# A Rationale-Driven Team Plan Representation for Autonomous Intra-Robot Replanning*

Philip Cooksey[1] and Manuela Veloso[2]

***Abstract*— For autonomous multi-robot teams, the individual team members are tasked with completing their assigned tasks as defined by a team plan provided by a centralized team planner. However in complex dynamic domains, the team plans are generated by the team planner with assumptions due to the complexity of modeling the domain. Failures in execution are therefore inevitable for the team members, and as such, replanning will occur for the team. In this paper, we introduce a rationale-driven team plan representation that provides rationales on why actions were chosen by the team planner. During a failure, the individual team members autonomously use our described intra-robot replanning algorithm to select all applicable replan policies for a given rationale. We then describe a method to learn the predicted cost of each replan policy, given a state of the environment, in order for the individual robots to select the lowest costing replan policy to improve team performance.**

## I. Introduction and Related Work

Centralized multi-robot team planners in complex dynamic domains often have incomplete knowledge of their environments, have poorly modeled dynamics, and have simplified assumptions about their environments. These shortcomings manifest into execution failures by which the individual team members fail to successfully complete their assigned action(s). Intra-robot replanning introduced and formalized in [1] is an approach where the individual robot replans locally using the information provided by the team plan or by the environment to fix a team plan failure. In our previous work, we showed the benefits of intra-robot replanning in a competitive dynamic domain [2]. In this paper, we will focus on describing our rationale-driven team plan representation, our intra-robot replanning algorithm, and our method to learn the predicted cost of a replanning algorithm.

In the literature, the standard approach to team planning is to divide the problem into a hierarchy from abstract actions to concrete actions in the environment [3], [4], [5]. This division of computation makes the problem more solvable in complex domains. We follow the Skills, Tactics, and Plays (STP) hierarchy as described by [6]. Skills are concrete actions that are very specific to the domain. Tactics combine skills together in order to accomplish more complex tasks. Plays assign roles and a series of tactics, with their specific parameters, to guide the team of robots towards their goal.

*This work is supported in part by AFRL and DARPA under agreement #FA8750-16-2-0042, ONR grant N00014-09-1-1031, and AFRL grant number FA87501220291. The views and conclusions contained in this document are those solely of the authors.

[1]Philip Cooksey is with Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA. pcooksey@andrew.cmu.edu

[2]Manuela Veloso is with the School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA. veloso@cs.cmu.edu


Fig. 1. Three Light Autonomous Underwater Vehicles (LAUVs) from the Laboratório de Sistemas e Tecnologia Subaquática (LSTS) in Porto, Portugal that were used to gather the depth data in Figure 3.

Plays are essentially the team plan and the series of tactics are the actions that must be executed by each robot for the team plan to be successful. For our purposes, STP provides a useful division between the team plan and tactics executed by the individual robots.

We assume the team plans are generated in a centralized fashion and the individual robots execute the team plan, i.e, their assigned tactics, in a decentralized fashion. We also assume that the individual robots can communicate with the centralized team planner, but that the cost of communicating with the centralized team planner is high [7]. To better illustrate such a domain, we describe an example domain using Autonomous Underwater Vehicles (AUVs), see Figure 1.

As a quick introduction into the AUV domain, we assume that a centralized controller offshore is generating team plans for a team of AUVs. To clarify, we are not concerned with the method by which the team plans are generated in this paper; we will assume human operators create the team plans in this domain. We are concerned with the team plan representation, specifically the information included in the team plan for the individual autonomous robots. We assume the AUVs are placed into the ocean having already received the team plan, and from then on the only communication between the AUVs and the centralized controller is through a satellite link which is very expensive and time consuming to use. We also assume the AUVs can quickly and cheaply communicate with each other over wifi if they are close enough in proximity and are not underwater. To reiterate, the AUVs cannot communicate underwater nor can they receive a GPS signal for localization. Therefore, the AUVs will often

