ScRATCHS: Scalable and Robust Algorithms for Task-based Coordination from High-level Specifications

Austin M. Jones[1], Kevin Leahy[1], Cristian Vasile[2], Sadra Sadraddini[2], Zachary Serlin[1,3], Roberto Tron[3], and Calin Belta[3]

[1] MIT Lincoln Laboratory, Lexington, MA, USA,
[2] Massachusetts Institute of Technology, Cambridge, MA, USA,
[3] Boston University, Boston, MA, USA

Abstract. Existing approaches for coordinating teams of heterogeneous agents either consider small numbers of agents, are application-specific solutions, or do not adequately address requirements, e.g., deadlines or inter-task dependencies, common to real-world applications. We develop a framework called Scalable and Robust Algorithms for Task-based Coordination from High-level Specifications (ScRATCHS) to coordinate such teams. We define a specification language, called capability temporal logic (CaTL), to describe rich, temporal properties involving tasks requiring the participation of multiple agents with multiple capabilities, e.g., sensors or end effectors. An example specification is "Ensure at least 10 airborne cameras and 3 airborne lidars are surveying Site A for at least 15 minutes simultaneously during every hour-long period. Make sure that 5 cameras are always observing Site B. Send 10 lidars to Site B within 3 hours of deployment and remain there until 4 ground vehicles with infrared sensors arrive 2 hours later." Arbitrary missions and team dynamics are jointly encoded as constraints in a mixed integer linear program (MILP), which can be solved efficiently using commercial off-the-shelf solvers. ScRATCHS also enables optimization of the resulting plan to be maximally robust to agent attrition at the penalty of increased computation time. The flexible specification language, fast solution time, and optional robustness of ScRATCHS provide a first step towards a multi-purpose on-the-fly planning tool for a supervisor tasking large teams with multiple capabilities enacting missions with multiple tasks. We validate our approach using randomized computational experiments and via a hardware demonstration.

1 Introduction

One of the main challenges of multi-agent systems is the deployment of teams of heterogeneous agents that can work together to complete a task that cannot be performed by a single agent nor by a team of homogeneous agents. An example of such a capability is coordination between airborne robots with downward-facing visual and infrared cameras

This material is based upon work supported by the Under Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Under Secretary of Defense for Research and Engineering.

and ground robots with manipulators capable of moving rubble to find survivors in an urban environment after a natural disaster. The problems of planning and coordination (PAC) for such teams is very complex, as heterogeneity prohibits the arbitrary exchange of one agent for another. It is challenging to develop scalable algorithms for these teams, as more distinct possibilities must be searched when generating a team plan. It is also difficult to develop algorithms that are robust to agent attrition. That is, the inability to re-task any arbitrary agent to assume the responsibility of an agent that has dropped out of the team hampers the ability of a heterogeneous team to "self-reorganize".

Most work in general PAC algorithms for multi-agent systems assumes homogeneity of agent capabilities to avoid these complications [6, 28]. Typically, PAC algorithms for heterogeneous teams are *ad hoc* solutions that depend heavily on subject matter expertise and are specialized to a single family of capabilities/platforms and a single, unique mission [29, 14], or do not consider temporal deployment requirements [22, 4]. In terms of the multi-robot task allocation taxonomy of [16], this fits best in the single-task, multi-robot, time-extended assignment (ST-MR-TA) category.

In this work, we introduce a framework called Scalable and Robust Algorithms for Task-based Coordination from High-level Specifications (ScRATCHS) in which a human supervisor can task a team of heterogeneous agents on-the-fly by specifying a high-level mission. Our framework applies to arbitrary distributions of capabilities among agents and to arbitrary missions expressed as temporal logic (TL) specifications. ScRATCHS removes the need to design custom high-level planners for every new possible combination of robot platforms that can work in a team and allows a supervisor flexibility to accomplish new missions with the available team of robots.

We use tools from formal methods to develop PAC algorithms for agents with heterogeneous capabilities that are scalable with the number of agents and are robust to agent attrition. We achieve scalability by defining and solving mixed integer linear programs (MILPs) that are equivalent to the defined PAC problems. Advances in solution techniques make it possible to solve MILPs with hundreds of thousands of variables and constraints with limited computation time [20, 30, 13]. We achieve robustness by defining and directly optimizing a measure of robustness to attrition that can be computed by mixed integer linear constraints derived from the TL specifications. This approach is complete (up to the precision of the used numerical solver), meaning that if a solution exists, it will be found. Further, by using such an optimization-based approach, we can check very quickly for some kinds of infeasibility, i.e., if the linear relaxation of the MILP is found to be infeasible, the MILP is also infeasible.

ScRATCHS is illustrated in Figure 1. We are given a team of agents whose capabilities, e.g., sensors or end effectors, are known *a priori*. The agents work in a known shared environment partitioned into regions, labeled with the tasks that may be completed in there. An operator overseeing the team of agents has access to a library of *tasks* that the agents can perform. A task description consists of the labels of the regions where the task must be performed, the required number of agents with each type of capability that are required to perform the task, and the amount of time required for the agents to complete the task. From these tasks, an operator uses our specification language, called capability temporal logic (CaTL) to generate a specification that gives absolute or relative timing of task completion, repetition frequencies, and task inter-dependencies such as sequencing or synchronization. Our algorithm then encodes the dynamics of the agents moving throughout the environment, the specification, and the selected measure of robustness into a MILP. The resulting plan from the MILP is handed to a motion planner to generate a collision-free motion plan for the team.

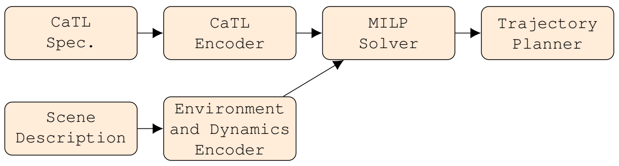


Fig. 1: Schematic overview of ScRATCHS.

Our contributions to PAC for teams of agents with heterogeneous capabilities are:

- The formulation of a specification language called capability temporal logic (CaTL) that can be used to specify behaviors for teams of agents with heterogeneous capabilities with interval-time semantics,
- The encoding of agents moving in a shared environment subject to a CaTL specification as a MILP,
- A notion of robustness measure to optimize a plan's tolerance to agent attrition,
- A complete planning and deployment framework that goes from symbolic level specifications all the way down to motion plans, and
- Extensive computation results that show the performance of our approach and experimental trials involving large teams of heterogeneous robots.

