Adaptive Autonomy in a Search and Rescue Scenario

Mirgita Frasheri*, Baran Cürüklü*, Mikael Ekström*, Alessandro Vittorio Papadopoulos*
*IDT, Mälardalen University, Sweden.
Email: *{mirgita.frasheri,baran.curuklu,mikael.ekstrom,alessandro.papadopoulos}@mdh.se

***Abstract*—Adaptive autonomy plays a major role in the design of multi-robots and multi-agent systems, where the need of collaboration for achieving a common goal is of primary importance. In particular, adaptation becomes necessary to deal with dynamic environments, and scarce available resources. In this paper, a mathematical framework for modelling the agents' willingness to interact and collaborate, and a dynamic adaptation strategy for controlling the agents' behavior, which accounts for factors such as progress toward a goal and available resources for completing a task among others, are proposed. The performance of the proposed strategy is evaluated through a fire rescue scenario, where a team of simulated mobile robots need to extinguish all the detected fires and save the individuals at risk, while having limited resources. The simulations are implemented as a ROS-based multi agent system, and results show that the proposed adaptation strategy provides a more stable performance than a static collaboration policy.**

I. Introduction

Adaptive autonomy (AA) is the ability of agents to adapt autonomously their behavior to continuously changing circumstances [1]. The implications of this definition are twofold: (i) each agent decides itself on its autonomy levels, and (ii) such decisions are taken continuously during execution. Furthermore, to change autonomy levels means to change the dependence relations among agents [2], e.g., an agent becomes less autonomous if it depends on the assistance of other agents to complete its task.

This paper deals with the development of *local* reasoning mechanisms, adapting the level of cooperation among agents based on several factors in order to improve the overall performance. To this end, AA is modeled herein through the *willingness to interact* [3], which allows agents to decide when to give or ask for help. AA behavior is relevant in those scenarios in which agents should be able to solve issues locally—especially when assistance from human operators is not available, e.g., due to unreliable communication channels. One such application domain is Search and Rescue (SAR). In this paper, a previously developed ROS-based agent simulation [4] has been extended with a SAR scenario, and used to evaluate the adaptive autonomous behavior of a group of agents in a software simulation. Nevertheless, the long-term goal is to develop AA strategies that can be used both in real and artificial environments, and for heterogeneous agents.

II. Background and Related Work

This section gives an overview of the SAR domain and autonomy models, and it presents related work in multi-agent systems (MASs) coordination and cooperation.

A. The SAR Domain

The SAR domain has served as a testbed for multi-robot and multi-agent research in the past years. The RoboCup Rescue competitions have been established since the 2000s [5], where researchers would validate their findings either in simulation (Rescue Simulation League), or with real robotic platforms (Rescue Robot League) [6]. The Rescue Simulation League is divided further into the virtual and agent competitions. In regards to the virtual robot competition, attempts are being made to provide interfaces to ROS (Robot Operating System) and Gazebo, due to the fact that the latter allow for better code-sharing among the community [7]. Kleiner et al. proposed the RMasBench [8], that could serve as a benchmark for coordination algorithms such as distributed constraint optimization problems (DCOP), whilst hiding low level details. Task allocation is usually expressed as the problem of assigning tasks to agents while also minimizing some cost function [9], [10]. The problem addressed in the present paper relates to how agents should interact with each other, i.e., when they should ask or give help based on their state, at any point in time. In particular, the ROS-based adaptive autonomous agent simulation developed in [4] was extended with the implementation of a SAR scenario.

Autonomy has been studied extensively in the literature, albeit there is no unified theory or general framework yet. Several autonomy changing schemes such as adaptive autonomy, adjustable autonomy, and mixed-initiative interaction, have been compared in the context of the coordination of large-scale teams of robots in a simulated Wilderness Search and Rescue (WiSAR) [11]. Agents with adaptive autonomy attempt at all times to keep the higher levels of autonomy. Furthermore, they do not go to the lowest level, in which the operator makes all the decisions. In the case of adjustable autonomy, the operator decides on the different levels of autonomy of agents. Whereas, mixed-initiative interaction allows both human and agent to make decisions on autonomy levels based on the circumstances. In these conditions, the third scheme yields the best results in terms of individuals found. Another approach uses a WiSAR scenario to evaluate the benefits of sliding autonomy in producing better paths (compared to two other methods that do not employ changes in autonomy), whilst not increasing the workload of the human operator [12].

