Science Robotics
AAAS

Supplementary Materials for

In situ bidirectional human-robot value alignment

Luyao Yuan *et al.*

Corresponding author: Luyao Yuan, yuanluyao@ucla.edu; Mark Edmonds, markedmonds@ucla.edu; Hongjing Lu, hongjing@ucla.edu; Yixin Zhu, yixin.zhu@ucla.edu; Song-Chun Zhu, sczhu@stat.ucla.edu

Sci. Robot. **7**, eabm4183 (2022)
DOI: 10.1126/scirobotics.abm4183

The PDF file includes:

Other Supplementary Material for this manuscript includes the following:

Supplementary Methods

Observation model and belief of states

The game map is a 20×20 tile board, initially covered by overlays waiting to be explored. When a robot in the team enters a tile, the surrounding eight neighbors will be revealed. In total, we design sixteen types of tiles in the map, including the empty tile, wall, resource, seven types of devices, and six types of wires. To mimic real-world scenarios, we further add stochastic noise to robots' observation; there is a 15% probability that a robot mistakenly perceives a device/wire tile. For the first time mistakenly perceived tile, the tile will be randomly classified as another device if it is a device. If the tile is a wire, it has a 40% probability of being observed as empty and a 60% probability of being randomly seen as another type of wire. If a misperceived tile got misperceived again, this misperceived tile has a 98% probability of repeating the same misperception and 2% of being randomly observed as in the first time. Observations in different steps are sampled independently for different robots. The robots use a Hidden Markov Model (HMM) to model the observation and solve the belief of every tile using the forward algorithm.

In addition to processing noisy observations for every tile individually, robots can further resolve the observational ambiguities by modeling the compatibility of the neighboring tiles. For example, a tile is very likely to contain a wire if its two neighbors contain devices, even if currently detected as empty. We leverage energy-based model and data-driven Markov Chain Monte Carlo (MCMC) (*1*) to formulize the compatibility of tiles in circuits. Let $c \in \Omega$ denote a circuit configuration; a configuration is a set of non-overlapping circuits in the map. The energy of each configuration is modeled using a potential function, $U(c)$, measuring its possibility of being a valid configuration given the observation so far. The potential function, $U(c)$, accommodates all the attributes of the configuration, including its compatibility with the observations, number of circuits, size of each circuit, and compatibility of neighboring tiles. Computationally, we adopt the Metropolis-Hastings (MH) algorithm (*2*) to find the best configuration optimizing the target distribution $\pi(c) = \exp(-U(c))/Z$. Specifically, we define five types of dynamics to traverse Markov chains from c to $c' \in \Omega$: shift one circuit by an offset, merge two circuits into one, split one circuit into two, delete a circuit, and add a circuit. The proposal distribution used for MH, $Q(c, c')$, is set to uniform for all c', ensuring $Q(c, c') = Q(c', c)$ for most configurations. For fast interactions with human users, we run MH for 100 steps or stop early if the proposal were rejected 3 consecutive times; the final configuration is used for robot planning.

Of note, robots need to form a belief over the states of the entire map to make a rational decision, including the contents of both the explored and unexplored tiles. The above energy-based model only handles the observations from the explored regions. For the unexplored tiles, robots first sample the total number of resources and circuits from two Poisson distributions with $\lambda_1 = 7, \lambda_2 = 5$, respectively. Next, additional resources and circuits which are not in the explored regions are randomly placed in the unexplored regions to form a complete map. This process is repeated 2000 times to generate a set of maps, which becomes the robots' belief of the physical state, $b(s)$. All samples have equal weights; invalid maps, such as extra circuits that are not fit in the unexplored region, are removed.

Scouts plan space construction

According to the game rules, each scout only need to choose movement targets among the task destination tile in the upper left corner of the map, tiles with unclaimed resources, and unexplored tiles in the map. To construct the plan space to be assigned to individual scout, all combinations of K (the number of scouts) targets for the robot team are first generated from all possible targets. Next, for every combination, each scout of the robot team is assigned to one of the K targets. Given the target, we use the A^* algorithm to find the shortest path for one scout to find the designated tile. To encourage exploration, every unexplored tile has a cost of 1, explored empty or device tile has a cost of 1.01, and wall tile has an infinite cost.

Suppose there are M possible targets in the map, $|\mathcal{T}| = \prod_{i=0}^{K-1}(M-i) = O(M^K)$. At the beginning of the game, because most of the map is unexplored, M is close to 400 such that $\mathcal{T}$ is too large to be processed exactly in real-time; real-time condition is crucial for running interactive experiments online in this study. To address this issue, we choose K targets not from all possible target tiles, but from a heuristic subset of it. The heuristic subset consists of unexplored corners of the map, unclaimed resources, centers of unexplored partial circuits, and centers of unexplored regions. To find the centers of unexplored regions, we adopt the fast marching method to calculate the shortest distance from every unexplored tile to the boundary of an unexplored region and choose K tiles with the longest distance as the centers. Furthermore, given K targets, scouts are assigned to their targets so that the scout team has the minimum total path length, reducing the cardinality of $\mathcal{T}$ by a factor of $K!$. When there are fewer possible targets on the map, we allow scouts to go to the same targets.

