Failure is Not an Option: Policy Learning for Adaptive Recovery in Space Operations

Steve McGuire, P. Michael Furlong, Christoffer Heckman, Simon Julier, Daniel Szafir, and Nisar Ahmed

***Abstract*—This letter considers the problem of how robots in long-term space operations can learn to choose appropriate sources of assistance to recover from failures. Current assistant selection methods for failure handling are based on manually specified static lookup tables or policies, which are not responsive to dynamic environments or uncertainty in human performance. We describe a novel and highly flexible learning-based assistant selection framework that uses contextual multiarm bandit algorithms. The contextual bandits exploit information from observed environment and assistant performance variables to efficiently learn selection policies under a wide set of uncertain operating conditions and unknown/dynamically constrained assistant capabilities. Proof of concept simulations of long-term human–robot interactions for space exploration are used to compare the performance of the contextual bandit against other state-of-the-art assistant selection approaches. The contextual bandit outperforms conventional static policies and noncontextual learning approaches, and also demonstrates favorable robustness and scaling properties.**

***Index Terms*—Learning and adaptive systems, human factors and human-in-the-loop, space robotics and automation.**

I. Introduction

ROBOTIC systems tasked with procedurally handling complex tasks are not currently capable of operating indefinitely without external interaction [1], [2]. This is a particularly important consideration for planetary exploration missions utilizing both humans and robots. Such missions rely on complex interplays between different actors, with many possible faults that might occur during autonomous operation. Robots might be required to interact directly with humans in the planetary environment, in an on-site habitat, and back on Earth. Current robotic systems are unable to operate in this environment without aid, and will likely remain powerless to do so for the foreseeable future. It is thus natural to consider how autonomous robots can leverage external assistance to recover from failures and resume operations.

Manuscript received September 10, 2017; accepted January 15, 2018. Date of publication February 2, 2018; date of current version February 27, 2018. This letter was recommended for publication by Associate Editor Dr. M. Howard and Editor D. Lee upon evaluation of the reviewers' comments. This work was supported by a NASA Space Technology Research Fellowship. *(Corresponding author: Steve McGuire.)*

S. McGuire and N. Ahmed are with the Department of Aerospace Engineering Sciences, University of Colorado at Boulder, Boulder, CO 80309 USA (e-mail: stephen.mcguire@colorado.edu; nisar.ahmed@colorado.edu).

P. M. Furlong is with SGT, Inc., NASA Ames Research Center, Mountain View, CA 94043 USA (e-mail: furlong@cmu.edu).

C. Heckman and D. Szafir are with the Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309 USA (e-mail: christoffer.heckman@colorado.edu; daniel.szafir@colorado.edu).

S. Julier is with the Department of Computer Science, University College London, London WC1E 6BT, U.K. (e-mail: s.julier@ucl.ac.uk).

Digital Object Identifier 10.1109/LRA.2018.2801468

This letter introduces a novel learning-based framework that enables a team (robots, humans, software agents) to cooperatively recover from a failure of autonomy by learning optimal assistance task allocations under dynamic and uncertain agent performance and environmental conditions that are difficult to comprehensively model a priori. The new optimal assistance allocation framework leverages contextual multi-armed bandit reinforcement learning algorithms. The key idea behind our approach is to exploit empirical information from observed environment and assistant performance variables in the form of *context features*, in order to efficiently learn optimal assistant selection policies under a wide set of uncertain operating conditions and unknown/dynamically constrained assistant capabilities. Unlike teleoperation strategies or designated supervision strategies that are typically used for highly choreographed and tightly coupled space operations, our approach allows the full set of *actors* (i.e. agents who can assist an autonomous agent) to include any robot in the operating environment, proximal software agents and humans, as well as distant software agents and humans back on Earth – thus providing potentially great flexibility to future mission designs. As shown in proof of concept numerical simulations of dynamic space mission failure recovery scenarios, the adaptive nature of our contextual bandit framework allows it to significantly outperform conventional static policies and non-contextual policy learning approaches for assistant selection. Our results also show that contextual bandits demonstrate robustness to other practical uncertainties such as unknown state transition probabilities and imperfect context feature vector specifications while scaling favorably to large problem spaces.

Section II of this letter presents background and related work, and describes the formal optimal assistant allocation problem. Section III describes our novel contextual multi-armed bandit framework to solve this problem, which to our knowledge has not been applied previously to autonomous failure recovery. Section IV describes the simulated application scenario used to evaluate our approach against other state of the art assistant selection techniques, and Section V presents and discusses the numerical results. Section VI presents conclusions and outlook for future work.

II. Background and Problem Statement

This work is motivated by a hypothetical Mars mission scenario, derived from existing Earth studies [3] and proposed future manned exploration missions [4]. The crew has been given a set of high-level *goals* which define the mission. Each goal is comprised of a set of *tasks* that must be completed. For example,

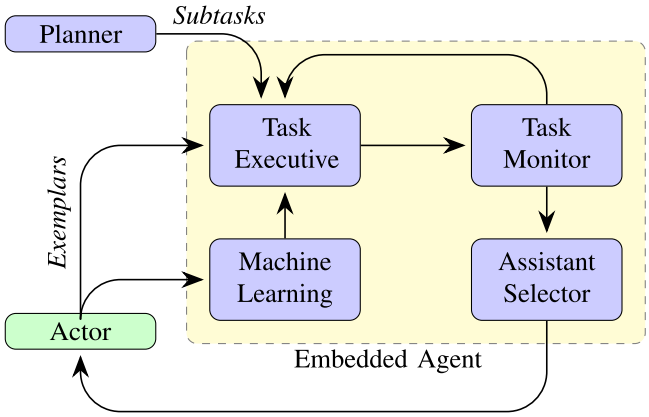

Fig. 1. Information flow in an idealized learning robot.

