

## 489 A Algorithm: Differentiable Surrogate Assisted Scenario Generation

490 The improvements proposed in Sec. 4 result in two versions of our algorithm. In Algorithm 1,  
491 we present DSAS with the state-of-the-art DQD algorithm CMA-MAEGA in the inner loop. SAS  
492 follows a similar structure but with CMA-MAE in the inner loop.

493 On each iteration of the outer loop, we initialize a new surrogate archive to store solutions that the  
494 surrogate model predicts are high performing and diverse (line 3). Then, we begin the inner loop  
495 (line 5). On line 6, we evaluate the current solution point  $\theta$  with the surrogate model to obtain  
496 the predicted objective  $\hat{f}$ , measures  $\hat{m}$ , and the branching gradients  $\nabla_{\hat{f}}$  and  $\nabla_{\hat{m}}$ . We then add  
497 the solution  $\theta$  to the surrogate archive (line 8) based on the predicted evaluations, after applying  
498 the regularization penalty (line 7). Next, we generate a batch of solutions based on the branching  
499 gradients (line 9). For each solution, we sample gradient coefficients, which, combined with the  
500 gradients, produce a new candidate solution (lines 10-12). We evaluate each new candidate solution  
501  $\theta'_i$  with the surrogate model (line 13), apply the regularization penalty (line 14), and add the solution  
502 to the surrogate archive (line 15). After processing a batch, we update the search parameters of  
503 CMA-MAEGA to move the search towards maximizing the QD objective (line 17).

504 After completing an inner loop, we select a subset of solutions from the surrogate archive to label  
505 (line 19). For each set of scenario parameters  $\theta$ , we generate-and-repair a scenario (line 21), evaluate  
506 the robotic system on the scenario (line 22), update our dataset by adding the scenario labeled with  
507 the true objective  $f$ , measures  $m$ , robot occupancy grid  $y_r$ , and human occupancy grid  $y_h$  (line 23),  
508 and finally add the scenario to our ground-truth archive (line 24). After updating the training data  
509 with newly labeled scenarios, we train the occupancy predictor for both the robot (line 27) and  
510 human (line 28), then train the surrogate model to predict the objectives and measures (line 29). The  
511 inner loop in future iterations exploits the more accurate surrogate model to produce better scenarios.

## 512 B Surrogate Model Details

513 Our surrogate model follows a two-stage prediction process by first predicting the robot and the  
514 human occupancy grids given the scenario parameters as input, followed by a downstream prediction  
515 of the objective and measures.

516 The occupancy predictor (blue arrows in Fig. 6) consists of deconvolution layers followed by batch  
517 normalization and ReLU that treat the scenario parameters as a  $1 \times 1$  image with the number of  
518 channels equal to the solution size and expand it into a  $32 \times 32$  image. In the shared control teleop-  
519 eration domain, there is only one occupancy grid since only the robot arm is moving. In the shared  
520 workspace collaboration domain, there are two occupancy grids (stacked into two channels) corre-  
521 sponding to the robot and the human motion. We pass each channel in the final output through a  
522 softmax operator and minimize the KL divergence loss between the predicted and the true occupancy  
523 grids.

524 The downstream predictor (red arrows in Fig. 6) consists of a fully connected network with linear  
525 layers followed by batch normalization and ReLU to extract features from the scenario parameters.  
526 It also consists of convolutional layers followed by batch normalization and leaky ReLU to extract  
527 features from the occupancy grids. We pass the features through a linear layer and minimize the  
528 mean squared error (MSE) between the predicted and the true objective and measures.

529 The losses for the occupancy predictor and the downstream predictor have different scales and hence,  
530 are hard to balance. Thus, we separately train both networks on data obtained from ground-truth  
531 evaluations for 100 epochs in each outer iteration using Adam [62] optimizer with a learning rate  
532 of 0.0001 and batch size of 64. We first train the occupancy predictor, freeze the weights, and then  
533 train the downstream predictor by leveraging occupancy predictions from the occupancy predictor.  
534 We implement and train the networks with the PyTorch library [63].

---

**Algorithm 1:** Differentiable Surrogate Assisted Scenario Generation (DSAS).
 

---

**Input:**  $N$ : Maximum number of evaluations,  $N_{exploit}$ : Number of iterations in the model exploitation phase,  $\theta_0$ : Initial solution for CMA-MAEGA,  $B$ : Batch size for CMA-MAEGA

**Output:** Final version of the ground-truth archive  $\mathcal{A}_{gt}$

```

1 Initialize the ground-truth archive  $\mathcal{A}_{gt}$ , the dataset  $\mathcal{D}$ , robot occupancy predictor  $sm_r$ , human
  occupancy predictor  $sm_h$ , objective and measure predictor  $sm$ 
2 while  $evals < N$  do
3   Initialize CMA-MAEGA with the surrogate archive  $\mathcal{A}_{surr}$  and initialize solution  $\theta$  to  $\theta_0$ 
4   Initialize CMA-ES parameters  $\mu, \Sigma$ 
5   for  $itr \in \{1, 2, \dots, N_{exploit}\}$  do
6      $\hat{f}, \nabla_{\hat{f}}, \hat{m}, \nabla_{\hat{m}} \leftarrow sm(\theta, sm_r(\theta), sm_h(\theta))$ 
7      $\hat{f} \leftarrow \hat{f} - reg(\theta)$ 
8      $\mathcal{A}_{surr} \leftarrow add\_solution(\mathcal{A}_{surr}, (\theta, \hat{f}, \hat{m}))$ 
9     for  $i \in \{1, 2, \dots, B\}$  do
10       $c \sim \mathcal{N}(\mu, \Sigma)$ 
11       $\nabla_i \leftarrow c_0 \nabla_{\hat{f}} + \sum_{j=1}^k (c_j \nabla_{\hat{m}_j})$ 
12       $\theta'_i \leftarrow \theta + \nabla_i$ 
13       $\hat{f}', *, \hat{m}', * \leftarrow sm(\theta'_i, sm_r(\theta'_i), sm_h(\theta'_i))$ 
14       $\hat{f}' \leftarrow \hat{f}' - reg(\theta'_i)$ 
15       $\mathcal{A}_{surr} \leftarrow add\_solution(\mathcal{A}_{surr}, (\theta'_i, \hat{f}', \hat{m}'))$ 
16    end
17    Update  $\theta, \mu, \Sigma$  via CMA-MAEGA update rules
18  end
19   $\Theta \leftarrow select\_solutions(\mathcal{A}_{surr})$ 
20  for  $\theta \in \Theta$  do
21     $scenario \leftarrow G(\theta)$ 
22     $f, m, y_r, y_h \leftarrow evaluate(scenario)$ 
23     $\mathcal{D} \leftarrow \mathcal{D} \cup (\theta, f, m, y_r, y_h)$ 
24     $\mathcal{A}_{gt} \leftarrow add\_solution(\mathcal{A}_{gt}, (\theta, f, m))$ 
25     $evals \leftarrow evals + 1$ 
26  end
27   $sm_r.train(\mathcal{D})$ 
28   $sm_h.train(\mathcal{D})$ 
29   $sm.train(\mathcal{D}, sm_r, sm_h)$ 
30 end
  
