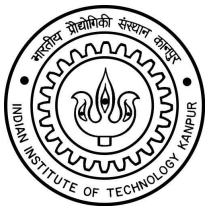


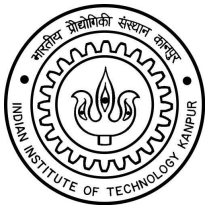
CS638 Book Chapter

Motion Planning With Probabilistic Guarantee

Jatin Jindal
Sahil Dhull
Vibhor Porwal

Instructor

Dr. Indranil Saha



IIT Kanpur

Introduction:

In real time dynamic environments, offline algorithms and controls won't work efficiently for robotic systems. Moreover, since the environment is dynamic it is not sure if a certain specification could be satisfied or not. And mostly we encounter such environments for robotic systems and hence arises the need to develop such efficient (and online) algorithms which satisfy certain specifications with some good probability. The algorithms that we will see here are useful in various domains like controlling unmanned aircraft and surgical steering needles.
Goal of this chapter is to find control strategy for robot and maximum (or minimum) probability to satisfy a specification in various kinds of environments like discrete (or continuous) time dynamic environments.

1 Control Strategy for Discrete Time Dynamic Environment

1.1 Objective:

Given the motion specification of a robot in the form of PCTL (probabilistic computation tree logic) formula, we need to find a control strategy that maximizes the probability of satisfying the PCTL specification in a dynamic environment.

1.2 Problem Description:

The environment is dynamic, containing a set of disjoint regions. Some regions are separated from each other by doors which can open or close. We assume that robot can determine its current region accurately and motion primitives allow it to move from one region to another provided the path is not blocked by closed door. Moreover, door transition occurs in synchronisation with robot transitions.
There are 3 different settings that we discuss:

1. **Setting 1:** Assumption is that when a robot observes a door, it knows perfectly well if it is opened or closed. Moreover, robot knows a priori on the probability that each door is open or closed.

2. **Setting 2:** Here the assumption on priori probability is removed. But 100% sensing accuracy is still assumed.

3. **Setting 3:** Here both assumptions are removed. However, another assumption is made i.e. success and failure rates of door observation are known and fixed (given in the form of probabilities).

Definition 1.1. Markov Decision Process (MDP) $\mathcal{M}$ is a tuple (S,s_0,Act,T,Π,L,c), where S is a finite set of states, s_0 $\in$S is the start state, Act is finite set of actions, $T : S \times Act \times$ S $\rightarrow$ [0,1] is transition probability function, Π is a finite set of atomic propositions, L:S$\rightarrow 2^{\Pi}$ is the labelling function assigned to each s in S, Cost(c): $S \times Act \rightarrow \mathcal{R}^{\geq 0}$

Definition 1.2. Syntax of PCTL:
$\phi ::= true \mid \pi \mid \neg\phi \mid \phi \wedge \phi \mid P_{\bowtie p}[\psi] \mid \varepsilon_{\bowtie c}[\phi]$ state formulas
$\psi ::= X\phi \mid \phi \mathcal{U}^{\leq k}\phi \mid \phi \mathcal{U}\phi$ path formulas
$\bowtie \in \{\leq, <, \geq, >\}; p \in [0,1], c \in [0,\infty]$ and $k \in \mathcal{N}$

Definition 1.3. Semantics of PCTL: For any state $s \in S$, the satisfaction relation $\models$ is defined inductively as follows:
(1) s $\models$ true for all $s \in S$;
(2) s $\models \pi \Leftrightarrow \pi \in L(q)$;
(3) s $\models (\phi_1 \wedge \phi_2) \Leftrightarrow s \models \phi_1 \wedge s \models \phi_2$;
(4) $s \models \neg\phi \Leftrightarrow s \not\models \phi$;

(5) $s \models P_{\bowtie p}[\psi] \Leftrightarrow p^s_\mu \bowtie p$;
(6) $s \models \varepsilon_{\bowtie c}[\phi] \Leftrightarrow e^s_\mu \bowtie c$;
p^s_μ is the probability of all the infinite paths that start from s and satisfy ψ under policy μ and $e^s_\mu(\phi)$ denotes the total expected cost of reaching a state that satisfy ϕ from s under μ. Moreover, for any path $\omega \in$Path.
(1) $\omega \models X\phi \Leftrightarrow w(1) \Leftrightarrow \phi$
(2) $\omega \models \phi_1 \mathcal{U}^{\leq k} \phi_2 \Leftrightarrow \exists i \leq k, \omega(i) \models \phi_2 \wedge w(j) \models \phi_1 \forall j < i$;
(3) $\omega \models \phi_1 \mathcal{U} \phi_2 \Leftrightarrow \exists \geq 0, \omega \models \phi_1 \mathcal{U}^{\leq k} \phi_2$

Definition 1.4. Probabilistic Computation Tree Logic (PCTL)
We will be only dealing with PCTL of form $P_{max=?}[\phi_1 \mathcal{U} \phi_2]$ which is the maximum probability for which there exists a policy π such that the formula "ϕ_1 until ϕ_2" is satisfied.

Definition 1.5. Mixed Observability Markov Decision Process (MOMDP) [5] is a tuple (S,s_0,X,Θ,Act,T,O,L), where S is the set of fully observable states, s_0 is the initial fully observable state, X is the set of hidden states, Θ is finite set of observations, Act is a finite set of actions, T(s,x,α,$s^{'}$) = P($s^{'}$ |s,x,α) is probability of making a transition to fully observable state $s^{'}$ if action α is applied in state s and hidden state is x, O($s^{'}$,$x^{'}$,α,o) = P(o| $s^{'}$,$x^{'}$,α) describe the probability of observing o from state s when hidden state is x after action α, and L:S$\rightarrow 2^{AP}$ is the labelling function.

1.3 Setting 1:

In this setting, assumption is that when a robot observes a door, it knows perfectly well if it is opened or closed. Moreover, robot knows a priori on the probability that each door is open or closed.

1.3.1 MDP Model construction:

$\mathcal{M}$ = (S,s_0,Act,T,L).

- S is the set of all regions.
  S = $R_1 \cup \bigcup_{r \in R_2} \{(r,0),(r,1)\}$ where R_1 is the regions having no door and R_2 is set of regions having door.
- Act is set of actions which are motion primitives available at each region/state.
- Assumption: Transition Probability is only dependant on current region.
  Hence T(s,α,$s^{'}$) = P($s^{'}$ |s,α).
- L is the labelling function assigning properties to various regions, which are used in PCTL specification formation.

1.3.2 PCTL Control Synthesis:

Once the MDP is contructed, problem is solved by Genral Algorithm as mentioned in Appendix. Inputs are an MDP and a PCTL specification and output will be policy and maximum probability.

1.4 Setting 2:

Here the assumption on priori probability is removed i.e. Robot doesn't know the probability of a door being open or closed. But sensors are still assumed to be perfect i.e. 100% sensing accuracy is still assumed.

1.4.1 MOMDP Model construction:

The Tuple (S,s_0,X,Θ,Act,T,O,L) is as follows:

- S = R (set of regions)

- X = $\bigcup_{b_i \in B} b_i$ where is b_i is boolean denoting state of each door i.e. $b_i = 1$ means door is closed and $b_i = 0$ means door is open.
- $\Theta = \{open, closed\}$ denoting states of door.
- Act is the set of primitives available at the state/region.
- T(s,x,α,$s^{'}$) = P($s^{'}$ |s,α) for the regions without door.
  For the regions with door,
$$T(s, x, \alpha, s^{'}) = \sum_{s \in S} P(s^{'} \mid s, \alpha) P(o \mid s^{'}, x^{'}, \alpha)$$
- Since we have perfect sensing, $P(o \mid s^{'}, x^{'}, \alpha) = 1$ if o is open and 0 if o is closed.
  $O(o, s^{'}, x^{'}, \alpha) = P(o \mid s^{'}, x^{'}, \alpha) = 0$ or 1
- L is the labelling function assigning properties to various regions over which PCTL specification is defined.

