Proceedings of the Ninth Goal Reasoning Workshop Submitted 10/2021; published 11/2021

A Goal Reasoning Model for Autonomous Underwater Vehicles

Mark A. Wilson MARK.WILSON@NRL.NAVY.MIL
David W. Aha DAVID.AHA@NRL.NAVY.MIL
Navy Center for Applied Research in AI, Code 5510
Naval Research Laboratory, 4555 Overlook Ave SW, Washington, DC 20375 USA

Abstract

In previous work we integrated goal reasoning (GR) models onboard autonomous underwater vehicles (AUVs) and demonstrated their ability to interact with AUV control systems and direct AUVs to accomplish different goals based on sensor feedback from the environment. However, those GR models did not employ cost estimates when estimating goal utility. Further, those models use reactive, single-goal selection and did not consider the impact of altering short-term goals on the long-term mission success. In this paper, we address these needs by developing a new GR model for use in embodied agents, extend it to coordinate multi-vehicle teams, and describe experiments that assess the multiagent model efficacy in AUV scenarios. Our preliminary results with four goal sequence optimizers, and their comparison with two baselines, do not show simultaneous advantages in reward attained and risk incurred, although one GR method incurred lower risk than both baselines.

1. Introduction

To be truly autonomous, intelligent agents should respond robustly to unexpected situations, with minimal guidance from human operators. This capability is particularly advantageous for autonomous underwater vehicles (AUVs), which may have limited or no communication with operators for long periods of time.

Effective human operators may respond to unexpected situations by altering an agent's goals, such as when a new goal is afforded by changing circumstances, or when a current goal becomes useless or unsatisfiable. Enabling agents to dynamically (1) deliberate on the utility of their own goals and (2) alter their goals accordingly can increase their autonomy, reliability, and trustworthiness. This can be achieved by using goal reasoning (GR) models for agent control (Aha, 2018). A GR model can improve an agent's ability to control AUVs in long-term autonomous missions.

In previous work (Wilson et al., 2018), we described an integration of a GR model with the MOOS-IvP vehicle autonomy framework (Benjamin et al., 2010) and demonstrated its ability to direct AUVs. Our previous GR model extended an automated planning agent; some other GR models adopt a similar structure (e.g., Klenk et al., 2013), or one of higher-level reasoning cycles supervising automated planning (e.g., Oxenham and Green 2017). Although useful for solving discrete planning problems, our previous planning model was not well-suited to represent complex, dynamic spatiotemporal environments such as those encountered by AUVs. In particular,

representing temporal change of multiple interdependent variables can become increasingly burdensome or even infeasible as the complexity of the represented system increases. This can undermine the agent's ability to reason about complex, dynamic elements of its environment.

For instance, an AUV might be assigned a mission to survey several regions in an area for possible mines. The AUV should be able to decide the order for surveying those regions and react intelligently to an intrusion by another vessel during its mission, taking into account considerations such as:

- Is the AUV in danger of colliding with the other vessel? If so, how should it alter its mission?
- Does the AUV need to refine its estimate of the vessel's position and velocity?
- Should the AUV attempt to gain more information about the other vessel (e.g., by classifying its type)?
- How much additional risk (e.g., of collision) does the AUV incur by maneuvering to gain more information about the vessel?

These factors are time-dependent and involve multiple interdependent spatial variables related to the maneuvering of the AUV and other vessels. Although some automated planning models (e.g., Fox & Long 2006) include the ability to reason about multiple time-dependent continuous processes, they may require integration with specialized solvers to efficiently compute values that express a complicated interdependence, such as the maximal risk incurred by executing a particular maneuver while a surface vessel proceeds from position x, y with speed s and heading h. Our previous model did not consider, or even compute, accurate cost estimates for executing its plans when selecting goals. We also found that it could be burdensome for a designer to express these factors in a planning language.

Moreover, our previous GR model was reactive and myopic in its goal selection strategy; it selected an immediate goal to pursue without assessing the long-term impact on future mission goals. For instance, the AUV described above will need to know, if it decides to refine its knowledge of the other vessel, whether it will later have sufficient power to complete its survey mission.

In this paper, we introduce a GR model that evaluates the utility of long-term sequences of goals based on a spatiotemporal problem model, directly integrating route planning over spatial components of the embodied GR problem and treating goal sequence selection as an instance of the Orienteering Problem (OP) (Golden et al., 1987). We also present an extension of this model to a multiagent problem and extend goal sequence optimizers to incorporate precedence constraints between goals. Finally, we describe an experiment to assess our model's ability to control AUVs. Our results indicate that it is possible for some optimization techniques to make informed decisions that trade preferential cost reduction against reward maximization in a real-time underwater environment.

2. Related Work

Our work pertains to the application of GR in underwater environments, where the agent controls multiple agents (one per AUV) using a method for goal selection that, to our knowledge, has not previously been investigated with GR agents.

Substantial work has been published on **maritime applications** of GR agents. Klenk et al. (2013) describe ARTUE's use of rules for detecting expectation violations during simulated AUV control, which inspired some of our earlier work (e.g., Wilson et al., 2014; 2016). More recently we reported on in-water tests of GR agents installed onboard AUVs (Wilson et al., 2018). However, we have not previously addressed the problem of controlling *multiple* vehicles, nor addressed the in-situ problem of selecting and sequencing a set of goals under time constraints.

Nelson and Etemadyrad (2018) use GR techniques to improve the performance of an active sonar system by dynamically tuning its transmitter and receiver parameters. In simulated studies, they assessed GR's ability to decide which direction to steer the system's beam, and reported improved target location estimation and tracking. In our navigation task, we instead address the *oversubscription problem* (i.e., where resource or time limitations prevent all goals from being achieved) (Smith, 2004) and select a set of goals rather than a single next goal to pursue.

In three more recent examples involving the GR control of simulated AUVs, Bride et al. (2021) describe GRAVITAS, an agent that uses model checking to assess whether jointly selected goals conflict, in which case it generates deconfliction plans. Next, Vilchis-Medina et al. (2021) describe a non-monotonic reasoning application of GR where goals involve conducting transects to film fish. Finally, Kondrakunta et al.'s (2018) agent distinguishes whether an expectation violation (found during an underwater mine-clearing mission) threatens its current goal and, if so, uses explanation patterns to formulate alternative goals. We instead focus on selecting and sequencing a subset of goals (so as to maximize rewards and minimize costs), where not all goals can be achieved within a given time budget, and goals cannot conflict.