In our approach, we formulate missions as high-level specifications using TL. Temporal logics have seen success for single agents systems [1, 2] and, increasingly, for multi-agent systems [9, 10, 7, 21], including for heterogeneous teams [27]. Much of this work uses automata to capture these specifications, which provide a useful framework for reasoning about specifications, but can lead to issues of computational complexity and scalability. In [3, 17], the authors use automata-based methods to determine independent sub-specifications for individual agents, whose interleaved behavior is guaranteed to satisfy the global specification. We avoid the computational complexity of [3, 17] by describing the joint state of the team as numbers of agents with each capability present in each region [26] at a given time rather than as a product of automata.

The work most related to ours have used MILPs to generate plans for teams of agents under TL specifications [12, 11, 26]. In contrast to [12], which focuses on time-abstract specifications, the logic we define and consider in this paper, Capability Temporal Logic (CaTL), uses interval-time semantics. CaTL is a fragment of signal temporal logic (STL) [19] and thus contains concrete deadlines and other timing requirements. The work in [11] also uses concrete timing requirements, but focuses on the density of homogeneous agents present throughout the environment. Likewise, the authors of [26] propose a method for coordinating heterogeneous teams, but use time-abstract semantics. We, on the other hand, use similar specifications on the number of agents that must perform a particular task, while also specifying precise timing requirements.

Additionally, the named papers consider minimizing total travel time, which can lead to a solution that needlessly fails when individuals fail or are early/late. Our approach maximizes tolerance of the plan to agent attrition.

2 Models and Specifications

In this section, we define models for the kinds of teams we want to coordinate and a specification language, CaTL, for describing behaviors of these systems. We are motivated by the following hypothetical example from precision agriculture.

Example 1. Consider a large farm that grows diverse crops in spatially separated locations as illustrated in Figure 2. Each colored region corresponds to a different type of crop, with the exception of the red regions that correspond to areas which the robots cannot traverse, e.g., areas where heavy equipment are in operation or where the terrain is too rough for the robots to traverse. To aid in monitoring these crops, a fleet of ground based robots with different sensing modalities has been deployed to keep track of plant health. The fleet as a whole has visual(Vis), infrared (IR), ultraviolet (UV), and soil moisture (Mo) sensors. Every robot in the fleet has at most two sensors and the assignment of sensors to robot is fixed *a priori*.Each of the crops in the field has distinct monitoring requirements. The tasks that the team of robots must perform during a 24 hour deployment are listed in Table 1

1. All robots must avoid the red obstacle regions at all times
2. Within 10 hours of deployment, two visual and two IR sensors must remain within each green crop region at the same time for at least 1/2 hour.
3. Half hour soil moisture readings must be made in each blue crop region every 10 hours
4. Within 4 and 12 hours after deployment, two UV and two visual sensors must be in the yellow crop region for a half hour hours
5. Within 1 and 9 hours of deployment and within 10 and 15 hours of deployment, two visual sensors must be in each orange region and remain there for 1 hour.

Table 1: List of precision agriculture tasks

2.1 Environment

Definition 1. The *Environment* is given by a tuple $Env = (Q, E, W, AP, L)$ where:

- Q is a finite set of states that correspond to regions of a workspace
- $E \subseteq Q \times Q$ is a set of edges such that $(q_1, q_2) \in E$ iff an agent in the environment can traverse from the region associated with q_1 to the region associated with q_2 without passing through any other region
- $W : E \to \mathbb{R}$ is an edge weight such that $W((q_1, q_2))$ is the maximum amount of time required for an agent to traverse q_1 before entering q_2.
- AP is a set of atomic propositions that define what types of tasks may be performed in the environment

- $L : Q \to 2^{AP}$ is a mapping that labels each state in the environment according to which tasks may be performed in that region

When constructing an environment from a partitioned workspace, forbidden regions are omitted from Q and transitions to/from those regions are omitted from E.

Example 1 (Continued). The set of crops shown in Figure 2(a) leads to the environment model shown in Figure 2(b). There are 8 regions $\{q_1, \dots q_8\}$. An edge exists between regions if the two regions share a facet or vertex, i.e., if they are connected geographically. The weight between regions is the transit time required to travel from the point farthest away from the shared facet or vertex to the shared facet or vertex. The set of propositions $\{\pi_{green}, \pi_{blue}, \pi_{orange} \dots\}$ in the model corresponds to the types of regions (colored crops/obstacles) and the labeling function applies the labels to the appropriate regions.

2.2 Agents

Let Cap be a finite set of capabilities that an agent can have and let J be a finite index set representing all agents.

Definition 2. An *Agent* $j \in J$ is given by a tuple $A_j = (q_{0,j}, Cap_j)$ where $q_{0,j} \in Q$ is the initial location of the agent in the shared environment and $Cap_j \subseteq Cap$ is a finite set of capabilities.

Example 1 (Continued). The set of capabilities is given by $Cap = \{Vis, UV, IR, Mo\}$. Agent A_1 located in the upper left hand corner of Figure 2 is described by $A_1 = (q_1, \{UV, Mo\})$

Definition 3. An *input signal* for an agent j is a mapping $u_j : \mathbb{R} \to E$ where $u_j(t) = e$ indicates that agent j starts traversing edge e at time t. Each input signal has the properties $u_j(t) = e$ where $e = (q_1, q_2) \Rightarrow A_j$ is in state q_1 at time t and $u_j(t) = e \Rightarrow u_j(\tau) = \emptyset$, $\forall \tau \in (t, t + W(e))$. The input signal u_j induces a *trajectory* of agent A_j, denoted $s_j : \mathbb{R} \to Q \cup E$, such that $s_j(0) = q_{0,j}$ and $u_j(t) = (q_1, q_2) \Rightarrow s_j(\tau) = (q_1, q_2)$, $\forall \tau \in [t, t + W(e)) \wedge s_j(t + W(e)) = q_2$.

