2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Hierarchical Multi-objective Planning: From Mission Specifications to Contingency Management

Xuchu (Dennis) Ding Brendan Englot Alessandro Pinto Alberto Speranzon Amit Surana

***Abstract*— We propose a hierarchical planning framework for mission planning and execution in uncertain and dynamic environments. We consider missions that involve motion planning in large, cluttered environments, trading off mission objectives while satisfying logical/spatial/temporal constraints. Our framework enables the decomposition of the planning problem across different layers, leveraging the difference in spatial and temporal scales of the mission objectives. We show that this framework facilitates contingency management under unanticipated events. Interaction between the various layers requires consistent model abstractions and common message semantics. To satisfy these requirements, we adopt a generic knowledge-based architecture that is independent from a specific application domain. We show a specific instance of our framework using a Constrained Markov Decision Process (CMDP) planner at the higher level and a Multi-Objective Probabilistic Roadmap (MO-PRM) planner at the lower level. The resulting planning system is tested in a realistic scenario where an agent is tasked with a mission in a large urban threat rich environment under dynamic uncertain conditions. The mission specification includes a Linear Temporal Logic (LTL) formula that defines the desired behaviors, a list of metrics to be optimized and a list of constraints on time, resources and probability of mission success.**

I. INTRODUCTION

Complex missions for aircraft such as cargo re-supply and medical evacuation in a cluttered, city-like environment are subject to several objectives and constraints. One may need to minimize fuel consumption as well as mission time, maintain a desirable probability of mission success, comply with procedural requirements, etc. Moreover, mission contexts are highly dynamic, forcing the mission management system to react and adapt to new situations. These missions also require agile motion in complex and partially known environments perceived by noisy sensors whose inaccuracy is affected by a dynamic environment with constantly changing conditions such as lighting and weather.

For this class of missions, objectives and constraints can be divided into two major categories: *mission* and *motion*. The first category is related to the overall mission problem and does not capture explicitly the motion aspects. For example, a mission objective could be to reach a landing zone while avoiding no-fly zones, then wait for supplies to be loaded on the aircraft, and finally return to the base, all while minimizing fuel and exposure to hazards. The mission however does not specify how the vehicle should move to avoid obstacles, which trajectories should be followed, and what their costs are. The second category deals with the motion of the vehicle and is concerned with faster time scales that involve dynamic obstacles and threats. The separation between the mission and motion levels hints at the possibility of creating a hierarchical planning system that can solve the entire planning problem more tractably and efficiently.

The major problems to be addressed in building such a system are (1) the abstraction of motions at the mission level including costs and probabilities; (2) the management of contingencies that may arise at any level of the hierarchy.

The main objective of this paper is the definition of a general framework for managing complex missions in cluttered, partially known environments. Such a framework should 1) support optimization with respect to multiple costs, 2) handle logical/spatial/temporal constraints, and 3) deal with contingencies at multiple temporal and spatial scales.

There has been a growing interest in solving these types of problems. Recently, some work has been focused on combining high-level mission specifications with sampling-based motion planners, such as rapidly-exploring random trees (RRT/RRT*) or probabilistic roadmaps (PRM/PRM*). See [1] and references therein for a description of sampling-based planners. Karaman and Frazzoli [2] proposed to combine μ-calculus with RRT or PRM, starting by incrementally building transition systems representing feasible trajectories and then retaining transition systems that satisfy specifications leveraging μ-calculus model-checking. However the representation of mission specifications using μ-calculus can be cumbersome and unnatural to human operators. More recently in [3] and in [4] Linear Temporal Logic (LTL) specifications were instead used for mission specification, in combination with sampling-based planners. However, for large environments these approaches may not scale well, since graph algorithms must be carried out in the product space between the low-level roadmap and the automata corresponding to the specification. In [5] the authors also combined LTL specifications with motion planning based on navigation functions. Although the method enables the generation of paths that satisfy specifications, navigation functions are known to scale badly in high dimensional configuration spaces.

The hierarchical architecture described in this paper is, from a conceptual point of view, similar to 3T [6] and RCS [7] architectures where deliberative planners reside at higher layer and reactive planners are implemented at the lower layer. The architecture proposed in this paper explicitly adds mechanisms to incorporate complex world model defined through an ontology and comprising a knowledge base. Also, the type of planners considered in this paper (both at the deliberative and reactive levels) are not only designed in a way that enable mission execution at multiple scales but they are designed to ensure an effective contingency management. Furthermore, compared to more recent work [8], the architecture and planners, described in

The authors are with United Technologies Research Center. This work was supported by United Technologies Research Center under the Autonomy initiative. Email: {DingX, EnglotBJ, PintoA, SperanA, SuranaA}@utrc.utc.com.

this paper, are as domain independent as possible and can deal with various sources of uncertainty.

In order to enable scalability, expressiveness of the mission specification language, multiple objectives and uncertainty, we previously proposed [9] a hierarchical planning framework in which a large and obstacle-rich environment is partitioned into cells. Detailed path planning is abstracted into motion primitives between cells. High-level actions are then delegated to a path planner that determines the detailed path. This structure accommodates environments whose size are out of reach for traditional path planning. More specifically, in [9] the mission level plan is modeled as the optimal policy of a Constrained Markov Decision Process (CMDP) where actions are motion primitives, while the path planner at the lower level is a PRM. The CMDP planner handles most of the complexity (though at an abstract level) which includes optimizing a primary mission objective under different types of constraints: behavioral specifications as a temporal logic formula and expected cost constraints such as threat exposure. The planning framework described above, while promising, does not address the problem that the abstraction may not be consistent and sound in-between layers, and needs to be augmented to manage contingencies at multiple temporal and spatial scales.

