

A Appendix

Code Availability: We will make our code available upon acceptance.

The organization of the appendix is as follows:

1. Subsection A.1 describes the details of the submodular maximization function we have used in our experiments.
2. Subsection A.2 presents the environment setups and exact parameters we have used for our simulations.
3. Subsection A.3 provides the exact details for the network configurations used in our experiments.
4. Subsection A.4 presents all the simulation results for all tasks and network configurations along with our methods' performance superiority.
5. Subsection A.5 provides ablation studies over the hyperparameters and simulation parameters.
6. Subsection A.6 presents the complexity analysis and optimality bounds of our allocation policies.
7. Subsection A.7 presents our 5G network data collection setup.

A.1 Submodular Maximization Parameters

This section provides the exact parameters and functions used in the submodular maximization method given in Section 4.

A.1.1 Informativeness Function

The informativeness function \mathcal{U} measures the informativeness of each robot i in the fleet. To measure the informativeness of the robots, we use a weighted combination of two functions. The first function measures the uncertainty of the robot policy π_R in the environment, and the second function measures the risk of the robot i violating the constraints \mathbf{C} . Then, the informativeness function can be defined as follows:

$$\mathcal{U}(i) = \alpha_{\mathcal{U}} \mathcal{U}_{\text{unc}}(i) + (1 - \alpha_{\mathcal{U}}) \mathcal{U}_{\text{risk}}(i). \quad (7)$$

Here, $\alpha_{\mathcal{U}}$ is a hyperparameter that controls the weight of the uncertainty in the overall informativeness measure, and $\mathcal{U}_{\text{unc}}(i)$ and $\mathcal{U}_{\text{risk}}(i)$ are the uncertainty and risk functions of the robot i , respectively. When the robot is taking discrete actions, the uncertainty function $\mathcal{U}_{\text{unc}}(i)$ is defined as the entropy of the robot policy π_R , and when the robot is taking continuous actions, the uncertainty function is defined as the ensemble variance of the robot policy π_R [40]. The risk function $\mathcal{U}_{\text{risk}}(i)$ is defined as the likelihood of the robot i exiting the constraint space \mathbf{C} [15].

For both uncertainty and risk functions, if the value of the function for the robot i is below a certain threshold, we set it to zero. We define this threshold parameter as $\mathcal{U}_{\text{unc}}^{\text{thres}}$ and $\mathcal{U}_{\text{risk}}^{\text{thres}}$ for uncertainty and risk functions, respectively. We present specific $\mathcal{U}_{\text{unc}}^{\text{thres}}$ and $\mathcal{U}_{\text{risk}}^{\text{thres}}$ parameters for the experiments in Table 2.

A.1.2 Similarity Function

The similarity function \mathcal{S} measures the similarity between two robots i and j in the fleet. In our experiments, we utilize both the similarity between the robots and the similarity between the actions taken by the robots. More formally, the similarity function is defined as follows:

$$\mathcal{S}(i, j) = \alpha_{\mathcal{S}} \frac{s_i \cdot s_j}{\|s_i\| \|s_j\|} + (1 - \alpha_{\mathcal{S}}) \frac{a_i \cdot a_j}{\|a_i\| \|a_j\|} \quad (8)$$

where s_i and s_j are the states of the robots i and j , respectively; a_i and a_j are the actions taken by the robots i and j , respectively. $\alpha_{\mathcal{S}}$ is a hyperparameter that controls the weight of the state similarity in the overall similarity measure.

551 A.1.3 Constraint Violation Function

552 The constraint violation function \mathcal{C} measures the violation of the constraints by the robots. In our
 553 experiments, we used the constraint violation as an indicator function that returns α_C if the constraint
 554 is violated and 0 otherwise. The α_C is a parameter that controls the relative importance of the
 555 constraint violation in the overall objective function. In our experiments, we set $\alpha_C = 10000$
 556 to prioritize the robots with constraint violations. The constraint violation function is defined as
 557 follows:

$$\mathcal{C}(i) = \begin{cases} \alpha_C & \text{if } s_i \notin \mathbf{C} \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

558 Here, the \mathbf{C} refers to the safe states that the robot can operate without any human intervention. As
 559 the constraint function causes the system to prioritize the robots violating the constraints, in initial
 560 time steps, we set the $\alpha_C = -10000$ to ensure that the robots violating the constraints are not
 561 prioritized. This is because, in the initial steps, the robots explore the environment, and collecting
 562 more informative data is more critical than the constraint violations. We control the length of this
 563 period in which the constraint violating robots are not prioritized by the t_W parameter.

