Intelligent Agent Supporting Human-Multi-Robot Team Collaboration

Ariel Rosenfeld[a,*], Noa Agmon[a], Oleg Maksimov[a], Sarit Kraus[a]

[a]*Department of Computer Science, Bar-Ilan University, Ramat-Gan, Israel 52900.*

Abstract

The number of multi-robot systems deployed in field applications has risen dramatically over the years. Nevertheless, supervising and operating multiple robots simultaneously is a difficult task for a single operator to execute. In this article we propose a novel approach for utilizing automated advising agents in assisting an operator to better manage a team of multiple robots in complex environments. We introduce an advice provision methodology and exemplify its implementation using automated advising agents in two real-world human-multi-robot team collaboration tasks: the Search And Rescue (SAR) and the warehouse operation tasks. Our intelligent advising agents were evaluated through extensive field trials, with over 150 human operators using both simulated and physical mobile robots, and showed a significant improvement in the team's performance.

Keywords: Human-Multi-Robot-Interaction, Human-Robot-Interaction, Automated Agents, Advising Agents

1. Introduction

In recent years multi-robot systems have been applied to complex tasks that used to be performed by humans alone. These tasks include fire-fighting [1], landmine detection [2], decontamination of radiation [3], agricultural work [4], construction [5], underwater missions [6], warehouse operation [7] and Search And Rescue (SAR) [8]. The use of multiple robots for executing these tasks increases robustness and improves efficiency compared to the use of a single robot [9].

*Corresponding author

Email address: arielros1@gmail.com (Ariel Rosenfeld)

In most multi-robot systems today, the part of the human worker is often assumed to be marginal. Two hidden assumptions are made in this case: the first is that the robots perform relatively smoothly, with the infrequent need for human intervention; the second is that the human operator is only required to perform a single task at any given moment. Reality, however, can be more complicated on both accounts. The prioritization of the operator's tasks has mostly been overlooked in multi-robot system design, leaving the prioritization in each operator's hands. Sub-optimal prioritization of the operator's tasks has been shown to result in sub-optimal performance of the robot team and in a high cognitive workload for the operator [10, 11, 12].

In this article, we present a novel methodology that enhances operators' performance by using an *intelligent advising agent*. The agent provides advice for the operator regarding which actions s/he should take and acts as a smart filter between the robots' requests and the human operator. Our methodology, which uses a *myopic search heuristic*, is not restricted to any given hardware or algorithm used by the robots.

We evaluate our approach in two distinct tasks; the warehouse operation [7] and the SAR [8] tasks. The advising agents are designed, implemented and evaluated in extensive human experiments in two *realistic* real-world scenarios (in simulated environments and physical deployment) using more than 100 human subjects[1] and 10 physical robots. Experimental results show that our advising agents were able to significantly enhance the operators' performance when managing a large team of mobile robots in both the SAR and warehouse operation tasks.

This article makes the following contributions;

1. The main contribution of the article is in showing, for the first time, that an intelligent agent which provides advice for a human operator engaging in multi-robot supervision and control can lead to better performance of the human-multi-robot team.
2. We introduce the myopic advice optimization heuristic and demonstrate its effectiveness in overcoming large state spaces such as the ones common in multi-robot environments.
3. We show that in complex environments, such as the warehouse operation task, simple advising policies which are based on shallow planning may be outperformed by other policies which are based on deeper planning.

[1] All experiments were authorized by the corresponding IRB.

However, we further identify a potential tradeoff in which generated advice by simple policies is shown to be followed significantly more often by technically-oriented human operators.

The article is organized as follows. In Section 2, we survey related work. In Section 3, we present our problem setting and our proposed myopic search solution. In Section 4, we present the instantiation and evaluation of our advice provision methodology in the warehouse operation and SAR tasks and in Section 5 we discuss the cross-task insights and findings. Finally, in Section 6, we provide a summary and list recommendations for future work in this area.

2. Related Work

The deployment of robots in real-world environments has shown that they usually face difficulties in completing their tasks. Specifically, failures are common. In such situations, a human operator must get involved in order to solve the problem. That is, robots are usually *semi-autonomous* and should be supported by a human operator whenever they cannot handle the situation autonomously. For example, common household robotic vacuum cleaners may get stuck underneath a cabinet or on a thick carpet. The robots cannot then finish their task until a person provides assistance. Furthermore, while robot technology consistently becomes more autonomous and less prone to malfunctions, certain functions will still remain in human hands for the foreseeable future due to the human's ability to understand macro implications of decisions, other humans' intentions, and ethical responsibilities [13].

Human operators may be occupied by numerous tasks simultaneously and may be unable to provide the assistance the robots require instantaneously [14]. For example, in a warehouse operation setting, human workers may be employed in packing merchandise, refiling inventory and handling robot malfunctions, all at the same time. Olsen et al. [15] define the notion of *fan-out*, which is the number of robots that can be controlled by a single operator. They show that the effectiveness of a task performed by multiple robots that are controlled by a single operator reaches a plateau as the number of robots grows (the *fan-out plateau*). They provide a quantitative way of measuring the fan-out, the *fan-out equation*, which is defined as the ratio between the time a robot operates effectively following an interaction with the operator (the gain), and the duration of the interaction itself (the cost). Wang et al. [16] claim that there exists a *fan-out* effect, which means that the number of robots that a human operator can effectively operate at once lies

"somewhere between 4 and 9+ robots depending on the level of robot autonomy and environmental demands".

Expanding the human span of control over teams of semi-autonomous robots and improving the performance of human supervised multi-robot systems can be done using one (or both) of the following approaches (supported also by the fan-out equation [15]): (1) Improving the robot's hardware and software—thus relying less on human supervision (making the robots more autonomous); or (2) Improving the efficiency of the Human-Robot Interaction (HRI). Assuming we are given a team of robots, and we cannot control the reliability of its hardware or software, this article deals with improving the HRI in order to allow a person to control a team of many (possibly unreliable) robots (see [17] for a recent review of HRI status and challenges). Specifically, we consider improving the HRI by using an intelligent advising agent which provides advice for the operator regarding which actions s/he should take.

It is well established in multi-robot system literature that a single operator may get overwhelmed by the number of requests and messages from the robots, resulting in sub-optimal performance. For example, Chien et al. [18] have studied robots that could self-report encountered faults in SAR tasks. In their reported experiment, participants performed a foraging task while assisted by an event log, under different task loads (3 robots vs. 6 robots). The results show that participants in the 6-robot scenario did not perform better than those controlling only 3, while some even performed significantly worse. The results also show that operators devoted their resources in a sub-optimal way, leaving fewer resources for more urgent and critical tasks. These findings are consistent with previous studies with ground robots. For example, Velagapudi and Scerri [19] examined a simulated urban SAR scenario where a human operator controlled $4, 8$ and 12 robots. They reported a drop in the overall performance when the number of robots increased from 8 to 12. In [20] the authors identified that it is possible to allow the operator to control a larger team of robots by performing some of the tasks asynchronously (i.e., offline). Namely, the authors divided their SAR mission into 2 main parts: The synchronous part, where the operator is required to control the robots' navigation and obstacle avoidance *online* while handling errors as they occur. The asynchronous part consists of imagery analysis, namely the operator views the recorded images or video in hindsight and not in real-time. This approach allows a single operator to control a larger team yet it relies on the assumption that such offline analysis is applicable. In this work, we assume all tasks must be carried out in a synchronous manner.

Chen et al. [21] created RoboLeader, an agent that helps collaboration be-

tween a human operator and a team of robots in a simulated search environment. The agent's responsibility is to interpret the operator's intent into instructions to the team, so that the operator will not need to interact directly with the robots. The authors further investigate the influence of unreliable behavior of the system on the operator in [22]. It was shown that by using RoboLeader the mission was generally completed faster. Nevertheless, when comparing a system with 4 robots to one which includes 8 robots, having more robots only harmed the total performance, even while using RoboLeader. As apposed to the RoboLeader approach, in our research the operator does interact directly with the robots, and the agent assists the operator in deciding what task and robot to handle.

Methods for allowing n operators to oversee a team of more than n robots is also examined in teams of autonomous aerial vehicles (UAVs). Cummings et al. [23] note that the key factor for allowing one operator (or a small number of operators) to efficiently control a large number of UAVs is maintaining a low cognitive load on the human operator. The authors describe different degrees of autonomous operation, from fully autonomous to fully (remotely) controlled, and conclude that the performance of these systems relies on the cognitive abilities of the human operator. Ramchurn et al. [24] focused on assisting human operators in handling task-allocation decisions when controlling a team of UAVs, specifically when needing to handle dynamic changes in the environment. In an experiment with two operators jointly handling six UAVs, they show that when assisted by an automated task allocation, the load on the operators was significantly lower compared to manual task allocation, though the total performance of the system was not always improved.

Rosenthal et al. [25, 26] shifts the responsibility for human assistance to the robot. In their research, the robots should request (and receive) help from humans for actions they could not have performed alone due to lack of capabilities. However, in the presence of many robots' requests, Rosenthal's approach could (potentially) overwhelm an operator.

Several proposals have addressed the HRI ineffectiveness by providing intelligent interfaces and interaction modes that have been shown to ease the human operator's burden and increase the system's performance. Most common is the use of gesture-based interaction means. For example, Micrie et al. [27] presented and examined a hand and finger identification algorithm for controlling multiple robots efficiently. In a similar fashion, Stocia et al. [28] provided a human-friendly gesture-based system which maps human inputs into robotic commands. However, other modalities can be leveraged as well. For example, in [29], the authors allow a user to identify and classify an individual or a group of robots

using haptic stimuli, and name them using a voice command. In [30], the authors investigate the challenge of generating beneficial communication between robots to a human teammate for the purposes of coordinating joint actions.

Özgelen and Sklar [31] propose and evaluate a model for capturing the complexity of task scheduling problems from the human operator's perspective. Different techniques have been proposed to address this operator's challenge, for example in [32], where the authors investigate the manipulation of an object by a team of robots who follow a single human leader, thereby reducing the need for the human operator to interact directly with the robots. In the same spirit, Bevacqua et al. [33, 34] have investigated a framework in which the human operator is also involved in the scene and is thus co-located with the robots. As a result, the human operator cannot be fully dedicated to the robotic platforms, and can only provide sparse and sketchy commands. In this work, however, we focus on settings in which the human operator is capable of being fully dedicated and is needed for the smooth operation and supervision of the robots during their operation due to their limitations.

Our proposed methodology is not restricted to any given hardware or algorithm used by the robots. Namely, we consider these factors to be constants. However, if this assumption were to be relaxed, the human operator could also be asked to assist in the optimization of the robots' task allocation by taking an active part in their computation and coordination. Such an approach was recently proposed in [35]. The integration of this notion within our advice provision methodology is left for future work.

It is well known that people's cognitive abilities are limited [36]. Consequently, people have been shown to seek advice in order to improve the outcomes of their (imperfect) decisions [37, 38, 39]. That is, the person maintains her decision autonomy yet includes others in her decision process. In this work, we present automated advising agents for efficient human-multi-robot team collaboration. The advice, in our setting, is a form of relating recommendations or guidance from an automated agent to its human user. We focus our attention on agents that offer their advice based solely on the observed behavior of the users and the environment. This setting is referred to as *implicit interaction* [40]. Namely, the agent refrains from requesting explicitly defined information from the user.

Advising agents have been investigated and deployed in different settings where humans are in need of assistance. These agents have been shown to enhance human positive behavior and decrease human negative behavior. For example, in saving energy by setting a vehicular climate control system in a more beneficial way [41, 42], presenting better arguments in discussions [43, 44], saving fuel by

using better traffic routes [45] and in better economical behavior in buyer-seller interaction [46, 47]. The interest in advice providing agents is manifested in a special issue of the ACM transactions on interactive intelligent systems (TiiS) dedicated to human interaction with artificial advice givers [48]. In its general setting, an advising agent is faced with an optimization problem which it needs to solve [49]. Unfortunately, in many real-world settings, as well as in our settings, calculating optimal advice is intractable due to the exponential size of the state space and the high uncertainty induced by the environment, robots and humans. To the best of our knowledge, other than our preliminary work [50], no work has addressed the possibility of utilizing advising agent technology in assisting a human operator to better manage a team of robots in complex environments.

Myopic search, which is also known as lookahead search, is one of the most widely used techniques for handling intractable decision optimization problems [51]. In myopic search, instead of searching through the entire relevant state space, an agent uses a local, *bounded depth* search tree of possible actions and reactions. The agent then chooses the action that maximizes its payoff in its search space. Agents deploying myopic search are especially prominent in human-agent negotiation [52], economical decision-making [53] and game playing [54].

