Knowledge acquisition plans: Generation, combination, and execution

Dylan A. Shell   Jason M. O'Kane

***Abstract*— This paper contemplates the possibility of asking robots questions and having them use their ability to go out into the environment and probe it, in combination with what they already know of the world, to provide answers. We describe a method whereby a robot system efficiently answers such questions on the basis of reasoning about observations as they are made, interrelationships between multiple pieces of evidence, and what they imply.**

A central idea in the approach is to maintain a separation of concerns so that managing 'what is known' is decoupled from 'how it is learned'. This idea is realized in a graph-based representation well-suited to algorithmic manipulation and composition, exposing synergies rife for optimization. We show how to use this representation to leverage both informational overlap between multiple simultaneous queries and availability of multiple robots working in concert to answer those queries. We demonstrate these ideas in a simple case study and present data illustrating how plan quality (in terms of cost to execute) can be improved through an optimization operation that is robot agnostic.

I. Introduction

Some of the earliest formulations for robot planning were based on theorem proving approaches. One extremely well-known example is the STRIPS system, which 'employs a resolution theorem prover to answer questions of particular models' [5]. The present paper explores the implications of a perhaps innocent-looking change in this formulation: What if, rather than answering questions about *models* of the world, we form plans whose purpose is to answer questions about *the world itself*?

Example 1 (Minding the reef): Consider a setting in which an autonomous underwater robot is deployed near a vulnerable coral reef. Development of such robots has been a subject of active research for some time [10], [13], [14], because of the possibility of capturing valuable environmental data rapidly and nondestructively.

Marine scientists may be interested in a variety of queries about the overall health of the corals within the reef, such as the presence of various types of corals at various sites, or the existence and patterns of various maladies including bleaching or physical breakage. They may wonder, for example, whether certain types or coral are more susceptible to bleaching than others in current conditions, or whether physical breakage is correlated with bleaching among certain types of coral. They may also have various types and number of robots or static sensors at their disposal to resolve such queries. These possibilities are illustrated in Figure 1.

The technical content of this paper centers on a plan representation called a *decision graph* suitable for these kinds of knowledge acquisition tasks. Focusing on plans that answer queries about a structured world, we adopt a perspective that treats many of the usual aspects of planning (for instance, *effects* and *costs*) at a high level of abstraction. The resulting algorithms exploit the fact that the dynamics of what is known (i.e., the evolution of beliefs) differ from the mechanics of robot motions—a fact that should impact how we tackle planning for information-oriented tasks. This distinction manifests itself as a separation between a *plan*, which is concerned with knowledge to acquire from the world and its implications for the queries of interest, from the plan's *executor*, which holds the responsibility for selecting appropriate actions to acquire the knowledge called for by the plan. The interface between these two layers takes the form of a *cost model*, which models the execution costs incurred by the executor in carrying out a given plan.

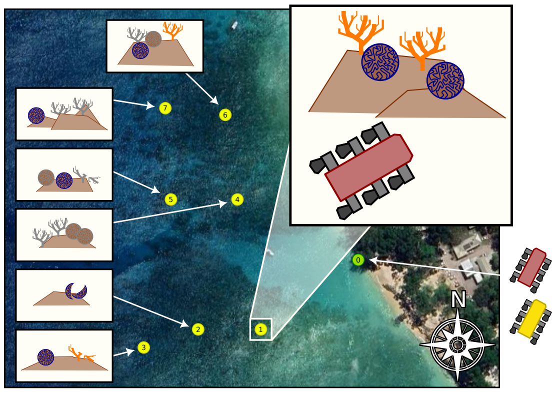

Fig. 1. An underwater Aqua robot travels over the reefs in Holetown Lagoon off the central west coast of Barbados. The autonomous robot aims to resolve questions about the corals present thereupon. Several different locations may be surveyed, each of which may contain some combination fire coral and brain coral. The status of coral at each site may be classified as healthy or unhealthy—if bleached or broken. The sites are the Northernmost sample locations in the study [27], and robots are deployed from the spot at the right (labelled '0'). Among the questions we examine is how performance can be improved through the use of a second robot.

Within that structure, we describe how collections of tasks, in the form of multiple queries, can be synthesized into a single composite task (i.e, a many-to-one *multiplexer*), where overlapping information yields savings in work. In addition, when multiple robots are available to perform work, tasks can be shared (i.e., a one-to-many *demultiplexer*), obtaining efficiencies via parallelism. Composing these pieces yields a many-to-many form where both benefits can accrue.

In exploring this family of questions, this paper builds upon the authors' prior work [25], which introduced a query language for observation-grounded queries and a family of planning algorithms that produce plans that resolve single instances of such queries with a single robot. We now push beyond the limitations of that prior work, significantly generalizing the structure of these knowledge-gathering plans.