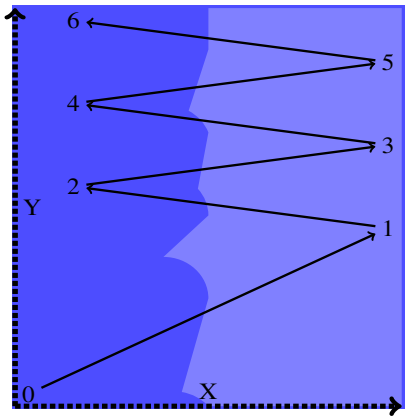


Fig. 2. A simulated environment where two AUVs criss-cross the boundary line of fresh water (light blue) and salt water (dark blue) to gather scientific data about the boundary. At each location the AUVs connect to each other and synchronize their movements and then they yoyo to the next location. See Figure 3 for the yoyo maneuver between two points.

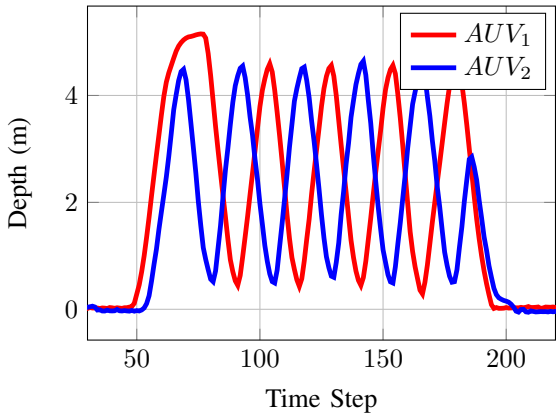


Fig. 3. Real depth data collected by two AUVs. They sync their movements before time step 50 then execute a yoyo maneuver where one AUV is at the top of the water column while the other is at the bottom of the water column. After a set number of yoyos, they ascend at time step 200. They must connect again after step 200 before moving to the next location.

fail to arrive at their destination when traveling underwater, ultimately causing a team plan failure. Further discussion of the AUV domain is provided in Section III.

In order for the individual robots to replan locally without the centralized controller, i.e., intra-robot replanning, the team plans must contain the *rationales* for why the actions in the team plan were chosen. Otherwise, the individual robots may fail to comply with the reasoning of the centralized controller. There has been research in single robot domains that validate the rationales of a plan, monitor those rationales, and update a plan when those rationales change [8], [9], [10]. However, this work was limited to single-robot domains, and as such it assumed a single robot was generating the plan, thereby having complete knowledge of its own reasoning. Additionally, this work did not focus on the replanning aspect of the problem. There is some limited research in distributed multi-robot domains where each robot creates a rationale graph of relevance to other robots, shares it with the robots, and updates the robots when a rationale is violated [11]. However, the work did not explore how to replan when a rationale changes, nor did it fully describe the rationales or the rationale graph used by the robots. Our AUVs have the capability to replan locally, and as such, there is an opportunity to improve the performance of the AUVs executing their team plans in complex dynamic environments like the ocean.

## II. Problem

In order for autonomous intra-robot replanning to be successful within a team, the individual robots must understand the rationales of the team planner for the actions being executed. Further, once the rationales are known, the individual robots must have a way of selecting the policies that may enable the failing rationales of the team plan, such policies are called *replan policies*. Finally, the individual robots must have a method of choosing a single replan policy from those applicable ones with two considerations: most importantly, enable the failing rationale given the state of the environment, and, secondly, do so in a cost-effective manner.

## III. Domain

To illustrate our problem, we are using a simulated AUV domain where two AUVs follow a defined path together (simultaneously), criss-crossing the boundary line between fresh and salt water in order to collect multiple data points in the water column for oceanography research. The simulated path is shown in Figure 2. The AUVs will complete a yoyo maneuver by ascending and descending multiple times between two points on the ocean surface. In order to collect more sensor data in the water column, the AUVs time their movements to have inverse motion in the z-axis (depth). Shown in Figure 3 is depth data collected by two real AUVs completing the described yoyo maneuver between two points on the ocean surface.