We assume *non-expert* human operators, which are common in many real-world applications. For example, fire-fighters which deploy robots in burning buildings to detect flame sources or human victims, cannot be expected to train solely on the control of robots and cannot be considered to be experts. Furthermore, we could not expect a fire squad to have a robot expert on hand at every given moment. Nevertheless, we assume that the operator has decent technological skills and has undergone some (basic) training with the robots.

We focus on situations where it is not feasible to collect data on each specific operator. For example, when an operator has very little experience in robot supervision and control, and where such experience is expensive to attain. In particular, the agent should be able to support the operator from its first interaction with the system.

Note that the proposed approach is not limited to multi-robot systems and could potentially be useful for single-robot and multiple-operator applications as well. In this study we focus on environments in which a single operator supervises and operates multiple robots.

3. The Advice Optimization Problem

We consider a set of N robots engaged in a cooperative task. A single human operator, denoted by O, is in charge of supervising the robots as well as performing different domain specific actions. The *state space* S consists of all information regarding the robots (e.g., robot location, battery capacity, operational status) and the task, which is domain specific. An instance of the state space is denoted by $s \in S$. The operator, O, can perform *actions* during the task, at any time t, from a predefined set - A. Note that O can choose to execute no actions, i.e., $NULL \in A$. The task takes place during a predefined time interval $\mathbb{T} = [0, T]$, where 0 denotes the beginning time of the task and T is its termination time.

The advice optimization problem consists of several components:

- *Advice:* A piece of advice is defined as a possible action $a \in A$ suggested by an automated agent for the human operator to perform. In this study, we assume that the agent can only provide one piece of advice at a time. The operator is not obligated to accept the agent's advice, namely, she does not have to perform the suggested actions.

- *Operator Model:* In state s and time t, the operator, according to her abilities, endures a cost (usually in terms of time) for performing action a, denoted by $C_o(s, a)$. If a is *infeasible* in state s (defined by the domain characteristics), then $C_o(s, a) = \infty$. We assume that $C_o(s, NULL) = 0$. In the context of this study, we assume that the cost is measured solely in terms of time. We refer to this cost function as the *operator model*.

- *System Dynamics:* $P_o(s_1, a, s_2, C_o(s_1, a))$ provides the probability of reaching state s_2 given action a in state s_1 and the operator model, $C_o(s_1, a)$. Note that the transition from s_1 to s_2 given a may not be instantaneous. Specifically, during the transition from s_1 to s_2, intermediate states may be experienced. The intermediate states may be the same as s_1 (i.e., during a's execution, the world state does not change from s_1) or they could be different from s_1 (i.e., during a's execution a new state is experienced). We define the sequence of intermediate states as $Int(s_1, a, s_2, C_o(s_1, a))$. Namely, $Int(\cdot)$ is a function specifying the sequence of intermediate states experienced when performing action a in order to transition from state s_1 to state s_2 given $C_o(s_1, a)$. For simplicity, throughout this article we assume $Int(\cdot)$ returns a single intermediate state $s' = Int(s_1, a, s_2, C_o(s_1, a))$ which can be the same as s_1 or different. As a result, $s' = Int(s_1, a, s_2, C_o(s_1, a))$

does not depend on the time that has passed from the execution of action a. For example, robot X has a low battery (s_1). The operator decides to change its battery (a), and therefore, for a time of $C_o(s_1, a)$, the operator is engaged in the task of changing the battery of the robot (s'). We assume s' does not depend on time. Following a successfully battery change, a new state in which the robot has a new battery is experienced (s_2).

- *System Reward:* $R_o(s)$ provides a real value representing the expected reward of state s, usually in terms of task fulfillment. For example, in a game setting, one can define $R_o(s)$ to be the expected number of points gained per one minute of game play given the current state s.

- *Discounting:* $\gamma \in (0, 1]$ is the discount factor, which represents the difference between future (uncertain) rewards and present rewards. For instance, γ can capture the uncertainty of receiving future rewards.

Ideally, we would like the operator to execute the **optimal policy** $\pi_o^* : S \times \mathbb{T} \rightarrow A$ which maximizes the expected accumulative reward from its deployment given $S, A, P_o(\cdot), Int(\cdot), R_o(\cdot)$,$\gamma$ and $C_o(\cdot)$, which are defined above. Finding π_o^* naturally translates into a Markov Decision Process (MDP) [55] consisting of state space S, action space A, a transition model composed of P_o and Int, a discounting factor of γ and a reward function determined by R_o and C_o. However, calculating π_o^* is intractable for real-world domains due to the exponential size of the state space and the high uncertainty induced by the environment, robots and operators. This difficulty also stems from the complexity of many multi-robot problems, such as the task allocation problem which is NP-hard [56]. In this article, we consider a more tractable advice optimization problem which uses a myopic, k-steps-lookahead heuristic.

The k-Myopic *Advice Optimization Heuristic*

The k-Myopic Maximization Advice Optimization (*k-max-MYAO*) Heuristic is defined as follows.

Given a state $s \in S$, time t and $k \geq 1$, we define the $Value$ function capturing the expected cumulative future reward by using k sequential pieces of advice, starting at state s at time t. That is, given that the task had terminated (i.e., $t \notin \mathbb{T}$), no future reward is possible. Otherwise, the agent can provide advice until it exhausts its advice budget k. When a piece of advice is given, $Value$ weighs

the possible resulting states using the system dynamics function $P_o(\cdot)$, and contemplates both the intermediate state $Int(\cdot)$ and the recursive calculation of the expected cumulative reward by using $k-1$ pieces of advice in the resulting state.

$$Value(a,s,t,k) \tag{1}$$
$$= \begin{cases} 0 & \text{If } t \notin \mathbb{T} \\ \int_{s' \in S} P_o(s,a,s',C_o(s,a))(\int_{z=t}^{min\{t+C_o(s,a),T\}} \gamma^{z-t} R_o(Int(s,a,s',C_o(s,a))dz & \text{If } k \geq 1. \\ +\text{max}_{a' \in A} Value(a',s',t+C_o(s,a),k-1))ds') & \\ \int_{z=t}^{T} \gamma^{z-t} R_o(s) & \text{otherwise.} \end{cases}$$

Using Equation 1, k-max-MYAO heuristic generates advice a^* such that:

$$a^* = \text{argmax}_a Value(a,s,t,k) \tag{2}$$

In words, in state s at time t, an agent suggests the action a^* which maximizes (minimizes) the expected *k-steps-lookahead* reward. Namely, in state s and time t the value of providing advice a using a k-max-MYAO heuristic is calculated as follows: given that the task has terminated, the value is naturally set to 0. Otherwise, we integrate over all infinitesimal states and calculate the expected reward from performing a and additional $k-1$ actions from the resulting state. That is, given the probability of reaching each state s', we integrate over time and calculate the discounted expected reward from both the intermediate state and the resulting state recursively. Once the advising budget is exhausted the expected reward is easily computed. Note that $Value(\cdot)$ is monotonic non-decreasing in k. The reason is that $NULL \in A$, and therefore, Equation 2 will not return any $a \neq NULL$ unless it improves the expected cumulative reward over suggesting $NULL$, which is assumed to carry no cost (see the beginning of this section).

The K-min-MYAO problem is defined symmetrically by replacing argmax and max with argmin and min, respectively.

Only in extremely small environments (i.e., small state space, action space and $\mathbb{T}$) can a dynamic programming approach be used to calculate and store $Value(\cdot)$ for every a,s,t,k. For example, in a simple grid-world of size 10×10 with 4 actions and $T=10$, there are $4 \cdot 100^{10}$ entries for $Value(\cdot)$, which is impractical. In realistic environments, such as the ones we examine in this article, $Value(\cdot)$ has to be calculated online given specific a,s,t,k.

Algorithm 1 solves the *k-max-MYAO* for a given k.

Algorithm 1 k-Step-Lookahead Advice Provision

Require: k, T

```
s ← InitialState()
t ← 0
repeat
    max ← −∞
    for each a ∈ A do
        expRwd ← Value(s, a, t, k)                ▷ Eq. 2
        if expRwd > max then
            advice ← a
            max ← expRwd
    OUTPUT advice
    s ← GetCurrentState()
    t ← GetTime()
until t ≥ T
```

Algorithm 1 runs in $O(|A|^k)$. A is assumed to be fixed for a given domain. Nevertheless, iterating over the entire set A can be unnecessary when some actions cannot be performed given state s. For example, if a robot has malfunctioned the agent will not advise to tele-operate it, but it might advise to fix the robot (as long as the operator has not fixed the robot already). Therefore, a simple enhancement to Algorithm 1, which we use in this study, is to restrict the agent from considering non-practical actions given state s. Practically, we use an efficient priority queue in which all eligible tasks are maintained. Each task a is assigned with a priority which is $Value(s, a, t, k)$, where s is the current state, t is the current time and k is the search depth. A designated process adds and updates the priority of the tasks every few seconds.

Scaling remains an issue for large values of k due to the recursive definition of $Value(s, a, t, k)$. Therefore, for large k values, Algorithm 1 may be infeasible. In our empirical evaluation (Section 4), we were able to solve the k-*max-MYAO* for $k = 1, 2$ very quickly (less than 1 second of calculation time) using a PC with 2 CPUs, each with 6 cores, where each core operates at 2.8 GHz. For solving the *3-MYAO* ($k = 3$) heuristic in real-time (under 1 second for calculating the advice) a more sophisticated, multi-threaded implementation of Algorithm 1 was deployed. We were unable to efficiently solve for $k = 4$ in our setting. Therefore, in this study, we focus on the k-*max-MYAO* heuristics of $k < 4$.

4. Empirical Evaluation

In this section we seek to investigate our working hypothesis, which is two-fold: 1) We hypothesize that our advice provision methodology will enhance the human-multi-robot team performance in real-world settings. 2) We hypothesize that deeper search methods will bring about more beneficial pieces of advice in terms of their long-term benefit. Therefore, we empirically evaluate our advice provision approach in extensive human experiments. We performed the evaluation in two *realistic* real-world scenarios (using both simulated environments and physical deployment) with more than 150 human subjects. The experiments demonstrate our approach's benefit in two distinct simulated real-world scenarios as well as with physical robots.

In order to correctly solve the optimization problem (Equation 2), we need to define $C_o(\cdot)$, $R_o(\cdot)$, $P_o(\cdot)$, $Int(\cdot)$ and γ for the specific settings. However, advising agents should be able to support the operator from its first interaction with the system. Therefore, throughout this section we use *generalized models*. Specifically, our advising agents use four generalized models: an *operator model* to approximate $C_o(\cdot)$; a *reward model* to approximate $R_o(\cdot)$; a *transition model* to assess $P_o(\cdot)$; and an *intermediate model* to capture $Int(\cdot)$. Note that given personalized models, $C_o(\cdot)$, $P_o(\cdot)$ and $R_o(\cdot)$, our agent could also provide personalized advice. However, these models are unavailable in the scope of this work, and will be considered in future work.

In order to avoid overloading the above notations, we use a superscript to indicate the domain, e.g., $R^w(\cdot)$ and $R^s(\cdot)$ represent the reward function of the warehouse operation and the search and rescue domains, respectively.

Per evaluation domain, we will first define the domain's task, followed by the training of the generalized models and selection of γ as described above. Last, we present the evaluation results.

4.1. Warehouse Operation

We instantiate the k-MYAO heuristic (Section 3) for providing advice in a warehouse operation task[2]. We will first describe the warehouse operation task, followed by our agents' design and evaluation. The simulation specification and code release are available in Appendix A.

[2]A short video summarizing the work presented in this section is available at `http://www.youtube.com/watch?v=rC1a4c6Voco`

4.1.1. The Warehouse Operation Task

Modern warehouses use robots to move merchandise from one location to another. The robots move the requested merchandise to the packaging stations where human workers fill orders. The human workers may also be required to handle different abnormal situations. For example, during the robots' work, products may fall in the work area, potentially causing the robots to get stuck. While advanced methods are used to lay out the robots' pathways and their actions, thus far, the prioritization of the human workers' tasks has been left in each operator's hands.

To address this issue, we concentrate on a small warehouse acting as a fulfillment center (also known as a packing center) where there is *one* human worker and a large number of mobile ground robots. In our setting there are 10 robots.