The T-REX architecture (Rajan et al., 2012) is a framework for robotic control that has been applied on AUVs to adaptively execute scientific missions. T-REX distributes responsibilities for planning and control among a set of *reactors* that monitor state variables and dispatch updates to modeled variables. T-REX aims to provide concurrent control among many reactors and thus provides a high degree of flexibility coupled with a model for consistent hierarchical variable dispatch, whereas our work employs a monolithic goal reasoner responsible for monitoring execution, selecting mission goals, and dispatching plans. T-REX models variable updates as representing reactor-level goals that propagate from one reactor to another, while we focus on mission-level goals. Finally, T-REX employs a planning model based on satisfying timeline constraints (Smith et al., 2008), whereas our planning model is based on modeling time and distance, and satisfies timeline constraints by default.

Several research efforts have addressed the problem of **multiagent goal reasoning**. Jaidee et al. (2013) describe a case-based GR algorithm for playing a real-time strategy game. Their approach employs a team of cooperating agents, where each learns a control policy for bots of one class (e.g., archers, knights, or peasants). Roberts et al. (2015) instead describe a goal refinement approach for deploying a set of agents in a simulated disaster relief mission. Both efforts use a centralized controller that selects and assigns a sequence of goals during a scenario, without reference to time

or resource constraints. In contrast, we focus on the problem of selecting and sequencing a set of goals under time constraints.

Golpayegani and Clarke (2016) describe a model in which a community of agents self-organize to form dynamic coalitions, within which agents collaborate to achieve a shared goal while also pursuing a set of individual desires. While these collaborations are identified in situ, this contrasts with the dynamic problem we address, which assumes a fixed team structure in which goals are simultaneously selected, sequenced, and assigned to team members to optimize rewards.

Hofmann et al. (2019; 2021) describe their sophisticated approach that won the 2019 RoboCup Logistics League championship, where a set of robots is tasked to assemble products for dynamically generated orders. Their CLIPS Executive performs high-level decision making for this oversubscription task. It implements a variant of the goal lifecycle (Roberts et al., 2015), where agents share partial world models, goal achievement requires resources whose access is constrained (e.g., due to use by another team's robots), and lockout mechanisms are used to prevent conflicts. Their control model is distributed, and goals are structured in goal trees. As in our work, their multiagent task involves dynamic goal selection and leverages a pre-existing plan library. However, their agents select a *single* goal to pursue next, whereas our problem involves selecting and sequencing a set of goals at once.

Rahimi et al. (2021a) describe SMASH, a human value-focused BDI system for controlling devices in a Smart Home Internet of Things (IoT) environment. SMASH executes a GR cycle in which a set of goals is selected for processing when given a set of beliefs, user-provided goals, goal status information, and related information. Goals are selected incrementally when their conditions (i.e., beliefs) are satisfied in activation rules and ordered high in importance by goal impact rules and correlate well with the user's priorities and preferences. Rahimi et al. (2021b) extended SMASH with reinforcement learning techniques to learn user behaviors and values. While their multiagent GR system incrementally selects a set of goals for agents to perform, their selection is not modelled as an OP, the goals are not sequenced, and selection is not made under time constraints.

Substantial research has focused on **goal selection** in GR agents. T-ARTUE (Powell et al., 2011) uses case-based reasoning to select goals and active and interactive learning methods to expand and refine its goal selection knowledge. In contrast, our agents do not expand their goal-selection knowledge while underwater, where they have no contact with their human operator. M-ARTUE (Wilson et al., 2013) instead assesses candidate goals using multiple reward functions to trade off exploration and exploitation in partially-observable domains. Our agent does not leverage any intrinsic motivation to pursue exploration, although desirable exploration can be rewarded through exploration-oriented tasks, such as seafloor surveys. Also, both ARTUE variants select only a single goal in response to discrepancies, rather than a set of goals, and they operate in a discrete rather than a continuous domain.

Smith (2004) uses an OP representation to solve oversubscription problems in a Mars rover domain. We instead introduce time-dependent costs between task nodes, dynamically computed from a routing graph, and extend our work to a team of agents. Also, Smith uses state-dependent projections of the OP graph to reason about state-dependent costs, whereas we instead compute task costs dynamically during planning.

Núñez-Molina et al.'s (2020) DQP architecture uses deep Q-learning to select subgoals predicted to minimize the length of a solution path in a discrete video game scenario and an automated planner to generate a plan to achieve it. Their approach addresses the problem of selecting from large goal spaces, attempts to optimize overall performance through goal ordering, and considers the impact of individual goal selection on overall performance. Our agents instead: do not use reinforcement learning; operate in a continuous domain; and exploit a pre-existing library of tasks that the robot can complete. Bonanno et al. (2016) also use deep learning for goal selection, in this case for a Minecraft scenario. Similar to our approach, it uses a predefined library of tasks in a continuous domain. Cox (2016) instead describes a formal extension of automated planning to GR and characterizes the problem of goal selection as one of formulating or changing goals, rather than of learning a goal selection function. However, all three efforts select a single goal at a time, rather than select and order a complete goal sequence.

BDI agents have been designed to pursue multiple goals simultaneously. For example, Tinnemeier et al. (2007) describe a mechanism that allows BDI agents to process and select among incompatible goals. In contrast, our agents pursue goals only sequentially, and cannot select simultaneous conflicting goals.

Kondrakunta and Cox (2017) describe a goal-selection technique for a simulated construction domain. Similar to ours, their technique involves time-varying values, task-dependent costs and rewards, and time limits on execution. However, our approach implements secondary preferential costs to consider during goal selection, rather than using time as a single cost. Also, their approach is implemented within a cognitive architecture (MIDCA) using automated planning representations, and selects goals based on a utility ratio of reward over cost, rather than explicitly maximizing reward within a budget. Finally, our approach dynamically selects a goal sequence based on costs derived from observed conditions *in situ*.

Siler and Cox (2018) extend the concept of preference-based planning to encompass problems in which the complete set of goals is unachievable, using partial orders over goals to express the qualitative preferences of human operators as to which goals should be included. Similar to our problem, they focus on achieving a preferred sequence of goals within budgets. However, their approach differs by considering qualitative ordering rather than parametric reward and by using automated planning, without preferential routing, in a discrete, atemporal domain.

3. Goal Reasoning for an Embodied Agent

To improve our GR model for use in embodied agents, we seek to incorporate accurate estimates of the utility (i.e., balancing rewards and costs) of pursuing goals and the ability to select long-term sequences of goals. To do so, we directly incorporate a model of spatial navigation and execution, and we optimize goal sequences using utility estimates by attempting to maximize the agent's reward while respecting cost budgets (e.g., total mission time). Our approach to GR for AUV control requires, among others, the following capabilities (which are detailed below):

1. A GR model of spatiotemporal domains for use in embodied agents;
2. A route planner to generate vehicle trajectories and estimate their costs, tightly integrated with GR;
3. A goal selection function to select and order a sequence of potential mission goals to pursue.

3.1 Goal Reasoning Model

3.1.1 State Model

To reason about embodied-agent domains, our state representation must model spatiotemporal features that describe the dynamics of the environment and the entities which serve as loci for many of the agent's goals. Our model supports two classes of features:

- Features that may be represented as points (e.g., another vessel or a mine-like object on the ocean floor);
- Features that may be represented as regions (e.g., an area to search or an obstacle to avoid).