Subtask Queue	
Subtask	**Status**
Move to location	Completed
Drill sample	Completed
Move to location	*Failed*
Grasp container	Queued

Available Helpers			
Attributes	**Actors**		
Actor ID	**A**	**B**	**C**
Location	EVA	Habitat	Earth
Past Perf (1-10)	4	7	9
Stress Level (1-10)	7	2	6
...	...	...	...
Est Performance	*Medium*	*High*	*Low*

Fig. 2. Robot obtains assistance with the *move to location* subtask.

suppose a goal is to characterize the geology of a particular area [4]. Tasks include: conducting a pre-outing inspection; traveling to the area of interest; selecting particular rocks of interest; extracting samples; analyzing the samples; and interpreting the analysis to make inferences about greater geology of Mars. Each task is composed of a series of *subtasks*, atomic units of work which cannot be subdivided further [5]. For example, the subtasks for a robot traveling to an area of interest include: route planning; obstacle identification; obstacle avoidance; and localization.

Ideally, a robot should be fully capable of completing every subtask on its own to meet all the mission goals. However, this ideal is generally unachievable: the robot will eventually encounter unknown or poorly characterized phenomena that impacts its ability to complete certain subtasks. For instance, while executing a navigation subtask, a robot might experience unusual slippage which means it cannot reach its destination. This inability to complete a subtask to specification is called a *failure*. In the event of failure, the onboard agent must select an appropriate aid source while considering the mission impact of its choice. Once aid has been rendered, the robot resumes autonomous operations. A key constraint on this process is that human aid sources will have other primary responsibilities besides monitoring robotic operations.

Fig. 1 shows the architecture of an idealized learning robot. High-level goals are decomposed into subtasks by the *planner* to be completed by the *executive*. A *task monitor* observes the progress that the executive is making towards completion of each task [6]–[8]. Should the task either fail completely or be predicted to not meet requirements, an *assistant selector* is consulted to determine which *assistant* to consult. Assistants may be consulted for two reasons: task recovery and observing exemplars from which to learn better quality solutions in the future. In either case, the *best* assistant should be consulted; the definition of best is a *reward* function of mission objectives and resources.

The current practice in space robotic operations is to designate a single human operator to monitor and assist robots all the time using tightly choreographed operations composed offline and orchestrated online by mission operators and engineers [9], [10]. These designations can be determined through a combination of expert opinion and design choices, e.g. as in the static allocation tables of [11].

This static approach limits multi-robot/multi-tasking operations due to the issue of fan-out, where the human's attention becomes divided between multiple competing robots or external tasks [12]. Dedicated supervision also represents an inefficient use of crew time (one of the most valuable resources in a manned space expedition), while tight operations choreography incurs yet another significant mission cost in terms of ground-based contingency planning and system troubleshooting. In contrast to [13], where humans and robots are working together in physical proximity to accomplish a tightly-coupled task, humans in our scenario have other primary responsibilities besides monitoring robotic operations. While precise tasking and movements of possible human aids may not be possible, much relevant information will generally be readily available to robots at a coarse level in envisioned future mission systems; e.g. in our scenario, robots can query whether a human is inside the habitat or outside in a spacesuit. It is thus desirable for assistant allocation mechanisms to automatically exploit any available information and dynamically adapt to specific robotic fault recovery needs in response to evolving human capabilities and constraints (Fig. 2, e.g. individual levels of skill, experience, cognitive/physical workload, location constraints, costs for disrupting human activities, etc.)

A. Related Work

Dynamic assistant selection can be framed as an optimal allocation problem based on collective utility maximization for a single robot over the set of actors (where an actor's contribution to system utility is defined through a set of local utility functions based on each actor's state and assistance outcomes). Classical centralized [14], [15] and distributed [16] task allocation schemes require coherent utility functions to model the overall expected payoff of actor-task assignments. Importantly, most allocation schemes assume that the underlying task execution process and associated utilities are well-modeled and time-invariant, i.e. they are brittle and cannot respond to the system's actual performance [17].

Optimal assignment problems under uncertainty have also been addressed through partially observable Markov decision

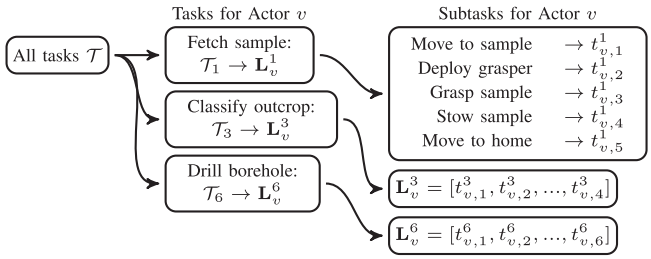

Fig. 3. Example of task decomposition.

processes (POMDPs), which are typically solved offline to determine an optimal online actor assignment policy. In alternative formulations, POMDPs have been used to optimize spatio-temporal assignments of robots to tasks [18] and other human-robot collaboration efforts [19]. POMDPs require accurate system models to evaluate both actions and rewards; however, such models are often unavailable or challenging to develop for long-term space missions. Optimal allocation policies are also extremely difficult to find for POMDPs with high-dimensional state spaces [20].

Reinforcement learning (RL) can be used to enable integration of feedback from action choices in order to improve future choices. Parker, et al's behavior-based L-ALLIANCE [21] is an early example of an RL-based adaptive task allocation system. A key limitation of that work is that it considers time to complete objectives as the only performance criteria, when in fact it is only one of many factors that influence overall task and mission performance. Nevertheless, the balance between "exploration and exploitation" inherent to RL allows performance data from assigned actors to be gathered and leveraged over time, without requiring accurate *a priori* knowledge of system models (thus making it attractive to consider for space missions). Since failure recovery mechanisms currently do not take advantage of either state-aware assistance or learning from past requests for assistance, the novel contribution of our work is to exploit both of these techniques within an adaptive context-based RL framework to improve and inform future assistance decisions.

B. Formal Assistance Allocation Problem Statement

Assume there exists a set $\mathcal{A} = \{a_v\}_{v=1}^{N_A}$ of actors capable of assisting an onboard agent during an autonomy failure; each actor $a_v \in \{\text{onboard agent, offboard agent, human}\}$. Each onboard agent maintains a work queue of N_T tasks: $\mathcal{T} = \{\mathcal{T}_s\}_{s=1}^{N_T}$. Every task $\mathcal{T}_s$ assigned to a single agent a_v is implemented by an ordered list $\mathbf{L}_v^s$ of atomic work primitives, called *subtasks*, such that $\mathbf{L}_v^s = [t_{v,1}^s, t_{v,2}^s, \ldots, t_{v,N_{sv}}^s]$. $\mathbf{L}_v^s$ is a complete list of N_{sv} subtasks that implements every task $\mathcal{T}_s$ assigned to onboard agent a_v. Fig. 3 shows an example task decomposition for a single actor with three assigned tasks.