Definition 4. Given a team of agents $\{A_j\}_{j \in J}$, let $G \subseteq 2^{Cap}$ be the set of unique combinations of capabilities present in the collection $\{Cap_j\}_{j \in J}$. The *team trajectory* is a mapping $s_J : \mathbb{R} \to \mathbb{Z}^{+,|J| \times Q}$ that maps each time t to the team state $s_J(t) = [n_{Q,G}(t), n_{E,G}(t)] \in \mathbb{Z}^{+,|G|(|Q|+|E|)}$. The vector $n_{Q,G} = [n_{q,g}(t)]_{q \in Q, g \in G}$ is defined such that

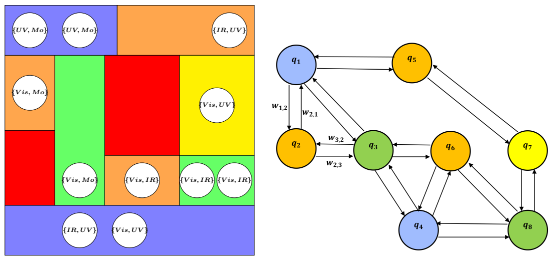
$$n_{q,g}(t) = \sum_{j \in J} I(s_j(t) = q) I(Cap_j = g), \tag{1}$$

where I is the indicator function. That is, $n_{q,g}$ is the number of agents with capability set g in state q. The vector $n_{E,G}(t) = [n_{e,g}]_{e \in E, g \in G}$ is defined such that

$$n_{e,g}(t) = \sum_{j \in J} I(s_j(t) = e) I(g = Cap_j) \tag{2}$$

i.e., $n_{e,g}$ is the number of agents with capability set g that are traversing edge e.

That is, the team trajectory corresponds to how many agents with each type of capability are present in each region and traversing along each edge at each time.



(a) (b)

Fig. 2: (a) Schematic of the precision agriculture motion coordination problem. Region colors correspond to crop types. Discs are robots with their associated capabilities listed. Capabilities include: ultraviolet sensing (UV), moisture sensing (Mo), infrared sensing (IR), and vision (Vis). The team of robots are used for crop monitoring tasks, such as "Within 3 hours of deployment, two visual and two IR sensors must remain within each green crop region at the same time for at least 1 hour." (b) Associated Environment.

3 Capability Temporal Logic (CaTL)

Here, we define the syntax and semantics of capability temporal logic (CaTL), a specification language for teams of heterogeneous agents. The atomic unit of a CaTL formula is a *task*, different from full STL [19], which has arbitrary predicates.

Definition 5. A *counting proposition* $cp_i = (c_i, m_i) \in Cap \times \mathbb{N}$ is true if at least m_i agents with capability c_i are present and false otherwise [26].

Definition 6. A *task* is a tuple $T = (d, \pi, \{cp_i\}_{i \in I_T})$ where $d \in \mathbb{R}$ is a duration of time, $\pi \in AP$ is an atomic proposition, each $cp_i \in Cap \times \mathbb{N}$ is a counting proposition corresponding to how many agents with each capability should be in each region labeled π, and I_T is the index set of counting propositions associated with task T.

In English, a task $T = (d, \pi, \{(c_i, m_i)\}_{i \in I_T})$ is satisfied if for d time units, each of the regions labeled π contains at least m_i agents with capability c_i for all $\{c_i\}_{i \in I_T}$.

Definition 7. The *syntax* of CaTL is given in the Backus-Naur form [18] as

$$\phi := T \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \phi_1 \mathcal{U}_{[a,b)} \phi \mid \Diamond_{[a,b)} \phi \mid \Box_{[a,b)} \phi \tag{3}$$

where ϕ is a CaTL formula, T is a task, $\wedge$ is conjunction, $\vee$ is disjunction, $\Rightarrow$ is implication, $\mathcal{U}_{[a,b)}$ is time-bounded until, $\Diamond_{[a,b)}$ is time-bounded eventually, and $\Box_{[a,b)}$ is time-bounded always.

Definition 8. The *qualitative semantics* of CaTL are defined over pairs (s_J, t) where t is a time index. The semantics of a task are defined as

$$\begin{aligned}(s_J, t) \models T \Leftrightarrow \forall \tau \in [t, t+d), \forall q \in L^{-1}(\pi), \forall \{cp_i = (c_i, m_i)\}_i \in I_T,\\ \textstyle\sum_{g : c_i \in g} n_{q,g}(\tau) \geq m_i\ ,\end{aligned} \tag{4}$$

while the remaining semantics are defined identically to STL [19]. A team trajectory satisfies a CaTL formula ϕ, denoted $s_J \models \phi$, if $(s_J, 0) \models \phi$.

Proposition 1. CaTL is a proper fragment of STL. *Proof (sketch)* Each task T can be expressed as an STL formula. The remainder of syntax and semantics for CaTL is included in STL. Thus, any CaTL formula has an equivalent STL formula. Because CaTL requires that all predicates be part of a task T, STL is not equivalent to CaTL.

Example 1 (Continued). Each of the tasks from Table 1 may be expressed in CaTL as[5]:

1. $\psi_2 = \diamondsuit_{[0,10)}(0.5, \pi_{green}, \{(IR, 2), (Vis, 2)\})$
2. $\psi_3 = \square_{[10,20)}\diamondsuit_{[0,5)}(0.5, \pi_{blue}, \{(Mo, 1)\})$
3. $\psi_4 = \diamondsuit_{[4,12)}(1, \pi_{yellow}, \{(UV, 2), (Vis, 2)\})$
4. $\psi_5 = \diamondsuit_{[1,9)}(1, \pi_{orange}, \{(Vis, 2)\}) \wedge \diamondsuit_{[10,15)}(2, \pi_{orange}, \{(Vis, 2)\})$

In addition to the qualitative semantics of CaTL, which can be used to determine whether a given team trajectory satisfies a specification, we consider the availability robustness, which measures the minimum number of agents J_R that can be removed from (added to) a given team to invalidate (satisfy) the given measure. Optimizing this quantity results in a plan that is robust to agent attrition. Formally, we have:

Definition 9. The *availability robustness* ρ_a of a trajectory is defined

$$\rho_a(s_J, t, \phi) = \begin{cases} \min |J_R| \text{ s.t. } (s_{J \setminus J_R}, t) \not\models \phi & \rho_a(s_J, t, \phi) \geq 0 \\ -\min |J_R| \text{ s.t. } s_{J \cup J_R}, t) \models \phi & \rho_a(s_J, t, \phi) < 0. \end{cases} \tag{5}$$

The availability robustness for a given team trajectory s_J and formula ϕ can be computed recursively using a set of *quantitative semantics*.