When to propose

In the first half of the game, the robot team makes a proposal when any one of the following four events happen:

- A new device is found in the map
- A new resource is found in the map
- The detection of a device has been changed. *E.g.*, a previously detected wire is classified as a bomb after the scouts move closer.
- No proposals have been made for the past 2 steps.

In the second half of the game (after a scout's Manhattan distance to the destination is smaller than its distance to the starting point), the robot team makes a proposal when any of these four events happen and the expected parameter change caused by the optimal plan is above a threshold. To calculate the expected parameter change, agents calculate the parameter change triggered by all feedback combinations (2 options for each scout, 8 combinations in total) using Eq. (7) and average the change by the probability of each feedback using Eq. (2). We use 0.05 as the threshold. That is, if any dimension has an expected parameter change larger than 5 percent, the scouts will generate proposals. Along with the proposal, different formats of explanations will be provided according to the participant's explanation group, following the description in the Experimental design section.

Parameter learning function derivation

Agents with level-2 Theory-of-Mind (ToM) updates their belief by considering both the literal meaning and the pragmatic meaning of the human feedback. Hence, the agents try to maximize $p(m^H(fb)|\tau;\theta)$ and $q(m^H(fb)|\bar{\theta},\tau;\theta)$ w.r.t. θ simultaneously. Of note, in both terms, θ denotes the user's true value function, not to be confused with the robots' value denoted by $\bar{\theta}$. $p(m^H(fb)|\tau;\theta)$ captures the literal meaning of the feedback, that is, the probability of a user with value function θ gives out $m^H(fb)$ based on how the proposal τ is compliant with θ. $q(m^H(fb)|\bar{\theta},\tau;\theta)$ captures the pragmatic meaning of the feedback, namely the probability of a cooperative user with value θ gives out $m^H(fb)$ because it is more helpful to the scouts to align to the true value. Thus, agents with level-2 ToM maximize a new likelihood $p(m^H(fb)|\tau;\theta)q(m^H(fb)|\bar{\theta},\tau;\theta)$ w.r.t. θ. Let $g(m) = \frac{\partial \log p(m|\tau;\theta)}{\partial\theta}$, we have

$$\begin{aligned}
&\frac{\partial}{\partial\theta}\,\mathrm{log}p(m^H(fb)|\tau;\theta)q(m^H(fb)|\bar{\theta},\tau;\theta)\\
&\quad= g(m^H(fb)) + \frac{\partial \log q(m^H(fb)|\bar{\theta},\tau;\theta)}{\partial\theta}\\
&\quad= g(m^H(fb)) + 2\beta_2\big(\bar{\theta} + \eta g(m^H(fb)) - \theta^*\big)\\
&\qquad - 2\beta_2 \sum_{m\in FB} \frac{(\bar{\theta} + \eta g(m) - \theta^*)\exp(\beta_2\|\bar{\theta} + \eta g(m) - \theta^*\|^2)}{\sum_{m'\in FB}\exp(\beta_2\|\bar{\theta} + \eta g(m') - \theta^*\|^2)}\\
&\quad= g\big(m^H(fb)\big) + 2\beta_2\eta\Big(g\big(m^H(fb)\big) - \mathrm{E}_{m\sim q(m|\bar{\theta},\tau;\theta)}\big[g(m)\big]\Big).
\end{aligned} \tag{S1}$$

Multiplying Eq. (S1) with learning rate η, we have the update function in Eq. (7). Eq. (5) gives the closed-form of $g(m)$.

Sequential explanation generation

Fig. S1 depicts the overall explanation generation algorithm. At time step t, the explainer takes in a tuple $h_t = \{(m^E_{t-1}, ss_{t-1}, o_t)\}$ as the input, where $m^E_{t-1} \in M^E$ is the explanation of the previous round, $ss_{t-1} \in SS$ is user's satisfactory score estimated by human user's feedback of the previous round, and $o_t \in O$ is the current observation. Given the sequential input history $H_t = \{h_k, k = 1, ..., t\}$, the explanation objective is to generate an explanation m^E_t that maximizes the expected score:

$$m^E_t = \underset{m^E \in M^E}{\arg\max}\, \mathbb{E}_{\hat{ss} \sim p(ss|H_t)}[\hat{ss}(a^E)] - \lambda_c \text{cost}(m^E), \tag{S2}$$

where $a^E \in A^E$ is an extracted attribute vector of m^E, cost$(\cdot)$ a pre-defined cost function, and λ_c a constant factor.