After a concise review of related work (Section II) and a precise formulation the problem of planning for knowledge acquisition (Section III), we introduce executors and cost models (Section IV) and discuss several ways that knowledge acquisition plans can be created and composed with one another (Section V). These concepts are exercised in a detailed case study (Section VI). Discussion and concluding marks (Section VII) appear in the usual place.

II. Related work

In the broadest sense, planning problems featuring uncertainty about important aspects of the world have posed a major long-term challenge to the field [8], [12], [16]. Closely related concepts appear in the contexts of active sensing [17], [19], [21] and sensor selection [7], [18], [22], [24], [26], [30]. More narrowly, there exists a rich and growing literature on *informative path planning* [2], [23], [32], in which actions are selected based on their potential to enable the robot to gather information, often expressed via entropy reduction or information gain [6]. The present work differs in that, in planning robots' actions, we lay stress on knowledge—as distinguished from data *per se*. That is, our algorithms direct the robot toward collection of data that has value for resolving the queries in light of both the previous observations and the known structure of the world.

We represent the world through possible world semantics, so answering a query involves determining whether all models have identical query values. Hence, the work has a close relation to the literature on model checking [1].

Our work also dovetails with other approaches that use graph-based representations of robots' plans, including behavior trees [15] and procrustean graphs [11], [20]. Along that axis, the distinctive feature of our work is its laser focus on observation, to the extent the details about actions are fully abstracted out of the plans to the level of executors.

A final closely-related line of work addresses *embodied question answering* [3], [4], [9], [28], [31]. That work, which tends to focus on overcoming the difficulties of robot perception, is complementary to our own, which emphasizes the value that can be extracted from prior work knowledge.

III. Formulation

This section formalizes the knowledge acquisition planning problem addressed in later sections. It expands and generalizes the original formulation for observation-grounded queries in the authors' prior work [25].

A. Properties, situations, and the world model

A robot system —which may be comprised of a single robot, a cooperating team, a collection of motionless sensors, or some other sort of sensorimotor assemblage— operates in some environment. Important facts about that environment are expressed as a set P of boolean *properties* $P = \{p_1, \ldots, p_{|P|}\}$. In any particular execution, some subset of the properties in P will hold. A *situation* $s : P \to \{\text{YES}, \text{NO}\}$ indicates precisely which properties hold. The *situation space*, which contains all possible situations, is denoted $\mathcal{S}$. We assume that s is static and initially unknown.

Example 2 (properties): Recall the setting of Example 1. Let L_{barb} denote the set of 8 sites in this scenario. We might model this scenario with distinct properties at each of the sites of interest indicating, for example, whether fire coral is present at that site at all, whether bleached fire coral is present at that site, and whether broken fire coral is present at that site, along with three parallel properties for brain coral. All told, such a model would have 80 properties spread across the 8 sites in L_{barb}.

Though the situation is unknown, the system will generally have access to a general *world model* $\mathcal{W} \subseteq \mathcal{S}$ indicating classes of situations that can actually occur.

Example 3 (world model): Continuing the example, we might know that certain types of coral cannot be present in certain locations owing to persistent temperature or salinity differences. Moreover, the properties referring to bleached or broken corals can hold only when either fire coral or brain coral are present. The world model $\mathcal{W}$ should contain precisely the situations that obey these kinds of constraints.

B. Query sets

The system's objective is to resolve each of a collection of questions posed in a finite, non-empty *query set* $\mathfrak{Q}$, which is given as input. Each query $\mathcal{Q} \in \mathfrak{Q}$ is a boolean interrogative about the prevailing situation, expressed as a subset of the situation space, $\mathcal{Q} \subset \mathcal{S}$.

Example 4 (queries): A variety of queries might be of interest on our reef. For example, we might have a hypothesis regarding species specific bleaching in the reef, and ask if there are any places with fire coral that is bleached while other coral is not:

$$\mathcal{Q}_{\mathcal{A}} := \left\{ s \in \mathcal{S} \;\middle|\; \begin{array}{c} \exists l \; s(p_{l,\text{fire\_bleached}}) = \text{YES} \\ \wedge s(p_{l,\text{has\_brain\_coral}}) = \text{YES} \\ \wedge s(p_{l,\text{brain\_bleached}}) = \text{NO} \end{array} \right\}. \tag{A}$$

Or, whether all the bleached coral is broken:

$$\mathcal{Q}_{\mathcal{B}} := \left\{ s \in \mathcal{S} \;\middle|\; \begin{array}{r} \forall l \; \big(s(p_{l,\text{fire\_bleached}}) = \text{YES} \implies \\ s(p_{l,\text{fire\_broken}}) = \text{YES}\big) \wedge \\ \big(s(p_{l,\text{brain\_bleached}}) = \text{YES} \implies \\ s(p_{l,\text{brain\_broken}}) = \text{YES}\big) \end{array} \right\}. \tag{B}$$

Queries (A) & (B) will reappear in subsequent discussion. Of course, other more complex queries may be posed too.

C. Plans as decision graphs

To resolve a query set, our system must take action to collect information about the prevailing situation. We express plans to accomplish this goal in a branching graph form.