564 A.2 Experimental Setups and Parameters

565 As stated previously, we run simulations using four different environments: ANYmal, Allegro Hand,
 566 Humanoid, and Ball Balance. Each environment has its own defined tasks, success criteria, and
 567 constraint violations. For the ANYmal robot, a constraint violation occurs when there is excessive
 568 force on the robot’s knees, indicating that the robot has fallen on its knees, or when no force is
 569 exerted on the bottom of its toes, indicating that the robot has fallen on its torso. For the Ball
 570 Balance environment, a constraint violation occurs when the ball is no longer on the plate. In the
 571 Allegro Hand environment, a constraint violation happens when the cube is no longer in the robot’s
 572 hand. For the Humanoid environment, a constraint violation occurs when the robot’s position is
 573 below the termination height, indicating that the Humanoid has fallen down.

574 The definition of success is specific to each task. For instance, in locomotion tasks, success is
 575 achieved if the robot does not violate constraints and reaches a reward amount that exceeds a pre-
 576 defined reward threshold. For goal-specific tasks such as Ball Balance and Allegro Hand, success
 577 corresponds to reaching the goal state without violating constraints. For Ball Balance, a goal state
 578 may be one where the ball on the plate is moving within a radius smaller than the plate’s radius,
 579 indicating that the robot successfully managed to control and balance the ball. For Allegro Hand,
 580 the goal state may be defined as holding the cube stable after rotating it so that the red surface faces
 581 up. That is how a single success corresponds to different achievements depending on the specific
 582 tasks assigned to each robot.

583 For all experiments, the key parameters are fixed and do not depend on the allocation policies:
 584 $N_{\text{human}} = 5$, $N_{\text{robot}} = 100$, $T = 10,000$ time steps, $t_R = 5$ time steps and $t_T = 5$ time steps.
 585 The hyperparameters that vary depending on the task, along with the values that yielded the best
 586 performances, are provided below in Table 2. $|S|$ and $|A|$ are the dimensionalities of the state
 587 and action spaces, respectively, $\mathcal{U}_{\text{unc}}^{\text{thres}}$ and $\mathcal{U}_{\text{risk}}^{\text{thres}}$ are the uncertainty and risk threshold values below
 588 which the uncertainty and risk are treated as zero, t_W is the period during which constraints are
 589 not prioritized, allowing the robot policies to be improved by selecting informative robots rather
 590 than resetting failing robots in the first t_W time steps, *threshold* is the marginal increase threshold
 591 below which the robots are not prioritized, α_S is the parameter which controls the weight of the state
 592 similarity in the overall similarity measure, and α_U is the parameter that controls the weight of the
 593 uncertainty in the overall informativeness measure.

594 A.3 Network Configurations

595 Here, we explain the details of the network configurations used in our experiments. We have used
 596 four different network configurations to evaluate the adaptability of the allocation algorithms in
 597 different network conditions. Additionally, we show the connection probabilities in each network
 598 configuration in Figure 5. The network configurations are as follows:

Task	$ S $	$ A $	$\mathcal{U}_{\text{unc}}^{\text{thres}}$	$\mathcal{U}_{\text{risk}}^{\text{thres}}$	t_W	$threshold$	α_S	α_U
AllegroHand	88	21	0.53	0.12	1250	0.04	0.37	0.53
AnyMAL	48	12	0.19	0.49	1000	0.69	0.72	0.05
BallBalance	24	3	0.47	0.21	1750	0.51	0.98	0.46
Humanoid	108	21	0.18	0.20	2500	0.23	0.50	0.10

Table 2: **Simulation environment hyperparameters for each task.**

599 **Always:** Always is a simple network configuration where the probability of connection to all the
600 robots is set to 1. In this network configuration, our supervisor allocation problem is equivalent to
601 the Interactive Fleet Learning (IFL) problem presented in [15], where the supervisor can connect to
602 all the robots at all times.

603 **Mixed-Scarce:** Mixed-Scarce is a network configuration where the probability of connection to
604 robots can be set to two different values. In this network configuration, we first divided the robots
605 into two groups with ratios of 0.7 and 0.3. We then set the probability of connection to the robots
606 in the first group to 0.9 and the probability of connection to the robots in the second group to 0.1.
607 This network configuration is used to evaluate the adaptability of allocation algorithms when the
608 connectivity to the robots is heterogeneous. Ideally, the supervisor should allocate more resources
609 to the robots with higher connectivity to maximize the performance of the fleet.

610 **Ookla:** Ookla is a network configuration where the probability of connection to the robots is set
611 based on the Ookla cellular network performance data [51]. This dataset includes the download
612 speed, upload speed, and latency of the cellular network in different locations. We use the download
613 speed as the metric to determine the probability of connection to the robots. We first divided the
614 data collection points into a grid of 10×10 cells. We then calculated the average download speed of
615 the data collection points in each cell. After that, we log-normalize the average download speed of
616 each cell to be in the range of $[0.5, 1]$. We have set the lower bound to 0.5 to ensure that the robots
617 in the cell with the lowest download speed have a non-zero probability of connection. We then set
618 the probability of connection to the robots in each cell to be the normalized average download speed
619 of the cell. This network configuration is used to evaluate the adaptability of allocation algorithms
620 when the connectivity to the robots is based on real-world cellular network performance data, which
621 is heterogeneous and has a more complex structure than the Mixed-Scarce network configuration.

