Analysis of Perceived Helpfulness in Adaptive Autonomous Agent Populations

Mirgita Frasheri, Baran Cürüklü, and Mikael Ekström

Mälardalen University, Västerås, Sweden
{mirgita.frasheri, baran.curuklu, mikael.ekstrom}@mdh.se

Abstract. Adaptive autonomy allows agents to change their autonomy levels based on circumstances, e.g. when they decide to rely upon one another for completing tasks. In this paper, two configurations of agent models for adaptive autonomy are discussed. In the former configuration, the adaptive autonomous behavior is modeled through the willingness of an agent to assist others in the population. An agent that completes a high number of tasks, with respect to a predefined threshold, increases its willingness, and vice-versa. Results show that, agents complete more tasks when they are willing to give help, however the need for such help needs to be low. Agents configured to be helpful will perform well among alike agents. The second configuration extends the first by adding the willingness to ask for help. Furthermore, the perceived helpfulness of the population and of the agent asking for help are used as input in the calculation of the willingness to give help. Simulations were run for three different scenarios. (i) A helpful agent which operates among an unhelpful population, (ii) an unhelpful agent which operates in a helpful populations, and (iii) a population split in half between helpful and unhelpful agents. Results for all scenarios show that, by using such trait of the population in the calculation of willingness and given enough interactions, helpful agents can control the degree of exploitation by unhelpful agents.

Keywords: Adaptive Autonomy, Collaborative Agents, Multi-Agent Systems.

1 Introduction

Adaptive autonomous (AA) agents are software agents which are able to decide on whether to be more or less autonomous, with respect to some task, given specific circumstances. Autonomy can both refer to the behaviour of an agent, in any given context, as well as the agent's relationship with other entities. The latter can include other AA agents, non AA agents, or human operators that operate in a nearby environment (physical or virtual), such that communication between them is possible. The decision to change the autonomy level could also lie on the human operator, or be a result of the cooperation between human and agent. Thus, adaptive autonomy is one of many concepts that involves the change of the autonomy level of an agent. A non-exhaustive list of theories and definitions, in addition to AA is as follows: adjustable autonomy, mixed-initiative interaction, sliding autonomy, collaborative control and so on.

Adjustable autonomy refers to a system in which the human is the one who makes the decision with respect to the autonomy of an agent [1]. Nonetheless, it has also been used as a generic term for the different means in which decision-making regarding autonomy could be shared between human and agent [2]. In mixed-initiative interaction, agent and human are both able to make a decision, depending as well on the circumstances [1]. Collaborative control [3] is an early approach, which departed from a classical view of human/master - agent/slave, into one in which both were peers. Any inconsistencies between them were resolved through dialogue. Nonetheless, the human was the one who would set the global goals for the agent. Sliding autonomy represents another approach in which two modes (full autonomy and tele-operation) could be switched on the task level [4]. Consequently, an operator is able to conduct some tasks, whilst the system performs autonomously for others, depending on the circumstances. Agent autonomy has been studied extensively in the literature. The 10 levels

Table 1: The 10 levels of autonomy proposed by Parasuraman *et al.*

HIGH	10. The computer decides everything, acts autonomously, ignoring the human
	9. informs the human only if it, the computer, decides to
	8. informs the human only if asked, or
	7. executes automatically, then necessarily informs the human, and
	6. allows the human a restricted time to veto before automatic execution, or
	5. executes that suggestion if the human approves, or
	4. suggest an alternative
	3. narrows the selection down to a few, or
	2. The computer offers a complete set of decision/action alternatives, or
LOW	1. The computer offers no assistance: human must take all the decisions/actions

of autonomy have been proposed by Parasuraman *et al.* [5] (Table 1). Another scheme is through the dimensions of self-sufficiency, i.e. being able to do a task without outside assistance, and self-directedness, i.e. being able to choose one's own goals. Johnson *et al.* [2] refer to them as the descriptive and prescriptive dimensions of autonomy, respectively. Furthermore, they add a third dimension, that of inter-dependencies between team-mates (either agent or human). The former could be either hard, i.e. necessary for the successful outcome of a task, or soft, i.e. not necessary, however, could improve on the performance of a task. Castelfranchi defines autonomy using dependence theory [6], i.e. if an agent a_i lacks any means (e.g. ability, knowledge, or external resources) to perform a task t and relies/depends on another agent a_j for its provision, then a_i is not autonomous from a_j with respect to t. Moreover, this kind of autonomy/non autonomy has a social nature, and is to be distinguished from autonomy from the environment, or in other words autonomy with respect to how to react to incoming stimuli. This paper assumes [6], thus allows agents to adapt their autonomy by deciding on whether to depend on each other. Section 2 provides more information on the research conducted in the field of AA and alike concepts. It also puts the work presented in this paper into perspective with respect to the literature.

A model for an AA agent has been proposed previously [7], in which adaptation is modeled through the willingness of agents to give assistance to each other. The agent's own performance – calculated as the number of tasks completed over tasks attempted – influences the willingness to give assistance. If the performance is high, the agent will be more willing to help, the opposite also being true. This is **Configuration 1** ($C1$) of the agent model. $C1$, alongside dedicated simulations are shortly described in this paper, specifically in Section 3.1 - 3.3 and Section 5.1. Subsequently, the AA agent architecture was extended through the incorporation of another behaviour: willingness to ask for help (Section 3.4). Thus, the current model incorporates both directions of communication with respect to allowing agents to help each other. This is **Configuration 2** ($C2$).

The next step is to consider the interactions from a population perspective. The characteristic of the population under study is the perceived helpfulness of its individuals taken as a group. It is defined by the number of times in which agents have been willing to assist an agent a_i, over the total number of requests for help made by a_i. Each agent in the group can estimate the perceived helpfulness (i) of the population, and (ii) of individual agents. Both measures are combined and used in the calculation of the willingness to give help, and extend $C2$ (Section 4). The perceived helpfulness of the population is referred throughout the text as part of *agent culture*. The main objective is to analyse how the (most) helpful AA agents can avoid extensive exploitation of their resources by other agents. The corresponding simulations and results are summarized in Section 5.2. Three hypotheses are evaluated.

Hypothesis 1 (H1): *Exploitation of an agent by its peers can be lowered by considering the helpfulness of the population as a whole as well as that of an individual agent in the calculation of willingness to give help.*

Hypothesis 2 (H2): *An agent population can adapt to an agent configured so as to exploit by considering the helpfulness of the population as a whole as well as that of an individual agent in the calculation of the willingness to give help. Moreover, the efficiency of such isolation is disproportional to the population size.*

Hypothesis 3 (H3): *Agents in a mixed population, where half of the population is helpful and the other half is otherwise, can reduce exploitation while still helping each other by considering the helpfulness of the population as a whole as well as that of an individual agent in the calculation of the willingness to give help.*

Finally, the paper concludes with a discussion and reflections for future work (Section 6), and conclusions (Section 7).

2 Related Work

Agent autonomy is an extensively discussed topic in the literature. A close examination of the related work points at six (6) main directions of research: (i) design of user interfaces which aid human/robot(agent) collaboration, (ii) specific algorithms that allow for autonomy levels of agents/robots to be changed such as Markov Decision Processes, (iii) design of policy systems for the regulation of agent behaviour, (iv) works that aim at comparing different schemes for changing autonomy levels and works that motivate the need for such change, (v) design methodologies for the creation of systems that support inter-dependencies between systems, (vi) general architectures and frameworks. The next paragraphs provide a compact description of relevant literature in each of the mentioned directions. Furthermore, the research conducted in this paper is put into perspective with the existing work found in the literature.

(i) User interfaces are used as means to allow both agent and human to monitor each other, and consequently change autonomy levels if perceived necessary. The initiation of change can come from both sides. A system able to capture the user's skill, can change its autonomy accordingly [8]. In this case skills are of a navigational, manipulation (gripping), and multi-robot coordination nature. On the other hand, a human operator can have the flexibility to command a robot at several levels such as: low-level control, way-point control, high-level control (sending goals like "bring the can of coke") [9]. Other types of interfaces are aimed at aiding a human to monitor and control a group of robots, which may need only occasional support [10] [11]. Other work is specific to navigational issues, in which a human assists path planning of UAVs (unmanned aerial vehicles) by providing spatial and temporal constraints [12]. The 3T agent architecture [13] has been extended to allow for the human in the loop of the decision-making of the agent [14]. The addition enables the system to keep track of what the human does, so as not to lose the common picture between human/system. Overall, the challenge for these interfaces is to allow for the common ground not to be lost and be accessible by all parties [15] [16]. Moreover, an important consideration to be made has to do with how much autonomy the agent/robot is intended to have [17].

(ii) Attributes such as task urgency and dedication level to the organization haven been used to guide the agent's reasoning with respect to when to take more initiative (increase its autonomy) [18]. Autonomy levels can also be changed at the task level, i.e. one task needs tele-operation from the human, whereas another can be conducted autonomously [4]. Furthermore, several task allocation algorithms have been proposed. In one, tasks are mapped to agents, and the human is able to accept/reject such mapping and trigger task allocation from the start [19]. Others categorize tasks in two groups: tasks which the agent can perform autonomously, and tasks that need human assistance [20]. The classification influences algorithm design. Colored Petri Nets are used in the formalization of team plans and addition of interrupt mechanisms, which allow the human to intervene in case of need [21]. Markov Decision Processes (MDPs) are employed to map help requests from agents with available humans

– assuming that agents detect when they are in trouble [22]. In other threat recognition and target identification applications, the system is implemented to query a human when it fails [23].

(iii) Regulatory systems (e.g. policies) are discussed in the context of regulating agent behaviour, due to bringing predictability and thus coordination [24]. An example is the Kaa system [25] which extends the KAoS policy system, by introducing a central agent (the Kaa) to override and adjust policies during run-time. If Kaa cannot reach a decision, then the human is introduced in the loop. Another approach involves the implementation of transfer-of-control strategies (through MDPs), which specify how control should be transferred between human and agents [26]. This has been applied in the E-elves platform (personal assistant agents) which ran at the University of Southern California.

(iv) The ability to change autonomy levels is considered a desirable features of systems, which can allow them to operate in human-team like fashion [27]. Scenarios with and without the ability to change autonomy have been compared [28] [29]. Decision-making frameworks such as master/slave, peer-2-peer and locally autonomous are dynamically shifted to show the superiority compared to static autonomy. However, the authors use data from previous experiments to apply the right decision framework for each environmental condition; there is no reasoning embedded in the agents. Different implementations of dynamic autonomy have been compared, which are adaptive autonomy (agents change their own autonomy), mixed-initiative interaction (both human and agent are able to change autonomy), and adjustable autonomy (the human is able to change the autonomy) [1]. In their simulations, mixed-initiative interaction performs better in terms of victims identified in search and rescue simulation scenario.