In [13], a fire-fighter scenario is adopted in order to evaluate the reasoning mechanisms that allow fire-fighter agents to change their autonomy based on circumstances. Agent reasoning is based on two heuristic rules: (i) the more urgent a task becomes, the higher its priority, e.g., if the agent's health is running low then the task of taking a rest becomes more urgent as time passes, (ii) dedication to the organization, i.e., if the agent takes wrong information from the operator then it will take more initiative on its own.

B. Agent Cooperation and Coordination

Cooperation and coordination in MASs has been studied in other contexts that do not necessarily focus on the autonomy aspect. Theoretical approaches are concerned with providing taxonomies and definitions that set apart concepts such as cooperation, coordination, and collaboration [14] [15]. Recent years have also seen a rise of interest in what are called open MAS (parts of the system come from competing third parties) [16] and pervasive systems (embedded sensors, actuators, etc.) modeled as MASs [17]. Surveys in the area have identified the following classes: stigmergy, chemical, physical, biochemical, field-based, and swarms [17], [18]. Attention has also been paid to design patterns that target the interaction among components in self-adaptive systems [19], among which the stigmergy pattern (interaction through the mediation of the environment), centralized pattern, and peer to peer pattern. Others propose a framework in which specifications related to permissions and obligations among agents are adjusted at run-time [16]. Earlier works use a central coordinator, e.g., Kaa [20], which approves or rejects changes in permissions suggested by agents, and refers to an operator in case a decision is not reached. It is assumed that when such permissions change, so does the autonomy of agents.

III. The Adaptation Strategy

In this Section, the agent model proposed in [3], is revised, and the proposed adaptation strategy for AA is described.

A. The Agent Model

An agent is defined as a computer system that perceives and acts in its environment through its sensors and actuators respectively, and is capable of autonomous action [21]. Physical agents in addition to the previous characteristics, are able to manipulate the physical world through their effectors (e.g., legs, wheels, or manipulators). Any such agent needs to possess the necessary skills, abilities, knowledge, and resources to perform a particular task [22]. Moreover, an agent's operation is limited by its available resources, such as the battery level (e.g., for a robot), the allowed power consumption, or the available computational resources (e.g., for a software agent). Therefore, the agent can be described in terms of its (i) battery level, (ii) equipment (sensors, motors, manipulators, and actuators), (iii) skill/ability set, and (iv) knowledge. The agent's knowledge can refer to what the agent knows about itself, other agents, and the environment.

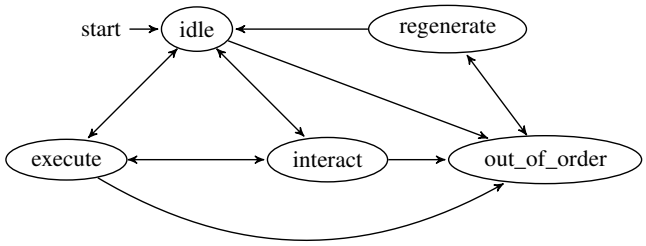


Figure 1: The proposed agent model composed of five states [3].

The agent model adopted in this paper is a finite-state machine, shown in Figure 1. The agent has five states: *idle*, *execute*, *interact*, *out_of_order* and *regenerate*. All agents start their operation in the *idle* state. In this state, a task can be generated. Consequently, the agent will switch to *execute*, where either it executes its task, or it can decide to interact with other agents asking for assistance. When the task is completed, the agent returns to *idle*. When an agent receives a request from another agent, it will switch to *interact*, where it will decide to accept or discard it. In the former case, the agent will put its current task, if any, back into a FIFO queue, and start the new task. Otherwise it will return to its previous state, either *idle* or *execute*. It is assumed that these queues are infinite long. Moreover, any agent cannot execute more than one task at the same time. When the energy level of an agent is low, it will switch to *out_of_order*, and soon after to *regenerate*, during which the recharging process takes place. If *regenerate* is successful, the agent goes to *idle*, and continues its normal operation. If *regenerate* fails, it will go to *out_of_order*. In the current setting, the agent is always assumed to regenerate successfully.

B. Willingness to Interact

Adaptive autonomous behavior is modeled through the willingness to interact, composed of two components referred to as the willingness to ask for, and give help [3]. The willingness to ask for help represents the likelihood with which an agent will ask another agent for help during the execution of a task (*execute* state). Whereas, the willingness to give help represents the likelihood with which an agent will provide help upon a request from another agent (*interact* state). In this paper, the goal of the agent is to decrease the disposition of asking for help and increase the disposition of giving help, when possible, promoting the cooperation among agents, assuming that cooperation can improve the performance of the MAS [23].