Our concerns with this domain are:

1) The AUVs must connect at each location in order to synchronize their movements so that they can perform the behavior shown in Figure 3;
2) Unknown water currents will often push the AUVs off their intended course while they perform the yoyo maneuver;
3) Communication for replanning between the AUVs and the centralized controller by satellite communication is very costly.

To handle these concerns, we implement intra-robot replanning with our new rationale-driven team plan representation to handle failures locally, if possible.

## IV. Rationale-Driven Team Plan Representation

The definition of a rationale needs to be clearly defined before we can proceed to our representation. First, a rationale is a constraint that reduces the planning space of a planning problem. Second, we assume a rationale produces a true or false result. As an example, the constraint $AT\{X = 1, Y =$

$2\}$ reduces the location space to exactly one position and gives the rationale for why the action $GOTO\{X = 1, Y = 2\}$ may have been selected. Thus, we define a rationale as a function with a finite set of parameters:

$$f(p_1, p_2, ..., p_n) = true \vee false \quad (1)$$

that returns true or false given those specific parameters. Our example $AT\{X = 1, Y = 2\}$ constraint can be defined as the following rationale:

$$AT(x, y, 1, 2) = x \equiv 1 \wedge y \equiv 2 \quad (2)$$

where $x$ and $y$ are the location variables of the AUV. This rationale is only true if the AUV is at the specific location $(1, 2)$. Our rationale-driven team plan representation adds these rationales to the team plan alongside the actions they constrain.

Therefore, in the STP hierarchy, we redefine a Play as the following tuple $\langle \mathbb{A}, \mathbb{R}, \mathbb{T} \rangle$:

- $\mathbb{A}$ are the applicability conditions of the Play, i.e., when the team plan can be used;
- $\mathbb{R}$ is the set of roles in the team plan;
- $\mathbb{T}$ is the ordered set of tactics, where a tactic $t_j$ = $\langle \mathbb{R}_i, p, \mathcal{T} \rangle$
  - $\mathbb{R}_i$ is the assigned role,
  - $p$ are the parameters for this tactic,
  - **$\mathcal{T}$ is the set of rationales for this tactic**.

Each Tactic has a set of rationales $\mathcal{T}$, previously called team plan conditions in [2], that need to be satisfied if there is a failure by the Tactic.

## V. Intra-Robot Replanning Algorithm

We define our intra-robot replanning problem as a tuple $\langle \mathcal{R}, \mathcal{S}, \mathcal{P}, \mathcal{T}, \mathcal{F} \rangle$:

- $\mathcal{R}$ is the set of robots on the team;
- $\mathcal{S}$ is the current state;
- $\mathcal{P}$ is the set of replan policies (i.e., replan tactics); where a policy $\mathcal{P}_j$ = $\langle \alpha, \beta \rangle$:
  - $\alpha$ are the input parameters to the replan policy,
  - $\beta$ are the rationales that can be enabled by this policy (effects);
- $\mathcal{T}$ is a set of rationales for the current tactic;
- $\mathcal{F} \subseteq \mathcal{T}$ are the rationales that are invalid, i.e., failing.

We assume there exists a replan policy that calls the centralized team planner and can therefore enable all possible sets of rationales including the empty set of rationales. We also assume that the replan policies do not invalidate rationales, just enable them. There can be multiple policies that enable the same rationale so the problem becomes selecting the applicable policies and then choosing the best option given the current state of the environment.

In Algorithm 1, the invalid rationales are provided in $\mathcal{F}$. For every invalid rationale, the applicable replan policies $\mathbb{P}$ are selected by checking that the replan policy can enable the invalid rationale as provided in $\beta$ of $\mathcal{P}_j$ in the problem definition. If $\mathcal{P}_j$ has the invalid rationale in $\beta$ then the replan policy is placed into the $\mathbb{P}$ queue. Then the policies are sorted. There are different options for sorting the replan policies, and part of the complexity of intra-robot replanning is choosing a good option based on the environment, see Section VII. While the rationale is invalid, the replan policies of the queue are executed and then popped from the queue. There is always the assumed replan policy that calls the centralized team planner so the loop will not cycle forever.