For this work, a warehouse simulation system was designed and implemented (see Appendix A for the system's specification and code)[3]. Figure 1 provides a snapshot of the warehouse simulation.

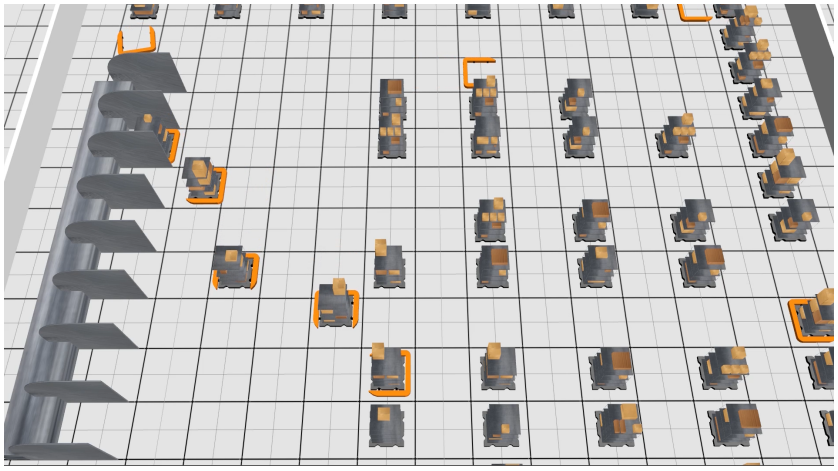

Figure 1: Simulated Warehouse Environment. The robots, represented in orange in the center of the screen, move the merchandise to the packaging stations, represented on the left side.

When a robot needs the operator's attention, for example when it gets stuck or when it arrives at a packing station, it signals the operator using a designated

[3]During the time in which this work was conducted there was no standard simulator for warehouse planning. Environments such as Amazon Robotics Picking Challenge (https://www.amazonrobotics.com/#/pickingchallenge) focus on picking and stowing tasks which should be performed autonomously. Therefore, such environments cannot be used in this study.

graphical interface. For that purpose, we designed a GUI (Figure 2) that provides real-time feedback on the task's and robots' states (see Appendix A for more details).

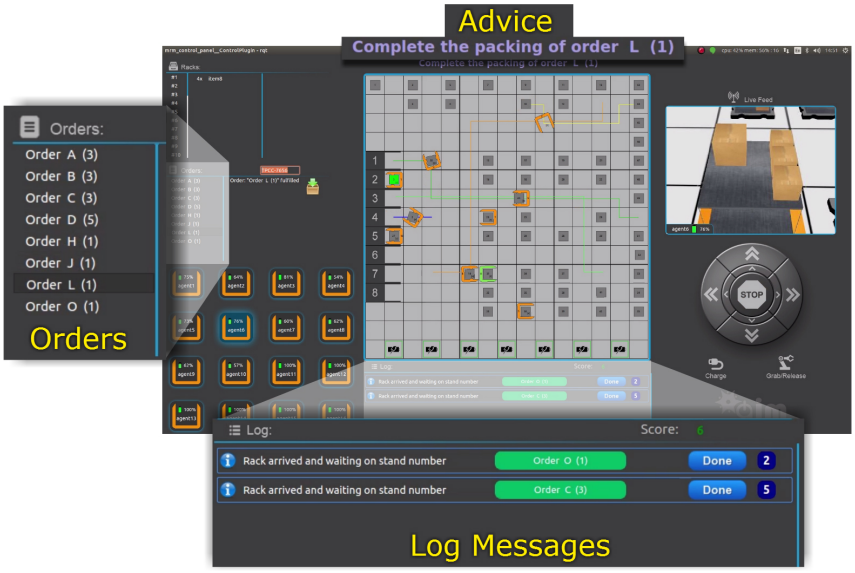

Figure 2: The graphical user interface used for the warehouse operation task in this study.

Overall, the human worker has to make complicated decisions in real-time while taking into consideration the active orders, the time, the movement of the robots and her own personal task load.

4.1.2. Training

We will first define the task's state space S. In a preliminary experiment we found out that obstacles on the warehouse floor cause a substantial disturbance to the robots. We allow the human operator to mark the position of an obstacle and thereby instruct the robots to avoid the problematic spot. We refer to these instructions as "restricted cells" (more details are available in Appendix Appendix A). Similar to [31], we define an obstacle or a restricted cell as *critical* if it prevents the completion of a task (e.g., if it blocks a robot's only path to a packing station). Specifically, if a robot cannot arrive at its destination due to the existence of an obstacle or a restricted cell, it is considered critical. We represent each world state $s \in S$ using the following set of state variables: the number of inac-

tive robots ($Inactive(s)$); the number of active orders ($Orders(s)$); the number of obstacles ($Obstacles(s)$); the number of critical obstacles ($CObstacles(s)$); the number of restricted cells ($Restricted(s)$) and the number of critical restricted cells ($CRestricted(s)$).

A is the action set that the operator can take. Recall that A is also the advice set from which the agent can propose advice. We define A as the instantiations of the following 7 action schemes: "Robot i is waiting for your command", "Unload item x at station y.", "Complete the packing of order z.", "Clear an obstacle from the floor.", "Obstacle was detected – restrict its cell." "Clear a critical obstacle from the floor.", and "Critical obstacle was detected – restrict its cell.".[4]

Training $R^w(\cdot)$:

We define $R^w(s)$ as the expected number of filled orders per minute starting at state s without human intervention. Namely, in order to learn $R^w(\cdot)$, we ran 150 hours of sessions, each one lasting 12.5 minutes, in our warehouse simulation environment without a human operator present. During these sessions, the system receives 26 orders which are uniformly distributed over the 12.5 minutes. Two orders out of the 26 consist of 3 requested products, eight consist of 2 requested products and the remaining orders consist of a single requested product. The products requested in each order are selected at random. The described setting was chosen empirically to allow sufficient load on the system yet avoid traffic congestion over the warehouse floor. In order to learn the effect that fallen products have on the robots' productivity, we use between 0 and 3 products that are uniformly scheduled to fall on the warehouse floor during each session. As no human operator is present during the session, the fallen products caused robots to get stuck. Recall that the human worker can keep the robots from going through problem spots by marking these spots as "restricted spots". To quantify the effect such restricted spots have on the robots' productivity, during each session we set between 0 and 3 restricted spots at random on the map.

Each recorded session was translated into 690 segments, each representing one minute starting at a different second (each session lasts 750 seconds). Each segment was translated into a set of state variables according to the system state

[4]There are 520 instantiations of the 7 action schemes: "Robot i is waiting for your command" (10 options), "Unload item x at station y." (480 options), "Complete the packing of order z." (26 options), "Clear the obstacle from the floor." (1 option), "Obstacle was detected – restrict its cell." (1 option), "Clear a critical obstacle from the floor." (1 option), and "Critical obstacle was detected – restrict its cell." (1 option) where $i \in [1, \ldots, 10]$, $x \in [1, \ldots, 60]$, $y \in [1, \ldots, 8]$ and $z \in [A, B, \ldots, Z]$. See Appendix A for more information.

s at the beginning of the segment and labeled with the number of filled orders during those 60 seconds. Specifically, from each state s, we extract $Inactive(s)$, $Orders(s)$, $Obstacles(s)$, $CObstacles(s)$, $Restricted(s)$ and $CRestricted(s)$.

We model $R^w(s)$ using the following Equation:

$$R^w(s) = e^{-(\alpha_0 Restricted(s)+\alpha_1 CRestricted(s)+\alpha_2 Obstacles(s)+\alpha_3 CObstacles(s))} \cdot \alpha_4 Orders(s) - \alpha_5 Inactive(s)^2 \quad (3)$$

where $\alpha_0, \ldots, \alpha_5$ are learned using non-linear, least squares regression [57].

This form of function assumes that only some of the active orders are eligible for packing during the next minute. This portion is then reduced by the number and types of obstacles and restricted cells in an exponential manner. It also assumes that the squared number of inactive robots decreases the expected number of filled orders to be completed in the next minute. This form of function was compared to more than 100 other forms; Some of the other function forms that were tested include other parameters that were found to be insignificant, such as the distance between the robots, the number of open packing stations, etc. These forms yielded a lower fit to the data we collected (as described next). All parameters are assumed to be positive, and in fact reached positive values.

Training $C^w(s,a)$ and $P^w(s,a,s',C(s,a))$: We define $C^w(s,a)$ as the expected time for completing a by the average human operator and $P^w(s,a,s',C(s,a))$ as the probability of transitioning from s to s' using advice a, given $C(s,a)$. To learn the operator's model $C^w(s,a)$ and the transition model $P^w(s,a,s',C(s,a))$, we recruited 30 Computer Science senior undergraduate students, ranging in age from 21 to 39 (mean=26, s.d.=2.8) with similar demographics, 19 males and 11 females, to participate in a warehouse operation task equipped with a *naïve* advising agent. Our naïve advising agent maintains an open list of possible advice. The agent adds advice to the list once the advice is eligible, that is, once the advice can be carried out by the human worker. For example, once a shelf arrives at the packing station, the agent adds the mission "Unload item x at station y." according to the relevant item and station. The agent provides the advice on a first-come, first-served fashion (i.e., naïvely). If the advice at the top of the list has become irrelevant or the advice was already performed by the operator, then the advice is discarded and the next advice is used instead. The agent presents the advice in both textual format (see Figure 2) as well as in a prerecorded, human-voice message, played in the operator's head-set. Note that the agent *filters and prioritizes* robots' messages, yet in a naïve way. For motivation, subjects were paid 1 NIS (about 25 cents) per order they pack, and 0.5 NIS per advice to which

they adhere. Note that in this training phase we want to learn the time it takes for a subject to complete different pieces of advice rather than to estimate the likelihood that one would choose to execute a piece of advice, which is captured in $P^w(s, a, s', C(s, a))$.

All pieces of advice that our agent gave were successfully executed and completed. Therefore, we set $P^w(s, a, s', C(s, a))$ to be 1 whenever completing advice a in s is possible and leads to s', and 0 otherwise. Note that human workers are *not* assumed to adhere to all pieces of advice provided by the agent when no motivation is given for doing so (as is the case in the following evaluation). However, we do assume that successful completion of the agent's advice is achievable given the human worker's capabilities. Due to the naïve agent's advice provision approach, combined with the monetary reward for adhering to its advice, the subjects' performance was poor in terms of filled orders. On average, subjects filled less than 8 orders compared to over 20 in all tested settings without the naïve agent in Section 4.1.3. Therefore, the obtained data will not be used next in evaluating the quality of the advice provision methods. When analyzing the operators' behavior, we were unable to find good *cross-subject* features that would help us to predict the expected time it would take to complete a piece of advice. Therefore, we used the mean execution time of a across our subjects as the estimation for $C^w(s, a)$.

Estimating $Int^w(s, a, s', C(s, a))$: In order to estimate $Int(s, a, s', C(s, a))$, which returns the intermediate state experienced in the process of transitioning from state s to s' using a, we used the help of an expert. An expert-based function $S \times A \times S \rightarrow S$ was articulated, mapping each $< s, a, s' >$ tuple to an intermediate state s''. For example, when an obstacle falls on the warehouse floor (s) and the action $a = clean$ is performed, the state in which the robots cannot approach the area is experienced (s'') before reaching the state in which the obstacle has been removed (s').

Setting γ: In our setting, we do not distinguish between present and future rewards as we assume the task's time ($\mathbb{T}$) is predefined. Therefore, we chose to set γ to 1.

Overall, 150 hours of simulations were used to train $R^w(\cdot)$, 30 human subjects were recruited for training $C^w(s, a)$ and $P^w(s, a, s', C(s, a))$, and a single human expert was needed for estimating $Int^w(s, a, s', C(s, a))$.

4.1.3. Experimental Design

We design and evaluate 3 agents, denoted as *W-AID1*,*W-AID2* and *W-AID3* that are aimed at solving the *k-max-MYAO* ($k = 1, 2, 3$) using Algorithm 1. Namely, the agents provide the best advice (per their optimization problem defi-

nition) to the human operator. The presented advice may change at time t if the advice was successfully completed or when a more urgent and productive piece of advice becomes available, making our agents *adaptive* to the changing environment.

Examining the potential benefit of W-AID1:

Preliminary testing with a few Computer Science students (who did not participate in the data collection phase in Section 4.1.2) pointed out a significant improvement in the operators' performance when performing the task for the second time. Therefore, we first use a *between-subjects* experimental design where each subject performed the warehouse operation task twice (a week apart) and the subjects' results from the first session are disregarded. This procedure allowed subjects to gain sufficient understanding of the system in their first trial yet avoid potential biases such as the subjects remembering specific configurations for their second trial.