We model the process of computing $\mathbb{E}[\hat{ss}(a^E)]$ as an HMM by introducing a hidden variable $\upsilon \in \Upsilon$, which corresponds to the human user's explanation utility; see Fig. S1 for a graphical illustration of the computing process. At time step t, we compute the expected score as:

$$\begin{aligned}\mathbb{E}_{\hat{ss} \sim p(ss|H_t)}[\hat{ss}(a^E)] &= \sum_{ss \in SS} p(ss|a^E, ss_{1:t-1}, o_{1:t}, a^E_{1:t-1}) ss \\ &= \sum_{ss \in SS} \left(\sum_{\upsilon_t \in \upsilon} p(\upsilon_t | ss_{1:t-1}, a^E_{1:t-1}, o_{1:t}) p(ss|\upsilon_t, a^E) \right) ss.\end{aligned} \tag{S3}$$

Let $\mathcal{K}(a^E_{t-1}, o_t) = p(\upsilon_t | \upsilon_{t-1}, a^E_{t-1}, o_t)$ be the transition matrix that encodes the transition probability from mental states υ_{t-1} to υ_t, and $\mathcal{F}(a^E) = p(ss|\upsilon_t, a^E)$ be the score function that models the distribution of satisfaction scores. We have:

$$p(\upsilon_t | ss_{1:t-1}, e_{1:t-1}, o_{1:t}) = \sum_{\upsilon_{t-1} \in \upsilon} p(\upsilon_{t-1} | ss_{1:t-1}, a^E_{1:t-1}, o_{1:t-1}) \mathcal{K}(a^E_{t-1}, o_t), \tag{S4}$$

where $p(\upsilon_t | ss_{1:t}, a^E_{1:t}, o_{1:t}) = \alpha_t$ is computed by an iterative process:

$$\begin{aligned}p(\upsilon_t | ss_{1:t}, a^E_{1:t}, o_{1:t}) &\propto \mathcal{F}(a^E_t) \odot \left(\mathcal{K}(a^E_{t-1}, o_t)^{\mathrm{T}} p(\upsilon_{t-1} | ss_{1:t-1}, a^E_{1:t-1}, o_{1:t-1}) \right) \\ &= \mathcal{F}(a^E_t) \odot \left(\mathcal{K}(a^E_{t-1}, o_t)^{\mathrm{T}} \alpha_{t-1} \right),\end{aligned} \tag{S5}$$

where $\odot$ is an element-wise product operator. Therefore, Eq. (S3) can be written as

$$\mathbb{E}_{\hat{ss} \sim p(ss|H_t)}[\hat{ss}(a^E)] = \sum_{ss \in SS} \frac{ss}{Z} \alpha_t^{\mathrm{T}} \mathcal{K}(a_{t-1}^E, o_t) \mathcal{F}(e), \tag{S6}$$

where Z is a normalization constant of $p(ss|H_t)$; see Alg. S2 for the full computational flow.

Algorithm S1: Overview of the Scout Exploration Game

```
Set $t = 1$, initialize $s^t$, agent's mental state $x_0^R$;
while not task-complete($x_{t-1}^R$) do
    $o_t \sim$ observation_model($s_t$)    // collect observations from the environment
    $\widehat{x_t^R}$ = update_state_belief($x_{t-1}^R, o_t$)    // update belief given observations
    $m_t^R \sim$ proposal_explanation_generation($\widehat{x_t^R}$)    // generate messages (proposal &
     explanation) to the user
    $x_t^R$ = update_value_belief($\widehat{x_{t-1}^R}, m_t^R, m_t^H$)  // update beliefs given user feedback
    $\mathbf{a}_t^R \sim$ action_policy($x_t^R$)    // agent's policy
    $s_{t+1} \sim$ game_dynamics($s_t, \mathbf{a}_t^R$)    // state transition
    $t = t + 1$
end
```

Algorithm S2: Explanation Generation

```
Input  : templates - all explanation templates
Output: $\{m_1^E, m_2^E, ...\}$
$t \leftarrow 1$
while not stopped do
    explanations ← FillSlots(templates)
    Get $O_t$, $ss_{t-1}$ from agent
    $m^E \leftarrow None$
    for $m_i^E$ in explanations do
        $a^E \leftarrow$ ExtractAttribute($m_i^E$)
        Compute $\mathbb{E}[\hat{ss}(a^E)]$ according to Eq. (S6).
        $m^E \leftarrow \arg\max_{\{m^E, m_i^E\}} \mathbb{E}[\hat{ss}(a^E)] - \text{cost}(m^E)$
    end
    $m_t^E \leftarrow m^E$, $t \leftarrow t + 1$
end
```