To represent these features, our model includes the following aspects of the environment that may affect goal decisions:

- Points of interest $P = \{p_1, p_2, \ldots, p_l\}$, with $p_i = \langle pos_i, vel_i \rangle$ (i.e., a position and a velocity)
- Regions $R = \{r_1, r_2, \ldots, r_k\}$, with $r_i = \langle \langle x_1^i, y_1^i \rangle, \langle x_2^i, y_2^i \rangle, \ldots, \langle x_h^i, y_h^i \rangle \rangle$ (i.e., a sequence of vertices defining a polygonal area)

Regions are typically predefined (e.g., as areas an operator wants to survey), while points of interest may be added, removed, or updated during execution. For the purpose of our AUV missions, we use a primarily 2D model of the environment (i.e., x-y or latitude-longitude coordinates), but these representations could be extended to higher dimensions for more advanced missions or other domains. The region in which the entire mission will take place is encompassed by an operations area. Figure 1 illustrates our model of spatial features in the environment.

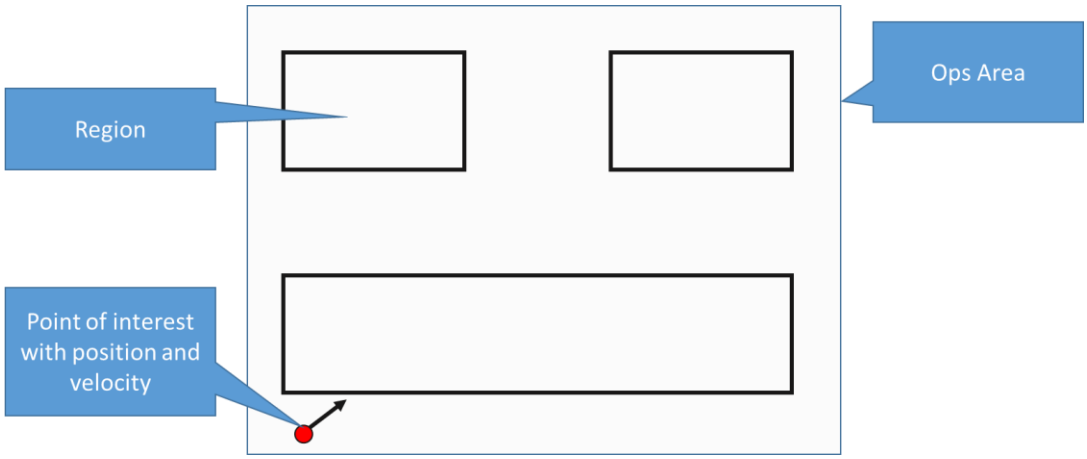

Figure 1: Spatial features of the environment in the embodied goal reasoning model, demonstrating a mission layout used during test missions in Boston Harbor.

Additionally, the state represents the current configuration (i.e., position, heading, and speed) for the vehicle, and beliefs about teammates' configuration in a multi-vehicle problem.

3.1.2 Goals, Tasks, and Behaviors

Automated planning models often represent goals as desired states that an agent can achieve by choosing a correct sequence of discrete actions. In our continuous embodied domain, we abandon

action models and adopt an alternative ontology, similar to that of Balakirsky et al. (2017), in which goals are achieved by completing predefined *tasks*. There is a one-to-one correspondence between goals $\boldsymbol{G}$ and tasks $\boldsymbol{T}$. Tasks are completed by executing a sequence or set of predefined *behaviors*. Each behavior encodes reactive control instructions to drive a robot or vehicle in a continuous domain, and may be parameterized by the particulars of a given task.

For instance, the goal to identify mine-like objects on a seafloor region may be achieved by completing a *survey* task. When it is time to complete the task, the agent should activate a set of behaviors that direct the robot to drive long, straight lines, with a sensor activated, over that particular region.

This ontology maps naturally onto the structure of the MOOS-IvP autonomy framework we use to help us control AUVs; it contains a library of predefined behaviors, implemented in C++, to guide a vehicle to complete tasks. We do not consider discrete actions the agent may take; we model only tasks to be completed. The agent must select which tasks to pursue and in what sequence, and task descriptions map to behaviors that should be activated and the appropriate parameters for them. Typically, tasks and their implementing behaviors are defined with respect to spatial features in the environment, as described above.

Each task T has a reward value $r(T)$ and a cost vector $\vec{c}(T)$, both of which may be parameterized by the particulars of the task (e.g., surveying a larger region garners more reward but takes more time than a smaller region). The primary cost of a task is the time it will take to complete; secondary costs express preferential behavior (e.g., by modeling risk as a cost, we can encourage the agent to pursue low-risk rather than high-risk tasks). Task rewards and costs are used during goal-sequence selection to compute a high-reward sequence that does not violate user-defined cost budgets. To account for the dynamic nature of the environment, tasks' costs are time- and state-dependent. For instance, a survey task's risk may be high if another vessel is projected to enter the region while the survey is underway, but low otherwise.

3.2 Route Planning

Tasks are completed by executing behavior-driven maneuvers with respect to physically situated features of the environment; thus, tasks themselves are spatially situated. Every task T has a start position and an end position in the agent's configuration space. (For example, to survey a region of the seabed, the AUV must be present at that region, and it will finish the task at the end of the final leg of the survey.)

To navigate between tasks, we construct a probabilistic roadmap (PRM) (Kavraki et al., 1996) in the operations area, with the requirement that it intersect every predefined region of interest for the agent. A PRM is a graph of spatial connectivity: the graph's nodes are positions in space, and an edge e connects two positions pos_1 and pos_2 if the distance between them is less than some maximum d_{max} and there is a navigable path the vehicle can follow from pos_1 to pos_2. Nodes are distributed at random in the free configuration space of the vehicle until sufficient connectivity (i.e., a node in every region of interest, with no isolated subgraphs) is achieved. Figure 2 depicts an illustrative PRM in the environment of Figure 1. The PRM is not used for guidance during task execution (which is driven by reactive behaviors), but only for navigation between tasks.