1.4.2 PCTL Control Synthesis:

We only know the state of current door, so we assume all other doors are open. This leaves us with n+1 configurations, where n is the number of doors. For each configuration, MDP is constructed, and optimal policy is obtained. Let Π denote set of policies for all configurations/scenarios with pi_0 being scenario in which all doors are open.
We partition S into S^{yes} (states satisfying PCTL formula with probability 1), S^{no} (states satisfying PCTL formula with probability 0) and $S^{?}$ (remaining states).
Now while running, robot chooses the policy dynamically from Π based on its current configuration (e.g. if it enters a region containing door and door is closed, it will choose corresponding π_i), applies that policy to obtain new state, until the state is in either S^{yes} or S^{no}. The algorithm is given below with line 10 removed (line 10 is for setting 3).

Algorithm 1 Setting 2 and 3 Online Algorithm [5]

Require: Set $\Pi = \{\pi_0, \pi_1, ..., \pi_N\}$
1: $\pi_* = \pi_0$
2: Act(s) $\leftarrow \pi_*$(s)
3: $s^{'}$ = execute Act(s)
4: s = $s^{'}$
5: **while** True **do**
6: **if** s$\in S^{yes} \vee s \in S^{no}$ **then**
7: **break**
8: **else**
9: **if** P(closed| $s^{'}$, b_i,Act(s))=1 **then**
10: π_i = SOLVE
11: $\pi_* = \pi_i$
12: **end if**
13: **goto** 2
14: **end if**
15: **end while**

1.5 Setting 3:

Here both assumptions are removed (i.e. robot doesn't know any priori probability of door being open or closed, and neither does it sense the state of door with perfection). However, another assumption is made i.e. success and failure rates of door observation are known and fixed (given in the form of probabilities).

1.5.1 MOMDP Model construction:

Here, uncertainity in robot's sensor is also considered. Only the transition probabilities for the regions containing a door change as per the following equation,

$$T(s, x, \alpha, s^{'}) = \sum_{s \in S} P(s^{'} \mid s, \alpha) P(o \mid s^{'}, x^{'}, \alpha)$$

Now, $P(o \mid s^{'}, x^{'}, \alpha)$ will not be directly 0 or 1 depending on state of door, but we also incorporate some error/inaccuracy.

1.5.2 PCTL Control Synthesis:

Π is calculated offline similar to last setting. When a region having a door is encountered, robot can either keep applying same policy or execute a new policy. The policy selected corresponds to the scenario that is most likely based on observation and given by following one-step lookahead,

$$p_s = \max_{\pi \in \Pi} \{ \sum_{s' \in S^?} T(s, x, \pi, s^{'}) \cdot p_{s'} + \sum_{s' \in S^{yes}} T(s, x, \pi, s^{'}) \} \quad -- > SOLVE$$

1.6 Extension to Continuous-time Systems

One of the key limitations of the above approaches is the assumption that the doors only change state in synchrony with the motion of the robot i.e Discrete Time system.
Now we will consider Continuous time systems (and hence Continuous Stochastic Logic or CSL specifications) in a dynamic environment within a given time bound.

Assumptions: Robot knows about state of door only when it is in a region adjacent to door. And to account for noisy actuators, it is considered that if robot is to move to a region, it may actually move to some other adjacent region. Change in state of door is assumed to follow Poisson distribution, so the time between 2 subsequent switching events are exponential random variables. The time for which robot stays in a given region is modeled as exponential distribution.

Definition 1.6. Continuous Time Markov Decision Process (CTMDP) $\mathcal{M}$ is a tuple (S,Act,**R**,T,L) where S, Act, T, L are same as defined in an MDP. **R**: S×Act×S→ $\mathcal{R}_{\geq 0}$ is a rate function such that for each s∈S there is a pair $(\alpha, s^{'})$ ∈Act×S with **R**(s,α,$s^{'}$) > 0.

Definition 1.7. Locally Uniform CTMDP A CTMDP $\mathcal{M}$ is locally uniform if $\forall s \in S$ and $\forall \alpha, \beta \in$ Act(s), E(s,α) = E(s,β), where E(s,α) = $\sum_{s' \in S}$ **R**(s,$\alpha, s^{'}$) is the exit rate. So we denote the exit rate of state s as **E(s)** in locally uniform CTMDP.

Definition 1.8. Discrete Time Markov Decision Process (DTMDP)
For a given CTMDP $\mathcal{M}$, the embedded locally uniformized discrete time Markov decision process (DTMDP) is (S, Act, $T^{u(\mathcal{M})}$,L) where $\forall \alpha \in$ Act(s),

$$T^{u(\mathcal{M})}(s, \alpha, s^{'}) = \begin{cases} \frac{E(s,\alpha)}{\mathbf{E}(s)} T(s, \alpha, s^{'}) & if s \neq s^{'} \\ \frac{E(s,\alpha)}{\mathbf{E}(s)} T(s, \alpha, s^{'}) + 1 - \frac{E(s,\alpha)}{\mathbf{E}(s)} & if s = s^{'} \end{cases}$$

Definition 1.9. Path ω of CTMDP is an infinite sequence:

$$s_0 \xrightarrow{\alpha_0, t_0} s_1 \xrightarrow{\alpha_1, t_1} s_2 \xrightarrow{\alpha_2, t_2} ...$$

Path^{fin} denotes the set of non-empty finite sequences of states from ω and Path^{inf} denotes the set of non-empty infinite sequences of states from ω

Definition 1.10. Time Measurable Policy for a CTMDP $\mathcal{M}$ is defined as a mapping π: Path^{fin} × $\mathbb{R}_{\geq 0}$ × Act → $[0, 1]$, such that $\forall t_i \in \mathbb{R}_{\geq 0}$ and $\omega \in$ Path^{fin}, the functions $\pi(\omega, t, .)$: Path^{fin} × $\mathbb{R}_{\geq 0}$ → $[0, 1]$ yield a probability distribution over all $\alpha \in$ Act.

Definition 1.11. Late TTP Policies and, Piecewise constant and non-Zeno late TTP Policies:
Time measurable policies that are based on the current state and the total elapsed time are known as late total time positional (late TTP) policies.
A late TTP policy is piecewise constant and non-Zeno if for any state s ∈ S and any time bound t, the frequency at which actions change is finite.

Definition 1.12. Periodic Policy is a piecewise constant and non-Zeno late TTP policy if $\forall s \in S$ and k$\in \mathbb{N}, \exists \alpha \in$Act such that for the elapsed time within the interval [k$\mathbf{T}$; (k + 1)$\mathbf{T}$) of period $\mathbf{T}$, the chosen action is α.

Definition 1.13. Continuous Stochastic Logic (CSL) has the syntax:
CSL State Formula:

$$\Phi := true \mid a \mid \neg\Phi \mid \Phi_1 \vee \Phi_2 \mid \Phi_1 \wedge \Phi_2 \mid \mathbb{P}_{\sim\lambda}[\phi] \mid \mathbb{S}_{\sim\lambda}[\Phi]$$

where a ∈ AP, ϕ is a path formula, $\sim\in \{<, \leq, >, \geq\}$ is a comparison operator, and $\lambda \in [0, 1]$ is a probability threshold.
CSL Path Formula:

$$\phi := \mathcal{X}^I\Phi \mid \Phi_1\mathcal{U}^I\Phi_2$$

where Φ, Φ_1, and Φ_2 are state formulae and I $\subseteq \mathbb{R}_{\geq 0} \cup \{\infty\}$

1.6.1 Control Synthesis of CTMDPs with CSL Path Formulae

- Next Optimal Operator ($P_{max=?}\mathcal{X}^I\Phi$):
  Let s be the current state of system. So we have to find optimal action to satisfy the formula within I time period in exactly 1 transition.

  $$\pi^*(s) = arg\ max_{\alpha\in Act(s)}(e^{-\mathbf{E}(s)\text{inf I}} - e^{-\mathbf{E}(s)\text{sup I}}) \sum_{s'\models\Phi} T(s, \alpha, s')$$

  So we define a indexed probability vector,

  $$\bar{\Phi} = \begin{cases} e^{-\mathbf{E}(s)\text{inf I}} - e^{-\mathbf{E}(s)\text{sup I}} & if s \models \Phi \\ 0 & o/w \end{cases}$$