We evaluate the *W-AID1* agent compared to the condition in which no advising agent is deployed. We recruited an additional 30 Computer Science students, who did not participate in the data collection phase in Section 4.1.2, to participate in the warehouse operation task. Subjects were BSc and MSc Computer Science students, ranging in age from 21 to 39 (mean=26, s.d.=2.8) with similar demographics, 15 males and 15 females. Each subject was trained prior to the experiment: s/he watched a short video explaining the task[5], underwent a structured 1-on-1 hands-on tutorial on the system by our research assistant and had to pass a short test. The test was conducted to make sure that the operator was capable of successfully completing the task. During the test, the subject had to fill 3 orders without any time limit while s/he encountered two simulated malfunctions. Subjects were motivated to fill as many orders as possible during the task using a small monetary reward (1 NIS per order). However, they did not receive any monetary reward for accepting the agent's advice. Similar to the setting used in Section 4.1.2, during each session the system receives 26 orders which have been spread uniformly across the task's 12.5 minutes. Two orders out of the 26 consist of 3 requested products, eight consist of 2 requested products and the remaining orders consist of a single requested product. Furthermore, 9 simulated malfunctions are set uniformly. In order to support our between-subjects experimental design, all subjects experienced the same series in which orders and

[5]The tutorial video is available on `http://www.youtube.com/watch?v=q-0bxm8n3Hw` (in Hebrew).

malfunctions arrived at the system. All 30 subjects were first asked to perform the task without the *W-AID1* agent's help to get an understanding of the system. Then, a week afterwards, half of the subjects (15) were asked to perform the task once again without the *W-AID1* agent's help while the other half were equipped with the *W-AID1* agent. Only the subjects' second trial results were considered in the following analysis.

In order to examine the possible influence that the advising agent may have on the subjects' perceived workload, upon completion of the task subjects were asked to answer a standard NASA-TLX questionnaire [58]. The NASA Task Load Index (NASA-TLX) is a widely used, subjective, multidimensional assessment tool for perceived workload in a given task or setting.

Subjects were informed that they could turn off the advising agent if and whenever they choose.

Comparing W-AID1 to W-AID2:

We further evaluate the *W-AID1* agent, this time as compared to the *W-AID2* agent. We recruited 30 additional Computer Science students, who had not participated in the study thus far, to participate in the warehouse operation task. The subjects were senior BSc Computer Science students, ranging in age from 20 to 30 (mean=25, s.d.=2.2) with similar demographics, 20 males and 10 females. To compare the agents, each subject performed the task twice (a week apart); once equipped with the *W-AID1* agent and once equipped with the *W-AID2* agent. This time, subjects were counter-balanced as to which condition was applied first. The experiment protocol was the same as the one used above for comparing the *W-AID1* agent to the no-advising agent condition. Specifically, subjects experienced the same series in which orders and malfunctions arrived at the system, and the scores from their first sessions were disregarded.

Comparing W-AID2 to W-AID3:

This time, we evaluate the *W-AID2* agent as compared to the *W-AID3* agent. Unfortunately, we were unable to recruit additional Computer Science students from our university. Therefore, for this experiment, we required 30 Bachelor students who major in other fields (6 Exact Sciences, 6 Humanities, 12 Social Science, 6 Life Sciences). Naturally, these subjects had not participated in the study thus far. Subjects were ranging in age from 19 to 27 (mean=23, s.d.=1.3) with similar demographics, 16 males and 14 females. Similar to the comparison of *W-AID2* and *W-AID1*, each subject performed the task twice (a week apart); once equipped with the *W-AID2* agent and once equipped with the *W-AID3* agent (subjects were counter-balanced as to which condition was applied first). However, unlike Computer Science students, this subject group was much more diverse in

their technical abilities and knowledge, as shown by the high variance in their experiment scores in Section 4.1.4. This fact prevents us from using a between-subjects experimental design. therefore we used a within-subjects experimental design where both of each subject's trials were considered.

Unfortunately, in preliminary testing, we noticed that the previously used series in which orders and malfunctions arrive at the system makes the advice produced by *W-AID2* and *W-AID3* coincide in their provided advice in more than 90% of the cases compared to approximately 80% in the case with *W-AID1* and *W-AID2*. Therefore, a new series in which orders and malfunctions arrived at the system was sampled. Following, in yet another preliminary testing, we made sure that the *W-AID2* and *W-AID3* coincide in less than 85% of their provided advice. Note that as we use a within-subjects experimental design, we are no longer restricted to using the same series as used before.

4.1.4. Results

When comparing the *W-AID1* condition to the condition in which no agent was deployed, the results show a statistically significantly higher average number of filled orders under the *W-AID1* condition for the subjects' second trial. Subjects equipped with the *W-AID1* agent averaged 24 packed orders (out of a possible 26) while subjects that were not assisted by an agent averaged 22 orders.

The results also show that subjects equipped with the *W-AID2* agent packed significantly more orders compared to subjects equipped with the *W-AID1* agent. Subjects equipped with the *W-AID1* agent averaged 23.8 packed orders (out of a possible 26) while subjects equipped with the *W-AID2* agent averaged 25 orders.

Both of the above results were found to be significant using a standard t-test with $p < 0.05$ (a Shapiro-Wilk Normality Test [59] was executed to ensure a valid use of the t-test). See Figure 3 for a summary.

The results further show that while equipped with the *W-AID3* agent, subjects packed significantly more orders compared to the condition in which they were equipped with the *W-AID2* agent. Under the *W-AID3* condition, subjects packed 21.2 orders on average, compared to 18.4 orders under the *W-AID2* condition. This result was found to be significant using a pairwise standard t-test with $p < 0.05$. As the subject group for this experiment was significantly different from the one employed in the above two experiments and used a differently sampled order and malfunction series, the results cannot be directly compared with the results of the previous two experiments. See Figure 4 for a graphical summary.

Note that a high variance was recorded for this subject group compared to the previous ones. Specifically, while in the Computer Science student group standard

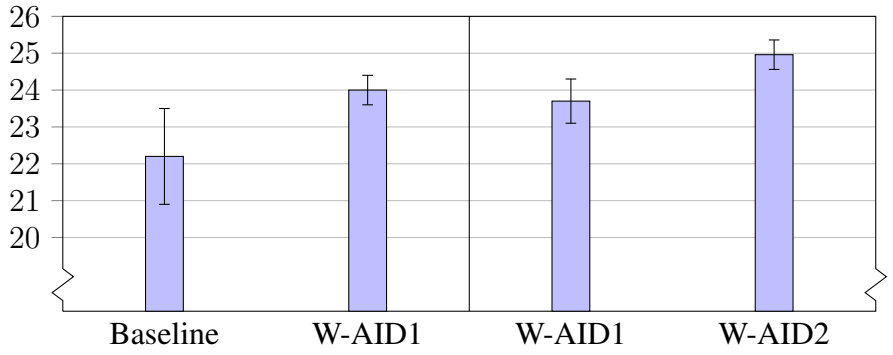

Figure 3: Average number of filled orders per advising condition in the warehouse operation task. Error bars indicate standard errors.

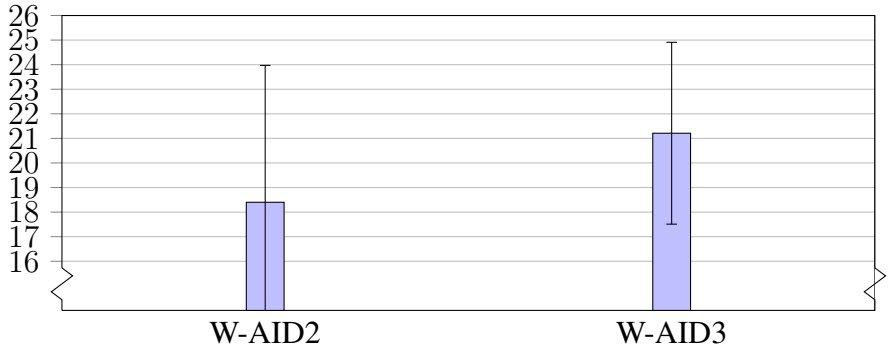

Figure 4: Average number of filled orders per advising condition in the warehouse operation task. Error bars indicate standard errors.

devisions varied between 0.4 and 1.3 across the different conditions, here, for students who do *not* major in Computer Science, subjects who used the *W-AID2* and *W-AID3* agents recorded standard deviations of 5.5 and 3.7, respectively.

We further analyze the subject's acceptance rate of the agent's advice. Advice is considered *accepted* if it is performed by the subject while it is displayed on the GUI.

In our first two experiments, under both the *W-AID1* and *W-AID2* conditions, subjects were presented with approximately the same number of pieces of advice (56.5 under the *W-AID1* condition compared to 56.1 under the *W-AID2* condition with no statistically significant difference between the two). Nevertheless, contrary to what the authors initially expected, subjects equipped with the *W-AID1* agent, which was found inferior to the *W-AID2* agent in terms of packed orders,

accepted significantly more of the agent's advice than subjects equipped with the *W-AID2* agent. Specifically, while subjects were presented with approximately the same average number of pieces of advice under both conditions, considering the subjects' second trial, subjects equipped with the *W-AID1* agent accepted an average of 51.8 pieces of advice (91.6% of the total number of pieces of advice presented), which is significantly higher than subjects equipped with the *W-AID2* agent, who accepted an average of 45.2 pieces of advice (80.6%), $p < 0.05$. Note, however, that despite these results, *W-AID2* is superior to *W-AID1* in terms of packed orders. Subjects' acceptance rate for an agent's advice is defined as the percentage of accepted advice by all subjects combined. Figure 5 presents the subjects' acceptance rate per 1 minute of trial for both *W-AID1* and *W-AID2*.

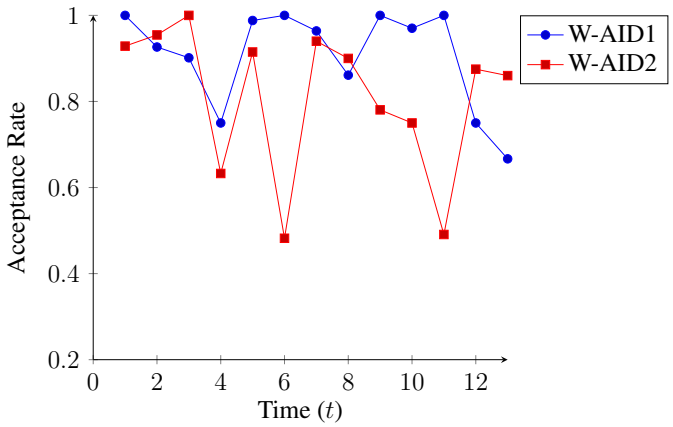

Figure 5: Subjects' acceptance rate per 1 minute for each of the tested agents. Results represent the subjects' second trials.

We now turn to analyze the subjects' acceptance rate of the agent's advice in the third experiment. First, similar to the first two experiments, subjects received approximately the same number of pieces of advice under both conditions (64.3 pieces of advice under the *W-AID2* condition compared to 64.7 under the *W-AID3* condition, without a statistically significant difference). However, a surprising result was found concerning the subjects' acceptance rate: under both conditions, subjects accepted more than 98% of the advice (98.6% under the *W-AID2* condition compared to 98.4% under the *W-AID3* condition, without any statistically significant difference between the two). There are two main possible answers to explain the observed high acceptance rate: 1) as our subject pool consisted of non-technically-oriented students, students heavily relied on the provided advice

regardless of condition; 2) the newly sampled orders and malfunction series were more complex than the previously tested series, which could explain why human operators rely heavily on provided advice.

The results did not show a statistically significant difference in the reported average workload using the TLX questionnaires under the tested conditions. An average TLX score of 50 was recorded for the baseline condition, whereas *W-AID1* and *W-AID2* averaged 55 and 53, respectively, in the first two experiments. Similarly, *W-AID2* and *W-AID3* averaged 47 and 48, respectively.

Recall that the agent presents the advice in both textual format as well as in a prerecorded, human-voice message, played in the operator's head-set. These messages may be distracting if the advice is not beneficial to the operator. Despite having the option, *none* of the subjects turned off their agents in any of the examined conditions.