Each edge e in our PRM has a cost vector $\vec{c}(e)$, expressing the same primary and secondary costs as described above for tasks. We estimate the cost of navigating from one task to the next by

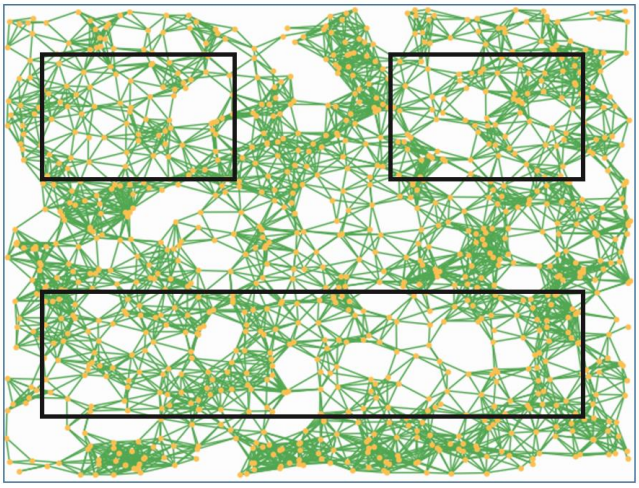

Figure 2: A probabilistic roadmap connecting regions inside the operations area from Figure 1, with randomly distributed nodes. This example contains no fixed obstacles.

selecting the nearest PRM nodes to the start and end positions of the tasks and using an A* search routine to find the shortest distance between the nodes, combining costs according to a preferential weight scheme selected by the human operator.

During periodic updates to the agent's belief state, we recompute time-dependent costs for the PRM's edges. For example, the risk cost of an edge will be high at times when other vessels are near it, and low otherwise. Likewise, the navigation cost of an edge will be higher when ocean currents are counter to the direction of motion than when they are aligned with it. Using this time-dependent PRM, we are able to compute the cost of maneuvering from one task to the next, based on time of navigation (Chabini & Lan, 2002), during goal-sequence selection.

3.3 Goal Sequence Selection

Once the agent has models of the costs and rewards associated with tasks and navigation, we can model the goal-sequence selection problem as a graph, with each node representing a task $T \in \boldsymbol{T}$ and associated reward $r(T)$, and edges between nodes representing the costs $\vec{c}(T_1, T_2, t)$ to navigate to and execute the next task at a given time t. Special nodes are inserted to represent the current state and the end of the mission. Figure 3 illustrates this structure.

The problem of selecting a goal sequence can be expressed, using this graph structure, as an instance of the OP (Golden et al., 1987), an NP-hard problem sharing characteristics with the Traveling Salesman Problem (TSP) and the Knapsack Problem. The OP is the problem of selecting a subset of nodes $\boldsymbol{T}' \subseteq \boldsymbol{T}$ on the graph to visit and a sequence $\pi = \langle T'_1, T'_2, \ldots, T'_n \rangle$ in which to visit them (i.e., a path through the graph to visit all selected nodes), such that the reward is maximized without violating cost budgets $\vec{B}$ (e.g., a time budget and a risk budget). That is, we wish to select $\boldsymbol{T}' \subseteq \boldsymbol{T}$ and π to maximize $\sum_{T' \in \boldsymbol{T}'} r(T')$, subject to $\forall i \sum_{T' \in \boldsymbol{T}'} c_i(T') < B_i$.

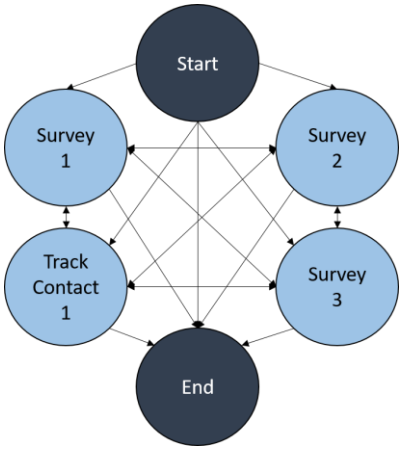

Figure 3: Graph structure representing available tasks as nodes, with costs to navigate and complete the next task associated with edges and rewards associated with completing task nodes.

Additionally, for some tasks, we consider the possibility of partially completing the task. For instance, the vehicle may not have time to survey the entirety of a region, but may have time to survey half of it. We define a discrete number of mutually-exclusive variants of these tasks, representing different levels of completion. For instance, we might consider variants of the survey task representing one-fourth, one-half, three-fourths, or full completion, and only one of these variants may be selected for any given plan.

4. Goal Sequence Selection Optimizers

The cost for any edge in the goal-sequence selection OP will depend on the time taken to reach the edge (i.e., on the path already taken to reach the preceding task). Thus, the particular OP expressing the single-vehicle goal-sequence selection problem is a Time-Dependent OP (TDOP). Solving the TDOP with linear programming optimization tools, in this domain, involves either an impractically large problem matrix or very large time discretization steps (and consequent imprecision). Instead, we designed and implemented several anytime optimizers for the exact goal-sequence selection problem. They dynamically compute edge costs based on the partial solution preceding the edge in question. The performance of time-dependent A* routing allows us to evaluate many possible goal sequences without pre-caching cost values. Sections 4.1-4.4 describe the optimizers we implemented for solving the single-vehicle TDOP before extending to a multi-vehicle problem.

4.1 Branch and Bound

The branch-and-bound algorithm is a tree search that recursively expands nodes (i.e., in our problem, visiting tasks that have not yet been visited on the current branch of the tree) while tracking the minimum known value at a solution for an objective function. At every step the partial solution is examined to see if it violates the known minimum of the objective function; any branch that does violate the known minimum is pruned. This algorithm was a logical option as it is

commonly used for NP-hard optimization problems and was developed for use on the TSP (Little et al., 1963).

Our branch-and-bound optimizer is unique among our optimizers in that it does not maximize reward; because it is impossible to know how much reward a partial solution will eventually accrue, no branch would ever be pruned. Instead, the optimizer minimizes cost, including an additional "missed reward cost" for unachieved goals (although this additional cost is not used in the bounding/pruning operation, to preserve correctness). This approach requires an additional parameter that weights the missed reward cost versus navigation and task costs in the problem.

Branch-and-bound has several shortcomings, including the need to determine a good value for the weight parameter, the lack of directed search, and the unlikelihood of finding certain solutions under time constraints. However, the branch-and-bound optimizer provides a useful baseline for other optimizers.

4.2 Greedy Randomized Adaptive Search Procedure

The Greedy Randomized Adaptive Search Procedure (GRASP) (Feo & Resende, 1995) is a metaheuristic algorithm that proceeds in two phases. In the *construction* phase, a random solution is greedily constructed. In the *refinement* phase, a local search is used to find improvements on the constructed solution. We selected GRASP optimization as a possible approach as it has been used on several variants of the orienteering problem (e.g., Souffriau et al., 2008).

In our GRASP optimizer, the construction phase proceeds by randomly selecting a task that is not yet in the goal sequence, and inserting it at the optimal position in the existing solution, where the "optimal" position is defined as the one that incurs the lowest cost (since the task will garner the agent the same reward at any position). This is repeated until no further tasks can be inserted without violating the budgets.

The local-search phase accepts the constructed solution and a list of tasks that were not inserted during the construction phase. The optimizer attempts to swap each of these remaining tasks with a task in the solution that will maximally increase the overall reward of the solution.