  And further we can calculate the probability of satisfying $\mathcal{X}^I\Phi$ from each state by multiplying the Transition probability matrix and the vector $\bar{\Phi}$.
  The case of $P_{min=?}$ can be solved in a similar way to that of $P_{max=?}$

- Until ϵ-Optimal Operator ($P_{max=?}[\Phi\mathcal{U}^I\Psi]$):
  This kind of specification is solved using a method called Fixed Point Characterization.
  We assume that I = [0,t] and $\mathbf{P}$(s,α,s',x) = $\mathbf{E}$(s)$s^{-\mathbf{E}(s)x}$T(s,α,s') represents the probability that a transition will occur from s to s' after applying action α within x time units (x≤t).
  Let $p_{max}(s, t)$ denote the maximum probability of satisfying the formula within t time units. The function (s,t) → $p_{max}(s, t)$ is the fixed point of higher-order operator Ω:(S×$\mathbb{R}_{\geq 0}$ → [0, 1]) → (S×$\mathbb{R}_{\geq 0}$ → [0, 1]), defined $\forall s \in S, t \in \mathbb{R}_{\geq 0}$ and measurable function F:S×$\mathbb{R}_{\geq 0}$, s.t.

  $$\Omega(F)(s, t) = \begin{cases} 1 & if s \models \Psi \\ \int_0^t \max_{\alpha\in Act(s')} \mathbf{P}(s, \alpha, s', x)F(s', t - x)dx & if s \models \Phi \wedge \neg\Psi \\ 0 & o/w \end{cases}$$

  The solution to the nontrivial case implies solving recursively a set of Volterra integral equations. There are 2 possible approaches: Numerical Integration and Transformation of Integral equation into a system of differential equations. But these methods are time consuming and numerically unstable.

What we can do is consider first T units of time in the interval [0,t], and split the integral as follows:

$$p_{max}(s,t) = \int_0^T max_{\alpha \in Act(s)} \mathbf{P}(s,\alpha,s^{'},x)\dot{p}_{max}(s^{'},t-x)dx + \int_T^t max_{\alpha \in Act(s)} \mathbf{P}(s,\alpha,s^{'},x)\dot{p}_{max}(s^{'},t-x)dx$$

For sufficiently small T, it becomes the following:

$$p_{max}(s,t) \approx max_{\alpha \in Act(s)}(1-e^{-\mathbf{E}(s)T}) \sum_{s' \in S} T(s,\alpha,s^{'}) + p_{max}(s^{'},t-T) + e^{-\mathbf{E}(s)T} p_{max}(s,t-T)$$

First summand represents the probability that only one transition occurs during the interval [0,T]. The result of this will be lower bound for $p_{max}(s,t)$.
To find the upper bound, let $\mathbf{E}(s) = \max_{s \in S}\mathbf{E}(s)$ be the maximum exit rate. We know the number of transitions in [0,T] is poisson distribution, so in the worst case, the maximum ϵ error $= \frac{(ET)^2}{2}$ (follows from derivation of taylor expansion of exponential function). For a fixed ϵ upper bound, the number of steps k satisfy $\epsilon \frac{(ET)^2}{2k}$.
Based on the above approximation, a discrete time MDP $\tilde{\mathcal{M}}$ = (S,Act,$\tilde{\mathcal{T}}$,L) defined as:

$$\tilde{\mathcal{T}}(s,\alpha,s^{'}) = \begin{cases} (1-e^{-\mathbf{E}(s)T})T(s,\alpha,s^{'}) & if\ s \neq s^{'} \\ (1-e^{-\mathbf{E}(s)T})T(s,\alpha,s^{'}) + e^{-\mathbf{E}(s)T} & if\ s = s^{'} \end{cases}$$

For a given upper bound on approximation error, ϵ, an ϵ-optimal policy can be found by solving the maximization problem:

$$\pi^*(s,k) = arg \max_{\alpha \in Act} \Big(\sum_{s' \models \Phi \wedge \neg\Psi} \tilde{T}(s,\alpha,s^{'})p(s^{'},k-1) + \sum_{s' \models \neg\Phi \wedge \Psi} \tilde{T}(s,\alpha,s^{'})p(s^{'},k-1) \Big)$$

here $\pi^*(s,k)$ denotes the ϵ-optimal periodic policy that when applied at s generates the probability of reaching a Ψ-state starting from state s in at most k steps in the induced discrete MDP $\tilde{M}$. Hence, using dynamic programming techniques (such as value iteration), $p^{max}(s,t)$ can be found upto an ϵ error.
The solution for min probability case is similar.

- Complexity:
  The overall time complexity for CSL control synthesis is linear in size of the formula and polynomial in size of model for PCTL.

1.6.2 CTMDP Model of Robot Motion in a Changing Environment

Till now we constructed the CTMDP based on given CSL path formula. In this section we apply that to scenario discussed in the paper[6].

CTMDP Model Construction:

Definition 1.14. A changing environment is a tuple ϵ = (R,D,A,C,H) where,

- R = $\{r_1, r_2, ..., r_k\}$ is the set of k mutually disjoint regions.
- D = $\{d_1, d_2, ..., d_N\}$ is the set of N doors.
- A $\subseteq R \times R$ denotes the adjacency list of regions (without door in between)
- C = $\{c_1, c_2, ..., c_N\}$ is set of boolenas denoting the state of each door. $c_i = 1$ means door i is closed and 0 means door i is open.
- H $\subseteq D \times R$ is adjacency relation between regions and doors.

Let R_d denotes the regions having door adjacent and R_{nd} denotes the rest of regions.
So **CTMDP** $\mathcal{M}$ = (S, Act, R, T, L) is as follows:

- S = $R_{nd} \cup \bigcup_{r \in R_d} \{(r,0),(r,1)\}$
- Act includes the set of action primitives available at each region and decisions that allow robot to move in dynamic environment.
- Since the transition only occur at the end of a sojourn in a region, T(s,α,$s^{'}$) = Pr($s^{'}$ |s,α).
- The sojourn time in s under some act is exponentially distributed on interval [0,x] i.e. E(s,α)$e^{-E(s,\alpha)x}$, with the exit rate E(s,α) = $\sum_{s \in S} \mathbf{R}(s,\alpha,s^{'})$.
- L is the labelling function, containing labels assigned to different regions.

Dynamic Indoor Concurrent Environment (DICE):[6]
DICE was developed to capture the motion of robot in a dynamic environment. It is a simulation/experimental platform based on Robotic InDoor Environment (RIDE) simulator.

1.7 Uncertainity about Environment

To find the control policy satisfying a given LTL specification with maximum probability for a robot with probabilistic control and uncertain observations in graph-like environment.
An example of this kind of setting is as follows: We can model a task where a robot is operating in an indoor environment, and is required to pick-up and deliver items among some rooms. Non-determinism occurs in observations because items may or may not be available when a robot visits a room.

1.7.1 Problem Settings:

Probability Measure : We define $\text{Paths}^M_{\mathcal{M}}$ and $\text{FPaths}^M_{\mathcal{M}}$ as the set of all infinite and finite paths of an MDP $\mathcal{M}$ under a policy M starting from any initial state s_0. Let $\text{Paths}^M_{\mathcal{M}}(s_0,\ldots,s_n)$ denote the set of all paths in $\text{Paths}^M_{\mathcal{M}}$ with the prefix $s_0 s_1 \ldots s_n$.
We will define the probability measure Pr^M over $\text{Paths}^M_{\mathcal{M}}$ containing $\text{Paths}^M_{\mathcal{M}}(s_0 \ldots s_n)$ to be,

$$Pr\{\text{Paths}^M_{\mathcal{M}}\} = l(s_0) \prod_{0 \le i \le n} \mathcal{P}(s_i, \mu(s_i), s_{i+1})$$

where, l is the initial state distribution.
Environment Model : We will consider the region in which robot moves to be represented as follows:

$$\epsilon = (V, \delta_\epsilon, \Pi)$$

where V is the set of vertices, $\delta_\epsilon \subseteq V \times V$ is the set of edges, and Π is the set of atomic propositions. We will assume that the environment is non blocking which means that there is no vertex with outdegree 0.
Robot Motion Model : The robot draws its actions from a finite set U, $A : V \to 2^U$ which provides the set of actions available at a vertex of the environment. $P_m : V \times U \times V$ is the probabilistic transition function defined in the usual manner.
Our specification will be given as an LTL formula over Π.
We will consider the problem setting where the observations of the properties of environment are probabilistic. We have a probability function $P_0 : V \times \Pi \to [0,1]$ and $P_0(v,\pi)$ represents the probability that atomic proposition π is observed at a vertex v when v is visited. Let Π_v and Z_v respectively be the atomic propositions that can be observed at vertex v and set of all possible observations at vertex v.