Several simplifying assumptions are introduced, along with restrictions on the nature of the subtasks and assignments. First, utility is earned by accumulating reward over a series of actor assignments in response to subtask failures. In considering actor assignments, any subtask may only be assigned to a single actor for execution (i.e no teams of opportunity or multi-robot teams). Since the onboard autonomy is assumed to have made a "best effort" attempt at completing sub-tasks, and since robot computing capabilities are assumed to be similar, this work does not consider situations where robots assist one another. The subtask decomposition of higher-level goals is provided externally, e.g. by a dynamic task decomposition algorithms [22] or human intervention [6]. The subtask capabilities of the robot (t_v^s) are fixed at deployment time. While any subtask in the decomposition may fail, failure likelihoods are unknown *a priori*. Finally, assistance outcomes are only obtained for those actors assigned to assist the robot (i.e. no oracles are available for comparing alternatives).

Problem: Onboard agent a_u must assign a single actor from set $\bar{\mathcal{A}} = \mathcal{A} \setminus \{a_u\}$ of size $N_{\bar{A}}$ to recover from failed subtask t_j and maximize mission utility U. That is, given an $N_{\bar{A}}$ by 1 vector of rewards $r_{v,j}^{k,s}$ corresponding to actor a_v executing subtask $t_{v,j}^s$ at timestep k, discover a set of $N_{\bar{A}}$ indicator values $\gamma \in \{0, 1\}$ such that

$$U = \sum_{k=1}^{\infty} \sum_{v=1}^{N_{\bar{A}}} \sum_{s}^{N_T} \sum_{j=1}^{N_{sv}} \gamma_v^{k,s} r_{v,j}^{k,s} \tag{1}$$

is maximized subject to the constraint that one and only one $\gamma_v^{k,s} = 1$ per k (only one actor may assist at a given time). We refer to the process by which actors are assigned to subtask requests as a *policy*.

III. Multi-Arm Bandit Assistance Allocation

The optimal selection of an appropriate actor on the basis of dynamically changing information can be viewed as an instance of a *contextual multi-arm bandit problem* [23]. Multi-arm bandits (MABs) constitute a widely applicable class of reinforcement learning problems, and contextual MABs in particular are distinguished by the availability and exploitation of dynamically varying 'contextual information' that can be used to further inform learned policies.

In conventional context-free MABs [24], all learning occurs by examining the observed empirical reward earned by selecting a particular actor. However, in the contextual case, additional observations (called *context vectors*) of each actor can be made *before* selecting a particular actor. These observations provide insight into the otherwise unobservable dependencies that may influence an actor's performance.

The contextual information is provided by two context vectors: a shared context vector $\mathbf{z}^k$ and an actor-specific context vector $\mathbf{x}_v^k$. Subtask and environment parameters (such as subtask difficulty and ambient weather conditions) common to all actors are stacked together in $\mathbf{z}^k$, while actor-specific context parameters (e.g. current location and duty day information) are maintained separately in $\mathbf{x}_v^k$ for each actor v.

A. Linear Multi-Arm Bandits

A contextual selection policy uses the context vector $\mathbf{x}_v^k$ to select the actor v^* to maximize the expected reward r at each time step k:

$$v^* = \underset{v}{\operatorname{argmax}}\, \mathbf{E}\left[r_v^k | \mathbf{x}_v^k\right] \tag{2}$$

In the standard linear multi-arm bandit problem [25], the expected actor reward estimates are linear in the observed context features $\mathbf{x}_v^k$ and estimated actor-specific parameter vector θ_v, such that

$$\mathbf{E}\left[r_v^k|\mathbf{x}_v^k\right] = \left(\mathbf{x}_v^k\right)^T \theta_v. \tag{3}$$

This standard formulation condition strictly assumes that no contextual properties are shared between actors, and hence that the parameters θ_v are disjoint between actors. We maintain this assumption for now to introduce the underlying theory behind our allocation technique (before relaxing this later). To estimate θ_v in this case, a *design matrix* $\mathbf{D}_v$ is formed. This consists of a stacked set of past observed context vectors, as well as vector $\mathbf{b}_v$ that stacks the past rewards:

$$\mathbf{D}_v^k = \left[\,\mathbf{x}_v^1, \mathbf{x}_v^3, \ldots, \mathbf{x}_v^w\,\right]^T, \mathbf{b}_v^k = \left[\,\mathbf{b}_v^1, \mathbf{b}_v^3, \ldots, \mathbf{b}_v^w\,\right]^T \tag{4}$$

The design matrix, the parameter vector, and the observed rewards form a linear system of the form $\mathbf{Ax} = \mathbf{b}$:

$$\mathbf{D}_v^k \theta_v^k = \mathbf{b}_v^k \tag{5}$$

A maximum likelihood estimate of θ_v can then be computed at time k via regularized least squares:

$$\theta_v = \left(\left(\mathbf{D}_v^k\right)^T \mathbf{D}_v^k + \mathbf{I}_d\right)^{-1} \left(\mathbf{D}_v^k\right)^T \mathbf{b}_v^k \tag{6}$$

where $\mathbf{I}_d$ is the identity matrix of dimension d corresponding to the number of observed context vectors and rewards.

This result can now be generalized to the full assistance allocation problem, where contextual information is *not* solely disjoint and independent between actors, since properties such as the subtask difficulty will be common across all actors. To accommodate common observations, the underlying linear model is augmented with the set of shared observations $\mathbf{z}_k$ at time k and estimated environmental parameters β to create a *hybrid linear multi-arm bandit* of the form:

$$\mathbf{E}\left[r_v^k \mid \mathbf{x}_v^k, \mathbf{z}^k\right] = \left(\mathbf{z}^k\right)^T \beta + \left(\mathbf{x}_v^k\right)^T \theta_v \tag{7}$$

Maximum likelihood estimates β and θ_v can be obtained similarly as in (6). Once all context vectors $\{\mathbf{z}^k, \mathbf{x}_1^k, \ldots, \mathbf{x}_{N_A}^k\}$ are observed, the implementation of the contextual MAB follows [25, Algorithm 2]. In particular, we re-estimate β and θ_v after each time step.