This paper is an extension of the hierarchical planning framework described in [9] both from the functional and architectural perspectives. Specifically, the hierarchical planning framework proposed in this paper presents the following contributions: 1) incorporates multiple objectives at different planning layers, thereby being able to handle a rich class of missions – this also requires extending the PRM path planning algorithm to handle multiple objectives as a multi-objective PRM (MO-PRM); 2) facilitates (re)planning under dynamic, uncertain and unanticipated events; and 3) provides a flexible architecture that enables reusability of modules across mission domains. We show a specific instance of our framework where a CMDP planner is used at the mission level, and a MO-PRM planner is used at the motion level. However, in principle this hierarchical framework can use different planners for both levels (*e.g.,* a graph planner in the higher level and an optimal control solver in the lower level). The resulting planning system is tested in a realistic simulation scenario where an agent is tasked with a mission in a large urban threat-rich environment subject to dynamic contingent events.

II. Hierarchical Planning Framework

A. Mission Model

The types of missions we consider include scenarios in which a vehicle navigates in a highly cluttered environment, such as a city, with partial knowledge of obstacles, e.g. from satellite images or GIS databases. Regions of the map may be labeled with properties such as `Pickup`, `Dropoff` and `SafeRegion`. The desired mission might require visiting these regions in a certain priority order, remaining in a region for a certain time, etc. Bounds on the probability of satisfying the desired behavior are also provided as part of the specification. For example, the mission must be completed with at least 70% probability. Other mission objectives might include exposure to threats, which depends on the threat level assigned to the regions in the map. Therefore, a planning system for these missions must explore the trade-off between mission success probability and threat exposure. However, the exact position of threats in a region is only known at run-time when the vehicle is in the proximity of the region to be traversed. Thus, the planning system should be able to relate the overall probability of mission success to local path segments or, in other words, to operate at different spatial and temporal scales.

Although prior information about the environment is usually available, it is also uncertain and becomes outdated much faster than the mission completion time due to the dynamic nature of the environment. This suggests that solutions in which a very detailed plan is generated for the entire mission starting from the initial conditions may not be suitable.

B. Planner Structure

The planner is comprised of an upper layer, responsible for the mission level management and human interaction, a middle layer that handles the motion planning and a bottom layer that generates feasible control actions from trajectories or waypoints. In the upper and middle layers the planners solve multi-objective optimization problems. These are posed as optimization problems with a primary cost and a set of secondary costs as constraints from which we can obtain the full Pareto solution, as discussed further in Section III. The multi-objective optimization problems are developed such that the primary cost functions and some of the constraints/secondary costs "overlap" between layers. This means that costs/constraints are functions of the same variables, although at different scales. Namely, mission related costs/constraints at the upper layer are abstractions of the corresponding motion costs/constraints at the middle layer. Although the idea of using a hierarchical planner is not new, the way we propose to structure the planners and their interaction is novel and, as we will clarify in the sequel, it offers the required flexibility and robustness to tackle realistic missions in large dynamic environments.

A specific instantiation of the hierarchical planning framework we propose is shown in Figure 1 . At the top, Level I, we have the high-level mission planner based on CMDP, at Level II we have the low-level planner based on MO-PRM and at the Level III a standard trajectory-tracking or waypoint-following controller. At each layer there are well defined planners and their composition must be done through consistent model abstractions. This specifically requires that states and actions are consistent between layers. We further detail this in Section III. Note in Figure 1 we show the "overlap" between the costs at various levels. We will show how these are chosen for a case study in Section V.

There are many advantages of the proposed framework. First of all it enables reusability of planners. Of course, the degree of reusability and domain independency will be higher at the upper layers and decreases as we move down in the hierarchy to the trajectory tracking/waypoint following controller. Secondly, the proposed framework also captures the natural presence of different time and spatial scales. The high-level planner deals with the overall mission whose

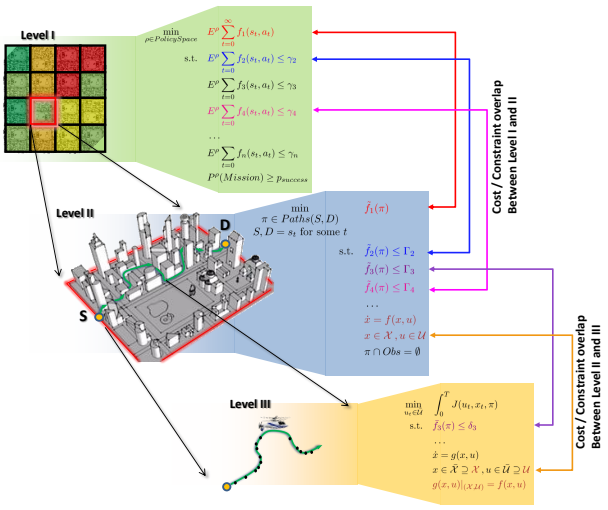


Fig. 1. Overview of the proposed hierarchical multi-objective planner. Level I corresponds to the mission level planner, Level II to the motion/path planner and Level III to a trajectory tracking/waypoint following controller.

domain is generally abstracted into a coarse representation over a large time-scale. The low-level planner determines an obstacle-free path in a subset of the domain providing waypoints at a shorter time scale. This is in general beneficial as, when solving a multi-objective PRM, the search space grows with the number of secondary costs and does not scale well for large scenarios. Further, it also enables contingency management at multiple spatial and temporal scales.