1.7.2 Solution and Reformulation:

Problem 1.9.1 : Assume that the initial location(state) of the robot is v_0, our problem is to find an reactive control strategy in the form of an infinite sequence of actions for the robot, $C = \{V_0, V_1, \dots\}$ where $V_i : V \times 2^{\Pi} \to U$. Under control strategy C, the trajectory produced is $r = v_0 v_1 \dots$, at time i action is $V_i(v_i, o_i)$. It is worthwhile noting that the trajectory and its corresponding word are not unique given C and v_0 because of non-determinism in both motion and observation of the robot.
Formally we can state the problem as follows : Given the environment model, ϵ, the robot motion model U, A, P_m, the observation model P_0 and an LTL specification, we have to find a control strategy C that maximizes the probability that the word generated under C satisfies ϕ.
Solution Approach : We will first construct an MDP $\mathcal{M}$ corresponding to our system, and then synthesize an optimal policy for it, due to the fact that observations at each vertex are independent the policy that maximizes the probability of satisfaction of ϕ in $\mathcal{M}$ corresponds to optimal control strategy for our problem.
MDP Construction and Reformulation of Problem: Given environment ϵ , robot motion model U, A, P_m and observation model P_0 we construct MDP $\mathcal{M} = (S, \mathcal{U}, \mathcal{A}, \mathcal{P}, l, \Pi, h)$ as follows: $\mathcal{S} = \{(v, Z) | v \in V, Z \in Z_v\}$, $\mathcal{U} = U$, $\mathcal{A}((v, Z)) = A(v)$, $l(s) = \prod_{\pi \in Z} \mathcal{P}(v_0, \pi) \times \prod_{\pi \notin Z} (1 - \mathcal{P}(v_0, \pi))$ if $s = (v_0, Z)$ for any $Z \in Z_{v_0}$, $l(s) = 0$ otherwise, $h((v, Z)) = Z \; \forall (v, Z) \in S$ and $\mathcal{P}((v, Z), u, (v', Z') = P_m(v, u, v') \times \left(\prod_{\pi \in Z'} P_0(v', \pi) \times \prod_{\pi \notin Z'} (1 - P_0(v', \pi))\right)$
Note here that h is the labelling function.
Now we will reformulate our problem as follows:
Problem 1.9.2 : Given an MDP $\mathcal{M}$ and an LTL specification ϕ, find a policy that maximizes $Pr^M_{\mathcal{M}}(\phi)$.
By our MDP construction it is clear that a strategy $C = \{V_0, \dots\}$ is a solution to Problem 1.9.1 iff the policy $\mathcal{M} = \{\mu_0, \mu_1, \dots\}$ where $\mu_i((v_i, Z_i)) = V_i(v_i, Z_i) \; \forall \; i$ is a solution to Problem 1.9.2 For a detailed proof look at [8].
Next we present the algorithm for solving Problem 1.9.2 For detailed analysis and explanation refer to [8]:

Algorithm 2 Given MDP $\mathcal{M}$, LTL formula ϕ, generate the optimal policy M [8]

1: Translate the LTL formula ϕ to a deterministic Rabin automata R_ϕ
2: Generate the product MDP $\mathcal{M}_P = \mathcal{M} \times R_\phi$ and accepting state pairs $F_P = \{(L^P_1, K^P_1), \dots, (L^P_k, K^P_k)\}$
3: Find all accepting maximum end components for all pairs in F_P, and find their union B_P.
4: Find the stationary policy $\{\mu^*_P, \mu^*_P, \dots\}$ maximizing the probability of reaching B_P.
5: Generate the policy $M^*_P = \{\mu^P_0, \dots\}$ as follows: $\mu^P_i(p) = \mu^*_P(p)$ if $p \in S_P \; B_P$. Otherwise, p is in at least one accepting maximum end component. If it is (S'_P, A'_P) and $A'_P(p) = \{\mu_1, \dots, \mu_m\}$ then $\mu^P_i(p) = u_j$ where $j = i \bmod m$.
6: Generate the policy $M^* = \{\mu_0, \mu_1, \dots\}$ induced by M^*_P.

Now, here is the final algorithm to solve Problem 3.3.1:

Algorithm 3 Generate the optimal control strategy C^* given $\epsilon, U, A, P_m, P_0, \phi$ [8]

1: Generate the MDP $\mathcal{M}$ from the environment model ϵ, the motion primitives U, the actions A, the motion model P_m and the observation model P_0.
2: Use Algorithm 2 to generate optimal policy M^* for $\mathcal{M}$.
3: Generate the control strategy $C^* = \{V_0, V_1, \dots\}$ corresponding to M^* by setting $V_i(v, Z) = \mu_i((v, Z))$ for all i.

2 Presence of Multiple Agents

2.1 Incremental control synthesis in presence of Multiple agents

To generate optimal control policy for a robot in presence of independent uncontrollable agents in a graph like environment when mission specification is given as an sc-LTL formula.
This kind of problem arises in many real life settings for example, consider a setting where a car(robot) is required to go from one location to another without colliding with the pedestrians(agents) crossing the road can be formulated as the above problem.

2.1.1 Problem Settings:

First we will define what is a Markov Chain.
Markov Chain : A markov chain is a tuple $M = (Q_M, q_M^0, \delta_M, \Pi_M, \mathcal{L}_M)$ where Q_M are the set of states, q_M^0 is the initial state, Π_M is the set of atomic propositions, and $\mathcal{L}_M$ is the satisfaction map. $\delta_M : Q_M \times Q_M \to [0, 1]$ is the transition probability function that satisfies $\sum_{q' \in Q_M} \delta_M(q, q') = 1 \; \forall \; q \in Q_M$.
We will model agents as MDPs with a unique initial state.

2.1.2 Solution and Reformulation:

System Model: We model our environment as a graph,

$$\epsilon = (V, \to_\epsilon, \mathcal{L}_\epsilon, \Pi_\epsilon)$$

where V is the set of vertices, $\to_\epsilon \subseteq V \times V$ is the set of edges, and $\mathcal{L}_\epsilon$ is the function which assigns labels to each node in V.
We will model our robot as a Deterministic transition system T. We assume that our system is synchronous i.e. the robots and agents make transitions synchronously by choosing edges of the environment.
we define, $\Pi_T = \{(T, \mathcal{L}_\epsilon(q)) | q \in Q_T\}$, here Q_T are set of states of robot, and is a subset of V and $(L)_T(q) = (T, \mathcal{L}_\epsilon(q))$ similarly we define $\Pi_i = \{(i, \mathcal{L}_\epsilon(q)) | q \in Q_i\}$ and $(L)_i(q) = (i, \mathcal{L}_\epsilon(q))$. The set of propositions Π becomes $\Pi_T \vee \Pi_1 \cdots \vee \Pi_n$. Here, n is the no of agents in our system.
Problem Formulation : Given a robot T, a set of independent agents $M_1, \ldots, M_n$, functioning on environment ϵ, and an sc-LTL specification over Π find a control policy μ^* that maximizes the probability of satisfaction of ϕ or if a probability threshold p_t is given then the probability that the system satisfies ϕ under this policy is at least p_t. If there is no such policy then return false.
Solution Outline : We can solve the above problem by reducing it to a Maximum reachability problem(MRP) on the MDP representing the parallel composition of all system components. The time complexity of this algorithm is exponential in the no of agents. Thus, this approach is not practically feasible. Here, we will give an efficient incremental solution for our problem. Each iteration of our algorithm will involve synthesis of an optimal control policy considering only a subset of agents, verification of the policy and reduction of the size of the system using synthesis and verification steps.
There are two modes in which our algorithm functions, avoid mode and reach mode. Avoid mode is when the no. of agents that can satisfy the specification is not greater than the no. of agents that can violate the specification and vice versa for reach mode. Then we synthesize an optimal policy considering only a subset of agents (subset of agents that can satisfy ϕ if avoid otherwise set of agents that can violate ϕ). Now we synthesize optimal control policy for that subset of agents by solving an MRP. Once we have synthesized the optimal policy for this subset of agents, we induce this policy on this partial system and obtain a Markov chain, then we calculate the probability that the policy synthesized satisfy the specification for the complete product system (steps 36-39 of Algorithm 4). Now, we reduce the size of the system A_i by removing the transitions from A_i that are no longer needed in subsequent steps. For a proof that reduction does not affect correctness of our algorithm see [9].

