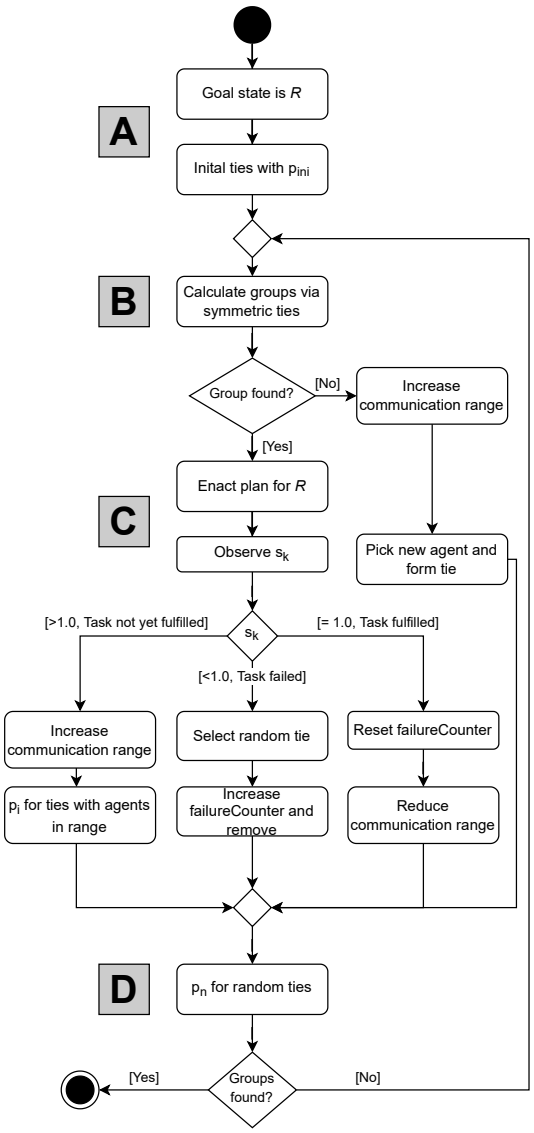

Figure 2: UML activity diagram of the agent's behavior with the presented fault detection algorithm

performTask returns the outcome of a group as defined by the agents in $\mathcal{A}$ and their ties. Agents are situated in an environment after [23] that is partially observable (relevant information might be missed), dynamic (it may change while agents deliberate), and discrete (in terms of time and states).

3.3 Cooperation of agents

We follow Panzarasa et al.'s [21] definition of agent cooperation, consisting of:

- the practical basis: what an agent wants to achieve
- the practical problem: what to do to fulfill an intention
- the recognition of the potential for cooperation: the identification by an agent of an opportunity to collaborate with one or more agents on the resolution of a practical problem

Using [21] as a starting point to detect and understand how to collaborate, we integrate our vertebrate algorithm: obviously, an agent intends to achieve a given state, e.g. elevating the defined motor housing MH1 by 50cm. The practical basis is that the intended state is not given and pre-conditions may apply, e.g. that the motor housing requires R and a single unit is insufficient as its $size < R$ (cf. Def.3.3). The practical problem is, in our use case, to use local ties to find agents for a group to achieve R and avoid malfunctioning ones.

The recognition to cooperate on a given problem basis from [21, 30] gives a group's cooperation as the ability of a group g_i to achieve a state R iff there is some action sequence e_i that is a plan for g_i either to achieve R directly or to bring about some necessary conditions to achieve R. Panzarasa et al. [21] give the formalization of the notion of group ability as: $\forall g_i \forall t_i\ \mathcal{J}\text{-}CAN(g_i, R)(t_i) \equiv \exists e_i\ s.t.\ plan(g_i, e_i, R)(t_i) \vee plan(g_i, e_i, \mathcal{J}\text{-}CAN\,(g_i, R))(t_i)$ where $\mathcal{J}\text{-}CAN$ denotes the joint ability of multiple agents, and t_i denotes time. We see our algorithm as help for agents to a) determine how to get to a notion of cooperation that is how to find local agents that allow J-CAN over ties, b) how to recover if J-CAN is not given despite the best effort of all agents, and c) detect and mark which agents malfunction in a decentralized manner over time.

3.4 Implementation of agent behaviour

We explain the steps of our algorithm along Fig. 2 and present the application in examples, based on our running example of Ex. 1.1. The examples below are given from the perspective of Agent5 which controls Unit5.