An agent computes the *willingness to ask for help* at time t, $\gamma_t \in [0, 1]$, by applying a correction to its initial value, as

$$\gamma_t = \mathrm{sat}(\gamma_0 + u_t), \tag{1}$$

where, $\mathrm{sat}(x) := \min(\max(x, 0), 1)$, and u_t is the *adaptive correction* at time t, computed as

$$u_t = \sum_{i=1}^{n} \phi_i f_t^{i\gamma}, \tag{2}$$

where n is the number of factors, ϕ_i, $i = 1, \ldots, n$, are weights (constant or calculated at runtime), such that $\phi_i \in [0, 1]$, $\sum_{i=1}^{n} \phi_i = 1$, and $f_t^{i\gamma}$, are the considered factors at time t.

An agent computes the *willingness to give help* at time t, $\delta_t \in [0, 1]$, by applying a correction to its initial value, as

$$\delta_t = \text{sat}(\delta_0 + v_t). \tag{3}$$

The correction v_t integrates the effect of several factors, and it is computed as

$$v_t = \sum_{i=1}^{n} \zeta_i f_t^{i\delta} \tag{4}$$

where n is the number of factors, ζ_i, $i = 1, \ldots, n$, are weights (constant or calculated during runtime), such that $\zeta_i \in [0, 1]$, $\sum_{i=1}^{n} \zeta_i = 1$, and $f_t^{i\delta}$ are the considered factors at time t. In the following paragraphs, the time subscript is omitted for the sake of simplicity.

The k-th factor f^{k*} is updated at every iteration based on the current measurements ψ_k of relevant quantities, e.g., the battery level of the agent, the accuracy of the agent in carrying out a specific task, etc, and on a minimal acceptable value $\psi_{k\,\min}$. The update rules for the willingness to give and ask for help are the following:

$$f^{k\gamma} = \begin{cases} \psi_{k\,\min} - \psi_k, & \psi_k > \psi_{k\,\min} \\ (1-\alpha)(\psi_{k\,\min} - \psi_k) + \alpha, & \psi_k \leq \psi_{k\,\min} \end{cases} \tag{5}$$

$$f^{k\delta} = \begin{cases} \beta(\psi_k - \psi_{k\,\min}) & \psi_k > \psi_{k\,\min} \\ \beta(1-\alpha)(\psi_k - \psi_{k\,\min}) - \alpha, & \psi_k \leq \psi_{k\,\min} \end{cases} \tag{6}$$

where $\alpha \in \{0, 1\}$ and $\beta \in \{-1, 1\}$ are *control parameters*, and influence the calculation of different factors based on the desired behavior. In general $f^{k*} \in [-1, 1]$. When $f^{k\gamma} \to 1$, the likelihood of asking for help is high, while when $f^{k\gamma} \to -1$ the likelihood of asking for help is low. Analogously, when $f^{k\delta} \to 1$, the likelihood of giving help is high, while when $f^{k\delta} \to -1$ the likelihood of giving help is low. Note that, f^{k*} does not necessarily dominate all the other factors. As a result, a_i might still decide to ask for or give help when the threshold is exceeded.

The main rationale of (5) and (6) is that the factors should be adjusted proportionally to the distance between the current measurement ψ_k and the threshold value $\psi_{k\,\min}$. In particular, when $\psi_k \geq \psi_{k\,\min}$, agent a_i does not need help to complete its task, so it will decrease the likelihood of asking for help, while, at the same time, it will increase its willingness to give help to other agents.

In order for any factor to dominate all the factors (f^{k*}) given certain conditions, the corresponding weights ϕ_i in (2), or the corresponding weights ζ_i in (4), should be set to 1. Since $\sum_{i=1}^{n} \phi_i = 1$ and $\sum_{i=1}^{n} \zeta_i = 1$, the weights for all other factors are set to zero. If two or more of the factors f^{k*} is equal to one, then the weight of one of them is set to one, whilst all others are set to zero. When no factor is equal to one, weights for all factors are all equal and calculated as $\phi_i = 1/n$ and $\zeta_i = 1/n$.

C. Factor description

In order to calculate the corrections u and v, nine factors ($n = 9$) have been identified in [3] as relevant in the process of deciding when to ask and give help during runtime. All the factors follow the update rules (5) and (6).