Definition 10 (Quantitative Semantics (availability robustness)). The availability robustness of a task is computed recursively according to the following rule

$$\rho_a(s_J, t, T) = \min_{i \in I_T} \min_{t' \in [t, t+d)} \min_{q \in L^{-1}(\pi)} \sum_{g : c_i \in g} n_{q,g}(t') - m_i \,, \tag{6}$$

and robustness for formulas is calculated as it is for STL [8].

4 Problem Formulation and Approach

Here, we formalize the problem we consider in this paper.

Problem 1 (Maximize Availability). Given a team of agents $\{A_j\}_{j \in J}$ operating in a shared environment $Env = (Q, E, W, AP, L)$ and a CaTL specification ϕ, find a set of input signals $\{u_j\}_{j \in J}$ such that $\rho_a(s_J, 0, \phi)$ is maximized.

Problem 1 corresponds to finding a plan for the team of agents that tolerates the most agent drop-out. This problem formulation is useful in situations when agent attrition is likely or in which an agent's ability to complete its part of a task is uncertain. We solve Problem 1 by formulating and solving an equivalent MILP. We encode the qualitative semantics of CaTL as mixed integer linear constraints on trajectories of a discrete-time linear system that models the environment. Formulating Problem 1 as a MILP allows us to use commercial off-the-shelf optimization software with optimized heuristics and solution algorithms to find solutions more quickly than we could by using standard automata-theoretic graph search techniques.

[5] Task 1 is achieved by omitting red regions and transitions from our construction of Env.

5 Integer Linear Programming Encoding

In this section, we reformulate Problem 1 as a MILP. For this purpose, we make the following assumption about the environment:

Assumption 1. The edge weight functions (transition times) are defined such that $W(q, q') = k\delta t,\ k \in \mathbb{N}$ where δt is a time step no larger than the minimum value of W. Further, $u_j(t) = \emptyset\ \forall t \notin \{k\delta t\}_{k\in\mathbb{N}}$, i.e., transitions can only happen at a set of discrete times. That is, the transition times can all be specified by integers.

To enable MILP encodings under Assumption 1, we define a mapping $\mathcal{W} : Q \times Q \to \mathbb{N}$ such that $W((q, q')) = \mathcal{W}((q, q'))\delta t$ for $q \neq q'$. To enable agents waiting at a state q, we define the weights for "self-loops" $\mathcal{W}((q, q)) = 1$.

5.1 Team dynamics

Let $z_{q,g,k} := n_{q,g}(k\delta t)$ be the number of agents with capability set g in the region associated with state q at time index k. Define $u_{e,g,k}$ as the number of agents with capability set g entering e at time $k\delta t$. The initial positions of the agents are encoded in the equality constraints

$$z_{q,g,0} = \sum_{j\in J} I(q_{0,j} = q) I(Cap_j = g)\ \forall q \in Q,\ g \in G\,. \tag{7}$$

We use node and edge balance equations

$$z_{q,g,k} = \sum_{(q',q)\in E} u_{(q',q),g,k-\mathcal{W}((q',q))} \tag{8a}$$

$$\begin{gathered}\textstyle\sum_{(q,q')\in E} u_{(q,q'),g,k} = \sum_{(q',q)} u_{(q',q),g,k-\mathcal{W}((q',q))}\ , \\ \forall q \in Q, g \in G, k = 0, \ldots, K\end{gathered} \tag{8b}$$

where $u_{e,q,k} = 0\ \forall e \in E,\ q \in Q,\ k < 0$. These equations together form a linear system with $O((|Q| + |E|)|G|\Lambda)$ dimensions where $\Lambda := \max_{e\in E} \mathcal{W}(e)$. The inputs to the system $(u_{e,c,k})$ as well as the states are all integer.

Proposition 2. Under Assumption 1, a team input signal $u = [u_j]_{j\in J}$ and the induced team trajectory s_J conform to Definitions 3 - 4 only if a set of variables

$$\{z_{q,g,k}\}_{q\in Q, g\in G, k=0,\ldots K} \cup \{u_{e,g,k}\}_{e\in E, g\in G, k=0,\ldots K}$$

satisfy constraints (7)-(8).

5.2 Task satisfaction

The satisfaction of a CaTL formula ϕ can be converted to a set of mixed integer linear constraints using encodings derived from STL encodings as given in [23, 25]. These encodings consist of binary variables $\{z_{\phi,k}\}_{k\in K}$ such that $z_{\phi,k} = 1 \Leftrightarrow (s_J, k\delta t) \models T$. CaTL formulae can be built from applying recursive encodings to sets of variables and constraints involving $\{z_{T,k}\}$. Here, we give the encodings from the constraints $\{z_{T,k}\}_{k=0}^{K}$ as functions of the variables $\{z_{q,g,k}\}_{q\in Q, g\in G, k=0,\ldots,K}$.