(v) Jonson *et al.* have argued the need for the analysis of inter-dependencies between systems, and its use in the design phase [2]. Moreover, they have proposed one such methodology [30], namely Co-active Design. This method includes the following steps. (i) Inter-dependencies in the system are identified. (ii) Mechanisms are designed to address each inter-dependency. (iii) The effects of these mechanisms on present inter-dependent relationships are analyzed. The aim is to make automation a team-player. In this respect several challenges have been identified [31] such as: basic compact, adequate models, predictability, directability, revealing status and intention, goal negotiation, collaboration, attention management, and cost control.

(vi) The agent architecture STEAM [32] has extended the Soar agent [33] to include support for teamwork. Team operators – reactive team plans – are introduced. These are an addition to the agent's plans that do not require teamwork. The solution includes a synchronization protocol so that agents can coordinate with respect to team plans. The DEFACTO framework [34] aims at providing support for transfers of control in continuous time, resolving human-agent inconsistencies, and making actions interruptible for real-time systems. Team THOR's Entry in the DARPA Robotics Challenge [35] brings forward a motion framework for a humanoid robot which allows for low-level control, scripted autonomy (i.e. invocation of robot movements by calling predefined scripts), and enables issuing high-level commands.

The research discussed in this paper fits mostly within (ii) and (vi). On one hand, algorithms are being developed to allow agents to assist and ask each other for assistance during their run-time. There are no classifications of tasks that either need assistance or not. In principle, an agent might require help for any task, due to changing circumstances. Assume an agent a_i which at time T_1 is able to perform task t. The same agent, at time T_2 might not be able anymore to continue on its own. One reason could be that, its battery levels have gone down. On the other hand, these algorithms fit within a general agent architecture, which models how an agent executes during its run-time.

3 Agent Model

This section describes configurations 1 and 2 of the agent model.

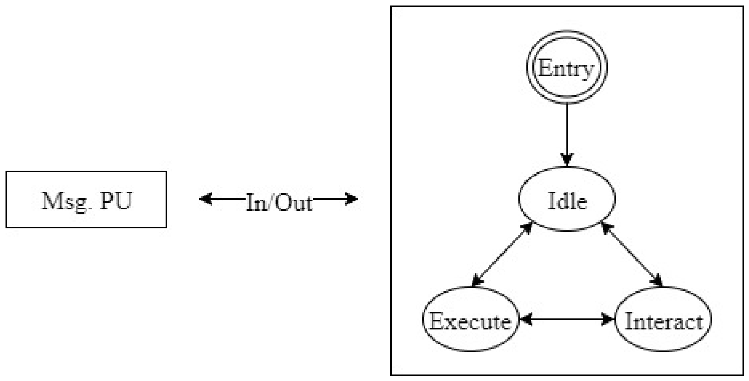


Fig. 1: C1 agent model with three states [7]. The Msg. PU represents the module in which messages coming from other agents are handled

3.1 Early Work (C1)

The agent model proposed previously [7] consists of the three (3) states, *interact*, *execute*, and *idle* (Figure 1). All agents in the population have a willingness to give assistance to each other. This concept is represented by a probability value, which defines the likelihood for such an event to occur. Assume an agent a_i which is in either the states of *idle* or *execute*. If the agent is in *execute*, it means that it is already dedicated to a task. Agent a_i can be in either *idle* or *execute* when a request for assistance is received, and will switch immediately to the *interact* state, where the decision of whether to accept the request will be made.

If a_i accepts the request, it will switch to *execute* with the new corresponding task, whilst the old one will be dropped for good. Otherwise, it will switch to its previous state, i.e. either *idle* or *execute* and continue with the old task.

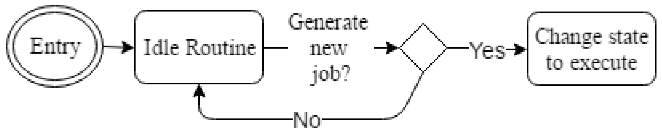


Fig. 2: Flowchart for the *idle* state [7].

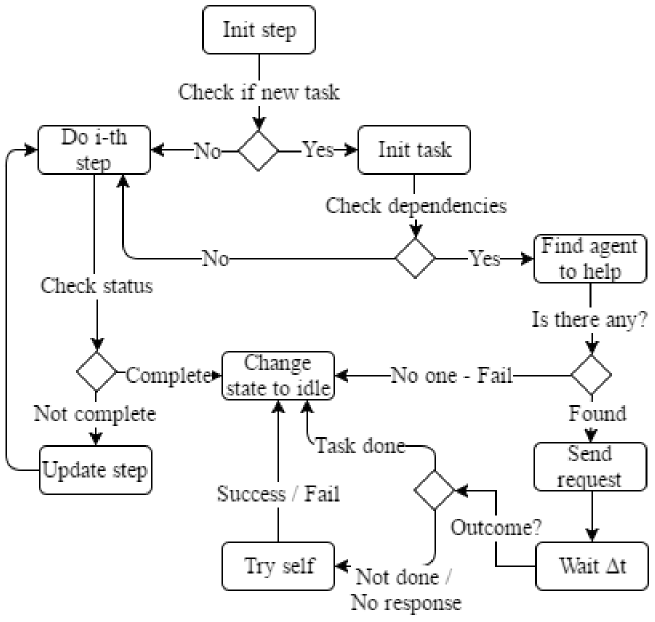


Fig. 3: Flowchart for the *execute* state [7].

An agent always starts its operation in the *idle* state. In this state, the agent is not dedicated to any task or goal. Nonetheless, a task could be generated with a probability P (Figure 2). This task is picked from a list of tasks which the agent is able to do (specified before runtime). When a task is generated, or a request for help is accepted, the agent switches to *execute*. The assumption is made that if the agent is not interrupted, it will always complete its task successfully (Figure 3). At the beginning of each task, an agent will check for dependencies on other tasks. In the case of C1, dependencies are assumed to be

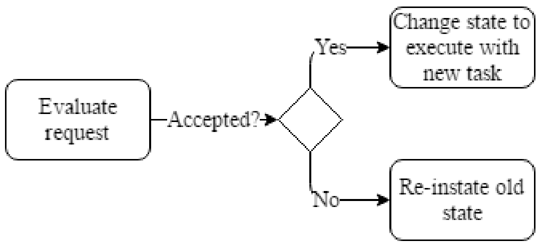


Fig. 4: Flowchart for the *interact* state [7].

fixed and known before-hand by all agents. If there is any, it will issue a help request to some known agent a_j which is able to perform the task. Then it will wait for a finite amount of time (constant determined before runtime and equal to all agents) for a response from a_j. If there is no response, the agent will give up on waiting, update the history of interactions with a_j, and attempt to do the task by itself with a lower probability of success. There are other options which could be implemented as well. (i) The agent can first try by itself, and if it fails, asks another for help. (ii) After a_j fails, the agent attempts with another agent a_k. When a task is finished, the agent returns to *idle*.

Agents keep track of the outcomes of the interactions with each other. As a result, they are able to compute the perceived willingness to help and expertise with respect to some task of every other agent. An agent a_i will select another agent a_j to ask for help based on a_j's helpfulness in the past, i.e. based on a_j's perceived willingness to help. During its operation, the agent keeps track of the outcomes for each interaction, which is used to determine helpfulness. It is important to note the difference between the perceived helpfulness and expertise. The former is an indicator of how much another agent has been willing to help, whereas the latter captures the actual success rate of the agent that has been trying to help. As a result, the perceived willingness is a more optimistic measure in which to judge other agents.

When the agent gets a request from another agent for help, then it will switch to the *interact* state which cannot be interrupted, i.e. it can be considered as an atomic step (Figure 4). This has the implication that the requests will be processed one at a time in a FIFO manner. There are two possible outcomes from the *interact* state. The agent can drop the past activity and go into *execute* with a new task, or it discards the requests and continues with what it was doing before receiving the request. The willingness to give help is the determining factor that shapes agent behaviour. The agent performance will in turn influence the willingness to give help in the following way. If the agent calculates that it has dropped too many tasks, then its willingness will decrease. On the other hand, if it has completed most of the tasks it has attempted, then its willingness will increase.

3.2 Interactions between Agents

In order to resolve the dependencies between them, agents need to interact with each other. This means that, if an agent a_i doing a task t identifies that it needs to depend on a_j to complete t, then a_i will need to interact with a_j. Dependencies themselves could be known in advance (as assumed for the agent described in 3.1) or could arise during runtime. Moreover, they can arise at the beginning or during the execution of a task t.

There are several types of possible interaction between agents:

1. Non-committal interaction. Agents could broadcast certain messages to others in the vicinity. These messages can contain different kinds of information related to e.g. identity, offered services, warnings ("There is fire in corridor x", "path from x_1 to x_2 is blocked"). Other agents are able to accept or disregard them. Nonetheless, no dialogue is being established by the involved parties. This means that the agent sending a broadcast does not expect any reply or commitment from others. In this work, this interaction is used by agents to make themselves known to each other. In principle, other agents could be able to evaluate the trustworthiness of the broadcasting agent by examining the following: (i) is the information useful, and (ii) is it true?
2. One-to-one dialogue. Agent a_i misses specific information, and queries a_j. In this case, a one-to-one dialogue is being established, in which one party expects a reply from the other, so that it is able to fill its knowledge gaps. The validity of the information a_j provides could be evaluated, as well as its perceived helpfulness to a_i.
3. One-to-one delegation. Similarly to one-to-one dialogue, a kind of dialogue is established in this case as well, in the form of a request to complete a task. This means that a_i will ask a_j to perform an activity on which $a_i's$ success with respect to a task depends. In general, a_i could be still able to succeed by itself, but with a lower probability. a_i is able to evaluate the behaviour of a_j based on (i) its perceived helpfulness and (ii) shown expertise. a_j as well will perform an evaluation in order to determine whether to assist a_i. This type interaction is as well implemented in this paper.
4. One-to-many dialogue/delegation. There are two ways to interpret this scenario. (i) There could be a chain of one-to-one interaction which emerges, e.g. a_i asks a_j, which asks a_k and so on. In this paper, such kind of chains can emerge. (ii) An agent can start parallel interactions with a number of other agents (B, C, etc.) by asking each of them to perform some specific subtask.