The **battery factor** f^{1*} is the amount by which agent a_i's battery level $\psi_1 = b \in [0, 1]$ will affect its willingness to interact. f^{1*} is computed with $\psi_{1\,\min} = b_{\min} + b_{\tau_j}$, where $b_{\min} \in [0, 1]$ is the minimum battery threshold for which a_i can operate in the *execute* state, $b_{\tau_j} \in [0, 1]$ is the amount of energy required by the task τ_j, $\alpha = 1$, and $\beta = 1$.

The **knowledge factor** f^{2*} is the amount by which the agent's confidence on its knowledge level $\psi_2 \in [0, 1]$ will affect its willingness to interact. f^{2*} is computed with $\psi_{2\,\min} = 0$, $\alpha = 1$, and $\beta = 1$.

The **skill/ability factor** f^{3*} is the amount by which the agent's skill efficiency level $\psi_3 \in [0, 1]$ in performing a task will affect its willingness to interact. f^{3*} is computed with $\psi_{3\,\min} = 0$, and $\alpha = 1$.

The **equipment factor** f^{4*} is the amount by which the agent's equipment accuracy level $\psi_4 \in [0, 1]$ for performing a task will affect its willingness to interact. f^{4*} is computed with $\psi_{4\,\min} = 0$, $\alpha = 1$, and $\beta = 1$.

The **resource factor** f^{5*} is the amount by which the agent's resource quality level $\psi_5 \in [0, 1]$, i.e., how much the type of tools in the agent's possession fit the task to be performed, will affect its willingness to interact. f^{5*} is computed with $\psi_{5\,\min} = 0$, $\alpha = 1$, and $\beta = 1$.

The **performance factor** f^{6*} is the amount by which the agent's current performance level $\psi_6 \in [0, 1]$ will affect its willingness to interact. Performance is a general indicator of how well an agent's outcome is with respect to the tasks attempted in the past. f^{6*} is computed with $\psi_{6\,\min} \in [0, 1)$, $\alpha = 0$, and $\beta = 1$.

$f^{7\gamma}$ is the **task progress factor**, covers the progress towards completion $\psi_7' \in [0, 1]$ of the current task, while $f^{7\delta}$ is the **task trade-off** factor, and it covers the trade-off $\psi_{7''} \in [0, 1]$ between a task τ_j currently in execution, and a new task proposed by another agent τ_j'. The two factors are computed as (5) and (6), with $\psi_{7\,\min} \in [0, 1)$, $\alpha = 0$, and $\beta = 1$.

The **environment factor** f^{8*} is the amount by which the agent's likelihood of succeeding in its environment $\psi_8 \in [0, 1]$ will affect its willingness to interact. This likelihood could be estimated based on the difficulties an agent perceives in its environment, e.g., obstacles, harsh conditions and so on. f^{8*} is computed with $\psi_{8\,\min} \in [0, 1)$, the acceptable level for this likelihood, $\alpha = 0$, and $\beta = 1$.

The **collaboration factor** f^{9*} is the amount by which the agent's estimated likelihood of a successful collaboration with another $\psi_9 \in [0, 1]$ will affect its willingness to interact. f^{9*} is computed with $\psi_{9\,\min} \in [0, 1)$, the acceptable level of such likelihood, $\alpha = 0$, and $\beta = -1$, through equation (6). It can be observed that this factor affects the willingness to give and ask for help in the same way.

In this paper, only factors $f^{1-5\gamma}$ are considered to be dominant, because it is assumed that in case either battery,

abilities, equipment, knowledge, or tools are not adequate then the agent should not attempt the task autonomously. Moreover, it is assumed that the agent is able to estimate the different factors.

IV. Simulation setup

The following paragraphs describe the implementation of the SAR scenario, and the implementation of the agent model. ROS [24] has been used as the underlying communication middleware. The agents and environment are implemented as ROS nodes (executables), and communicate with each other through the ROS middleware.