C. Contingency Management

Contingency management is the problem of recognizing, assessing, and responding to unanticipated events or conditions that impact plan execution [10]. These events could occur both externally, to the planning system, or be internally generated. External events include vehicle system/payload degradation/failure, changes in weather conditions, unexpected changes in battle space (e.g. pop-up threat), changes in mission objectives, etc. Internal events could arise due to inability of a lower level planner to execute commands issued by a higher level planner. Often external contingency events lead to internal contingency events, for example due to pop-up threats the lower level path planner cannot meet constraints on exposure commanded by the high-level planner.

Contingency management involves two main steps: 1) identification/assessment and 2) planning/replanning. The first step requires monitoring of actions and keeping the world model up-to-date. This enables the detection of both unanticipated changes in the world and effects of action execution. These aspects are further discussed in Section IV including architectural implications. For the second step one can consider both deliberative/reactive approaches. Deliberative planners account for unanticipated events at the planning stage, while a reactive planner triggers replanning during execution. The replanning problem can be decomposed into plan validation and plan repair. Plan validation verifies if the current plan being executed remains feasible and optimal under contingency. If the plan is no longer valid, plan repair is required. A planner can be reactive to certain events and deliberative about others, and this is a design choice. Below we discuss some design guidelines in context of the hierarchical planning framework introduced above, and argue why hierarchical structure facilitates it.

First, we note that the contingency events described above typically occur at different temporal and spatial scales and have different impacts on mission execution. In a friendly environment, pop-up threats will occur less often compared to a more hostile environment. The knowledge of temporal and spatial scales of the contingency events and their impact on system performance and safety has a profound effect on how contingencies can be managed. For example, if the events are likely to occur frequently or can have great impact on mission execution, a deliberative approach to planning can be taken, compared to reactive planning, which is suitable for events which rarely occur and have low impact on mission performance. On the other hand, contingencies that occur locally (such as discovery of an unknown obstacle along a planned path) can be managed by a lower-level path planner, whereas changes in weather conditions (global change) can be managed by a higher-level planner. Thus, hierarchical planning structure naturally provides a mechanism for separation of concerns without making any single planning layer overly complex.

Secondly, we argue that from a planning perspective the different external contingency events in the physical world can be mapped to a few classes of internal contingency events such as `NoGo` (command cannot start, pre-condition is not met), `Timeout` (execution time for command exceeds a given threshold), `ConstrViolated` (goal can be reached but constraint is violated) and `OffGoal` (execution of command resulted in a state of the world which is different from intended effect of the action). Thus, rather than developing contingency management on a event by event basis, one can develop a planning/replanning approach by focusing on these few classes, with obvious benefits.

As mentioned above, the proposed planner requires overlap between the costs and constraints at different levels. This not only ensures consistency between the various planners but also enables contingency handling at multiple scales. Indeed, if a contingency emerges at one layer, one can recognize two possible outcomes. In the first case, the contingency is detected and managed at the same layer at which it emerges. In that case the system will incur a higher cost but no constraints will be violated. This cost will be used by the higher layer to determine the deterioration of performance, e.g., of the mission or trajectory tracking capabilities. In the second case, the contingency is detected but it cannot be handled by the same layer where it occurs. This will result not only in a higher cost but also one or multiple constraint violations. The lower-lever violation is reported to higher-level layers that will compute a new policy based on this information. Note that a higher layer will not only be informed of the violation, but also its intensity (quantitative amount by which the constraint was violated) thus enabling better re-planning. We will illustrate such aspects of contingency management in a case study in Section V.

III. LAYERS: MISSION AND MOTION PLANNERS

A. High-level CMDP Mission Planner

In this paper, we use a labeled (and finite) multi-objective Markov Decision Process (MDP) or Constrained MDP (CMPD) developed in [9] as the finite model for the high-level planner. Here, we briefly review this CMDP planing framework, for details we refer the reader to [9]. The policy M for this model is one that minimizes expected total primary cost function subject to constraints derived from other cost functions:

$$\begin{aligned} \min_{M\in\mathbb{M}} \quad & J^M_{g_0}(\mathcal{M}), \quad \text{subject to} \\ & J^M_{g_i}(\mathcal{M}) \le c_i\,, \quad i=1,\dots,N \\ & \Pr^M_{\mathcal{M}}(\phi) \ge p_\phi\,, \end{aligned} \tag{1}$$

for a given set of mission constraints $c_i \in \mathbb{R}^+, i=1,\dots,N$ and $p_\phi \in [0,1]$. Here, $\mathbb{M}$ is space of all polices for the MDP $\mathcal{M}$, $J^M_{g_i}$ is expected total cost for cost function g_i, g_0 is the primary cost function and $g_i, i=1,\cdots,N$ are secondary costs. In addition to standard costs functions such as fuel consumption, threat exposure, etc, the labeled CMDP framework also allows one to capture mission specification defined by temporal logic formulas such as Linear Temporal Logic (LTL). The last constraint $\Pr^M_{\mathcal{M}}(\phi)$ in (1) pertains to probability of satisfying logic formula ϕ being greater than a desired probability p_ϕ. In this paper we restrict to *syntactically co-safe* LTL (scLTL) formula (see [11], [12] for details) for which CMDP solution approach was developed in [9]. This approach uses a dual MDP representation and produces an optimal policy as a solution of a linear program (LP). Unlike, MDP optimal policies, the CMDP optimal policies lie in the class of *randomized stationary* policies. Additional details of the solution approach can be found in [9].