Definition 1: A decision graph plan *is a directed acyclic graph (V, E) and an associated positive integer m, in which:*

1) *The vertex set is comprised of* decision vertices V_d *and* leaf vertices V_ℓ, *with $V = V_\text{d} \sqcup V_\ell$.*
2) *Each decision vertex $v \in V_\text{d}$ is labeled with a set $P_v \subseteq P$ of properties. Moreover, E contains $2^{|P_v|}$ out-edges from v, each labeled with one of the $2^{|P_v|}$ distinct subsets of P_v.*

3) *Each leaf vertex $v \in V_\ell$ is labeled with an* outcome $o(v) \in \{\text{YES}, \text{NO}\}^m$.
4) *A single vertex $v_\text{I} \in V$ is designated as the* initial vertex*, which may be either a decision vertex or a leaf vertex.*

The integer m is called the degree *of the plan. The* width *of the plan is the maximum number of properties associated with any decision vertex,* $\max_{v \in V} |P_v|$. *We write $\mathcal{P}$ for the space of all decision graph plans.*

The intuition is that the system can address a set of m queries by executing a degree m decision graph plan thusly, tracking a current vertex through the execution:

1) Start at the initial vertex.
2) Upon reaching a decision vertex, act in the world to determine the values of the associated properties. (We defer discussion about which actions to execute to Section IV.) Select the appropriate out-edge to follow according to the observed properties of the prevailing situation.
3) Upon reaching a leaf edge, terminate the execution. Return the m YES/NO values labeling this vertex.

For a given query set $\mathfrak{Q}$, a plan π of degree $|\mathfrak{Q}|$, and a situation s, we say that the *outcome* of π in s for each $\mathcal{Q} \in \mathfrak{Q}$ is either YES or NO according to the corresponding label of the leaf vertex reached by this tracing process. A basic example of a plan appears in Figure 2.

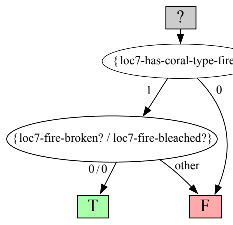
?
{ loc7-has-coral-type-fire? }
1
0
{ loc7-fire-broken? / loc7-fire-bleached? }
0 / 0
other
T
F

Fig. 2. An example of a very small plan that resolves whether there is healthy fire coral at location 7. Reaching the leaf vertex 'T' means the answer is YES, while leaf 'F' means NO.

The key question is whether the outcomes for a particular plan accurately reflect the reality of the world. That is, does the plan answer the question correctly? The next definition formalizes that idea.

Definition 2: A decision graph plan π correctly resolves *a query set $\mathfrak{Q}$ in a world $\mathcal{W}$ if, for each query $\mathcal{Q} \in \mathfrak{Q}$, for every situation $s \in \mathcal{W} \cap \mathcal{Q}$, the outcome of π for $\mathcal{Q}$ in s is* YES*, and for every situation $s \in \mathcal{W} \cap (\mathcal{S} \setminus \mathcal{Q})$, the outcome of π for $\mathcal{Q}$ in s is* NO.

The intuition is that a plan correctly resolves a query set if it accurately and definitely determines, for each query $\mathcal{Q} \in \mathfrak{Q}$, whether the prevailing situation s matches the conditions expressed in $\mathcal{Q}$, that is, whether or not $s \in \mathcal{Q}$.

IV. EXECUTORS AND COST MODELS

The formulation in Section III represents, in a certain sense, a fully-formed planning problem: A planning algorithm would need to select decision nodes and their associated branching structure ensuring that appropriate leaves are reached in each situation. On the other hand, the curious reader may have observed that many of the most important elements of traditional robot planning problems — movements, locations, distances, sensors, optimality criteria and so on— are (so far) absent from this formulation.

This abstraction is intentional, because it forms the basis for a decoupling of decisions about *what* information about the world is useful to produce correct responses to the queries from decisions about *how* that information should be acquired.

Interface between these two separated concerns occurs via *cost models* and *executors*.

Definition 3: A cost model *is a function $c : \mathcal{P} \to \mathbb{R}^{\geq 0}$ that maps decision graph plans to nonnegative real numbers.*

A cost model expresses, at the time a plan is being constructed, how efficiently that plan can be executed.

The appropriate cost model in a given scenario depends on the robot hardware on which the plan will be executed. For full generality, we refrain from imposing any specific structure on the cost model, though there are certainly settings where one might split the cost into contributions from separate terms (e.g., for the robot platform, for the environment, etc.) Given a plan, the executor is the entity responsible for orchestrating the collection of evidence. It begins at a plan's initial vertex and, while the vertex is not yet a leaf, the executor marshals the robot system hardware to collect the evidence needed (in terms of properties) to select an appropriate out-edge. Tracing forward this way, it eventually reaches an outcome.