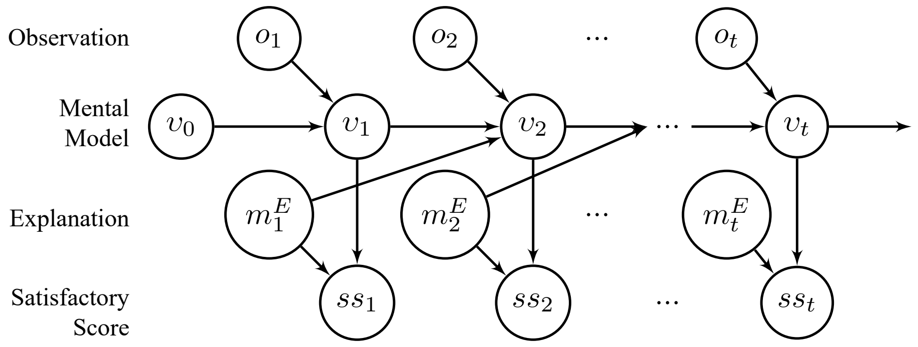


Figure S1: **Temporal evolution of explanation generation as a function of t.** After receiving the explanation m_{t-1}^E and map observation o_t, the user update its mental model v_t. Based on the new mental model, the user gives a satisfactory score ss_t of the new explanation m_t^E.

Explanation templates

Table S1 shows the templates used for sequential explanation generation.

Domain	Attribute	Template
Brief Explanation		
Make Proposal		I propose to move along the {color} trajectory. This trajectory may {benefit} than {percentile}% of other sampled trajectories.
		I propose to move along the {color} trajectory. This trajectory may {benefit} than {percentile}% of other sampled trajectories.
		I propose to move along the {color} trajectory, which is in the {top_or_bottom} {refined_percentile}% of sampled trajectories when optimizing {key_factor}.
Full Explanation		
Explain Circuit	`hasDevice`	{scout} has found a device! The device at {position} is classified as {classification}, because the device {has_or_lacks} {components}.
	`updateCircuit`	The classification of device at {position} has been updated to {classification}, because the device {has_or_lacks} {components}.
Explain History	`hasHistory`	Scout's value has been updated by {update}. Because \n {history}.
	`isRitualized`	Scout's value has been updated by {update}.
Explain State	`hasKeyFactor`	The scouts think that {key_factor} is/are likely to be the most important factor(s).
		The scouts think that {key_factor} is/are likely to be the most important factor(s).
		The factor(s) {key_factor} is/are the most important in the scouts current estimation.
	`hasKeyFactor` `isRitualized`	The scouts think the key factor(s) is/are likely to be {key_factor}.
Explain Proposal	`hasValue`	{scout} wants to {benefit}. This proposal, however, will sacrifice {value_dec} in the future.
		{scout} wants to {benefit} in the future at the cost of {cost}.
	`hasValue` `hasTarget`	{scout} [aims|targets] at {target}. This proposal will {benefit}, which, on the other hand, will sacrifice {value_dec} in the future.
	`hasValue` `isRitualized`	{scout}'s proposal wants to {benefit}.
	`isCounterfactual`	Comparing with [removing|excluding] {scout}'s trajectory in our plan, including it [tends to|may] {benefit}, but will also {cons}.
	`isCounterfactual` `hasTarget`	{scout} [aims|targets] at {target}. Comparing with [removing|excluding] this trajectory in our plan, including it [tends to|may] {benefit}, but will also {cons}.
	`isCounterfactual` `isRitualized`	Comparing with [removing|excluding] {scout}'s trajectory in our plan, including it [tends to|may] {benefit}.
	`hasTarget` `isRitualized`	{scout} [aims|targets] at {target}. Including it [tends to|may] {benefit}.
	`isRitualized`	{scout} wants to {benefit}. Comparing with {advantage}, the trajectory is {other_effect}.

Table S1: **Attributed explanation templates for sequential explanation generation.** Slots in brackets (*e.g.*, {color}) are produced by the planner or a short program that organizes semantic information. Given such inputs, the syntactic templates are selected w.r.t. users' satisfactory score. Specifically, the brief explanation is selected in random using templates of "Make Proposal"; whilst the full explanation first estimates users' preferred attribute set (*i.e.*, explanation utility) for each domain and then selects the corresponding templates.

Additional game details

Fig. S2 shows the ground-truth map used in the Scout Exploration Game. Fig. S3 shows the value functions used in the Scout Exploration Game.