A. Scenario instantiation

The environment is a 2D space with a specific width $grid_w = 30$ and height $grid_h = 30$, and is generated at the beginning of a simulation. Agents can move in this space, from a point A to a point B, according to the shortest Euclidean path between two points. Obstacles and collisions are not considered because they are not crucial to the level of interactions discussed here. In this space, a fixed number of fires ($n_f = 40$) is generated at random locations (x, y), with the same intensities $I \in \{75, 100\}$ (the value of the intensities depends on the difficulty of the simulation). The intensities are kept fixed in time for simplicity. After a fire is extinguished, the trapped individuals (who lie at the same location as fires) become visible and can thereafter be saved. The number of these individuals at a particular location is the same for each location, $n_v \in \{15, 20\}$ (depending on the difficulty of the simulation). Moreover, two base stations, one for fire brigades and one for ambulances, are initiated at two random locations in the grid.

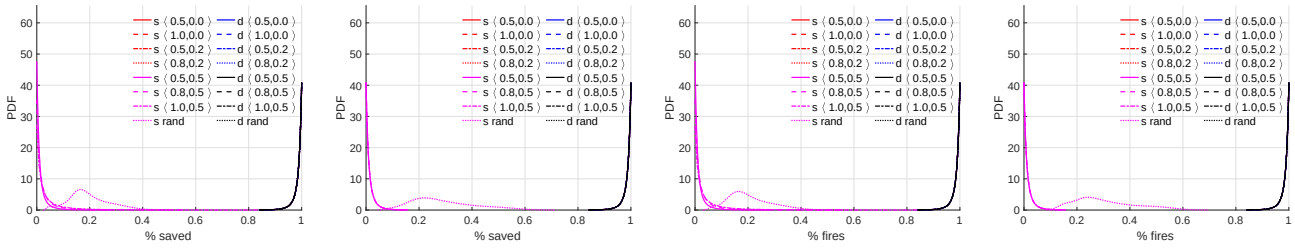
The environment node publishes continuously the current state of fire intensities, fire status (active/inactive), number of trapped individuals, and location in the grid. Fire extinguishing, and individual extraction is simulated by having the appropriate agent make a ROS service call to the environment node. As a result, the corresponding variable (fire intensity or number of trapped individuals) will decrease by a predefined $step = 1$.

B. Agent instantiation

Agents are implemented as ROS nodes. Their interactions with each other are realized through the ROS publish/subscribe mechanism (for broadcasting), and action server mechanism (for one to one calls). There are three types of agents: fire brigades, ambulances, and police. Fire brigade agents put out fire. Ambulances extract and carry individuals to safety. Police agents detect the fires/victims within their range, and broadcast this message to the others. They all broadcast their identity and abilities (e.g., fire extinguishing and transportation), as a result they are known to one another.

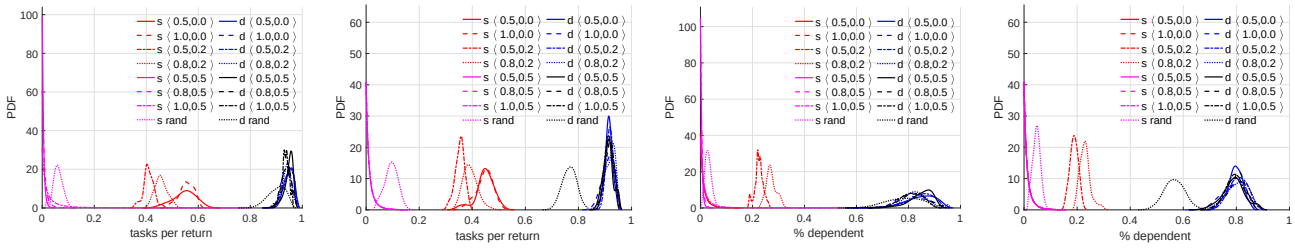
Throughout the whole the simulation, agents perform continuous Levy walks in the generated 2D space, when in the *idle* state. The Levy walk algorithm was implemented as it has been considered an efficient strategy for search algorithms independent of the target distribution [25]. During these walks, agents publish their location to the environment node, and are able to detect fires/individuals if they are within a specified Euclidean distance ($d_v = 10\% \times grid_w = 10\% \times grid_h = 3$). Trapped individuals are not visible as long as the status of a fire is active. Fire brigade agents can also be notified by police agents about the existence of a fire, as a result their information is not restricted to what is in their range of visibility at a particular time. If there are visible fires, a fire brigade agent will generate a corresponding task for extinguishing it. The task will have as many iterations as the fire intensity. If more than one fire location is visible, then one is chosen randomly and pursued. Similarly, if there are visible trapped individuals, ambulance agents will pick one randomly and generate a corresponding task with the number of iterations being equal to the number of individuals. When a task is generated, agents will walk to the location of interest. They are assumed not to need help for reaching any location in the 2D space. No initial plan, or explicit global coordination is assumed. Nevertheless, it might happen that several agents go for the same fire, or individuals to extract. When an agent decides to ask for help, it will ask the appropriate agents it knows one by one, until either a task succeeds, or the list of agents to ask is exhausted. Furthermore, an agent that is already waiting upon another for a task, will discard on the spot any help requests it receives. This is to ensure that agents do not wait on one another pointlessly.