Example 5 (single Aqua executor): For the coral reef scenario, an executor for a single Aqua robot would determine whether properties hold by navigating the robot to the appropriate site, having it make measurements of specific signals and then employ machine perception techniques (e.g., a sensor plus a machine-learning-based classifier). The executor cares about physical locations for the specific sites in Barbados mentioned in the world in Figure 1 because (and only because) they underpin the mode by which properties are accessed.

The executor encapsulates hardware-specific aspects, but does so to a far greater degree than is usually understood: many of the details of how the system and physical world impinge upon one another are never exposed to the planner. The cost model itself is the sole connection.

Example 6 (single Aqua cost model): To associate a cost model with the executor of 5, note that the example's reef can be modeled as a fully-connected collection of spatially diverse sites (locations), with the travel time between each estimated based on the distance and prevailing currents.

Notice that if the decision vertex has properties at different sites, then this executor must choose a sequence in which to visit those sites. In handling coral reef properties, plans with unit degree never raise such challenges for this executor in handling. But the relationship between properties and locations is not part of the general formulation,[1] *so even*

[1]This is departs from [25], in which the formulation insisted upon directly association between properties and locations.

a plan with unit degree could have left open the question of where to sense provided a property can be ascertained from multiple locations. Contrariwise, plans comprising vertices with high out-degree, but where all the properties can be determined from a single place, are handled straightforwardly by the executor in Example 5. Assuming the executor is only provided plans of this sort, we can give a simple cost model.

Thus, one reasonable cost model associates a cost to plan π by keeping track of the pose of the Aqua. Any specific sequence of properties translates into a sequence of sites in L_{barb} to visit; their cumulative cost of navigation is obtained by summing a number proportional to the time to travel from site to site along the sequence. Additionally, each property has a nonnegative site-dependent scalar associated with observing it (for the time to determine its presence/absence). We compute a worst-case cost for π, taking the maximum across all tracings that reach leaves. (In the implementation, dynamic programming ensures that the value is obtained in time linear in the number of edges in π.)

An important aspect of this formulation is that the plan and the executor are independent objects; a given plan π may be executed with a variety of executors, each with its own cost model. (Note, however, that each plan is likely to be better-suited to low-cost execution with certain executors than with others.)

Example 7 (executor for two Aquas): Suppose the executor now has access to an additional, similarly capable Aqua robot. In this case, by thinking of an executor that pools information from both devices, we can consider both robots operating in concert. Then the executor must resolve how their coordination occurs. Each robot possesses the navigational and perceptual capabilities of Example 5 individually. The executor, being run perhaps on a base station on the shore, employs communication to form the union of properties sensed and determine where each of the robots should be dispatched to.

Analogously with before, we assume that while each plan vertex may have edges with any number of properties, these can be determined from at most two locations. For such cases, an executor might assign each Aqua to the nearer of two locations; if a single location is sufficient, then only the nearest one would move.

Example 8 (two Aqua cost model): This cost model generalizes Example 6 by computing a value, but now with an appropriate simulation of the assignment process employed by the executor.

Both previous executors pursue the problem of knowledge acquisition through the dispatching of mobile sensors. As a third and contrasting example, we consider an alternative, namely, to increase pervasiveness.

Example 9 (executor for a sensor network): Consider a system that consists of seven buoys with suspended sensing packages [29] deployed at locations 1–7 on the reef in Figure 1. Each is solar powered and includes a radio link for communication. In order to ascertain the presence/absence of a property, the buoy at that location wakes, senses its environment and makes a determination, then broadcasts the data over the network.

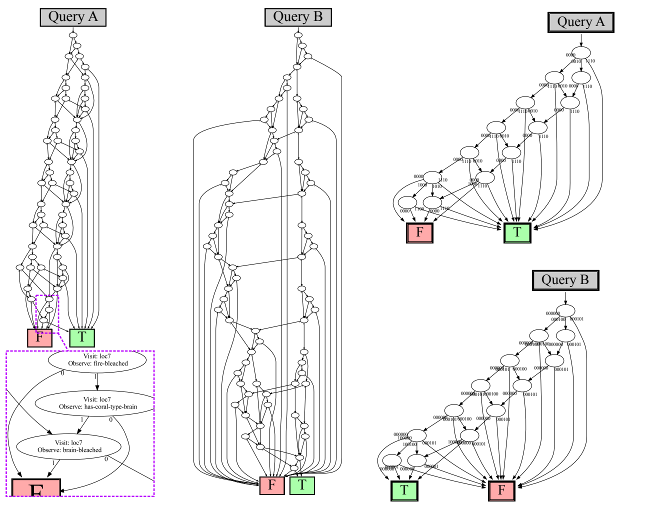

Fig. 3. Plans to acquire knowledge to answer query $\mathcal{Q}_A$ or $\mathcal{Q}_B$ of the coral reef scenario, respectively. At left, the two plans have branches which depend on binary determinations, i.e., they have width one. This can be seen directly in the details included within the inset. By "rolling" those plans so that decisions at each coral site appear together, we obtain the two plans on the right. Though they appear visually similar, the $\mathcal{Q}_A$ plan has width 2, on the $\mathcal{Q}_B$ plan has width 3—the former requires fewer properties to be sensed per location than the latter.