Johnson *et al.* [2] consider in their work soft and hard interdependencies between agents. Each level of interaction described in the previous paragraph could refer to either depending on the concrete scenario. For instance, a non-committal broadcast message could contain an alarm (e.g. "There is fire in x corridor") and be decisive for the outcome of a task (and even well-being of the agents). Thus it represents a hard interdependence. However, an ordinary informational message (e.g. "path from x_1 to x_2 is blocked") if disregarded could only delay the execution of some task without hindering its success. As a result, it can be considered as a soft interdependence. Furthermore, the difference with Barber *et al.* [29] is that the decision to assist another agent lies on the agent itself. This means that, a_i can ask to delegate a task to a_j, and a_j reasons and decides whether to accept such delegation.

3.3 Agent Organization and Autonomy

Agent organization will have an impact on how autonomy is shaped for each individual agent that makes up the population. There are two possibilities. (i) There is a hierarchy between agents which could be predefined or could emerge (e.g Barber *et al.* [29] consider how environmental conditions could be used to evaluate which hierarchy fits best a specific scenario). (ii) Agents are peers with each other.

In the first case, an agent a_i which is a superior of a_j is able to delegate to a_j any task it sees fit with some assurance that a_j will comply. Delegating to a_j does not necessarily mean that a_i cannot perform the task by itself. It can very reasonably be assumed that a_i is able to perform the task but simply prefers to conserve its resources, and ask a_j instead. a_j on the other hand either has some freedom in which it can refute to obey a_i (e.g. what a_i asks endangers a_j in ways that may or may not have been foreseen by a_i) or it does not and it will always have to comply. As a result, in general a_j will be dependent on the will of a_i, given that the power relations between them hold.

In the second case, a_i and a_j are peers, thus no power relations between them can be assumed. If a_i needs help, it will make a request to a_j, which in turn will decide based on its willingness to give help whether to assist a_i. As such, it is a_i which depends on the will of a_j. Nevertheless other factors might come into play. If a_i has been helping a_j in the past, then the latter could be more inclined to return the favor, thus becoming easier to interfere with. Furthermore, the motives of a_j might not be genuine (help a_i because it has helped me), but they could in fact be more along the lines of: help a_i so it can continue helping me in the future.

Note that, some form of dependence between agents is present in both scenarios. Moreover, dependence always constitutes a risk [36]. Even when there are power relations, an agent choosing to delegate to another, is choosing to depend, and thus is giving away some of its autonomy. The agent that delegates might be able to do the task by itself, if another agent fails. However, if the output of a task is expected within a certain time, then a delay could mean failure. On the other hand, if the agent cannot perform the task by itself, then it is even more dependent on the agent it asks for help. Therefore, the level of autonomy cannot always be well defined and can be blurred. In this paper, agents are assumed to be peers.

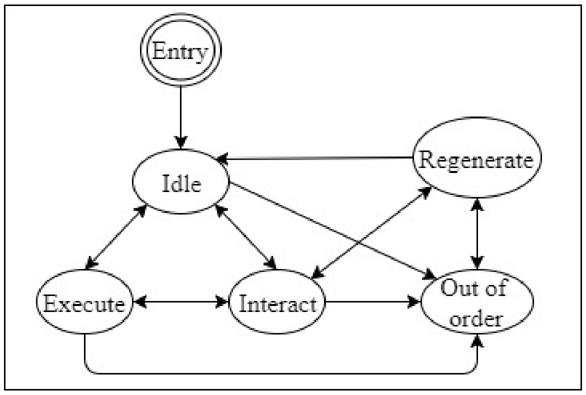


Fig. 5: Agent model extended to include two more states: *regenerate* and *out_of_order*

3.4 The AA Agent (C2)

The agent model described in Section 3.1 has been extended with respect to two dimensions. (i) Two more supporting states have been added (Figure 5), the *regenerate* and *out_of_order* states respectively. If the agent reaches critical levels of battery (the start-up energy level, and the critical level are arbitrarily specified before runtime) it will switch to the *out_of_order* state, and immediately from that state it will switch to *regenerate* where the recharge process is simulated. Next, the agent will switch to either *idle* or *interact* (if there are any requests pending). The agent can go to *out_of_order* from any other state (in principle from *regenerate* as well). From the implementation perspective, ROS services (for one-to-one interaction) have been switched with the ROS action server mechanism. The limitation of the current implementation is that the execution of a task in the *execute* state is simulated as a singular step, by pausing the system for a finite amount of time (which represents the completion time for a task). As a result, an agent reasons on whether it should ask for help at the beginning of each task.

(ii) The agent's adaptive autonomous behaviour is shaped by the willingness to interact which is composed of the willingness to give help (δ), and the willingness to ask for help (γ). The willingness to ask for help represents the probability that an agent will ask another for assistance given its current circumstances (in this case, dependencies are assumed to rise during runtime – in contrast to C1). Moreover, the factors assumed to influence each facet of the willingness to interact have been analyzed, and a corresponding computational model has been proposed from which δ and γ are calculated. The calculation is done according to algorithms 1 and 2.a. The notation used in both has the following meaning: b - battery, e - equipment, k - knowledge, a_i - abilities, n_t - tools, μ - agent performance, e_R - environmental risk, t_p - task progress/trade-off, a_R - perceived agent risk (complementary to perceived helpfulness). The notations with the subscript t refer to abilities, resources needed by the task. In the cases when it misses it refers to what the agent has at its disposition.

Algorithm 1 Agent's reasoning process on when to ask for help

```
procedure REASONING ON ASKING FOR HELP($b, e, k, a, t, \mu, e_R, t\_p, a\_R$)
    $\gamma \leftarrow \gamma_0$
    if $b - b_t < b_{min}$ or $e_t \not\subset e$ or $k_t \not\subset k$ or $a_t \not\subset a$ or $t_t \not\subset t_t$ then ▷ Consider internal resources
        $\gamma \leftarrow 1$
        return $\gamma$
    else
        $\gamma \leftarrow \gamma - 5\Delta\gamma$
        if $e_R$ increase then                          ▷ Consider environment risk
            $\gamma \leftarrow \gamma + \Delta\gamma$
        else
            $\gamma \leftarrow \gamma - \Delta\gamma$
        if $a\_R$ increase then                         ▷ Consider agent risk
            $\gamma \leftarrow \gamma - \Delta\gamma$
        else
            $\gamma \leftarrow \gamma + \Delta\gamma$
        if $\mu$ increase then                          ▷ Consider own performance
            $\gamma \leftarrow \gamma - \Delta\gamma$
        else
            $\gamma \leftarrow \gamma + \Delta\gamma$
        if $t\_p$ good then                             ▷ Consider task progress
            $\gamma \leftarrow \gamma - \Delta\gamma$
        else
            $\gamma \leftarrow \gamma + \Delta\gamma$
    return $\gamma$
```

3.5 ROS

The Robot operating system (ROS) [37] serves the role of a middleware by implementing different communication mechanisms such as: (i) publish/subscribe, (ii) services, and (iii) action servers. A ROS executable is called a node. Nodes publish and subscribe to channels referred to as topics. Nodes are able to find each other through the ROS master, which serves the purpose of a name domain system. Once a node is up, it will register itself with the master, which in turn connects the nodes when required. Services are point to point communications between one client and one server. Action servers improve on services by allowing the server node to send progress feedback to the client after an initial server call.

In this work, in both C1 and C2 configurations, agents are composed of two ROS nodes. The main agent node contains the logic, and the message processing

Algorithm 2.a Agent's reasoning process on when to give help

```
procedure REASONING ON GIVING HELP($b, e, k, a, t, \mu, e_R, t\_p, a_R$)
    $\delta \leftarrow \delta_0$
    if $b - b_t < b_{min}$ then                              ▷ Consider internal resources
        $\delta \leftarrow 0$
        return $\delta$
    else
        $\delta \leftarrow \delta + \Delta\delta$
        if $e_t \not\subset e$ then                           ▷ Consider internal resources
            $\delta \leftarrow \delta - \Delta\delta$
        else
            $\delta \leftarrow \delta + \Delta\delta$
        if $k_t \not\subset k$ then                           ▷ Consider internal resources
            $\delta \leftarrow \delta - \Delta\delta$
        else
            $\delta \leftarrow \delta + \Delta\delta$
        if $a_t \not\subset a$ then                           ▷ Consider internal resources
            $\delta \leftarrow \delta - \Delta\delta$
        else
            $\delta \leftarrow \delta + \Delta\delta$
        if $t_t \not\subset t$ then                           ▷ Consider external resources
            $\delta \leftarrow \delta - \Delta\delta$
        else
            $\delta \leftarrow \delta + \Delta\delta$
        if $e_R$ increase then                                ▷ Consider environment risk
            $\delta \leftarrow \delta - \Delta\delta$
        else
            $\delta \leftarrow \delta + \Delta\delta$
        if $a_R$ increase then                                ▷ Consider agent risk
            $\delta \leftarrow \delta - \Delta\delta$
        else
            $\delta \leftarrow \delta + \Delta\delta$
        if $\mu$ increase then                                ▷ Consider own performance
            $\delta \leftarrow \delta + \Delta\delta$
        else
            $\delta \leftarrow \delta - \Delta\delta$
        if $t\_p$ good then                                   ▷ Consider task progress
            $\delta \leftarrow \delta + \Delta\delta$
        else
            $\delta \leftarrow \delta - \Delta\delta$
    return $\delta$
```

unit node handles published data from other agents. Nodes are written in the python language, which is supported in ROS (alongside other supported languages such as C++ and Java). In C1, the services are used to implement the one-to-one delegation interaction. In C2, service calls are replaced with action server calls. At present there is no substantial difference between the two imple-

Algorithm 2.b Reasoning on δ with perceived helpfulness

```
procedure ADDITIONAL REASONING ON $\delta(a\_R, a\_S, C)$
    $\delta \leftarrow \delta_p$                                    ▷ $\delta_p$ calculated in Algorithm 2.a
    if $a_R >= 0.5$ and $C <= 0.5$ then
        $\delta \leftarrow LOW$
    else if $(a_R < 0.5$ and $C <= 0.5)$ or $(a_R < 0.5$ and $C > 0.5)$ then
        $\delta \leftarrow \delta + k * \Delta step$
    else if $a_R >= 0.5$ and $C > 0.5$ then
        if $a\_S < 3$ then
            $\delta \leftarrow \delta - \Delta\delta$
        else
            $\delta \leftarrow LOW$
```

mentations in the code. Nevertheless, the transition was made in order to allow more flexibility for future development.