```

---

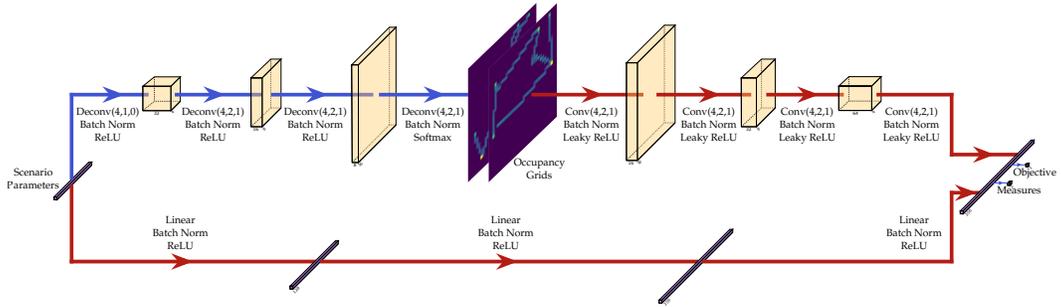


Figure 6: Architecture of the surrogate model including the occupancy predictor (blue arrows) and the downstream predictor (red arrows).

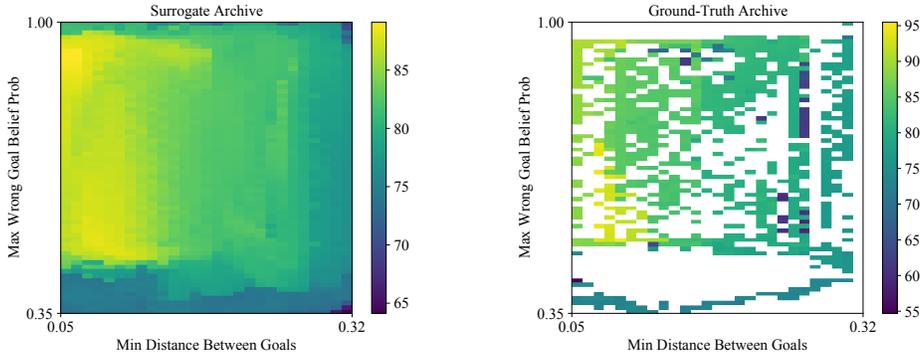


Figure 7: Comparison between the surrogate archive (left) after an inner loop and the corresponding ground-truth archive (right) after evaluating the solutions in the surrogate archive.

Table 1: Mean absolute error of the objective and measure predictions by the surrogate models.

Domain	DSAS			SAS		
	Objective MAE	Measure 1 MAE	Measure 2 MAE	Objective MAE	Measure 1 MAE	Measure 2 MAE
Shared Control Teleoperation	0.35	0.01	0.01	0.64	0.02	0.01
Collaboration I	3.41	0.02	0.09	3.47	0.02	0.08
Collaboration II	3.22	0.27	0.56	3.39	0.29	0.59

## 535 B.1 Evaluating the Surrogate Model Predictions

536 We evaluate the predictions of the surrogate model similar to DSAGE [21] by taking the dataset  
 537 generated in one trial of an algorithm and treating it as the test set for the trained surrogate model  
 538 from another trial of the same algorithm. Table 1 shows the mean absolute error (MAE) in all three  
 539 domains for the surrogate models trained as a part of both DSAS and SAS. Note that *Measure 1* and  
 540 *Measure 2* columns in the table correspond to the respective measures in each domain described in  
 541 Sec. 5.

542 The surrogate model is able to accurately predict the measures in the shared control teleoperation  
 543 domain since they can be calculated directly from the solution and do not depend on the robot policy.  
 544 In contrast, measures that depend on the robot policy such as the maximum wrong goal probability  
 545 (*Measure 2* in the collaboration I domain) have a comparatively higher error, with predictions being  
 546 off by around 9% on average.

547 Furthermore, we observe that the percentage of predictions landing in their true archive cell is only  
 548 around 2-4% in all domains. Nonetheless, the predictions are close to their true archive cell as evi-  
 549 dent in the MAEs. We also confirm this by computing the average Manhattan distance between the  
 550 predicted archive cell and the true archive cell for each solution. In the shared control teleoperation  
 551 domain, the average Manhattan distance was 6.48 and 11.26 for DSAS and SAS respectively. The  
 552 average Manhattan distances for DSAS and SAS were 11.26 and 9.88 in the collaboration I domain,  
 553 and 6.89 and 7.20 in the collaboration II domain, indicating that the predicted archive cells are only  
 554 a few cells away from the true archive cells on average.