4.2. Search And Rescue

We further instantiate the k-MYAO heuristic (Section 3) for providing advice in a Search And Rescue (SAR) task.[6] We will first describe the SAR task, followed by our agent's design and evaluation. The simulation specification and code release are available in Appendix B.

4.2.1. The SAR Task

Search And Rescue (SAR) involves the exploration of disaster scenes while locating, identifying and rescuing victims. Rescue robots provide valuable assistance to rescue workers, as seen in past disaster environments such as the 2001 World Trade Center collapse [60], the 2004 Mid Niigata earthquake in Japan, the 2005 Hurricanes Katrina, Rita and Wilma in the United States [61] and the 2011 Tohoku earthquake and tsunami in Japan [62] (see [63] for a comprehensive review of the use of robotics in disaster environments and [8] for a more specific review on robotic SAR settings). We focus on a SAR setting in which a *single human operator* is required to search for certain objects[7] using a large team of unreliable robots. In our setting there are 10 robots. Namely, we focus on the search for objects rather than the rescue of objects. The human operator remotely supervises and controls the robots. The operator is also required to provide assistance to the robot in both abnormal situations as well as when a suspicious object

[6] A short video summarizing the work presented in this section is available at `https://www.youtube.com/watch?v=mSh67zb0Zm4`.

[7] The objects can be victims, flames, weapons, etc.

is encountered. For example, when a robot gets stuck or disoriented, the human operator can manually control it, and when a robot detects a suspected desired object the human operator can manually drive the robot to get a better view of the object and then prove whether or not the object is one that is desired by classifying it.

Similar to the warehouse operation task (Section 4.1.1), in state-of-the-art SAR applications, the robots' pathways, actions and the automatic identification of victims use sophisticated algorithms. However, the prioritization of the human operators' tasks has been left in each operator's hands.

For this task, a SAR simulation system was built in which 2 simulation environments were integrated. Unfortunately, we could not use a standard SAR simulator (such as the Robocup SAR simulator[8]) since we wanted to test the methodology using real robots in a physical environment. Therefore, we constructed a simulator to resemble the real deployment environment (see Appendix B for the system's specification and code release). Furthermore, existing simulators are aimed at specific missions, such as modeling fires and their expansion. To the best of our knowledge, these simulators are not used for search.

Our SAR task is conducted in two distinct environments:

Environment 1 – A real office building floor consisting of an "L" shaped, narrow corridor with small and mid-sized offices (See Figure 7). We will next evaluate Environment 1 in both simulation and *physical deployment*. Thus, we denote the simulation as Environment 1s and the physical deployment as Environment 1p. Figures 6 and 7 provide snapshots of Environment 1s and Environment 1p, respectively.

[8] `http://roborescue.sourceforge.net/blog/`

Figure 6: Simulated SAR Environment. The robots are represented in gray in the center of the screen, the black lines represent the structure walls and the red arrows and blue lines indicate the robots' planned trajectories. Note that this is *not* the user interface but only the simulation.

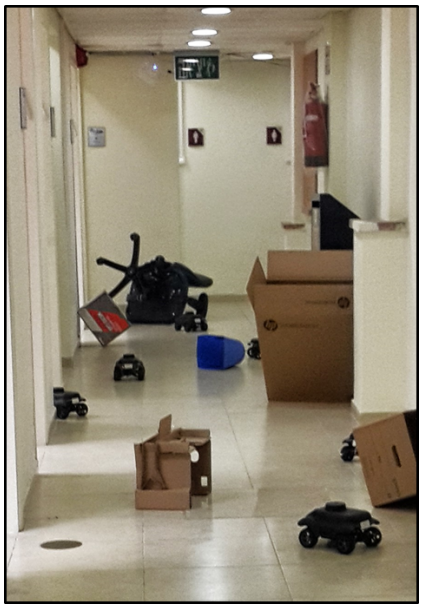

Figure 7: Enviorment 1p; one edge of the "L" shaped corridor of an office building floor (see Figure 9 for the operator's point of view of the environment).

Environment 2 – A simulated medium-sized factory yard taken from the pop-

ular CounterStrike© computer game called “Assault”. “Assault” is considered to be very realistic and is one of the most popular maps in the whole CounterStrike© series[9] (See Figure 8). Environment 2 was only evaluated in simulation and will be denoted as Environment 2s.

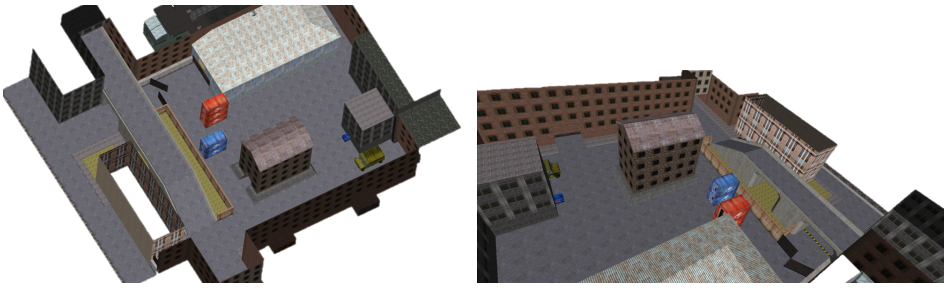

Figure 8: Environment 2s - an open terrain model from CounterStrike©.

In order to control and supervise the robots, we designed a GUI (Figure 9) that provides real-time feedback on the task and robots’ states. When a robot needs the operator’s attention, for example when it gets stuck or when it detects a suspected victim, it signals to the operator using both prerecorded voice messages and visual signals (see Appendix B for complete details on the SAR simulation and GUI).

[9] *http://counterstrike.wikia.com/wiki/Assault*

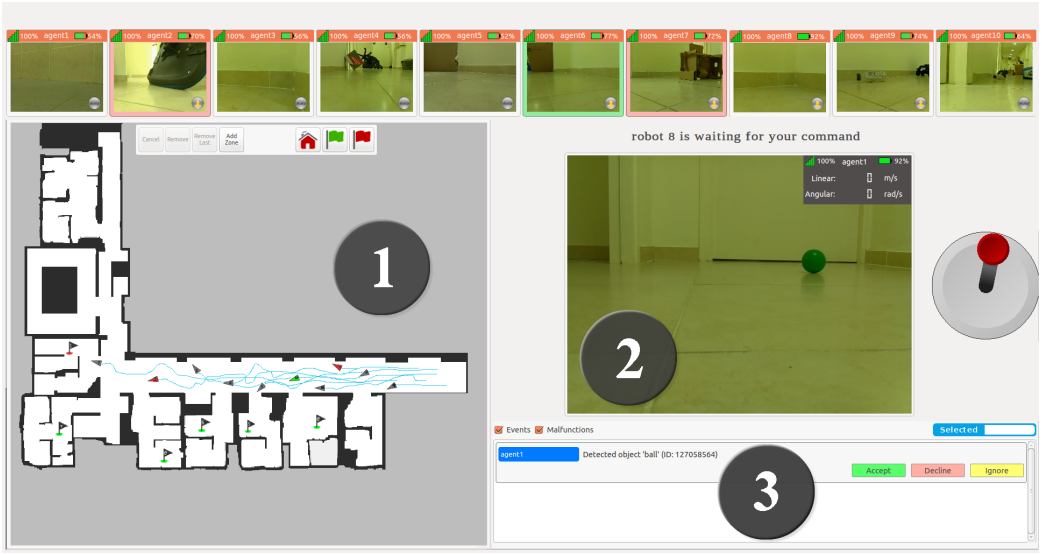

Figure 9: The graphical user interface used for the SAR task in this study. 2D map of the terrain including the robots' reported positions and their footprints is available in 1, an enlarged camera view of the robot of interest is available in 2 and the robots' requests are presented in 3. The agent's advice is presented above the camera view.

Note that the SAR task is remarkably different from the warehouse operation task in several aspects: (1) SAR task's termination time is usually unknown during task execution; (2) Unlike the fully stochastic process in which orders arrive at the warehouse system, the target objects in SAR are revealed by the robots themselves. Namely, the modeling of potential rewards is much more challenging; (3) The operator is likely to prefer rewards as early as possible in SAR as time is critical (e.g., detecting fire sources, injured victims); (4) Robots are more likely to face difficulties in completing their missions in SAR.

4.2.2. Training

We will first define the task's state space S. We wish to define S in a generic fashion to allow it to be used across different terrains (both in physical and simulated deployment as the ones described above). Therefore, we could not use terrain specific features such as the robots' positions, room sizes, the position of obstacles, etc. We represent $s \in S$ using the following set of state variables: the number of active robots ($Active(s)$), the number of detected objects ($Objects(s)$), the average Euclidean distance between robots ($Distance(s)$) and the minimal Euclidean distance between a pair of robots ($Minimal(s)$).

A is the action set that the operator can take. Recall that A is also the advice set from which the agent can propose advice. We define A as the instantiations of the following 7 action schemes: "Tele-operate robot i to a better position", "Tele-operate robot i away from robot j", "Send robot i home for repairs", "Robot i is stuck, try to get it loose", "Robot i detected an object", "Robot i is waiting for your command" and "Spread the robots around, they are too close to each other".[10]

Training $R^s(\cdot)$:

We define $R^s(s)$ as the expected time to find the next desired object in its environment. We model $R^s(s)$ using the following Equation:

$$R^s(s) = \alpha_0 - \alpha_1 \cdot ln(Active(s)) + \alpha_2 \cdot Objects(s)^2 - \\ \alpha_3 \cdot Objects(s) - \alpha_4 \cdot Distance(s) - \alpha_5 \cdot Minimal(s) \quad (4)$$

Where $\alpha_0, \ldots, \alpha_5$ are learned using linear least squares regression.

This form of function assumes that there is α_0 time to find a desired object, which is then reduced by the average distance between robots and the minimal distance between a pair of robots in a linear way. It also assumes that the number of active robots reduces the expected time in a marginally decreasing fashion. The objects detected so far increase the expected time (as objects are harder to find), yet, in the early stages of the task, finding the first objects is assumed to indicate that the robots have achieved some effectiveness in terms of position. This form of function was compared to more than 100 other forms. Some of the other functions that were tested included one or more of the following modifications to the above function: the use of $Active(s)$ or $Objects(s)$ as an additive variable; $Active(s)$ as having a multiplicative impact or $Objects(s)$ as having an impact depending on $Distance(s)$ or $Minimal(s)$. These other forms yielded a lower fit to the data we collected in both simulated environments – Environment 1s and Environment 2s (as described next). All parameters are assumed to be positive, and in fact reached positive values.

[10]There are 141 instantiations of the 7 action schemes: "Tele-operate robot i to a better position" (10 options), "Tele-operate robot i away from robot j" (90 options), "Send robot i home for repairs" (10 options),"Robot i is stuck, try to get it loose" (10 options), "Robot i detected an object" (10 options),"Robot i is waiting for your command" (10 options) and "Spread the robots around, they are too close to each other" (1 option) where $i, j \in [1, \ldots, 10]$.

In order to collect data to learn $R^s(\cdot)$, which can be very expensive as an operator is needed to support them, we suggest using a simulated *utopic environment*. In a utopic environment, there will be no need for human interference with the robots' work, as the robots will never malfunction. To simulate an ideal environment, we use non-malfunctioning robots in virtual simulation. Accordingly, we define $R^s(\cdot)$ to be the expected time to find the next object in the *utopic environment*. That is, we wish to learn from a *simulated utopic environment* and utilize the gathered data to learn an advising policy which is tested in both physical and simulated environments which are far from utopian (Section 4.2). Hence, in order to learn $R^s(\cdot)$, we ran 150 1-hour sessions in each environment. During these sessions, the robots autonomously searched for 40 randomly placed green objects (the green objects represent the desired objects in our SAR settings). In each session, we placed random obstacles and used a different number of robots ranging from 1 to 10. To avoid the need for a human operator, we set an empty malfunction schedule and instructed the robots to avoid waiting for an operator's response about detected objects.

Each recorded session was translated into vectors, each representing 1-second during the session. That is, every recorded second was translated into a vector consisting of the representation of the state experienced in that second s, and labeled with the time it took to find the next desired object in that session. Specifically, from state s, we extract $Active(s), Objects(s), Distance(s)$ and $Minimal(s)$.