**Algorithm 1** Intra-Robot Replanning algorithm for selecting applicable replan policies, sorting them, and then executing the policies until the rationales are true.

**Require:** $\mathcal{F} \wedge \mathcal{S} \wedge \mathcal{R} \wedge \mathcal{P}$
  **for all** $\mathcal{F}_i \in \mathcal{F}$ **do**
    **for all** $\mathcal{P}_i \in \mathcal{P}$ **do**  # Find applicable replan policies
      **if** $\mathcal{F}_i \subseteq \beta \in \mathcal{P}_i$ **then**
        $\mathbb{P}$.push($\mathcal{P}_i$)
      **end if**
    **end for**
    $\mathbb{P} \leftarrow$ sort($\mathbb{P}$)
    **repeat**
      $\mathbb{P}.front(\alpha)$  # Execute with $\alpha \subseteq \mathcal{S} \wedge \mathcal{R}$
      $\mathbb{P}$.pop()
    **until** $\mathcal{F}_i(p_1, ..., p_n) = true$
  **end for**

## VI. Experiments

We developed a simulated environment in order to test the rationale-driven team plan representation with the intra-robot replanning algorithm over multiple experiments with different environmental conditions. In this section, we will describe:

A. The constants used in the experiments,
B. The tactics used by the AUVs,
C. The rationale used in the team plan,
D. The replan policies used by the AUVs,
E. The experiments and their results.

### A. Constants

Everything in the simulation is defined in kilometers and seconds. The constants below are assumed:

1) AUV speed is 0.0128 km/s (2.5 knots)
2) Wifi radius is 0.1 km
3) Satellite cost is 600 seconds (10 mins)
4) Wifi cost is 3 seconds
5) Maximum wait time ($\mathbb{W}$) for a wifi connection is 300 seconds (5 mins)

### B. Tactics

The tactics used in our experiments are:

1) Initial: Starting tactic for getting the AUV ready.
2) Connect {ID}: Connect over wifi to the AUV with the defined ID.
3) Goto {X} {Y}: The AUV stays on the surface and heads directly to the (X, Y) location. We assume it can receive a GPS signal so it does not get lost, but it cannot receive a wifi signal unless it stops.

4) YOYO {X} {Y}: The AUV performs a series of descents and ascents in the water column towards the (X, Y) location. As the AUV is underwater, it cannot receive a GPS or wifi signal so the YOYO maneuver oftentimes does not arrive at the destination location.
5) SATELLITE: The AUV connects to the centralized controller through a satellite link.

### C. Rationale

We focus on a rationale that can be enabled by different replan policies. The AUV's state, $\mathcal{S}$, knows if it is connected over wifi, $\mathbb{C}$, and to what ID it is connected, $\mathbb{C}_{ID}$. The CANCONNECT rationale checks if the AUV has a connection with the $ID$.

$$\begin{aligned} &\textsc{CanConnect}(ID) = \\ &(\mathbb{C} = true) \wedge (\mathbb{C}_{ID} = ID) \end{aligned} \tag{3}$$

We assume the internal state, $\mathcal{S}$, is provided to the rationale, and as such we are only explicitly expressing the inputs that are provided by the team planner.

### D. Replan Policies

For the first three replan policies, an attempted CONNECT lasts 3 seconds and it uses the ID provided by the rationale. The final CONNECT continues until the maximum wait time $\mathbb{W}$, after which the replan policy fails.

