Intent-driven Strategic Tactical Planning for Autonomous Site Inspection using Cooperative Drones*

Dorian Buksz[1], Anusha Mujumdar[2], Marin Orlić[2], Swarup Mohalik[2], Marios Daoutis[2], Ramamurthy Badrinath[2], Daniele Magazzeni[1], Michael Cashmore[1], and Aneta Vulgarakis Feljan[2]

***Abstract*— Realization of industry-scale, goal-driven, autonomous systems with AI planning technology faces several challenges: flexibly specifying planning goal states in varying situations, synthesizing plans in large state spaces, re-planning in dynamic situations, and facilitating humans to supervise, give feedback and intervene. In this paper, we present Intent-driven Strategic Tactical Planning (ISTP) to address these challenges. We demonstrate its efficacy through its application for radio base station inspection across several locations using drones. The inspection task involves capturing images, thermal images or signal measurements - called knowledge-objects - of various components of the base stations for downstream processing. In the ISTP approach, an operator indicates her goals by flying the drone to different components of interest. These goals are generalized to capture the intent of the operator, which are then instantiated in new situations to generate goals dynamically. Towards planning and re-planning in large state spaces to achieve these goals efficiently, we extend the Strategic-Tactical Planning paradigm. All the components of ISTP are integrated in an intuitive UI and demonstrated through a real life use-case built on the UNITY simulator platform.**

I. Introduction

Autonomous drone-based inspection in large-scale dynamic environments such as telecommunications sites can bring significant improvement over manual inspection in terms of safety, time and cost. Task planning in such goal-driven autonomous systems can be performed using automated planning [1] as a core methodology. There is a demand for end-to-end solutions for planning autonomous inspection tasks using robotic agents with human experts in the loop. However, while automated planning as a technology has seen increasing application in several use cases [2], a complete solution for large-scale applications is lacking, and poses several challenges. First, existing planners cannot cope with industry-scale state spaces. Second, accurately capturing and specifying current and goal states automatically in varying situations is complicated. Finally, human operator supervision and agency is key for autonomous AI planning-based systems to be widely adopted and trusted, but production-ready approaches for this do not currently exist. In this paper, we present a solution for planning autonomous inspection tasks using robotic agents such as drones. We pose the planning problem as an *intent-driven planning* problem, and then apply a highly efficient hierarchical decomposition-based planning approach, known as *Strategic Tactical Planning* (STP) to solve it. Together, the solution is termed Intent-driven Strategic Tactical Planning (ISTP).

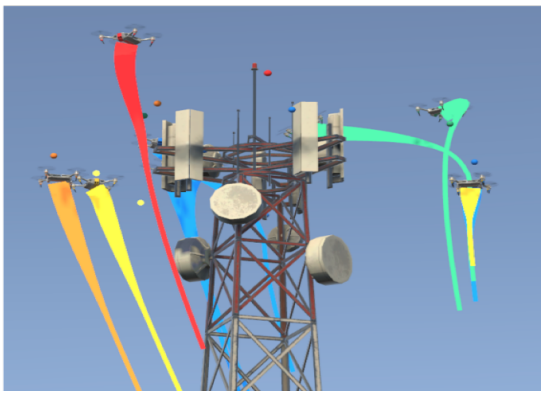


Fig. 1: Drones operated by the ISTP system. Videos at `https://rebrand.ly/istp1` and `https://github.com/EricssonResearch/droneacharya`

We apply ISTP to the context of autonomous inspection of Radio Base Stations(RBS) using drones. RBS contain key equipment such as antennas, radio units, microwave devices etc., which enable mobile communication, and which need to be routinely inspected [3]. Acquisition of *knowledge-objects* such as camera visuals, signal strength, thermal measurements are currently performed manually by field engineers. Such inspection tasks expose engineers to safety risks, aside from being time-consuming and expensive. Therefore, it is envisaged that in the near future, RBS inspections be carried out autonomously by fleets of drones working cooperatively, on installations spread over large geographical areas [3].

In the ISTP approach, an operator indicates her goals by flying the drone to different components of the base station and acquiring intended knowledge-objects. This can be done remotely from within a Network Operating Center (NOC). From this training sequence, the system captures the specific goals achieved. In order to realize true autonomy, the intent behind these goals is derived through generalization. When a different set of base stations are to be inspected by a (possibly different) fleet of drones, the intent is instantiated to new goals, and ISTP derives plans to achieve those and initiates their execution. Finally, the system accounts for dynamic changes in the environment by re-planning for updated current or goal states. The goals, intents, plans and plan execution status are

*This work has been supported by the SCOTT project (Secure COnnected Trustable Things) (www.scottproject.eu), which has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 737422.

! Disclaimer: The document reflects only the author's view and the Commission is not responsible for any use that may be made of the information it contains.

[1]King's College London

[2]Ericsson Research

visible to, and editable by a human.

The primary contributions of this paper are to: (1) introduce an intent-driven approach to task planning where goals are generated on-demand in new situations, (2) extend the STP paradigm to handle planning problems with heterogeneous agents and actions (3) design efficient re-planning methods within STP. The end-to-end solution flow of ISTP is integrated in a tool with an intuitive UI, targeted towards field operations engineers. We also carry out evaluations through simulation of a real-life use case, which demonstrates the efficacy of ISTP.

The rest of the paper is organized as follows. In Section II, we briefly review related literature. Section III describes the autonomous RBS inspection scenario. In Section IV, we formulate the problem and describe the high-level solution architecture and its components. We describe the implementation in V, and perform an evaluation of the solution approach in Section VI. We touch upon scope for future work and conclude in Section VII.