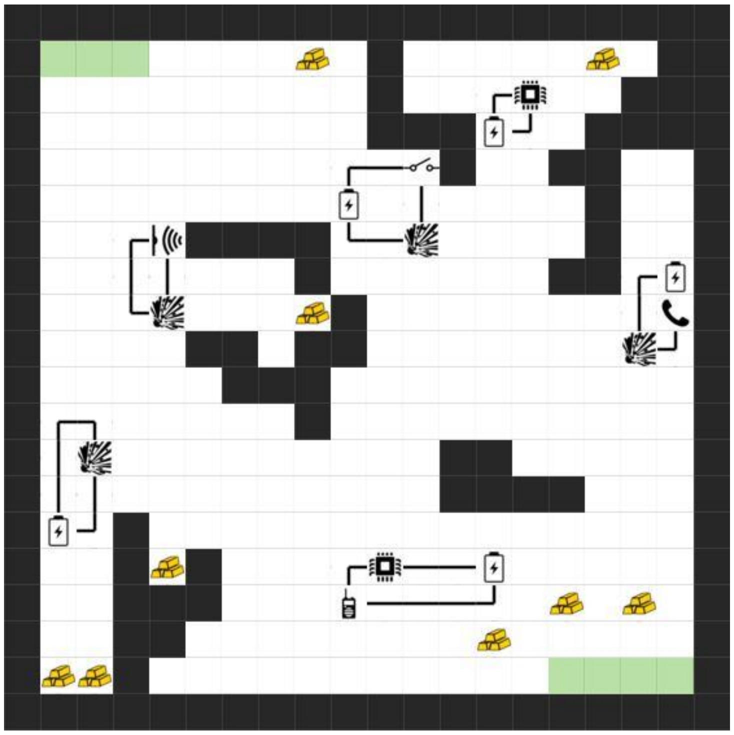
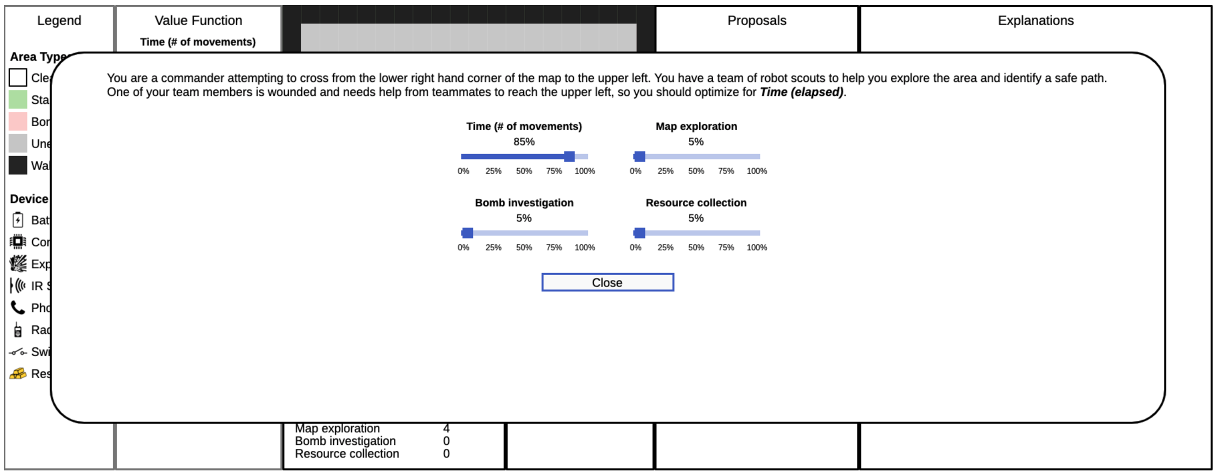


Figure S2: **The ground-truth map used in the game.** The map is a 20×20 grid board. The three scouts, as a robot team, will set off in the bottom right corner and move towards the top left corner. There are six potential explosive circuits and nine resources. All grids are initially covered with overlays. A grid will be disclosed when a scout detects the content in it. Every scout can detect eight neighbor grids. We used a static map: Content in the grids will not change during the game, and the detected grids will not be covered again. Only scouts can move on the map.