555 Thus, despite inaccuracies in placing the solutions into their true archive cells, the solutions in  
 556 the surrogate archive are diverse with respect to the true measure functions. Hence, when these  
 557 solutions are evaluated, they occupy different parts of the ground-truth archive and rapidly improve  
 558 the QD-score. Fig. 7 shows the surrogate archive after one inner loop and the corresponding ground-  
 559 truth archive obtained by evaluating the solutions in the surrogate archive in the collaboration I  
 560 domain.

## 561 C Mixed Integer Program for Repairing Scenarios

562 To ensure that the objects in the scenario generated by QD search satisfy the object arrangement  
 563 constraints in the shared workspace collaboration domain, we adopt a generate-then-repair strategy.  
 564 We formulate a mixed integer program (MIP) with constraints to ensure that the objects in the  
 565 scenario are inside the workspace boundaries and not in collision with each other. Since we wish the  
 566 repaired scenario to be as close as possible to the generated scenario, we set the MIP objective to be  
 567 the  $L_2$  distance between the original position and the repaired position of the objects. The quadratic  
 568 objective makes the MIP a mixed integer quadratic program (MIQP).

### 569 C.1 Variables and MIP Objective

570 We treat the  $x$  and  $y$  coordinates of each object as the MIP variables. Let  $x'_i$  and  $y'_i$  be the coordinates  
 571 of object  $i$  in the generated scenario and let  $x_i$ , and  $y_i$  be the corresponding coordinates after MIP  
 572 repair. We set the objective to be:

$$\min \sum_i (x_i - x'_i)^2 + (y_i - y'_i)^2 \quad (2)$$

### 573 C.2 Constraints

574 Let  $x_r^{(min)}$ ,  $x_r^{(max)}$ ,  $y_r^{(min)}$ , and  $y_r^{(max)}$  be the minimum and maximum allowed  $x$  and  $y$  values  
 575 respectively for objects in a rectangular workspace region  $r$ . For each workspace region, we need  
 576 to construct a binary variable  $z_{ir}^{(in)}$ , which resolves to true if object  $i$  occupies workspace  $r$ . We  
 577 create four auxiliary decision variables  $z_{ir}^{(up)}$ ,  $z_{ir}^{(dn)}$ ,  $z_{ir}^{(lt)}$ , and  $z_{ir}^{(rt)}$ , representing the four boundary  
 578 constraints of the rectangle. Specifically,  $z_{ir}^{(up)}$  represents if object  $i$  occupies  $\langle x_i, y_i \rangle$  coordinates  
 579 below the top of the bounding rectangle for region  $r$ . The variables  $z_{ir}^{(dn)}$ ,  $z_{ir}^{(lt)}$ , and  $z_{ir}^{(rt)}$  satisfy  
 580 the same conditions for the bottom, left, and right of the bounding rectangle, respectively. For each  
 581 pair of object  $i$  and region  $r$ , we add the following constraints to the MIP to resolve the decision  
 582 variables:

$$x_r^{(min)} \leq x_i + \infty(1 - z_{ir}^{(lt)}) \quad (3)$$

$$x_i \leq x_r^{(max)} + \infty(1 - z_{ir}^{(rt)}) \quad (4)$$

$$y_r^{(min)} \leq y_i + \infty(1 - z_{ir}^{(dn)}) \quad (5)$$

$$y_i \leq y_r^{(max)} + \infty(1 - z_{ir}^{(up)}) \quad (6)$$

583 In the above constraints, the  $\infty$  value represents a sufficiently large constant (e.g., the maximum of  
 584 the width and height of a global bounding box) that causes the constraint to always be satisfied. For  
 585 example, in Eq. 3, the inequality is always satisfied if the binary decision variable  $z_{ir}^{(lt)}$  is false as  
 586 we do not need to put any constraints if we do not occupy region  $r$  with object  $i$ . However, if the  
 587 variable is true, we require that the coordinate  $x_i$  is to the right of the  $x$ -boundary  $x_r^{(min)}$ . We create  
 588 an equivalent constraint for the remaining three rectangular constraints (see Eq. 4-Eq. 6).

589 Finally, we add a constraint that resolves the decision variable  $z_{ir}^{(in)}$  to true if all four rectangular  
 590 constraints hold:

$$4 \leq z_{ir}^{(lt)} + z_{ir}^{(rt)} + z_{ir}^{(dn)} + z_{ir}^{(up)} + \infty(1 - z_{ir}^{(in)}) \quad (7)$$

591 Once again, if  $z_{ir}^{(in)}$  is false, the inequality holds as we do not need to satisfy the rectangle inclusion  
 592 constraints if our object  $i$  is not in region  $r$ . Otherwise, all four inclusion variables must be true, by  
 593 summing to four, to indicate that the object  $i$  occupies region  $r$ .

594 We then add an additional constraint to ensure that each object occupies at least one region:

$$\forall i, \sum_r z_{ir}^{(in)} \geq 1 \quad (8)$$

595 Next, we ensure that all pairs of objects in the scene do not overlap. To do this, we constrain the  
 596 bounding boxes of each object to not overlap. Let  $a_i$  be half of the side length of the bounding box

597 of object  $i$ . There are four ways a pair of objects with axis-aligned bounding rectangles can avoid  
 598 overlapping: object  $i$  is left of object  $j$ , object  $i$  is right of object  $j$ , object  $i$  is above object  $j$ , or  
 599 object  $i$  is below object  $j$ . We create indicator variables representing these conditions as  $c_{ij}^{(lt)}$ ,  $c_{ij}^{(rt)}$ ,  
 600  $c_{ij}^{(up)}$ ,  $c_{ij}^{(dn)}$ , respectively. Next, we add the following constraints to the MIP to correctly set the  
 601 collision indicator variables:

$$(x_i + a_i) \leq (x_j - a_j) + \infty(1 - c_{ij}^{(lt)}) \quad (9)$$

$$(x_j + a_j) \leq (x_i - a_i) + \infty(1 - c_{ij}^{(rt)}) \quad (10)$$

$$(y_i + a_i) \leq (y_j - a_j) + \infty(1 - c_{ij}^{(dn)}) \quad (11)$$

$$(y_j + a_j) \leq (y_i - a_i) + \infty(1 - c_{ij}^{(up)}) \quad (12)$$

602 If there is no collision between  $i$  and  $j$ , at least one of the four indicator variables must be true.  
 603 Hence, we set an additional constraint to ensure no collision:

$$\forall_{i,j}, c_{ij}^{(lt)} + c_{ij}^{(rt)} + c_{ij}^{(dn)} + c_{ij}^{(up)} \geq 1 \quad (13)$$

604 We solve the MIP problem with IBM’s CPLEX optimization library [64].

## 605 **D Domains**

606 The following subsections provide a brief description of the search space, objective, and measure  
 607 functions in our domains.