II. Related Work

Automated planning for robotics has a rich literature [4], and it naturally extends to the domain of inspection tasks [5]. The Planning Domain Definition Language (PDDL) [6] has standardized the modelling of AI Planning problems and facilitated the development of planners. Though planning systems have been in use for a while, researchers continue employing various techniques to build more robust and scalable planners for real-life scenarios. Prominent among these are techniques such as Hierarchical Task Networks-based planners [7], that exploit the hierarchical structure of domains, and rely on a top-down approach to exploit pre-constructed decomposition schemes as plans for lower level tasks. In this paper, we extend a hierarchical decomposition technique called Strategic Tactical Planning (STP) [8] for efficient planning and re-planning.

For autonomous operation in dynamic environments, re-planning upon failure is crucial. Various re-planning models, such as re-planning as restart, re-planning to reduce computation and re-planning for multi-agent scenario [9] have been suggested in literature. Our work leverages the *re-planning as restart* approach for mitigating re-planning procedures. Markov Decision Process and Reinforcement Learning based approached are not currently attractive as they require large data sets for training and don't provide safety guarantees.

Literature around intent specification in human-AI systems has been growing [10], and recent research has focused on intents in robotic missions [11] [12], such as in robotic motion planning in the presence of humans [13]. In addition, a closely related area is in systems learning from human demonstrations, which has been increasingly seen to be important in robotics [14]. However, few production-ready solutions include intent specification in large scale planning systems, which is a gap we address in this paper.

III. Autonomous Site Inspection

In this section, we detail the environment and challenges addressed by ISTP. In our scenario, we have an urban deployment of 21 base stations inspired by the habitat of Seoul [15]. We consider 7 roof-top sites with 3 base stations each (Fig. 7). Each base station has components such as antennas and radio units to be inspected. A fleet of 12 drones, equipped with various types of sensors is available for completing the inspections tasks. The state information of the base stations, drones and the environment are considered to be updated in a knowledge base. The challenges manifest as (1) intent specification, and (2) efficient planning and robust execution in large-scale dynamic systems.

We note that state-of-the-art planning techniques need the inspection task goals to be specified as a set of predicates in a PDDL problem specification [6]. However, the large scale of the RBS inspection problem makes such manual specification infeasible. The limited first-order syntax with quantification available in PDDL may help to some extent. However, it is often the case that the required knowledge-objects are specified through complex relations residing in the knowledge base and thus are not modelled in the PDDL description. For example, the operator may want to specify the intent *"inspect only antennas for which more than two work orders have been created in the last month"*, but the work order related knowledge is irrelevant from the view of inspection. Therefore, we address the problem of flexible goal specification using intents specified and evaluated on the knowledge-base.

The second class of challenges, i.e. planning in large-scale dynamic systems, time-critical inspection tasks and geographical spread of the sites, demand scalable methods to generate efficient plans. Planning methods need to be able to exploit the spatio-temporal nature of the problem, as well as heterogeneous capabilities of the drones to scale to large number of base stations and drones while extracting maximum concurrency. In addition, since the environment is highly dynamic, any solution must provide fast re-planning procedures to handle unforeseeable state changes caused by events such as sudden weather changes, drone malfunctions, or if the operator decides to modify inspection tasks during execution. Further, while the solution is expected to be fully autonomous, it is critical that human operators can supervise each process in it and provide feedback, as well as modify goals when necessary.

In order to design a solution which meets the above criteria, we formulate the RBS inspection problem as an intent-driven planning problem, which in turn allows a classical planning problem to be generated on demand from a knowledge base with the most up-to-date state information.

IV. Solution Architecture

The ISTP system architecture for the RBS inspection problem is shown in Fig. 2. It consists of two key blocks - the "Intent-driven" block in blue, and the "Strategic Tactical Planning" block in yellow. The intent-driven block deals with capturing and generalizing the intent of an expert engineer. It includes an intent generator which captures the intent of a new situation as it is detected and generates a classical planning

problem by augmenting information from a knowledge base (KB) with the information from the captured intent.

The strategic tactical planning component takes as input this classical planning problem, and has a planning component to generate plans, an execution module to dispatch actions for execution, and a monitor to check if plans have failed, or there is a change in the goal set, which triggers a re-planning through the re-planning component. The knowledge needed by these blocks consist of (1) state information, i.e., entities and relations in the system, stored in the NOC knowledge base KB_{noc}, and (2) the action templates A_d with domain information containing capabilities of the drones. We first introduce some preliminaries and notations.

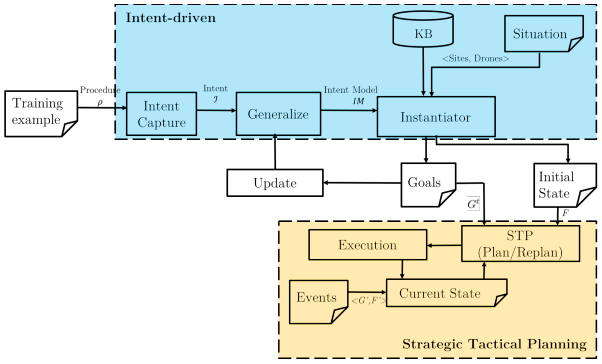


Fig. 2: Block diagram showing the system architecture of ISTP.

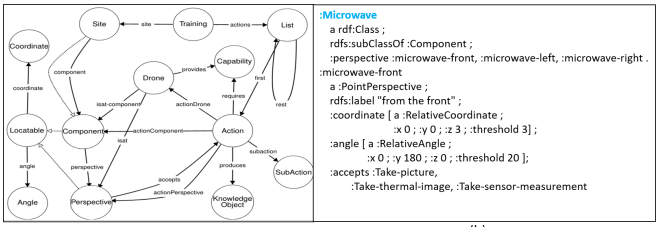
A. Preliminaries and Formulation

Definition: A classical *Planning Problem* (CPP) is defined as a tuple $\Pi = \langle D, F, A, I, G \rangle$, where D is a set of domain elements, F is a finite set of fluents (predicates whose truth value may change over time) over D, A is a set of durative action templates, $I \subseteq F$ represents the initial state, and $G = \{g_1, ..., g_n\} \subseteq F$ is a set of goal states. Each instantiated action $a \in A$ can be interpreted as a state transformer which when applied on a state $\mathcal{S}$ results in a successor state $\delta(a, \mathcal{S})$ after time $dur(a)$. The solution to a CPP Π is a sequence of timestamped actions $\langle a_0, t_0 \rangle ..., \langle a_n, t_n \rangle$ with $a_i \in A$, $t_i \in R$ and $t_i \leq t_j$ when $i \leq j$, such that when the actions are triggered at the timestamps, the initial state is transformed into a final state satisfying G.