Each agent starts at the same level of battery, at $\psi_1 = 1$, corresponding to the maximum available energy of each agent. During task execution, the agent's energy level is decreased with a certain level, in each iteration step. Also, when an agent moves to the location of interest, its energy level is reduced proportionally to the covered distance. There is a low threshold $b_{\min} = 0.3$ under which the agent will go to *out of order* and thereafter recharge. Agents are assumed to have the necessary knowledge for performing tasks, with knowledge $\psi_2 = 1$. The same assumptions hold for abilities ($\psi_3 = 1$), and equipment ($\psi_4 = 1$). Initially, agents have the necessary resources depending on their function, i.e., fire brigades have water $r_w = 25$ units of water, whereas ambulances have space for $r_s = 5$ individuals inside their vehicle. When the agent has enough resources to extinguish the fire or to carry an individual, the respective resource factor will be $\psi_5 = 1$, and it will be $\psi_5 = 0$ otherwise. During the run of each task, these resources decrease according to their usage. After that a task is completed, each agent will move to its corresponding base station and reset its own resources. Agents are assumed to be able of estimating: environmental risk, potential collaborator risk, their own performance, progress of their current task, and task trade-off between two tasks. In these simulations, the likelihood of success in the environment is kept constant for simplicity. The likelihood of success with a potential collaborator is estimated based on the the percentage of past successful interactions. Agents calculate their performance through $\psi_6 = {}^{t_c}/_{t_{\text{tot}}}$, where t_c is the number of completed tasks, and t_{tot} is the total number of attempted tasks. Task



(a) % of saved individuals for SC1 (b) % of saved individuals for SC2 (c) % of extinguished fires for SC1 (d) % of extinguished fires for SC2

Figure 2: PDFs of the results for the *% saved* and *% fires* metrics, in SC1–2.



(a) Task per return to base for SC1 (b) Task per return to base for SC2 (c) % dependent for SC1 (d) % dependent for SC2

Figure 3: PDFs of the results for the *task per return* and *% dependent* metrics, in SC1–2.

progress is calculated by $^{it_c}/_{it_{\text{tot}}}$, where it_c is number of iterations completed, and it_{tot} is the total number of iterations. Finally, the task trade-off for two tasks τ_1 and τ_2 is calculated through $^{R_2-R_1}/_{R_2+R_1}$, where R_1 and R_2 are the respective rewards. Finally, the thresholds for these factors are set to 0 for simplicity, thus $\psi_{2-8\,\text{min}} = 0$.

V. Results

The hypothesis, evaluated through simulations, is formulated as follows: "Agents with adaptive autonomy perform better than agents that do not display such behavior, in the context of a simulated search and rescue scenario". Performance is assessed across several metrics (Section V-A). There are two modes of execution for any of the simulation runs, (i) static mode, in which the components of the willingness to interact are set in the beginning, do not change during runtime, and are independent of factors, and (ii) dynamic mode, in which the components of the willingness to interact change during runtime (every update is done against the same initial value specified for each component) based on factors. For both modes, the same couples $\langle \delta_0, \gamma_0 \rangle$ for the components of the willingness to interact is used. In the evaluation, the following couples are considered: $\langle 0.5, 0.0 \rangle$, $\langle 1.0, 0.0 \rangle$, $\langle 0.5, 0.2 \rangle$, $\langle 0.8, 0.2 \rangle$, $\langle 0.5, 0.5 \rangle$, $\langle 0.8, 0.5 \rangle$, $\langle 1.0, 0.5 \rangle$, and a random configuration. In the first seven configurations, each agent in the population is initialized with the same willingness to interact. Whereas, in the random configuration, each agent is initialized with different random values for δ_0 and γ_0. Furthermore, simulations, for each configuration and both modes, are run across two scenarios which differ in difficulty (the random values for the willingness to interact are the same within a scenario, but differ across scenarios). Difficulty is defined as the ratio between the required resources on each site ($w \in \{75, 100\}$ and $s \in \{15, 20\}$) and the agent's initial resources ($r_w = 25$ or $r_s = 5$). All simulations were ran on a system with 24 cores, 8GB RAM, with Ubuntu 14.04 LTS operating system, and ROS Indigo[1].