### 608 **D.1 Shared Control Teleoperation**

609 A teleoperation task involves a user providing joystick inputs to a robot arm with the intention of  
 610 reaching a goal in the environment. It is generally hard for users to teleoperate a 6-DoF robot arm to  
 611 the correct configuration [51]. Thus, in shared control teleoperation, the robot attempts to infer the  
 612 human goal from a set of candidate goals by observing the low-dimensional joystick inputs provided  
 613 by the user.

614 Following the shared control teleoperation framework from previous work [51], the robot solves a  
 615 POMDP with the user’s goal as a latent variable while it updates its belief about the goal based on  
 616 the human input trajectory assuming a noisily-optimal user. To enable real-time decision-making,  
 617 the robot performs hindsight optimization to approximate the POMDP and assumes a first-order  
 618 approximation of the value function. This results in the robot’s actions being a weighted average  
 619 of the optimal path towards each goal, where the weights are proportional to the respective goal  
 620 probabilities.

621 To formalize the scenario generation problem in the shared teleoperation domain, we follow the  
 622 QD formulation of prior work [1, 2]. The environment parameters are the positions of the two  
 623 goal objects in a bounded workspace, constrained to be reachable by the robot arm. The simulated  
 624 human provides a trajectory of joystick inputs towards their goal object, parameterized by a set  
 625 of waypoints. The human model parameters are disturbances to these waypoints. The scenario  
 626 parameters  $\theta$  include the environment and human model parameters. The objective function  $f$  in  
 627 the QD search is the time taken to reach the correct goal, with a maximum time limit of 10 seconds  
 628 if the robot fails to reach the goal. The search aims to find scenarios that are diverse with respect to  
 629 the noise in human inputs and the scene clutter, thus the measures  $m$  are the human variation from  
 630 the optimal path and the distance between goals.

### 631 **D.2 Shared Workspace Collaboration**

632 We consider a package labeling task, which instantiates the human-robot shared workspace collab-  
 633 oration domain of previous work [42, 53]. The human and the robot have different actions, i.e., the

634 human labels a package while the robot presses a stamp, and they share a set of goals, i.e., boxes to  
635 perform the task. The human and the robot cannot work simultaneously on the same object and the  
636 task finishes when all boxes are labeled and stamped.

637 We assume that the human picks a label for an object from a starting point and moves towards that  
638 object. Different boxes require different labels, thus we model the human as attempting to reach the  
639 box corresponding to the label they picked up, regardless of the robot’s actions. On the other hand,  
640 the robot can switch its goal while moving, since stamping can be performed on any goal object  
641 with the same tool. This domain is more complex than the shared control teleoperation task because  
642 it includes manipulating a sequence of objects, rather than reaching a single object, and the objects  
643 are in disjoint workspace regions.

644 As in the shared control teleoperation task, the robot reasons over the human goal by treating the  
645 human as noisily-optimal. However, unlike in shared control teleoperation, the robot attempts to  
646 avoid the goal intended by the human.

647 The scenario parameters consist of the locations of three goal objects in a larger, disconnected  
648 workspace. We set the workspace boundaries to the quadrants of the L-shaped table in Fig. 1 that are  
649 reachable by both the human and the robot arm. We model the human as moving to their goal while  
650 avoiding obstacles by solving a softmax MDP. The objective  $f$  is again the time to task completion  
651 since we wish to find challenging scenarios.

652 We choose two sets of measures  $m$  described below:

653 **Minimum distance between goal objects and maximum wrong goal probability:** We adopt the  
654 minimum distance measure from the shared control teleoperation domain in previous work [1].  
655 Furthermore, one of the failure scenarios found in that work was caused by incorrect inference of  
656 the human goal by the robot. Thus, we set as our second measure the maximum probability that is  
657 assigned to the wrong goal by the robot during the task, to search for potential failures in which the  
658 robot actually infers the human goal correctly.

659 **Robot path length and total wait time:** In the shared workspace collaboration task that we con-  
660 sider, there are two main sources of delay: the robot needing to move across the two workspaces to  
661 reach different goals, and the wait time caused due to both the human and the robot wanting to work  
662 on the same goal. Hence, we choose the path length of the robot and the total wait time as the two  
663 measures to see how the team performance changes as these are varied.

## 664 E Human and Robot Policies

### 665 E.1 Robot Policy

666 We adopt the robot policy defined in prior HRI works [51, 42] that introduced the domains consid-  
667 ered in this paper. In both domains in this paper, the robot solves a POMDP with human goal as the  
668 latent variable. As in prior work [51], the robot assumes that the human is stochastically optimal  
669 and updates its belief based on observed human actions. It performs hindsight optimization to cal-  
670 culate the values and update the belief in real-time, followed by a first-order approximation to select  
671 the optimal action that maximizes the Q-value. In both domains, we follow the cost function def-  
672 inition in the corresponding prior work [51, 42], which makes the resulting optimal value function  
673 proportional to the distance to the goal and the optimal policy a straight line.

674 We briefly discuss the specifics of the robot policy in the two domains below. In both domains,  
675 the robot action is computed as the twist that should be applied to its end effector, which is then  
676 converted to the required joint velocities by inverse kinematics computation.