An *Intent-driven Planning Problem* can be seen as a higher-level specification from which a classical planning problem is generated on-demand. It is defined as $\langle D, KB, A, \mathscr{S}, \mathscr{R}, \mathscr{I} \rangle$, where D is a set of domain elements, KB is a knowledge base with a set of ground predicates (base) over D and rules to derive other predicates (derived), capturing the declarative knowledge, A is the set of durative action templates capturing the procedural knowledge, $\mathscr{S}$ specifies a subset of the domain entities, $\mathscr{R}$ is a subset of relations in KB, and $\mathscr{I}$, called *Intent*, specifying a set of goals to be derived. We term the pair $(\mathscr{S}, \mathscr{R})$ a *situation* S. An intent is of the form $\bigwedge[q(\overline{X}) \rightarrow p(\overline{X'})]$, where $\overline{X}$ and $\overline{X'}$ are parameters of the fluents p and q, $X' \subseteq X$. An intent selects those predicates $p(\overline{V'})$ for which $q(\overline{V})$ holds true in the knowledge base and $\overline{V}$ and $\overline{V'}$ match on the parameters $\overline{X'}$. With the notation defined, we now detail each of the components in the solution.

B. NOC Knowledge Base

The NOC knowledge base KB_{noc} has an extensible ontology capturing all the aspects related to the operation and management of a telecom network in the form of triples in the Resource Description Framework (RDF) [16]. Fig. 3(a) shows the part of the ontology with the two main concepts in the RBS inspection problem: *component* and *drone*, their attributes and relations. In the ontology, a base station *site* has a number of components viz. antenna, microwave device etc. A *component* has one or more *perspectives* each with *coordinate* and *angle* attributes and a set of *knowledge-objects* (one or more of {image, thermal-image, signal-measurement}) that are available for that perspective. A drone has a set of capabilities (image capture, thermal measurement), positional predicates *is-at-component* and *is-at-perspective*, and device state predicates such as *charge_level*. Fig. 3(b) is an example representation of a Microwave component in the knowledge model. The relations and attributes are defined, and the truth values are kept updated in KB_{noc}, and are accessible by the ISTP system (Fig. I(a) has some examples).



(a) (b)

Fig. 3: (a) Ontology of the site-inspection domain (b) Knowledge about a microwave component, expressed in RDF format in the KB_{noc}

Action templates: The action templates relevant for RBS inspection are *goto, change-perspective, take-image, take-thermal-image, take-signal-measurement, charge*. The templates are encoded as durative actions in a PDDL domain file A_d. In our system, these are encoded within KB_{noc}, to enable updating the action definitions. An RDF-PDDL translator creates A_d. We show the definition of the action template for *goto* in Table I(b). The complete PDDL domain is available at `https://github.com/EricssonResearch/droneacharya/`. With this knowledge about states and actions available in KB_{noc}, operations key to the solution such as intent generalization and instantiation, and CPP generation can be performed. These are described in the following sections.

; Sites, Components and Perspectives (is-perspective ?p - perspective ?c - component) (is-available ?k - knowledge-object ?p - perspective) (has-capability ?d - drone ?c - capability) (is-at ?d - drone ?c - component ?p - perspective) (know ?k - knowledge-object ?c - component ?p - perspective) (is-launch-pad ?lp - perspective) (is-clear-perspective ?p - perspective ?c - component) (is-charging-dock ?c - component) ; Drones (charge-level ?d - drone) (max-charge-level ?d - drone) (distance ?sc ?dc - component) (velocity ?d - drone) (spare-batteries)	(:durative-action goto :parameters (?drone - drone ?srcComp ?destComp - component ?srcPersp ?destPersp - perspective) :duration (= ?duration (/ (distance ?srcComp ?destComp) (velocity ?drone))) :condition (and (at start(is-clear-perspective ?destPersp ?destComp)) (at start(is-at ?drone ?srcComp ?srcPersp)) (at start(is-perspective ?destPersp ?destComp)) (at start(>= (charge-level ?drone)(distance ?srcComp ?destComp)))) :effect (and (at start(is-clear-perspective ?srcPersp ?srcComp)) (at start(not (is-clear-perspective ?destPersp ?destComp))) (at start(not (is-at ?drone ?srcComp ?srcPersp))) (at start(decrease (charge-level ?drone) (distance ?srcComp ?destComp))) (at end(is-at ?drone ?destComp ?destPersp))))
(a)	(b)

TABLE I: (a) Fluents used to define the RBS inspection domain (b) The action template for *goto* in the PDDL domain definition.

C. Intent Capture and Specification

When it is completely clear, an intent can be specified directly as a formula that connects the knowledge objects to the knowledge base. However, most times, it is easier to specify intent through example demonstrations.

This intuitive approach has the advantage of not needing custom specification languages. This is the approach followed in our current solution (refer to Algorithm 1), where an engineer first executes a sequence of actions ρ, collects the knowledge objects G_ρ via $get\_knowledge\_objects(\rho, KB_{noc})$, and then generalizes the concrete goal to get the intent. In our RBS inspection domain, the generalization from a knowledge-object, say, *know(thermal-image, antenna1, top)* resulting in $\mathscr{I}$ is: *know(thermal-image, X, Y)* $\leftarrow$ X.class == antenna1.class $\wedge$ *is-perspective(X, Y)*

This approach of intent capture and generalization is suitable for any domain where there are large number of goal states and where it is complex to specify the goal states for a specific situation directly through a formal intent, e.g. a formula in some specification language.

