
Proceedings Paper:
Bharadwaj, S., Vinod, A.P., Dimitrova, R. et al. (1 more author) (2020) Near-optimal reactive synthesis incorporating runtime information. In: Proceedings of 2020 IEEE International Conference on Robotics and Automation (ICRA). 2020 IEEE International Conference on Robotics and Automation (ICRA), 31 May - 31 Aug 2020, Paris, France. IEEE , pp. 10342-10348. ISBN 9781728173962

https://doi.org/10.1109/ICRA40945.2020.9196581

Near-Optimal Reactive Synthesis Incorporating Runtime Information

Suda Bharadwaj[1], Abraham P. Vinod[1], Rayna Dimitrova[2], Ufuk Topcu[1]

[1]The University of Texas at Austin
[2]The University of Sheffield, UK

***Abstract*— We consider the problem of optimal reactive synthesis — compute a strategy that satisfies a mission specification in a dynamic environment, and optimizes a performance metric. We incorporate task-critical information, that is only available at runtime, into the strategy synthesis in order to improve performance. Existing approaches to utilising such time-varying information require online re-synthesis, which is not computationally feasible in real-time applications. In this paper, we pre-synthesize a set of strategies corresponding to *candidate instantiations* (pre-specified representative information scenarios). We then propose a novel switching mechanism to dynamically switch between the strategies at runtime while guaranteeing all safety and liveness goals are met. We also characterize bounds on the performance suboptimality. We demonstrate our approach on two examples — robotic motion planning where the likelihood of the position of the robot's goal is updated in real-time, and an air traffic management problem for urban air mobility.**

I. Introduction

As autonomous systems become more widely used in society, we require provable guarantees of performance and safety in complex missions [1], [2]. In many applications, it is not enough for an autonomous agent to satisfy its mission objective, but it is often required that it also optimizes some performance metric. Due to limits on communication, sensing, or computational power, the autonomous agent may have access to information that may be available only at the time of execution. Traditional approaches either ignore this information or can only make use of it at the cost of heavy computation or high memory requirements [3], [4]. We propose a correct-by-construction switching strategy that utilizes such information at runtime for improved performance while guaranteeing the satisfaction of high-level mission specifications, and also alleviates the shortcomings of the existing methods to enable real-time deployment.

For example, consider a motion-planning problem for a service robot as shown in Figure 1. A high-level mission for the robot is to meet the human infinitely often, while ensuring that it always has sufficient battery power (rechargeable by returning to a charging station). Given the probability of the human's location based on past observations (runtime information), the proposed approach finds the human in a shorter period of time (compared to strategies that ignore this probability information), while satisfying the safety specification.

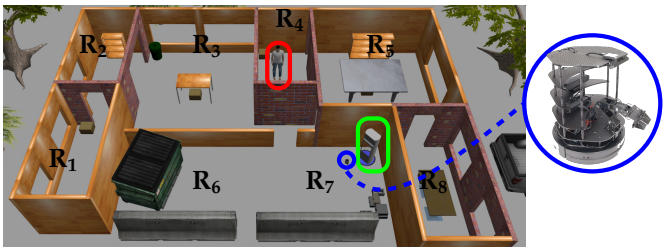


Fig. 1: Path planning environment for a turtlebot (in blue) to infinitely often service a human (in red). The robot can recharge at a charging station (in green).

For another, more complex example, consider the coordination of landing a collection of autonomous air vehicles in urban air mobility (UAM) operations [5], [6]. We seek to optimize performance (reduce maximum delay in aircraft landing) while ensuring safe takeoff and landing operations [7]. The on-demand nature of UAM means knowledge of air traffic demands is not available at design time. This necessitates a method that can use traffic information gained at runtime to adjust behavior for improved performance and safety. Previous approaches implement runtime safety enforcement [8]–[11], but cannot handle more general specifications.

The two scenarios described previously illustrate a *reactive* planning problem. The autonomous system has to react to an uncontrolled environment, and guarantee correctness with respect to a given mission specification for all possible behaviours of the environment for all time points in the future. The standard approach to solve such a planning problem is to use *reactive synthesis* [12], [13]. In particular, there is a large body of work focused on synthesis for a fragment of linear temporal logic (LTL) called GR(1) [14]–[17]. The solution time is polynomial in the state space of the game structure, and exponential in the number of atomic propositions. Therefore, this approach typically relies on offline planning, that prevents easy incorporation of runtime information. In problems with a continuous state space, a discrete abstraction is used that preserves correctness. Such controllers however, can be significantly subopti-

This material is based upon work supported by the Office of Naval Research (N00014-18-1-2829), National Aeronautics and Space Administration (80NSSC19K0209), and U.S. Army Research Laboratory (ACC-APG-RTP W911NF).

mal with respect to the performance objective [4].

We consider the *runtime information* as a (possibly continuous) parameter associated with the environment. The work in [4] allows for near-optimal behaviour on continuous executions, however the authors focus on a specialized cost metric. Additionally, their method relies on online re-synthesis, which is not feasible for real-time deployment. This work was later extended in [3] to account for delay costs arising from a potentially adversarial environment. However, it relies on the discretization of the continuous parameter space, which fails to scale with the number of atomic propositions in the synthesis problem.

Our approach incorporates parametrized runtime information by switching between pre-computed strategies. First, for a given set of *candidate instantiations* of the parameter we synthesize offline optimal strategies that satisfy all task specifications. Next, we obtain bounds on the suboptimality incurred by the use of these policies at *all* other parameter values. This computation does not require discretization of the parameter space. At runtime, we dynamically switch strategies based on the these suboptimality bounds, thereby incorporating runtime information into the offline synthesis of correct-by construction policies. To this end, we derive a switching function that guarantees the resulting execution is provably correct, and near-optimal.