1) GOTO$_r$ {$\beta$ = CANCONNECT()}: Uses GOTO to the failed location and then CONNECT.
2) CONNECT-GOTO$_r$ {$\beta$ = CANCONNECT()}: Attempts CONNECT, and then proceeds like GOTO$_r$.
3) STAR-GOTO$_r$ {$\beta$ = CANCONNECT()}: Attempts CONNECT. Then centered at its current location, it creates five evenly spaced points on the circumference of a circle with a radius equal to the wifi distance. It will GOTO each location and attempt to CONNECT. The first location is chosen at random and then it always proceeds to the further point away from its current location until all points have been attempted. If unsuccessful, it proceeds like GOTO$_r$. This was inspired by the sector search pattern described in [12] for search and rescue, but was changed to ensure that the AUVs always traveled the same distance before attempting to connect.
4) GOTO-SATELLITE$_r$ {$\beta$ = any rationale}: Uses GOTO to the failed location and then SATELLITE.
5) SATELLITE$_r$ {$\beta$ = any rationale}: SATELLITE to receive a new plan.

### E. Experiments

For our experiments, we are following our example domain with two AUVs criss-crossing the ocean using the team plan shown in Table I and assuming the AUVs start at location (0, 0). A visualization of the plan's path is in Figure 2. The team plan serves as the base case without any rationales, i.e., how replanning would be handled prior to adding rationales to the plan. In this case, the AUVs would call the centralized team planner through a satellite connect when failures occur.

TABLE I
TEAM PLAN PROVIDED TO AUVS WITHOUT ANY RATIONALES

| **AUV1** | **AUV2** |
|---|---|
| INITIAL | INITIAL |
| CONNECT AUV2 | CONNECT AUV1 |
| YOYO 8 4 | YOYO 8 4 |
| CONNECT AUV2 | CONNECT AUV1 |
| YOYO 1 5 | YOYO 1 5 |
| CONNECT AUV2 | CONNECT AUV1 |
| YOYO 8 6 | YOYO 8 6 |
| CONNECT AUV2 | CONNECT AUV1 |
| YOYO 1 7 | YOYO 1 7 |
| CONNECT AUV2 | CONNECT AUV1 |
| YOYO 8 8 | YOYO 8 8 |
| CONNECT AUV2 | CONNECT AUV1 |
| YOYO 1 9 | YOYO 1 9 |
| END | END |

We assume that there exists a current that pushes the AUVs off course when they are performing the YOYO maneuver. The simulation uses a simplified model of currents that are only applied to the AUV's location after the simulation has determined the YOYO maneuver finished. The AUVs pop up near the intended destination based on random distributions, by which the ocean currents are being modeled. There has been work on modeling ocean and wave currents for real-time simulations, but this level of simulation goes beyond the scope and need of this work [13].

In this domain, each AUV will complete a set number of yoyos and then surface to connect with the other AUV before beginning the next traversal. The only purpose of the specific locations is to direct the AUVs in a certain direction and so the only rationale added to the team plan is that they need to be able to connect when they surface. The defined location does serve as a location known by both AUVs in case they cannot connect where they surface, which is how we designed the replan policies to use it.

We therefore add CANCONNECT to every YOYO tactic in the team plan shown in Table I. See below for a general example:

YOYO {X} {Y}; CANCONNECT({ID});

*1) Experiment 1:* We assume there is an ocean current in the positive y-axis modeled by a normal distribution with mean 2 and variance 0.1, and a uniform distribution from 0 to 0.1 in the x-axis. See Figure 2 for axes.

Figure 4 shows the time taken by the AUVs to complete the team plan using a specific replan policy for every CANCONNECT rationale. The first three replan policies have very consistent performances because they always go towards the failed location and are affected only slightly by the ocean current. Relying on satellite communications for every failure is clearly very costly. The intra-robot replanning policies that attempt to connect over wifi perform better. However, the AUVs have a limited wifi radius of 0.1 km and so they cannot always connect when they surface. STAR-GOTO$_r$ attempts to find the other AUV but the search patterns can add more time, which is why its performance has a higher variance. On the other hand, CONNECT-GOTO$_r$ only adds wifi costs