Algorithm 1: **Intent Capture**(ρ)

1 **Output:** $\mathscr{I}$
2 ρ = get_example_procedure() ;
3 G_ρ = get_knowledge_items(ρ, KB);
4 $\mathscr{I}$ = generalize($\mathcal{G}_\rho, KB$);

Algorithm 2: **generateCPP**($\langle D, KB, A, \mathscr{S}, \mathscr{R}, \mathscr{I} \rangle$)

1 **Output:** $\langle F, A, I, G^t \rangle$
2 $\hat{\mathscr{S}}$ = get_relevant_elements($\mathscr{S}$, KB) ;
3 F = get_relevant_predicates($\hat{\mathscr{S}}$, KB) ;
4 I = get_current_state($\hat{\mathscr{S}}$, KB) ;
5 G^t = derive_goals($\mathscr{I}$, KB) ;
6 return $\langle F, A, I, G^t \rangle$.

From this (derived or specified) intent $\mathscr{I}$, and a given situation $S = (\mathscr{S}, \mathscr{R})$, an IDP $\langle D, KB, A, \mathscr{S}, \mathscr{R}, \mathscr{I} \rangle$ can be formed. The solution to an IDP is a timed plan Π which is obtained by solving a CPP $\langle F, A, I, G \rangle$, derived from the IDP instance. The CPP derivation schema is shown in Algorithm 2. In the *generateCPP* algorithm, all the relevant domain elements $\hat{\mathscr{S}}$ are derived as a reflexive, transitive closure of $\mathscr{R}$ with $\mathscr{S}$ as the base elements[1], the action templates are carried over, I is the set of fluents restricted to $\hat{\mathscr{S}}$ and G is the set of goals derived from KB using $\mathscr{I}$.

D. Plan Synthesis using STP

Having derived a classical planning problem from the IDP, efficient planning and execution are to be performed. In our work, we employ STP due to its promise in efficiently solving large problems, and in re-planning when deviations arise while maintaining planning efficiency. The main challenge in applying STP is identifying scenario characteristics suitable for decompositions. Domains such as the Road Traffic Accident (RTAM) and Driverlog benchmarks, the UAV domain [8] and the domain under consideration in this paper are particularly well-suited for STP's application due to the following characteristics:

- Serializable sub-goals: this means that when a goal has been reached, no action in the domain may undo it.
- Inherent geographic clustering: RTAM's accident location, Drivelog's *at* predicate, the UAV domain missions location and the RBS position allows STP to reason over the domains and form clusters.
- Atomic nature of tasks allows for further clustering

The STP approach decomposes the set of top-level goals ($G^t = g^{t_1}, g^{t_2}, g^{t_3}, ...$) (Algorithm 3 line 1) into serializable subsets (sg^t) and abstracts the obtained subproblems into macro-actions with the g^s strategic goals (Fig. 4). The resulting subproblems have lower complexity than the initial problem and represent the tactical level (more on complexity reduction in section VI-A) .

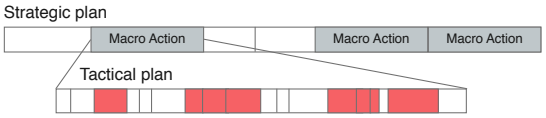


Fig. 4: STP Execution

Algorithm 3: **STPProcedure**(I, G^t)

1 G^s = decompose(G^t);
2 **for** *every strategic goal* $g^s \in G^s$ **do**
3 $\Pi^t(g^s) = \langle D, F, A, I(sg^t(g^s)), sg^t(g^s), \infty \rangle$; // offline
4 **if** $\Pi^t(g^s)$ = *null* **then**
5 **return** false; // exit as problem unsolvable
6 **end**
7 g^s.duration = $\Pi^t(g^s)$.makespan;
8 **end**
9 $I^{'}$ = getCurrentState()
10 $\Pi^s(G^s) = \langle D, F, A, I^{'}(G^s), G^s \rangle$;
11 executed($\Pi^s(G^s)$) = **dispatchStrategic**($\Pi^s(G^s)$); // Algorithm 4 is called
12 **if** *!executed*($\Pi^s(G^s)$)) **then**
13 **jump** to line (2); // trigger strategic replan
14 **end**

Algorithm 4: **dispatchStrategic**($\Pi^s(G^s)$)

1 **for** *action in* $\Pi^s(G^s)$.*actions* **do**
2 **if** *action* = $a^{em}(g^s)$ **then**
3 $I^{'}$ = getCurrentState();
4 $\Pi^t(g^s) = \langle D, F, A, I^{'}(sg^t(g^s)), sg^t(g^s), g^s.duration \rangle$; // online
5 **if** $\Pi^t(g^s)$ = *null* **then**
6 executed($\Pi^t(g^s)$) = false;
7 **return** to STPProcedure line (10) ; // trigger strategic replan
8 **end**
9 executed($\Pi^t(g^s)$) = **dispatchTactical**($\Pi^t(g^s)$); // Algorithm 5 is called
10 **if** *executed*($\Pi^t(g^s)$) **then**
11 $G^s = G^s$ - g^s
12 **end**
13 **else**
14 **jump** to line (3) // trigger tactical replan
15 **end**
16 **end**
17 **end**

Definition: A Tactical Plan is a tuple $\Pi^t := \langle \Pi, T \rangle$, where Π represents a set of timestamped actions that constitutes the plan and T represents the duration of the plan (Algorithm 3 line 3, Algorithm 4 line 4). The strategic level is responsible for the decomposition of G^t based on geographical and inspection type criteria. sg^t represents the goals of a tactical problem, and each tactical problem is encapsulated

[1]Even though $\mathscr{S}$ is finite, $\mathscr{R}$ is necessary to generate a proper subset of the elements in the knowledge base

Algorithm 5: **dispatchTactical**($\Pi^t(g^s)$)

```
1  for action in Π^t(g^s) actions do
2      if g^t ∈ action.effect then
3          completed(action) = dispatch(action);
4          if completed(action) then
5              topLevelGoals(g^s) = topLevelGoals(g^s) - g^t
6          end
7          else
8              executed(Π^t(g^s)) = false;
9              return to Algorithm 4; // trigger tactical replan
10         end
11     end
12 end
```

as a strategic goal $g^s \in G^s$(the set of all strategic goals). STP extends the model with expressive macro-actions that optimize agent behaviour at the strategic goal level.