Initialization (A). As part of the initial state a_0 (cf. Def. 3.3), all N agents have the goal state R that needs the help of other agents to be reached. R is initially known, e.g. because it is programmed into the agent as specialized behavior or received from an outside source. Agents are initialized with p_{ini} to form initial and new ties with agents within their units' communication range. Agents have no further knowledge of the remaining agent population.

Example 3.1. *Fig. 3: Agent5 is initialized with the task to lift MH1, a motor housing. Lifting MH1 is a task of size $R = 3$. Agent5 perceives all other units and their agents in Unit5's communication range and thus sees Unit7, Unit9, and Unit13 nearby. With p_{ini}, Agent5 tries to form a unidirectional tie to the units' respective Agent7, Agent9, and Agent13 - the result is a tie with Agent7 and Agent9.*

Group building (B). Agents want to reach the goal state R and recognize that they have to be supported by others. Of course, this involves planning ahead on how to reach R. We assume that this planning may be solved via the given plan libraries. Based on formed plans, agents determine connected components in a depth-first-search (DFS) among their ties to detect symmetric ones, that is if both agents possess a unidirectional tie to each other, and calculate suitable groups (see *appl*, Def. 3.3). All agents that are connected together via symmetrical ties form a single group that cooperates together for R.

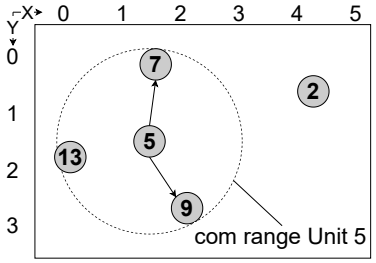

Figure 3: Initialization: Agent5 formed unidirectional ties to Agent 7 and 9, inside of Unit5's communication range (see Ex. 3.1)

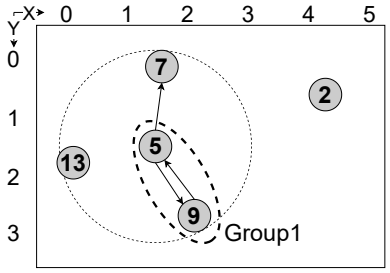

Figure 4: Group forming: Agents 5 and 9 have symmetrical ties and form Group1 (see Ex. 3.2)

Agents without a group will successively increase their communication range to contact randomly any other perceivable agent that is not already in their ties and actively try to form a tie without a probability attached.

Example 3.2. *Fig. 4: Agent5 concludes that it needs help for R. Agent5 evaluates its ties and asks Agent7 and Agent9, if they have unidirectional ties to Agent5. Agent7 has none, but Agent9 does. Agent5 and Agent9 form Group1 as they have a symmetrical tie.*

Task result (C). After enacting their plans, agents observe the outcome of their group's cooperation as defined by the agents' ties (see $performTask$, Def. 3.4), but do not have any knowledge about the individual units' `size`. Agents can observe if the group failed if the group needs more agents to even attempt the task, or if the group failed, where agents use the outcome s_k for decision (see *act*, Def. 3.3). In an application, this can e.g. be realized by observing if the motor housing is lifted or not, although enough units participate.

- $s_k = 1.0$, ideal outcome. `faultCounter` is reset to zero for all group members. The communication range is successively decreased, as a suitable group was found within reach and no further search has to be conducted.
- $s_k > 1.0$, partners missing. Agents increase the communication radius and try to build a new unidirectional tie. Agents first try to get an agent with `faultCounter=0` (as it promises a functioning group). If none are available, they pick a random agent i according to the probability

$$p_i = \frac{\texttt{faultCounter}_i}{\sum_{j=1}^{h} \texttt{faultCounter}_j}$$

among the h agents in range (via *perceive*, Def. 3.3). Note that i may already have a tie with the initiating agent.
- $s_k < 1.0$, unsuccessful cooperation. Every group member chooses one random tie, increases this agent's `faultCounter` by one, and then deletes its unidirectional tie to it (see *act*, Def. 3.3).