The main contributions of this paper are: 1) a novel switching protocol between pre-synthesized correct strategies that improves performance, 2) correctness of the switching protocol with respect to the mission specification, and 3) characterization of the suboptimality bounds on performance. We demonstrate the proposed approach on a motion planning problem for a service robot, and a traffic scheduling problem for UAM.

II. Preliminaries

1) ***Basic notation****:* We consider reactive systems with a finite set E of Boolean *inputs*, controlled by the *E*nvironment, and a finite set A of Boolean *outputs*, controlled by the *A*gent. Together, they define the system's input alphabet $\Sigma_E = 2^E$ and the output alphabet $\Sigma_A = 2^A$. We define $\Sigma = \Sigma_E \times \Sigma_A$.

2) ***Game structures****:* We model the interaction between the agent and its environment as a two-player game. Formally, the game is played on a *game structure* which is a tuple $\mathcal{G} = (G, g_0, \Sigma, \delta)$, where:

- G is a finite set of states and $g_0 \in G$ the initial state;
- $\Sigma = \Sigma_E \times \Sigma_A$ is the alphabet of actions available to the environment and the agent respectively;
- $\delta : G \times \Sigma \to G$ is a complete transition function, that maps each state, input (environment action) and output (agent action) to a successor state.

At every state $g \in G$ (starting with g_0), the environment chooses an input $\sigma_E \in \Sigma_E$, and then the agent chooses some output $\sigma_A \in \Sigma_A$. These choices define the next state $g' = \delta(g, (\sigma_E, \sigma_A))$, and the process then continues from g'. This order of moves ensures that at each step the agent's action reacts to the current action of the environment. The resulting (infinite) sequence $\overline{\pi} = (g_0, \sigma_{E,0}, \sigma_{A,0}, g_1)(g_1, \sigma_{E,1}, \sigma_{A,1}, g_2)\dots$ is called a *play*, where g_0 is the initial state, and for every $i \geq 0$ we have that $g_{i+1} = \delta(g_i, \sigma_{E,i}, \sigma_{A,i})$. For a play $\overline{\pi}$ and an integer $m \in \mathbb{N}$ we define $\overline{\pi}[0, m] = (g_0, \sigma_{E,0}, \sigma_{A,0}, g_1)\dots(g_{m-1}, \sigma_{E,m-1}, \sigma_{A,m-1}, g_m)$ to be the prefix consisting of the first m elements of $\overline{\pi}$. For $m = 0$, $\overline{\pi}$ is the empty word. The set $\mathit{Plays}(\mathcal{G})$ is the set of all plays in the game $\mathcal{G}$, and the set $\mathit{Prefs}(\mathcal{G})$ is the set of all finite prefixes of the plays in the $\mathit{Plays}(\mathcal{G})$. Plays starting at a given arbitrary (not necessarily initial) state $g \in G$ of $\mathcal{G}$ are defined analogously. We denote with $\mathit{Plays}(\mathcal{G}, g)$ the set of plays starting at a state g.

3) ***Winning conditions****:* The *winning condition* for the agent in a game $\mathcal{G}$ is given as a set of plays $\varphi \subseteq \mathit{Plays}(\mathcal{G})$ that specifies the set of plays that result in the agent winning the game. We consider games in which the agent has a *Generalized Reactivity 1* (GR(1)) winning condition, which are common in a variety of practical applications. In the following, we make use of the linear temporal logic (LTL) operators *always* $\square$ and *eventually* $\diamondsuit$. For full details on LTL syntax and semantics, we refer the reader to [18].

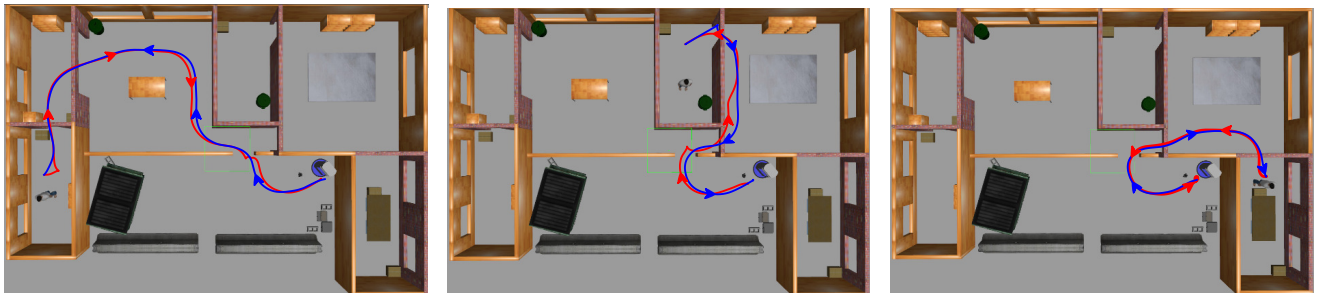
A GR(1) winning condition is defined by sets of states $S_E, S_A \subseteq G$, $E_i \subseteq G$ for $i = 1, \dots, m$ and $F_j \subseteq G$ for $j = 1, \dots, n$, and consists of all plays $\overline{\pi}$ such that if $\overline{\pi} \in \square S_E \cap \square \diamondsuit E_i$ for all $i = 1, \dots, m$, then $\overline{\pi} \in \square S_A \cap \square \diamondsuit F_j$ for all $j = 1, \dots, n$. Intuitively, for a play $\overline{\pi}$ to be winning, whenever the environment satisfies the assumptions $\square S_E, \square \diamondsuit E_1, \dots, \square \diamondsuit E_m$, then the agent must satisfy all the guarantees $\square S_A, \square \diamondsuit F_j, \dots, \square \diamondsuit F_n$. By abuse of logical operators, we abbreviate GR(1) conditions as

$$\varphi = \left(\square S_E \wedge \bigwedge_{i=1}^{m} \square \diamondsuit E_i\right) \rightarrow \left(\square S_A \wedge \bigwedge_{i=1}^{n} \square \diamondsuit F_i\right).$$