Definition: Given a strategic goal g^s achieved by the tactical plan $\Pi^t(g^s)$, an *Expressive Macro Action* is the tuple $a^{em}(g^s) := \langle con,\ dur,\ eff \rangle$, where *con* represents conditions that must be valid throughout the execution of the action, *dur* represents the duration estimation of the tactical plan encapsulated by the macro action, and *eff* the effects after the action is completed.

The abstracted representations are used to create a problem with less complexity than the initial problem. STP first computes an offline tactical plan (Algorithm 3 line 3) for each subgoal. The obtained plans are encapsulated via macro-actions as strategic goals g^s to obtain a strategic problem (Algorithm 3 line 9) solvable by state-of-the-art planners. Once the mission begins, the tactical plans are regenerated online (Algorithm 4 line 4) within the schedule of the strategic plan. STP derives the necessary facts for each problem it generates based on the goals (tactical or strategic) of the problem. As we show in the Evaluation section, the STP approach allows handling much larger state-spaces and outputs better quality plans than the purely tactical approach.

Re-planning: STP uses the re-planning as restart [9] methodology when performing a replan. During real-time execution, re-planning may be necessary for several reasons:

- a deviation during execution of a tactical level plan (Algorithm 5 lines 7-9),
- new goals G' injected into the system (Algorithm 1 line 3) with change in G^t),
- changes in action space, e.g., if a particular drone's sensor fails, which prevents certain actions.

Feedback loop in STP:

Constraints are those that may arise after execution starts. These constraints reflect the dynamic aspects of the environment - such as drone malfunctions, sudden weather changes, or an update on the set of inspection tasks.

We distinguish two types of constraints, as shown in Fig. 5. *Pre-start* constraints are those known prior to execution start-time, such as predetermined weather events. These constraints are defined in the model and instantiated in the initial strategic problem. *Post-start constraints*, on the other hand, arise after execution starts, and reflect the dynamic aspects of the environment - such as drone malfunctions, sudden weather changes, or updates on the goals. Such constraints have been determined *a priori* and are included in the domain as possible re-planning triggers, without being instantiated in the initial problem. An instantiation is added to the model only when such a constraint occurs during execution. Any alteration to the initial model will automatically trigger a replan (Fig. 5, with details in section V).

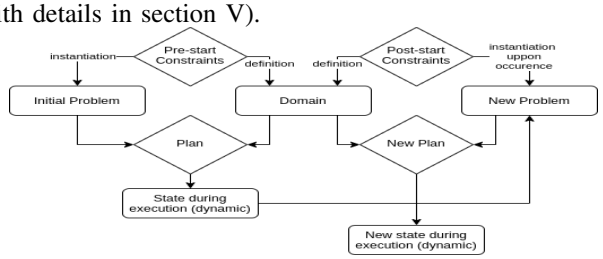


Fig. 5: Pre & Post Start-time Constraints

E. ISTP end-to-end flow

Bringing the above solution pieces together, the complete flow of the ISTP solution is laid out in Algorithm 6. First, the intent is captured from a training procedure ρ provided by a human expert. The generalized intent $\mathscr{I}$ is captured and stored in the KB_{noc}. Alternately, the general intent can also be directly specified by a human. Then, given a situation S, the intent-driven planning problem is converted to a classical planning problem, after which STPProcedure is called for planning and execution.

Algorithm 6: **End-to-end flow of ISTP**

```
1 I = intentCapture(ρ);
2 Given Situation S = (S, R)
3 Form IDP = ⟨D, KB, A, S, R, I⟩ ;
4 ⟨F, A, I, G^t⟩ = generateCPP(D, KB, A, S, R, I) ;
5 π = STPProcedure(I, G^t)
```

V. Implementation Setup

In this section, we detail the specific implementation of the ISTP solution architecture as well as the user interface.

Intent-driven Block Implementation: : The intent capture and generalization components are programmed using SWI-Prolog and using the Cliopatria toolkit [17], with the knowledge base modelled in RDF. Converting the IDP to a CPP is done via the goal and initial state derivation, involving inferencing between the intent formula and the KB, also written in SWI-Prolog. The classical planning problem is modelled in the Planning Domain Definition Language (PDDL) 2.2 [6].

STP block Implementation: : STP is implemented via ROSPlan [18], a framework developed for utilizing PDDL planning in the Robotic Operating System (ROS) [19]. Our implementation, shown in Fig. 6, is an extension of the work presented in [8]. The initial problem and domain are passed to the Strategic Controller node, which decomposes the set of top-level goals G^t into strategic goals (g^s) and generates the strategic problem. During these processes, the tactical plans are generated (Step (a)) and solved (Step (b)) offline in order to compute the macro actions duration estimates. The Strategic Controller uses the estimates to output the abstracted strategic problem to be solved and executed at the strategic level (Step (c)). Once the macro-actions are generated, they are executed by regenerating their equivalent

tactical plan online (Step (d)). As shown in Fig. 6, we extended the previous ROSPlan implementation [8] with multiple tactical planner interfaces that are used concurrently for the offline pre-processing. This resulted in a planning time reduction proportional to the number of added planner interfaces. We also modified the strategic controller node to simulated weather disturbances and drone failures post-start constraints. In a non-simulated environment, the user can input any post-start constraints (Step (e)) via ROS messages [19]. We selected OPTIC [20] as the planner for solving both the strategic and tactical planning problems, as it is a robust temporal and metric planner which supports timed initial literals. A replan as restart [9] is triggered if a plan fails to execute correctly, both at the tactical and strategic levels. In case of a replan at the tactical level, the tactical controller node regenerates the problem with the current state information such as completed goals, and state updates which occurred before the plan failure. In case of a replan at the strategic level, the strategic controller node takes into account completed strategic goals along with state changes that occurred before the replan was triggered, without new subgoal decomposition.