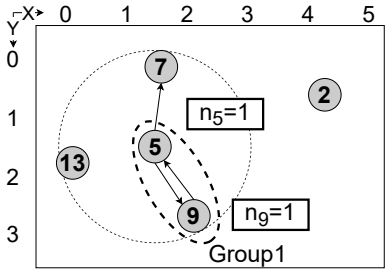

Figure 5: Task Result: Agents 5 and 9 of Group1 cannot perform task R yet (see Ex. 3.3)

We modify the equations of [3] such that each group k tries to get a task outcome of $s_k = 1.0$ with $s_k = R/(\sum_{j=1}^{h} size_j)$ by using its respective h units, reflected by $performTask$ (Def. 3.4). A group with a task R will thus tend in the trivial case towards R members when all units have $size_j = 1.0$ to get to the ideal usage of $s_k = 1.0$.

Example 3.3. *Fig. 5: Group1, consisting of Agent5 and Agent9, wants Unit5 and Unit9 to attempt the task of $R = 3$ to lift the motor housing. Unit5 and Unit9 have $n_5 = n_9 = 1$, giving $s_1 = R/(n_5 + n_9) = 3/2 = 1.5$. Agent5 and Agent9 observe with $s_1 > 1.0$ that the power of their units is not sufficient. With an increased communication range, both agents look for a new partner.*

Example 3.4. *Fig. 6: In an alternative attempt, Group1 consists of Agent5, Agent9, and Agent13, and with the task of $R = 3$ from Ex. 3.3. With Unit13's $n_{13} = 100$, the outcome is $s_1 = 3/102 = 0.03$. Group1 observes that $s_1 < 1.0$ gives an unsuccessful outcome for the task R. Agents 5 and 9 choose randomly Agent13 and penalize it by increasing its* `faultCounter` *by one and deleting their ties, while Agent13 chose to penalize Agent5 in the same way. Agent13's* `faultCounter` *is now two, Agent5's* `faultCounter` *is one, and Agent9's* `faultCounter` *is still zero.*

Example 3.5. *Fig. 7: After finding a new partner via symmetric ties, Group1 consists of Agent5, Agent9, and Agent7, and attempts the task of $R = 3$ from Ex. 3.3 again. With Unit7's $n_7 = 1$, the outcome is $s_1 = 3/3 = 1.0$. Group1 observes that $s_1 = 1.0$ gives an ideal outcome.*

New ties (D). All agents have a chance $p_n = 0.01\%$ (via a_0, cf. Def. 3.3), as in [3], to form a random new tie with another random agent inside the communications range.

Example 3.6. *Fig. 8: Agent5 tries to form a new tie after the task outcome. Inside Unit5's communication range are still Agents7, 9, and*

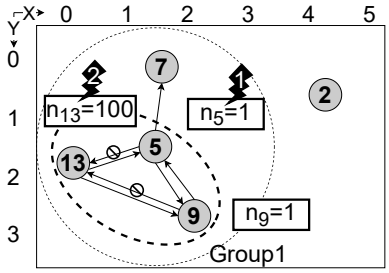

Figure 6: Task Result: Agents 5, 9 and 13 of Group1 fail to perform task R. Agent13 receives two faultCounters as a penalty, Agent5 one. Agents 5 and 9 remove their ties to Agent13. Note that Agent5's communication range increased (see Ex. 3.4)

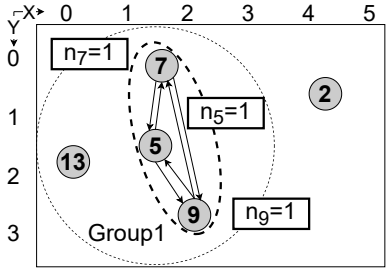

Figure 7: Task Result: Agents 5, 9, and 7 form Group 1 and successfully cooperate to achieve the task R (see Ex. 3.5)

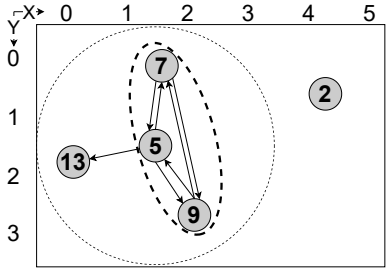

Figure 8: New ties: Agent5 forms a new random tie with agents inside the communication range, here Agent13 (see Ex. 3.6)

13. Here, Agent5 picks randomly Agent13 from the agents inside the range for a new unidirectional tie.

The process is finished, when all agents found successful groups for their units where each group k has optimal usage $s_k = 1.0$ within tolerances. After that, successful agents have no need to search for further units and cease the process. The remaining singletons will still try to find suitable groups, but will not find a suitable group, as no functioning agent is willing to form groups. From these singletons, the agent with the highest `faultCounter` is the one that is regarded as malfunctioning.