The novel contribution of our work is the application and evaluation of the hybrid linear contextual multi-arm bandit to the assistant allocation problem, where each subtask type has its own bandit. To our knowledge, this method has not yet been applied to failure recovery in autonomous systems.

IV. Example Simulated Mission Application

We developed a simulation based on planetary exploration analogue missions [26], [27] to validate and demonstrate the implementation and necessity of our adaptive assistant selection framework. Our simulation models aspects of the exploration environment as well as two categories of human actors: on-planet explorers and ground-support controllers. To validate the necessity of an adaptive context-based learning approach, we compare the performance of five policies for assistant selection. We briefly describe the dynamics of our simulation:

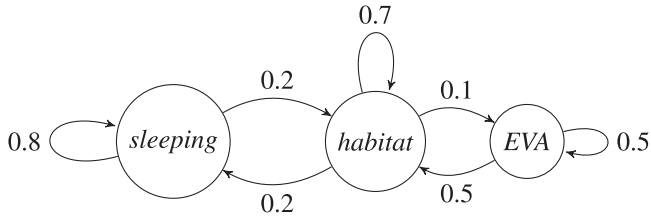

Fig. 4. Local human state transition model.

1) Subtask Types: In the current experiment, $C = \{navigate, manipulate, handle\_sample\}$. We also define the mapping $\mathcal{C}(t_j) \rightarrow C$ to denote the 'parent' category task of a given subtask t_j.

2) State Variables: Our simulation includes two types of dynamics as functions of simulator ticks t: environmental and human. Two environmental state variables (e_s, e_t) are modeled: season and light level. Season simulates the changing length of a day in a rotating planetary body, defined as:

$$e_s(t) = \frac{2\pi t}{t_{th} t_{hd} t_{dy}} \mod 2\pi \tag{8}$$

where t_{th}, t_{hd}, and t_{dy} are time constants encoding ticks per hour, hours per day, and days per year respectively. Season is continuous over the interval [−1,1] with a complete cycle taking one year. Light level simulates the rising and setting of the Sun and corresponding changes in lighting in the environment. Light level is continuous over the interval [0,1] with a cycle taking one day, computed as:

$$e_l(t) = \max\left(\frac{\sin\frac{2\pi t}{t_{th}\, t_{hd}} - \sin e_s}{2}, 0\right) \tag{9}$$

The complete environment state vector at time k is thus $\mathbf{z}^k = [e_s, e_l]$.

Six human state variables are modeled: location, time in location, time awake, time since last food, stress level, and subtask proficiency [28]; a more complete human state vector would provide in-depth observations of physiological, cognitive, and psychological parameters of human crewmembers. We use a simplified human model here as a means of demonstrating the usefulness of our work.

Location (h_l) is defined separately for local humans (i.e. planetside humans) and remote humans (i.e. ground controllers). For local humans, three states are defined: *sleeping*, *habitat*, and *EVA*. To transition between the states, a probability table is defined in Fig. 4. In the case of remote ground controllers, location changes are dictated by their on- and off-duty time. Ground controllers go off-duty after 8 hours of on-duty time and return for work 12 hours later.

Time in location (h_{tl}) is defined by the length of time that an actor has not transitioned to a new location, in hours. Time awake (h_{ta}) is the defined as the length of time that an actor has not been in the *sleeping* state, in hours. Time since last food (h_{tf}) measures the amount of time that an actor has gone without food. Food events are drawn from $\exp(\lambda_f)$ to obtain the next eating time. In the current simulation, $\lambda_f = 3$ hours. Human actors do not eat when they are asleep. Stress level (h_s)

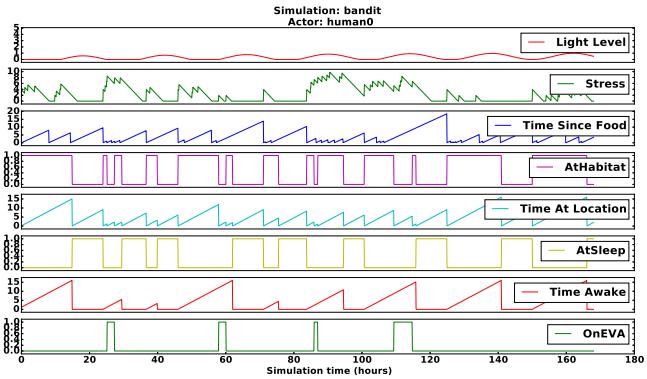

Fig. 5. Representative simulation state evolution for a single actor.

measures the duress of an actor, where location transitions and a draw from $\exp(\lambda_s)$ add to stress, while stress is otherwise reduced by a constant every time step. In the current simulation, $\lambda_s = 4$ hours, adding 1 unit of stress per event, while reducing stress by 0.1 unit per tick. Proficiency ($\mathbf{h}_p$) models an actor's improvement over time as the same subtask is executed repeatedly. Over time, the effect is increased productivity (assuming that the environment and internal actor state is otherwise identical). This vector quantity with an entry per subtask type is initialized to 1.0 and increases at a rate of 0.1 per subtask assignment instance.

For each actor v at timestep k, their complete state vector is $\mathbf{x}_v^k = [h_{ta}, h_{tc}, h_{tf}, h_s]$. Evolution of each of these variables is shown in Fig. 5. Actor location h_l and subtask proficiency $\mathbf{h}_p$ are accounted for separately in the costs and benefits.

3) Benefits: Benefit is the progress towards mission goals that an actor's assignment to a subtask yields. For example, the benefit obtained in a *navigation* subtask might be the error in the final position compared to the desired position. In the simulation, the benefit is obtained as a dot product of the complete state vector for an actor v, $[\mathbf{z}^k\ \mathbf{x}_v^k]$, executing subtask t_j with a weight $\mathbf{w}_v^j$ for that particular subtask.