4 Perceived Helpfulness

This paper extends the ideas presented and discussed in previous work [7] by including in the agent's reasoning, characteristics of the population in which the agent operates. These additions take place in Configuration 2. One such characteristic is the perceived helpfulness of the whole population of agents (or the part of the population with which an agent is able to communicate and interact with). This is the target in this work. The perceived helpfulness of the population is referred to as part of its *culture*. Other facets can be part of agents' culture, such as perceived willingness to ask for assistance. However, the latter is not treated here.

Assume an agent a_i found in an environment among other agents (no particular hierarchy is imposed). During its operation, a_i sends to and receives from others help requests. Continuously, upon each request, a_i has to decide whether to help or turn down the request. Moreover, a_i will need to depend on other agent on particular circumstances. Previously, it has been showed that an agent configuration with high willingness to give help, in situations where dependencies are low (i.e. it asks for help in few cases), results in the highest number of tasks completed (performance measure) for the whole population [7]. It is reasonable to think that such a population will succeed with respect to the number of tasks completed.

Note that, it is not always possible to assume that an agent will find itself among agents configured in the same manner. As such, an agent a_i with $\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$, operating around a population of agents configured oppositely, i.e. $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$, will be exploited and will not get the assistance it needs from its selfish peers. Therefore, a_i needs to take into account the behaviour of individual agents as well as the whole population, so that it can conserve its resources when faced with exploiters.

In order to aid the agent such that it is not exploited, Algorithm 2.a is extended through Algorithm 2.b. The main behaviour expected to be produced by this extension is the following. If an agent a_i finds itself in a selfish society then, if the perceived helpfulness of the particular agent a_j asking for help is below a given threshold, a_i will lower its δ to a value as low as 0.1, which means that if it will help with a probability 0.1. Conversely, if a_j's perceived helpfulness is above a given threshold, then independently of the population, a_i will increase its δ by $k * \Delta step$. The reason behind such choice is that, a_i should not penalize other helpful agents, even though the agent population it resides in is selfish. Finally, if the agent population is helpful, i.e. if the perceived helpfulness is above a given threshold, although a_j's perceived helpfulness is low, then a_i penalizes a_j only after a certain number of unhelpful interactions. Thus, a_j is given the chance to change its behaviour so as to become more helpful.

5 Simulations

This section describes the simulation setups and results for configurations 1 and 2 of the agent model.

5.1 Setup and Results for C1

C1 Setup The agents are implemented as ROS nodes [37] and interact through the publish/subscribe mechanisms and services. The simulation is limited to three types of interactions, the non-committal broadcast (implemented through ROS publish/subscribe), and the one-to-one delegation (implemented through ROS services), and the emergent chain of one-to-one delegations. The agents make themselves continuously known to each other by broadcasting their identity and the list of tasks the are able to execute. Note that, agents are not assumed to have the same global goals. Thus each one has its own objective, however can put its capabilities to the service of others if the need arises.

Simulations were setup in order to investigate the utility of the agent population. Utility is measured on two levels, (i) the degree of completion of dependent tasks (CD), and (ii) the degree of dropped tasks (DD). Dependent tasks are tasks which depend on other tasks in order to have a higher chance for a successful outcome. As such, CD is calculated by dividing the number of dependent tasks completed over the number of dependent tasks attempted (Equation 1).

$$CD = \frac{Depend\ Tasks\ Completed}{Depend\ Tasks\ Attempted} \tag{1}$$

DD is calculated by dividing the number of dropped tasks over the number of attempted tasks (Equation 2).

$$DD = \frac{Tasks\ not\ Completed}{Tasks\ Attempted} \tag{2}$$

Two parameters were manipulated in the tests, the dependency degree between tasks, and the willingness to give help (δ). The former refers to the percentage of tasks (in a given set) that depend on other tasks for a higher probability of success. The latter represents the probability that an agent will assist another agent when requested.

In the simulations there are 10 tasks that are defined as abstract entities. Each task is simulated as a $for\ loop$ which runs for a specific amount of iterations. The list of tasks an agent is able to do is a subset of 10 tasks. More than one agent is able to execute the same task. As a result, there is diversity with respect to whom an agent can ask for help. On each run of the simulation, the same agent provides the same set of tasks. Moreover, the dependencies between tasks are given before-hand. In this work, the dependence is limited to one task, as opposed to many. An agent asking for help will wait $\Delta t = 60sec$ before dropping the request. This value is given before runtime and is assumed equal to all agents.

Three sets of simulation runs were conducted. Each set includes three independent runs for the same fixed parameters: population size (*popsize*), dependency degree (*depdeg*), and Greek delta δ. Simulation set 1: δ is static, i.e. does not change throughout one simulation run, and *popsize* takes the values 10 and 30 for separate runs. The parameter δ takes its values in the segment $[0.0, 0.25, 0.5, 0.75, 1.0]$. *Depdeg* takes values in $[10\%, 25\%, 50\%, 75\%, 100\%]$. Simulations were ran for both population sizes, for each combination of δ and dependence degree. The combination of all three parameters means that the first set consists of 50 simulation runs. Simulation set 2: *popsize* is fixed to 10, whereas δ is in a finer grained segment $[0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]$. The segment for *depdeg* is the same as in the previous set. Simulation set 3: *popsize* is equal to 10, however in this case δ is dynamic, i.e. it changes during runtime. The initial values of δ are in the segment $[0.0, 0.3, 0.5, 0.7, 1.0]$. In the final set, two simulations were run, (i) only one agent has dynamic δ, (ii) all agents have dynamic δ. The segment for *depdeg* remains the same.

During all simulation runs, agents can decide to perform a task t_i, or can receive a request for that task. At the beginning of a task, the agent checks the list of dependencies. If there is any such dependency, then the agent chooses whom to ask for assistance, by consulting its list of known agents which are able to perform t_i. The selection is done in the following way. The agent perceived as the most helpful in the past is chosen with probability 0.7, or random with probability 0.3. The 0.3/0.7 ratio is arbitrary. This scheme helps the agents to explore their options. The perceived helpfulness (*ph*) of an agent is computed by dividing the times it has given a response over the total times it was requested for help (Equation 3).

$$ph = \frac{Requests\ Handled}{Total\ Requests} \tag{3}$$

C1 Results The simulations were conducted in order to verify the hypothesis that agents with dynamic willingness to give help complete more of their dependent tasks as compared to agents with static willingness. The corresponding results are visualized as heat maps (Figure 6), in which the x-axis represents the *depdeg*, an the y-axis represents δ, and the intensity of the color represents the percentage of completed tasks summed over all agents in the population, computed as a mean over the three independent runs.

Outcomes from the first set of simulation runs are shown in Figures 6a, 6b, 6d, and 6e. It is possible to observe that for low dependence degree (10%), agents with a low willingness to give help (0.0), complete roughly 30% of their dependent tasks. This figure agrees with the probability that an agent is able to achieve a task by itself, when asking for help has failed. On the contrary, agents with willingness to give help, complete more dependent tasks, without noticeably impacting DD. Moreover, *popsize* does not show to have an impact on neither CD (Figures 6a, 6b) nor DD (Figures 6c, 6d). This is the reason why *popsize* equal to 10 was used in the remaining simulations. The outcomes for the second set of simulation runs (given in c and f) are consistent with the first set.

The third set of simulation runs covers the case in which δ is dynamic. As such, in the y-axis for Figures g, j are shown δ's initial values, δ_{init}. It is observed that for lower dependence degree, the population as a whole accomplishes more dependent tasks, thus CD increases, as compared to the static δ scenarios. This holds for both cases, i.e. only one agent has dynamic δ (Figures 6g, 6i) and all agents have dynamic δ (Figures 6h, 6j). Furthermore, the maximum CD is reached in the case where all agents have dynamic δ . On the other hand, as the dependence degree increases, so does DD and this is consistent through the three sets of runs. When all tasks depend on some other task (i.e. dependence degree is equal to 100%), DD reaches its maximum value, and CD is circa 0.3, which represents the probability for an agent to accomplish the task by itself.

The parameter DD influences the willingness to give help, and as a result the behaviour of the agent (as is observed in Figures 6a, 6b). In these simulations, two thresholds $\theta_{low} = 0.3$ and $\theta_{high} = 0.7$, are used in order to regulate δ as a function of DD as follows. If DD is higher than θ_{high}, then δ will decrease with a step of $\delta_{step} = 0.05$. If DD is lower than θ_{low}, then δ will increase with the same δ_{step}. If DD lies between θ_{low} and θ_{high}, the agent will compare the current DD with the one before. If such difference is bigger than 0.01 in absolute value, then δ will be updated, i.e. will increase if DD has gone down, and decrease otherwise.

5.2 Setup and Results for C2

C2 Setup Agents are implemented as ROS nodes [37]. The communication between them is realized by using (i) the publish/subscribe mechanism in ROS and (ii) a tailored action server mechanism able to handle multiple requests at once which extends ROS's Simple Action Server implementation. Agents continuously broadcast their identity and list of tasks they can perform to each other through the publish/subscribe. In order to issue requests for help (one-to-one communications), agents use the action-server mechanism.