4 EXPERIMENT SETUP AND RESULTS

4.1 Experiment setup and measures

Agent types and experimentation setup were implemented with GAMA.[1] All code and measurements are available online.[2] The experiment setup contains a quadratic shop floor with a side length of 250m and round units with a diameter of 1m. Each unit is controlled by a distinct agent as an artifact. The communication range can be changed in 0.1m steps. We used four setups of starting positions:

- checkerboard style with 50 units (setup A)
- diagonal with 50 units (setup B)
- random setup with 50 units (setup C50)
- random setup with 105 units (setup C105)

We varied the initial communication range with $com = \{0\text{m}, 10\text{m}, 25\text{m}, 50\text{m}\}$, which influences the possible initial ties an agent can have. We fixed the probability to build random new ties with agents in range of $p_n = 0.05\%$ per simulation cycle, the probability to create initial ties to $p_{ini} = 5\%$, and $R = 3$. Agents had no further knowledge about other units or agents. All units, but one unit, were initialized with $size = 1$ as functioning, and one unit was initialized with $size = 10000$ as a faulty unit.

We measure the cycles needed until the algorithm terminates (time to finish, TTF) as well as the average communication range and amount of unidirectional ties of each agent at the end of the simulation. We evaluated the fault detection by comparison of all units: at the moment when the number of expected groups N/R was found across the population, we evaluated the experiment in terms of TTF, range, ties, and which unit had the highest amount of `faultCounters`, which was then interpreted as the identified faulty unit. If the identified unit was the same as the unit we initialized earlier as a faulty unit, the error was detected successfully. Obviously, the evaluation and stopping of the algorithm for measurement is a process that we as observers centrally controlled, but which had no influence on the agent's decentralized and distributed execution of our algorithm.

4.2 Results of our experiments

Figure 9 shows our results across setups and 100 repetitions for each scenario. Tab. 1 gives exemplary results of the measures for the checkerboard setup with varied communication ranges. Setups A, B, and C50 have very similar behavior with an error detection of $\geq 93\%$, setup C105 of $\geq 88\%$, while forming the expected groups for task size R. By reducing the communication range and resetting the ties, agents, and units end up with 6-9 ties to their neighbors and a communication range of 35-45m among all scenarios, which leads to a stable separation of the population into groups by excluding possible disruptive influences after group formation.

The variation of the initial communication range leads to diverse results: while setups A, B, and C50 tends to finish in about after 2500-3200 cyc, runs with a wider communication range leads to worse results, e.g. for $TTF_A = 2986cyc$ or $TTF_{C50} = 3200cyc$. We observe similar results for setup C105, where $TTF = 4313cyc$ at 10m as the best result, compared to $TTF = 12257cyc$ at 50m as the worst - a factor about 3 worse. A wider communication range,

[1] https://gama-platform.github.io/
[2] https://github.com/wintechis/vertebrate-fault-detection