622 **5G Network:** 5G Network is a network configuration where the probability of connection to the
623 robots is set based on the real-world 5G network performance data. Please refer to Section A.7 for
624 more details on the data collection process. The collected data was divided into 100 groups, with
625 average latency and throughput calculated for each group and normalized to a value between 0.015
626 and 1. A lower bound of 0.015 ensures a non-zero connection probability for robots with the lowest
627 throughput and highest latency. Robots in groups with throughput below 0.4 and latency above 0.6
628 were assigned a normalized value of 0.015. The connection probability for each group corresponds
629 to the normalized average throughput and latency. This configuration evaluates the adaptability
630 of allocation algorithms to realistic, heterogeneous connectivity based on real-world 5G network
631 performance, which is more complex than other network configurations.

632 A.4 Numerical Results

633 In this section, we present the numerical values for all allocation policies and for all tasks under
634 each network configuration to demonstrate that our method outperforms the baseline algorithms in
635 all simulated scenarios, providing a novel approach to the supervisor allocation problem. We also
636 present the percentage performance differences between our methods (ASA and n-ASA) and other
637 methods. For better comparability, we exclude the random method from the comparison since it
638 does not include any prioritization and randomly selects the robots. We present all numerical results
639 recorded in the final timestep ($t = 10,000$) in Table 3 and the percentage differences in Figure 6.

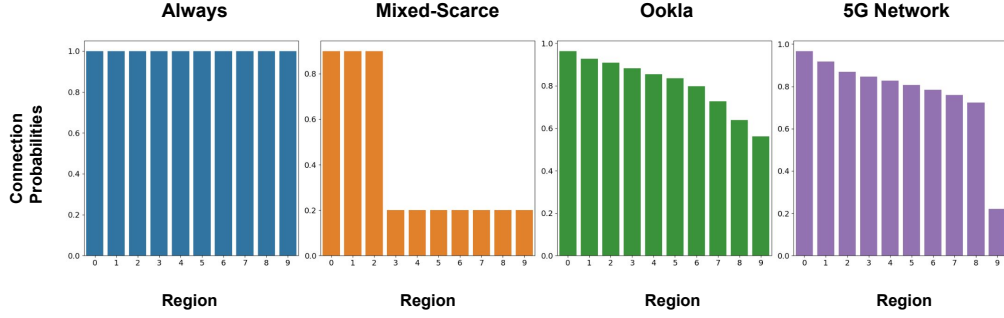


Figure 5: **Connection Probabilities for each Network Configuration.** This figure shows the connection probabilities of the robots in the fleet for each network configuration. For easier visualization, we have grouped 100 robots into 10 groups of 10 robots each and presented the average connection probability for each group. We can see that from the Always network to the 5G network, the connection probabilities of the robots get more heterogeneous. This heterogeneity in the connection probabilities is crucial for evaluating the adaptability of the allocation algorithms in different network configurations.

NETWORK	ALLOCATION POLICY	ALLEGROHAND		ANYMAL		BALLBALANCE		HUMANOID	
		RoHE	CUMULATIVE SUCCESS	RoHE	CUMULATIVE SUCCESS	RoHE	CUMULATIVE SUCCESS	RoHE	CUMULATIVE SUCCESS
ALWAYS	RANDOM	10.48	649.7	2.18	160.0	3.61	1796.0	0.01	1.67
	FT	11.10	2751.0	-	-	2.87	1437.0	-	-
	FE	10.03	2021.67	2.81	127.33	6.82	1185.0	1.04	296.67
	FD	13.48	3352.33	3.12	189.0	7.92	1499.67	1.93	424.67
	N-ASA (OURS)	16.30	5094.67	3.76	246.33	10.67	1802.67	2.22	503.33
	ASA (OURS)	14.87	5064.33	3.41	241.0	10.87	1785.67	2.28	530.0
MIXED-SCARCE	RANDOM	3.39	302.0	0.65	79.33	2.64	1319.0	0.00	0.67
	FT	4.05	1713.0	-	-	1.98	988.0	-	-
	FE	3.69	1616.33	0.89	110.33	3.54	1327.67	0.09	45.0
	FD	4.94	2031.0	2.75	1209.0	3.32	1253.33	0.33	159.33
	N-ASA (OURS)	8.15	3850.67	2.46	244.33	5.85	1580.67	1.95	499.67
	ASA (OURS)	8.16	3817.33	2.29	235.33	7.27	1621.0	1.85	491.0
OOKLA	RANDOM	8.45	576.33	1.97	164.67	3.57	1782.33	0.01	1.67
	FT	9.37	2583.33	-	-	2.77	1383.33	-	-
	FE	8.34	2072.33	2.22	126.0	7.46	1301.33	0.70	235.33
	FD	10.65	2986.33	2.76	186.67	6.45	1443.33	1.47	392.33
	N-ASA(OURS)	13.99	4694.33	3.20	240.0	10.96	1791.67	2.25	506.67
	ASA (OURS)	13.75	4936.0	3.11	225.67	11.41	1741.33	2.15	510.33
5G NETWORK	RANDOM	2.47	228.67	0.36	47.33	2.81	1401.33	0.00	0.00
	FT	4.75	2131.67	-	-	0.69	344.67	-	-
	FE	3.98	1789.67	1.07	137.0	3.01	1315.0	0.37	169.33
	FD	5.40	2213.0	1.40	200	2.75	1209.0	0.79	319.33
	N-ASA (OURS)	7.66	3705.33	1.61	246.33	7.45	1758.33	1.44	444.33
	ASA (OURS)	7.43	3658.0	1.63	239.33	6.39	1703.33	1.51	470.0