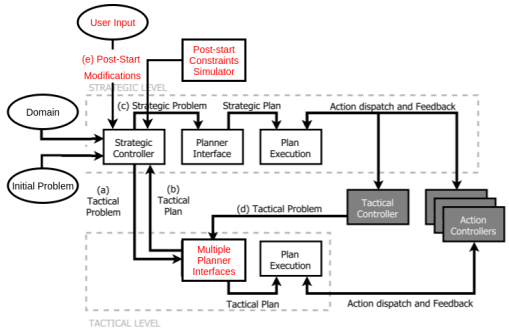


Fig. 6: ROS/ROSPlan Implementation. The items in red represent our extension to the past implementation

User Interface: The user interface (UI) demonstrates the application of the ISTP end-to-end flow. The UI first requests the field engineer to show an example procedure for inspecting a simplified problem, such as a single RBS with one drone. From this procedure, which is a series of actions, the intent is captured and generalized. A new situation is then chosen by an engineer, e.g., "inspect all 21 base stations in a particular section of the city". This new situation, along with the intent generates a planning problem, which is planned for by the STP planner. The trained actions as well as the generated mission can be reviewed and expanded to include extra information, e.g. post-processing tasks. A simulation view runs the mission and collects the resulting knowledge objects. Videos of mission execution visualization are at `https://rebrand.ly/istp1` and `https://github.com/EricssonResearch/droneacharya`. The UI was created in Unity[2], which interfaced with translators from the Knowledge Base (RDF-JSON) and from the Planner (PDDL-JSON)[3]. The Unity model emulates the visual model of the environment, with the base stations, their components and the drone motion.

VI. Evaluation

The Seoul scenario has 21 base stations $\times$ 6 antennas $\times$ 19 inspections = 2394 possible top-level goals g^t and over 3500 entries of initial state information which, consequently, create a huge state space. In this section, we will show our evaluation of the two main blocks of our solution architecture, i.e. intent capture and STP. STP is evaluated against other planning approaches such as purely tactical, hierarchical abstractions and STP to find solutions to our scenario requirements. The results show that STP is not only the best approach in terms of efficient planning, but also mandatory for handling realistically-sized dynamic scenarios. For all the experiments, a Dell XPS 15 9560 laptop was used.

A. STP vs Purely Tactical

Unfortunately, the large state space does not allow for a direct comparison between the STP and purely tactical approaches for the entire model. We attempted this by using the full information to create an input problem for OPTIC and tried to compute a plan in 7200 seconds. However, OPTIC was not able to process the file in order to find a solution in the given time.

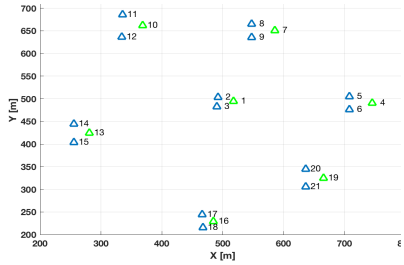


Fig. 7: The city model with 21 base stations, with those in green representing stations with charging points.

To illustrate the superiority of STP over purely tactical, we designed a set of problems with very few elements. The problems have 6 drones with identical configurations, and an incremental number of stations (1-10), with each station having 4 inventory-mapping goals. The results in Table II show that not only does STP drastically reduce the complexity of the search environment in terms of plan time and number of states explored, but also obtains better overall plan makespans.

B. STP Re-planning Stress Test

The ROSPlan implementation allows for tasks to be modified at any time, and has been equipped with services that simulate weather disturbances and drone engine failures. We use the ROSPlan-based simulator to stress-test our system under 3 post-start re-planning scenarios. At stage 1, we simulate the addition of inspection tasks across all 21 radio

[2]Unity Real-Time Development Platform (`https://unity.com/`)

[3]Planner services can be found on the Ericsson Research github here: https://github.com/EricssonResearch/scott-eu. These services assist in translating between the data structures needed by the knowledge model, Unity and PDDL. The services do this by creating and leveraging a shared ontology of PDDL, which include the semantics of actions, objects, and predicates.

Radio Stations	Top-Level Goals	Strategic Goals	Purely Tactical			Strategic Tactical Planning		
			Planning Time	Plan Makespan	States Evaluated	Planning Time	Plan Makespan	States Evaluated
1	4	1	110.08	**23.056**	1780	**8.03**	48.363	**155**
2	8	2	572.15	78.731	5895	**16.19**	**74.889**	**312**
3	12	3	949.78	102.289	8538	**24.8**	**77.785**	**469**
4	16	4	1206.67	352.145	9728	**32.96**	**325.072**	**639**
5	20	5	Failed	Failed	Failed	**41.56**	**365.684**	**796**
6	24	6	Failed	Failed	Failed	**50.52**	**392.21**	**953**
7	28	7	Failed	Failed	Failed	**77.11**	**440.574**	**7848**
8	32	8	Failed	Failed	Failed	**82.37**	**440.574**	**7449**
9	36	9	Failed	Failed	Failed	**91.56**	**444.651**	**7189**
10	40	10	Failed	Failed	Failed	**161.2**	**710.162**	**27124**

TABLE II: Purely tactical vs STP on a set of simplified problems that have from 1 to 10 radio stations, 4 inventory-mapping goals per station and 6 drones of identical configurations. All time in seconds.

stations. At stage 2, we simulate a weather induced flight ban to one of the inspection sites on top of the changes from stage 1. At stage 3, we add malfunctioning drones on top of the modifications present at stage 2. We evaluate each stage for missions with various number of starting tasks.