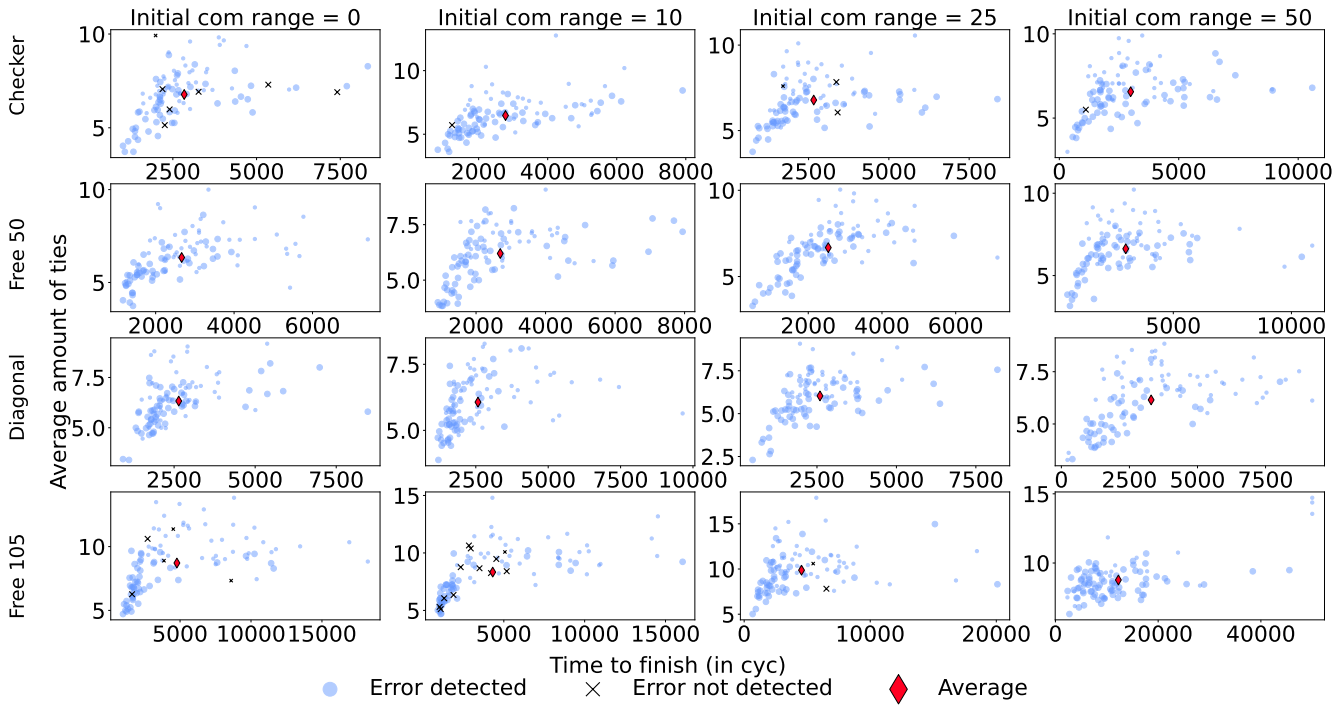

Figure 9: Measures of all repetitions varied across the scenarios showing the TTF and average amount of ties at the end of the run. Each line shows one scenario, and each column a different initial communication range. Data points show if a run has correctly detected the malfunctioning agent (circle) or not (cross). Small circles and crosses show if a run finished with more than the average amount of ties of the respective scenario. A diamond marks the overall average of the respective scenario.

paired with the higher possibility to form ties at the beginning of the algorithm, leads to more unidirectional ties for agents and hence more possible members in groups. The result is a longer run time for group formation and thus worse TTFs.

In terms of communication range, the results have to be interpreted carefully: units in groups reduce the range successively, so the range will tend towards 0 as units need the range only for the detection of other units. Whenever a group is successfully formed the unit has no need to detect other units. The faulty unit on the other hand tends towards the maximal range because as many other units as possible shall be detected. Thus, the extremes of the individual communication ranges have to be considered as well.

With an average detection rate of 98.06% from all setups, while forming cooperative groups, avoiding fixed assumptions, and using indirect communication, we see the usage of our algorithm as beneficial for distributed fault detection.

5 DISCUSSION

We took inspiration from the domain of biology and used the algorithm of [3] that describes the foraging of animals in a group, Cantor and Farine give specialized foraging groups of primates and birds as an example. The conditions for foraging animals are similar (no global view and tend towards the maximization of group share) to the usage of multiple agents in cooperation for a specific task. Where Cantor and Farine adjust the formation of their groups with respect to the ideal usage of an animal group for a given food resource including birth and death, we adjust the algorithm by introducing indirect communication, a reflection on task outcomes with respect to malfunction and respective marking, and the formation of stable sub-groups (instead of population wide groups as in [3]).

We are aware that our use case of group building and failure detection, combined with our assumptions, can also be solved by centralized fault detection algorithms as well. There is no limitation that a statistical approach could come to a similar result as we do, with better performance, e.g. a centralized algorithm that controls all units, knows that groups of R shall be created, and may assume `size`$= 1.0$ for all agents. This approach could easily choose randomly R agents from the population and observe the outcome, until it picks groups that fail - the agents from this group would then be recombined randomly with others in succinct steps, until only one agent is left, which is the malfunctioning one (if we assume N/R works out evenly and only one agent is left).