Table 3: **Numerical results for all network and task simulations:** We present RoHE and cumulative success values in the final timestep ($t = 10,000$) for 4 different environments under 4 different network configurations.

640 A.5 Ablation Studies

641 In this section, we conduct further experiments using our adaptive submodular allocation, ASA,
 642 policy to explore the following: (1) the sensitivity of the system to the ratio of the number of robots
 643 N_{robot} to the number of humans N_{human} (Figure 7), (2) the impact of varying the minimum inter-
 644 vention time t_T (Figure 8), and (3) the impact of changing the hard reset time t_R (Figure 9). Each
 645 experiment is averaged over three different random seeds, and the shaded regions correspond to one
 646 standard deviation. We plotted four different metrics: (1) cumulative success, (2) RoHE, (3) cumu-
 647 lative hard resets, and (4) cumulative idle time. Cumulative hard resets represent the total number
 648 of hard resets performed by human supervisors when the robots violate constraints. Cumulative idle
 649 time is the total time, in time steps, that robots remain idle while waiting for a hard reset.

650 **Number of Humans:** We tested the ASA policy to evaluate its sensitivity to different numbers of
 651 human supervisors (Fig. 7). Keeping the number of robots constant, we simulated scenarios with 1,
 652 5, 10, 25, and 50 human supervisors. In all simulated tasks, fleet performance was the worst when
 653 there was a single human supervisor due to insufficient human resources. Interestingly, the RoHE
 654 value was higher for the BallBalance task with a single human supervisor. This is because, with
 655 only one supervisor, most of their time is allocated to robots violating constraints. These constraint-

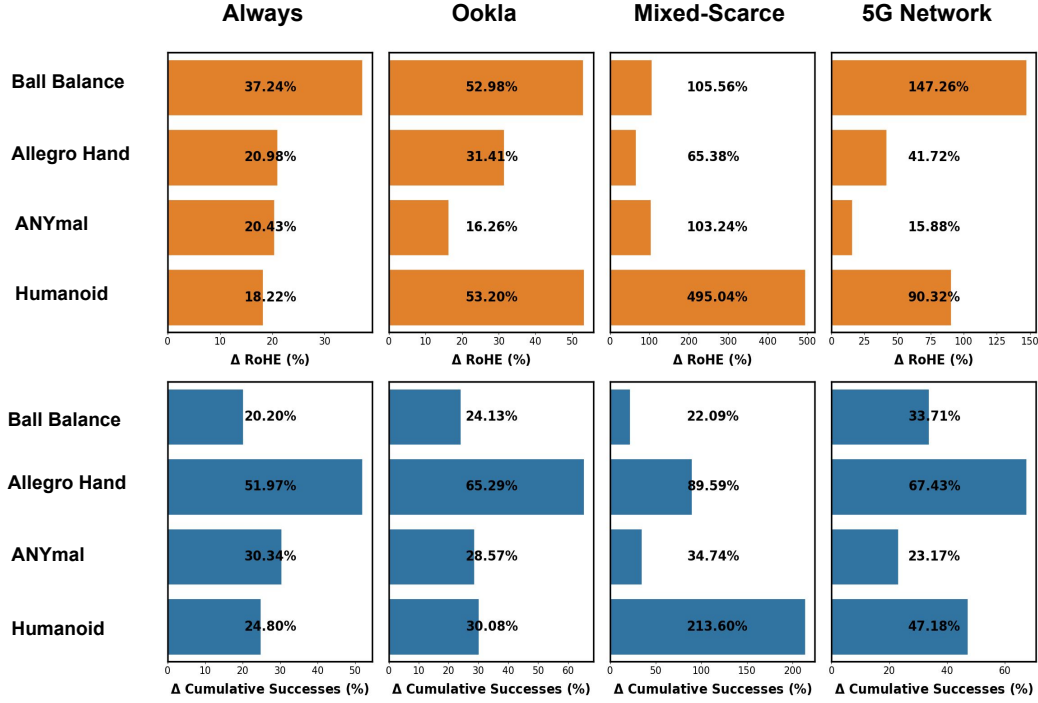


Figure 6: **Our ASA and n-ASA policies outperform other benchmarks across all environments and network combinations.** Here, the first set of bar plots represent the percentage difference in RoHE and the second set of bar plots represent the percentage difference in cumulative successes between our best method and the best baseline method.

violating robots, which would otherwise be idle if not teleoperated and reset, contribute more when they are actively managed by the supervisor. As the number of human supervisors increases, idle time decreases because more human resources are available, resulting in shorter idle periods before robots are teleoperated and reset. However, despite the cumulative success values rising with more supervisors, the RoHE values tend to decrease. This happens because allocating more humans doesn't always lead to a higher return on human effort. The most informative and important robots are already being selected, so adding more supervisors doesn't necessarily result in a significant marginal gain. Therefore, a low number of human supervisors is insufficient as robots remain idle for long periods and violate constraints more frequently, while a large number of supervisors creates a surplus and decreases efficiency.