For a given task $T = (d, \pi, \{cp_i\}_{i\in I_T})$, we define a variable $z_{q,c_i,m_i,k} \in \{0,1\}$ that we wish to be valued to 1 if at least m_i agents with capability c_i are in region q at time k. This can be accomplished with the constraints

$$-Mz_{q,c_i,m_i,k} + \sum_{\{g|c_i\in g\}} z_{q,g,k} \geq m_i - M \; , \tag{9}$$

where M is a sufficiently large number, e.g., $M \geq 1 + \max\{\max_i\{m_i\}, |J|\}$. Under a simple assumption of feasibility of the CaTL formula, $\forall i \; m_i \leq |J|$, M can be taken larger than $1 + |J|$. We next define integer variables $z_{\pi,c_i,m_i,k} \in \{0,1\}$ that we wish to be valued 1 if at least m_i agents with capability c_i are in each region $q \in L^{-1}(\pi)$ at time k. This can be accomplished with the set of constraints

$$\begin{array}{c} z_{\pi,c_i,m_i,k} \geq \sum_{q\in L^{-1}(\pi)} z_{q,c_i,m_i,k} - |L^{-1}(\pi)| + 1 \\ z_{\pi,c_i,m_i,k} \leq z_{q,c_i,m_i,k} \; \forall q \in L^{-1}(\pi) \; . \end{array} \tag{10}$$

Next, we define integer variables $z_{\pi,I_T,k} \in \{0,1\}$ that we wish to be valued 1 if at least m_i agents with capability c_i are in each region $q \in L^{-1}(\pi)$ $\forall i \in I_T$. This can be accomplished with the set of constraints

$$\begin{array}{c} z_{\pi,I_T,k} \geq \sum_{i\in I_T} z_{\pi,c_i,m_i,k} - |I_T| + 1 \\ z_{\pi,I_T,k} \leq z_{\pi,c_i,m_i,k} \; \forall i \in I_T \; . \end{array} \tag{11}$$

Finally, we define integer variables $z_{T,k} \in \{0,1\}$ that we wish to be valued 1 if the task will be completed at time $k+d$ and 0 otherwise, written as the constraints

$$\begin{array}{c} z_{T,k} \geq \sum_{\ell=k}^{k+d} z_{\pi,I_T,\ell} - d + 1 \\ z_{T,k} \leq z_{\pi,I_T,\ell} \; \forall \ell = k,\ldots,k+d \; . \end{array} \tag{12}$$

5.3 Objective Functions

Here, we present equivalent MILP encodings for the availability robustness. The encodings recursively define intermediate variables $r_{a,k,\varphi}$ whose values are equivalent to $\rho_a(s_J, k\delta t, \varphi)$ where φ is a subformula of a given CaTL formula ϕ. When φ is non-atomic, the encodings for $r_{a,k,\varphi}$ are equivalent to standard recursive encodings of STL [25, 11]. Note that these encodings require the given formula to be in positive normal form (PNF), i.e., contain no negations ($\neg$).

Proposition 3. Every CaTL formula in PNF is equivalent to an STL formula in PNF.

In what follows, we give the encodings at the atomic task level T, i.e., $r_{a,k,T}$. Following the conventions of [25], we replace (9) with

$$\sum_{\{g|c_i\in g\}} z_{q,g,k} - m_i + M(1 - z_{q,c_i,m_i,k}) \geq r_{a,0,\phi}, \tag{13a}$$

$$\begin{array}{c} \sum_{\{g|c_i\in g\}} z_{q,g,k} - m_i - Mz_{q,c_i,m_i,k} \leq r_{a,0,\phi} \\ \forall k = 0,\ldots,K \forall (c_i, m_i) \text{ appearing in } \phi \; . \end{array} \tag{13b}$$

As pointed out in [24], applying recursive quantitative semantics of any STL formula in PNF leads to compositions of minimum and maximum operators applied to predicate values over time. Thus, the value of $\rho(s_J, 0, \phi)$ must be equal to some value of the *margin* of a predicate at a certain time, i.e., by how much a function of the value of that signal exceeds (or falls below) the constant bound of a predicate. In our case, this corresponds to how many agents of a certain capability c_i exceed the threshold m_i ($\sum_{\{g|c_i \in g\}} -m_i$) or how many more would need to be added to meet m_i ($m_i - \sum_{\{g|c_i \in g\}}$).

Proposition 4. Problem 1 is equivalent to solving the integer linear program

$$\max_{\{u_{e,g,k}\}} r_{a,0,\phi} \text{ subject to } (7) - (8),\ (10) - (12),\ (13)\,. \qquad (14)$$

Proof. This follows directly from Prop. 2, Prop. 3, and Theorem 1 from [24].

6 Computational Experiments

Here we characterize the computational requirements of our methodology via an extension of the precision agriculture case study used as a running example throughout this paper. We consider a fixed specification $\phi_{pa} = \bigwedge_{i=2}^{5} \psi_i$ for all of the experiments. All computational times, number of variables, and number of constraints for experiment are summarized in (mean/max) format. All robustness values are given as means. For these experiments, we focus on the problem of region planning and ignore low-level motion planning. We consider the results of solving Problem 1 completely (denoted as the "Robust" method) and of solving the problem until the first feasible solution, i.e., one which satisfies ϕ_{pa}, is found (denoted the "Feasible" method). These experiments were performed on a PC with 32 cores with 2.10 GHz processors and 64 GB of RAM.

Characterization We generate 50 random 3×3 grid transition systems. Edge weights are chosen uniformly from $W = \{1, 2\}$. The label of each region in the graph is drawn uniformly from $AP = \{\pi_{blue}, \pi_{orange}, \pi_{yellow}, \pi_{green}\}$. For each randomly generated environment, we consider teams of 20 agents from four classes each with two capabilities drawn from $\{Vis, UV, IR, Mo\}$. We ensure all individual capabilities are covered in the classes. The initial states of each of the agents are selected uniformly at random.

We solve Problem 1 for each of these instances. We record the time to achieve the solution, the number of variables, number of constraints, and availability robustness ρ_a. Results from these experiments are shown in Table 2. These results indicate that

Method	Computation Time (s)	Variables	Constraints	ρ_a
Robust	4.63/43.44	14887/15027	10946/11238	1.54
Feasible	0.72/0.94	14931/15067	11005/11359	0

Table 2: Summary statistics (mean / max) for Experiment 1.

although the encoded MILP for this problem can be quite large, the computation time is reasonably short. There is a large difference in time between the time to find an optimal solution and the time to find the first feasible solution. Because there are excess agents available to perform this task, there are many satisfying solutions. However, finding the most robust solution requires more time to search through these solutions.