Nevertheless, beware of the strong assumptions as a centralized algorithm needs to know all units for communication and to pick units for groups, observe all outcomes, save the results in a centralized storage, and evaluate the results. Two disadvantages can easily be seen for this (admittedly well-performing) centralized algorithm, that is increasing computational and storage needs with the number of agents, and a strong connection (for availability and knowledge) to all units [5]. Thus, we emphasize that our algorithm runs in a

	TTF (cycles)			Ties			Com range (meters)			Detection (%)
init com	**avg**	**median**	**stdev**	**avg**	**median**	**stdev**	**avg**	**median**	**stdev**	**avg**
0m	2839.14	2508.5	1362.402	6.77	6.83	1.332	44.1048	38.38	21.36	93
10m	2777.94	2310.0	1455.939	6.47	6.39	1.451	41.8133	37.44	21.44	99
25m	2659.96	2179.0	1436.042	6.78	6.72	1.290	45.0069	39.61	16.11	97
50m	2986.72	2490.5	1959.683	6.56	6.63	1.332	37.6962	37.35	13.54	99

Table 1: Samples from measures of the checkerboard scenario with varied communication range

distributed and decentralized fashion and avoids the necessity of a global network for the population, and may scale through the use of restricted communication and stigmergy. Both can be easily extended for applications with decentralized task allocation [16] or relying on trust [22]. Two further points reduce the computational load on the participants, and give scalability:

1) Agents do not have to calculate tedious models to come to a residue or threshold for fault detection (as pointed out e.g by [6, 24]). Admittedly, our required detection of success is application specific and might therefore also make use of models, but not with a focus on agent performance. Where other approaches require direct observation of individuals to detect deviations, our approach requires only the group's outcome. This is especially useful where no such observation is per se possible, e.g. if four units try to carry a heavy box, at least the units on opposite sides cannot directly observe each other.

2) While the application of stigmergy can be challenging [31], indirect communication leads to the scalability of our approach. Agents expose their unit's failure markers, such that other agents read them directly, without the need to save others' performances. Thus, with an increasing amount of agents, the time to find suitable groups will clearly rise as more options are available (cf. Sec. 3), but not the needed memory for each agent. Still, synchronization of read-write-operations can become a problem with increasing group size when multiple agents try to increase the performance marker simultaneously, but locks may lead to respective solutions [27].

We see our algorithm as a feature-based anomaly detection approach as it relies on the comparison of expected outcomes and actual outcomes of cooperation in the respective agent's environment [2]. While the comparison requires an understanding of a "successful outcome" as compared to an "unsuccessful outcome" to decide about the performance of a group, the notion of success is tied strongly to the understanding of the task itself, which we assume to be understood by the controlling agent.

6 LIMITATIONS

The application of our algorithm in its current state has limitations: Agents have to measure the task fulfillment of units to decide on the task outcome (cf. Sec. 5). For the calculation, we assume that a malfunctioning agent has $\texttt{size} > R$ such that cooperation fails miserably with $s_k < 1$. The values for `size` follow Cantor and Farine [3] where the individual per capita share would be based on equal distribution of a group's foraging result. Hence, different `sizes` can be seen as group members that require excessively more food such that the outcome of the group foraging becomes unfeasible.

We tested our algorithm to detect one single malfunctioning agent, which was done successfully (cf. Sec. 4). Still, we assume that if several are present, the algorithm can be applied in a loop among remaining agents that did not find groups for R s.t. one after another failing agent is detected. The remaining, functioning agents with no mark of failure would be singletons and could possibly not fulfill R as there are not enough units.

Concerning the simulations, the validity of our results is limited to the presented settings only, as additional factors may influence the simulation outcome. We observe that the number of units and their initial locations on the shop floor can of course favor or hinder error detection and group building. Furthermore, simulations are by nature simplified abstractions from reality and prone to statistical fluctuations, especially when varying values for probabilities to build groups. We address these concerns by using different setups to measure the algorithm's performance. We are confident our results can be reproduced in similar settings.

7 CONCLUSION AND OUTLOOK

We present our algorithm that helps agents in multi-agent cooperation to detect and isolate malfunctioning units. Our algorithm is inspired by the behavior of vertebrates that form their groups based on the outcome of food foraging. Similarly, our agents use our decentralized algorithm to decide on the outcome of their group and how to adjust their relations, while detecting malfunctioning agents. Positive and neutral outcomes enable group forming, while negative outcomes lead to penalization with failure markers. The use of stigmergy and limited communication range enable the algorithm's distributed nature, independence of agents, and scalability.