The overall algorithm is presented in Algorithm 4.

3 Controller Synthesis with optimality guarantee

3.1 MDP cost minimization

The problem is to synthesize a policy for an MDP such that this policy satisfies an specification almost surely, if such a policy exists. We also want an optimizing proposition to be repeatedly satisfied, and formulate a criterion in terms of minimizing the expected total cost in between satisfactions of the optimizing propositions. The results presented here [10] are useful in various domains, like engineering, biology and economics in problems such as controlling unmanned aircraft and surgical steering needles.

3.1.1 Problem Setting:

We have a dynamical system modelled as a Labelled MDP. We define a labelled MDP below as follows:

Definition 3.1. Labelled Markov Decision Process
We will consider MDP which also has a cost function, $g : S \times U \to \mathbb{R}^+$ where, $g(i, u)$ is the expected cost when action $u \in U(i)$ is taken at state i.

Control policy is defined in the same manner as done in the previous section. A stationary policy is one for which the same control is applied at each stage.
The definition of probability measure is same as defined in the previous sections. Now, we will formally state our problem,
We are given an LTL specification of the form, $\phi = \square\diamond\pi \wedge \psi$. The atomic proposition π is called the optimizing proposition and ψ is an arbitrary LTL formula. The specification means that π should be satisfied infinitely often and ψ should be satisfied. We will assume that there is some policy M of $\mathcal{M}$ such that the MDP under M satisfies ϕ almost surely i.e. $Pr^M_{\mathcal{M}}(\phi) = 1$.
We denote by $\mathbb{M}$ to be the set of all policies, and $\mathbb{M}_\phi$ to be the set of all policies that satisfy ϕ almost surely, we want a policy such that ϕ is almost surely satisfied and the expected cost in between satisfaction of π is minimized. We denote by S_π to be the set of all states where π is true. Each visit to a state in S_π will be called as completing a cycle. The first path starting in an initial state and ending up in S_π is the first cycle, the second cycle is the path starting from the first cycle and ending up in S_π. If there is a path $r^M_{\mathcal{M}} = s_0 s_1 \dots$ we denote $C(r^M_{\mathcal{M}}, N)$ to be the cycle index up to stage N.
Problem 3.1.1 : Find a policy $M = \{\mu_0, \mu_1, \dots\}$, $M \in \mathbb{M}_\phi$ that minimizes,

$$J(s_0) = \lim_{N\to\infty} \sup E\left\{\frac{\sum_{k=0}^{N} g(s_k, \mu_k(s_k))}{C(r^M_{\mathcal{M}}, N)}\right\}$$

where $E\{.\}$ is the expectation.
This problem is related to the standard Average Cost Per Cycle(ACPC) problem, which consist of minimizing

$$J^s(s_0) = \lim_{N\to\infty} \sup E\left\{\frac{\sum_{k=0}^{N} g(s_k, \mu_k(s_k))}{N}\right\}$$

3.1.2 Solution:

We call $J(s_0)$ to be the optimization criterion. First we will define what is a weak accessibility condition, We say that an MDP $\mathcal{M}$ satisfies the weak accesibilty(WA) condition if $\exists$ $S_r \subseteq S$ such that (i) $\exists$ a stationary policy where j is reachable from i for any $i, j \in S_r$ and (ii) states in S
S_r are transient under all stationary policies.

Algorithm 4 Given $T, M_1, \ldots, M_n, \phi$, [optional : p_t] output a control policy μ^* such that $Pr^{\mu^*}(\phi) \geq Pr^{\mu}(\phi)\ \forall\ \mu$ if p_t is not given, otherwise $Pr^{\mu^*}(\phi) \geq p_t$ [9]

$i \leftarrow 0$, $\mathcal{M} = \{M_j | (j,p) \in \phi, j \in \{1, \ldots, n\}\}$
Form $\mathcal{M}^+$ and $\mathcal{M}^-$ using ϕ, for details refer to [9]
$\mu^* \leftarrow \{\}$, $Pr^{\mu^*}(\phi) \leftarrow 0, \mathcal{M}_i \leftarrow \{\}, A_i \leftarrow T, i \leftarrow 1$
s = s
if $|\mathcal{M}^+| \leq |\mathcal{M}^-|$ **then**
  $mode \leftarrow avoid$, $\mathcal{M}_i^{new} \leftarrow \mathcal{M}^+$
else
  $mode \leftarrow reach$, $\mathcal{M}_i^{new} \leftarrow \mathcal{M}^-$
end if
if $\mathcal{M}_i^{new} = \{\}$ **then**
  Add an arbitrary element from $\mathcal{M}$ to $\mathcal{M}_i^{new}$
end if
while True **do**
  $\mathcal{M}_i \leftarrow \mathcal{M}_{i-1} \vee \mathcal{M}_i^{new}$, $\overline{\mathcal{M}_i} \leftarrow \mathcal{M} - \mathcal{M}_i$
  $A_i \leftarrow A_{i-1} \bigotimes \mathcal{M}_i^{new}$
  if $mode = reach$ **then**
    $\mathcal{L}_{A_i}(q) = \mathcal{L}_{A_i}(q) \vee \{(j,p) | (j,p) \in \phi, M_j \in \overline{\mathcal{M}_i}\} \forall q \in Q_{A_i}$
    $\Pi_{A_i} = \Pi_{A_i} \vee (\vee_{M_j \in \overline{\mathcal{M}_i}} \Pi_{M_j})$
  end if
  $P_i \leftarrow A_i \bigotimes F$
  Synthesize μ_i that maximizes $Pr_{P_i}^{\mu_i}(\phi)$ using P_i.
  if p_t given and $Pr_{P_i}^{\mu_i}(\phi) < p_t$ **then**
    Fail : $\nexists \mu$ such that $Pr^{\mu}(\phi) \geq p_t$
  else
    if $Pr_{P_i}^{\mu_i}(\phi) = 0$ **then**
      Fail : ϕ can't be satisfied.
    else
      if $\mathcal{M}_i = \mathcal{M}$ **then**
        Success : Return μ_i.
      else
        if $mode = reach$ **then**
          $\mathcal{L}_{A_i}(q) = \mathcal{L}_{A_i}(q) - \{(j,p) | (j,p) \in \phi, M_j \in \overline{\mathcal{M}_i}\}\ \forall\ q \in Q_{A_i}$
          $\Pi_{A_i} = \Pi_{A_i} - (\vee_{M_j \in \overline{\mathcal{M}_i}} \Pi_{M_j})$
          Perform the same operation on P_i.
        end if
        Obtain the MC $M_{P_i}^{\mu_i}$ induced on P_i by μ_i.
        $M_{\mathcal{M}}^{\mu_i} \leftarrow M_{P_i}^{\mu_i} \bigotimes \overline{\mathcal{M}_i}$
        $R_i \leftarrow M_{\mathcal{M}}^{\mu_i} \bigotimes F$
        Compute $Pr^{\mu_i}(\phi)$ using R_i. For details on computation refer [9]
        if $Pr^{\mu_i}(\phi) > Pr^{\mu^*}(\phi)$ **then**
          $\mu^* \leftarrow \mu_i$, $Pr^{\mu^*}(\phi) \leftarrow Pr^{\mu_i}(\phi)$
        end if
        if $Pr^{\mu_i}(\phi) = Pr^{\mu^*}(\phi)$ **then**
          Success : Return μ^*.
        end if
        if p_t given and $Pr^{\mu^*}(\phi) \geq p_t$ **then**
          Success : Return μ^*.
        else
          $\mathcal{M}_i^{new} \leftarrow \{M_j\}$, where M_j is the smallest agent in $\overline{\mathcal{M}_i}$, reduce the size of A_i, $i \leftarrow i+1$
        end if
      end if
    end if
  end if
end while