Example 10 (sensor network cost model): A straightforward yet appropriate indicator of cost in this setting may be the number of nodes that wake from their low-power hibernation mode, sense the reef with their sensor package, and communicate the result. The worst-case cost can be computed by identifying the longest chain from initial vertex to a leaf in the given policy π. (To ease interpretation in examples below, we report costs for the sensor network simply as integer values, rather than converting them into conventional units of energy.)

The case study in Section VI explores the impact of these variations in the executor and cost model.

V. Creation, Composition and Operations on General DGs

With the previous examples in mind, we turn to how one initially obtains and then manipulates decision graph plans. First, it is possible that the query $\mathcal{Q}$ that has been posed has either $\mathcal{Q} \subset \mathcal{W}$ or $\mathcal{Q} \subset \mathcal{S} \setminus \mathcal{W}$; these are straightforward, from the perspective information gathering, as the world model itself yields the result. More generally, when logical implication won't suffice, whether the query can be answered or not is dependent upon the executor's power to muster actions that will examine the needed properties. Unlike most planning problems—finding *some* plan is easy: one simply scrutinizes every accessible property.

Our earlier paper [25] employed a specific data structure (the reduced ordered binary decision diagram) in order to represent $\mathcal{W}$ and $\mathcal{Q}$. That work showed how to perform a conditioning operation within that representation to obtain plans which sequence observations in order to answer queries. The leftmost graph in Figure 3 is an example of a plan produced using those methods (along with the inset specifically showing labels consistent with that prior treatment). Those plans apply to a narrower context than has been formulated here and they are a strict subset of those within Definition 1.[2] But this inclusion means they are examples of decision graph plans, and we may use them as a sort of generative starting point.

a) Widening by rolling: Since plans are acyclic graphs, they can be thought of as being layered. If π is a width w plan, we can *widen* to produce a plan π' with width $w' \geq w$ by collapsing two adjacent layers in a single layer labelled with the union of their properties. Focusing on a layer ℓ in π, the decision vertices therein have at most 2^w departing edges. If vertex v has children $v_1, v_2, \ldots, v_k$ on layer $\ell + 1$, then we can rewire edges from v to its grandchildren by forming a Cartesian product of labels. Should any edges departing v skip more than a single layer, one can treat these as though 'virtual' vertices have been introduced along the edge.

Figure 3 shows a pair of unit width plans in the coral reef example. The leftmost one resolves query $\mathcal{Q}_A$ and its immediately adjacent one resolves $\mathcal{Q}_B$. These each correspond to a plan on the right-hand side obtained by widening. In this case, adjacent layers involving properties which can be determined at the same site were rolled together.

b) Pair composition: With a plan π_1 of degree m to answer queries $\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_m$ and another, π_2 with degree n, to answer $\mathcal{Q}'_1, \mathcal{Q}'_2, \ldots, \mathcal{Q}'_n$ there are two ways one might compose them:

– Firstly, through *linear sequencing*. Imagine an executor that, after reaching a leaf in π_1, notes the outcome, and then proceeds to execute from the root of π_2. Upon reaching a leaf, the m earlier values are joined to the n outcomes. This serial progression can, obviously, be represented by a direct operation on the decision graph itself. One simply adjoins suitable copies of π_2 below the leaves of π_1. We denote the result of this operation, an $n + m$ degree plan, by $\pi_1 \to \pi_2$.

A clear interpretation for $\pi_1 \to \pi_2$ is that of *amnesia*: π_2 essentially proceeds to behave as if none of the knowledge obtained for the prior queries has any bearing on it, and memory of the outcome returns at the end.

– Second, we might consider a form of *parallel composition*. At any point during an execution, the values of some properties will have been determined—either directly through measurement, or indirectly via the subset of $\mathcal{W}$ which is consistent with the measurements. The composition $\pi_1|\pi_2$ is a single graph of degree $m + n$, which is built by avoiding superfluous edges that would depend upon known properties. To reduce the complexities in performing this operation, our implementation of $\pi_1|\pi_2$ only applies to two plans with identical orderings

[2]Specifically: they are acyclic graphs that have a strict, non-repetitious global ordering on properties and always collapse subgraphs that are isomorphic to one another. Also, within the terminology of Section III-C, they can only have unit width and unit degree.