4.3 Genetic Algorithm

Genetic algorithms are a metaheuristic approach to optimization problems that represent solutions as a sequence of codes. Over repeated iterations, groups of solutions are evaluated, downselected, combined, and new solutions randomly generated. Although genetic algorithms provide undirected search, we believe including this approach provides another useful baseline for techniques that are more targeted to the structure of the orienteering problem.

In our genetic algorithm optimizer, individual solutions represent a sequence of some subset of goals, with some of the goals in the solution optionally disabled. At each iteration, the solutions are randomly partitioned and the best individuals from each partition retained. Other solutions are randomly paired and a random subsequence of goals swapped between the two (crossover) and randomly subjected to mutations such as enabling or disabling goals, or switching one goal for another.

Our genetic algorithm optimizer uses 20 candidate solutions per generation, retains the four best solutions to the next generation, and alters the 16 remaining solutions by:

- Swapping a random subsequence of tasks between adjacent solutions;
- With probability $p_0 = 0.05$, probabilistically applying one of three mutations to the solution.

These three mutations are (1) enabling or disabling a task, (2) changing a task in the solution for another task, and (3) changing the completion level of a task in the solution.

4.4 Ant Colony

Ant colony optimization is a bio-inspired approach to an explore-and-exploit search procedure for path-finding in graphs, in which several solutions are constructed per iteration and promising solution paths are given additional weight for future iterations.

Dorigo and Gambardella (1997) applied ant colony optimization to the TSP; we adapt their approach to the TDOP. Although we altered the algorithm to start all ants at the vehicle's current position (versus at a random node) and to permit the optimizer to consider a solution complete after visiting only a subset of tasks, few major alterations were required, as node expansion rules for the TDOP are straightforward (i.e., successor nodes are simply tasks that have not been visited).

We use $n = 10$ ants per group traveling through the problem, and adopt Dorigo and Gambardella's values of $q_0 = 0.9$ (i.e., the probability of selecting the successor with the highest pheromone level, rather than probabilistically selecting a successor by its quality) and $\beta = 2.0$ (the exponent applied to utility for computing successor quality).

5. Distributed Multi-Vehicle Teams

As AUVs proliferate, missions employing teams of vehicles are becoming more common. Teams of vehicles can divide tasks to accomplish missions in less time, and can provide wider sensor coverage and a greater variety of sensing modalities than a single vehicle. However, the ability to respond to dynamic environments during mission execution is still useful.

During underwater operations, communications with AUVs are limited and unreliable, making it difficult to provide centralized mission direction. Thus, we aim to provide a *distributed* autonomy model in which individual AUV agents react to their own sensor inputs and provide state updates to their teammates on an opportunistic basis when communications are available. Vehicles can quiesce to a shared plan as they receive relevant data about the environment from their teammates.

5.1 Orienteering Problem for Teams

For an agent to determine what goals it should pursue, it needs to know what goals its teammates will pursue given the same data. Therefore, each agent needs to construct a complete plan for the entire team on each goal reasoning cycle, using the same GR and planning models as its teammates. Constructing a complete plan for multiple agents to visit nodes with time dependent costs or rewards is frequently studied in the context of tourist scheduling (Gavalas et al., 2015) and includes time windows when nodes are "open" for visiting; this is a variant of the orienteering problem known as the *time-dependent team orienteering problem with time windows* (TDTOPTW). Since we do not consider time windows, we are concerned only with the *time-dependent team orienteering problem* (TDTOP). The OP graph model for the goal sequence selection problem can be extended by introducing start and end nodes for each vehicle, with the start nodes representing each agent's state at the beginning of planning and the restriction that the i^{th} end node can only be visited by the i^{th} vehicle. Figure 4 illustrates the same goal selection problem as Figure 3, but for a 2-vehicle team.

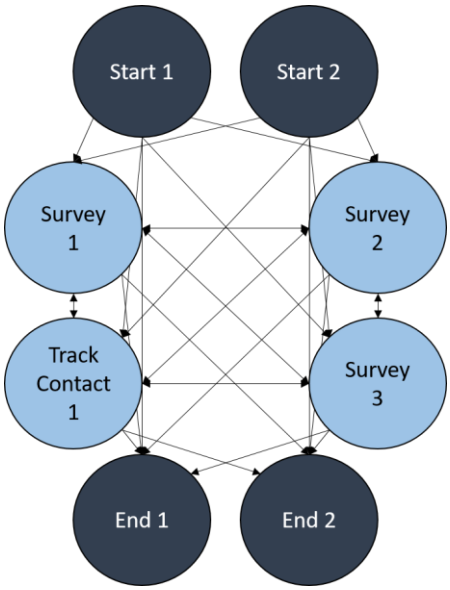

Figure 4: Graph structure representing a team-based goal selection problem

5.2 Optimizers for Teams

There are several possible approaches to extending the optimizers described in Section 4 that use multiple vehicles to solve TDTOP problems. Initially, we have opted for straightforward modifications. For the branch-and-bound optimizer, the search expansion is modified to include expansions for each vehicle on the team. For the GRASP optimizer, during the construction phase, each task is assigned to a successive vehicle (i.e., the first random task is assigned to the first vehicle, the second random task to the second vehicle, and so on). For the genetic algorithm optimizer, each genome in the sequence encodes not only a task and its activation state, but also the vehicle to which it is assigned, and a fourth mutation alters the vehicle to which a task in a candidate solution is assigned. Finally, for the ant colony optimizer, we modify state expansion to generate child nodes considering both the next task and which vehicle will be assigned to it.

6. Precedence Constraints

Although goals can frequently be accomplished in any order, it is sometimes the case that one goal must be achieved before another. For example, in the AUV domain, a vehicle must scan the seabed for objects of interest before attempting to acquire additional data and classify any detected objects. To express these requirements, we can impose a partial ordering on the set of goals. This provides an additional constraint for the goal sequence OP, stating that any dependent goal G_2 can be achieved only after a prerequisite goal G_1. We investigate two techniques for solving goal sequence selection with precedence constraints, as described in Sections 6.1-6.2.

6.1 Topological Sorting

We extended many of our existing solvers to the precedence-constrained TDTOP problem using a technique originally developed for precedence constraints in the multi-TSP (McMahon et al., 2020). This approach uses Kahn's algorithm (Kahn, 1962) for topological sorting to place tasks into layers, with each successive layer containing all tasks that can be accomplished once the previous layer is complete. Each layer is an ordinary (i.e., non-precedence-constrained) multiagent TSP and can be solved as such.

In applying this approach to the TDTOP of goal sequence selection, the major difficulty is that, unlike in the TSP, an OP solution for one layer may leave out prerequisite tasks for future layers. Thus, layers are not truly independent problems, though each layer can be solved as though its tasks contain no precedence constraints. To address this problem, we filter tasks whose prerequisites are not satisfied by previous layers out of each layer's task list before solving. Each layer can then be solved using one of the existing TDTOP optimizers. To obtain high-quality solutions in optimizers with randomized behavior, we re-run the per-layer optimizer several times and the overall optimization over all layers several times, taking the best overall solution.