B. Low-level MO-PRM Path Planner

We implement high-resolution path planning locally to realize the state transitions of the MDP. The local planning problem is also a multi-objective problem. Continuous multi-objective path planning in two and three dimensions has been achieved by gradient descent, paired with sampling of the Pareto front to identify feasible solutions in the presence of added constraints [13]. Genetic algorithms were applied to multi-objective path planning in [14] and [15]. Multi-objective planning over configuration space roadmaps was considered in [16]. Given such a roadmap, a number of dynamic programming based methods may be used for multi-objective optimization, see [17], [18], [19]. To achieve real-time planning in high-dimensional spaces, RRTs have been adapted to plan under task constraints, leveraging both the rejection of infeasible samples and projection of samples onto constraint manifolds to obtain a feasible solution [20]. Rejection sampling in belief space to satisfy constraints on robot collision probability was proposed in [21].

In this paper we combine the PRM algorithm, which constructs a random graph that maps the collision-free configuration space, with a fast multi-objective graph search [22] to identify paths that lie on the Pareto front, leading to the Multi-Objective PRM (MO-PRM). This approach enables the re-use of a roadmap for searches under different objectives and constraints. A PRM is initially constructed using a Euclidean distance metric to compute edge weights among neighboring nodes. An additional set of edge weights is then computed according to a secondary objective function.

A search is performed over an expanded graph in which each layer contains the nodes of the PRM at a specific level of secondary cost. This representation requires secondary costs to be quantized; the resolution can be tuned to suit the needs of the application. Connections among nodes of the expanded graph represent feasible transitions from one cost level to another. Forcing primary costs to be non-negative, a natural outcome under a Euclidean distance metric, permits the expanded graph to be searched using Dijkstra's algorithm for primary-optimal paths that achieve specific levels of secondary cost. Requiring secondary costs to be strictly positive, which we leverage in our application of the method, ensures that accumulated secondary cost will increase along every step of the path. An efficient upward sweep is possible in this case, which yields worst-case complexity of $O(|N|^2|B|)$, where $|N|$ is the number of nodes in the original PRM and $|B|$ is the total number of cost layers, see Algorithm 1 of [22]. The upward sweep method does not require the explicit construction of the expanded graph, but it does require the selection of a start node or terminal node for which the optimal cost-to-come or cost-to-go is stored and updated for every PRM node, at every secondary cost level, over the course of the search.

Our implementation of the search, summarized in Algorithm 1, begins by selecting a start node s and storing the cost-to-come associated with every PRM node at every secondary cost level. As a preprocessing step, two unconstrained Dijkstra searches over the original PRM are performed with respect to the primary and secondary cost functions, $f(i,j)$ and $g(i,j)$. Let $U_i \in \mathcal{U}$ and $V_i \in \mathcal{V}$ be node i's optimal costs-to-come with respect to the primary and secondary objectives, respectively. Let $\tilde{U}_i \in \tilde{\mathcal{U}}$ and $\tilde{V}_i \in \tilde{\mathcal{V}}$ be the primary and secondary costs associated with V_i and U_i, respectively. After preprocessing, the algorithm then sweeps upward from one level of secondary accumulated cost to the next, and the outcome of each iteration is $W^b_i \in \mathcal{W}$, the primary cost at each node i in the PRM subject to maximum allowable secondary cost $b \in B$. If $b < V_i$, then node i cannot be reached feasibly and $W^b_i = \infty$. If $b \ge \tilde{V}_i$, then node i can be reached using the unconstrained optimal solution and $W^b_i = U_i$. When a limiting case does not apply, the neighbors of node i are examined and the parent node giving minimum primary cost to node i under secondary budget b is adopted. The search will gradually recover the Pareto front of unique paths that lie on the secondary cost levels between V_i and $\tilde{V}_i$. In a representative query to reach a goal node g with maximum allowable secondary cost b_{con}, we assume that a maximum realizable secondary cost b^* is dictated by the discretization scheme.

IV. INTEGRATION AND ARCHITECTURE

In this section we provide details regarding the software implementation of the hierarchy that uses planning algo-

Algorithm 1 $W_g^{b^*} \leftarrow MOPRM(s, g, f(i,j), g(i,j), b_{con}, B)$

$G \leftarrow BuildPRM(s, g, f(i,j))$
$(\mathcal{U}, \tilde{\mathcal{V}}) \leftarrow Dijkstra(s, G, f(i,j))$
$(\mathcal{V}, \tilde{\mathcal{U}}) \leftarrow Dijkstra(s, G, g(i,j))$
for $b \in B, i \in N$ **do**
  if $b \geq V_i$ **then**
    if $b = V_i$ **then**
      $W_i^b \leftarrow \tilde{U}_i$
    else
      if $b < \tilde{V}_i$ **then**
        $W_i^b \leftarrow LocalSearch(b, i, \mathcal{W}, G, f(i,j), g(i,j))$
      else
        $W_i^b \leftarrow U_i$
      end if
    end if
  else
    $W_i^b \leftarrow \infty$
  end if
end for
$W_g^{b^*} \leftarrow RetrieveOptimalPath(\mathcal{W}, b_{con}, g)$
return $W_g^{b^*}$

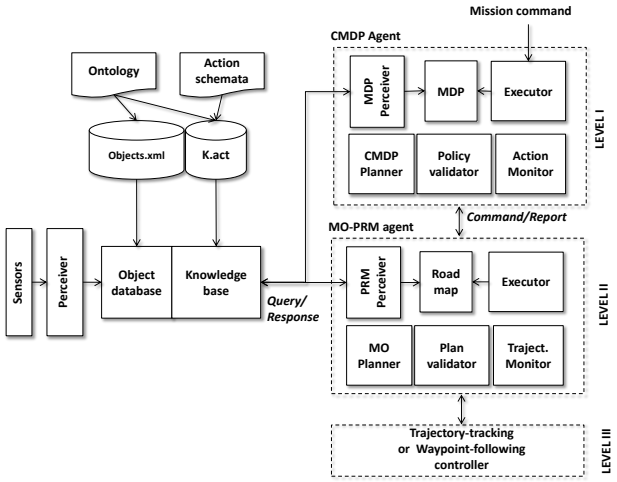


Fig. 2. Hierarchical planner architecture.

rithms outlined in the previous sections. The intent of this section is also to highlight that integrating planning algorithms into a hierarchy to operate in a dynamic world where contingencies may occur requires a considerable amount of effort. The planning algorithms are only two of the 14 blocks in the software architecture (see Figure 2) .