A stationary policy induces a Markov Chain with a set of recurrent classes. A state that is not present in any recurrent class is called transient. We say that a stationary policy is unichain if the Markov Chain induced by that policy contain one recurrent class. We call an MDP to be unichain if every stationary policy is unichain. An MDP is called weakly communicating if it satisfies the WA condition. If this MDP satisfies the WA condition with $S_r = S$ then MDP is called communicating. We will solve a related problem first, which is finding a policy for a communicating that minimizes $J(s_0)$. We solve this problem using the method present in [11].
Now assuming that we have the solution for this problem, we will solve our original problem. For that we will first construct the product MDP $\mathcal{P} = M \times R_\phi$ where R_ϕ is the rabin automata for ϕ. There is a one-one correspondence between a path $s_0 s_1 \dots$ on $\mathcal{M}$ and a path $(s_0, q_0)(s_1, q_1) \dots$ on $\mathcal{P}$. The costs along these paths are also the same. Using this one-one correspondence we can easily induce a policy from $\mathcal{P}$ to a policy for $\mathcal{M}$.
Now, we will define Accepting Maximal End Components as follows:

Definition 3.2. Given $(L_\mathcal{P}, K_\mathcal{P}) \in F_\mathcal{P}$, an end component $\mathcal{C}$ is a communicating MDP $(S_\mathcal{C}, U_\mathcal{C}, P_\mathcal{C}, K_\mathcal{C}, S_{\mathcal{C}_\pi}, g_\mathcal{C})$ such that $S_\mathcal{C} \subseteq S_\mathcal{P}, U_\mathcal{C} \subseteq U_\mathcal{P}, U_\mathcal{C}(i) \subseteq U_i \forall i \in S_\mathcal{C}, K_\mathcal{C} = S_\mathcal{C} \wedge K_\mathcal{P}, S_{\mathcal{C}_\pi} = S_\mathcal{C} \wedge S_{\mathcal{P}_\pi}$, and $g_\mathcal{C}(i,u) = g_\mathcal{P}(i,u)$ if $i \in S_\mathcal{C}$, $u \in U_\mathcal{C}(i)$. If $P(i,u,j) > 0$ for any $i \in S_\mathcal{C}$ and $u \in U_\mathcal{C}(i)$ then $j \in S_\mathcal{C}$, in which case $P_C(i,u,j) = P(i,u,j)$. An accepting maximal end components(AMEC) is the largest such end component such that $K_\mathcal{C} \neq \{\}$ and $S_\mathcal{C} \wedge L_\mathcal{P} = \{\}$

Now, our final approach to solve Problem 3.1.1 is summarised here[10]:

1. Obtain the product MDP $\mathcal{P} = \mathcal{M} \times \mathcal{R}_\phi$.
2. Obtain the set of reachable AMECs, denoted as A.
3. For each $\mathcal{C} \in A$: Find a stationary policy $\mu^*_{\to\mathcal{C}}(i)$, defined for $i \in S$ $S_\mathcal{C}$, that reaches $S_\mathcal{C}$ with probability 1. Now find a stationary policy $\mu^*_{cy\ \mathcal{C}}(i)$, defined for $i \in S$ $S_\mathcal{C}$, minimizing $J(s_0)$ for MDP $\mathcal{C}$ and set $S_{\mathcal{C}_\pi}$ while satisfying the LTL constraint, Set $\mu^*_\mathcal{C} = \mu^*_{\to\mathcal{C}}(i)$ if $i \notin S_\mathcal{C}$ and $\mu^*_\mathcal{C} = \mu^*_{cy\ \mathcal{C}}(i)$ if $i \in S_\mathcal{C}$
4. Denote the ACPC of $\mu^*_{cy\ \mathcal{C}}(i)$ by $\lambda_\mathcal{C}$.
5. Now, set $\mathcal{C}^* = \arg\min_{\mathcal{C} \in A} \lambda_\mathcal{C}$ and we have the optimal policy as $\mu^*_{\mathcal{C}^*|\mathcal{M}}$ ($\mu^*_{\mathcal{C}^*|\mathcal{M}}$ is the policy of $\mathcal{C}^*$ induced on $\mathcal{M}$).

4 Application :

Now, we will consider the **problem of controlling a stochastic version of a Dubins vehicle**[4] such that the probability of satisfying a temporal logic specification over a set of properties at the regions in a partitioned environment is maximized. We assume that the vehicle can determine its precise initial position in a known map of the environment. Vehicle is equipped with noisy actuators and, during its motion in the environment, it can only measure its angular velocity using a limited accuracy gyroscope.

A Dubins vehicle ([Dub57]) is a unicycle with constant forward speed and bounded turning radius moving in a plane.

where $(x, y) \in \mathbb{R}^2$ and $\theta \in [0, 2\pi]$ are the position and orientation of vehicle in world frame, u is the control input, U in the control constraint set, ϵ is a random variable denoting the actuator noise which is assumed to be arbitrary continuous probability distribution supported on the bounded interval $[\epsilon_{max}, \epsilon_{max}]$. The forward speed is normalized to 1 and ρ is the minimum turn radius. The state of the system is given by q = $[x, y, \theta]^T$.

This abstraction can be used for any finite set of control units, but the fact that the optimal Dubins path used only three input**([Dub57])**, we assume

$$U = \{-1/\rho, 0, 1/\rho\}$$

We define: $W = \{u + \epsilon | u \in U, \epsilon \in [-\epsilon_{max}, \epsilon_{max}\}$ as the set of applied control units. We assume that the time is uniformly discretized into stages of length Δt where stage k is from (k-1)Δt to kΔt. The control input and the applied control input at stage k as $u_k \in U$ and $w_k \in W$ resp.

The noise ϵ is piece-wise constant, hence the applied control is also piece-wise constant, i.e., $w : [(k-1)\Delta t, k\Delta t] \to W$ in constant over each stage. The robot is equipped with only one sensor, and at stage k it returns the interval $[\underline{w}_k, \bar{w}^k] \subset [u_k - \epsilon_{max}, u_k + \epsilon_{max}]$ containing the applied control input.

Hence, we assume that the vehicle can precisely determine its intial state $q_{init} = [x_{init}, y_{init}, \theta_{init}]^T$ in a known map of the environment. While the vehicle moves gyroscope measurements $\underline{w}_k, \bar{w}^k$ are available at each stage k. We define a vehicle control strategy as a map that takes as input a sequence of measured interval $[\underline{w}_1, \bar{w}^1][\underline{w}_2, \bar{w}^2]..[\underline{w}_{k-1}, \bar{w}^{k-1}]$ and return the control input $u_k \in U$ at stage k.

Solution:

1. Through **quantization and discretization**, we first constructed a finite approximation for the motion of the vehicle in the form of a Markov Decision Process (MDP).

   Given a state q_{k-1}, the state trajectory $q_k(t)$ can by derived by integrating the system from the initial state q_{k-1} and taking into account that the applied control in constant and equal to w_k. The trajectory will be denoted by $q_k(q_{k-1}, w_k, t)$. For practical purposes we assume that length of $\underline{w}_k, \bar{w}_k$ is constant and we denote it by $\Delta\epsilon$ and also assume that $n\Delta\epsilon = 2\epsilon_{max}$. The interval $[-\epsilon_{max}, \epsilon_{max}]$ can be partitioned into n intervals: $[\underline{\epsilon}_i = -\epsilon_{max} + (i-1)\Delta\epsilon, \bar{\epsilon}_i = -\epsilon_{max} + i\Delta\epsilon$. At each stage k, gyroscope returns the measured interval $[u_k - \underline{\epsilon}_k, u_k + \bar{\epsilon}_k]$, and since we know the u_k, we get to know the noise interval(epsilon) interval. We define a representative value $\epsilon_i = \frac{\underline{\epsilon}_i + \bar{\epsilon}_i}{2}$. E is the set of these representative values. So, we define $W_d = \{u + \epsilon | u \in U, \epsilon \in E\}$ as the finite set of control inputs. Let $\omega : U \to W_d$ be a random variable, where $w(u) = u + \epsilon$ with probability mass function 1/n. So, this approximates the set of applied control inputs; for a state q_{k-1} and input u_k, QS returns:

   $$q_k(q_{k-1}, w(u_k), t) = q_k(q_{k-1}, u_k + \epsilon, t)$$

   with probability 1/n.