Minimum Intervention Time: While keeping the number of robots fixed, we varied the minimum intervention time and ran our policy. We observed that when the minimum intervention time is very long, such as 100 or 500 time steps, the robot fleet performance significantly decreases. This is because human supervisors spend a lot of time teleoperating a single robot, which results in lower RoHE and cumulative success values, and a substantial increase in idle time. Conversely, when the minimum intervention time is very short, such as 1 time step, performance improves in terms of both RoHE and cumulative success for most tasks. This is because each human supervisor spends less time on a single robot and can attend to more robots within 10,000 time steps, thus enhancing overall fleet performance as the minimum intervention time decreases.

Hard Reset Time: Finally, we ran the ASA policy with different hard reset times. We observed that as the hard reset time increases, the fleet performance decreases. This is because it takes longer for human supervisors to reset the robots, resulting in fewer hard resets within 10,000 time steps. Consequently, the idle time increases, reducing the overall performance of the robot fleet.

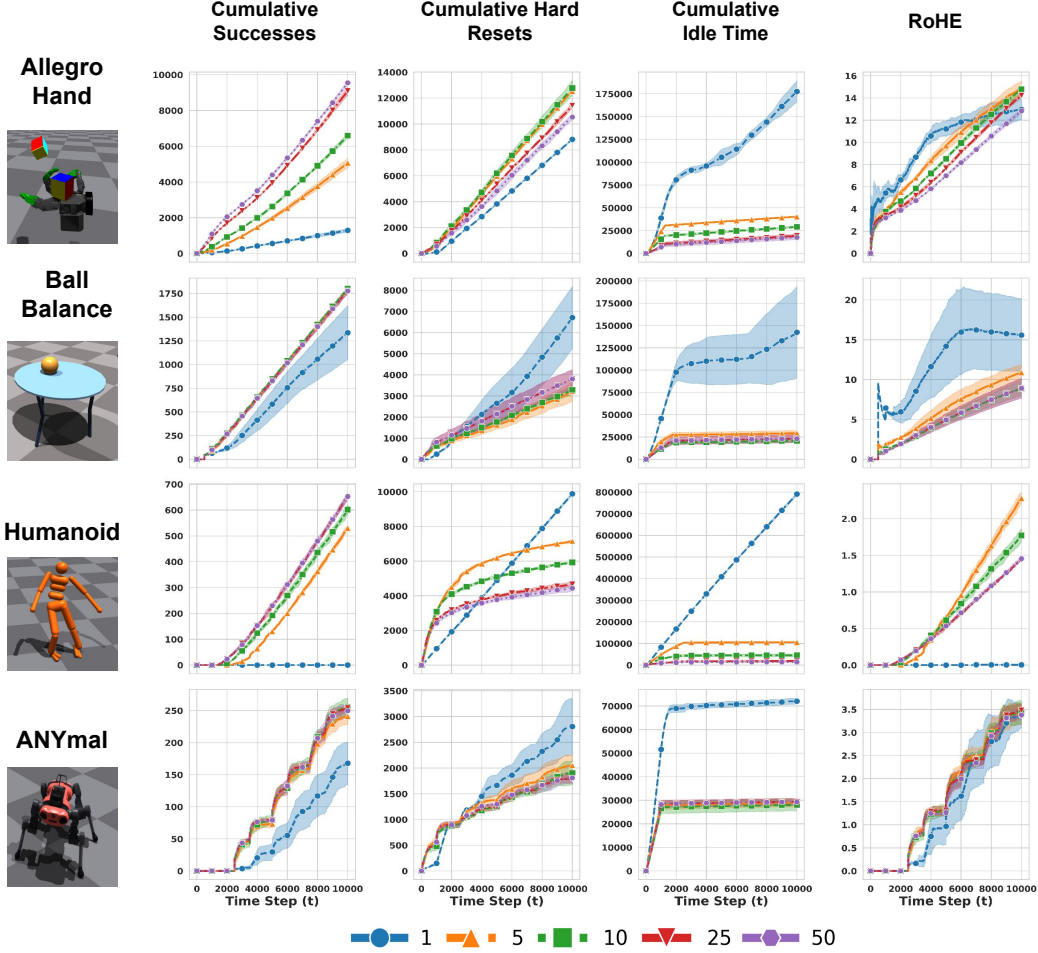


Figure 7: **Simulation results with different numbers of human supervisors:** ASA policy simulation results for each task under Always network configuration where $N_{\text{robot}} = 100$ but the number of human supervisors N_{human} vary.