The main features of the architecture presented in Figure 2 are flexibility, re-usability, and run-time adaptability. The resulting mission management system needs to operate in a dynamic environment. Thus, there is a continuous stream of data entering the system through sensors. We use the generic term *sensor* to denote any block that retrieves data related to objects of interest in the world such as GPS/IMU for navigation, map updates, non-cooperative entities and their location etc. Data is processed to create an internal representation of the external world: the *world model*. A *perceiver* denotes a block that interacts with the world model. The first perceiver on the left of Figure 2 transforms sensor data into the world model.

To be flexible and reusable across missions, the world model is divided into a generic database of objects (a procedural representation) and a knowledge base (a declarative one). The object database contains a set of objects O. Each object $o \in O$ has a type $\tau(o)$ and a list of properties $P(o)$. A property is a reference to another object in the database, a list of references to other objects, or a primitive quantity (i.e. Boolean, double etc.). The knowledge base is specified using a multi-sorted First Order Logic and contains the definition of types, predicates, functions and action schemata that define the capabilities of an agent. The knowledge base holds a working memory of facts (i.e. ground predicates) that are true about the world. The object database and the knowledge base are kept consistent[1] and are in fact configured using the same *Ontology*. The ontology only defines object types (e.g. *Helicopter*) and properties (e.g. a Helicopter has a position). A separate file contains the description of actions that an agent is able to execute. This action description is called action schemata. Using a concept similar to the one presented in [23], an action a has a name $\texttt{name}(a)$, a list of typed parameters $V(a) = (v_1, \ldots, v_n)$, a precondition $\texttt{pre}(a)$ which is a conjunction of predicates over the variables $V(a)$ defining when the action is enabled, and a list of effect specifications $\texttt{eff}(a) = \{(p_i, \texttt{eff}_i)\}$. An effect specification is a pair where the first element denotes the probability of the effect to occur, and the second element is a conjunction of predicates over the variables $V(a)$ describing the changes in the state of the system after the action is executed (see Section V for an example of the action schemata).

The probability values associated with state transitions are in general functions of the parameters of the action schema. However, in more complex situations, probabilities depend on details that are not exposed to the level of the planning domain simply because modeling such details would unnecessarily complicate the planning problem. For example, consider the case where the transition probabilities depend on the threat distribution in the environment. Such distribution is not exposed at the level of the knowledge base. However, such information is stored in the object database which has a more detailed representation of objects and specific algorithms can be used to annotate such probabilities. Thus, we leave the flexibility of not specifying the probabilities for each effect and defer their computation to an annotation step that runs only before such information is needed (i.e. to create a probabilistic transition system). Action schemata provide an implicit representation of the state space reachable by the system.

The hierarchical planning architecture is shown on the right of Figure 2. Each agent shares a common architecture where the planning algorithm is only one of the components used at run-time. Agents may have their own internal representation of the world. For example, the high-level agent, which uses the CMDP planner, requires a MDP representation of the world model. A perceiver block is needed to

[1] A specific interface between the two makes sure that the set of constants in the knowledge base is equivalent to the set of objects in the database, and that the working memory of the knowledge base is consistent with the values of the properties of objects in the object database.

read the common world model and generate such internal representation. An executor block manages the run-time decisions that need to be made. For example, when a new mission command is received by the high-level agent, the executor runs the planning algorithm and starts its execution one action at a time. Each action is monitored to check for any contingency and negative reports from the low-level agent. Moreover, changes in the world model propagate into changes to the MDP. Such changes may render the current policy unable to satisfy mission constraints. A plan validator is used to perform such checks and trigger re-planning.

The perceiver block of the CMDP agent is a key component. It needs to query the current state of the system from the world model and unfold the implicit representation of the state space provided by the action schemata into an explicit representation, namely a MDP with additional information such as transition costs and state labels. The perceiver block continuously checks for changes in the world model and keeps the MDP updated. Its main computational steps are the following: (1) Query the current state from the knowledge base; (2) Starting from the current state, perform a full AND-OR graph search using the action schemata also retrieved from the knowledge base; this generates the state and action space of the MDP, and all possible transitions between states; (3) Assign probabilities to the AND transitions and costs to the OR transitions of the generated states (such assignments are called *procedural attachments*); (4) Assign labels to the states of the generated graph (these labels correspond to the ones used in the LTL specification of the mission). Step (2) is the most computationally intensive but can be efficiently implemented. For example, in the case study in V, the MDP containing 2000 states and 30000 actions was generated in 6 seconds).

The MDP generated by the perceiver is used by the CMDP planner to find a policy that satisfies the mission requirements. Once the policy is generated, its execution starts. The executor block of the CMDP agent selects the optimal action according to the policy, say action a, and uses an action map to send the action to the lower level planner. Notice that in our system, the lower level agent is a MO-PRM as described in previous section. This agent accepts only movement commands. Let $Move(Charlie1, HeliPad, Zone1)$ be a movement command. Such command will be associated with constraints coming from the costs attached to the MDP. The MO-PRM agent will query the object database to retrieve the geometric properties of position $HeliPad$ and $Zone1$ and will start running the MO-PRM algorithm to find a set of paths that successfully execute the command. Several contingencies may arise as described in Section II-C. Such contingencies are handled by the CMDP agent either by selecting the next best action according to the policy (deliberative contingency management), or by re-planning (reactive contingency management).