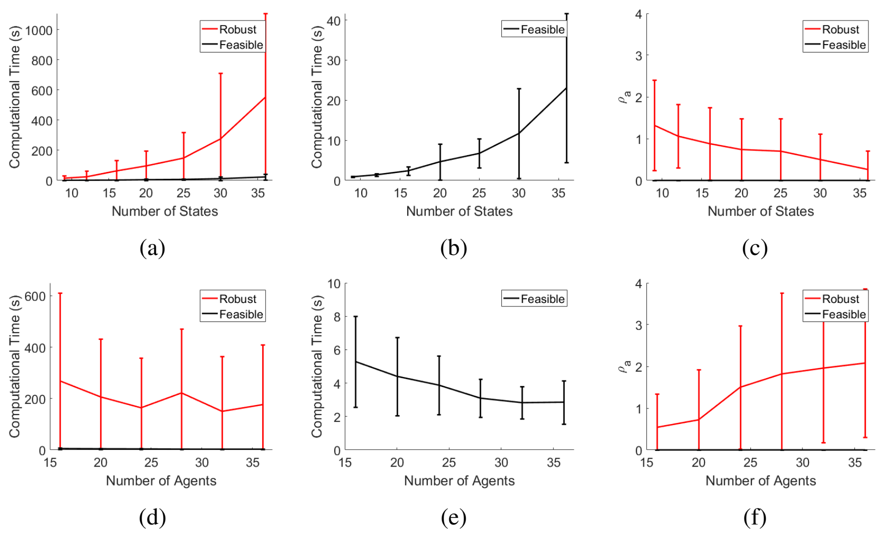


Fig. 3: Results for Experiment 2 varying the number of states (top) and Experiment 3 varying the number of agents (bottom). The first column shows overall computational time to find the first feasible solution and maximally robust solutions. The second column shows computational time to find the first feasible solutions, and the third column shows availability robustness for both the maximally robust and first feasible solutions.

Scalability with Environment Size For this experiment, we maintain the team size at 20 and vary the number of states in the environment. The results are summarized in Table 3 and visualized in Figures 3a, 3c, and 3e. These results indicate that the size of the environment (and thus the number of tasks that are required to be performed) can have a large effect on the computation time required to achieve a solution. This is due to an increase in the number of variables that must be tracked, the number of constraints used to describe the environment, and an increase in the length of the expected paths of the agents. In future work, this effect may be mitigated during the process of abstracting the environment model by grouping adjacent regions in which no service is required. -12pt

Method	Env. Size	Comp. Time (s)	Method	Env. Size	Comp. Time (s)	ρ_a
feasible	9	0.91/1.27	robust	9	14.87/87.13	1.32/
feasible	12	1.40/2.39	robust	12	23.92/249.72	1.06
feasible	16	2.39/6.29	robust	16	62.82/324.78	0.88
feasible	20	4.65/33.98	robust	20	96.22/416.52	0.74
feasible	25	6.74/22.47	robust	25	147.84/864.703	0.70
feasible	30	11.70/67.92	robust	30	275.49/3062.48	0.50
feasible	36	23.05/91.79	robust	36	550.94/2441	0.26

Table 3: Summary statistics (mean / two standard deviation) for Experiment 2.

Scalability with Team Size For this experiment, we maintain the environment size at 25 and vary the size of the team. The results are summarized in Table 4 and illustrated in Figures 3b, 3d, and 3f. Note that increasing the number of agents does not increase the computation time of either the first feasible solution or the optimal solution. This is due to the fact that as we increase the number of available agents, more feasible solutions are available at higher levels of availability robustness, i.e., there are more "good" solutions from which to choose. Increasing the number of agents will, however, make the motion planning problem more challenging, especially in tight workspaces.

These findings indicate that ScRATCHS can solve practical problems in the deployment of heterogeneous teams. The gap between the very fast time to a first feasible solution and the time between a more robust solution implies that presenting an "anytime" planning tool to a human supervisor would be useful, allowing them to make a trade off between timeliness and quality of solution. Improvements in computation time could be made by limiting the number of states in the environment one has to consider, similar to [31], and by developing a smoother or more granular approximation to the availability robustness.

Method	Team Size	Comp. Time (s)	Method	Team Size	Comp. Time (s)	ρ_a
feasible	16	$5.29/14.82$	robust	16	$268.44/1978.16$	0.54
feasible	20	$4.40/15.25$	robust	20	$206.43/897.0$	0.72
feasible	24	$3.88/11.24$	robust	24	$163.99/853.28$	1.50
feasible	28	$3.10/7.26$	robust	28	$221.80/1328.00$	1.82
feasible	32	$2.83/7.00$	robust	32	$150.10/1513.85$	1.96
feasible	36	$2.86/9.12$	robust	36	$176.23/1015.38$	2.08

Table 4: Summary statistics (mean / two standard deviation) for Experiment 2.

7 Hardware Demonstration

To assess the implementability of ScRATCHS, we performed a hardware demonstration of the above algorithms using 10 heterogeneous robots with three unique platforms and four unique sensing capabilities in an indoor motion capture environment. The demonstration mimics a precision agriculture scenario where a team of robots must inspect crops, harvest crops, survey for pests, deter pests, and estimate water reservoir levels simultaneously. To accomplish these tasks, agents must either take pictures of the prescribed region (inspect crops, survey for pests, and estimate water reserves), or simply be present in the prescribed region for the prescribed time (harvest crops and deter pests). The task definitions for this demonstration are shown in Table 5 and the specification for this demonstration is shown in Equation 15.

The demonstration consists of three Crazyflie 2.0 nano-UAV (CF), a large 220mm custom UAV (LD), and six iRobot Create2 ground robots (GR). Each of these platforms communicates using the Robot Operating System (ROS) kinetic architecture and carries a camera that can be oriented either forward (F) or downward (D). The set of capabilities for this demonstration is $Cap_{exp} = \{CFD, LDF, GRF, GRD\}$. The GRD capability is performed using a downward RGBC color sensor (considered as a single pixel camera), and coexists on four of the iRobot Create2 platforms with a forward facing camera.