Training $C^s(s,a)$ and $P^s(s,a,s',C(s,a))$:

We define $C^s(s,a)$ as the expected time for completing a by an average human operator and $P^s(s,a,s',C(s,a))$ as the probability for transitioning from s to s' using advice a, given $C^s(s,a)$. To train $C^s(s,a)$ and $P^s(s,a,s',C(s,a))$, we recruited 30 Computer Science undergraduate students, ranging in age from 18 to 34 (mean=25, s.d.=3.4) with similar demographics to participate in a simulated SAR task equipped with a *naïve* advising agent. Given state s, our naïve agent provided advice a which, upon completion, is expected to bring about a new state s', where $R^s(s') + 5sec < R^s(s)$. That is, the agent suggests advice that, if completed successfully, will reduce (at least) 5 seconds from the expected time to find the next object. Similar to the naïve agent in the warehouse operation task (Section 4.1.1), the agent only offers advice which can be completed and only if the human operator is not currently engaged in the suggested task (for example, if a robot has malfunctioned the agent will not advise tele-operating it, but it might advise fixing the robot). The agent presents the advice in both textual format (see Figure 2) as well as a prerecorded, human-voice message played in the operator's head-set. Note that the agent *filters and prioritizes* robots' requests and messages,

yet in a naïve way. For motivation, subjects were paid 1 NIS (about 25 cents) per green object they found, and 0.5 NIS per advice to which they adhere.

Similar to the training in the warehouse operation task (Section 4.1.2), all of the advice that our agents gave was executed and completed. Therefore, we set $P^s(s, a, s', C(s, a))$ to be 1 whenever completing a advice is possible, and 0 otherwise. Furthermore, when analyzing the operators' behavior, we were unable to find good *cross-subject* features that will help us predict the expected time it would take to complete advice. Therefore, we used the mean execution time. That is, $C^s(s, a)$ is equal to the average execution time of a across our subjects.

Estimating $Int^s(s, a, s', C(s, a))$:

Due to the short time frames in which pieces of advice are executed (SAR is a fast-paced task)

Setting γ:

In search and rescue settings, we assume that the termination time of the task is unknown. Therefore we define $\mathbb{T}$ to be $[0, \infty]$. We further assume that the operator is likely to prefer rewards as early as possible in SAR settings as time is critical. Therefore, γ is defined as the probability for the task to end, given the task's expected duration. We assume a geometrical distribution, so γ is set to 1-(the *calculation interval* divided by the *expected task duration*).[11] The calculation interval was set to 1 second in our experiments.

Overall, 150 hours of simulations were used to train $R^s(\cdot)$ and 30 human subjects were recruited for training $C^s(s, a)$ and $P^s(s, a, s', C(s, a))$.

4.2.3. Experimental Design

In preliminary testing, we noticed that the *1-min-MYAO* heuristic returns almost the exact same advice as *2-min-MYAO* and *3-min-MYAO* in our SAR settings in both Environment 1s and 2s (in more than 95% of the cases we examined). This finding made the myopic search of depth larger than 1 unnecessary (a discussion of this phenomena is provided in Section 5). Therefore, we design and evaluate a single agent named *S-AID* to solve the *1-min-MYAO* using a minimization format of Algorithm 1. The agent provides the best advice per its optimization problem in both textual format (see Figure 2) as well as in a prerecorded, human-voice message. The presented advice might change at time t if the advice was successfully completed or if more urgent and productive advice is available according to

[11] The geometric distribution was chosen to allow subjects to easily understand the nature of the task. Naturally, other distributions may be considered as well.

the current state s_t, making our agent *adaptive* to the environment. We used an efficient priority queue and updating methods to avoid long calculations.

Unlike in the warehouse operation task evaluation (Section 4.1.3), we did not encounter a significant improvement in operators' performance when performing the task for a second time in our preliminary testings. Therefore, in order to evaluate our agent we used a *within-subject* experimental design where each subject performed the SAR task twice (a week apart); once without the agent and once equipped with the agent. We recruited 32 subjects to participate in the simulated SAR task (16 per simulated environment (Environments 1s and 2s)) and another 12 subjects participated in Environment 1p, totaling 44 subjects. Subjects who participated in the simulated task were BSc and MSc Computer Science students, ranging in age between 20 and 33 (mean 25), 16 females and 28 males, whereas the 12 participants in Environment 1p were researchers and workers from our university, ranging in age between 20 and 38 (mean 27), 8 males and 4 females, who did not take part in warehouse operation experiments. Subjects were counterbalanced as to which condition was applied first.

Each subject was trained before each run; they underwent a structured 1-on-1 hands-on tutorial on the system by our research assistant and had to pass a short test. The test was conducted to make sure the operator was capable of successfully completing the task. During the test, a subject was in control of 3 robots and without any time limit had to find and classify 2 objects in the terrain while s/he encountered 3 simulated malfunctions.

The SAR task took 40 minutes (in simulation) and 15 minutes (in physical deployment) in which the subjects had to search for the green objects placed in predefined (yet, randomly selected) positions in the environment. Physical deployment was set to 15 minutes due to the ample effort in preparing and overseeing such physical development trials. For example, each robot had to be charged, activated and placed prior to each trial, a highly time-consuming process was needed to place the obstacles and objects prior to each trial and their removal immediately afterwards to allow workers of the office floor to move around freely, etc. In all tested environments, subjects were motivated to find and classify as many green objects as possible using a small monetary reward (1 NIS per object). However, they did not receive any monetary reward for accepting the agent's advice. In Environment 1s (simulated office building), we placed 40 green objects around the office floor. We set a malfunction schedule such that each robot would encounter 1.5 malfunctions (on average) during the simulation. All robots started from the main entrance of the floor. In Environment 1p (real deployment in an office building), we scattered 20 green objects (see Figure 7). We set a malfunction schedule

such that every robot would encounter 0.8 malfunctions (on average) during the deployment. Half of the robots started from the main entrance of the office space, whereas the other half started from the back entrance. In Environment 2s (simulated open terrain), we scattered 40 green objects. We set a malfunction schedule such that every robot encounters 2 malfunctions (on average) during the simulation. All robots started from the Assault deployment point (next to the Humvee in Figure 8). After completing the task, subjects were asked to answer a standard NASA-TLX questionnaire.

Subjects were informed that they could turn off the advising agent if and whenever they choose.

4.2.4. Results

Results reported in this section were found to be significant using the Wilcoxon Signed-Rank Test [64] (an alternative to paired-sample t-test for dependent samples when the population cannot be assumed to be normally distributed).

In **Environment 1s**, the results show a statistically significant increase in the average number of detected objects by the robots (37.7 vs. 31.2 objects, $p < 0.001$) as well as their average covered terrain (529 vs. 462 square meters, $p < 0.05$) for the condition in which the agent was present. Furthermore, a decrease in the average time that robots stay idle (1540 vs. 1860 seconds, $p < 0.05$) and a reduced average workload (55 vs. 62 TLX, $p < 0.1$) were also recorded.

In **Environment 1p**, a major **100%** increase in the average number of detected objects was recorded (14.1 vs. 7 objects, $p < 0.001$) as well as an increased average for covered terrain (462 vs. 305 square meters, $p < 0.05$). A significant decrease in the average time that robots stay idle (2720 vs. 3244 seconds,$p < 0.05$) and a reduced average workload (55 vs. 62 TLX, $p < 0.1$) were also recorded.

Similarly, in **Environment 2s**, the results show an increased average number of detected objects by the robots (34.1 vs. 30.1 objects, $p < 0.001$) as well as their average covered terrain (1144 vs. 845 square meters, $p < 0.05$) for the condition in which the agent was present. Furthermore, a significant decrease in the average time that robots stay idle (1053 vs. 1455 seconds, $p < 0.01$) and a reduced average workload (57 vs. 61 TLX, *not statistically significant*) were also recorded.

See Figure 10 for a graphical summary.

Overall, more than 95% of the advice was followed by the subjects without a distinct pattern with respect to time.

All but 4 of the 44 subjects who participated across the 3 environments showed an increased number of detected objects while equipped with our agent.

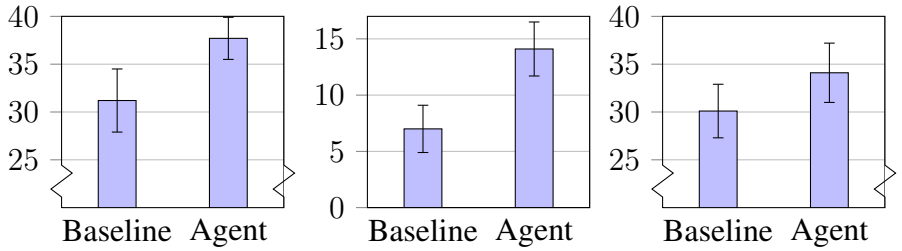

Figure 10: Average number of detected objects across advising conditions and environments. From left to right – Environment 1s, Environment 1p and Environment 2s. Error bars indicate standard errors.

Recall that the agent presents the advice in both textual format as well as in a prerecorded, human-voice message, played in the operator's head-set. These messages may be distracting if the advice is not beneficial to the operator. Despite having the option, *none* of the subjects turned off the agent.

Despite major differences between our simulated, utopian training environment and the test environments, our approach yields solid advising policies used in both non-utopian simulated environments and in physical deployment of robots.

5. Discussion

The proposed *k-MYAO* heuristic, which implements a k-steps-lookahead search, was evaluated in both the warehouse and SAR tasks. In the warehouse operation task, using a subject group of Computer Science students, the results show that the *2-MYAO* heuristic, deployed by the *W-AID2* agent, significantly outperforms the *1-MYAO* heuristic deployed by the *W-AID1* agent, which in turn significantly enhanced human operator's performance compared to the baseline condition of no advising agent. Using an additional group of students, this time we evaluate the *3-MYAO* heuristic, deployed by the *W-AID3* agent, which significantly outperforms the *2-MYAO* heuristic deployed by the *W-AID2* agent. In the SAR task, the results show that the *2-MYAO* heuristic deployed by the *S-AID* significantly enhanced human operator's performance compared to the baseline condition of no advising agent.

Interestingly, unlike the warehouse operation task, the *1-MYAO* and *2-MYAO* heuristics return almost the exact same advice in the SAR task. We believe that this phenomena is attributed to the major differences between the two tasks as specified in Section 4.2.1. Particularly, we believe that the high uncertainty regarding future rewards and the high likelihood of robot malfunctions create a large

incentive to prefer short-term rewards over long-term ones, which makes myopic search deeper than 1 unnecessary.

Note that in our environments we were unable to run a 4-steps-lookahead heuristic in real-time due to the large search space despite using a state-of-the-art PC with 2 CPUs, each with 6 cores, each core operating at 2.8 GHz. Therefore, we conjecture that in environments with a small number of possible actions (small $|A|$) which are strongly connected, our k-steps-lookahead approach will outperform simple 1-step-lookahead solutions. Namely, given a small number of actions, our approach scales better with k. Furthermore, given that the actions are interconnected and their order of execution can significantly effect the task's performance, there is a larger margin for improvement by deeper search methods which are deployed by k-steps-lookahead agents. Moreover, we believe that in complex environments with relatively low uncertainty, such as our warehouse operation task, k-steps-lookahead agents where $k > 1$ can enhance operators' performance significantly more than a simple 1-step-lookahead heuristic as the planning is more robust. However, additional factors such as the subjects' likelihood to accept different pieces of advice and the time-depth tradeoff have to be considered.

The acceptance rate of advice suggested by *W-AID2* was significantly lower than for the advice suggested by *W-AID1* for the group of Computer Science students in the warehouse operation task. Human operators, like all people, are limited in their capacity to process and plan actions over long sequences [36] and have been shown to use more intuitive mechanisms for problem solving over performing lookahead searches [65]. Therefore, we conjecture that, in some cases, subjects may not comprehend the reason for the suggested advice and therefore did not follow it. This property is disadvantageous as people are more likely to adhere to an agent's advice which they can understand [46, 47]. Nevertheless, this hypothesis is devalued given the results of the non-Computer-Science subjects we recruited for evaluating the *W-AID2* and *W-AID3* in the warehouse operation task. The results show that in almost all cases, subjects fully adhered to the provided advice. Further study is needed to determine whether the technical abilities of the operator should require special attention in the advice provision process.

The results from both domains show that operators rarely follow advice 100% of the time. For actual deployment, the operator model can be updated to reflect these findings. However, a deeper investigation is needed to resolve the question of *why* a certain piece of advice was accepted in a given setting and time and another was not accepted by an operator. The above is an open challenge at the moment.