The architecture in the framework proposed in this paper is designed so that all blocks in Fig. 2 are domain independent except the perceivers. Therefore, if the mission scenario is changed, only the ontology, the action schemata and the procedural attachments need to be altered, but the rest of the agents remains untouched and need not be re-implemented.

V. CASE STUDY

A. Mission Scenario

We select a section (10km by 10km) of the city of Chicago [24] as our urban environment, which must be navigated by an autonomous helicopter. The map is a subset $R \subset \mathbb{R}^3$ which includes of a set of buildings (obstacles) $O_i \subset R, i = 1, \cdots, N_b$ each of which can be represented as a polytope. Let $O = \cup_{i=1,\ldots,N_b} O_i$. The complement of O in R is then the free space where the agent can navigate.

We assume that there are threats associated with some buildings with indices $I_{threat} \subseteq \{1, \cdots, N_b\}$. Each threat $i \in I_{threat}$ has a value $e_i \in \mathbb{R}$ indicating the severity of the threat. Given a state trajectory in $\mathbb{R}^3$ from start state $p_s = (x_s, y_s, z_s)$ to goal state $p_g = (x_t, y_t, z_t)$, we define a threat function $T_{(s,t)}$ to quantify the exposure along the trajectory as:

$$T_{(s,t)} = \sum_{i \in I_{threat}} \left[\iiint_s^t \frac{e_i \, dx \, dy \, dz}{\|p_s - p_g\|^2} \right] \tag{2}$$

weighed by threat severity e_i. We assume in this case study that all trajectories are executed at constant speed. Consequently, the threat exposure function, given in (2), penalizes both the proximity to each threat and the duration of exposure. While traversing along a given path, the vehicle can be lost due to threat exposure. In this case the vehicle will be assigned a proposition `Disabled`. The mission task is to minimize the total path length to arrive at a goal region while keeping the probability of being disabled by threats below a certain bound.

B. States and Actions construction

To construct the mission abstraction, we follow the approach from [9]. The map is partitioned into *cells*, and the obstacle free boundary along cell is referred to as the *facets*. With this abstraction, at the CMDP planner level, navigation through the environment reduces to traversing from cell to cell through their facets. In a state s, an action $a \in A(s)$ corresponds to traversing a cell from a facet to an adjacent facet while remaining within the cell. As detailed in Section IV, the CMDP abstraction described above can be automatically constructed by specifying the ontology of the mission scenario and the action schemata. In this case, the ontology is used to establish the notions of points, regions (obstacles), facets, cells and the helicopter. For example, the type facet can be specified by the following ontology description.

```
<Type t_name="Facet">
    <LowerLeft t_name="Point"></LowerLeft>
    <UpperRight t_name="Point"></UpperRight>
    <SharingSameCell t_name="Facet"
        multiple = "true"></SharingSameCell>
    <Goal t_name="bool"></Goal>
</Type>
```

This ontology specifies that a facet is defined by two points (`LowerLeft` and `UpperRight`). Also there is a relation (`SharingSameCell`) between a facet f with a set of other facets, which correspond to the set of facets that share the same cell with f. Finally, there is a boolean function which indicates if a facet is assigned a proposition `Goal`.

The action schema to construct the CMDP model is as follows.

```
Move(Helicopter:agent, Facet:start, Facet:end)
pre    : CurrentFacet(agent,start) and
             SharingSameCell(start,end)
             and not Disabled(agent)
effect: [p1] not CurrentFacet(agent,start)
             and CurrentFacet(agent,end)
        [p2] not CurrentFacet(agent,start)
             and CurrentFacet(agent,end)
             and Disabled(agent)
        [p3] CurrentFacet(agent,start)
```

This action schema specifies that each action (go from a facet `start` to `end`) has three possible outcomes : 1) vehicle finds a path and reaches `end` without being disabled by threats; 2) vehicle finds a path and reaches `end` and disabled by threats; 3) vehicle fails to find a path and thus stays at the same facet `start`. The entire state and action space of the MDP is generated automatically by the perceiver, applying the algorithm described in Sec. IV.

C. Costs and Probabilities

For the low-level MO-PRM planner, every action is characterized by a primary and secondary cost. In this case study, the primary cost of an action is the *length* of the path that executes it, and the secondary cost of an action is the *threat exposure* along the path as per Eq. (2). Prior knowledge of the world is used to derive the expected values of these costs through an offline learning phase. During the offline learning phase, the MO-PRM planner is issued queries for every facet-to-facet pair that may be executed during mission execution. A representative state is selected for each facet, and the query from start state s to goal state t is issued multiple times (*i.e.,* as a Monte-Carlo trial) to assess the frequency with which a collision-free path can be determined by MO-PRM. From this, we can compute the probability of finding a path using the path planner for any facet-to-facet pair i, j; we denote this probability as $\texttt{Pr}_{\texttt{MO-PRM}}(i,j)$. Moreover, the expected primary and secondary costs for each facet-to-facet pair are denoted as $g_{\texttt{length}}(i,j)$ and $g_{\texttt{threat}}(i,j)$, and are also computed from the Monte-Carlo trials. The expected length and threat exposure are set to the mean value of $\tilde{U}_t$ and $\tilde{V}_t$ observed among collision-free queries, respectively.