6.2 Bee Colony Optimization

Yu et al. (2019) addressed the TDTOP with time windows in the context of scheduling tourist visits using a bee colony optimizer combined with simulated annealing. The time-dependent team orienteering problem with time windows is similar to the precedence-constrained TDTOP. Two major differences are that, in their problem, the time dependence arises in the context of rewards rather than costs, and their constraints on when nodes may be visited are in the form of acceptable time windows, rather than ordering requirements. However, the difference in time dependence (rewards or costs) does not affect the operation of the algorithm, and the given approach for visit-time violations in their bee colony optimizer is simply to drop the affected node from the solution, which is easily applied to precedence constraints instead of time-window constraints (i.e., the algorithm simply skips a task in a solution if its prerequisites are not satisfied).

Bee colony optimization proceeds from an initial set of randomized solutions, improving each by local search, and focusing extra local search steps on a solution selected randomly, weighted by solution quality. We adapt the approach of Yu et al. (2019) by altering the constraint checking to accommodate precedence constraints. We also altered their method for neighborhood generation during the local search phase. Their neighborhood generation proceeds by altering the solution in one of four ways:

1. Move a randomly selected task from its position in the solution to immediately after another randomly selected task in the solution;
2. Swap the position of two randomly selected tasks in the solution;
3. Choose a subsequence of tasks in the solution and reverse their order;
4. Change the time window of a selected task.

Since we do not consider time windows, we removed the fourth alteration and replaced it with (1) altering the completion level of task in the solution and (2) assigning a task from the solution to a different vehicle. All parameter values are set as in Yu et al. (2019).

7. Experiments and Results

To assess the efficacy of the goal reasoner in controlling AUVs, we performed simulated testing in randomized trials for a mine-countermeasures mission using MOOS-IvP's uSimMarine and related MOOS simulation tools. The operations area was a 1-kilometer-square zone in Boston Harbor where we have performed in-water trials in the past. Three regions, two small and one large, containing mine-like objects, were designated to be surveyed, and for any mine-like objects to be reacquired and scanned after surveying. A team of two AUVs were assigned to perform these tasks. As additional optional goals, the AUVs were permitted to loiter in one place for a short amount of time (which garners no reward but permits an AUV to "wait out" a high-risk situation), surface and relay a message to topside operators, or refine data on a detected surface vessel.

Two surface vessels were simulated to provide potential collision risks to the AUVs; simulation of these vehicles was designed to mimic behavior of real surface vessels observed in Boston Harbor. Each surface vessel either followed a randomly assigned heading through the operations area (mimicking, e.g., a ferry or shipping traffic) or wandered from random point to random point in the operations area (mimicking, e.g., a pleasure craft or fishing vessel). A global random seed synchronized the AUVs' randomized optimizers and also controlled the behavior of surface vessels and the distribution of mine-like objects; by varying the seed, we obtained different variants of the scenario.

As simplifying assumptions, we permitted the AUVs to directly observe the position, heading, and speed of the surface vessels at all times, and permitted the AUVs to communicate continuously throughout the scenario, providing updates to one another on observed mine-like objects. (Our MOOS-IvP tools include a simulation of sonar-based detection and estimation for surface vehicles, but we wished to test the goal reasoning component without confounding factors.)

The surface vessels provided a unitless "risk" cost to AUVs starting at a distance of 150m, increasing linearly to a maximal value of 100 per second at distance 0. For pathfinding, we weighted time cost at 20% and risk cost at 80% of the overall cost. The roadmap was constructed in batches of 5000 points until all regions were connected, with a maximum connection distance of 50m. The AUVs were given a time budget of 4 hours (14,400 s) and a risk budget of 1000. Optimizers were given 10s per goal reasoning cycle to construct a plan, and goal reasoning cycles were triggered either by discrepancies (a vehicle out of its expected position, or a new mine-like object discovered) or every 3 minutes. Simulations were run at 2x real time, with the optimizers' time limit adjusted to real time (i.e., the optimizers were given 20 simulation seconds to construct plans).

Due to time constraints, we did not examine the performance of the branch-and-bound optimizer, which has been the weakest of our optimizers in single vehicle scenarios. We include, as baselines,

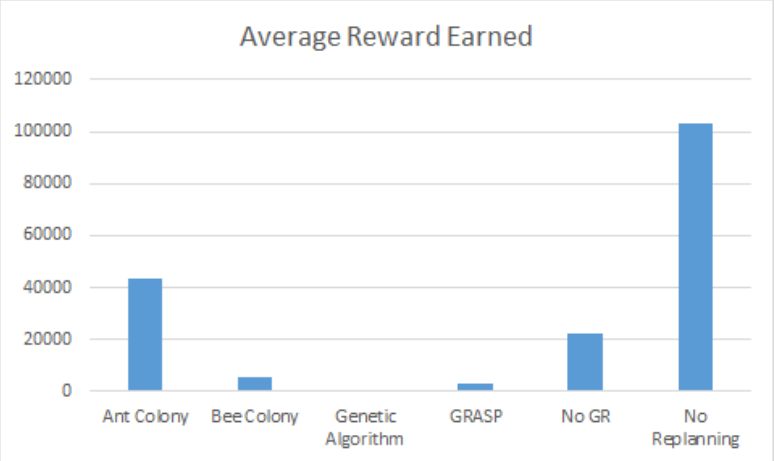

Figure 5: Average reward obtained from summing across five random variations of a simulated mine-countermeasures mission.

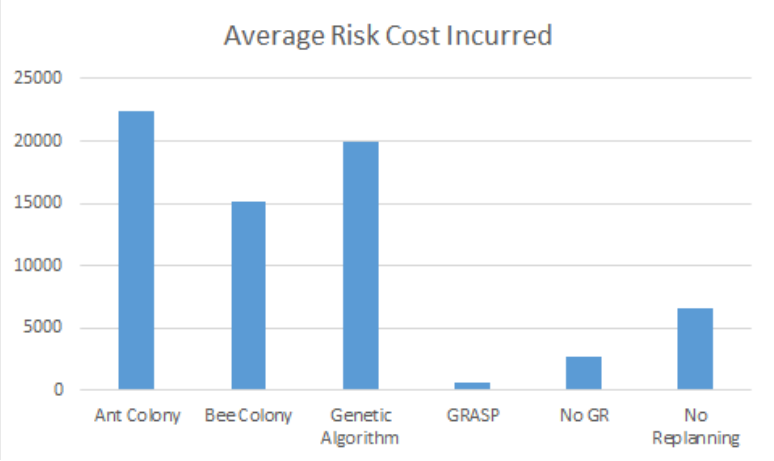

Figure 6: Average risk (i.e., cost) obtained from summing across five random variations of a simulated mine-countermeasures mission.

an agent that creates an initial plan using the GRASP optimizer and never replans, as well as an agent that creates an initial plan using the GRASP optimizer and replans, but cannot alter its goals or their order.