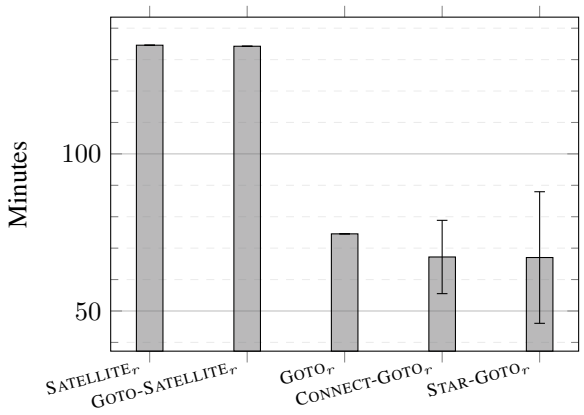


Fig. 4. *Experiment 1, ocean current in both x- and y-axes directions*: Average time and variance to complete the team plan using the stated replan policy.

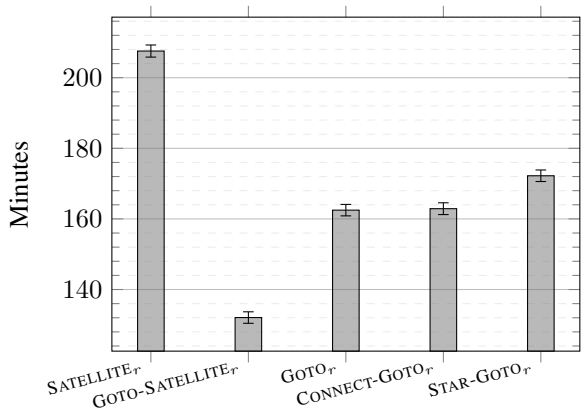


Fig. 6. *Experiment 3, large ocean currents with no wifi connections*: Average time and variance to complete the team plan using the stated replan policy.

to its time in comparison to GOTO$_r$.

*2) Experiment 2:* We assume there is an ocean current in the positive x-axis modeled by a normal distribution with mean 1 and variance 0.05, and 0 in the y-axis. Figure 5 shows that CONNECT-GOTO$_r$ and STAR-GOTO$_r$ perform reliably better than GOTO$_r$ because the two AUVs are often within the wifi radius after surfacing from a yoyo. Similar to *Experiment 1*, STAR-GOTO$_r$ has a higher variance because of the extra search time.

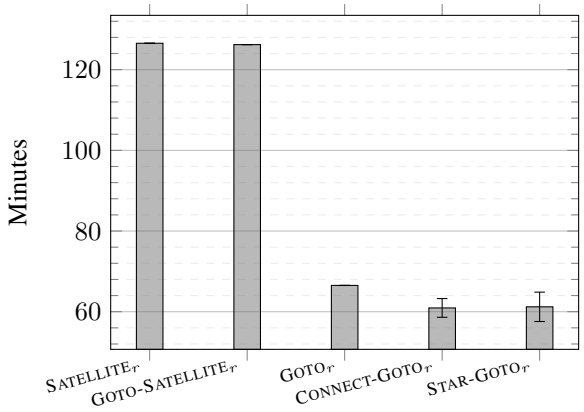


Fig. 5. *Experiment 2, ocean current in only the y-axis direction*: Average time and variance to complete the team plan using the stated replan policy.

*3) Experiment 3:* We assume there is an ocean current in the positive x-axis and y-axis both modeled by a normal distribution with mean 1 and variance 0.2. We are assuming the surface waves are too large and connection over wifi **cannot** be established. The replan policies that try to connect will always fail at the maximum wait time on their final CONNECTand then SATELLITE$_r$ will be executed. Therefore, their average times are consistent, and the time variance is mainly due to the the increase in the variance of the ocean current. Shown in Figure 6, GOTO-SATELLITE$_r$ is the best option because it first moves to the failed location and then immediately calls the satellite. Because both AUVs are together, the centralized controller connects them and they start their next yoyo. SATELLITE$_r$ however connects at the failed location, receives a plan to the failed location (to bring the AUVs together), and then cannot connect at the failed location so it must call the satellite again, being by far the worst option.