4) ***Strategies****:* A *strategy for the agent* is a function $\rho_A : \mathit{Prefs}(\mathcal{G}) \times \Sigma_E \to \Sigma_A$ which maps a prefix (the history of the play so far) and an action of the environment to an action of the agent. A *strategy for the environment* is a function $\rho_E : \mathit{Prefs}(\mathcal{G}) \to \Sigma_E$ that maps the prefix of the play so far to an action of the environment. We denote the sets of all strategies for the agent and for the environment by $\mathcal{M}_A$ and $\mathcal{M}_E$ respectively.

Every pair of strategies $\rho_A \in \mathcal{M}_A$ for the agent and $\rho_E \in \mathcal{M}_E$ for the environment define a play, denoted by $\Pi(\rho_A, \rho_\epsilon)$. More precisely, $\Pi(\rho_A, \rho_\epsilon) = \overline{\pi} = (g_0, \sigma_{E,0}, \sigma_{A,0}, g_1)(g_1, \sigma_{E,1}, \sigma_{A,1}, g_2)\dots \in \mathit{Plays}(\mathcal{G})$ where for every $i \geq 0$, $\sigma_{E,i} = \rho_E(\overline{\pi}[0, i])$ and $\sigma_{A,i} = \rho_A(\overline{\pi}[0, i], \sigma_{E,i})$. Similarly, we define the set of plays starting at a state g that are consistent with ρ_A, denoted $\mathit{Plays}(\mathcal{G}, \rho_A, g)$.

Given a game structure $\mathcal{G}$ and a winning condition φ for the agent, we seek to synthesize a strategy $\rho_A \in \mathcal{M}_A$ for the agent such that for every strategy $\rho_E \in \mathcal{M}_E$ for the environment it holds that $\Pi(\rho_E, \rho_A) \in \varphi$, i.e., all



(a) $\overline{p}_1 = [1, 0, 0]$ (b) $\overline{p}_2 = [0, 1, 0]$ (c) $\overline{p}_3 = [0, 0, 1]$

Fig. 2: Continuous trajectories resulting from executing policies corresponding to the solution of (2) for three different instantiations of runtime information vector $\overline{p}$: $\overline{p}_1 = [1, 0, 0]$, $\overline{p}_2 = [0, 1, 0]$, and $\overline{p}_3 = [0, 0, 1]$.

resulting plays satisfy φ. In such cases we say that ρ_A *satisfies* φ, denoted $\rho_A \models \varphi$.

III. Problem formulation

We represent *runtime information* as n-dimensional real vectors, for a given $n \in \mathbb{N}$. We denote the set of all possible vector values for the runtime information by $\mathcal{P} \subseteq \mathbb{R}^n$. We score the performance of each play in the game using runtime information in $\mathcal{P}$ via a performance metric, $J : Plays(\mathcal{G}) \times \mathcal{P} \to \mathbb{R}$.

Assumption 1. *For every $\overline{p} \in \mathcal{P}$ and every strategy $\rho_A \in \mathcal{M}_A$ such that $\rho_A \models \varphi$, there exists a strategy $\rho_E \in \mathcal{M}_E$ such that $J(\Pi(\rho_A, \rho_E), \overline{p}) \geq J(\Pi(\rho_A, \rho'_E), \overline{p})$ for every $\rho'_E \in \mathcal{M}_E$.*

Assumption 1 ensures a well-defined cost function using the metric J. We can thus define

$$C(\rho_A, \overline{p}) = max_{\rho_E \in \mathcal{M}_E} J(\Pi(\rho_A, \rho_E), \overline{p}) \tag{1}$$

as the cost function, with $C : \Sigma_A \times \mathcal{P} \to \mathbb{R}$. Given the runtime information $\overline{p} \in \mathcal{P}$, a strategy $\rho_A \in \mathcal{M}_A$ for the agent that satisfies φ is *optimal for* $\overline{p}$ if and only if it is a solution to the following optimization problem.

$$\begin{array}{ll} \underset{\rho_A \in \mathcal{M}_A}{\text{minimize}} & C(\rho_A, \overline{p}) \\ \text{subject to} & \rho_A \models \varphi \end{array} \tag{2}$$

Let $C^* : \mathcal{P} \to \mathbb{R}$ denote the optimal value of (2).

Example. *Consider Figure 1, where the robot has to infinitely often meet the human. Assume that the human can only be in rooms R_1, R_4, R_8. Let $q_1, q_2, q_3 \in [0, 1]$ be the probabilities of the human being in room R_1, R_4 and R_8 respectively. The runtime information is $\overline{p} = [q_1\ q_2\ q_3]^\top \in \mathcal{P} \subseteq \mathbb{R}^3$, where the set $\mathcal{P}$ is the probability simplex. The cost function is,*

$$C(\rho_A, \overline{p}) = \mathbb{E}[\textit{time to find human}] = \sum_{i=1}^{N} T_i(\rho_A)\, q_i \tag{3}$$

where T_i is the time taken to reach room i under the robot strategy ρ_A. Figure 2 shows the resulting continuous trajectories from executing policies when the information vector tells the robot the exact room occupied by the human.

Given a set of representative strategies associated with instances of the runtime information, the task at runtime then becomes one of choosing a strategy depending on the current value of $\overline{p}$. As we have a finite set of strategies to choose from, the resulting behaviour of the agent will be *approximately optimal*. Thus, we consider the problem of synthesizing an *approximately optimal switching function* that also guarantees φ.