A. Evaluation metrics

The quality of the obtained results is evaluated according to four metrics, as follows: (i) percentage of individuals saved in the simulation with respect to the total number of trapped individuals (*% saved*), (ii) percentage of fires extinguished in the simulation with respect to the total number of fires distributed in the 2D space (*% fires*), (iii) number of completed tasks per return to the base (*task per return*), (iv) percentage of dependent completed tasks with respect to all dependent tasks attempted (*% dependent*). A task is dependent when an agent needs assistance for its completion. The results are averaged over the whole population of agents and over the number of repetitions (30), for each simulation instance, i.e., for any combination of scenario difficulty and initial configuration of the willingness to interact.

B. Numerical results

Simulation results are displayed in Figures 2 and 3 as probability density functions (PDFs) obtained with 30 different runs of the considered scenarios. Each figure contains the outcomes with respect to one of the metrics, across the two difficulty scenarios (SC). Each sub-figure contains the outcomes for the eight initial configurations and the two modes of execution (the static and dynamic strategies). The red and magenta lines refer to the static case (indicated with 's' in the legend), while the blue and black lines refer to the dynamic strategy (indicated with 'd' in the legend).

Regarding the percentage of saved individuals, in both difficulty scenarios, all dynamic agents and static agents with $\gamma_0 \in \{0.0, 0.2\}$ manage to save all individuals within the

[1]The source code used in producing the results displayed in this paper is publicly available at https://github.com/gitting-around/gitagent-sar.git.

allotted time (Figures 2a-2b). Whereas, for static agents with $\gamma_0 = 0.5$ is nearly 0, and for the random configuration on average 20%. Similar considerations apply for the percentage of extinguished fires (Figures 2c-2d). This is to be expected, because if by the time the simulation ends, only half of the fires have been extinguished then only half of the individuals could have been saved.

Dynamic agents achieve higher percentage of completed tasks with respect to returns to the base, (Figures 3a-3b). In SC1–2, all the static agents perform worse than the dynamic counterparts. Nevertheless, among static agents, the ones with $\gamma_0 = 0.0$ achieve better results than others. Whereas in the dynamic mode, the random configuration seems to produce slightly worse results than the other configurations. The results are consistent across the two scenarios.

Dynamic agents perform noticeably better as compared to their static counterparts with respect to the percentage of completed dependent tasks (Figures 3c-3d). The outcomes are stable among different configurations and scenarios, with the dynamic random configuration always performing slightly worse than others (especially visible in SC2). In the static mode, better results are obtained for configurations where $\gamma_0 = 0.2$. These results agree with previous work [4], which shows that when agents need to depend on each other all the time, and are highly likely to give help, the performance of the system will degrade.

VI. Conclusion

The obtained results indicate that adaptive autonomous behavior can positively impact the performance of a population agents, in a context of a SAR scenario. Performance is assessed with respect to four metrics: percentage of individuals saved, percentage of fires extinguished, percentage of completed tasks with respect to the returns to base, percentage of completed dependent tasks. Among the two difficulty scenarios, the adaptive autonomous agents maintain a stable behavior, seemingly independent of the initial configuration. Whereas, static agents are quite dependent on the initial configuration. The results displayed in the paper can have a dependency on the platform used for the simulations. In more powerful platforms, all agents can perform better overall, and in less powerful ones, they are expected to have poorer performance overall, i.e., less individuals saved and extinguished fires within the allotted time for the simulation.

There are several directions for future work. Firstly, an agent reasons on each iteration whether to ask for help. This is particularly penalizing in the static case, where even with low willingness to ask for help on each iteration, most tasks end up being dependent tasks. Thus, the possibility that the agent decides on whether to ask for help once in a couple of iterations, needs to be investigated. Secondly, an agent can accept to help another agent, even if it does not fulfill energy, abilities, equipment, knowledge, or resource requirements. Consequently, it will ask for help a third agent. It is needed to compare this design choice, with the other option that involves the agent declining upfront to help in such scenario. Finally, cues regarding the grade of dependencies within the population can be used as an additional factor, allowing the agent to regulate its own behavior from the global perspective as well.