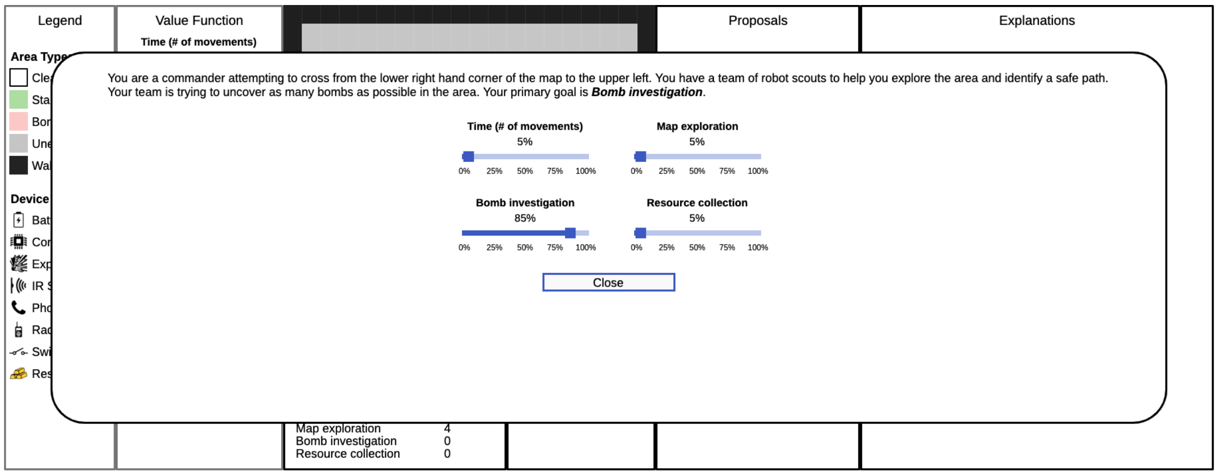


You are a commander attempting to cross from the lower right hand corner of the map to the upper left. You have a team of robot scouts to help you explore the area and identify a safe path. One of your team members is wounded and needs help from teammates to reach the upper left, so you should optimize for ***Time (elapsed)***.

Time (# of movements) 85%
Map exploration 5%
Bomb investigation 5%
Resource collection 5%

Map exploration	4
Bomb investigation	0
Resource collection	0

(A) **Time**

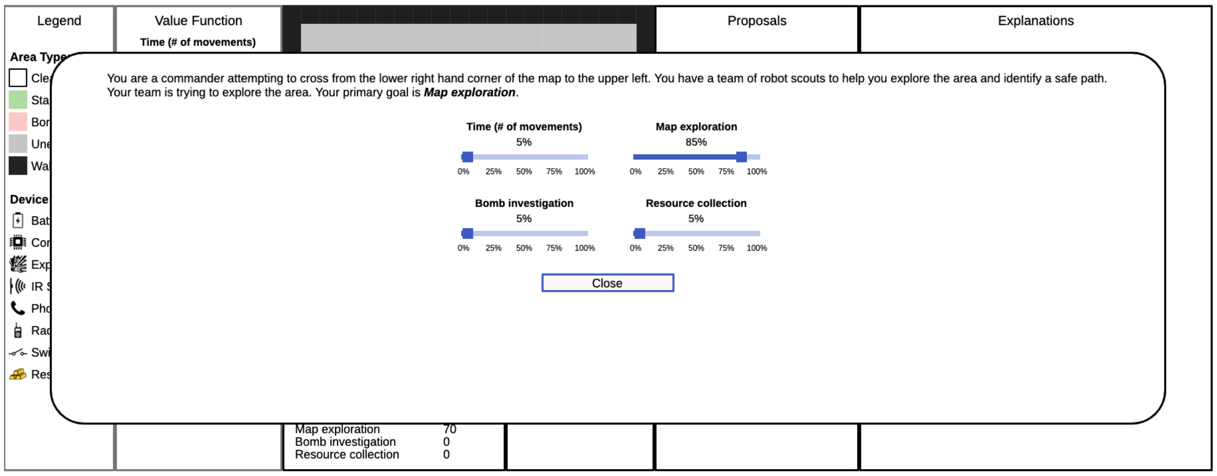


You are a commander attempting to cross from the lower right hand corner of the map to the upper left. You have a team of robot scouts to help you explore the area and identify a safe path. Your team is trying to uncover as many bombs as possible in the area. Your primary goal is ***Bomb investigation***.

Time (# of movements) 5%
Map exploration 5%
Bomb investigation 85%
Resource collection 5%