#### 677 E.1.1 Shared Control Teleoperation

678 In shared control teleoperation, the human provides an input action to the robot. The Q-value of this  
679 action is defined as the sum of the cost incurred while executing the action and the value at the new

680 position after action execution. The robot’s belief is then updated based on the difference between  
681 the value and the Q-value at the current position corresponding to each goal.

682 Hindsight optimization followed by first-order approximation results in the robot’s assistive action  
683 being a weighted average of the straight-line paths to each goal, weighted by the corresponding  
684 probabilities assigned to them in the belief.

685 In App. G.1, we consider a different robot policy called policy blending [52]. The robot fully follows  
686 the user inputs while updating its belief like before. Once the probability assigned to a goal is higher  
687 than a threshold, the robot takes over and moves to the predicted goal.

### 688 E.1.2 Shared Workspace Collaboration

689 In shared workspace collaboration, the human acts independently. Hence, we maintain two sets  
690 of value functions - one for the human and one for the robot. We calculate the human Q-value as  
691 the sum of the cost of executing the current action and the value at the new position after action  
692 execution, similar to the shared control teleoperation domain. The robot’s belief is updated based  
693 on the difference between human value and human Q-value at the current position corresponding to  
694 each goal.

695 We track the constraints on the robot’s goals with the feasible goal-set formulation from prior  
696 work [42]. For each potential human goal, the robot maintains a set of goals that it has not worked  
697 on and is different from the human goal. The goal set can be empty for some candidate human goals  
698 if the robot has finished working on all other goals. The robot then treats all the goals that it has  
699 not worked on as the feasible goal set corresponding to that human goal. For action calculation, the  
700 robot creates a mapping from each human goal to a corresponding goal-to-go, which is the goal with  
701 minimum value (the closest goal) in the corresponding feasible goal set.

702 The robot’s action is based on the robot’s value functions. Since we assume that the robot acts  
703 optimally, we do not explicitly calculate these values and simply assume a straight-line path to each  
704 goal. Hindsight optimization followed by first-order approximation once again results in the robot’s  
705 action being a weighted average of optimal actions towards each goal-to-go.

706 Specifically, let  $b(g)$  be the probability assigned to goal  $g$  and let  $F(g)$  be the goal-to-go correspond-  
707 ing to human goal  $g$ . Then, the weight corresponding to goal  $g'$  is given by  $\frac{b(g')}{\sum_{g:F(g)=g'} b(g)}$ .

## 708 E.2 Human Policy

### 709 E.2.1 Shared Control Teleoperation

710 In shared control teleoperation, we search for human policy parameters in the form of noise added  
711 to the waypoints from the starting location to the intended goal location. The human policy keeps  
712 track of the waypoints and computes the waypoint-to-go and the corresponding velocity based on  
713 the current position of the robot arm.

### 714 E.2.2 Shared Workspace Collaboration

715 In shared workspace collaboration, the human moves independently towards the goal and avoids  
716 obstacles on the way. We model the human policy through a softmax MDP whose values are pre-  
717 computed before simulating the scenario.

718 First, we discretize the space in which the human can move into a grid with cell sizes equal to the  
719 size of the goal object so that each goal is in one cell. We treat these cells as the states of the MDP  
720 and allow the human to move to any neighboring cell, receiving a reward of either  $-0.01$  for moving  
721 to an orthogonally adjacent cell,  $-0.01\sqrt{2}$  for moving to a diagonally adjacent cell,  $-1$  for moving  
722 into an obstacle, or  $1$  for moving into a goal cell. We set the discount factor to  $0.9999$  and perform  
723 softmax value iteration [54] with a softmax temperature of  $0.001$  to compute the Q-values for each  
724 state-action pair.

725 Since we have three goals in a scenario, we compute three sets of Q-values, one corresponding to  
726 each goal. Each value iteration instantiation treats the scenario’s other goals as obstacles.

727 During simulation, the human policy converts the current location of the human into the grid cell it  
728 belongs to, chooses the next grid cell based on the Q-values corresponding to the current goal, and  
729 returns the velocity required to move to the center of the next cell.

730 In App. G.2, we consider a new setting in which we search over two human model parameters: the  
731 inverse of softmax temperature (higher values result in a more rational human) and a multiplier to  
732 the velocity (higher multiplier makes the human move faster).

## 733 **F Implementation Details**

734 We implement surrogate assisted scenario generation in a server-client framework. The server sim-  
735 ulates a given scenario in OpenRAVE [65] while the client executes QD search to generate new  
736 scenarios.

### 737 **F.1 Scenario Simulation**

738 We adapt the scenario simulation code from the open-source implementation of shared autonomy  
739 via hindsight optimization [66] to include the feasible goal set formulation for the shared workspace  
740 collaboration domain (described in App. E.2) and to simulate generated scenarios instead of a fixed  
741 one.

742 We start a flask server that waits for the client to run QD search and send solutions to evaluate. Once  
743 we receive a candidate solution, we pass it through the MIP solver and instantiate the objects, the  
744 robot, and the human in the OpenRAVE simulator.

745 We discretize the simulation into *ticks*, with each tick being divided into three phases that are ex-  
746 ecuted in sequence: human action selection, robot action selection, and environment simulation.  
747 Human action selection and robot action selection follow the policy given in App. E.2 and App. E.1  
748 respectively. In the environment simulation phase, the actions are executed, moving the human and  
749 the robot to a new state.

750 The shared control teleoperation task executes these phases in a loop until the robot reaches the  
751 intended human goal or the time limit of 10 seconds is reached.

752 Since the shared workspace collaboration task consists of multiple steps, the human and the robot  
753 policies are wrapped into state machines. The human state machine has five states: a) *moving to*  
754 *a goal*; b) *waiting for space*; c) *working on a goal*; d) *resetting*; e) *done*. The human is initially  
755 in *moving to goal* state and simply selects actions according to the human policy. Once a goal is  
756 reached, the human waits till the goal is free to work on (*waiting for space*) and then starts working  
757 on the goal (*working on a goal*). Once the work is complete, the human switches to the terminal  
758 state, *done*, if that was the last goal or moves back to the initial position (*resetting*). To simulate  
759 working on the goal and moving back to the initial position, we simply pause the human for a  
760 specified amount of time. After the reset, the human starts moving to the next goal (*moving to goal*).