2. **Reachability Graph:** This is the graph which is formed by showing each transaction with positive value that is returned by QS.

3. **Construction of MDP:**
   Position Uncertainty: To find out whether the robot specifies the proposition or not, it is sufficient to find its projection in $\mathbb{R}^2$. Hence, we describe only position uncertainty of the vehicle when its nominal position is (x,y) as a disc: $D((x, y), \varepsilon)$ centered at (x,y) with radius ε where

   $$\varepsilon_k = max_{[x', y', \theta']^T \in \{\underline{q}_k, \bar{q}_k\}} \{||(x_k, y_k), (x', y')||\}$$

   where $\bar{q}_k(t) = \bar{q}_k(\bar{q}_{k-1}, u_k + \bar{\epsilon}_k, t), \underline{q}_k(t) = \underline{q}_k(\underline{q}_{k-1}, u_k + \underline{\epsilon}_k, t)$

   Environment Model and Specification: The vehicle moves in a static environment $X \subseteq R^2$ in which regions of interest are present. Let Π be the finite set of proposition that are satisfied in the environment. Let $[.] : 2^\Pi \to 2^X$ be a map such that $[\Theta], \Theta \in 2^\Pi$ be a set of positions in

X satisfying all and only propositions $\pi \in \Theta$.

Satisfying ϕ under uncertainty: Assume $q(t)$ and $\varepsilon(t)$ are partitioned into K state and uncertainty trajectories, resp, s.t. $q_k(t) = q(t')$ and $\varepsilon_k(t) = \varepsilon(t'), t' \in [(k-1)\delta t, k\delta t], k = 1, 2, ..K$. The conditions can be written as: (1) $D((x_k(t), y_k(t)), \varepsilon_k(t)) \subseteq [\pi_p]$ for some $t \in [(k-1)\Delta t, k\Delta t]$ and some k, i.e, it reaches a pickup location during the path, (2) $D((x_K, y_K), \varepsilon_k) \subseteq [\pi_d]$ i.e., at the end of trajectory is at drop position, (3) $D((x_k(t), y_k(t)), \varepsilon_k(t)) \cap [\pi_u] = \emptyset$ for all $t \in [(k-1)\Delta t, k\Delta t]$ and all k. The third condition is to ensure that it never reaches an unsafe place. Thus by analyzing $q_k(t)$ when the uncertainty trajectory is $\varepsilon_k(t)$, k=1,..K, we can answer it q(t), when the uncertainty trajectory is $\varepsilon(t)$, is satisfying.

4. Given the task specification as temporal logic statements, we find the MDP control policy using the tools from PCTL.

5. Finally ,the policy is translated to a vehicle feedback control strategy. It can also be shown that the probability that the vehicle satisfies the specification in the original environment is bounded from below by the maximum probability of satisfying the specification on the MDP.

Extension: The abstraction process can only deal with PCTL formulas where the propositions are classified into two non-intersecting sets according to whether they represent regions that must be reached or avoided. In the following section, we don't make this limiting assumption

PCTL formula transformation: [4]

In order to use the method presented at top, we start by removing any negation operators that appear in the initial formula. To do so we use the approach presented in as follows. We introduce the extended set of propositions E_π. In detail, we first define two new sets of symbols $E_{+\pi} = \{\varepsilon_\pi | \pi \in \Pi\}$and $E_{-\pi} = \{\varepsilon_{\neg\pi} | \pi \in \Pi\}$. Then,we set $E_\pi = E_{+\pi} \cup E_{-\pi}$. We also define a translation function pos$(\psi) : \pi \rightarrow_\psi E_\pi$ which takes as input a PCTL formula ψ in NNF and it returns a formula pos (ψ) where the occurrences of terms π and $\neg\pi$ have been replaced by the members ε_π and $\varepsilon_{-\pi}$ of E_π respectively. Since we have a new set of propositions, E_π, we need to define a new map$[\cdot]$ $E_\pi : E_\pi \rightarrow 2^X$ for the interpretation of the propositions. This is straight forward: $\forall \varepsilon \in E_\pi$, if $\varepsilon = \varepsilon_\pi$ then $[\varepsilon] = [\pi]$, else if $(\varepsilon = \varepsilon_{\neg\pi})[\varepsilon] = X \setminus [\pi]$ It can easily be seen that given a formula $\psi \in \psi_\pi$, a map$[\cdot]$: $\pi \rightarrow 2^X$ and a trajectory q(t)of the system,the following holds: q(t) satisfies ψ iff q(t) satisfies $pos(\psi)$.

5 Appendix

5.1 General algorithm to find the MDP control policies for different kind of PCTL formulas.

Case 1: Simple Probabilistic Computation Tree Logic Formulas

- **Next Operator:** For the "next" temporal operator, we present two algorithms. One finds the optimal control strategy, and the other determines all the satisfying policies. For the PCTL formulas that include only one P-operator, the optimal control strategy algorithm is always used.
  - **Next Optimal-** $\phi = P_{max=?}[X\phi_1]$:
    For this operator, we need to determine the action that produces the maximum probability of satisfying X$\phi_1$at each state. Thus, we only need to consider the immediate transitions at each state, and the problem reduces to the following:

$$x^*_{s_i} = \max_{a \in A(s_i)} \sum_{s_j \in Sat(\phi_1)} \sigma_a^{s_i}(s_j)$$

$$\mu^*(s_i) = arg \max_{a \in A(s_i)} \sum_{s_j \in Sat(\phi_1)} \sigma_a^{s_i}(s_j)$$

    where $x^*_{s_i}$ denotes the optimal probability of satisfying ϕ at the state $s_i \in S$, Sat(ϕ_1) is the set of states that satisfy ϕ_1, and μ^* represents the optimal policy. Similar procedure can be done to find the minimum probability.
  - **Next all-** $P_{\bowtie p}[X\phi_1]$: Aim to find all the policies that satisfy the formula
    First find the probabilities of satisfying $X\phi_1$ for each state–action pair, using the N̈ext OptimalÄlgorithm. Then eliminate all the policies that are not in the range of $\bowtie p$.
- **Bounded Until Operator:** two algorithms: optimal and stationary. The optimal algorithm produces a history-dependent control policy. The stationary algorithm results in a stationary policy and is used only for nested formulas.
  - **Bounded Until Optimal:** $\phi = P_{max=?}[\phi_1 \mathcal{U}^{\leq k} \phi_2]$:
    First group the MDP states into three subsets: states that always satisfy the specification S^{yes} , states that never satisfy the specification S^{no} , and the remaining states $S^?$:

$$S^{yes} = Sat(\phi_2)$$
$$S^{no} = S \setminus (Sat(\phi_1) \cup Sat(\phi_2))$$
$$S^? = S \setminus (S^{yes} \cup S^{no})$$

    Hence, the probability of the states in $S^{yes} = 1$ and in $S^{no} = 0$. The probabilities for the remaining states $s_i \in S^?$ are defined recursively. If k = 0, then $p^{s_i}_{max}(\phi_1 \mathcal{U}^{\leq k} \phi_2) = 0 \forall s_i \in S^?.For\ k > 0$

$$x^k_{s_i} = \max_{a \in A(s_i)} \left(\sum_{s_j \in S^?} \sigma_a^{s_i}(s_j) x^{k-1}_{s_i} + \sum_{s_j \in S^{yes}} \sigma_a^{s_i}(s_j) \right), s_i \in S^?$$

$$\mu^{*^k}_{s_i} = arg \max_{a \in A(s_i)} \left(\sum_{s_j \in S^?} \sigma_a^{s_i}(s_j) x^{k-1}_{s_i} + \sum_{s_j \in S^{yes}} \sigma_a^{s_i}(s_j) \right), s_i \in S^?$$

  - **Bounded Until Stationary**—$\phi = P_{\bowtie p}[\phi_1 \mathcal{U}^{\leq k} \phi_2]$: The algorithm is the same as the one for *Bounded Until Optimal* with the exception that the optimal actions determined at each iteration are fixed for the remaining iterations.