A.6 Complexity Analysis and Optimality Bounds for Allocation Algorithms

Here we present the complexity analysis and optimality bounds for our allocation algorithms.

A.6.1 Complexity Analysis

We now explain the complexity of our allocation algorithms in terms of the number of robots N_{robot} , and the number of humans N_{human} in the system and function evaluations. As we have discussed in Section 4, our allocation algorithm is based on a greedy algorithm that selects the robots based on the stochastic submodular maximization objective. It is a well-known result that the number of function evaluations for the greedy algorithm is $O(N_{\text{robot}}N_{\text{human}})$. As both of our algorithms are based on the greedy algorithm, the computational and time complexities of our algorithms, ASA and n-ASA, are both $O(N_{\text{robot}}N_{\text{human}})$.

A.6.2 Optimality Bounds

To establish the optimality bounds for our allocation algorithms, ASA and n-ASA, we utilize the theoretical results from the stochastic submodular maximization and adaptive submodular maximization literature. Specifically, the greedy algorithm used in stochastic submodular maximization

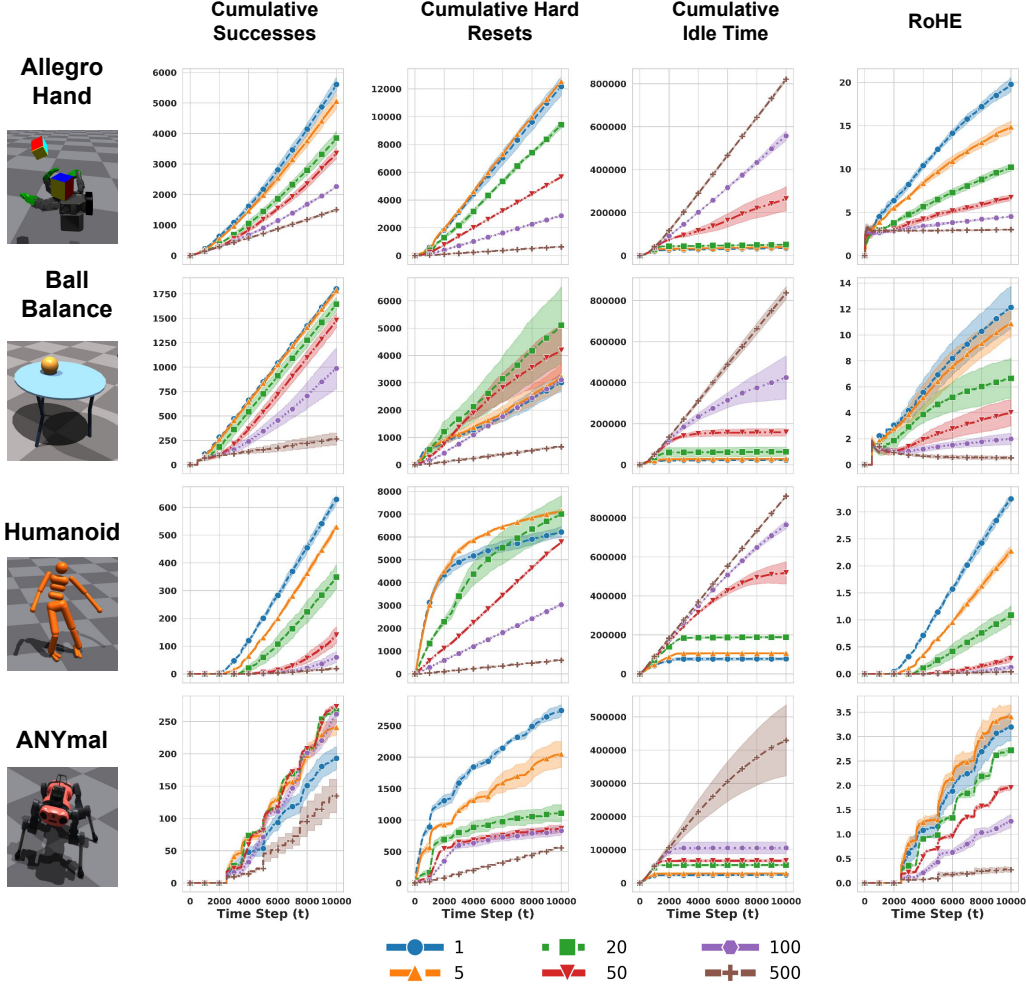


Figure 8: **Simulation results with different minimum intervention times:** ASA policy simulation results for each task under Always network configuration for $T = 10,000$ time steps where $N_{\text{robot}} = 100$, $N_{\text{human}} = 5$ and the minimum intervention time t_T varies.

is shown to approximate the optimal solution for the problem given in Equation 6. We now present the optimality bounds for each of our algorithms.