In initialization, a Beta prior is placed on each human's performance for each subtask type, where the hyperparameters α (indicating a failure count) and β (indicating a success count) are drawn from $\mathcal{U}[1, 10]$ for local humans and $\mathcal{U}[5, 10]$ for the remote humans. These parameter values are driven by an assumption that ground controllers are able to dedicate their time completely to performing a task and thus achieve higher performance. The weight vector $\mathbf{w}_v$ is then drawn from Beta$(\alpha_{v,j}, \beta_{v,j})$ for each subtask type j. The final benefit $b_{v,j}^{ks} = ([\mathbf{z}^k\ \mathbf{x}_v^k]^T \cdot \mathbf{w}_v)h_{j,p,v}$, where $h_{j,p,v}$ is the proficiency term from $\mathbf{h}_p$ corresponding to subtask type j.

4) Costs: We define costs as the resources which are expended when an actor is assigned to a subtask. In the real world, these costs could be objective measures such as consumables, bandwidth, or time, or subjective measures such as mission impact. For software agent actors, costs could include CPU and bandwidth usage. In our simulation, costs are a fixed value per subtask type $c \in C$, drawn from $\mathcal{U}[1, 10]$. Additionally, we discourage any subtask allocation request that awakens a sleeping human by adding a penalty term of 100 units based on the current value of h_l. To model the relative difficulty in interacting with computer controls in a spacesuit, any subtask assigned to an actor outside the habitat is also penalized 10 units. These costs are consistent with usability impact analysis for space system design [28].

5) Productivity as a Reward Measure: Since subtask assignment to individual actors is expected to yield different benefits and costs for each actor, we wish to select the highest-performing actor per unit cost. We thus introduce the idea of *productivity* [29] as our reward measure. Reward $r_{v,j}^{k,s}$ obtained by actor v at timestep k and subtask j implementing task s is defined as:

$$r_{v,j}^{k,s} = b_{v,j}^{k,s} \,/\, c_{v,j}^{k,s} \tag{10}$$

that is, the benefits of the assignment divided by the cost of that assignment. This measure allows a policy to recognize that choosing the cheapest or most beneficial assistance may not yield the maximum progress towards higher-level goals. Note that this measure only allocates subtasks; allocations at the task level are outside the scope of this work.

6) Policies: In this work, five policies for assistant allocation are considered: *random*, *uninformed static*, *linear multi-arm bandit*, *informed static*, and *KLempUCB*. In the *random* policy, an actor is assigned to a subtask uniformly over the set of all actors. No state information is used to inform the selection. In the *linear multi-arm bandit* policy, the current state vector composed of world state and actor state is prepared for each actor v at timestep k as $[\mathbf{z}^k\ \mathbf{x}_v^k]$. The linear multi-arm bandit algorithm is then consulted to estimate the expected reward of each actor v, with the actor estimated to return the greatest reward selected for execution. The multi-arm bandit is then updated with the actual observed reward earned by the selected actor.

In the *uninformed static* policy, the Beta priors are used to estimate overall performance of an actor given a subtask class. This policy is representative of the static analysis proposed in [11] as the state of the art. For each subtask type c in subtask class set C, the ratio of every actor's hyperparameters (α_c, β_c) is compared. The actor with the lowest ratio, i.e. highest number of successes vs. number of failures, is designated as the static actor for that subtask type.

In the *informed static* policy, subtask assignments are analyzed post-hoc to determine what an ideal static policy should have been using the ground truth data. To prepare an ideal static policy, a table is prepared to accumulate a count for the highest-earning subtask/actor pairing for each assignment event. After processing all assignment events, the actor with the maximum count for each subtask is then selected as the designated assistant. The simulation is then re-executed with the same initial conditions and subtask history to evaluate this policy's performance. While this policy is not physically realizable, the intent is to derive a high-performance static policy as a comparison benchmark.

In the Kullback-Leibler Empirical Upper Confidence Bound (*KLempUCB* [30]) policy, an empirical model of observations is used to predict future performance. This algorithm was chosen to represent an alternative multi-arm bandit policy compatible with our problem.

7) Regret as a Metric: The performance of each policy is described by *regret* [24]. Regret expresses the difference