Our experiments show that agents can efficiently identify the malfunctioning agent with a 98% chance while still forming groups for cooperation. We focus on the detection of a single, malfunctioning agent, but in the future want to the history of groups into account, which introduces the need to understand time. Further ideas include e.g. spatial mobility for agents with a fixed small, local communication range that move to find other group members.

All in all, we see the usage of our bio-inspired algorithm as beneficial and efficient for group forming and the detection of malfunctioning agents in a distributed manner.

ACKNOWLEDGMENTS

This work has been partially funded by the German Federal Ministry of Education and Research through the MANDAT project (Grant no. 16DTM107A).

REFERENCES

[1] Francesca Boem, Lorenzo Sabattini, and Cristian Secchi. 2016. Decentralized fault diagnosis for heterogeneous multi-agent systems. In *2016 3rd Conference on Control and Fault-Tolerant Systems (SysTol)*. 771–776. https://doi.org/10.1109/SYSTOL.2016.7739841

[2] David M. Bossens, Sarvapali Ramchurn, and Danesh Tarapore. 2022. Resilient Robot Teams: a Review Integrating Decentralised Control, Change-Detection, and Learning. *Current Robotics Reports* 3, 3 (01 Sep 2022), 85–95. https://doi.org/10.1007/s43154-022-00079-4

[3] Mauricio Cantor and Damien R. Farine. 2018. Simple foraging rules in competitive environments can generate socially structured populations. *Ecology and Evolution* 8, 10 (2018), 4978–4991. https://doi.org/10.1002/ece3.4061

[4] Mohammadreza Davoodi, Nader Meskin, and Khashayar Khorasani. 2016. Simultaneous fault detection and consensus control design for a network of multi-agent systems. *Automatica* 66 (01 Apr 2016), 185–194. https://www.sciencedirect.com/science/article/pii/S0005109815005592

[5] Rogério de Lemos et al. 2013. *Software Engineering for Self-Adaptive Systems: A Second Research Roadmap.* Springer Berlin Heidelberg, Berlin, Heidelberg, 1–32. https://doi.org/10.1007/978-3-642-35813-5_1

[6] Steven X. Ding. 2013. *Model-Based Fault Diagnosis Techniques.* Springer London. https://doi.org/10.1007/978-1-4471-4799-2

[7] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. 2006. Ant colony optimization. *IEEE Computational Intelligence Magazine* 1, 4 (2006), 28–39. https://doi.org/10.1109/MCI.2006.329691

[8] Edmund H. Durfee and Victor R. Lesser. 1991. Partial global planning: a coordination framework for distributed hypothesis formation. *IEEE Transactions on Systems, Man, and Cybernetics* 21, 5 (1991), 1167–1183. https://doi.org/10.1109/21.120067

[9] Peter Göhner and Michael Weyrich. 2014. Agent-Based Concepts for Manufacturing Automation. In *Multiagent System Technologies*, Jörg P. Müller, Michael Weyrich, and Ana L. C. Bazzan (Eds.). Springer International Publishing, Cham, 90–102.

[10] P. Grassé. 1959. La reconstruction du nid et les coordinations interindividuelles chez Bellicositermes natalensis et Cubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. *Insectes Sociaux* 6, 1 (01 Mar 1959), 41–80. https://doi.org/10.1007/BF02223791

[11] Meng Guo, Dimos V. Dimarogonas, and Karl Henrik Johansson. 2012. Distributed real-time fault detection and isolation for cooperative multi-agent systems. In *2012 American Control Conference (ACC)*. 5270–5275. https://doi.org/10.1109/ACC.2012.6315178

[12] Karuna Hadeli, Paul Valckenaers, Constantin Zamfirescu, Hendrik Van Brussel, Bart Saint Germain, Tom Hoelvoet, and Elke Steegmans. 2004. Self-Organising in Multi-agent Coordination and Control Using Stigmergy. In *Engineering Self-Organising Systems*, Giovanna Di Marzo Serugendo, Anthony Karageorgos, Omer F. Rana, and Franco Zambonelli (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 105–123.

[13] Francis Heylighen. 2016. Stigmergy as a universal coordination mechanism I: Definition and components. *Cognitive Systems Research* 38 (01 Jun 2016), 4–13. http://www.sciencedirect.com/science/article/pii/S1389041715000327

[14] O. Holland and C. Melhuish. 1999. Stimergy, Self-Organization, and Sorting in Collective Robotics. *Artificial Life* 5 (04 1999), 173–202.