Definition 1. *Given a game structure $\mathcal{G}$ and a set of strategies $\{\rho_{A_i} \in \mathcal{M}_A\}_{i=1}^N$ for the agent, a* switching function *is a function $\tau : Prefs(\mathcal{G}) \times \mathcal{P}^* \to \{1, \ldots, N\}$, which maps a play prefix and the sequence of values of $\overline{p}$ seen so far, to an index of a strategy in the given set.*

The set of plays resulting from applying the switching function τ to $\{\rho_{A_i} \in \mathcal{M}_A\}_{i=1}^N$ is defined as the set of plays $Plays(\{\rho_{A_i} \in \mathcal{M}_A\}_{i=1}^N, \tau)$ such that $\overline{\pi} = (g_0, \sigma_{E,0}, \sigma_{A,0}, g_1)(g_1, \sigma_{E,1}, \sigma_{A,1}, g_2) \ldots \in Plays(\{\rho_{A_i} \in \mathcal{M}_A\}_{i=1}^N, \tau)$ if and only if there exists $\overline{\gamma} \in \mathcal{P}^\omega$ such that for every $i \geq 0$, it holds that $\sigma_{A,i} = \rho_{A_{\tau(\overline{\pi}[0,i], \overline{\gamma}[0,i])}}(\pi[0, i], \sigma_{E,i})$.

Informally, we want to be able to *switch* between precomputed strategies based on values of the runtime information. In order to not violate the specification, switching needs to take into account the prefix of the play. We formalize this task below.

Problem 1. *We are given a game $(\mathcal{G}, \varphi)$, a set $\{\overline{p}_i \in \mathcal{P}\}_{i=1}^N$ of representative values of the runtime information, and strategies $\{\rho^*_{A_i} \in \mathcal{M}_A\}_{i=1}^N$ such that $\rho^*_{A_i}$ solves (2) for $\overline{p}_i$.*

Given $\epsilon > 0$, compute a collection $\{\mathcal{S}_i\}_{i=1}^N$ of subsets of $\mathbb{R}^n$ such that $\overline{p}_i \in \mathcal{S}_i$, and a switching function $\tau : Prefs(\mathcal{G}) \times \mathcal{P}^+ \to \{1, \ldots, N\}$ that satisfies the following conditions.

- *For all $i = 1, \ldots, N$ and all $\overline{p} \in \mathcal{S}_i \cap \mathcal{P}$ it holds that*

$$C(\rho^*_{A_i}, \overline{p}) - \epsilon \leq C^*(\overline{p}) \leq C(\rho^*_{A_i}, \overline{p}). \tag{4}$$

- *$\overline{\pi} \in \varphi$ for every play $\overline{\pi} \in Plays(\{\rho_{A_i} \in \mathcal{M}_A\}_{i=1}^N, \tau)$.*

IV. Synthesis of correct-by-construction strategies with near-optimality guarantees

We address Problem 1 by constructing a switching function that guarantees that the resulting plays satisfy the task specification φ. We also provide suboptimality

bounds for the performance when using the proposed method. We utilize existing synthesis techniques [3] to synthesize for each $i \in \{1, \ldots, N\}$ a strategy that is *correct*, i.e., satisfies φ, and optimal for the specific $\overline{p}_i$.

A. Suboptimality bounds for unknown runtime information

Let $\{\overline{p}_i \in \mathcal{P}\}_{i=1}^N$ be the given candidate instantiations of the runtime information, and let $\{\rho^*_{Ai} \in \mathcal{M}_A\}_{i=1}^N$ be the corresponding optimal strategies. That is, for each i, $\rho^*_{A_i}$ is an optimal solution of (2) for $\overline{p}_i$. Under Assumption 2 below, we can compute a collection of polytopes $\{\mathcal{S}_i\}_{i=1}^N$ such that ρ^*_{Ai} is ϵ-optimal, whenever $\overline{p} \in \mathcal{S}_i$.

Assumption 2. *The cost function $C : \mathcal{M}_A \times \mathcal{P} \to \mathbb{R}$ is such that for all $\rho_A \in \mathcal{M}_A$ and $\overline{p} = [q_1 \; q_2 \; \ldots \; q_n]^\top \in \mathcal{P} \subseteq \mathbb{R}^n$:*

$$C(\rho_A, \overline{p}) = \sum_{i=1}^{n} C(\rho_A, \overline{e}_i)\, q_i, \tag{5}$$

where $\overline{e}_i$ is the vector that has 1 at position i and 0 elsewhere.

We address Problem 1 by defining a collection of polytopes $\{\mathcal{S}_i\}_{i=1}^N$, where $\mathcal{S}_i = \{\overline{p} \in \mathbb{R}^n \mid H_i \overline{p} \leq \overline{b}_i\}$ and

$$H_i = \begin{bmatrix} C(\rho_{A_i^*}, \overline{e}_1) - C(\rho_{A_1^*}, \overline{e}_1) & \cdots & C(\rho_{A_i^*}, \overline{e}_n) - C(\rho_{A_1^*}, \overline{e}_n) \\ C(\rho_{A_i^*}, \overline{e}_1) - C(\rho_{A_2^*}, \overline{e}_1) & \cdots & C(\rho_{A_i^*}, \overline{e}_n) - C(\rho_{A_2^*}, \overline{e}_n) \\ \vdots & \vdots & \vdots \\ C(\rho_{A_i^*}, \overline{e}_1) - C(\rho_{A_N^*}, \overline{e}_1) & \cdots & C(\rho_{A_i^*}, \overline{e}_n) - C(\rho_{A_N^*}, \overline{e}_n) \\ C(\rho_{A_i^*}, \overline{e}_1) - C^*(\overline{e}_1) & \cdots & C(\rho_{A_i^*}, \overline{e}_n) - C^*(\overline{e}_n) \end{bmatrix} \tag{6}$$