Map exploration	4
Bomb investigation	0
Resource collection	0

(B) **Bomb investigation**

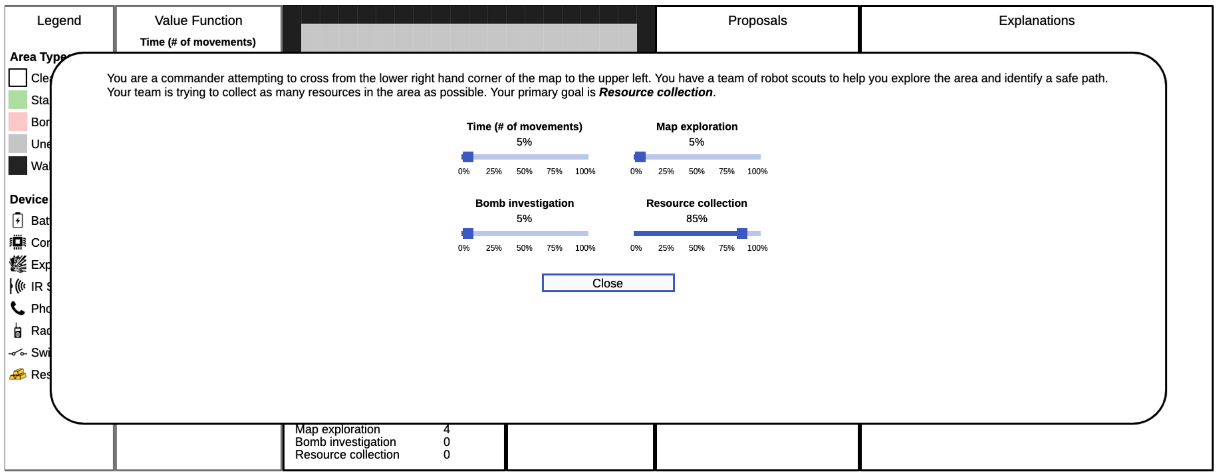


You are a commander attempting to cross from the lower right hand corner of the map to the upper left. You have a team of robot scouts to help you explore the area and identify a safe path. Your team is trying to explore the area. Your primary goal is ***Map exploration***.

Time (# of movements) 5%
Map exploration 85%
Bomb investigation 5%
Resource collection 5%

Map exploration	70
Bomb investigation	0
Resource collection	0

(C) **Map exploration**

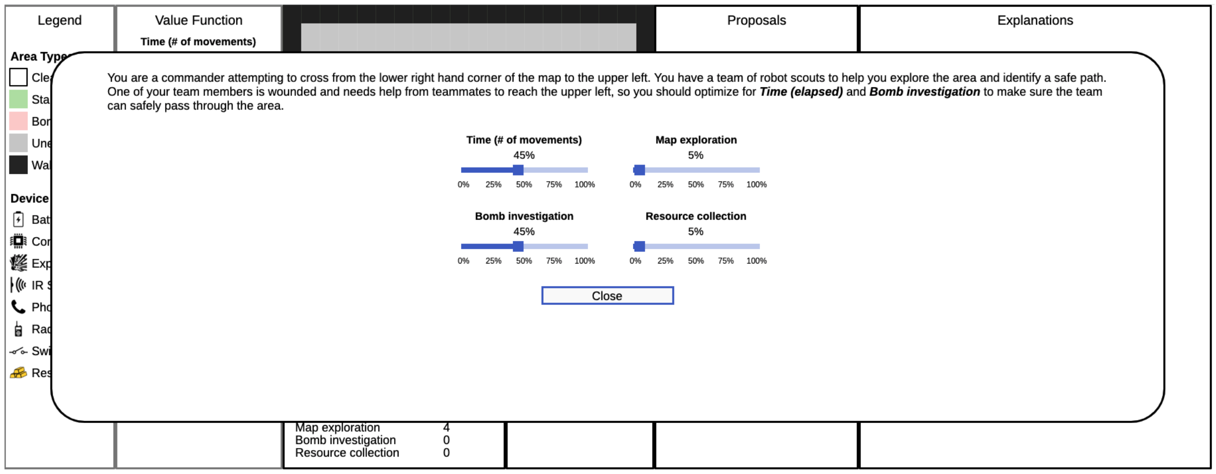


You are a commander attempting to cross from the lower right hand corner of the map to the upper left. You have a team of robot scouts to help you explore the area and identify a safe path. Your team is trying to collect as many resources in the area as possible. Your primary goal is ***Resource collection***.

Time (# of movements) 5%
Map exploration 5%
Bomb investigation 5%
Resource collection 85%