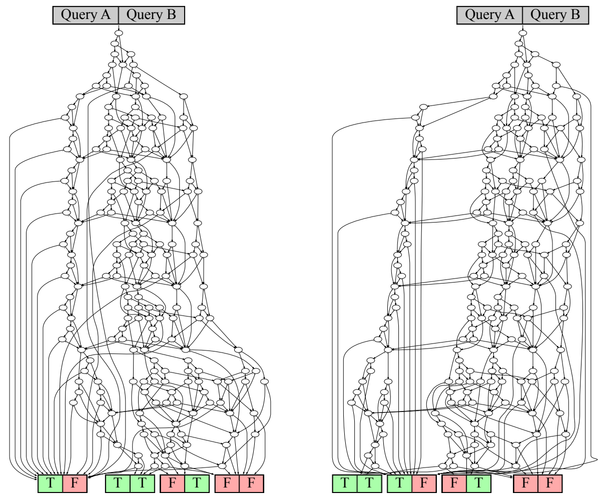

Fig. 4. The degree 2 plan on the left answers both $\mathcal{Q}_A$ and $\mathcal{Q}_B$ together. It is produced via parallel composition of the pair of unit width plans in Figure 3. For the single Aqua cost model this plan has cost 869.24. The plan on the right is the result of optimization under that cost model; it has a cost 816.63.

on all the properties; it produces a result which still preserves this same ordering. Representation of directly known information is encoded in the path from the root, ignoring $\mathcal{W}$. The indirect knowledge (from $\mathcal{W}$) is obtained thereafter by conditioning on $\mathcal{W}$ (via the operation introduced in [25]).

The graph on the left of Figure 4 illustrates a plan of degree 2 obtained via parallel composition of the plans in Figure 3.

c) Cost optimization: Even if it is easy to obtain some plan, one usually desires a plan that is efficient. Manipulations of the orders in which properties appear can directly affect the cost of execution. For a given cost model, one can apply local modifications (like swapping) to alter the order in which decision vertices appear, and then evaluate whether this has improved the plan's efficiency. There are two distinct ways efficiency can increase: either at the knowledge acquisition level, where determining the value of some property obviates the need to obtain the value of some other one; or at the execution level, where some sequence is cheaper to execute than another. For instance, the large body of literature dealing with finding efficient tours seeks these sorts of improvements.

By way of example, Figure 4 gives an instance of effective cost optimization. It considers the cost model with one Aqua and shows how the cost of the plan on the left has been reduced. The initial plan has cost 869.24, while the optimized one on the right saves 52 units of cost.

The preceding operations can be understood as providing a toolbox of basic manipulations which can be made to decision graph plans. Most interesting is that these can be combined in various ways. Figure 5 illustrates multiple pieces at play. It is a plan of degree 2, formed through parallel composition of the plans in Figure 3 in order to answer both $\mathcal{Q}_A$ and $\mathcal{Q}_B$. The plan has width 12, being

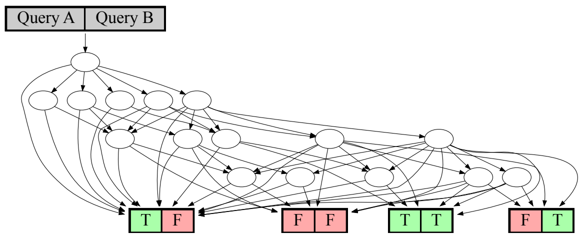

Fig. 5. A degree 2 plan optimized for a pair of Aquas. Using that cost model, it has cost 491.84. The plan has width 12, which generates clutter so that it was necessary to omit the edge labels.

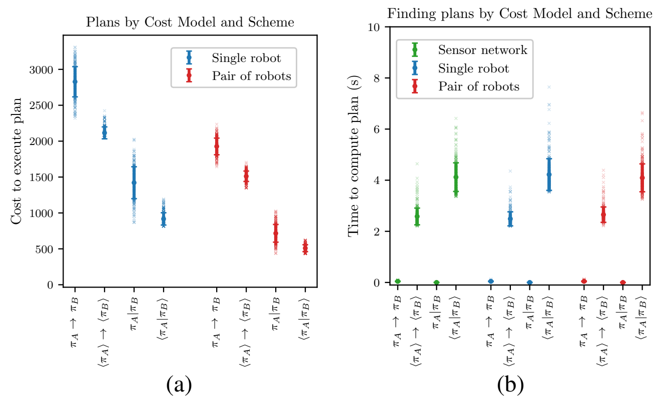

Fig. 6. (a) Plan quality, measured by the cost models, using sequential and parallel composition, optimization, as well as multi-robot costs. Points are from 250 repetitions, means and standard deviations are show. (The value of the sensor network remained a constant 7 for all, so has been omitted from the plot.) (b) Data on timing of operations across cost models. (This reports the commutate time for the same 250 repetitions appearing in Figure 6.) For both, we have written $\langle \cdot \rangle$ to denote an optimization operation.

rolled so that each decision vertex contains properties from at most two locations. Finally, optimization has been applied under the cost model for a pair of Aquas. The version before optimization (not shown) had cost 645.15, while the graph appearing in the figure has cost 491.84.