$$\overline{b}_i = [0 \; 0 \; \ldots \; 0 \; \epsilon]^\top \in \mathbb{R}^{N+1} \tag{7}$$

with $H_i \in \mathbb{R}^{(N+1) \times n}$. By (2) and Assumption 2, we have the following upper bound on $C^*(\overline{p})$ for any runtime information vector $\overline{p} = [q_1 \; q_2 \; \ldots \; q_n]^\top \in \mathcal{P}$,

$$C^*(\overline{p}) \leq C(\rho^*_{A_i}, \overline{p}) = \sum_{j=1}^{n} C(\rho^*_{A_i}, \overline{e}_j) q_j, \quad \forall i \in \{1, \ldots, N\}. \tag{8}$$

Theorem 1 below ensures that using ρ^*_{Ai} for vectors $\overline{p} \in \mathcal{S}_i$ results in ϵ-optimal performance, provided the polytopes $\mathcal{S}_i$ are non-empty. In other words, the difference between $C^*(\overline{p})$, the optimal performance for a runtime information $\overline{p}$, and $C(\rho^*_{A_i}, \overline{p})$, the attained performance due to choice of strategy $\rho^*_{A_i}$, can not be larger than ϵ, whenever $\overline{p} \in \mathcal{S}_i$. To complete the discussion, we provide a sufficient condition for non-empty polytopes $\mathcal{S}_i$ in Proposition 1.

Theorem 1. *Let the polytopes $\mathcal{S}_i$ be non-empty. Given runtime information $\overline{p} \in \mathbb{R}^n$, if $\overline{p} \in \mathcal{S}_i$ for some $i \in \{1, \ldots, N\}$, then*

$$C(\rho^*_{A_i}, \overline{p}) - \epsilon \leq C^*(\overline{p}) \leq C(\rho^*_{A_i}, \overline{p}).$$

Proof. The upper bound on $C^*(\overline{p})$ follows from (8). Consider the collection of polytopes $\mathcal{T}_i$ constructed using the first N hyperplanes in (6) and (7). For every $\overline{p} \in \mathcal{T}_i$,

$$C(\rho^*_{A_i}, \overline{p}) \leq C(\rho^*_{A_k}, \overline{p}), \; \forall k \in \{1, \ldots, N\} \setminus \{i\}. \tag{9}$$

In other words, among the N strategies $\rho^*_{A_{(\cdot)}}$, $\rho^*_{A_i}$ provides the tightest upper bound on $C^*(\overline{p})$ due to (8).

We next prove the lower bound. The last hyperplane in (6) and (7) guarantees that $C(\rho^*_{A_i}, \overline{p}) - \overline{\ell}^\top \overline{p} \leq \epsilon$ for every $\overline{p} \in \mathcal{S}_i$, where $\overline{\ell}_j = C^*(\overline{e}_j)$. On adding and subtracting $C^*(\overline{p})$, we have $C(\rho^*_{A_i}, \overline{p}) - C^*(\overline{p}) + C^*(\overline{p}) - \overline{\ell}^\top \overline{p} \leq \epsilon$. Since $C(\cdot, \overline{e}_j) \geq \ell_j$ by definition of $\overline{\ell}$, we have $C^*(\overline{p}) - \overline{\ell}^\top \overline{p} \geq 0$ for every $\overline{p} \in \mathcal{S}_j$. Therefore, $C(\rho^*_{A_i}, \overline{p}) - C^*(\overline{p}) \leq \epsilon$. □

Proposition 1. *Let Assumption 2 hold. For every $i = 1, \ldots, N$ the polytope $\mathcal{S}_i$ is non-empty, provided that $\epsilon \geq \max_i \left\{ C(\rho^*_{A_i}, \overline{p}_i) - \overline{\ell}^\top \overline{p}_i \right\}$. Here, $\overline{\ell} = [C^*(\overline{e}_1) \; C^*(\overline{e}_2) \; \ldots \; C^*(\overline{e}_n)]^\top \in \mathbb{R}^n$.*

Proof. The polytopes $\mathcal{T}_i$ (defined in the proof of Theorem 1) are non-empty, since they contain $\overline{p}_i$ by the optimality of $\rho^*_{A_i}$ in (2). The last hyperplane in (6) and (7) is also satisfied by $\overline{p}_i$, thanks to the use of ϵ in $\overline{b}_i$. Thus, its intersection with $\mathcal{T}_i$, which yields the polytope $\mathcal{S}_i$, is non-empty. □

B. Switching function construction

In order to guarantee that the plays resulting from switching between the synthesized representative strategies satisfy the specification φ, the switching function needs to keep track the satisfaction of the agent's liveness guarantees $\square\diamondsuit F_i$ in φ. Since the cost function C captures the cost of achieving the liveness guarantees, when the specification is satisfied due to a violation of the environment assumptions, no switching would be necessary, as the cost would be 0.

To ensure that the switching between strategies does not prevent the agent from infinitely often visiting each of the sets F_i, the switching function will keep track of these visits, and only allow switching to a different strategy once all the sets F_i have been visited under the current strategy. Furthermore, the switch can only occur from a state from which the next strategy can guarantee the satisfaction of φ. Below we make this intuition precise by providing the construction of the switching function as a finite-state system.

Let $\{\rho^*_{Ai} \in \mathcal{M}_A\}_{i=1}^N$ be a set of strategies for the agent such that $\rho^*_{A_i} \models \varphi$ for each $i \in \{1, \ldots, N\}$, and let $\{\mathcal{S}_i\}_{i=1}^N$ be the polytopes computed as in Section IV-A.