[15] Nicholas R. Jennings. 2000. On agent-based software engineering. *Artificial Intelligence* 117, 2 (2000), 277–296. https://doi.org/10.1016/S0004-3702(99)00107-1

[16] Vera A. Kazakova and Gita R. Sukthankar. 2020. Adaptable and stable decentralized task allocation for hierarchical domains. *The Knowledge Engineering Review* 35 (2020), e26. https://doi.org/10.1017/S0269888920000235

[17] Anastassia Kuestenmacher and Paul G. Plöger. 2016. Model-Based Fault Diagnosis Techniques for Mobile Robots. *IFAC-PapersOnLine* 49, 15 (2016), 50–56. https://doi.org/10.1016/j.ifacol.2016.07.613 9th IFAC Symposium on Intelligent Autonomous Vehicles IAV 2016.

[18] Dubravko Miljkovic. 2011. Fault detection methods: A literature survey. In *2011 Proceedings of the 34th International Convention MIPRO*. 750–755.

[19] Alexandros Mouzakitis. 2013. Classification of Fault Diagnosis Methods for Control Systems. *Measurement and Control* 46, 10 (2013), 303–308. https://doi.org/10.1177/0020294013510471 arXiv:https://doi.org/10.1177/0020294013510471

[20] Andrea Omicini, Alessandro Ricci, and Mirko Viroli. 2008. Artifacts in the A&A meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems* 17, 3 (01 Dec 2008), 432–456. https://doi.org/10.1007/s10458-008-9053-x

[21] Pietro Panzarasa, Nicholas R. Jennings, and Timothy J. Norman. 2002. Formalizing Collaborative Decision-making and Practical Reasoning in Multi-agent Systems. *Journal of Logic and Computation* 12, 1 (2002), 55–117. https://doi.org/10.1093/logcom/12.1.55

[22] Caroline Player and Nathan Griffiths. 2020. Improving trust and reputation assessment with dynamic behaviour. *The Knowledge Engineering Review* 35 (2020), e29. https://doi.org/10.1017/S0269888920000077

[23] Stuart Russell and Peter Norvig. 2009. *Artificial Intelligence: A Modern Approach* (3rd ed.). Prentice Hall Press, USA.

[24] Iman Shames, André M.H. Teixeira, Henrik Sandberg, and Karl H. Johansson. 2011. Distributed fault detection for interconnected second-order systems. *Automatica* 47, 12 (2011), 2757–2764. https://doi.org/10.1016/j.automatica.2011.09.011

[25] Weiming Shen and Douglas H. Norrie. 1999. Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey. *Knowledge and Information Systems* 1, 2 (01 May 1999), 129–156. https://doi.org/10.1007/BF03325096

[26] Rachel A. Smolker, Andrew F. Richards, Richard C. Connor, and John W. Pepper. 1992. Sex Differences in Patterns of Association Among Indian Ocean Bottlenose Dolphins. *Behaviour* 123, 1-2 (1992), 38 – 69. https://doi.org/10.1163/156853992X00101

[27] Andrew S. Tanenbaum and Maarten van Steen. 2006. *Distributed Systems: Principles and Paradigms (2nd Edition).* Prentice-Hall, Inc., USA.

[28] Luca Tummolini and Cristiano Castelfranchi. 2007. Trace Signals: The Meanings of Stigmergy. In *Environments for Multi-Agent Systems III*, Danny Weyns, H. Van Dyke Parunak, and Fabien Michel (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 141–156.

[29] H. Van Dyke Parunak. 2006. A Survey of Environments and Mechanisms for Human-Human Stigmergy. In *Environments for Multi-Agent Systems II*, Danny Weyns, H. Van Dyke Parunak, and Fabien Michel (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 163–186.

[30] Michael Wooldridge and Nicholas R. Jennings. 1999. The cooperative problem-solving process. *Journal of Logic and Computation* 9, 4 (08 1999), 563–592. https://doi.org/10.1093/logcom/9.4.563 arXiv:https://academic.oup.com/logcom/article-pdf/9/4/563/3887385/090563.pdf

[31] Ouarda Zedadra, Nicolas Jouandeau, Hamid Seridi, and Giancarlo Fortino. 2017. Multi-Agent Foraging: state-of-the-art and research challenges. *Complex Adaptive Systems Modeling* 5, 1 (02 Feb 2017), 3. https://doi.org/10.1186/s40294-016-0041-8