Optimality Bound for ASA: When the threshold threshold is set to 0 in Algorithm 1, the ASA algorithm is equivalent to the adaptive greedy algorithm for stochastic submodular maximization problem [47]. Golovin and Krause [47] show that the adaptive greedy algorithm achieves a $(1 - 1/e)$ -approximation to the adaptive optimal solution. Therefore, the ASA algorithm achieves a $(1 - 1/e)$ -approximation to the optimal solution for the problem given in Equation 6.

Optimality Bound for n-ASA: When the threshold threshold is set to 0 in Algorithm 1, the n-ASA algorithm is equivalent to the non-adaptive greedy algorithm for the stochastic submodular maximization problem [46]. Asadpour et al. [46] show that the non-adaptive greedy algorithm achieves a $(1 - 1/e)$ approximation to the optimal non-adaptive solution. Additionally, the optimal non-adaptive solution is a $(1 - 1/e)$ -approximation to the optimal adaptive solution [47]. Therefore, the n-ASA algorithm achieves a $(1 - 1/e)$ -approximation to the optimal non-adaptive solution and a $(1 - 1/e)^2$ -approximation to the optimal adaptive solution for the problem given in Equation 6.

A.7 Real World 5G Network Data

In addition to the simulated network connectivity data, we also evaluate our allocation policies using real-world 5G network connectivity data. We collected this data using two hardware components: a

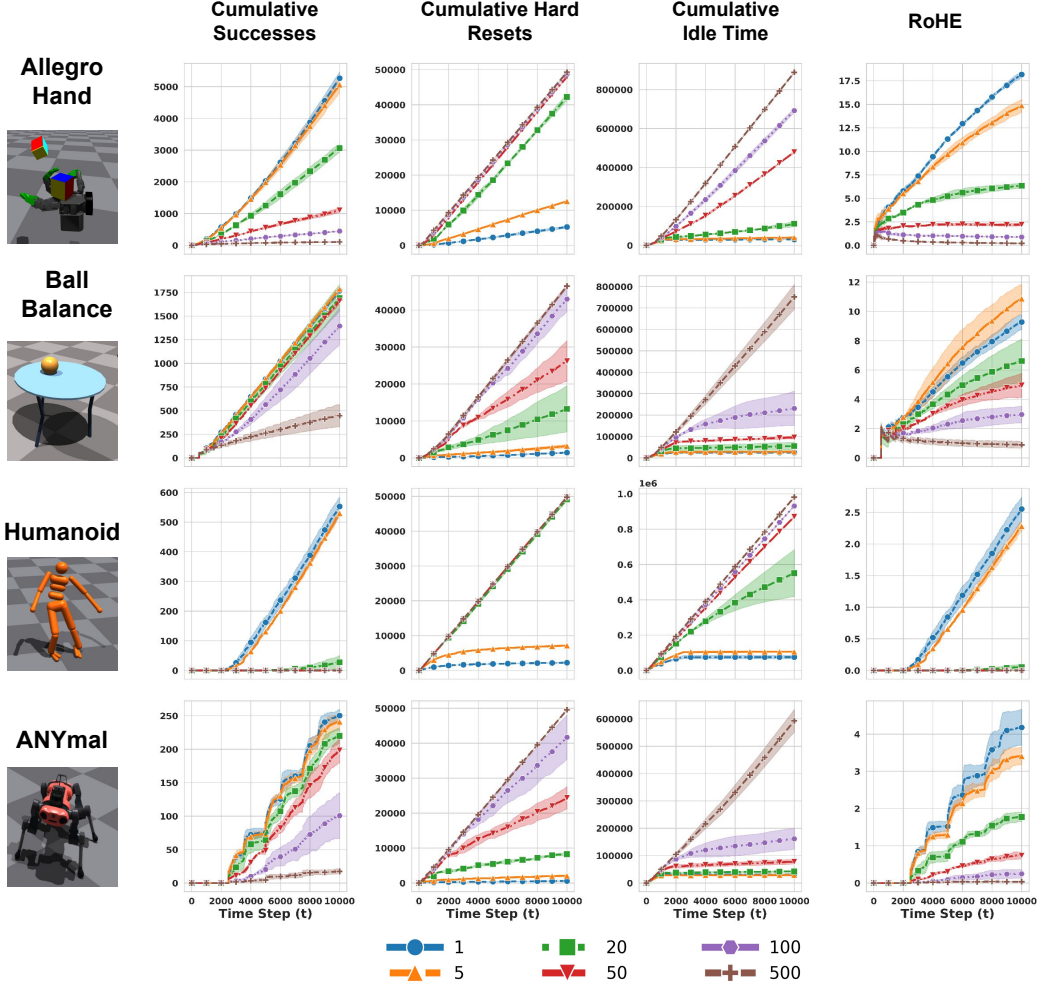


Figure 9: **Simulation results with different hard reset times:** ASA policy simulation results for each task under Always network configuration where $N_{\text{robot}} = 100$, $N_{\text{human}} = 5$ and the hard reset time t_R varies.