Map exploration 4
Bomb investigation 0
Resource collection 0

(D) Resource collection

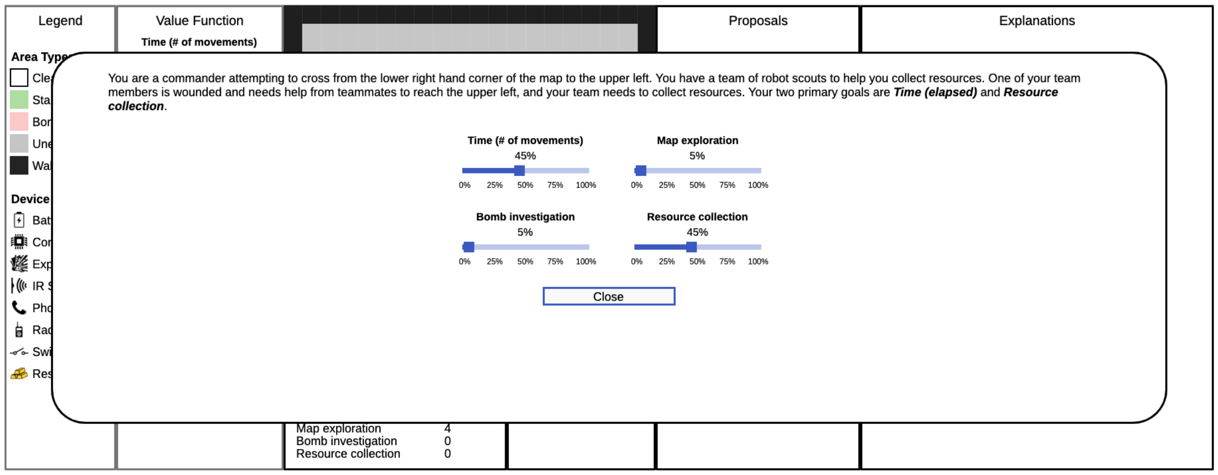


You are a commander attempting to cross from the lower right hand corner of the map to the upper left. You have a team of robot scouts to help you explore the area and identify a safe path. One of your team members is wounded and needs help from teammates to reach the upper left, so you should optimize for ***Time (elapsed)*** and ***Bomb investigation*** to make sure the team can safely pass through the area.

Time (# of movements) 45%
Map exploration 5%
Bomb investigation 45%
Resource collection 5%

Map exploration 4
Bomb investigation 0
Resource collection 0

(E) Time and bomb investigation

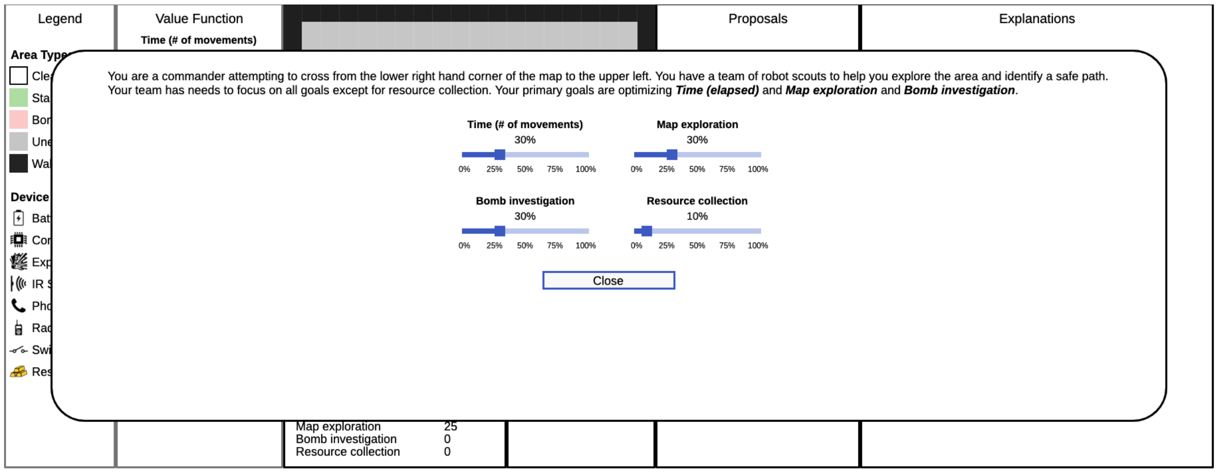


You are a commander attempting to cross from the lower right hand corner of the map to the upper left. You have a team of robot scouts to help you collect resources. One of your team members is wounded and needs help from teammates to reach the upper left, and your team needs to collect resources. Your two primary goals are ***Time (elapsed)*** and ***Resource collection***.

Time (# of movements) 45%
Map exploration 5%
Bomb investigation 5%
Resource collection 45%

Map exploration 4
Bomb investigation 0
Resource collection 0

(F) Time and resource collection



You are a commander attempting to cross from the lower right hand corner of the map to the upper left. You have a team of robot scouts to help you explore the area and identify a safe path. Your team has needs to focus on all goals except for resource collection. Your primary goals are optimizing ***Time (elapsed)*** and ***Map exploration*** and ***Bomb investigation***.

Time (# of movements) 30%
Map exploration 30%
Bomb investigation 30%
Resource collection 10%

```
Map exploration       25
Bomb investigation    0
Resource collection   0
```

(**G**) **Time, bomb investigation and map exploration.**

Figure S3: **Value functions used in the game.** One of the seven value functions will be randomly assigned to the human user before the game starts. There are four unary value functions (with a single dominant goal), two binary functions (with two equally important goals) and one trinary function (with three equally important goals). The value of each goal is represented with a percentage bar, and the sum of all values equals to 100%. Along with the value function, we also provide a background description for the participants to better grasp the semantics of the value.