761 The robot state machine has six states: a) *moving to a goal*; b) *replanning*; c) *waiting for space*; d)  
762 *working on a goal*; e) *resetting*; f) *done*. The state transitions are similar to those of the human state  
763 machine, except for the *moving to a goal* state. Since the robot can get into configurations close to  
764 self-collision or joint limits when following a straight line path, it needs to replan back to the start  
765 before moving again. We simulate this by switching to *replanning* state, moving the robot back to  
766 its initial position, and then switching back to *moving to a goal* state.

767 The shared workspace collaboration task ends either after 100 seconds or after both the human and  
768 the robot reach the *done* state.

769 **F.2 QD Search**

770 We implement QD search on the client by modifying the pyribs library [67] and the open-source  
771 code for DSAGE [21] to match Algorithm 1.

772 We implement the inner loop through a pyribs *scheduler* that interfaces a QD algorithm via two  
773 functions: *ask*, which outputs candidate solutions from the algorithm, and *tell*, which accepts the  
774 corresponding objective and measures, adds them to the archive, and updates the algorithm param-  
775 eters. The scheduler interfaces CMA-MAEGA and CMA-MAE for DSAS and SAS respectively.  
776 The inner loop runs fully on the client, exploiting the surrogate model described in App. B.

777 We then select a set of solutions from the surrogate archive and send it to the simulation server for  
778 evaluation. The objective and measures obtained from the simulation are returned by the server,  
779 which we add to the ground-truth archive and the dataset.

780 For baselines, we use the existing implementation of CMA-MAE and MAP-Elites in pyribs. Addi-  
781 tionally, for ease of execution, we implement Random Search similar to a QD algorithm in the pyribs  
782 framework. It simply returns a batch of uniformly randomly sampled candidate solutions whenever  
783 requested. Since these baselines do not leverage a surrogate model, the candidate solutions are  
784 always sent to the simulation server for evaluation.

785 To include objective regularization, we maintain two archives, the *final archive* that retains solutions  
786 maximizing the unregularized objective, and the *training archive*, which maintains scenarios that  
787 maximize the regularized objective to guide the QD search. The pyribs scheduler interfaces with  
788 the *training archive*, while solutions are directly added to the *final archive*. For surrogate assisted  
789 algorithms, the surrogate archive acts as the *training archive* while the ground-truth archive acts as  
790 the *final archive*.

791 We include the search details specific to the domains below.

792 **F.2.1 Shared Control Teleoperation**

793 In shared control teleoperation, we search over the  $\langle x, y \rangle$  coordinates of two goal objects and five  
794 noise variables that define the human path towards the goal, creating a 9-dimensional search space.

795 We define the measures as the distance between the goals  $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ , and the vari-  
796 ation in human input  $\sqrt{\sum_{i=1}^5 \theta_{h,i}}$ , where  $\theta_h$  refers to the five noise parameters in the generated  
797 solution. Following prior work [1], we assume the ranges of the measures to be  $[0, 0.32]$  for the  
798 distance and  $[0, 0.112]$  for variation, and create an archive with  $25 \times 100$  cells.

799 We adopt the hyperparameters for MAP-Elites from prior work [1], setting the standard deviation  
800 of perturbation,  $\sigma$ , to 0.01 for parameters corresponding to the goal coordinates and 0.005 for those  
801 corresponding to the human noise. For CMA-MAE, SAS, and DSAS, we set the initial standard de-  
802 viation for CMA-ES,  $\sigma_0$ , to 0.01, archive learning rate,  $\alpha$ , to 0.1, and minimum acceptance threshold,  
803  $min_f$ , to 0. We set all other hyperparameters to their default values defined in pyribs.

804 **F.2.2 Shared Workspace Collaboration**

805 In shared workspace collaboration, we search over the  $\langle x, y \rangle$  coordinates of three goal objects, cre-  
806 ating a 6-dimensional search space.

807 The four measure functions in our experiments are defined as follows:

- 808 1. Minimum distance between goal objects (archive range  $[0.05, 0.32]$ ; discretized into 27  
809 archive cells):  $\min_{i \neq j} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$
- 810 2. Maximum wrong goal probability (archive range  $[0.35, 1]$ ; discretized into 65 archive  
811 cells): Let  $b^{(max)}(t)$  be a function that returns the highest probability assigned by the robot  
812 to a goal other than the true human goal at time  $t$ . Maximum wrong goal probability is  
813 defined as the maximum value attained by  $b^{(max)}(t)$  during the scenario:  $\max_t b^{(max)}(t)$ .

Table 2: QD-score at the end of 10,000 evaluations.

	Shared Autonomy	Collaboration I	Collaboration II
DSAS	<b>21,400.33 ± 45.91</b>	106,874.93 ± 844.00	<b>19,261.95 ± 182.57</b>
SAS	21,043.49 ± 40.08	<b>112,962.22 ± 572.96</b>	<b>18,733.82 ± 182.40</b>
CMA-MAE	17,972.31 ± 74.71	87,399.75 ± 1,085.14	15,612.29 ± 284.34
MAP-Elites	11,757.84 ± 358.31	67,731.48 ± 576.30	<b>18,435.18 ± 398.87</b>
Random Search	9,647.24 ± 24.94	62,376.62 ± 200.68	13,856.14 ± 156.67

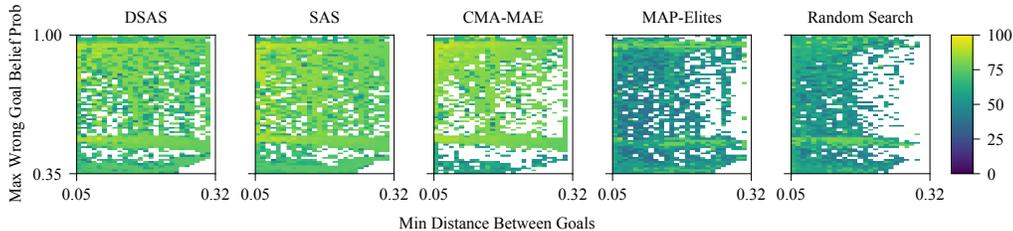


Figure 8: Comparison of the final archive heatmaps in the collaboration I domain.