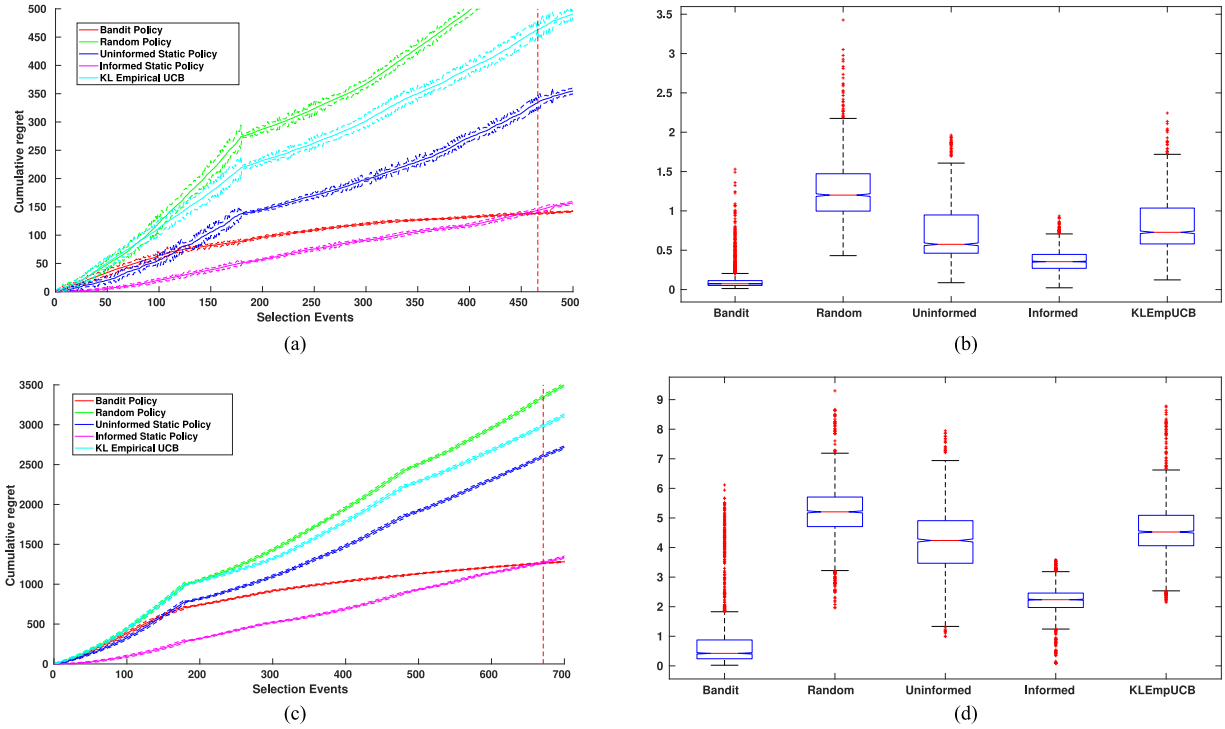

Fig. 6. Comparison of two sets of actor populations, each with three subtasks. In each condition, post-hoc comparisons using Tukey's HSD test revealed that the multi-arm bandit was significantly better in terms of minimizing regret than every other policy ($p < 0.001$). (a) Two local and two remote actors: dashed red line indicates significant difference after 465 selection events. $\pm 5\sigma$ bounds are in dashed lines. (b) Two local and two remote actors: the multi-arm bandit policy has a significant effect on regret. (c) Ten local and ten remote actors: dashed red line indicates significant difference after 670 selection events. $\pm 5\sigma$ bounds are in dashed lines. (d) Ten local and ten remote actors: the multi-arm bandit policy has a significant effect on regret.

between the reward earned by the policy under study and the best possible reward that could have been obtained by an omniscient oracle. Since decisions of a policy affect future conditions based on the past history of choices, we use one oracle per policy to compute regret. For example, in order to assess n policies, n oracles would be required (one per policy) even though all policies may be evaluating the same subtask sequence. This oracle is needed to evaluate the quality of our results and is not a structural requirement. We will use the metric of *cumulative regret* in the next section to analyze the performance of five allocation policies.

V. Simulation Results and Discussion

To demonstrate the effectiveness of our adaptive approach to assistant selection, we conducted a series of Monte Carlo simulations exercising each of the policies in Section IV. Our experiment investigates whether the multi-arm bandit selection strategy has a significantly better regret performance than alternative policies. We present an in-depth analysis of a nominal mission profile with a fixed number of actors and subtasks. We also assess the scalability of our algorithm by conducting a sensitivity analysis with increasing numbers of actors and subtasks. To provide a more direct comparison to other learning methods, we present both simulated time and the number of assignment events within each run. The number of assignment events corresponds to the size of the training set from which the multi-arm bandit draws its conclusions.

A. Baseline Case: 3 Subtasks

We use a nominal baseline scenario with three subtasks and between 2–10 local and remote humans. Each scenario was run 500 times, with a simulated assignment event occuring every five minutes. In each simulation set, the set of Beta parameters (α, β) are drawn only once to obtain weights for each subtask class, which are then held constant between runs. For each simulation, a one-way analysis of variance (ANOVA) test on the mean regret per time step for each policy was conducted to determine if the use of multi-arm bandit yielded an improvement.

Fig. 6 shows simulations for 2 local and 2 remote humans (a) and 10 local and 10 remote humans (c). In the 2 local and 2 remote case, we found a significant main effect of policy on regret, $\mathrm{F}(4, 10074) = 4588.82$, $p < 0.001$, $\eta^2 = 0.65$. In the 10 local and 10 remote case, we found a significant main effect of policy on regret, $\mathrm{F}(4,10074) = 7779.33$, $p < 0.001$, $\eta^2 = 0.76$. In each condition, post-hoc comparisons using Tukey's HSD test revealed that the multi-arm bandit was significantly better in terms of minimizing regret than every other policy $(P < 0.001)$; ANOVA analyses are shown in Fig. 6(b) & (d).

Fig. 7 examines a single run in detail. At selection event 100, the adaptive policy assigns a sample handling task (subtask 2) to a sleeping actor. This poor choice creates a large jump in the cumulative regret plot. Later at event 1540, the adaptive policy once again assigns a sample handling task (subtask 2) to the same actor who is not asleep, making an optimal choice. The bandit has thus sensibly adapted to the system dynamics, and updated θ_k^v in (7) accordingly.

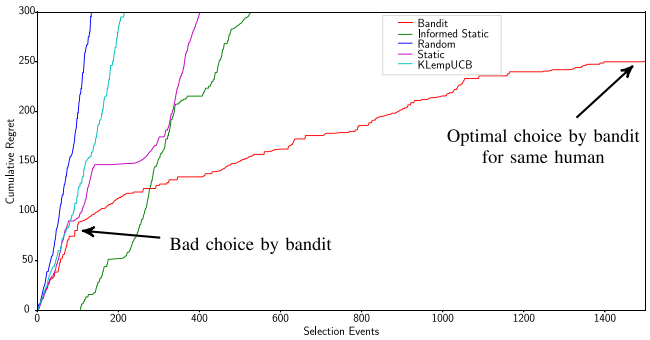

Fig. 7. An exploded view of a single run of the adaptive policy, zoomed in to emphasize the adaptive policy's performance at events 100 and 1450.

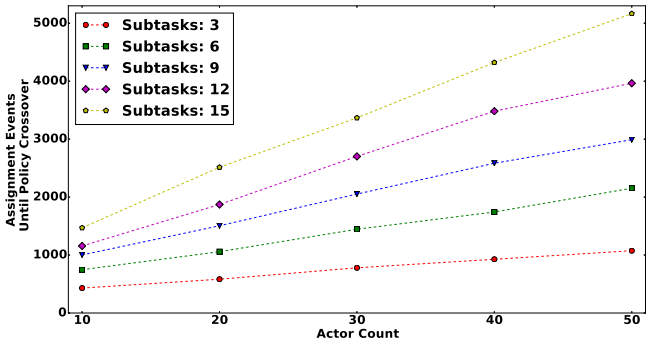

Fig. 8. Comparison of scaling with both actor count and number of subtasks.