For each ρ^*_{Ai}, let $W_i = \{g \in G \mid \mathit{Plays}(\mathcal{G}, \rho^*_{Ai}, g) \subseteq \varphi\}$ denote the set of states from which the specification can be enforced by following the strategy ρ^*_{Ai}.

We define a finite state transition system with states Q, initial state q_0, transition function θ and alphabet $(G \times \Sigma_E \times \Sigma_A \times G) \times \{\mathcal{S}\}_{i=1}^N$. The set of states is $Q = \{V \mid V \subseteq \{1, \ldots, n\}\} \times \{1, \ldots, N\}$, where n is the number of liveness guarantees in φ. States in Q track the guarantees that have been satisfied and contain the index of the currently chosen strategy. The initial state is $q_0 = (\{1, \ldots, n\}, 1)$. The transition function θ :

$Q \times ((G \times \Sigma_E \times \Sigma_A \times G) \times \{\mathcal{S}\}_{i=1}^N) \to Q$ is defined such that $\theta((V,i),((g,\sigma_E,\sigma_A,g'),\mathcal{S})) = (V',i')$, where

- if $V = \{1,\dots,n\}, \mathcal{S} = \mathcal{S}_j, g' \in W_j$, then, $V' = \emptyset, i' = j$,
- $V' = V \cup \{j \in \{1,\dots,n\} \mid g \in F_j\}$ and $i' = i$ otherwise.

That is, once all the sets F_j have been visited under the current strategy, we can switch to the i'-th strategy and reset the tracking set to $\emptyset$. Otherwise we record the indices of the visited sets and keep the strategy index the same. We can extend θ to words in the usual way.

We define the switching function such that $\tau(\varepsilon,\overline{p}_0) = \min(\{i \mid p_0 \in \mathcal{S}_i\} \cup \{N\})$, where ε is the empty word, and for every $\overline{\pi} = (g_0,\sigma_{E,0},\sigma_{A,0},g_1)\dots(g_k,\sigma_{E,k},\sigma_{A,k},g_{k+1})$ and every $\overline{\gamma} = \overline{p}_0\dots\overline{p}_{k+1}$ we let $\tau(\overline{\pi},\overline{\gamma}) = i$ where $\theta(((g_0,\sigma_{E,0},\sigma_{A,0},g_1),\mathcal{S}_{i_1})\dots((g_k,\sigma_{E,k},\sigma_{A,k},g_{k+1}),\mathcal{S}_{i_{k+1}})) = (V,i)$ for some V, where for all $j \geq 1$ we have $i_j = \min(\{i \mid \overline{p}_j \in \mathcal{S}_i \text{ and } g_j \in W_{i_j}\} \cup \{N\})$.

As the switching function τ only allows switching to a different strategy once all F_j have been visited under the current strategy, this means that if we switch strategy infinitely often then the liveness guarantee is satisfied. If, on the other hand, we stabilize at some strategy, φ is again guaranteed by the fact that this strategy satisfies the specification.

C. Discussion

The key advantage of our approach is that we avoid the re-synthesis of strategies for each new value of the runtime information, when the parameter set is covered by the collection of polytopes $\{\mathcal{S}_i\}_{i=1}^N$, $\mathcal{P} \subseteq \cup_{i=1}^N \mathcal{S}_i$. We also avoid discretization of the set $\mathcal{P}$. This enables on-board deployment of our approach with guaranteed ϵ-optimal performance. In contrast, traditional approaches rely either on re-synthesis or on discretization of the parameter space which requires either prohibitively high computational or memory costs [4], [19]. When $\mathcal{P} \not\subseteq \cup_{i=1}^N \mathcal{S}_i$, none of the synthesized strategies guarantees ϵ-optimal performance for the parameter values in $\mathcal{P} \setminus \cup_{i=1}^N \mathcal{S}_i$. In such cases, we can iteratively expand the candidate instantiations offline till the entire parameter space $\mathcal{P}$ is covered. Specifically, we add to the candidate instantiations randomly chosen parameter values in $\mathcal{P} \setminus \cup_{i=1}^N \mathcal{S}_i$. In future, we intend to investigate sufficient conditions under which such an expansion approach terminates in finite number of steps.

V. Experiments

All experiments we report on were performed on an Intel i5-5300U 2.30 GHz CPU with 8 GB of RAM. We used the tool `Slugs` [14] for the strategy synthesis.

A. Robot motion planning

We consider the example discussed in Section III (Figure 1). Formally, the specification is

$$\varphi = \square(h \in R_1 \cup R_4 \cup R_8) \to \Big(\square\diamondsuit(r = h) \wedge \square(\text{Energy} > 0)\Big),$$

Query point $\overline{p}$	Strategy	Cost		
		L. bound	U. bound	$C^*(\overline{p})$
$[0.1, 0.8, 0.1]$	ρ_{A_2}	7.19	10.4	9.5
$[0.0, 0.1, 0.9]$	ρ_{A_3}	6.39	9.6	7.9
$[0.6, 0.3, 0.1]$	–	–	–	11.1

TABLE I: Lower and upper bounds (Theorem 1) and the optimal value $C^*(\overline{p})$ for some runtime information vectors $\overline{p}$.