Figure 5 and Figure 6 display the average reward earned (by completing tasks) and risk incurred (by straying too close to surface vessels) for each of the optimizers and the two baseline agents, across five randomized trials in our test mission. All GR agents earn lower reward through task completion than the non-replanning agent, but this is a reasonable outcome, as responding to dynamic threats in the environment to avoid risk can make it difficult to complete tasks. Although three of four GR agents do not perform well in actually avoiding risk in simulation, the GRASP

agent does, on average, respect the risk budget of 1000. We observe that its average reward is also second-lowest among the agents, but believe this is a natural tradeoff for successfully avoiding threats in the environment.

Based on the results of our experiments, we conclude that an appropriate optimization technique (i.e., GRASP) may provide effective balancing of preferential costs against rewards when deciding what tasks to pursue in dynamic environments. An agent without any replanning capabilities gathers the most reward, but at the cost of ignoring changes in the environment completely, thereby incurring risk beyond the given budget. An agent that can replan, but not change its goals or their order, suffers from similar issues but on a lesser scale. Among the GR optimizers we examined, the GRASP optimizer is best able to respect budgets for preferential costs while operating effectively in dynamic environments. We hypothesize that its lower reward values are the result of the environment's dynamics frequently invalidating plans before they can be completed. However, future work is needed to assess this conjecture.

8. Conclusions and Future Work

We introduce a new goal reasoning (GR) model for use in embodied agents, specifically targeting its control of autonomous underwater vehicles (AUVs). Our model selects goals for an agent to pursue only after estimating and accounting for the estimated costs for achieving a goal, in addition to that goal's reward value. Furthermore, we use this model to select and sequence a subset of a given set of goals, where not all goals can be achieved within a limited time budget. We model this problem as a Time-Dependent Team Orienteering Problem (TDTOP), describe four goal sequence selection optimization algorithms and their extension to the multiagent setting, and compare their performance against two baseline systems on simulated mine-countermeasures missions. We found that the simplest baseline algorithm obtained the highest summed reward, but at a higher risk cost than incurred by the most risk-averse optimizer for our GR problem. Future work should include identifying the underlying properties of the optimizers that drive these results, and identifying strategies that may enable greater rewards for the GR agent variants in highly dynamic environments. Additional future work may include investigating alternatives to PRMs for navigation (e.g., some form of regular node distribution or tessellation of the operations area), as well as the ability to dynamically identify and add regions of interest to the agents' model of the environment.

Acknowledgements

The authors gratefully acknowledge the contributions of James McMahon (NRL), Jennifer Nelson (SSEP student employee), Gregory Hyde (NREIP student intern), Ari Goodman (NAVAIR Lakehurst), Jonathan Decker (NRL), Phil Baldoni (NRL), Luke Calkins (NRL), and Artur Wolek (ASEE postdoctoral fellow).

References

Aha, D.W. (2018). Goal reasoning: Foundations, emerging applications, and prospects. *AI Magazine*, *39*(2), 3-24.

Balakirsky, S., Schlenoff, C., Rama Fiorini, S., Redfield, S., Barreto, M., Nakawala, H., ... & Haidegger, T. (2017). Towards a robot task ontology standard. In *Proceeding of the*

International Manufacturing Science and Engineering Conference. Los Angeles, CA: ASME Press.

Benjamin, M., Schmidt, H., Newman, P., & Leonard, J. (2010). Nested autonomy for unmanned marine vehicles with MOOS-IvP. *Journal of Field Robotics*, *27*(6), 834-875.

Bonanno, D., Roberts, M., Smith, L., & Aha, D.W. (2016). Selecting subgoals using deep learning in Minecraft: A preliminary report. In D.W. Aha, A. Wagner, A. Gordon, & Y. Aloimonos (Eds.) *Deep Learning for Artificial Intelligence: Papers from the IJCAI Workshops*. New York, NY: Unpublished.

Bride, H., Dong, J. S., Green, R., Hóu, Z., Mahony, B., & Oxenham, M. (2021). GRAVITAS: A model checking based planning and goal reasoning framework for autonomous systems. *Engineering Applications of Artificial Intelligence*, *97*, 104091.

Chabini, I., & Lan, S. (2002). Adaptations of the A* algorithm for the computation of fastest paths in deterministic discrete-time dynamic networks. *Transactions on Intelligent Transportation Systems*, *3*(1), 60-74.

Cox, M.T. (2016). A model of planning, action, and interpretation with goal reasoning. *Proceedings of the Fourth Annual Conference on Advances in Cognitive Systems* (pp. 48-63). Evanston, IL: Cognitive Systems Foundation.

Dorigo, M., & Gambardella, L.M. (1997). Ant colonies for the travelling salesman problem. *Biosystems*, *43*(2), 73-81.

Feo, T.A., & Resende, Mauricio G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, *6*(2), 109–133.

Fox, M., & Long, D. (2006). Modelling mixed discrete-continuous domains for planning. *Journal of Artificial Intelligence Research*, *27*, 235-297.

Gavalas, D., Konstantopoulos, C., Mastakas, K., Pantziou, G., & Vathis, N. (2015). Heuristics for the time dependent team orienteering problem: Application to tourist route planning. *Computers & Operations Research*, *62*, 36-50.

Golden, B. L., Levy, L., & Vohra, R. (1987). The orienteering problem. *Naval Research Logistics*, *34*(3), 307-318.

Golpayegani, F., & Clarke, S. (2016). Goal-based multi-agent collaboration community formation: a conceptual model. In M. Roberts, D. Borrajo, M. Cox, & N. Yorke-Smith (Eds.) *Goal Reasoning: Papers from the IJCAI Workshop*. [makro.ink/ijcai2016grw].

Hofmann, T., Limpert, N., Mataré, V., Ferrein, A., & Lakemeyer, G. (2019). Winning the RoboCup Logistics League with fast navigation, precise manipulation, and robust goal reasoning. *Proceedings of RoboCup 2019: Robot World Cup XXIII* (pp. 504-516). Sydney, Australia: Springer.

Hofmann, T., Viehmann, T., Gomaa, M., Habering, D., Niemueller, T., & Lakemeyer, G. (2021). Multiagent goal reasoning with the CLIPS Executive in the RoboCup Logistics League. *Proceedings of the Thirteenth International Conference on Agents and Artificial Intelligence* (pp. 80-91). Vienna, Austria: SciTePress.

Jaidee, U., Muñoz-Avila, H., & Aha, D.W. (2013). Case-based goal-driven coordination of multiple learning agents. *Proceedings of the Twenty-First International Conference on Case-Based Reasoning* (pp. 164-178). Saratoga Springs, NY: Springer.