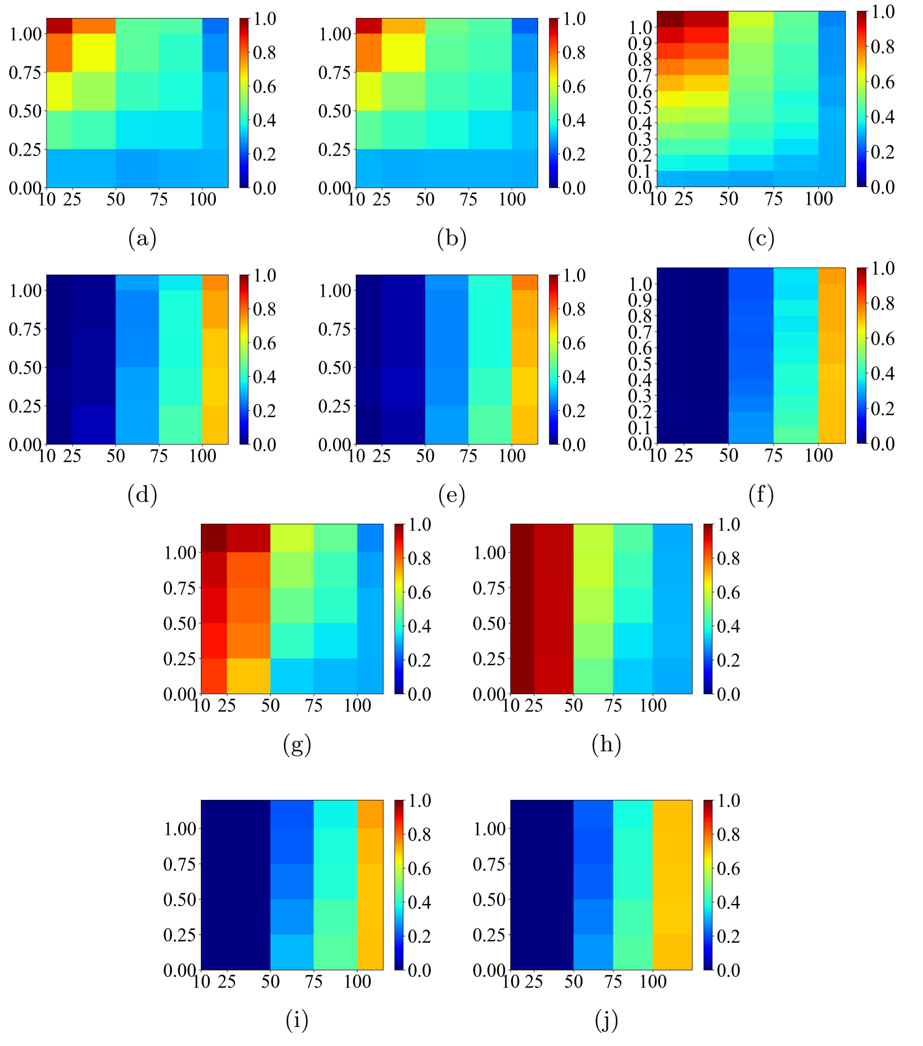


Fig. 6: Heat maps of CD and DD utility measures, for simulations with static δ and dynamic δ, and different popsize [7]. (a) CD for popsize $=$ 10 with static δ. (b) CD for popsize $=$ 30 with static δ. (c) CD for popsize $=$ 10 with finer resolution of static δ. (d) DD for popsize $=$ 10 with static δ. (e) DD for popsize $=$ 30 with static δ. (f) DD for popsize $=$ 10 with finer resolution of static δ. (g) CD for popsize $=$ 10, one agent with dynamic δ. (h) CD for popsize $=$ 10, all agents with dynamic δ. (i) DD for popsize $=$ 10, one agent with dynamic δ. (j) DD for popsize $=$ 10, all agents with dynamic δ.

Agent behaviour was studied in three different simulation scenarios in order to verify the three hypotheses H1, H2, and H3. The first scenario (S1) addresses H1. The population of agents contains one individual with $\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$, while the rest is configured with $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$. Thus one agent has high willingness to give help and low willingness to ask for help, whereas the rest of the population has low willingness to give help and high willingness to ask for help. The values in the tuples change during the simulation. Two types of simulations are ran. (i) All agents calculate their δ based on Algorithm 2.a. The simulation time is equal to $t_s = 1h$. (ii) All agents calculate their δ by applying to Algorithm 2.a the extension provided in 2.b. In this case simulation time is $t_s = 3h$. This is to ensure that enough interactions take place for the agents to be able to adapt.

The second scenario (S2) addresses H2. The opposite agent configuration is shown, i.e. one agent is configured with $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$, while the others start off with $\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$. In this case, an agent with low willingness to give help and high willingness to ask for help is put among agents with high willingness to give help and low willingness to ask for help. Two types of simulations are ran. (i) All agents calculate their δ through Algorithm 2.a, and (ii) all agents calculate their δ by applying to Algorithm 2.a the extension provided in 2.b.

In the final scenario (S3 which addresses H3), half of the population is configured with $\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$, while the other half is configured with $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$. As in the previous scenarios, two types of simulations are run. (i) All agents calculate their δ by using Algorithm 2.a. (ii) All agents calculate their δ by applying to Algorithm 2.a the extension provided in 2.b.

Every simulation was repeated for a population size *popsize* equal to 10 and 30. In all cases the difficulty of the simulation, i.e. the probability that an agent needs the assistance of another agent for a task, is set to a low value equal to 0.2. The perceived willingness to help of an agent a_j is calculated through Equation 3 by an agent a_i,

$$ph = \frac{rs}{rg} \tag{4}$$

where rs - total number of requests a_i has sent to a_j, rg - number of acceptance responses gotten from a_j. The agent to ask for help is chosen according to the following rules. As long as an agent a_i has a list of agents with which it does not have any past experience, it will choose randomly one of them. This is to ensure that an agent creates a past history with all the others early on in the simulation. Otherwise, the agent will choose randomly with a probability $P_1 = 0.4$ and according to Equation 5 with probability $P_2 = 0.6$,

$$\beta = max(\{ph_1, ...ph_i, ...ph_n\}) \tag{5}$$

where ph_i - perceived helpfulness of agent i, n - number of agents. Agent risk in this paper is expressed as $1 - ph$. The culture variable a_k is calculated by averaging on the perceived willingness of all agents with which there is past experience, as given in Equation 6,

$$C = \frac{\sum_{i=1}^{k} ph_i}{k} \tag{6}$$

where ph_i - perceived helpfulness of agent i, k - number of agents with which there is past experience. The agent's own performance is calculated as in Equation 7.

$$\mu = \frac{tc}{ta} \tag{7}$$

where tc - total tasks completed, ta - total tasks attempted. The variables for environmental risk and task progress do not change during the course of the simulation. Thus their effect on willingness does not change as well. Environmental risk is kept at a value equal to 0.2, and influences δ by $+\Delta step$, and γ by $-\Delta step$, where $\Delta step = 0.05$. Moreover, since reasoning on γ is done only at the beginning of a task, the agent cannot measure progress for itself, and thus the variable will affect γ by $-\Delta step$. Finally, since tasks are presently simulated as atomic steps, task trade-off has not been considered in these simulations.

C2 Results The results shown in this subsection are taken as a mean over three independent runs, in which all parameter settings are the same, and visualized through bar plots. Moreover, for every simulation scenario, results corresponding to agents with the same initial configuration are averaged.

Figures 7 and 8 display results which address H1. Figure 7 shows the numerical values for a population of agents with $popsize = 10$, whilst figure 8 for $popsize = 30$. It is evident that the agent $\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$ (figure 7, left side) adapts to the selfish population by comparing the rR/rRA ratio in figure 7 (a) and (b). Note also that agents $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$ flood each other with the high number of requests made. Observing the bottom graphs in (a) and (b), it is possible to estimate how much an agent works for itself and for others by comparing daO/daNO ratios. Agent $\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$ attempts dominantly more dependent tasks for others in (a), while the opposite is true in (b). This is reflected in the numbers for completed dependent tasks by comparing dco/dcNO ratios. Thus, H1 holds in the implemented scenario for both $popsize = 10,\ 30$.

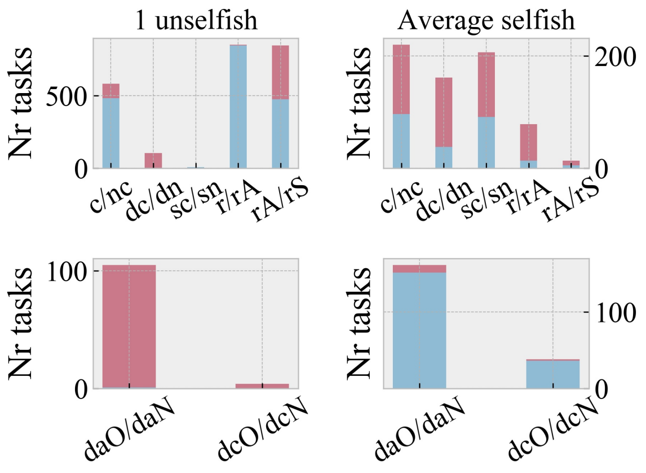
Figures 9 and 10 show results which address H2. Figure 9 gives the values for a population of agents with $popsize = 10$, whilst 10 for $popsize = 30$. Note that, by comparing the dc/dnc ratio in figure 9 (a) and (b), the selfish agent $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$ completes less dependent tasks in a society which will not be blindly helpful, but will tune helpfulness towards those individuals that respond in kind. As a result, the rest of the population ($\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$) will continue to do very well and will, after a period of learning, stop helping the selfish agent. Figure 9 (c) considers a scenario in which the selfish agent $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$ adapts to the population of agents by becoming more helpful, thus it performs better than the agent in (b). To get the two different behaviours, the factor k in Algorithm 2.b was taken equal to 4 in (b), and 8 in (c). Figure 10 shows that the selfish agent completes a higher percentage of dependent tasks. The reason is argued to be the size of the population. It can take a population longer to adapt to a selfish individual because more of the required interactions need to take place. Until that point is reached, the selfish agent will do fairly well. Thus, H2 also holds in the implemented scenarios.

Figures 11 and 12 show results which address H3. Figure 11 gives the values for a population of agents with $popsize = 10$, whilst 12 for $popsize = 30$. Note that agents using Algorithm 2.b adapt to the part of the population configured to exploit. Moreover, a helpful agent will achieve more tasks in average than its opposite. This holds for both population sizes. Nevertheless, overall performance for the helpful agents falls (if compared with results in figures 9 and 10). This is because, the algorithm to choose an agent to ask for help does always pick from the entire population of agents, helpful and selfish combined. A finer selective method could be applied, so that helpful agents always pick between each other. The effects of the population size in the completed dependent tasks is mildly present, however it is not dominant. Thus, H3 also holds in the current implementation.