- **Unbounded Until Operator** - $\phi = P_{max=?}[\phi_1 \mathcal{U} \phi_2]$:

  Again, we group the MDP states into three subsets: states that always satisfy the specification S^{yes} , states that never satisfy the specification S^{no} , and the remaining states $S^{?}$:

  Find S^{no} using algorithm 1.[All the states which are not reachable with positive probability to states in S^{yes} and $S^{yes} = Sat(\phi_2)$].

Algorithm 5 Find $S^{no} = \{s \in S \mid p_q^{max}(\phi_1 \mathcal{U} \phi_2) = 0\}$

```
R = Sat(φ2)
done = false
while done = false do
  R' = R ∪ {s ∈ Sat(φ1) | ∃a ∈ Act(s), ∃q' ∈ R s.t. σ_a^q(q') >0 }
  if R' = R then
    done = true
  end if
  R = R'
end while
return Q\R
```

We can compute the optimal probabilities for the states in $S^{?}$ by linear programming problem which in fact maximum reachability problem [1].
minimize $\sum_{s_i \in S^{?}} x_{s_i}$ subject to:
$x_{s_i} \geq \sum_{s_j \in S^{?}} \sigma_a^{s_i}(s_j).x_{s_j} + \sum_{s_j \in S^{yes}} \sigma_a^{s_i}(s_j)$
for all $s_i \in S^{?}$ and $(a, \sigma_a) \in Steps(s_i)$
The aforementioned linear programming problem can be solved using classical techniques, such as the simplex method, ellipsoid method [30], or value iteration. The complexity is a polynomial of the size of the MDP.

Case 2: Complex Probabilistic Computation Tree Logic Formulas:
The formulas which consists of more than one *Por* ε operator.
The problem can be solved if all the inner P operators are of the form $P \bowtie [\phi]$ rather than of the form $P_{max=?}[\psi]$
The general nested formula are of the form:

$$\phi = P_{max=?}[X \phi_R]$$

$$\phi = P_{max=?}[\phi_L \mathcal{U}^{\leq k} \phi_R]$$

$$\phi = P_{max=?}[\phi_L \mathcal{U} \phi_R]$$

Solution: first we find the set of initial states S_{ϕ_L}, from which ϕ_L is satisfied and the corresponding control policy μ_{ϕ_L} is find out. Similar things for ϕ_R are also fount out.
This is a sub=optimal algorithm. For the formulas of type-2 and type-3(the one with $\mathcal{U}$ opertor): we need to reach a state in S_{ϕ_R} only by going through states S_{ϕ_L}. For consistency of our solution, we consider only the stationary policies for ϕ_L and we execute action according to that in states S_{ϕ_L} until we reach the state in S_{ϕ_R}. Therefore, we update the MDP to M' in which we remove all the actions in S_{ϕ_L} which are not the part of the stationary policy. For non-blocking MDP, in blocked states add a self loop. To find the maximum probability of satisfying ϕ from the initial state s_0, use the optimal control algorithm for "Until" on M'. In fact, for these formulas, the algorithm may return a suboptimal policy or may not find a solution at all while one might exist. Because of the time complexity we consider only the optimal actions in state S_{ϕ_L}, for actual solution we need to consider all the possible action in this state. For the first case, we can find the optimal solutions by using the Next Optimal algorithm after finding all the satisfying states S_{ϕ_R}.

Case 3: Combining P-Operators by Boolean Operators:

Another method to construct complex PCTL formulas is to combine P-operators by the Boolean operators' conjunction which are of the form, i.e., $\phi = \phi_1 \wedge \phi_2 and \phi = \phi_1 \vee \phi_2$, where both ϕ_1 and ϕ_2 include a path formula. In other words, ϕ_1 and ϕ_2 include $P_{\bowtie p}[\psi]$.

To solve for formulas, i.e., $\phi = \phi_1 \wedge \phi_2$, first, we find the satisfying states S_{ϕ_1} and S_{ϕ_2} and their corresponding control policies μ_{ϕ_1} and μ_{ϕ_2} . In the case that ϕ_1 or ϕ_2 includes the temporal operator X ("next"), all the satisfying policies are considered. If ϕ_1 or ϕ_2 includes the temporal operator $\mathcal{U}^{\leq k}$, the stationary policy is considered. For the case of "until," the optimal control policy is determined. The solution exists only for the initial states, i.e., $S_\phi = S_{\phi_1} \wedge S_{\phi_2}$, given that μ_{ϕ_1} and μ_{ϕ_2} assign the same actions to these states. If $S_\phi = S_{\phi_1} \wedge S_{\phi_2}$, do not intersect, or the assigned actions for the states of S_ϕ by μ_{ϕ_1} and μ_{ϕ_2} differ, no solution exists. For the formulas, i.e., $\phi = \phi_1 v \phi_2$, we find the satisfying states S_{ϕ_1} and S_{ϕ_2} and their corresponding optimal control policies μ_{ϕ_1} and μ_{ϕ_2} . The policy that gives rise to the highest probability of satisfying the formula from the initial state s_0 is returned as a solution.

Similar to nested P-operators, our method of finding a control policy for the complex specifications is suboptimal. That is because we use stationary and optimal algorithms for the temporal operators "bounded until" and "until," respectively, which can only return the optimal actions regardless of the bound that is specified in the P-operator.

Complexity: The overall time complexity for PCTL control synthesis for an MDP from a formula ϕ is linear in the size of the formula and the polynomial of the size of the model.

References

[1] Morteza Lahijanian, Joseph Wasniewski, Sean B. Andersson, Calin Belta. "Motion planning and control from temporal logic specifications with probabilistic satisfaction guarantees" ICRA 2010: 3227-3232

[2] Morteza Lahijanian, Sean B. Andersson, Calin Belta "Temporal Logic Motion Planning and Control With Probabilistic Satisfaction Guarantees" IEEE 2012: 396-409

[3] Igor Cizelj, Calin Belta "Negotiating the probabilistic satisfaction of temporal logic motion specifications" IROS 2013: 4320-4325

[4] Cizelj, Igor and Belta, Calin "Probabilistically Safe Control of Noisy Dubins Vehicles" IROS 2012

[5] I. Medina Ayala, Sean B. Andersson, Calin Belta. "Temporal logic control in dynamic environments with probabilistic satisfaction guarantees" IROS 2011: 3108-3113

[6] I. Medina Ayala, Sean B. Andersson, Calin Belta. "Probabilistic control from time-bounded temporal logic specifications in dynamic environments" ICRA 2012: 4705-4710

[7] M. R. NeuhauBer and L. Zhang. "Time-bounded reachability probabilities in continuous-time Markov decision processes" In Quantitative Evaluation of Systems (QEST), IEEE CS Press, pp. 209–218, 2010.

[8] Xu Chu Ding, Stephen L. Smith, Calin Belta, Daniela Rus "LTL Control in Uncertain Environments with Probabilistic Satisfaction Guarantees"

[9] Alphan Ulusoy, Tichakorn Wongpiromsarn, Calin Belta. "Incremental control synthesis in probabilistic environments with Temporal Logic constraints"

[10] Xu Chu Ding, Stephen L. Smith, Calin Belta, Daniela Rus "MDP Optimal Control under Temporal Logic Constraints"

[11] D. Bertsekas, Dynamic programming and optimal control, vol. II. Athena Scientific, 2007.

Contributions:

- Jatin Jindal (160308) - 33.333%
- Sahil Dhull (160607) - 33.333%
- Vibhor Porwal (160778) - 33.333%