In the SAR task, we used a simulated environment to train the different models which are deployed in physical settings. This approach was already shown to be beneficial in learning robot control policies in the past [66]. Similarly, the reward model $R(\cdot)$ was trained in utopic environments without human intervention. These allowed for a faster and more economically efficient training process. Note that despite the inherent inaccuracy that this approach induces, it results in a solid advice provision agent that significantly enhances operators' performance. For future work, one can consider deploying the grounded simulation learning approach which provides a more accurate way to transfer knowledge from simulation to the real-world settings [67].

Interestingly, in the SAR task, the most significant difference was found in physical deployment settings. This result corresponds with previous results which show that people behave differently in simulation and real deployment [66]. We conjecture that in physical deployment settings subjects felt more stressed and pressured and, as a result, the advising agent had a larger margin for potential improvement. For example, some subjects mentioned that they were afraid to cause damage to the robots which is of course irrelevant in simulation. We find support for this hypothesis in the fact that subjects engaged in the SAR task in a physical deployment setting for less than half of the time they engaged in the task in a simulation setting (15 minutes compared to 40 minutes, respectively) yet reported the same average TLX scores for both conditions (55 for the agent condition and 62 for the baseline condition).

For most tasks, including the warehouse operation and SAR tasks, the reward model has to be learned. As there are presumably an infinite number of functional forms, we used the following basic procedure to select the functional form: each state feature was tested by itself and a single variable function was fitted to capture the observed influence of that feature on the task's reward. Then, we performed an exhaustive search over all possible additive and multiplicative variations of the single variable functions identified earlier. The function which provided the best fit for the collected data was chosen.

6. Conclusions

In this work, we presented a new approach for the enhancement of operator performance in multi-robot environments. Our extensive empirical study, with over 150 human subjects in the warehouse operation and search and rescue environments, both in simulation and real deployment, shows that intelligent agents are able to significantly enhance the performance of operators in complex multi-

robot environments. Our methodology can accommodate future advancements in robotics hardware and algorithms and is not restricted to a certain type or quantity of robots in the environment.

Despite enduring high uncertainty and noisy signals, operators manage to take advantage of the agents' advice and translate them into a better performance than their baseline scores, demonstrating the benefit of our agents and approach.

We conclude that the use of automated advising agents in robotics, and especially in multi-robot environments, is essential in bringing about better performance in real-world applications and enabling operators to control a larger number of robots simultaneously.

We intend to expand our proposed methodology and use the insights provided in this article to design and implement repeated-interaction agents. These agents could learn from previous interactions with the operator and tailor an advising policy for her. As part of this work we will examine the operator modeling for multiple interactions and the ability to deduce insights from one environment to another. Proceeding on a different path, we are currently working on adapting our methodology to help operators in more complex environments where robots engage in different tasks simultaneously. For example, a part of the robot team is performing a patrolling task while the other performs a SAR task. One interesting aspect, in this case, is to allow the human operator to indicate her preferences on trade-offs between the tasks, allowing the agent to reason over a number of tasks simultaneously and thus make its recommendations more suitable for the operator.

Acknowledgment

This article extends our previous reports [50, 68] in two major aspects: First, we extended the simple MYAO heuristic suggested in [50], which considered only a 1-step-lookahead search heuristic, to account for a k-steps-lookahead search. This allowed us to investigate a deeper myopic search heuristic that was found to be beneficial in our warehouse operation task. Second, by extending our preliminary investigation [50] with the warehouse operation task as presented in [68], including adding more than 100 subjects (more than doubling our subject pool from), we were able to enhance the credibility and validity of our previously reported results and generalize the benefits of our proposed methodology over two domains.

Our video entitled "Intelligent Agent Supporting Human-Multi-Robot Team

Collaboration"[12] was presented at the International Joint Conference of Artificial Intelligence (IJCAI) 2015. The video summarizes our paper [50], which is discussed in Section 4.2 of this article.

Appendix A. Warehouse Simulation

In this article, the warehouse operation task is conducted in a simulated warehouse which we built using the Gazebo robot simulation toolbox[13] in C++. The warehouse consists of 10 mobile ground robots capable of transporting shelves from one place to another across the warehouse floor. There are 52 shelves in the warehouse, each consisting of between 1 and 3 different products, which the robots can transport to 8 packing stations. At a packing station, the human worker can unload products from the shelves that have arrived.

In our simulated warehouse (which acts as a fulfillment center), when an order (a customer's requested set of products) arrives at the warehouse system, robots are sent automatically to move the relevant shelves to the packaging stations. Each task is assigned to the closest available robot. When a robot arrives at a packaging station, it is the human worker's job to remove the relevant merchandise from the shelf. In our simulation, we off-load the merchandise using each product's code as it appears on the code sheet provided to the human worker. This is meant to simulate the scanning of a product's bar code in a physical warehouse. Once the required products are unloaded from the shelf, the worker can send the robot off to move the shelf back to its original position and to continue its work. When an order has been filled, namely all requested products have been unloaded, the human worker can pack and complete the order. This is done by pressing the "package" button which appears after all of the necessary merchandise has been off-loaded. The code that is received during the packaging stage once again simulates the scanning process, this time the scanning of the address for delivery. The code must be copied into an appropriate Excel file. This process is carried out in order to make sure that the operator pays sufficient attention to her tasks. An order is removed from the system once it is packed.

In addition, robots may malfunction. For example, during the robots' work, products may fall in the work area, potentially causing the robots to get stuck. Once a product has fallen on the warehouse floor, the human worker can alert the robots so that they will avoid the problematic spot or s/he can remove the

[12] Available at `http://www.youtube.com/watch?v=mSh67zb0Zm4`.

[13] `www.gazebosim.com`

fallen product altogether. If the human worker decides to keep the robots from going through the problematic spot, then the robots will work out new, and possibly longer, paths. However, if the human worker decides to remove the product that fell on the warehouse floor, the robots will be restricted from approaching the area in order to avoid any danger to the human worker. The human worker decides when and where to restrict the robots' movements. Namely, a spot can be restricted even if no obstacle is present (e.g., keep it clear for future use) and a spot may be unrestricted even if an obstacle is present (e.g., "taking a chance" that the robot will not collide with the obstacle). The command to restrict (unrestrict) a spot in the warehouse is executed instantaneously once given by the human worker. Note that by restricting the robots' movements, the completion of the robots' tasks may be impaired. In this work, we differ between two types of obstacles or restricted cells: *critical*, which prevents the completion of a task (e.g., if it blocks a robot's only path to a packing station) and *non-critical*. The robots can also malfunction regardless of fallen products. For example, a technical problem could cause one of the robots to deviate from its course. In such a situation, the human worker will need to manually drive or guide the robot to its destination.

The human operator is provided with a GUI (Figure 2) that provides real-time feedback on the task's and robots' states. When a new order arrives, it is automatically added to the active orders area, which presents all unpacked orders. The robots' and shelves' positions are marked on a 2D grid-map of the warehouse floor along with the robots' planned paths. An enlarged camera view of the robot of interest is available as well as a command panel for controlling that robot manually. In order to set interest on a specific robot, the operator can click its thumbnail within the thumbnails area or on its location on the grid map. When a robot needs the operator's attention, its graphical representation blinks for 5 seconds and the interface plays a prerecorded, human-voice message in the operator's headset. The operator can view all of the robots' requests and provide solutions using the alarm log. For the operator's convenience, we maintain a first-come-first-serve request log indicating the (active) robot requests.

Overall, the operator can perform the following actions: (1) remove items from shelves at the packing stations; (2) pack a consumer's order when filled; (3) toggle the robots' status between manual and autonomous; (4) alert the robots to avoid a problematic spot (either critical or not) on the warehouse floor; (5) remove fallen objects from the warehouse floor; and (6) manually tele-operate a robot using the joystick widget or the keyboard arrow keys.

Our simulation and agents are available at `http://www.biu-ai.com/`

roboticAdvice/ for future research.

Appendix B. The SAR task specification

In the scope of our simulations (Environments 1s and 2s), we used the Gazebo robot simulation toolbox. To create Environment 1s, we manually mapped an office building floor (which is our lab floor at Bar-Ilan University, acting as Environment 1p – Figure 7) using a 30-meter laser and constructed a 3D virtual model to scale. To create Environment 2s, we obtained a 3D model of "Assault" terrain from the CounterStrike source kit (see Figure 8). These models were mounted to the Gazebo simulation alongside 1–10 robot models of type Hamster[14] which use the Robot Operating System (ROS)[15]. *Hamster* is an autonomous unmanned ground vehicle (AUGV) at the off-the-shelf price of $1600 US[16] per robot (See Figure B.11). It is a 4WD rugged platform with a built-in navigation algorithm that allows it to explore, map and localize in unknown areas. Hamster has 2 onboard Raspberry PI Linux servers for algorithm execution and an Arduino for low-level control. Hamster is mounted with an HD camera with h264 video streaming over WiFi and a 360° 6-meter range LIDAR laser. Each Hamster is 190mm in width, 240mm in length and 150mm in height.

Figure B.11: Hamster AUGV; one of the 10 identical robots used in this study.

The Hamster can be either in autonomous mode or manual mode. While in autonomous mode, the Hamster travels through the terrain without the operator's

[14] http://wiki.ros.org/Robots/Hamster
[15] http://www.ros.org/
[16] 2014 pricing.

intervention. In our task, the robots are required to explore the environment, given a 2D blueprint of the area (no mapping is required). However, the blueprint does not include randomly placed objects and obstacles scattered in the environment, for example bookstands and cupboards in Environment 1 and containers and barrels in Environment 2. The robots are given their initial position (deployment point) and localize using ICP laser matching [69] and the AMCL algorithm[17]. We used C++ for the implementation.

The robots execute a simple exploration algorithm, based on the given map. Given an estimation of the location of all robots in the team, a robot simply chooses to travel away from its peers, to a less crowded area (thus, a less explored area). The robot travels to its desired destination using the A^* path planning algorithm [70], and uses basic obstacle avoidance techniques to avoid both obstacles and other robots while doing so.

We account for three malfunction types that the Hamster can experience:

Sensing The camera/laser stops working. In such cases, the operator can send the robot back to home base, where the experiment operator can fix it.

Stuck The robot cannot move (for example due to an obstacle), and needs the operator to help it get loose. In some cases, the Stuck malfunction is terminal.

WiFi The robot stops sending and receiving data and disappears from the map. In such cases, the operator can mark the area where the robot was last seen as a "no entrance" zone. Upon losing the WiFi signal, the robot is programmed to return to its last point of communication.

In our simulations, the robots experience malfunctions according to a predefined malfunction schedule which determines when and which malfunctions will occur to each robot.

The GUI (see Figure 2) provides the operator with on-line feedback from the cameras mounted on the robots (Thumbnail area), the 2D map of the terrain including the robots' reported positions and their footprints (area 1), an enlarged camera view of the robot of interest (area 2), an action control bar, and a joystick widget. The action bar's commands and joystick functions are also available using keyboard and mouse shortcuts inspired by strategic computer games. For example, in order to set interest on a specific robot, the operator could click its

[17] *http://wiki.ros.org/amcl*

thumbnail camera or location on the map or could click on its number on the keyboard. Double clicking will center the map on the robot's location.

The operator can perform the following actions: (1) change the mode for each of the robots (autonomous or manual); (2) send a robot to a desired point on the map using the mouse's right click option (the robot would autonomously navigate to the destination point, when possible); and (3) manually tele-operate a robot using the joystick widget or the keyboard arrow keys. When a robot needs the operator's attention—in case of malfunctions or a detected green object that needs classification—it changes its mode to manual and signals the operator by blinking for 5 seconds and playing a prerecorded, human-voice message in the operator's headset. The operator can view the robots' requests and provide solutions using the alarm log (area 3). For the operator's convenience, we maintain a first-come-first-serve alarm log indicating the (active) alarms.

Our simulation and agents are available at `http://www.biu-ai.com/roboticAdvice/` for future research.

[1] J. Saez-Pons, L. Alboul, J. Penders, L. Nomdedeu, Multi-robot team formation control in the guardians project, Industrial Robot: An International Journal 37 (4) (2010) 372–383.

[2] P. Dasgupta, J. Baca, K. Guruprasad, A. Muñoz-Meléndez, J. Jumadinova, The comrade system for multirobot autonomous landmine detection in post-conflict regions, Journal of Robotics 2015.