Kahn, A.B. (1962). Topological sorting of large networks. *Communications of the ACM*, 5(11), 558–562.

Kavraki, L.E., Svestka, P., Latombe, J.C., & Overmars, M.H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, *12*(4), 566-580.

Klenk, M., Molineaux, M., & Aha, D.W. (2013). Goal-driven autonomy for responding to unexpected events in strategy simulations. *Computational Intelligence*, *29*(2), 187-206.

Kondrakunta, S., & Cox, M.T. (2017). Autonomous goal selection operations for agent-based architectures. In M. Roberts, M.T. Cox, & H. Muñoz-Avila (Eds.) *Goal Reasoning: Papers from the IJCAI Workshop*. Melbourne, Australia: [http://makro.ink/ijcai2017grw].

Kondrakunta, S., Gogineni, V.R., Molineaux, M., Munoz-Avila, H., Oxenham, M., & Cox, M.T. (2018). Toward problem recognition, explanation and goal formulation. In M. Molineaux, D. Dannenhauer, & M. Roberts (Eds.) *Goal Reasoning: Proceedings of the IJCAI Workshop*. Stockholm, Sweden: [https://dtdannen.github.io/faim2018grw].

Little, J.D.C., Murty, K.G., Sweeney, D.W., & Karel, C. (1963). An algorithm for the traveling salesman problem. *Operations Research*, *11*(6), 972–989.

McMahon, J., Baldoni, P., & Plaku, E. (2020). Precedence constrained multiple traveling salesman problem for dynamic multi-robot systems. Unpublished manuscript.

Nelson, J. K., & Etemadyrad, N. (2018). Prioritizing goals in cognitive sonar: Tracking multiple targets. *Proceedings of the Twenty-First International Conference on Information Fusion* (pp. 1-6). Cambridge, UK: IEEE Press.

Núñez-Molina, C., Nikolov, V., Vellido, I., & Fernández-Olivares, J. (2020). Goal reasoning by selecting subgoals with deep Q-learning. arXiv preprint arXiv:2012.12335.

Oxenham, M., & Green, R. (2017). From direct tasking to goal-driven autonomy for autonomous underwater vehicles. In M. Roberts, M.T. Cox, & H. Muñoz-Avila (Eds.) *Goal Reasoning: Papers from the IJCAI Workshop*. Melbourne, Australia: [http://makro.ink/ijcai2017grw].

Powell, J., Molineaux, M., & Aha, D.W. (2011). Active and interactive discovery of goal selection knowledge. In *Proceedings of the Twenty-Fourth Conference of the Florida AI Research Society*. West Palm Beach, FL: AAAI Press.

Rahimi, H., Trentin, I. F., Ramparany, F., & Boissier, O. (2021a). SMASH: A semantic-enabled multiagent approach for self-adaptation of human-centered IoT. arXiv preprint arXiv:2105.14915.

Rahimi, H., Trentin, I. F., Ramparany, F., & Boissier, O. (2021b). Q-SMASH: Q-learning-based self-adaptation of human-centered Internet of things. arXiv preprint arXiv:2107.05949.

Rajan, K., & Py, F. (2012). T-REX: Partitioned inference for AUV mission control. In R. Sutton & G. Roberts (Eds.) *Further Advances in Unmanned Marine Vehicles*. London, UK: The Institution of Engineering and Technology.

Roberts, M., Apker, T., Johnson, B., Auslander, B., Wellman, B., & Aha, D. W. (2015). Coordinating robot teams for disaster relief. In *Proceedings of the Twenty-Eighth International Flairs Conference*. Hollywood, FL: AAAI Press.

Siler, C., & Cox, M.T. (2018). Generating plans for qualitative goal preferences. In M. Molineaux, D. Dannenhauer, & M. Roberts (Eds.) *Goal Reasoning: Papers from the IJCAI Workshop*. Stockholm, Sweden: [https://dtdannen.github.io/faim2018grw].

Smith, D.E. (2004). Choosing objectives in over-subscription planning, *Proceedings of Fourteenth International Conference on Automated Planning and Scheduling* (pp. 393–401). Whistler, British Columbia, Canada: AAAI Press.

Smith, D.E., Frank, J., & Cushing, W. (2008). The ANML language. In R. Barták & L. McCluskey (Eds.) *Knowledge Engineering for Planning and Scheduling: Papers from the ICAPS Workshop*. Sydney, Australia: [http://ktiml.mff.cuni.cz/~bartak/KEPS2008/wsProgram.html].

Souffriau, W., Vansteenwegen, P., Berghe, G.V., & Oudheusden, D.V. (2008). A greedy randomised adaptive search procedure for the team orienteering problem. *Proceedings of the Ninth EU/MEeting on Metaheuristics for Logistics and Vehicle Routing* (p23-24). Troyes, France: Unpublished.

Tinnemeier, N.A.M., Dastani, M., & Meyer, J.-J. Ch. (2007). Goal selection strategies for rational agents. *Proceedings of the First International Workshop on Languages, Methodologies and Development Tools for Multiagent Systems* (pp. 54–70). Durham, UK: Springer.

Vilchis-Medina, J.L., Godary-Déjean, K., & Lesire, C. (2021). Autonomous decision-making with incomplete information and safety rules based on non-monotonic reasoning. *Robotics and Automation Letters, 6*(4), 8357-8362.

Wilson, M.A., McMahon, J., & Aha, D.W. (2014). Bounded expectations for discrepancy detection in goal-driven autonomy. In A. Saffiotti, N. Hawes, G. Konidaris, & M. Tenorth (Eds.) *AI and Robotics: Papers from the AAAI Workshop* (Technical Report WS-01-14). Quebec City, Quebec, Canada: AAAI Press.

Wilson, M.A., McMahon, J., Wolek, A., Aha, D.W., & Houston, B.H. (2016). Toward goal reasoning for autonomous underwater vehicles: Responding to unexpected agents. In M. Roberts, D. Borrajo, M. Cox, & N. Yorke-Smith (Eds.) *Goal Reasoning: Papers from the IJCAI Workshop*. [makro.ink/ijcai2016grw].

Wilson, M. A., McMahon, J., Wolek, A., Aha, D.W., & Houston, B.H. (2018). Goal reasoning for autonomous underwater vehicles: Responding to unexpected agents. *AI Communications, 31*(2), 151-166.

Wilson, M., Molineaux, M., & Aha, D.W. (2013). Domain-independent heuristics for goal formulation. In *Proceedings of the Twenty-Sixth Florida Artificial Intelligence Research Society Conference*. St. Pete Beach, FL: AAAI Press.

Yu, V., Jewpanya, P., Lin, S.-W., Perwira Redi, A.A.N. (2019). Team orienteering problem with time windows and time-dependent scores. *Computers & Industrial Engineering*, *127*, 213-224.