B. Scalability Study

To investigate the scalability of the multi-arm bandit policy, we set up a series of experiments varying both the number of subtasks and the number of actors among which these subtasks are allocated. To compare amongst subtask-actor count combinations, we use the same significance point as the baseline case, that is, the number of allocation events needed for the multi-arm bandit policy to match the performance of the informed static policy, shown in Fig. 8. *Assignment events until crossover* indicates the number of subtask allocations that are required before the bandit policy outperforms the informed static policy, corresponding to the red lines in Fig. 6(a) & (c). ANOVA analyses are shown in Fig. 6(b) & (d).

C. Robustness and Comparisons to Non-Context-Based RL

To investigate the robustness of our adaptive policy to unpredictable variations in human behavior, we introduced a stochastic variation in the probability transition tables used to model human actor states. This variation was introduced by using the transitions of Fig. 4 as a base distribution for a Dirichlet prior to obtain random transition probabilities. A set of 100 Monte Carlo simulations was then run with unique human state transition tables per actor in the simulation drawn from the prior. In the stochastic transition case, crossover between the adaptive bandit policy and the informed static policy occurred at a point similar to that of our base case with known transition models. This simulation provides evidence that our choice of an adaptive algorithm is robust to variations in the human transition table.

Our simulations also included a non-contextual adaptive multi-arm bandit policy, the Kullback-Leibler Empirical Upper Confidence Bound (KLempUCB) algorithm of [30]. This algorithm was chosen in part because of its non-parametric nature, which makes it nominally well-suited to our problem of allocating tasks to dynamically constrained humans with complex reward functions (Parker's L-ALLIANCE [21] and its derivatives are not suitable for comparison since they are only valid for tasks with deterministic completion times). As shown in Fig. 6 *KLempUCB* has performance that barely outperforms random selection with scale $C = 5$ discretized over 50 bins of reward on $[0, 2.5]$. We believe this is because *KLempUCB* represents a family of upper confidence bound algorithms that have specific tuning parameters and require specific assumptions about the behavior of arms in the multi-arm bandit problem; in particular, the requirement that non-chosen arms remain stationary is not applicable when each arm represents a human. While choosing optimal tuning parameters for UCB-style algorithms is an interesting problem, it is not the focus of our work. Indeed, it can be argued that requiring a user to tune parameters in order to get superior performance ties the particular adaptive policy too tightly to the underlying dynamics of the problem. As such, while we do not argue here that a linear hybrid model is per se the best possible multi-arm bandit algorithm for failure recovery, we can conclude that adaptive bandit algorithms that consider performance dependencies can achieve higher performance than those that do not.

D. Discussion

We have shown that a multi-arm bandit solution can produce superior results over a long series of runs, but we have not addressed the nature of individual runs. The multi-arm bandit can only make superior decisions in a parameter regime where sufficient information has been gathered to make a reasonable extrapolation about expected reward. This behavior is evident in the increasing number of simulation events required to outperform the *informed static* policy; when additional actors have been added to the system, more exploration is required in order to refine internal productivity predictions. As shown in the scalability study (Fig. 8), the number of assignment events required to match the *informed static* policy is linear in both the number of subtasks and the number of actors. The bandit has the advantage of being an unsupervised learning technique that incorporates new exemplars to the underlying matrices for each actor as new assignments arrive without requiring the underlying matrices to expand. To ensure that the system is functional out-of-the-box, analogue missions and physics-based simulations could be used to capture initial performance results of supporting actors under varying conditions, seeding the multi-arm bandit.

One may question why a static policy cannot be augmented to account for actor state. In simulation, the *ad hoc* approach may yield reasonable results; however, a major challenge is that in a space exploration application, the underlying reward models are known to be incomplete and require further refinement that can only be accomplished *in situ*. The multi-arm bandit is capable of utilizing prior information that is both uncertain and incomplete. Our system can inherit any *a priori* assumptions or known training data about actor performance by adjusting the

initialization criteria of the multi-arm bandit; in a similar fashion, the evaluation criteria for subtask performance (corresponding to reward earned) can be arbitrarily defined and need not be identical across subtask classes. Once the system is activated, the multi-arm bandit is capable of determining the relationship between conditions and rewards, resulting in superior performance.

VI. Conclusions and Future Work

In this letter, we have developed a framework for robots to request assistance that is capable of utilizing rich past performance data to enlist the aid of the best assistant in a changing world with difficult or impossible to model dynamics. Our problem is motivated by a need to allow robots to function in situations where failures will occur. This framework has applications beyond space robotics to any application where autonomy failures may occur that require external intervention. Our framework enables onboard agents to recognize actor specialization and direct assistance requests to the most appropriate actor based on a black-box performance model, in contrast to the state of the art that relies on static analysis. Alternative allocation methods rely on system details such as accurate performance models that are simply not available in a space exploration scenario. We do not attempt to estimate internal parameters of any particular actor, as in a system identification problem, but rather only estimate current performance based on past experiences. In simulation, we have shown that our adaptive multi-arm bandit policy is capable of outperforming an informed static policy under changing actor and environmental conditions.

An important limitation of our algorithm, and of reinforcement learning techniques in general, is an assumption that even a poor selection of assistant will not result in mission-ending damage. Safe reinforcement learning is currently an active research topic.

Ongoing work includes validation of our approach via user studies with analogue mission profiles. In recent work [31], we have started to explore measurements of behavioral, cognitive, and physiological responses that can form the actor-specific context vectors. We also seek to implement realistic autonomy algorithms to evaluate our adaptive policy with a heterogenous set of specialized human and software agent actors in simulation and the real world.

References

[1] T. Estlin *et al.*, "Aegis automated science targeting for the MER Opportunity rover," *ACM Trans. Intel. Sys. Tech.*, vol. 3, no. 3, p. 50, 2012.

[2] M. Maimone, A. Johnson, Y. Cheng, R. Willson, and L. Matthies, "Autonomous navigation results from the Mars Exploration Rover (MER) mission," in *Experimental Robotics IX*. New York, NY, USA: Springer, 2006, pp. 3–13.