The robots operate simultaneously in an indoor $6m \times 9m$ arena, where robot positions are tracked with an Optitrack motion capture system. The robots are divided among three altitudes to augment their sensor modalities and to separate them aerodynamically. The Crazyflie nano-UAVs fly at a higher altitude than the custom UAV to avoid its considerable down-wash. The experimental space is partitioned into 12 regions $Q = \{q_0, \ldots, q_{11}\}$ as shown in Figure 4. The robots initial positions are shown in Figure 4(a).

$$\begin{aligned}\phi_{exp} = {} & \Diamond_{[0,20)}(T_{blue}) \\ & \wedge \Diamond_{[0,25)}(T_{H1} \vee T_{H2}) \\ & \wedge \Box_{[0,30)} \Diamond_{[0,15)}(T_{I1}) \\ & \wedge \Diamond_{[0,30)}(T_{I2}) \\ & \wedge \Diamond_{[0,30)}(T_{red1} \mathcal{U}_{[0,15)}(T_{red2} \vee T_{red3}))\end{aligned} \tag{15}$$

Name	**Duration** (d)	**Regions** ($L^{-1}(\pi)$)	**Counting Proposition** (c)	**Plain English**
T_{red1}	3	q_3	$\{(CFD, 2)\}$	Survey for pests
T_{red2}	2	q_3	$\{(GRF, 1)\}$	Deter pests
T_{red3}	3	q_3	$\{(LDF, 1)\}$	Deter pests
T_{H1}	5	q_5	$\{(GRF, 3)\}$	Harvest crops
T_{H2}	5	q_6	$\{(GRF, 3)\}$	Harvest crops
T_{blue}	3	q_{10}	$\{(GRD, 1), (CFD, 1)\}$	Estimate water reserves
T_{I1}	3	q_8	$\{(GRF, 3)\}$	Inspect crops
T_{I2}	2	$\{q_0, q_2\}$	$\{(GRF, 1), (CFD, 1)\}$	Inspect crops

Table 5: List of tasks used in Equation 15.

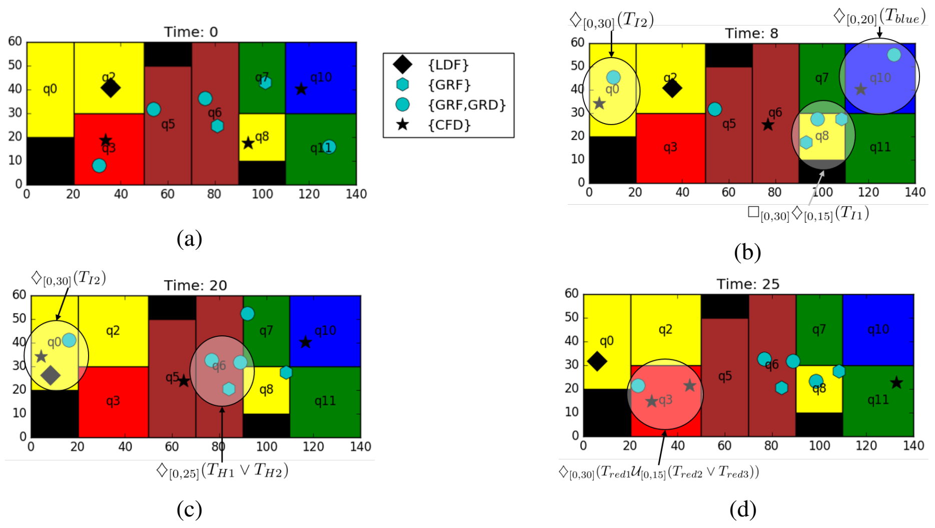

Time: 0
{LDF}
{GRF}
{GRF,GRD}
{CFD}
(a)
Time: 8
$\Diamond_{[0,30]}(T_{I2})$
$\Diamond_{[0,20]}(T_{blue})$
$\Box_{[0,30]}\Diamond_{[0,15]}(T_{I1})$
(b)
Time: 20
$\Diamond_{[0,30]}(T_{I2})$
$\Diamond_{[0,25]}(T_{H1} \vee T_{H2})$
(c)
Time: 25
$\Diamond_{[0,30]}(T_{red1}\mathcal{U}_{[0,15]}(T_{red2} \vee T_{red3}))$
(d)
q0 q2 q3 q5 q6 q7 q8 q10 q11

Fig. 4: Snapshots of solution to Equation 15 in the demonstration environment. Times 8, 20, and 25 show where specific components of the specification are satisfied and these areas are highlighted in (b)-(d).

In plain English, ϕ_{exp} specifies that "Within 20 time steps initiate estimation of the water reservoir level in q_{10}. Within 25 time steps initiate harvesting in either q_5 or q_6. Within at least every 15 time steps for the first 30 time steps initiate an inspection of crops in region q_8. Within 30 time steps initiate inspection in either region q_0 or q_2. Within 30 time steps initiate surveying for pests in the red region and then within 15 time steps, either a large drone or ground robot must enter the red region to deter pests."

Each black region in Figure 4 corresponds to an obstacle that no robot may traverse. These regions and the transitions to/from them are discarded in our construction of the environment transition system. Snapshots of the solution to Equation 15 at four key time steps (where components of the specification are satisfied) are shown in Figure 4 for the ten robots in the demonstration. This solution is also shown in the accompanying video.

The motion planning for the team of robots is performed with a sequential, timed, multi-agent rapidly exploring random trees (RRT) algorithm (more on RRT can be found in [5]) where an agent plans its entire trajectory and is then considered an obstacle (at specific time instants) to future planning agents. This planning is also stratified based on operation height (i.e. drones do not consider ground robots as obstacles). This allows the trajectories to have both flexibility in avoiding obstacles and other agents, as well as better fitting the execution time to the size of the experimental space.

ScRATCHS developed a motion plan to satisfy ϕ_{exp} in a total of 10.25s (9.16s to solve the MILP and 1.09s to generate the motion plan). This time scale is relevant to a multitude of real world planning and coordination tasks and is substantially faster than most teams of humans can solve moderately-sized PAC problems by hand [15].

8 Conclusions and Future Work

In this paper, we have developed a framework for scalable and robust deployment of teams of heterogeneous agents. ScRATCHS is able to build plans based on rich, TL specifications involving tasks that require the participation of multiple capabilities, e.g., sensing modalities, distributed across the team of agents. The planning problem is encoded as a large MILP, which can be efficiently solved using modern commercial off-the-shelf solvers. We validated this method using computational experiments, which showed the potential scalability of the method, and via a hardware demonstration, which illustrated the potential implementability and applicability of our approach.