- 814 3. Robot path length (archive range  $[1, 5]$ ; discretized into 20 archive cells): Let the robot's  
815 trajectory in the scenario be a function  $\tau : [0, 1] \rightarrow \mathbb{R}^2$ , with  $\tau(0)$  and  $\tau(1)$  denoting the  
816 coordinates of the start and end-points respectively. The robot path length is defined as the  
817 length of this trajectory:  $\int_0^1 \|d\tau\|_2$ .
- 818 4. Total wait time (archive range  $[0, 5]$ ; discretized into 50 archive cells): Let  $w(t)$  be a func-  
819 tion that returns 1 when either the robot or the human state machine is in *waiting for space*  
820 state (see App. F.1) and 0 otherwise. Total wait time is defined as  $\int_0^T w(t)dt$ , where  $T$  is  
821 the total scenario time.

822 Note that we approximate the integrals with discrete sums of the corresponding values at each sim-  
823 ulation tick.

824 We tuned the initial standard deviation for CMA-ES,  $\sigma_0$ , in the case of CMA-MAE, SAS, and DSAS  
825 and set it to 1. We also tuned the perturbation standard deviation,  $\sigma$ , for MAP-Elites and set it to 0.1.  
826 We set  $\alpha = 0.1$ ,  $min_f = 0$ , and all other hyperparameters to the default values provided in pyribs.

827 In the new setting described in App. G.2, we add two additional parameters to the search: the  
828 inverse of softmax temperature (higher values result in a more rational human) and the coefficient  
829 of velocity (higher coefficient makes the human move faster). We limit these parameters to ensure  
830 that the scenarios are not bottlenecked by an unrealistically slow or irrational human.

## 831 G Additional Results

832 We tabulate the results from our experiments in Table 2. We also show the final archives in the  
833 collaboration I (Fig. 8) and collaboration II (Fig. 9) domains.

834 We observe that the archives generated by DSAS and SAS are more densely packed compared to  
835 other algorithms in collaboration I. In collaboration II, we see that CMA-MAE, SAS, and DSAS  
836 find fewer solutions in the bottom left part of the archive (mostly corresponding to all goal objects  
837 in one workspace region) compared to MAP-Elites and random search, but find more and higher  
838 quality solutions in other parts of the archive which requires placing the goals in multiple regions.

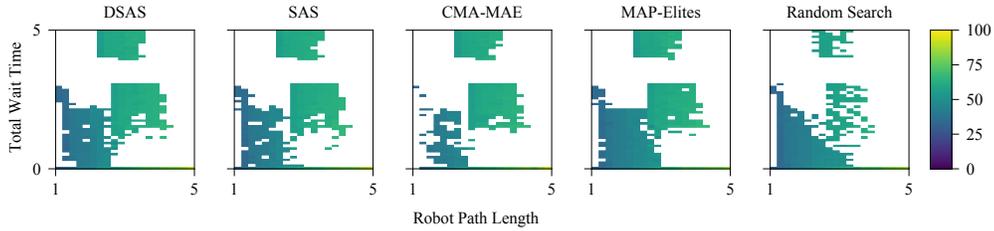


Figure 9: Comparison of the final archive heatmaps in the collaboration II domain.

Table 3: QD-score at the end of 10,000 evaluations.

	Teleoperation (Policy Blending)	Collaboration I (Human Policy Search)
DSAS	<b>41, 249.88 ± 205.56</b>	106, 573.18 ± 1, 461.34
SAS	40, 726.15 ± 300.61	<b>120, 789.83 ± 1, 378.82</b>
CMA-MAE	33, 797.07 ± 1, 455.82	<b>120, 687.02 ± 2, 959.76</b>
MAP-Elites	24, 151.97 ± 836.97	81, 006.04 ± 2, 483.23
Random Search	19, 850.68 ± 184.97	65, 513.84 ± 334.12

### 839 G.1 Additional Setting: Shared Control Teleoperation with Policy Blending

840 The QD formulation for scenario generation is independent of the robot and human policies. Here,  
 841 we show scenario generation with a new robot policy, policy blending (App. E.1), in the shared  
 842 control teleoperation domain without any modifications to the QD hyperparameters or the surrogate  
 843 model architecture.

844 Table 3 shows the QD-score at the end of 10,000 evaluations. We see that the surrogate assisted  
 845 algorithms outperform other algorithms, showing that these algorithms can work across multiple  
 846 robot policies. Note that the maximum time for a scenario was set to 20s, so the QD-scores are  
 847 around twice as large as in the main shared teleoperation experiments.

### 848 G.2 Additional Setting: Shared Workspace Collaboration with Human Policy Search

849 In the main shared workspace collaboration experiments, the scenario was only parameterized by  
 850 object locations. However, as described in our problem formulation, scenario parameters can also  
 851 include parameters of the human model. Here, we perform an additional experiment in which we  
 852 search for human model parameters in addition to the object locations to find failures in the collab-  
 853 oration I domain. We add two more scenario parameters related to human speed and rationality as  
 854 described in App. F.2 and run the QD algorithms with no other changes to the hyperparameters.