For the CMDP mission planner, every action is characterized by the transition probability $P(s,a,s')$, where the state $s = (f,d)$ is represented by a facet f and $d \in (1,0)$ (`Disabled` or not). The transition function $P(s,a,s')$ can be constructed in a straightforward fashion using $\texttt{Pr}_{\texttt{MO-PRM}}$ learned in each cell, as described in detail in [9]. Similarly, we abstract costs defined at the MO-PRM level to define a cost function $g_{\texttt{fuel}}(s,a)$ for fuel consumption:

$$\begin{aligned} g_{\texttt{fuel}}((i,0),a_{i,j}) \quad &= \quad Cg_{\texttt{length}}(i,j)\texttt{Pr}_{\texttt{MO-PRM}}(i,j) \\ &\quad +S(i)(1-\texttt{Pr}_{\texttt{MO-PRM}}(i,j)), \end{aligned}$$

where C is a constant exchange ratio between distance and fuel, $S(i)$ is the expected cost of not finding a path for pair (i,j) and thus idling at facet i and $a_{i,j}$ denotes transition from facet i to an adjacent facet j.

Consistency between the costs computed by the MO-PRM planner and the costs and transition probabilities of the CMDP planner is critical for proper functioning of the hiearchical framework, including contingency management, as discussed in Sec. II.

D. Mission Specification and Results

The mission task can be written as the scLTL formula: $\phi = \neg\texttt{Disabled}\,\mathsf{U}\,\texttt{Goal}$. For the mission planner, we formulate the following CMDP problem:

$$\min_{M\in\mathbb{M}} \quad J^M_{g_{\texttt{fuel}}}(\mathcal{M}), \quad \text{subject to}$$
$$\Pr^M_{\mathcal{M}}(\phi) \geq p_\phi,$$

Note that the optimal policy will drive the vehicle to actively avoid threat areas such that the mission success probability is guaranteed to be above p_ϕ.

Choosing $p_\phi = 0.65$, and the cost function $g_{\texttt{fuel}}$ as described in Section V-C, a sample of complete paths generated by the hierarchical planning framework (i.e. CMDP planner interacting with MO-PRM) is shown in Figure 3a.

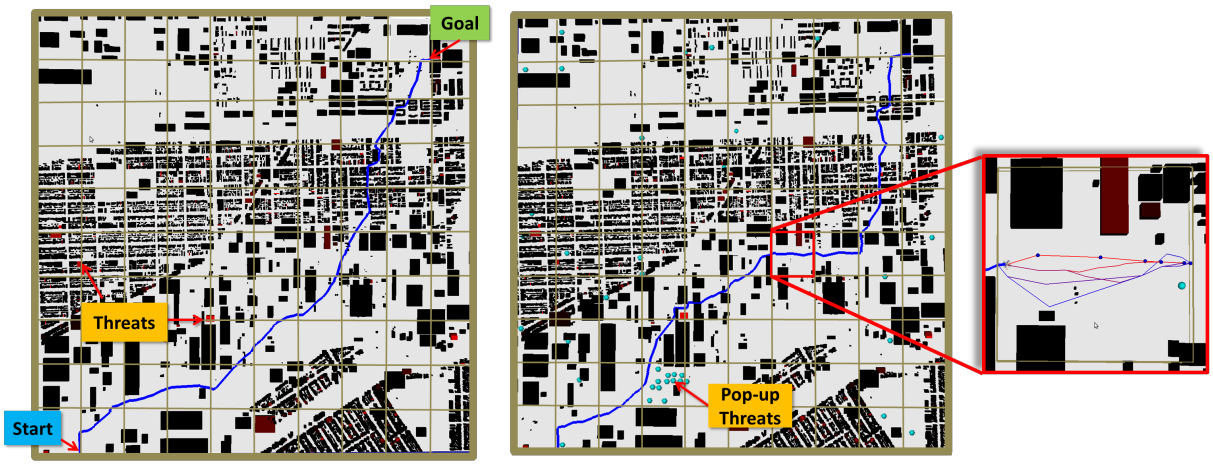
E. Contingency Management

During online mission execution, the expected threat level $g_{\texttt{threat}}(i,j)$ as computed in the offline phase based on prior knowledge is used as the maximum allowable threat exposure when a planning query is sent to the MO-PRM planner. This ensures that our abstraction is consistent and the probabilistic guarantees made by the CMDP planner are met during mission execution. However, if MO-PRM cannot find a collision free path satisfying the constraint, the low-level executor issues a contingency report (and newly perceived expected threat) to the CMDP planner. The higher level plan validator will validate the current plan with respect this information, and mission-level replan will be issued if the plan is no longer valid (*i.e.,* constraint can no longer be met). Note that such contingency falls into the category of `ConstrViolated` as discussed in Sec. II-C.

To test this scheme, we introduced random pop-up threats during the mission execution. A sample path as generated by the hierarchical planning framework with all pop-up threats indicated is shown in Figure 3b. Note that in several cases the low-level MO-PRM could not find valid paths with constraint met, thus high-level re-plans were carried out. Moreover, an example of the path (as well as the Pareto front) produced by the MO-PRM, in the presence of a contingent threat, is shown in Figure 3c.