References

[1] K. S. Barber, A. Goel, and C. E. Martin, "Dynamic adaptive autonomy in multi-agent systems," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 12, no. 2, pp. 129–147, 2000.

[2] C. Castelfranchi, "Founding agent's 'autonomy' on dependence theory," in *ECAI*, pp. 353–357, 2000.

[3] M. Frasheri, B. Çürüklü, and M. Ekström, "Analysis of perceived helpfulness in adaptive autonomous agent populations," *LNCS Trans. on Computational Collective Intelligence*, 2017.

[4] M. Frasheri, B. Çürüklü, and M. Ekström, "Comparison between static and dynamic willingness to interact in adaptive autonomous agents," in *ICAART*, 2018.

[5] S. Tadokoro *et al.*, "The robocup-rescue project: A robotic approach to the disaster mitigation problem," in *ICRA*, vol. 4, pp. 4089–4094, 2000.

[6] R. Sheh, S. Schwertfeger, and A. Visser, "16 years of robocup rescue," *KI-Künstliche Intelligenz*, vol. 30, no. 3-4, pp. 267–277, 2016.

[7] A. Visser, F. Amigoni, and M. Shimizu, "The future of robot rescue simulation workshop," *Benelux AI Newsletter*, vol. 30, no. 2, 2016.

[8] A. Kleiner *et al.*, "Rmasbench: benchmarking dynamic multi-agent coordination in urban search and rescue," in *AAMAS*, pp. 1195–1196, 2013.

[9] J. Parker *et al.*, "Exploiting spatial locality and heterogeneity of agents for search and rescue teamwork," *Journal of Field Robotics*, vol. 33, no. 7, pp. 877–900, 2016.

[10] D. Di Paola *et al.*, "Optimal control of time instants for task replanning in robotic networks," in *ACC*, pp. 1993–1998, 2016.

[11] B. Hardin and M. A. Goodrich, "On using mixed-initiative control: A perspective for managing large-scale robotic teams," in *HRI*, pp. 165–172, 2009.

[12] L. Lin and M. A. Goodrich, "Sliding autonomy for UAV path-planning: Adding new dimensions to autonomy management," in *AAMAS*, pp. 1615–1624, 2015.

[13] B. van der Vecht, F. Dignum, and J. C. Meyer, "Autonomy and coordination: Controlling external influences on decision making," in *WI-IAT*, vol. 2, pp. 92–95, 2009.

[14] F. Golpayegani *et al.*, "Multi-agent collaboration for conflict management in residential demand response," *Comp. Comm.*, vol. 96, pp. 63–72, 2016.

[15] H. van Dyke Parunak *et al.*, "A design taxonomy of multi-agent interactions," in *AOSE*, pp. 123–137, 2003.

[16] A. Artikis *et al.*, "Specifying and executing open multi-agent systems," in *Social Coordination Frameworks for Social Technical Systems*, pp. 197–212, Springer, 2016.

[17] F. Zambonelli *et al.*, "Developing pervasive multi-agent systems with nature-inspired coordination," *Pervasive and Mobile Computing*, vol. 17, pp. 236–252, 2015.

[18] S. Mariani and O. Andrea, "State-of-the-art and trends in nature-inspired coordination models," *The IEEE Intelligent Informatics Bulletin*, vol. 18, no. 2, pp. 35–40, 2017.

[19] M. Puviani, G. Cabri, and F. Zambonelli, "Patterns for self-adaptive systems: agent-based simulations," *EAI Endorsed Trans. on Self-Adaptive Systems*, vol. 1, pp. 1–15, 2015.

[20] J. M. Bradshaw *et al.*, "Kaa: policy-based explorations of a richer model for adjustable autonomy," in *AAMAS*, pp. 214–221, 2005.

[21] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *The knowledge engineering review*, vol. 10, no. 2, pp. 115–152, 1995.

[22] M. Johnson *et al.*, "Coactive design: Designing support for interdependence in joint activity," *Journal of Human-Robot Interaction, 3 (1)*, 2014.

[23] S. Kraus, "Negotiation and cooperation in multi-agent environments," *Artificial Intelligence*, vol. 94, no. 1, pp. 79 – 97, 1997. Economic Principles of Multi-Agent Systems.

[24] M. Quigley *et al.*, "ROS: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, 2009.

[25] M. Wosniack *et al.*, "Robustness of optimal random searches in fragmented environments," *Physical Review E*, vol. 91, no. 5, 2015.