855 We tabulate the QD-scores in Table 3. We see a small increase in the QD-scores of all algorithms  
 856 compared to the main experiments (Table 2), since the QD search can now control the human policy  
 857 to cause failures. Surprisingly, CMA-MAE performs similar to SAS. We hypothesize that this is  
 858 caused by the sensitivity of the scenario outcomes to the human model parameters: Changes to  
 859 human speed or rationality affect the human trajectory much more than changes to goal locations.  
 860 Hence, predicting the trajectory and scenario outcomes is much harder in this setting compared to  
 861 the main experiments. Thus, CMA-MAE, a model-free QD algorithm, performs as well as SAS and  
 862 outperforms DSAS.

863 However, the failures broadly fell into the same categories as those found in the main experiment.  
 864 We hypothesize that this results from the bounds of the human policy parameters. Rational and  
 865 fast human actions allow the robot to accurately predict the human’s goal, leading to fast scenario  
 866 completion. On the other hand, we have set the bounds on the parameters to not allow QD search to

867 make the human unrealistically slow or irrational. Hence, the failures found in this experiment are  
868 similar to those found with a fixed human policy.

### 869 **G.3 Ablation: Effect of Objective Regularization**

870 In Sec. 4, we proposed objective regularization as a way to guide QD search towards valid workspace  
871 configurations. While objective regularization benefits general QD search, we note that surrogate  
872 assisted methods like DSAGE inherit additional benefits. As the surrogate model makes predictions  
873 for all possible scenarios, and not only scenarios satisfying the workspace constraints, the QD search  
874 that exploits the surrogate model can move towards high-magnitude inputs in invalid regions of the  
875 scenario space when these inputs result in high objective values. Objective regularization helps pre-  
876 vent QD algorithms from exploiting errors in the surrogate model at extreme regions of the scenario  
877 parameter space.

878 To test the effect of objective regularization on performance, we choose the collaboration I domain  
879 and run 10 trials of DSAS, SAS, CMA-MAE, and MAP-Elites without objective regularization.  
880 Hence, due to numerical errors resulting from exploiting errors in the surrogate model, none of the  
881 SAS or DSAS runs without objective regularization could be completed.

882 We compare the results of MAP-Elites and CMA-MAE runs with their corresponding runs from the  
883 previous section that included objective regularization. Pairwise t-tests showed that MAP-Elites per-  
884 formed similarly with and without regularization, while CMA-MAE performed significantly worse  
885 without objective regularization ( $t = -7.08, p < 0.001$ ). We attribute this to the fact that perturba-  
886 tions of existing solutions in MAP-Elites are not guided by the objective values. On the other hand,  
887 CMA-MAE guides the search based on the objective improvements of the sampled solutions; hence  
888 objective regularization has a significant effect on performance.

## 889 **H Additional Real World Scenarios**

890 *Incorrect human goal inference with limited effect on robot motion (Fig. 5b):* We select a scenario  
891 from the archive generated by SAS with a relatively average scenario time of 77s and a very high  
892 maximum wrong goal probability of 0.9.

893 The human finishes working on G1 and the robot on G2. As the human moves towards G2, the robot  
894 incorrectly thinks that the human is moving to G3, which is near the optimal path to G2, causing  
895 the robot to slow down in anticipation of the human motion. After the human reaches G2, the robot  
896 continues moving to G3. Hence, the incorrect prediction does not affect the overall scenario time  
897 much.

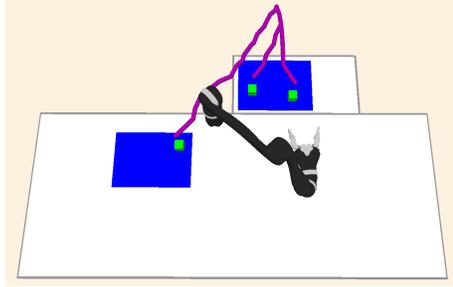
898 *Long wait time due to both teammates needing to work on the same goal (Fig. 5d):* Finally, we select  
899 a scenario from a DSAS archive in the collaboration II domain that has a high human and robot wait  
900 time.

901 This scenario was simple, albeit unanticipated. The human goes to G1, followed by G2, while the  
902 robot goes to G2, followed by G1. The team coordinates smoothly until both agents need to work  
903 on G3 to finish the task, causing a delay.

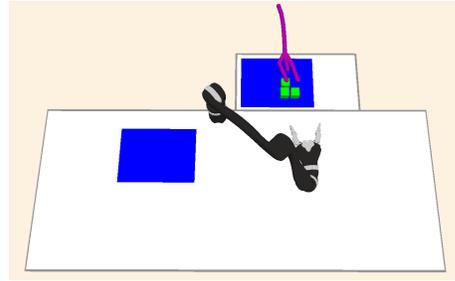
## 904 **I Scenarios with High Team Performance**

905 In addition to finding failures, QD scenario generation can also find scenarios that are ideal for  
906 human-robot collaboration. As an example, we modified the objective function in the collaboration  
907 I domain to  $100 - T$ , with  $T$  being the scenario completion time that has a maximum value of 100s.  
908 We ran SAS, which performed the best in this domain, and visualized example scenarios. We found  
909 two main types of success scenarios:

910 *Objects placed far apart to avoid confusion (Fig. 10a):* The first type of success involved placing  
911 the objects far apart to allow accurate goal inference. However, placing them too far would require



(a) High Team Performance Scenario 1



(b) High Team Performance Scenario 2

Figure 10: Examples of scenarios with high team performance. The purple line shows the simulated human path.

912 the human and the robot to move a lot, delaying completion. This scenario balanced these trade-offs,  
 913 leading to a relatively short robot path length of 1.7m, a low maximum wrong goal probability of  
 914 0.4, and a fast completion time of 38s. The resulting goal completion order also avoided the failure  
 915 found in Fig. 5d.

916 *Objects placed close together to quickly change goals (Fig. 10b):* The second type of success ignored  
 917 making goal inference easier but instead made it easier for the robot to correct itself if required.  
 918 Since the goals are close to each other, the robot can start moving towards them irrespective of  
 919 human actions. Once the human starts working on a goal, the robot can quickly switch to a different  
 920 goal. Despite having a high maximum wrong goal probability of 0.8, this scenario only took 31s to  
 921 complete.