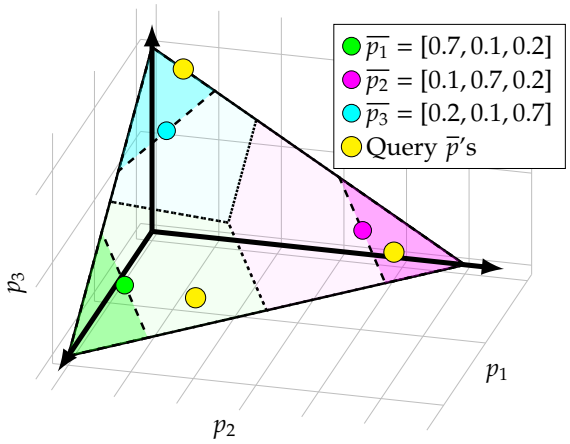


Fig. 3: State space partitioning of the runtime information vector $\overline{p}$ for $\epsilon = 3.2$. For runtime information vector belong to the darker shaded regions, we obtain ϵ-optimality by reusing a specific strategy $\rho_{A_{(\cdot)}}$, and avoid computationally expensive re-synthesis.

where h and r are variables modelling the human and robot positions respectively, and Energy is the robot's energy level. The cost function $C(\cdot)$ is given in (3). The runtime information $\overline{p}$ is the probability distribution over the human's possible locations - R_1, R_4, R_8. In our experiments, we used a Bayesian update to compute $\overline{p}$ using the current (and past) observations of the human's position.

We used three candidate instantiations of $\overline{p}$ (Figure 3). The robot only has enough charge to visit one of the three rooms and return to the charging station. The optimal robot strategy for each $\overline{p}_i$ corresponds to an *ordering* of which room to visit. Intuitively, the robot will visit rooms in decreasing order of likelihood of a human being there, by executing the continuous trajectories shown in Figure 2.

Figure 3 shows the partition of the space of $\mathcal{P}$ generated from the corresponding polytopes $\mathcal{S}_i$ for $\epsilon = 3.2$. The choice of ϵ is dictated by Proposition 1. The three candidate instantiations of $\overline{p}$ are represented by colored dots. The darker coloured regions are the polytopes $\mathcal{S}_i$, and they correspond to the regions of $\mathcal{P}$ in which the corresponding strategy is ϵ-optimal. Note that $\mathcal{P} \not\subseteq \cup_{i=1}^N \mathcal{S}_i$, and there are parameters in $\mathcal{P}$ were none of the three strategies are ϵ-optimal. The light shaded areas around each $\overline{p}_i$ corresponds to the portion

of the parameter space in which the correspondingly coloured strategy dominates the others, but is not ϵ-optimal.

Table I shows the proposed optimality bounds obtained from our approach (Theorem 1). On comparing with the lowest delay possible (computed via re-synthesis), we see that the computed bounds holds in the first two rows. The last row has $\overline{p} = [0.6, 0.3, 0.1]$ lying outside $\cup_{i=1}^{N} \mathcal{S}_i$ (outside of the dark shaded region in Figure 3), has no informative bounds. Here, ρ_{A_1} is the dominating strategy among $\rho_{A_{(\cdot)}}$, with $C(\rho_{A_1}, \overline{p}) = 13.6$.

A video of the simulation of the robot meeting the human (infinitely often) as the human moves in real-time can be found at `https://youtu.be/pn6afwf5INc`.

B. Urban air mobility traffic management

We now consider an automated air traffic management system for urban air mobility (UAM) operations. The controller is required to optimize the throughput of a multi-pad UAM port, along with bounding the delays experienced by vehicles and passengers. We synthesize a controller for a UAM hub, which consists of a grouping of multiple UAM vertiports. The hub has restrictions on the number of aircraft it is allowed to simultaneously land across all vertiports. Hence, a controller, if necessary, must make incoming air vehicles wait until it is able to safely allow them to land. In this example, we consider three vertiports — A (*red*), B (*yellow*), and C (*blue*) where an aircraft can request to land. Formally, the task specification is

$$\begin{aligned}\varphi = \square(&\text{Current Requests} < R) \rightarrow \\ &\square(\text{No. Landing Aircraft} < M) \wedge \\ &\square(\text{Land Request} \rightarrow \Diamond \text{Land Allowed}),\end{aligned}$$

where R is maximum number of simultaneous requests and M is the maximum number of aircraft allowed to land simultaneously. We model incoming aircraft as landing requests for vertiports drawn from a time-varying probability distribution. We model the performance metric as the *maximum delay*. The cost function is

$$C(\rho_A, \overline{p}) = \max_i(\text{delay}(V_i, \rho_A)) \cdot q_i \tag{10}$$

where $\overline{p} = [q_1, q_2, q_3, q_4]$, q_i is the probability of a request to land at vertiport hub V_i for $i = \{1, 2, 3\}$, and q_4 is the probability of no landing request, and $\text{delay}(V_i, \rho_A)$ is the processing delay at V_i under strategy ρ_A. We pre-compute strategies for three representative instantiations of the runtime information with each strategy taking 213 s to compute. Initially, we choose the true distribution to be one of the instantiations. At t = 150 minutes, the underlying probability distribution of landing requests changes such that the uninformed strategy performs poorly and the new probability value is not part of any of the representative instantiations. At t = 400 minutes, the probability distribution switches back to the initial probability distribution. The uninformed strategy is a fixed, runtime information-independent strategy that satisfies φ.