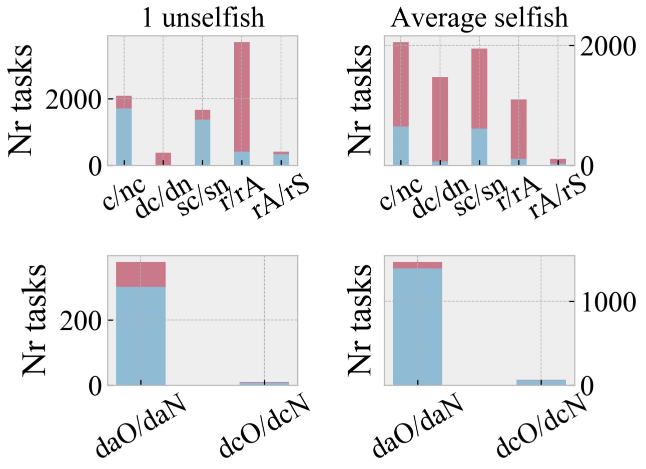
6 Discussion

The work presented in this paper describes an agent model that exhibits adaptive autonomous behaviour by manipulating its willingness to interact, which in turn is composed by the willingness to give help (δ) and the willingness to ask for help (γ). The willingness to interact is expressed with probabilities in order to model the non-deterministic aspect of interactions. In some cases the agent might be completely sure that it needs help, e.g. lacks one of the abilities required for a task. However, in other cases this may not happen. For instance, assume an agent which is progressing slowly on a task. Its willingness to ask for help will increase. However, let us also assume that the agent is among rather unhelpful or unsuccessful agents. This element will decrease the willingness because the probability that those agent will be helpful now is low. In this scenario, the agent's willingness is influenced by contradicting factors that do not necessarily lead to a deterministic choice. There is a risk associated in choosing to ask or not ask for help. Consequently, the willingness to ask for help implicitly assesses this risk, and such assessment is probabilistic. In economics, this kind of parameter is used to model risk tolerance [38]. Agents which are representatives of business entities, sign contracts with other entities based on their willingness. Signing some contracts might be not allowed and thus subject to fines. The latter are considered punishment for undesired behavior. The higher the fines, the higher the risk is of signing a contract with an agent.

In the first part of this paper ($C1$), the agent consists only of δ, which is calculated based on the agent's own performance. In the second part ($C2$), implements the whole willingness to interact behaviour. Additionally in $C2$, the perceived helpfulness of the population is used as an additional input in the calculation of δ. In this way, agents who are configured to be helpful can reduce exploitation by other agents which will not respond to helpfulness in kind. Algo-

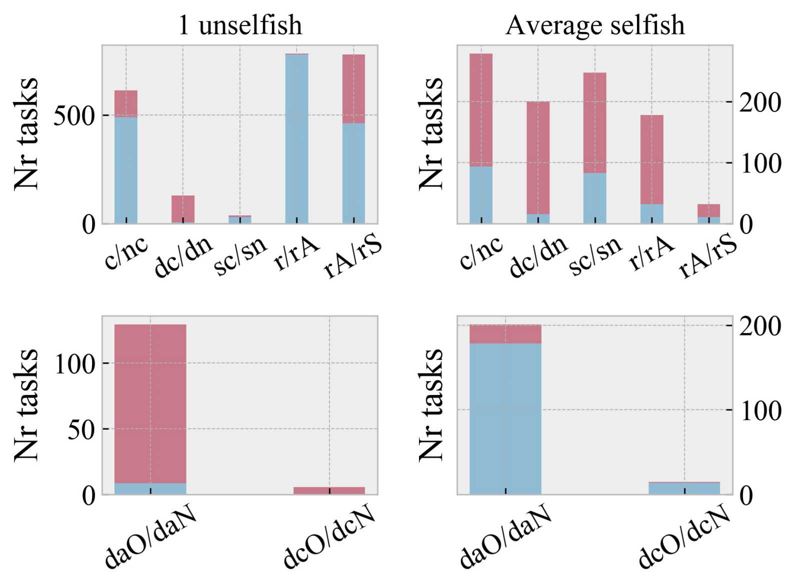


(a)

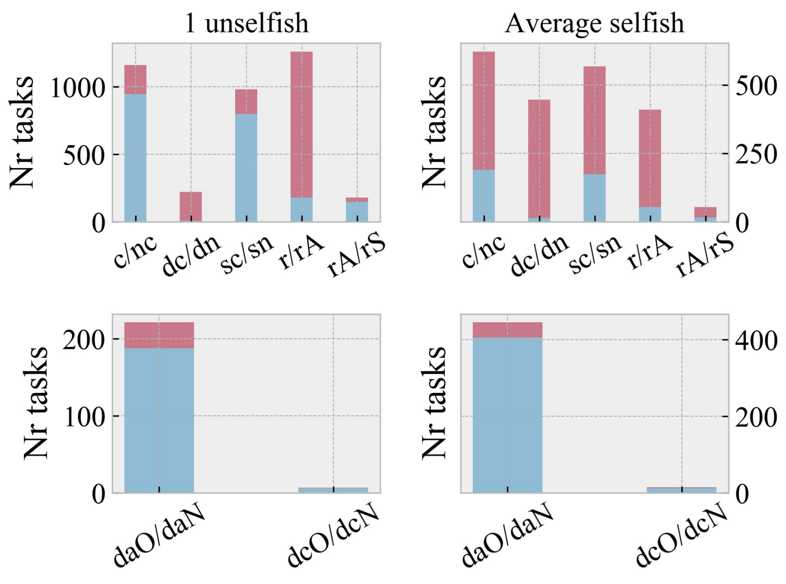


(b)

Fig. 7: Results for S1, for *popsize* = 10. Notation: c/nc - completed/not completed tasks (c - blue bottom bar, nc - top red bar), dc/dn - dependent completed/dependent not completed tasks, sc/sn - completed self-generated/not completed self generated, r/rA - request received/requests accepted, rA/rS - requests accepted/requests succeeded (as perceived by the agent who is helping, daO/daN - dependent self-generated tasks attempted/dependent not self-generated tasks attempted, dcO/dcN - dependent self-generated completed/ dependent not self-generated completed). (a) Agents update δ with Algorithm 2.a. (b) Agents update δ with the extension proposed in Algorithm 2.b. Figures on the left correspond to the agent $\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$, while those on the right are averaged over the agents $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$

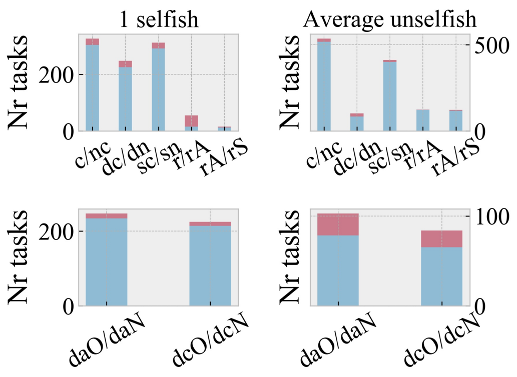


(a)

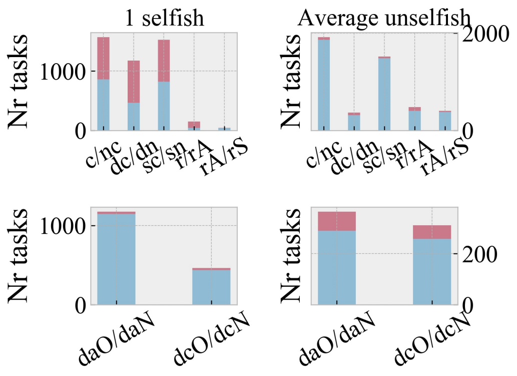


(b)

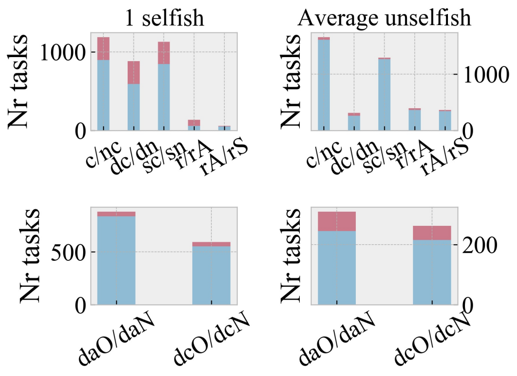
Fig. 8: Results for S1, for $popsize = 30$. Notation same as Figure 7. (a) Agents update δ with Algorithm 2.a. (b) Agents update δ with the extension proposed in Algorithm 2.b. Figures on the left correspond to the agent $\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$, while those on the right are averaged over the agents $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$



(a)

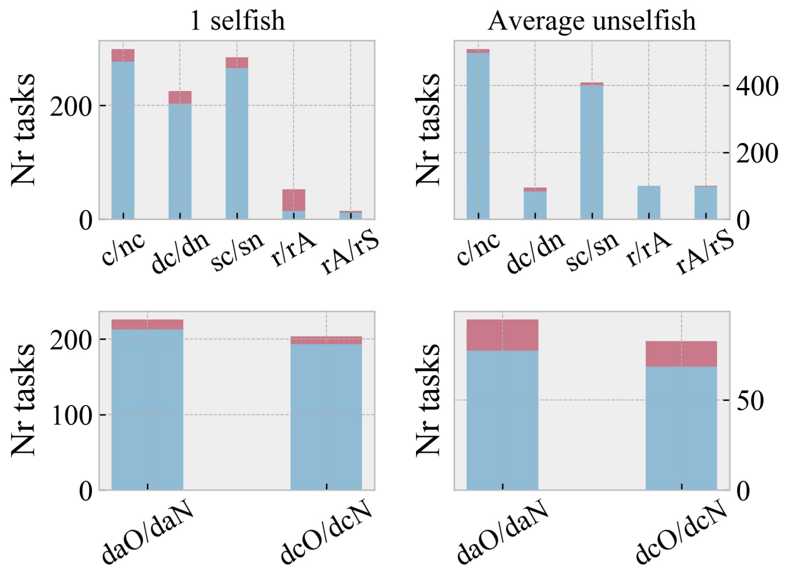


(b)

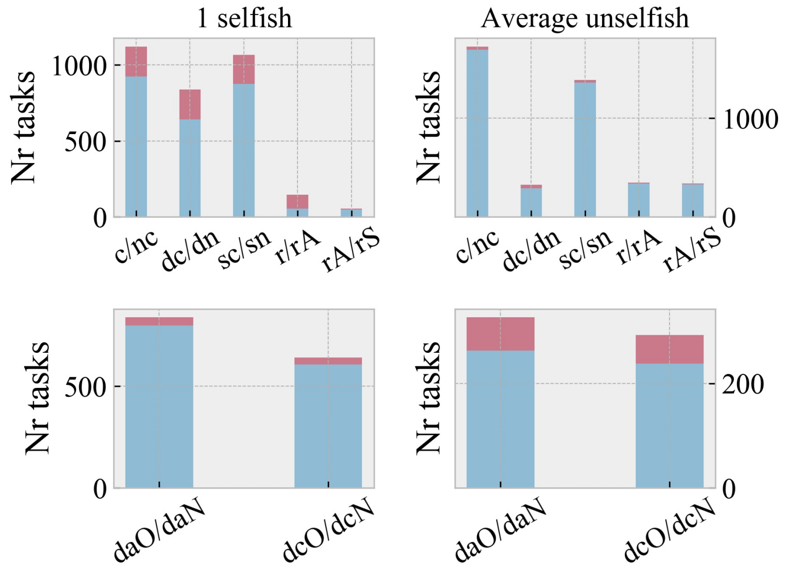


(c)

Fig. 9: Results for S2, for *popsize* $= 10$. Notation same as Figure 7. (a) Agents update δ with Algorithm 2.a. (b) Agents update δ with the extension proposed in Algorithm 2.b. Figures on the left correspond to the agent $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$, while those on the right are averaged over the agents $\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$. (c) ...