mobile edge device and a local server. The edge device, which can be a robot, a mobile phone, or a computer, acts as the connection client. The local server functions as the cloud. In our scenario, we consider the edge device to be the robot and the local server to be the cloud or the server from which human supervisors connect to the robots. The local server sends packets to the edge device, and the edge device responds with packets to confirm receipt. During this process, the local server calculates latency and throughput, saving this data to a local file. This continues for a predetermined data collection period of 24 hours. Two key aspects of this setup are: (1) the edge device is connected to a 5G cellular network, specifically 5G cellular provided by AT&T, and (2) it is mobile. This allows us to collect data anywhere, whether moving or stationary, for any desired period. To obtain data that realistically simulates human teleoperation connectivity, we collected data in a building where actual teleoperation and robotic tasks are conducted. After collecting the data, we divided and clustered it into 100 different groups. This division helps correlate the data with our fleet learning simulation environment, which has 100 robots in different locations. For each group, we calculated the average latency and throughput. We normalized the average values between 0 and 1, such that groups with high latency and low throughput values have a normalized value closer to 0, and groups with low latency and high throughput values have a normalized value closer to 1. Now that we have 100 different normalized values, we randomly assigned them to 100 simulated robots. We illustrate the data collection setup in Figure 10. Additionally, we present the average throughput and latency for each group as well as the connection probability for each grid cell in Figure 11.

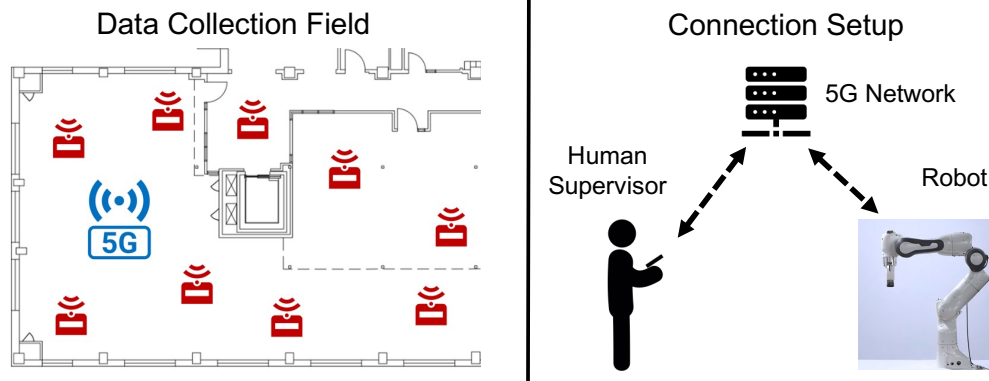


Figure 10: **Data Collection Setup for 5G Network.** We collected 5G network connectivity data from the real-world robotics laboratory floor, where the example floor plan is shown in the left figure. In the floor plan, the red devices represent the locations of the robots on the laboratory floor. We have collected the 5G network data from these locations using a 5G-enabled smartphone. Our data collection setup is shown in the right figure. For each location, we have established a connection between a human supervisor using a 5G-enabled smartphone and a robot server through a 5G base station and a 5G modem. We have collected various network parameters including throughput, latency, and signal strength for each location. We then processed this data to obtain the network connectivity information for the robots in our experiments.

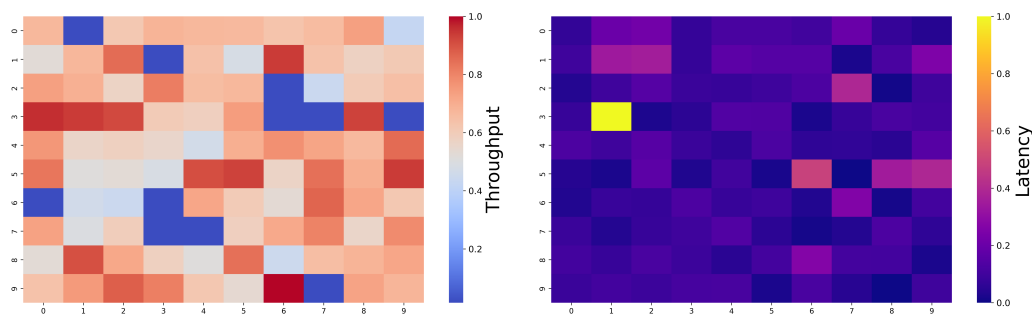


Figure 11: **5G Network Performance Metrics.** This figure shows the key performance metrics of the 5G network data collected from the real world. Here on the left, we show the average throughput for each group of robots. The throughput is normalized between 0 and 1, where 0 represents low throughput, and 1 represents high throughput. On the right, we show the average latency for each group of robots, where the latency is normalized between 0 and 1, where 0 represents high latency, and 1 represents low latency. We then use these metrics to determine the probability of connection to the robots in our experiments.