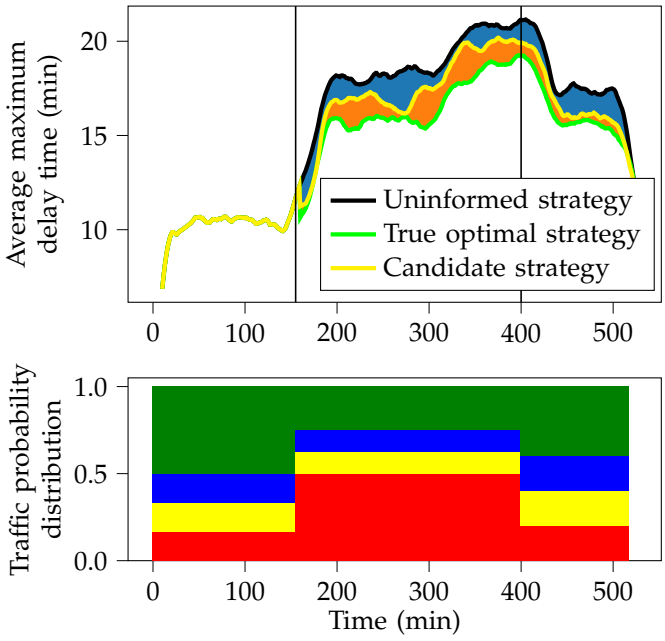


Fig. 4: Top: Average maximum delay times for $N = 100$ runs with landing requests drawn from a time varying probability distribution shown in bottom figure. Black vertical lines corresponds to a switch in strategy. Below: The probability of an aircraft requesting to land at hubs A (*red*), B (*yellow*), C (*blue*), or not land (*green*).

Figure 4 compares the proposed switching strategy against an optimal strategy that relies on re-synthesis (requires heavy computation) and an uninformed strategy (does not incorporate runtime information). Initially, during low traffic times, we see no deviation in delay times as expected. The proposed switching strategy provides a significant reduction of delay over time compared to an uninformed strategy. However, it is suboptimal to the strategy obtained via re-synthesis, which relies on heavy computation that rules out real-time execution. Since (10) does not satisfy Assumption 2, we do not have suboptimality bounds on the performance (Theorem 1). Empirically, the proposed approach shows an improvement in performance.

VI. Conclusion and future work

We present a method to integrate information about environment behaviour gained at runtime into reactive synthesis. Our technique provides significant performance gains over standard reactive synthesis without sacrificing any correctness or facing state space explosion. In future work we intend to investigate the use of counterexamples to generate more candidate instantiations of runtime information parameters in order to guarantee ϵ-optimality over the entire parameter set $\mathcal{P}$.

References

[1] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas, "Symbolic planning and control of robot motion [grand challenges of robotics]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 61–70, 2007.

[2] C. Finucane, G. Jing, and H. Kress-Gazit, "Ltlmop: Experimenting with language, temporal logic and robot control," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 1988–1993.

[3] G. Jing, R. Ehlers, and H. Kress-Gazit, "Shortcut through an evil door: Optimality of correct-by-construction controllers in adversarial environments," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November 3-7, 2013*, 2013, pp. 4796–4802.

[4] G. Jing and H. Kress-Gazit, "Improving the continuous execution of reactive ltl-based controllers," in *2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 6-10, 2013*, 2013, pp. 5439–5445.

[5] R. Goyal, "Urban air mobility (uam) market study," 2018.

[6] L. Gipson, "Nasa embraces urban air mobility, calls for market study," *NASA. November*, vol. 7, 2017.

[7] D. P. Thipphavong, R. Apaza, B. Barmore, V. Battiste, B. Burian, Q. Dao, M. Feary, S. Go, K. H. Goodrich, J. Homola, H. R. Idris, P. H. Kopardekar, J. B. Lachter, N. A. Neogi, H. K. Ng, R. M. Oseguera-Lohr, M. D. Patterson, and S. A. Verma, "Urban air mobility airspace integration concepts and considerations," in *2018 Aviation Technology, Integration, and Operations Conference*, 2018, p. 3676.

[8] S. Bharadwaj, S. Carr, N. Neogi, H. Poonawala, A. B. Chueca, and U. Topcu, "Traffic management for urban air mobility," in *NASA Formal Methods - 11th International Symposium, NFM 2019, Houston, TX, USA, May 7-9, 2019, Proceedings*, 2019, pp. 71–87.

[9] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *AAAI Conference on Artificial Intelligence*, 2018.

[10] B. Könighofer, M. Alshiekh, R. Bloem, L. Humphrey, R. Könighofer, U. Topcu, and C. Wang, "Shield synthesis," *Formal Methods in System Design*, vol. 51, no. 2, pp. 332–361, Nov 2017.

[11] S. Bharadwaj, R. Bloem, R. Dimitrova, B. Konighofer, and U. Topcu, "Synthesis of minimum-cost shields for multi-agent systems," in *American Control Conference (ACC)*, July 2019, pp. 1048–1055.

[12] N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of reactive(1) designs," in *Verification, Model Checking, and Abstract Interpretation, 7th International Conference, VMCAI 2006, Charleston, SC, USA, January 8-10, 2006, Proceedings*, 2006, pp. 364–380.

[13] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of reactive(1) designs," *J. Comput. Syst. Sci.*, vol. 78, no. 3, pp. 911–938, 2012.

[14] R. Ehlers and V. Raman, "Slugs: Extensible GR(1) synthesis," in *Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part II*, 2016, pp. 333–339.

[15] S. Bharadwaj, R. Dimitrova, and U. Topcu, "Synthesis of surveillance strategies via belief abstraction," *CoRR*, vol. abs/1709.05363, 2017, http://arxiv.org/abs/1709.05363.

[16] J. Alonso-Mora, J. A. DeCastro, V. Raman, D. Rus, and H. Kress-Gazit, "Reactive mission and motion planning with deadlock resolution avoiding dynamic obstacles," *Auton. Robots*, vol. 42, no. 4, pp. 801–824, 2018.

[17] S. Moarref and H. Kress-Gazit, "Reactive synthesis for robotic swarms," in *Formal Modeling and Analysis of Timed Systems - 16th International Conference, FORMATS 2018, Beijing, China, September 4-6, 2018, Proceedings*, 2018, pp. 71–87.

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

[19] S. L. Smith, J. Túmová, C. Belta, and D. Rus, "Optimal path planning for surveillance with temporal-logic constraints," *The International Journal of Robotics Research*, vol. 30, no. 14, pp. 1695–1708, 2011.