In the future, in addition to generating plans that are robust to attrition, we will equip the team with a monitor to keep track of progress and reactive synthesis techniques to alter the plan on-the-fly when attrition or delays occur. Finally, we will also investigate parallelization by breaking the team into sub-teams and equipping each sub-team with its own monitor and plan, thus allowing sub-teams to operate quasi-independently and reduce the amount of communication required to execute and alter the plan.

Additional work is required to connect ScRATCHS to other algorithms and routines that address its limitations. ScRATCHS assumes a fixed and known characterization of the environment. This limitation could be addressed by assigning some agents to have an exploration role and by equipping the team with estimation/learning capabilities. ScRATCHS assumes that all of its high-level plans can be enacted as motion plans. This limitation can be addressed by adding in iteratively re-solving the planning problem when unsatisfiable motion planning problems are encountered and by using finite abstractions based on agent dynamics as the basis for the environment description.

Bibliography

[1] C. Baier and J.-P. Katoen. *Principles of model checking*. MIT press, 2008.

[2] C. Belta, B. Yordanov, and E. A. Gol. *Formal methods for discrete-time dynamical systems*, volume 89. Springer, 2017.

[3] Y. Chen, X. C. Ding, A. Stefanescu, and C. Belta. Formal approach to the deployment of distributed robotic teams. *IEEE Transactions on Robotics*, 28(1):158–171, 2012.

[4] H.-L. Choi, L. Brunet, and J. P. How. Consensus-based decentralized auctions for robust task allocation. *IEEE transactions on robotics*, 25(4):912–926, 2009.

[5] D. Connell and H. M. La. Extended rapidly exploring random treebased dynamic path planning and replanning for mobile robots. *International Journal of Advanced Robotic Systems*, 15(3), 2018.

[6] J. Cortes and M. Egerstedt. Coordinated control of multi-robot systems: A survey. *SICE Journal of Control, Measurement, and System Integration*, 10(6):495–503, 2017.

[7] Y. Diaz-Mercado, A. Jones, C. Belta, and M. Egerstedt. Correct-by-construction control synthesis for multi-robot mixing. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 221–226, Dec 2015.

[8] A. Donzé and O. Maler. Robust satisfaction of temporal logic over real-valued signals. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 92–106. Springer, 2010.

[9] M. Guo and D. V. Dimarogonas. Multi-agent plan reconfiguration under local ltl specifications. *The International Journal of Robotics Research*, 34(2):218–235, 2015.

[10] M. Guo and D. V. Dimarogonas. Task and motion coordination for heterogeneous multiagent systems with loosely coupled local tasks. *IEEE Transactions on Automation Science and Engineering*, 14(2):797–808, April 2017.

[11] I. Haghighi, S. Sadraddini, and C. Belta. Robotic swarm control from spatio-temporal specifications. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 5708–5713. IEEE, 2016.

[12] S. Karaman and E. Frazzoli. Vehicle routing problem with metric temporal logic specifications. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 3953–3958. IEEE, 2008.

[13] J. Karlsson, C.-I. Vasile, J. Tumova, S. Karaman, and D. Rus. Multi-vehicle motion planning for social optimal mobility-on-demand. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7298–7305. IEEE, 2018.

[14] J. Kiener and O. Von Stryk. Cooperation of heterogeneous, autonomous robots: A case study of humanoid and wheeled robots. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 959–964. IEEE, 2007.

[15] J. Kim, C. Banks, and J. Shah. Collaborative planning with encoding of users' high-level strategies. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[16] G. Korsah, A. Stentz, and M. Dias. A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research*, 32(12):1495–1512, 2013.

[17] K. Leahy, A. Jones, M. Schwager, and C. Belta. Distributed information gathering policies under temporal logic constraints. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 6803–6808, Dec 2015.

[18] P. Linz. *An Introduction to Formal Languages and Automata*. Jones & Bartlett Learning, 2006.

[19] O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 152–166. Springer, 2004.

[20] H. D. Mittelmann. Selected benchmark results. In *INFORMS Annual Meeting*, 2016. URL `http://plato.asu.edu/talks/informs2016_bench.pdf`.

[21] Y. V. Pant, H. Abbas, R. A. Quaye, and R. Mangharam. Fly-by-logic: Control of multi-drone fleets with temporal logic objectives. In *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, 2018.

[22] A. Prorok, M. A. Hsieh, and V. Kumar. Fast redistribution of a swarm of heterogeneous robots. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, pages 249–255. ICST (Institute for Computer Sciences, Social-Informatics and , 2016.

[23] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia. Model predictive control with signal temporal logic specifications. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 81–87. IEEE, 2014.

[24] S. Sadraddini. Formal methods for resilient control, 2018. URL `https://open.bu.edu/handle/2144/27455`.

[25] S. Sadraddini and C. Belta. Robust temporal logic model predictive control. In *Communication, Control, and Computing (Allerton), 2015 53rd Annual Allerton Conference on*, pages 772–779. IEEE, 2015.

[26] Y. E. Sahin, P. Nilsson, and N. Ozay. Multirobot coordination with counting temporal logics. *arXiv preprint arXiv:1810.13087*, 2018.

[27] P. Schillinger, M. Bürger, and D. V. Dimarogonas. Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems. *The international journal of robotics research*, 37(7):818–838, 2018.

[28] B. Schlotfeldt, D. Thakur, N. Atanasov, V. Kumar, and G. J. Pappas. Anytime planning for decentralized multirobot active information gathering. *IEEE Robotics and Automation Letters*, 3(2):1025–1032, April 2018.

[29] R. Simmons, D. Apfelbaum, D. Fox, R. P. Goldman, K. Z. Haigh, D. J. Musliner, M. Pelican, and S. Thrun. Coordinated deployment of multiple, heterogeneous robots. Technical report, Carnegie-Mellon University, Pittsburgh, PA, School of Computer Science, 2000.

[30] V. Tjeng, K. Xiao, and R. Tedrake. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.

[31] C.-I. Vasile, D. Aksaray, and C. Belta. Time window temporal logic. *Theoretical Computer Science*, 691:27–54, 2017.