VI. Demonstrations and Evaluations

This section reports on our experience using our implementation of the aforementioned ideas. The insights obtained should add some extra detail because the discussion dealing with actual plans has, so far, mainly been piecemeal with brief pointers to figures illustrating a concept. We first provide a narrative discussion, to better connect the pieces (in figures) already seen; we then quantify differences.

We have expressed the coral reef domain within our Python implementation for manipulation of decision graph plans. The three cost models for the Aqua (Example 6, the pair of Aquas (Example 8), and the sensor network (Example 10) were implemented and queries $\mathcal{Q}_A$ and $\mathcal{Q}_B$ were used as the basis for the evaluation.

a) Single query (degree 1) evaluations: For $\mathcal{Q}_A$ and $\mathcal{Q}_B$ separately, forming a plan an evaluating it on each of the cost models. The particular plans are shown in Figure 3: for the sensor network cost model, both $\mathcal{Q}_A$ and $\mathcal{Q}_B$ plans have cost 7; for the single Aqua, they have costs 864.24 and 865.24, respectively; for the pair, the costs are 639.15 and 645.15. These are sensible findings: the more complex plan is slightly more expensive. Also, switching from a single robot to a pair results in a considerable saving. Finally, given the queries, the sensor network must, in the worst case, have the node at each of the 7 sites to sense the local properties.

Given the plans for $\mathcal{Q}_A$ and $\mathcal{Q}_B$, we used a block swapping method to optimize each plan. The sensor network model is already at the minimum for both and it remained so. Under the single Aqua cost model the plan costs reduced to 844.75 and 812.63; for pairs, they become 532.61 and 443.11. All represent reductions in cost achieved via effective reordering.

b) Multi-query (degree 2) evaluations: Next, we combined $\mathcal{Q}_A$ and $\mathcal{Q}_B$ into a degree 2 plan. The value of task overlap can be seen by comparing the costs for the combined query with the sum of the previous two. The sensor network has each node awake from hibernation once, so the cost to answer both queries is the same, i.e., 7. The single Aqua takes cost 869.24 (compared to 864.24 + 865.24). (The particular plan with cost 869.24 is the one appearing on the left in Figure 4.) The pair of robots takes 645.15 compared with 639.15 + 645.15.

The previous savings are obtained simply via parallel composition, the next obvious step is to optimize the resulting degree 2 plan. As before, the sensor network still uses 7 units. After optimization, the single robot has a cost of 816.63 (right side of Figure 4). Under the pair costing, after compression, 491.84 units are needed—the plan of Figure 5.

Beyond the examination of particular instances of specific decision graph plans, we also collected statistics across repeated optimization steps. Figure 6 gives a summary of these. The data show that: firstly, in comparing sequential and parallel composition, there is evidence that the latter successfully exploits task overlap; secondly, optimization is consistent in improving performance (except for the sensor network, which is already at a minimum); thirdly, by making a red *vs* blue comparison, the additional capabilities of a second robot can be utilized.

The final plot, Figure 6, shows that optimization is a much more expensive operation than either sequential or parallel composition. It appears to be independent of cost model.

VII. Conclusion & Future Work

The paper has presented an approach to the gathering of information by robots and allied systems. It has treated a problem similar to, but also subtly different from, a vast body of work. Rather than the emphasis being on a particular system, or a specific algorithm, instead it adopts a description centered on a practical representation of how knowledge evolves. The paper identifies operations that can be applied to such objects and shows how the result of doing so can be useful. For instance, it demonstrates how multiple robots might answer multiple queries in concert, using task overlap and their parallelism to obtain efficiencies.

There are many directions for future improvement: a specific one (suggested by a reviewer) is to consider how, in contrast to prior work optimizing individual plans, instead one might optimize plans so that they combine well together.

References

[1] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[2] Jonathan Binney and Gaurav S Sukhatme. Branch and bound for informative path planning. In *Proc. IEEE International Conference on Robotics and Automation*, 2012.

[3] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[4] Yuhong Deng, Naifu Zhang, Di Guo, Huaping Liu, Fuchun Sun, Chen Pang, and Jing Pang. MQA: answering the question via robotic manipulation. In *Robotics: Science and Systems*, 2020.

[5] Richard E Fikes and Nils J Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4):189–208, 1971.

[6] Yogesh Girdhar and Gregory Dudek. Optimal online data sampling or how to hire the best secretaries. In *Canadian Conference on Computer and Robot Vision*, pages 292–298, 2009.

[7] A. Haji-Valizadeh and K. A. Loparo. Minimizing the cardinality of an events set for supervisors of discrete-event dynamical systems. *IEEE Transactions on Automatic Control*, 41(11):1579–1593, 1996.

[8] Steven M. LaValle. Filtering and planning in information spaces. Technical report, Department of Computer Science, University of Illinois, 2009.

[9] Haonan Luo, Guosheng Lin, Yazhou Yao, Fayao Liu, Zichuan Liu, and Zhenming Tang. Depth and video segmentation based visual attention for embodied question answering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[10] Travis Manderson, Juan Camilo Gamboa Higuera, Ran Cheng, and Gregory Dudek. Vision-based autonomous underwater swimming in dense coral for combined collision avoidance and target selection. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and System*, pages 1885–1891, 2018.