[3] J. Kaiman, At japan's fukushima nuclear complex, robots aiding the cleanup after 2011 disaster, [Online; posted 10-March-2016] (March 2016).
URL `http://www.latimes.com/world/asia/la-fg-japan-fukushima-robots-20160310-story.html/`

[4] F. A. Auat Cheein, R. Carelli, Agricultural robotics: Unmanned robotic service units in agricultural tasks, Industrial Electronics Magazine, IEEE 7 (3) (2013) 48–58.

[5] H. Ardiny, S. J. Witwicki, F. Mondada, Are autonomous mobile robots able to take over construction? a review, International Journal of Robotics 4 (EPFL-ARTICLE-217004) (2015) 10–21.

[6] I. S. Kulkarni, D. Pompili, Task allocation for networked autonomous underwater vehicles in critical missions, Selected Areas in Communications 28 (5) (2010) 716–727.

[7] E. Guizzo, Three engineers, hundreds of robots, one warehouse, IEEE spectrum 7 (45) (2008) 26–34.

[8] Y. Liu, G. Nejat, Robotic urban search and rescue: A survey from the control perspective, Journal of Intelligent & Robotic Systems 72 (2) (2013) 147–165.

[9] C. Y. Wong, G. Seet, S. K. Sim, Multiple-robot systems for usar: key design attributes and deployment issues, International Journal of Advanced Robotic Systems 8 (1) (2011) 85–101.

[10] M. A. Goodrich, M. Quigley, K. Cosenzo, Task switching and multi-robot teams, in: Multi-Robot Systems. From Swarms to Intelligent Automata Volume III, Springer, 2005, pp. 185–195.

[11] J. Chen, P. Terrence, Effects of imperfect automation and individual differences on concurrent performance of military and robotics tasks in a simulated multitasking environment, Ergonomics 52 (8) (2009) 907–920.

[12] P. Squire, R. Parasuraman, Effects of automation and task load on task switching during human supervision of multiple semi-autonomous robots in a dynamic environment, Ergonomics 53 (8) (2010) 951–961.

[13] J. Y. Chen, M. J. Barnes, Human–agent teaming for multirobot control: a review of human factors issues, IEEE Transactions on Human-Machine Systems 44 (1) (2014) 13–29.

[14] S. Chen, T. Baarslag, D. Zhao, J. Chen, L. Shen, A polynomial time optimal algorithm for robot-human search under uncertainty, in: Proceedings of IJCAI International Conference on Artificial Intelligence, 2016.

[15] D. R. Olsen Jr, S. B. Wood, Fan-out: measuring human control of multiple robots, in: Proceedings of the SIGCHI conference on Human factors in computing systems, ACM, 2004, pp. 231–238.

[16] H. Wang, M. Lewis, P. Velagapudi, P. Scerri, K. Sycara, How search and its subtasks scale in n robots, in: Proceedings of the ACM/IEEE International Conference on Human-robot interaction, ACM, 2009, pp. 141–148.

[17] T. B. Sheridan, Human–robot interaction status and challenges, Human Factors: The Journal of the Human Factors and Ergonomics Society 58 (4) (2016) 525–532.

[18] S.-Y. Chien, M. Lewis, S. Mehrotra, K. Sycara, Imperfect automation in scheduling operator attention on control of multi-robots, in: Proceedings of the Human Factors and Ergonomics Society Annual Meeting, Vol. 57, SAGE Publications, 2013, pp. 1169–1173.

[19] P. Velagapudi, P. Scerri, Scaling human-robot systems, in: CHI, 2009.

[20] H. Wang, M. Lewis, S.-H. Chien, P. Velagapudi, Scaling effects for synchronous vs. asynchronous video in multi-robot search, in: Proceedings of the Human Factors and Ergonomics Society Annual Meeting, Vol. 53, SAGE Publications, 2009, pp. 364–368.

[21] J. Y. Chen, M. J. Barnes, Z. Qu, Roboleader: An agent for supervisory control of multiple robots, in: Proceedings of the ACM/IEEE International Conference on Human-robot interaction, IEEE Press, 2010, pp. 81–82.

[22] J. Y. Chen, M. J. Barnes, C. Kenny, Effects of unreliable automation and individual differences on supervisory control of multiple ground robots, in: Proceedings of the 6th International Conference on Human-robot interaction, ACM, 2011, pp. 371–378.

[23] M. Cummings, S. Bruni, S. Mercier, P. Mitchell, Automation architecture for single operator, multiple uav command and control, Tech. rep., DTIC Document (2007).

[24] S. D. Ramchurn, J. E. Fischer, Y. Ikuno, F. Wu, J. Flann, A. Waldock, A study of human-agent collaboration for multi-uav, in: Proceedings of IJCAI International Conference on Artificial Intelligence, 2015, pp. 1184–1192.

[25] S. Rosenthal, M. Veloso, Using symbiotic relationships with humans to help robots overcome limitations, in: Workshop for Collaborative Human/AI Control for Interactive Experiences, 2010.

[26] S. Rosenthal, J. Biswas, M. Veloso, An effective personal mobile robot agent through symbiotic human-robot interaction, in: Proceedings of AAMAS conference, International Foundation for Autonomous Agents and Multiagent Systems, 2010, pp. 915–922.

[27] M. Micire, E. McCann, M. Desai, K. M. Tsui, A. Norton, H. A. Yanco, Hand and finger registration for multi-touch joysticks on software-based operator

control units, in: Proceedings of the TePRA/IEEE International Conference, IEEE, 2011, pp. 88–93.

[28] A. Stoica, T. Theodoridis, H. Hu, K. McDonald-Maier, D. F. Barrero, Towards human-friendly efficient control of multi-robot teams, in: Proceedings of CTS conference, IEEE, 2013, pp. 226–231.

[29] S. Pourmehr, V. M. Monajjemi, S. A. Sadat, F. Zhan, J. Wawerla, G. Mori, R. Vaughan, You are green: a touch-to-name interaction in an integrated multi-modal multi-robot hri system, in: Proceedings of the ACM/IEEE International Conference on Human-robot interaction, ACM, 2014, pp. 266–267.

[30] A. S. Clair, M. Mataric, How robot verbal feedback can improve team performance in human-robot task collaborations, in: Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction, ACM, 2015, pp. 213–220.

[31] A. T. Özgelen, E. I. Sklar, An approach to supervisory control of multi-robot teams in dynamic domains, in: Proceedings of TAROS conference, Springer, 2015, pp. 198–203.

[32] D. Sieber, S. Music, S. Hirche, Multi-robot manipulation controlled by a human with haptic feedback, in: Intelligent Robots and Systems (IROS), RSJ/IEEE Conference, IEEE, 2015, pp. 2440–2446.

[33] G. Bevacqua, J. Cacace, A. Finzi, V. Lippiello, Mixed-initiative planning and execution for multiple drones in search and rescue missions, in: Proceedings of ICAPS conference, 2015, pp. 315–323.

[34] J. Cacace, A. Finzi, V. Lippiello, Multimodal interaction with multiple co-located drones in search and rescue missions, arXiv preprint arXiv:1605.07316.

[35] K. Kurowski, O. von Stryk, Online interaction of a human supervisor with multi-robot task allocation, in: Proceedings of IAS conference, Springer, 2016, pp. 965–978.

[36] G. A. Miller, The magical number seven, plus or minus two: some limits on our capacity for processing information, Psychological review 63 (2) (1956) 81.

[37] N. Harvey, I. Fischer, Taking advice: Accepting help, improving judgment, and sharing responsibility, Organizational Behavior and Human Decision Processes 70 (2) (1997) 117–133.

[38] I. Yaniv, Receiving other peoples advice: Influence and benefit, Organizational Behavior and Human Decision Processes 93 (1) (2004) 1–13.

[39] G. E. Schrah, R. S. Dalal, J. A. Sniezek, No decision-maker is an island: integrating expert advice with information acquisition, Journal of Behavioral Decision Making 19 (1) (2006) 43–60.

[40] A. Schmidt, Implicit human computer interaction through context, Personal technologies 4 (2-3) (2000) 191–199.

[41] A. Azaria, A. Rosenfeld, S. Kraus, C. V. Goldman, O. Tsimhoni, Advice provision for energy saving in automobile climate-control system, AI Magazine 36 (3) (2015) 61–72.

[42] A. Rosenfeld, A. Azaria, S. Kraus, C. V. Goldman, O. Tsimhoni, Adaptive advice in automobile climate control systems, in: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015, 2015, pp. 543–551.

[43] A. Rosenfeld, S. Kraus, Providing arguments in discussions on the basis of the prediction of human argumentative behavior, ACM Transactions on Interactive Intelligent Systems 6 (4) (2016) 30:1–30:33.

[44] A. Rosenfeld, S. Kraus, Strategical argumentative agent for human persuasion: A preliminary report, in: Proceedings of the Workshop on Frontiers and Connections between Argumentation Theory and Natural Language Processing, 2016.

[45] A. Azaria, Y. Gal, S. Kraus, C. V. Goldman, Strategic advice provision in repeated human-agent interactions, Proceedings of AAMAS conference (2015) 1–26.

[46] A. Elmalech, D. Sarne, A. Rosenfeld, E. S. Erez, When suboptimal rules.

[47] P. Levy, D. Sarne, Intelligent advice provisioning for repeated interaction, in: Proceedings of AAAI Conference on Artificial Intelligence, 2016.

[48] Acm transactions on interactive intelligent systems (tiis) (2016).

[49] A. Rosenfeld, Automated agents for advice provision, in: Proceedings of AAAI Conference on Artificial Intelligence, 2015, pp. 4391–4392.

[50] A. Rosenfeld, N. Agmon, A. A. Maksimov, Oleg, S. Kraus, Intelligent agent supporting human-multi-robot team collaboration, in: Proceedings of IJCAI International Conference on Artificial Intelligence, 2015.

[51] J. Pearl, Heuristics: intelligent search strategies for computer problem solving.

[52] S. Fatima, S. Kraus, M. Wooldridge, Principles of Automated Negotiation, Cambridge University Press, 2014.

[53] O. Hansson, A. Mayer, The optimality of satisficing solutions, CoRR.

[54] V. Mirrokni, N. Thain, A. Vetta, A theoretical examination of practical game playing: Lookahead search, in: Algorithmic Game Theory, Springer, 2012, pp. 251–262.

[55] C. C. White III, D. J. White, Markov decision processes, European Journal of Operational Research 39 (1) (1989) 1–16.

[56] G. A. Korsah, A. Stentz, M. B. Dias, A comprehensive taxonomy for multi-robot task allocation, The International Journal of Robotics Research 32 (12) (2013) 1495–1512.

[57] G. K. Smyth, Nonlinear regression, Encyclopedia of environmetrics.

[58] S. G. Hart, L. E. Staveland, Development of nasa-tlx (task load index): Results of empirical and theoretical research, Advances in psychology 52 (1988) 139–183.

[59] S. S. Shapiro, M. B. Wilk, An analysis of variance test for normality (complete samples), Biometrika 52 (3/4) (1965) 591–611.

[60] J. Casper, R. R. Murphy, Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 33 (3) (2003) 367–385.

[61] R. R. Murphy, S. Tadokoro, D. Nardi, A. Jacoff, P. Fiorini, H. Choset, A. M. Erkmen, Search and rescue robotics, in: Springer Handbook of Robotics, Springer, 2008, pp. 1151–1173.

[62] E. Guizzo, Japan earthquake: Robots help search for survivors, [Online; posted 14-March-2011] (March 2011).
URL http://spectrum.ieee.org/automaton/robotics/industrial-robots/japan-earthquake-robots-help-search-for-survivors

[63] R. R. Murphy, Disaster robotics, Cambridge MA: The MIT Press, Intelligent Robotics and Autonomous Agents series, 2014.

[64] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics bulletin 1 (6) (1945) 80–83.

[65] K. J. Gilhooly, Human and machine problem solving, Springer Science & Business Media, 2012.

[66] Humanoid robots learning to walk faster: From the real world to simulation and back, in: Proc. of Conference on Autonomous Agents and Multiagent Systems AAMAS, 2013.

[67] J. Hanna, P. Stone, Grounded action transformation for robot learning in simulation, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2017.

[68] A. Rosenfeld, O. Maksimov, S. Kraus, N. Agmon, Human-multi-robot team collaboration for efficent warehouse operation, in: Proceedings of ARMS workshop at AAMAS, 2016.

[69] P. J. Besl, N. D. McKay, Method for registration of 3-d shapes, in: Robotics-DL tentative, 1992, pp. 586–606.

[70] P. E. Hart, N. J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimal cost paths, IEEE Transactions on Systems Science and Cybernetics 4 (2) (1968) 100–107.