[3] T. Fong *et al.*, "Field testing of utility robots for lunar surface operations," in *Proc. AIAA SPACE 2008 Conf. Expo.*, p. 7886, 2008.

[4] B. G. Drake and K. D. Watts, "Human exploration of Mars design reference architecture 5.0, addendum #2," Tech. Rep. NASA/SP-2009-566-ADD2, Mars Archit. Steering Group, NASA Headquarters, Washington, D.C., USA, 2014.

[5] A. Elfes, C. R. Weisbin, H. Hua, J. H. Smith, J. Mrozinski, and K. Shelton, "The HURON task allocation and scheduling system: Planning human and robot activities for lunar missions," in *Proc. World Autom. Congr.*, 2008, pp. 1–8.

[6] B. Sankaran, B. Pitzer, and S. Osentoski, "Failure recovery with shared autonomy," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 349–355.

[7] R. A. Knepper, S. Tellex, A. Li, N. Roy, and D. Rus, "Recovering from failure by asking for help," *Auton. Robots*, vol. 39, no. 3, pp. 347–362, 2015.

[8] V. Verma, G. Gordon, R. Simmons, and S. Thrun, "Real-time fault diagnosis," *IEEE Robot. Autom. Mag.*, vol. 11, no. 2, pp. 56–66, Jun. 2004.

[9] T. Fong, C. Thorpe, and C. Baur, "Robot, asker of questions," *Robot. Auton. Syst.*, vol. 42, no. 3, pp. 235–243, 2003.

[10] D. Schreckenghost, T. Milam, and T. Fong, "Measuring performance in real time during remote human-robot operations with adjustable autonomy," *IEEE Intell. Syst.*, vol. 25, no. 5, pp. 36–45, Sep./Oct. 2010.

[11] M. Johnson, J. M. Bradshaw, P. J. Feltovich, C. M. Jonker, M. B. Van Riemsdijk, and M. Sierhuis, "Coactive design: Designing support for interdependence in joint activity," *J. Human-Robot Interaction*, vol. 3, no. 1, pp. 43–69, 2014.

[12] M. A. Goodrich and D. R. Olsen, "Seven principles of efficient human robot interaction," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2003, vol. 4, pp. 3942–3948.

[13] M. C. Gombolay, R. A. Gutierrez, S. G. Clarke, G. F. Sturla, and J. A. Shah, "Decision-making authority, team efficiency and human worker satisfaction in mixed human–robot teams," *Auton. Robots*, vol. 39, no. 3, pp. 293–312, 2015.

[14] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Res. Logistics Quart.*, vol. 2, nos. 1/2, pp. 83–97, 1955.

[15] B. P. Gerkey and M. J. Mataric, "Sold!: Auction methods for multirobot coordination," *IEEE Trans. Robot. Autom.*, vol. 18, no. 5, pp. 758–768, Oct. 2002.

[16] D. P. Bertsekas, "The auction algorithm for assignment and other network flow problems: A tutorial," *Interfaces*, vol. 20, no. 4, pp. 133–149, 1990.

[17] D. Zhu, H. Huang, and S. X. Yang, "Dynamic task assignment and path planning of multi-AUV system based on an improved self-organizing map and velocity synthesis method in three-dimensional underwater workspace," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 504–514, Apr. 2013.

[18] L. F. Bertuccelli and J. P. How, "Active exploration in robust unmanned vehicle task assignment," *J. Aerosp. Comput. Inf. Commun.*, vol. 8, no. 8, pp. 250–268, 2011.

[19] A.-B. Karami, L. Jeanpierre, and A.-I. Mouaddib, "Partially observable Markov decision process for managing robot collaboration with human," in *Proc. 21st Int. Conf. Tools Artif. Intell.*, 2009, pp. 518–521.

[20] J. Barry, L. P. Kaelbling, and T. Lozano-Pérez, "Det{H}*: Approximate hierarchical solution of large Markov decision processes," in *Proc. Joint Conf. Artif. Intell.*, vol. 11, pp. 1928–1935, 2011.

[21] L. E. Parker, "Task-oriented multi-robot learning in behavior-based systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 1996, vol. 3, pp. 1478–1487.

[22] D. S. Nau *et al.*, "SHOP2: An HTN planning system," *J. Artif. Intell. Res.*, vol. 20, pp. 379–404, 2003.

[23] T. Lu, D. Pál, and M. Pál, "Contextual multi-armed bandits," in *Proc. Int. Conf. Artif. Intell. Stat.*, 2010, pp. 485–492.

[24] A. Mahajan and D. Teneketzis, "Multi-armed bandit problems," in *Foundations and Applications of Sensor Management*. New York, NY, USA: Springer, 2008, pp. 121–151.

[25] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 661–670.

[26] D. Gingras *et al.*, "Lunar rover remote driving using monocameras under multi-second latency and low-bandwidth: Field tests and lessons learned," in *Proc. Int. Symp. Artif. Intell., Robot. Autom. Space*, Montreal, Canada, 2014.

[27] E. Reid, P. Iles, N. Cristello, M. Labrie, M. Musilova, and K. Staats, "Mobile robotic platform deployment as part of a Martian mission simulation," in *Proc. 12th Int. Symp. Artif. Intell. Robot. Autom. Space*, 2014.

[28] C. Fanchiang, "A quantitative human spacecraft design evaluation model for assessing crew accommodation and utilization," Ph.D. dissertation, Univ. Colorado at Boulder, Boulder, CO, USA, 2017.

[29] P. Furlong and M. Dille, "Productive information foraging," Tech. Rep. TM-2016-219376, NASA Headquarters, Washington, D.C., USA, 2016.

[30] O. Cappé *et al.*, "Kullback–Leibler upper confidence bounds for optimal sequential allocation," *Ann. Stat.*, vol. 41, no. 3, pp. 1516–1541, 2013.

[31] S. McGuire, P. M. Furlong, and N. Ahmed, "On the development of an online assistant selection dataset for planetary exploration systems," in *Proc. Robot. Sci. Syst. Workshop: Bridging Gap Space Robot.*, 2017.