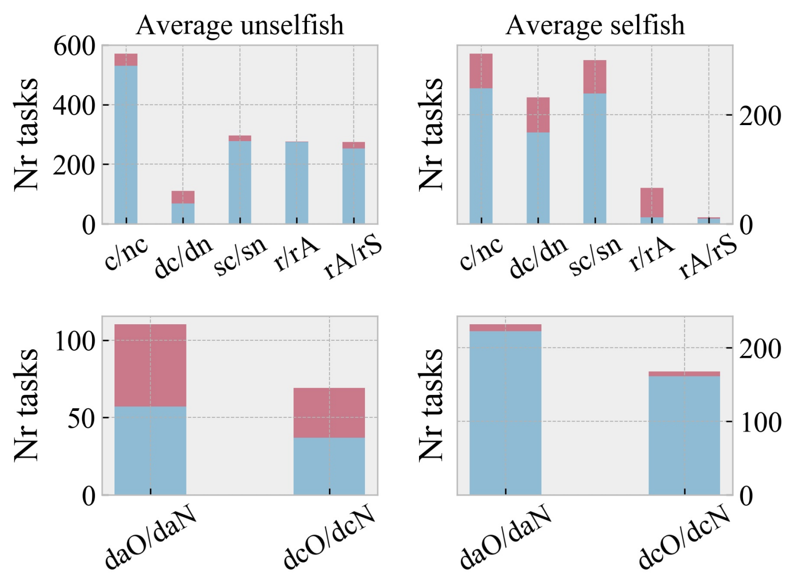


(a)

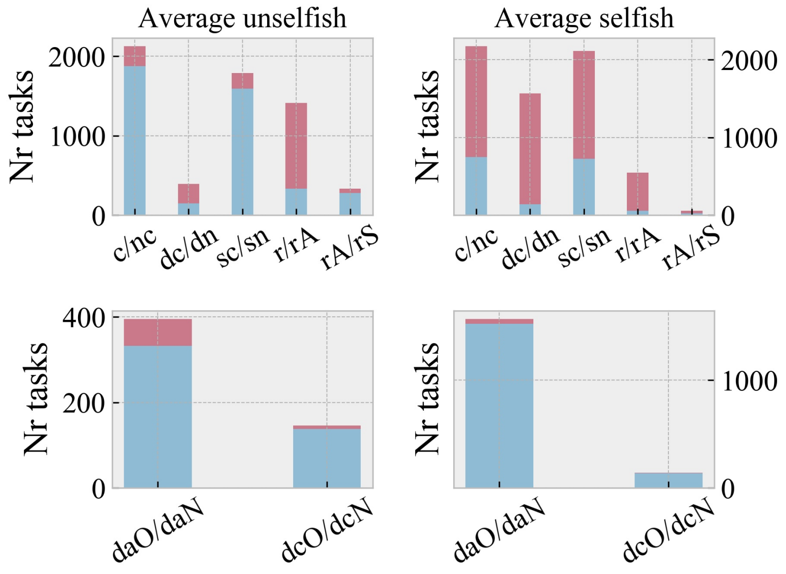


(b)

Fig. 10: Results for S2, for $popsize = 30$. Notation same as Figure 7. (a) Agents update δ with Algorithm 2.a. (b) Agents update δ with the extension proposed in Algorithm 2.b. Figures on the left correspond to the agent $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$, while those on the right are averaged over the agents $\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$

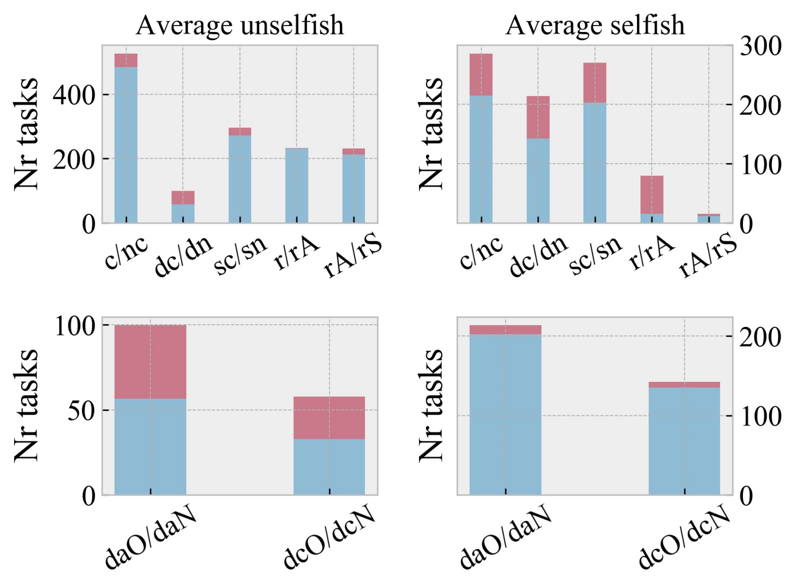


(a)

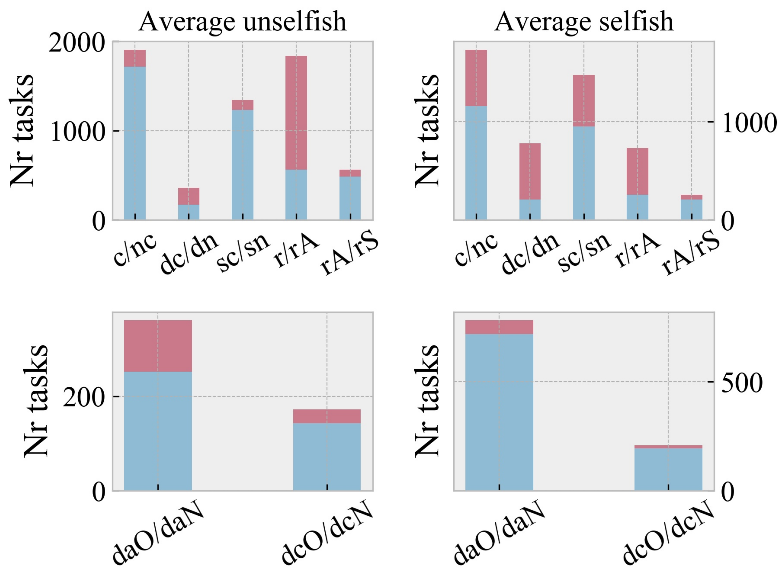


(b)

Fig. 11: Results for S3, for $popsize = 10$. Notation same as Figure 7. (a) Agents update δ with Algorithm 2.a. (b) Agents update δ with the extension proposed in Algorithm 2.b. Figures on the left correspond to the average of half the population of agents with $\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$, while those on the right are averaged over the other half $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$



(a)



(b)

Fig. 12: Results for S3, for $popsize = 30$. Notation same as Figure 7. (a) Agents update δ with Algorithm 2.a. (b) Agents update δ with the extension proposed in Algorithm 2.b. Figures on the left correspond to the average of half the population of agents with $\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$, while those on the right are averaged over the other half $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$

rithm 2.a was extended through 2.b and used in the simulations, however other similar algorithms could be implemented that achieve the same end. Algorithm 2.b is purely experience based, and thus agents will need to experience a certain number of interaction before the learning takes place. Moreover, its efficiency – number of interactions needed for the learning – depends on the size of the population, as well as the ratios of selfish and unselfish agents. Furthermore, only two types of agents are used, $\langle \delta_0 = 0.0, \gamma_0 = 1.0 \rangle$ and $\langle \delta_0 = 1.0, \gamma_0 = 0.0 \rangle$. It can be argued that a mixed population will have additional effects on the results presented here.

This work also relates to computational trust models for multi-agent systems. The reason is that, the willingness to interact is used when deciding to depend upon another, and allowing others to depend on one as well. According to Falcone *et al.* [39], by trusting, an agent has made a decision to rely on another. The other way around, i.e. allowing others to depend on one, could also be argued as a case for trust. The agent that is helping, is putting its resources in the service of another. It is reasonable to think that help should go more towards those who have returned the favour. There is a fair amount of trust models in the literature, as well as classifications based on different criteria for them [40]. These models typically are experience-based (judging on personal direct interactions), reputation-based (judging based on third party opinions), or combined [40].

The aim of future work is to investigate the concept of *agent culture* and trust more thoroughly. Apart from perceived helpfulness, other characteristics could be used, for example perceived expertise, or perceived load. There is a difference between perceived helpfulness and perceived expertise. The latter is a better measure for successful outcomes of tasks. Assume an agent a_i which asks for help an agent a_j, and a_j accepts to help. However, being willing to help does not guarantee success, in fact agent a_j can fail for any reason. From this perspective, a flaw of perceived helpfulness as an indicator of helpfulness becomes apparent. As a result, a combined measure of perceived helpfulness and expertise can give a better picture of the agents taken as individuals and as a whole. Nevertheless, the degree of load of agents can help refine the measure even further. Consequently, the problem that arises is how can an agent estimate the load of other agents, taken individually and as a whole. The perceived load could be combined with the perceived willingness to ask for help from a specific agent and be used in its analysis, and that of the whole population. The implementation of the agent has to be expanded so that task execution is simulated through separate steps, so as to be interrupt-able, so that it represents a more realistic scenario. Another desired feature for the system is to enable the agent to deal with several dependencies at the same time.

Application domains that motivate this research include but are not limited to search and rescue, agriculture, and other areas in which autonomous systems can assist, and sometimes replace, human labour due to the challenging working conditions. It may be desired for these systems to be deployed far from the operator, and work in conditions where the communication with the former can be unreliable. Also, it may be desired that the systems operate by aiding each other as the needs arise. This means that agents need to reason about when and how to interact with each other. The AA agent described in this paper represents one possible approach that can be applied in these types of application. Moreover, the modeling is done on a high-level, and hence is not task specific in fact tasks are defined in abstract terms.