OPTIC was given 300 seconds for re-planning, during which drones are assumed to stay at an agreed known safe state and wait for the new plan. As illustrated in Table III, our system was able to re-plan and find a solution for the maximum number of inspection tasks possible in the Seoul scenario, regardless of the simulated unforeseen complications.

Initial Tasks	Added Tasks	Total Tasks	Sudden Wind	Drones Malfunction	Replan
252	147	399	✔	✔	✔
504	294	798	✔	✔	✔
756	441	1197	✔	✔	✔
1008	588	1596	✔	✔	✔
1260	735	1995	✔	✔	✔
1512	882	2394	✔	✔	✔

TABLE III: Online re-planning stress test in varying size missions

C. STP vs Precomputed Hierarchical Abstractions

Hierarchical abstractions can be used to deal with large state spaces. To compare the results with STP, we used the same sub-goal decomposition, but in a pre-computed approach. In this approach, the sub-goals equivalent macro-action durations are not generated as in the STP bottom-up methodology. Instead, the duration is estimated using information obtained from previous plans. We used the *a priori* information to create four different types of duration estimates. The *mean* estimate is equivalent to the average duration across all previous tactical plans. The *conservative* estimate assigns the 80th percentile of the previous durations to all macro actions. For the *bucket-mean* estimates, the strategic goals were grouped based on the mission type (image,thermal-image etc.). Then, the average across each type was used for its respective macro-action. The *bucket-conservative* estimates made use of the same grouping present in bucket-mean, but instead used the 80th percentile of each group as an estimate. Strategic problems with varying sub-goals were created with the obtained pre-computed estimates and were evaluated against the actual plan durations. OPTIC was given 1500 seconds to find the best possible solution. As seen in Table IV, several strategic goals are likely to run out of time during execution because of the top-down methods underestimated duration.

Strategic Goal Duration Successful Estimation												
Strategic Goals	Mean			Conservative			Bucket-mean			Bucket-conservative		
	Batch 1	*Batch 2*	*Batch 3*	*Batch 1*	*Batch 2*	*Batch 3*	*Batch 1*	*Batch 2*	*Batch 3*	*Batch 1*	*Batch 2*	*Batch 3*
12	6	3	3	11	6	10	4	4	7	4	3	3
24	12	6	6	19	12	16	8	8	11	8	6	5
36	18	9	9	27	18	22	12	12	15	12	9	7
48	24	12	12	35	24	30	16	16	19	16	12	10
60	30	15	15	45	30	40	20	20	25	20	15	13
72	36	18	18	57	36	50	24	24	33	25	20	16
84	42	21	21	67	42	58	28	28	37	29	23	19

TABLE IV: The number of strategic goals likely to not run out of time during execution for different top-down representations on problems with various size and duration.

D. Purely Tactical Approach with Naive Decompositions

Due to the serializable nature of the planning tasks [21], it may be argued that other naive decompositions, manually performed, can be used to find efficient plans for the scenario in question. Even then, the dynamic aspect of the scenario cannot be mitigated without online re-planning. This makes such decompositions unsuitable. To illustrate this, we attempted to execute the same scenarios as in the Stress Test evaluation. We first decomposed the problem into 4 smaller problems and allocated 3 drones per each individual problem. We gradually added state information and inspection goals so as to have a state space small enough for a planner to compute a plan. We then attempted the exact re-planning scenario stages used in the Stress Test. Stage 1 was successful as we added new goals gradually after the previous ones were completed. However, at stage 2, the simulated sudden wind restricted the time window of an inspection site present in one of the 4 problems, and a plan was not found within the new time bound, thus disqualifying this approach[4].

E. Impact of intent capture

We now turn to the impact of intent capture on the RBS inspection task by illustrating how it avoids the time-consuming and error-prone process of manually enumerating goals for the PDDL problem definition. 2394 goals g^t are possible in the situation we consider. For a new inspection requirement, a naive method would be to display all the goals *know(knowledge-object, component, perspective)* and let the operator select the necessary knowledge-objects based on the criteria at hand - this can get cumbersome each time a new situation occurs. In ISTP, the system generalizes the requirement from example demonstrations ρ by an engineer, e.g., as shown in Table V (left column). This is essentially a learning problem from examples for which techniques such as inductive logic programming, inverse reinforcement learning, active learning and temporal logic mining etc. can be used. The main difference is that the methods can use a rich set of features including, for example, *manufacturer*, *make*, *height* etc. derived from the KB_{noc}, of the objects mentioned in the examples. Note that remaining within the PDDL description of the actions and goals, these features would be inaccessible to help in good generalization.

In the example in Table V, the system can collect the features *manufacturer* and *height* of the antenna in KB_{noc},

[4]At stage 2, the STP implementation called for a re-plan and allocated additional drones to complete the inspection in the new time window, as more drones complete the inspection tasks faster.

and learn that the inspection is meant for knowledge-objects of those antennas that are *owned by Ericsson*, and located above a *height of 100ft*. This is stored in the KB_{noc} as the intent behind the demonstration. With this intent, suppose that 80 antennas across 18 base stations met the height criterion. Then, 80 antennas$\times$(3 image + 2 signal-measurement + 2 thermal-image) perspectives = 560 goal states can be enumerated automatically.

Without the automatic intent capture, generalization and instantiation, these goals need to be manually looked up from the knowledge base for antenna height, perspectives and the signal measurements needed at each perspective, and hand-coded. For each new situation that arises, this manual, error-prone process would be far more pronounced in larger scale problems. By introducing intent-driven planning, we free up valuable human time that could instead be utilized for supervision and approval tasks.