VI. CONCLUSIONS

In this paper, we describe a novel hierarchical planning framework for mission planning and execution in uncertain and dynamic environments, supporting optimization with respect to multiple costs at different layers, logical/spatial/temporal constraints, and can deal with contingencies at multiple temporal and spatial scales. As part of this framework, we proposed a multi-objective PRM to deal with multiple constraints. In order to effectively implement our framework we also propose a generic knowledge-based architecture which is flexible, re-usable and independent from a specific application domain. We demonstrate a specific instance of our framework using a CMDP planner at a high-level and a MO-PRM planner at the lower level, and explain how the resulting hierarchical planning system is used by



(a) (b) (c)

Fig. 3. (a) Map of environment with a regular cubic disjoint partition. Selected buildings in the map are associated with a certain level of threat. Threat rich buildings are indicated with color red (high threat) and black (zero threat). The path followed by the vehicle when the mission is completed without pop-up threats is shown in solid blue. (b) The path followed by the vehicle when the mission is completed with pop-up threats is shown. Pop-up threats are light blue large dots in the map. We randomly introduced threats to random cells in the environment, and in particular a cluster of threats in cell (4,2). Note that the original plan (cutting across the cell horizontally as shown in (a)) would violates the threat constraint. (c) Pareto front for a query issued to the MO-PRM planner. Threat exposure of each path is depicted along a color gradient from red (minimal length) to blue (minimal exposure). We also show that an unanticipated pop-up threat, in light blue, influences the Pareto front.

an agent to plan/replan for a LTL mission specification in a large urban threat-rich environment under dynamic uncertain conditions.

ACKNOWLEDGEMENTS

The authors would like to acknowledge William M. Sisson II who has developed part of the software that enabled the implementation and testing of the hierarchical planner in simulation. We also thank Prof. Alexander Vladimirsky for helpful conversations on multi-objective path planning.

REFERENCES

[1] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[2] ——, "Sampling-based motion planning with deterministic μ-calculus specifications," *IEEE Conference on Decision and Control*, pp. 2222–2229, 2009.

[3] C. I. Vasile and C. Belta, "Sampling-based temporal locial path planning," 2013, http://arxiv.org/abs/1307.7263.

[4] L. I. R. Castro, P. Chaudhari, J. Tumova, S. Karaman, E. Frazzoli, and D. Rus, "Incremental sampling-based algorithm for minimum-violation motion planning," 2013, http://arxiv.org/abs/1305.1102.

[5] M. Guo, K. Johansson, and D. Dimaragonas, "Motion and action planning under LTL specifications using navigation functions and action description language," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013.

[6] R. P. Bonasso, D. Kortenkamp, D. P. Miller, and M. Slack, "Experiences with an architecture for intelligent, reactive agents," *Intelligent Agents II Agent Theories, Architectures, and Languages. Lecture Notes in Computer Science Volume*, vol. 1037, pp. 187–202, 1996.

[7] J. Albus, T. Barbera, and C. Schlenoff, "Rcs: An intelligent agent architecture," in *Proc. of 2004 AAAI Conference: Workshop on Intelligent Agent Architectures: Combining the Strengths of Software Engineering & Cognitive Systems*, 2004.

[8] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *IEEE International Conference on Robotics and Autonomation*, 2011, pp. 1470–1477.

[9] X. Ding, A. Pinto, and A. Surana, "Strategic planning under uncertainties via constrained markov decision processes," in *IEEE International Conference on Robotics and Automation*, May 2013.

[10] J. L. Franke, A. Hughes, S. M. Jameson, J. G. Clark, and R. J. Szczerba, "Holistic contingency management for autonomous unmanned systems," in *Proceedings of the AUVSIs Unmanned Systems North America*, 2006.

[11] A. Bhatia, L. Kavraki, and M. Vardi, "Motion planning with hybrid dynamics and temporal goals," in *Decision and Control (CDC), 2010 49th IEEE Conference on*. IEEE, 2010, pp. 1108–1115.

[12] O. Kupferman and M. Vardi, "Model checking of safety properties," *Formal Methods in System Design*, vol. 19, no. 3, pp. 291–314, 2001.

[13] I. Mitchell and S. Sastry, "Continuous path planning with multiple constraints," in *Proceedings of the IEEE International Conference on Decision and Control*, vol. 5, 2003, pp. 5502–5507.

[14] P. Vadakkepat, K. Tan, and W. Ming-Liang, "Evolutionary artificial potential fields and their application in real time robot path planning," in *Proceedings of the Congress on Evolutionary Computation*, vol. 1, 2000, pp. 256–263.

[15] O. Castillo, L. Trujillo, and P. Melin, "Multiple objective genetic algorithms for path-planning optimization in autonomous mobile robots," *Soft Computing*, vol. 11, no. 3, pp. 269–279, 2007.

[16] S. LaValle and S. Hutchinson, "Optimal motion planning for multiple robots having independent goals," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 912–925, 1998.

[17] P. Hansen, *Bicriterion Path Problems*. Heidelberg, Germany: Springer Verlag, 1980, pp. 109–127.

[18] J. Jaffe, "Algorithms for finding paths with multiple constraints," *Networks*, vol. 14, pp. 95–116, 1984.

[19] E. Martins, "On a multicriterion shortest path problem," *European Journal of Operations Research*, vol. 16, pp. 236–245, 1984.

[20] D. Berenson, S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *International Journal of Robotics Research*, vol. 30, no. 12, pp. 1435–1460, 2011.

[21] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2011, pp. 723–730.

[22] R. Takei, W. Chen, Z. Clawson, S. Kirov, and A. Vladimirsky, "Optimal control with budget constraints and resets," *CoRR*, vol. abs/1110.6221, 2011.

[23] H. L. S. Younes and M. L. Littman, "Ppddl1.0: An extension to pddl for expressing planning domains with probabilistic effects," Carnegie Mellon University, Tech. Rep. CMU-CS-04-167, 2004.

[24] "City of Chicago - GIS data," http://www.cityofchicago.org/city/en/depts/doit/supp_info/gis_data.html, 2013.