*4) Review:* These experiments show that a predetermined sorting of the intra-robot replanning polices in one environment will not always be the best option in another environment. Another challenge is handling unknown variables when deciding the order of the replan polices. In the next section, we describe a method to learn the predicted cost of each replan policy in order to select the lowest costing one.

## VII. Learning Predicted Costs of Replan Policies

In this section, we detail our learning approach to sorting the replan policies in our AUV domain. We use a state based approach that learns a predicted cost for each replan policy given the current state of the environment. We then demonstrate that learning the predicted costs can improve team performance compared to the deterministic approach in Section VI.

### A. Learning Method

We train the neural networks to learn a function from a state of the environment to a predicted cost. The predicted cost is the time it takes for the replan policy to complete (successfully or unsuccessfully) plus the time it takes to complete the next action in the team plan. Our predicted cost is therefore a two step lookahead, i.e., what it will cost to use this replan policy and what will the next action cost. For the AUV domain, we will use the following:

*1) Input State:*

- AUV's current location (X,Y)
- AUV's destination location ($\mathbb{X}$, $\mathbb{Y}$)

*2) Output:*

- Predicted cost of replan policy and the next action

The data to train the neural networks is collected over multiple runs of the simulation. For each replan policy, we

order it first and run the entire team plan. We save the current state when replanning starts, along with the cost it takes to replan plus the cost of the next action. Each AUV collects its own state and cost during the execution of the team plan. A neural network is then trained for each AUV for that particular replan policy using a supervised learning method. We repeat until we have trained a neural network for every replan policy.

For the learning experiment, we used the open source library OpenANN [14]. For each replan policy, we used a full-connected network with one hidden layer with 50 neurons. We used RECTIFIER activators for the hidden layer and a LINEAR activator for the output. For *Experiment 4*, we ran the simulator 5,000 times for each replan policy (ordered first) collecting the state and cost data. We then trained the neural networks using OpenANN's built-in conjugate gradient method. The neural networks were then used to sort the replan policies before execution by LEARNED.

### B. Experiment 4

We combined the ocean currents of the previous experiments in Section VI separated by the locations in Figure 2:

- 1 & 2 are *Experiment 1*
- 3 & 4 are *Experiment 2*
- 5 & 6 are *Experiment 3* (no wifi connection)

Figure 7 shows that the LEARNED method outperforms using the replan policies in a predetermined order for the complex environment. The improvement is made on the last two locations, 5 & 6, where there is no wifi connection. The predicted cost for using GOTO-SATELLITE$_r$ is less than any other replan policy for those two locations. This saves five minutes of wait time at each location, wasted by CONNECT-GOTO$_r$ when it attempts to connect over wifi but fails. And, we clearly see an improvement of roughly ten minutes for the LEARNED method. In this way, LEARNED could effectively use GOTO-SATELLITE$_r$ only when it was the best option given the state of the environment.

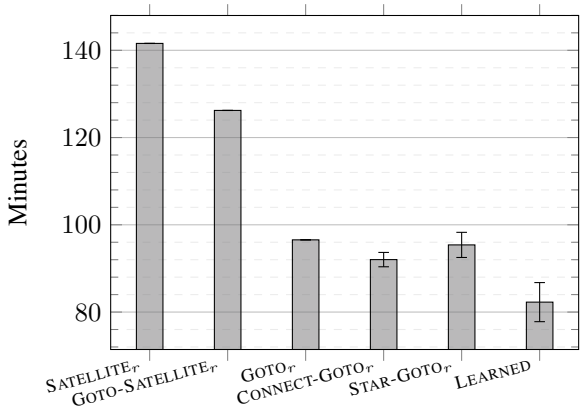


Fig. 7. *Experiment 4, changing ocean currents for different locations*: Average time and variance to complete the team plan using the stated replan policy. LEARNED performs the best in this complex domain by combining the benefits of CONNECT-GOTO$_r$ and GOTO-SATELLITE$_r$.