Training procedure ρ	**Generalized Intent** $\mathscr{I}$
take-image RBS-15-antenna-2 front take-image RBS-15-antenna-2 top-left take-image RBS-15-antenna-2 bottom-right signal-measurement RBS-15-antenna-2 top signal-measurement RBS-15-antenna-2 bottom	$know(K, C, P) \leftarrow$ $class(C, \text{“}antenna\text{”}) \wedge$ $owner(C, \text{“}ericsson\text{”}) \wedge$ $height(C) \geq 100$

TABLE V: An example procedure ρ and the captured intent

VII. Conclusions and Future Work

In this paper, we present an intent-driven strategic tactical planning approach for autonomous radio base station inspection using cooperative drones. We describe our end-to-end architecture and solution flow, spanning intent capture and generalization, goal derivation and efficient planning using STP. Efficacy of the approach is demonstrated in a real-life scenario which is marked by very large state space with multiple agents, mission types and dynamic environments while also demanding human supervision and agency. From the literature on automatic survey and inspection of infrastructures, and from customer feedback at the Mobile World Congress 2019, we are positive about the applicability of our approach in other fields like oil industry, shipyard inspection, urban mapping etc.

We highlight a number of future directions for research and implementation. In the current approach, intent capture is implemented only via generalization of goals. Other approaches such as plan/goal recognition could be used to expedite the process of intent capture. We also noticed that STP decomposition sometimes has a negative effect on plan efficiency, which needs to be investigated and characterized further. The current implementation allows only for sequential dispatch of actions. Concurrent dispatch and re-planning seem like a good avenue of future research. We emphasize that a true end-to-end solution will demand deeper research. Firstly, integration with physical robotic agents outside of simulation environments will bring in significant challenges, calling for the execution monitor and re-planning modules to be far more dynamic (e.g. via hybrid approaches like strategic planning and tactical reactive planning). Additionally, intent capture for physical robots will need integration of robust plan recognition modules. Further, plan explanations will be a significant challenge to be addressed in such human-AI collaboration use cases to ensure safety and build trust.

References

[1] M. Ghallab, D. Nau, and P. Traverso, *Automated Planning: Theory and Practice*. Morgan Kaufmann, 2004.

[2] D. Toropila, F. Dvorak, O. Trunda, M. Hanes, and R. Bartak, “Three approaches to solve the petrobras challenge: Exploiting planning techniques for solving real-life logistics problems,” in *2012 IEEE 24th International Conference on Tools with Artificial Intelligence*.

[3] M. Orlic, L. Mokrushin, A. Mujumdar, S. K. Mohalik, N. Linder, and A. Vulgarakis, “Towards zero touch - automating site inspection,” 2019, https://www.ericsson.com/en/blog/2019/3/towards-zero-touch---automating-site-inspection.

[4] F. Ingrand and M. Ghallab, “Deliberation for autonomous robots: A survey,” *Artificial Intelligence*, vol. 247, pp. 10–44, 2017.

[5] M. Cashmore, M. Fox, T. Larkworthy, D. Long, and D. Magazzeni, “Planning inspection tasks for AUVs,” in *2013 OCEANS-San Diego*. IEEE, 2013, pp. 1–8.

[6] S. Edelkamp and J. Hoffmann, “PDDL2. 2: The language for the classical part of the 4th international planning competition,” *4th International Planning Competition (IPC'04), at ICAPS*, 2004.

[7] S. Ontanón and M. Buro, “Adversarial hierarchical-task network planning for complex real-time games,” in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[8] D. Buksz, M. Cashmore, B. Krarup, D. Magazzeni, and B. Ridder, “Strategic-tactical planning for autonomous underwater vehicles over long horizons,” *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[9] K. Talamadupula, D. E. Smith, W. Cushing, and S. Kambhampati, “A theory of intra-agent replanning,” Arizona State Univ Tempe Dept of Computer Science and Engineering, Tech. Rep., 2013.

[10] D. P. Losey, C. G. McDonald, E. Battaglia, and M. K. O'Malley, “A review of intent detection, arbitration, and communication aspects of shared control for physical human–robot interaction,” *Applied Mechanics Reviews*, vol. 70, no. 1, 2018.

[11] C. Menghi, C. Tsigkanos, T. Berger, P. Pelliccione, and C. Ghezzi, “Property specification patterns for robotic missions,” in *Proceedings of the 40th ICSE*, 2018, pp. 434–435.

[12] N. G. Leveson, “Intent specifications: An approach to building human-centered specifications,” *IEEE Transactions on software engineering*, vol. 26, no. 1, pp. 15–35, 2000.

[13] A. Bordallo, F. Previtali, N. Nardelli, and S. Ramamoorthy, “Counterfactual reasoning about intent for interactive navigation in dynamic environments,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 2943–2950.

[14] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, “A survey of robot learning from demonstration,” *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[15] V. Yajnanarayana, H. Rydén, L. Hévizi, A. Jauhari, and M. Cirkic, “5G handover using reinforcement learning,” *ArXiv preprint arXiv:1904.02572*, 2019.

[16] R. Cyganiak, D. Wood, and M. Lanthaler, “RDF 1.1 concepts and abstract syntax,” W3C,” W3C Recommendation, Feb. 2014, http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.

[17] J. Wielemaker, W. Beek, M. Hildebrand, and J. Van Ossenbruggen, “Cliopatria: a SWI-Prolog infrastructure for the semantic web,” *Semantic Web*, vol. 7, no. 5, pp. 529–541, 2016.

[18] M. Cashmore, M. Fox, D. Long, D. Magazzeni, B. Ridder, A. Carrera, N. Palomeras, N. Hurtos, and M. Carreras, “ROSPlan: Planning in the robot operating system.” in *ICAPS*, 2015, pp. 333–341.

[19] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “ROS: an open-source robot operating system,” in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[20] J. Benton, A. Coles, and A. Coles, “Temporal planning with preferences and time-dependent continuous costs,” in *Twenty-Second International Conference on Automated Planning and Scheduling*, 2012.

[21] H. Yu, D. C. Marinescu, A. S. Wu, and H. J. Siegel, “Planning with recursive subgoals,” in *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*. Springer, 2004.