7 Conclusion

In this paper it is shown, through computer simulations, how the adaptive autonomous behaviour of an agent can be tailored to reflect the culture of the population within which it operates, such that the exploitation of the agent can be decreased. Firstly, it is shown how an agent highly willing to give help, can adapt to agents configured oppositely, and thus lower its exploitation. Secondly, when an unhelpful agent is introduced into a helpful population, it will be able to exploit up until the point in which the rest of the agents will have created a history of negative past experiences. The dependence on the population size is also present in the simulation results, i.e. one selfish agent will be able to succeed for longer in bigger populations because more interactions are needed. Nevertheless, if the agent can change its behaviour from selfish to unselfish, it will outperform agents which are more inflexible. Finally, when the population is split in half between selfish and unselfish agents, the latter is still able to decrease it exploitation after enough interactions have taken place. However, since the algorithm to select the agent is such that it will always pick from the whole pool, the performance of unselfish agents decreases as well, compared to the scenario in which the latter dominates. A finer selection algorithm could address this issue.

References

1. Hardin, B., Goodrich, M.A.: On using mixed-initiative control: A perspective for managing large-scale robotic teams. In: Proceedings of the 4th ACM/IEEE international conference on Human robot interaction. pp. 165–172. ACM (2009)
2. Johnson, M., Bradshaw, J.M., Feltovich, P.J., Jonker, C.M., Van Riemsdijk, B., Sierhuis, M.: The fundamental principle of coactive design: Interdependence must shape autonomy. In: Coordination, organizations, institutions, and norms in agent systems VI, pp. 172–191. Springer (2011)
3. Fong, T., Thorpe, C., Baur, C.: Collaborative control: A robot-centric model for vehicle teleoperation, vol. 1. Carnegie Mellon University, The Robotics Institute (2001)
4. Brookshire, J., Singh, S., Simmons, R.: Preliminary results in sliding autonomy for assembly by coordinated teams. In: Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on. vol. 1, pp. 706–711. IEEE (2004)
5. Parasuraman, R., Sheridan, T.B., Wickens, C.D.: A model for types and levels of human interaction with automation. IEEE Transactions on systems, man, and cybernetics-Part A: Systems and Humans 30(3), 286–297 (2000)

6. Castelfranchi, C.: Founding agent's 'autonomy' on dependence theory. In: Proceedings of the 14th European Conference on Artificial Intelligence. pp. 353–357. IOS Press (2000)
7. Frasheri, M., Çürüklü, B., Ekström, M.: Towards collaborative adaptive autonomous agents. In: 9th International Conference on Agents and Artificial Intelligence 2017 ICAART, 24 Feb 2017, Porto, Portugal (2017)
8. Lewis, B., Tastan, B., Sukthankar, G.: An adjustable autonomy paradigm for adapting to expert-novice differences. In: Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on. pp. 1656–1662. IEEE (2013)
9. Muszynski, S., Stückler, J., Behnke, S.: Adjustable autonomy for mobile teleoperation of personal service robots. In: RO-MAN, 2012 IEEE. pp. 933–940. IEEE (2012)
10. Birk, A., Pfingsthorn, M.: A hmi supporting adjustable autonomy of rescue robots. In: Robot Soccer World Cup. pp. 255–266. Springer (2005)
11. Goodrich, M.A., Olsen, D.R., Crandall, J.W., Palmer, T.J.: Experiments in adjustable autonomy. In: Proceedings of IJCAI Workshop on Autonomy, Delegation and Control: Interacting with Intelligent Agents. pp. 1624–1629 (2001)
12. Lin, L., Goodrich, M.A.: Sliding autonomy for uav path-planning: Adding new dimensions to autonomy management. In: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems. pp. 1615–1624. International Foundation for Autonomous Agents and Multiagent Systems (2015)
13. Dorais, G., Bonasso, R.P., Kortenkamp, D., Pell, B., Schreckenghost, D.: Adjustable autonomy for human-centered autonomous systems. In: Working notes of the Sixteenth International Joint Conference on Artificial Intelligence Workshop on Adjustable Autonomy Systems. pp. 16–35 (1999)
14. Kortenkamp, D., Keirn-Schreckenghost, D., Bonasso, R.P.: Adjustable control autonomy for manned space flight. In: Aerospace Conference Proceedings, 2000 IEEE. vol. 7, pp. 629–640. IEEE (2000)
15. Calhoun, G.L., Goodrich, M.A., Dougherty, J.R., Adams, J.A.: Human-autonomy collaboration and coordination toward multi-rpa missions. Remotely Piloted Aircraft Systems: A Human Systems Integration Perspective p. 109 (2016)
16. Barnes, M.J., Chen, J.Y., Jentsch, F.: Designing for mixed-initiative interactions between human and autonomous systems in complex environments. In: Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on. pp. 1386–1390. IEEE (2015)
17. Stubbs, K., Hinds, P.J., Wettergreen, D.: Autonomy and common ground in human-robot interaction: A field study. IEEE Intelligent Systems 22(2) (2007)
18. van der Vecht, B., Dignum, F., Meyer, J.C.: Autonomy and coordination: Controlling external influences on decision making. In: Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT'09. IEEE/WIC/ACM International Joint Conferences on. vol. 2, pp. 92–95. IEEE (2009)
19. Landén, D., Heintz, F., Doherty, P.: Complex task allocation in mixed-initiative delegation: a uav case study. In: International Conference on Principles and Practice of Multi-Agent Systems. pp. 288–303. Springer (2010)
20. Kim, S., Kim, M., Lee, J., Hwang, S., Chae, J., Park, B., Cho, H., Sim, J., Jung, J., Lee, H., et al.: Team snu's control strategies for enhancing a robot's capability: Lessons from the 2015 darpa robotics challenge finals. Journal of Field Robotics (2016)
21. Farinelli, A., Raeissi, M.M., Brooks, N., Scerri, P., et al.: Interacting with team oriented plans in multi-robot systems. Autonomous Agents and Multi-Agent Systems pp. 1–30 (2016)

22. Côté, N., Canu, A., Bouzid, M., Mouaddib, A.I.: Humans-robots sliding collaboration control in complex environments with adjustable autonomy. In: Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology-Volume 02. pp. 146–153. IEEE Computer Society (2012)
23. Chitalia, Y., Zhang, W., Hyun, B., Girard, A.: A revisit-based mixed-initiative nested classification scheme for unmanned aerial vehicles. In: American Control Conference (ACC), 2014. pp. 1793–1798. IEEE (2014)
24. Feltovich, P.J., Bradshaw, J.M., Clancey, W.J., Johnson, M.: Toward an ontology of regulation: Socially-based support for coordination in human and machine joint activity. In: International Workshop on Engineering Societies in the Agents World. pp. 175–192. Springer (2006)
25. Bradshaw, J.M., Jung, H., Kulkarni, S., Johnson, M., Feltovich, P., Allen, J., Bunch, L., Chambers, N., Galescu, L., Jeffers, R., et al.: Kaa: policy-based explorations of a richer model for adjustable autonomy. In: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems. pp. 214–221. ACM (2005)
26. Scerri, P., Pynadath, D.V., Tambe, M.: Towards adjustable autonomy for the real world. Journal of Artificial Intelligence Research 17(1), 171–228 (2002)
27. Goodrich, M.A., Schultz, A.C.: Human-robot interaction: a survey. Foundations and trends in human-computer interaction 1(3), 203–275 (2007)
28. Suzanne Barber, K., Goel, A., Martin, C.E.: Dynamic adaptive autonomy in multi-agent systems. Journal of Experimental & Theoretical Artificial Intelligence 12(2), 129–147 (2000)
29. Martin, C., Barber, K.S.: Adaptive decision-making frameworks for dynamic multi-agent organizational change. Autonomous Agents and Multi-Agent Systems 13(3), 391–428 (2006)
30. Johnson, M., Bradshaw, J.M., Feltovich, P.J., Jonker, C.M., Van Riemsdijk, M.B., Sierhuis, M.: Coactive design: Designing support for interdependence in joint activity. Journal of Human-Robot Interaction, 3 (1), 2014 (2014)
31. Klien, G., Woods, D.D., Bradshaw, J.M., Hoffman, R.R., Feltovich, P.J.: Ten challenges for making automation a" team player" in joint human-agent activity. IEEE Intelligent Systems 19(6), 91–95 (2004)
32. Tambe, M.: Agent architectures for flexible. In: Proc. of the 14th National Conf. on AI, USA: AAAI press. pp. 22–28 (1997)
33. Laird, J.E.: The Soar cognitive architecture. MIT Press (2012)
34. Schurr, N., Marecki, J., Tambe, M.: Improving adjustable autonomy strategies for time-critical domains. In: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1. pp. 353–360. International Foundation for Autonomous Agents and Multiagent Systems (2009)
35. McGill, S.G., Yi, S.J., Yi, H., Ahn, M.S., Cho, S., Liu, K., Sun, D., Lee, B., Jeong, H., Huh, J., et al.: Team thor's entry in the darpa robotics challenge finals 2015. Journal of Field Robotics (2016)
36. Castelfranchi, C., Falcone, R.: Principles of trust for mas: Cognitive anatomy, social importance, and quantification. In: Multi Agent Systems, 1998. Proceedings. International Conference on. pp. 72–79. IEEE (1998)
37. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: Ros: an open-source robot operating system. In: ICRA workshop on open source software. vol. 3, p. 5. Kobe (2009)

38. Cardoso, H.L., Oliveira, E.: Adaptive deterrence sanctions in a normative framework. In: Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT'09. IEEE/WIC/ACM International Joint Conferences on. vol. 2, pp. 36–43. IEEE (2009)
39. Falcone, R., Castelfranchi, C.: Social trust: A cognitive approach. In: Trust and deception in virtual societies, pp. 55–90. Springer (2001)
40. Pinyol, I., Sabater-Mir, J.: Computational trust and reputation models for open multi-agent systems: a review. Artificial Intelligence Review 40(1), 1–25 (2013)

8 Appendix

The source code to replicate the C1 simulations is publicly available on github, under the following URL. Whereas, for C2 simulations the source code is available under the following URL: $https://github.com/gitting-around/gitagent.git$. Finally, the simulations for this paper were conducted on HP EliteBook 840 laptop with Ubuntu 14.04 and ROS Indigo.