## VIII. CONCLUSION AND FUTURE WORK

We described a new rationale-driven team plan representation that provides the rationale for why the actions in the team plan were chosen by the team planner. We described our autonomous intra-robot replanning algorithm that determines the set of replan policies that can enable the rationales when there is a failure in the execution of a Tactic. We demonstrate that different replan policies are beneficial in different environments, and as such, learning to predict a cost for each replan policy leads to a better ordering of the replan policies. We then demonstrated that our method of learning the predicted costs for the replan policies did better than the predetermined orders. In future work, we would like to explore learning the cost in more complex environments with multiple different plans. We would also like to introduce more AUVs or other aquatic robots to the simulation for a larger team plan. Future work also includes generating team plans using the rationale-driven team plan representation.


## ACKNOWLEDGEMENTS

The authors would like to thank João Sousa and José Pinto at LSTS for use of their AUVs and meaningful discussions regarding the domain.


## REFERENCES


[1] K. Talamadupula, D. Smith, W. Cushing, and S. Kambhampati, "A theory of intra-agent replanning," *ICAPS 2013 Workshop on Distributed and Multi-Agent Planning (DMAP)*, 2013.

[2] P. Cooksey and M. Veloso, "Intra-robot replanning to enable team plan conditions," in *Proceedings of IROS'17, the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vancouver, Canada, September 2017.

[3] Z. Yan, N. Jouandeau, and A. A. Cherif, "A survey and analysis of multi-robot coordination," *International Journal of Advanced Robotic Systems*, vol. 10, December 2013.

[4] R. Simmons, T. Smith, M. B. Dias, D. Goldberg, D. Hershberger, A. Stentz, and R. Zlot, *A Layered Architecture for Coordination of Mobile Robots*. Dordrecht: Springer Netherlands, 2002, pp. 103–112.

[5] F. Pecora and A. Cesta, "Planning and Scheduling Ingredients for a Multi-Agent System," in *Proceedings of UK PLANSIG02 Workshop, Delft, The Netherlands*, 2002.

[6] B. Browning, J. Bruce, M. Bowling, and M. Veloso, "Stp: Skills, tactics, and plays for multi-robot control in adversarial environments," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 219, no. 1, pp. 33–52, 2005.

[7] S. Ferreira, J. Pereira, P. Dias, J. Sousa, T. A. Johansen, P. Loureno, and V. E. Hovstein, "Managing communication challenges in vehicle networks for remote maritime operations," in *OCEANS 2017 - Aberdeen*, June 2017, pp. 1–7.

[8] S. Kambhampati and S. Kedar, "Explanation-based generalization of partially ordered plans," in *Proceedings of AAAI*, 1991.

[9] M. M. Veloso, "Towards mixed-initiative rationale-supported planning," in *Advanced Planning Technology*, A. Tate, Ed. AAAI Press, May 1996, pp. 277–282.

[10] M. M. Veloso, M. E. Pollack, and M. T. Cox, "Rationale-based monitoring for continuous planning in dynamic environments," in *Proceedings of the Fourth International Conference on AI Planning Systems*, Pittsburgh, PA, June 1998, pp. 171–179.

[11] B. Coltin and M. Veloso, "Towards Replanning for Mobile Service Robots with Shared Information," in *Proceedings of the AAMAS'10 Workshop on Autonomous Robots and Multirobot Systems (ARMS)*, St Paul, MN, May 2013.

[12] S. Bernardini, M. Fox, and D. Long, "Combining temporal planning with probabilistic reasoning for autonomous surveillance missions," *Autonomous Robots*, vol. 41, no. 1, pp. 181–203, Jan 2017.

[13] E. Omerdic and D. Toal, "Modelling of waves and ocean currents for real-time simulation of ocean dynamics," in *OCEANS 2007 - Europe*, June 2007, pp. 1–6.

[14] A. Fabisch, "Openann," 2017, https://github.com/OpenANN/OpenANN.