[11] Grace McFassel and Dylan A Shell. Reactivity and statefulness: Action-based sensors, plans, and necessary state. *The International Journal of Robotics Research*, 42(6):385–411, 2023.

[12] Max Merlin, Neev Parikh, Eric Rosen, and George Konidaris. Locally observable Markov decision processes. In *ICRA 2020 Workshop on Perception, Action, Learning*, 2020.

[13] Md Modasshir, Sharmin Rahman, and Ioannis Rekleitis. Autonomous 3d semantic mapping of coral reefs. In *Proceedings of the Conference on Field and Service Robotics*, pages 365–379, 2019.

[14] Md Modasshir and Ioannis Rekleitis. Augmenting coral reef monitoring with an enhanced detection system. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1874–1880, 2020.

[15] Petter Ögren and Christopher I Sprague. Behavior trees in robot control systems. *Annual Review of Control, Robotics, and Autonomous Systems*, 5:81–107, 2022.

[16] Robert Platt Jr, Russ Tedrake, Leslie Kaelbling, and Tomas Lozano-Perez. Belief space planning assuming maximum likelihood observations. In *Robotics: Science and Systems*, 2010.

[17] Hazhar Rahmani, Dylan A. Shell, and Jason M. O'Kane. Planning to chronicle: Optimal policies for narrative observation of unpredictable events. *International Journal of Robotics Research*, 2022.

[18] Hazhar Rahmani, Dylan A Shell, and Jason M O'Kane. Sensor selection for detecting deviations from a planned itinerary. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and System*, pages 6511–6518, 2021.

[19] Allison Ryan and J Karl Hedrick. Particle filter based information-theoretic active sensing. *Robotics and Autonomous Systems*, 58(5):574–584, 2010.

[20] Fatemeh Zahra Saberifar, Shervin Ghasemlou, Dylan A Shell, and Jason M O'Kane. Toward a language-theoretic foundation for planning and filtering. *The International Journal of Robotics Research*, 38(2-3):236–259, 2019.

[21] Paolo Salaris, Riccardo Spica, Paolo Robuffo Giordano, and Patrick Rives. Online optimal active sensing control. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 672–678, 2017.

[22] Meera Sampath, Raja Sengupta, Stéphane Lafortune, Kasim Sinnamohideen, and Demosthenis Teneketzis. Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, 1995.

[23] Lukas Schmid, Michael Pantic, Raghav Khanna, Lionel Ott, Roland Siegwart, and Juan Nieto. An efficient sampling-based method for online informative path planning in unknown environments. *IEEE Robotics and Automation Letters*, 5(2):1500–1507, 2020.

[24] David Sears and Karen Rudie. Minimal sensor activation and minimal communication in discrete-event systems. *Discrete Event Dynamic Systems*, 26(2):295–349, June 2016.

[25] Dylan A Shell and Jason M O'Kane. Decision diagrams as plans: Answering observation-grounded queries. In *Proceedings of IEEE International Conference on Robotics and Automation*, 2023.

[26] Tae-Sic Yoo and S. Lafortune. NP-completeness of sensor selection problems arising in partially observed discrete-event systems. *IEEE Transactions on Automatic Control*, 47(9):1495–1499, 2002.

[27] Marko Tosic, Robert B. Bonnell, Pierre Dutilleul, and Hazel A. Oxenford. Runoff Water Quality, Landuse and Environmental Impacts on the Bellairs Fringing Reef, Barbados. In *Remote Sensing and Geospatial Technologies for Coastal Ecosystem Assessment and Management*, Lecture Notes in Geoinformation and Cartography, pages 521–553. Springer, 2009.

[28] Erik Wijmans, Samyak Datta, Oleksandr Maksymets, Abhishek Das, Georgia Gkioxari, Stefan Lee, Irfan Essa, Devi Parikh, and Dhruv Batra. Embodied question answering in photorealistic environments with point cloud perception. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[29] Marios Xanthidis, Alberto Quattrini Li, and Ioannis Rekleitis. Shallow coral reef surveying by inexpensive drifters. In *MTS/IEEE OCEANS - Shanghai*, pages 1 – 9, Apr. 2016.

[30] Xiang Yin and Stéphane Lafortune. A general approach for optimizing dynamic sensor activation for discrete event systems. *Automatica*, 105:376–383, 2019.

[31] Licheng Yu, Xinlei Chen, Georgia Gkioxari, Mohit Bansal, Tamara L Berg, and Dhruv Batra. Multi-target embodied question answering. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[32] Hai Zhu, Jen Jen Chung, Nicholas RJ Lawrance, Roland Siegwart, and Javier Alonso-Mora. Online informative path planning for active information gathering of a 3d surface. In *Proc. IEEE International Conference on Robotics and Automation*, 2021.