2009 American Control Conference
Hyatt Regency Riverfront, St. Louis, MO, USA
June 10-12, 2009



# A Real time Implementable All-Pair Dynamic Planning Algorithm for Robot Navigation Based on the Renormalized Measure of Probabilistic Regular Languages★

**Wei Lu**‡ wxl185@psu.edu **Ishanu Chattopadhyay**‡ ixc128@psu.edu **Goutham Mallapragada**‡ grm150@psu.edu **Asok Ray**‡ axr2@psu.edu

***Abstract*— The recently reported $\nu^\star$ planning algorithm is modified to handle on-the-fly dynamic updates to the obstacle map. The modified algorithm called *All-Pair-Dynamic-Planning*(*APDP*), models the problem of robot path planning in the framework of finite state probabilistic automata and solves the all-pair planning problem in one setting. We use the concept of renormalized measure of regular languages to plan paths with automated trade-off between path length and robustness under dynamic uncertainties, from any starting location to any goal in the given map. The dynamic updating feature of *APDP* efficiently updates path plans to incorporate newly learnt information about the working environment.**

***Index Terms*— Path Planning; Language Measure; Discrete Event Systems; Dynamic Updating**

## 1. Introduction & Motivation

In recent years, dynamic updating algorithms based on graph-search methods has been researched extensively [1] [2] [3], and the current state-of-art dynamic replanning algorithm is the Field D* reported in [3]. The proposed *All-Pair-Dynamic-Planning* algorithm (noted as *APDP* in the sequel) builds on the $\nu^\star$-algorithm proposed in [4]. $\nu^\star$ is a framework that is fundamentally different from the current state of the art graph search-based algorithms. It reduces the "search" problem into finding a solution to a sequence of linear algebraic systems (i.e. matrix operations). Upon completion of cellular decomposition, $\nu^\star$ optimizes the resultant *PFSA* via an iterative sequence of combinatorial operations that maximizes the language measure vector elementwise [5][6]. The time complexity of each iteration step is linear relative to the problem size (e.g., dimension of the *PFSA* state vector). This implies significant numerical advantage over search-based methods for high-dimensional problems. *APDP* adds the feature of dynamic updating to $\nu^\star$. It performs the path planning from any starting position to any goal position, based on the renormalized language measure of the *PFSA* model [5][6]. Although the underlying navigation model is probabilistic, the *APDP*-algorithm yields path plans that can be executed in a deterministic setting with automated trade-off between the path length and robustness of computation under dynamic uncertainties. Upon detection of changes in the environment, *APDP* dynamically updates path plan in an efficient way, with computation complexity comparable with the D* algorithm [1]. *APDP*-algorithm inherits the advantages of $\nu^\star$, together with the following:

1) *Very Fast Initial Planning APDP* computes feasible path plans from any starting position to any goal position with only one matrix inverse operation with $O(N)$ run-time complexity [4]. And the results can be conveniently retrieved from the *navigation cost matrix* $\mathbf{\Lambda}$ for different goals specified. Whereas for other dynamic planning algorithm such as D*, a new plan has to be computed every time a new goal is specified.
2) *Straight-forward Dynamic Updating Strategy*: unlike the complicated data structure and cost updating operations in graph algorithms like D*, *APDP* maintains a *navigation cost matrix* $\mathbf{\Lambda}$ and uses the same equations to update all the states in one setting, thus making *APDP* easy to understand and potentially very fast

The paper is organized into eight sections including the present one. Section 2 succinctly presents the underlying concept of signed real language measure and principles of language-measure-theoretic modeling Section 3 formulates the basic problem of path planning and Section 4 derives a decision-theoretic solution to this problem. Section 5 introduces the dynamic updating strategy upon newly learnt information about the environment. Section 6 introduces an efficient algorithm for dynamic updating. Section 7 presents an examples of planar robot path planning with inaccurate map information. The paper is summarized and concluded in Section 8 with recommendations for future work.

## 2. Brief Review of Language Measure Theory

This section summarizes the signed real measure of regular languages; the details are reported in [5]. Let $G_i \equiv \langle Q, \Sigma, \delta, q_i, Q_m \rangle$ be a trim (i.e., accessible and co-accessible) finite-state automaton model that represents the discrete-event dynamics of a physical plant, where $Q = \{q_k : k \in \mathcal{I}_Q\}$ is the set of states and $\mathcal{I}_Q \equiv \{1, 2, \cdots, n\}$ is the index set of states; the automaton starts with the initial state $q_i$; the alphabet of events is $\Sigma = \{\sigma_k : k \in \mathcal{I}_\Sigma\}$, having $\Sigma \bigcap \mathcal{I}_Q = \emptyset$ and $\mathcal{I}_\Sigma \equiv \{1, 2, \cdots, \ell\}$ is the index set of events; $\delta : Q \times \Sigma \to Q$ is the (possibly partial) function of state transitions; and $Q_m \equiv \{q_{m_1}, q_{m_2}, \cdots, q_{m_l}\} \subseteq Q$ is the set of marked (i.e., accepted) states with $q_{m_k} = q_j$ for some $j \in \mathcal{I}_Q$. Let $\Sigma^*$ be the Kleene closure of $\Sigma$, i.e., the set of all finite-length strings made of the events belonging to $\Sigma$ as well as the empty string $\epsilon$ that is viewed as the identity of the monoid $\Sigma^*$ under the operation of string concatenation, i.e., $\epsilon s = s = s\epsilon$. The state transition map $\delta$ is recursively extended to its reflexive and transitive closure $\delta : Q \times \Sigma^* \to Q$ by defining $\forall q_j \in Q,\ \delta(q_j, \epsilon) = q_j$ and $\forall q_j \in Q, \sigma \in \Sigma, s \in \Sigma^\star,\ \delta(q_i, \sigma s) = \delta(\delta(q_i, \sigma), s)$

***Definition 2.1:*** *The language $L(q_i)$ generated by a DFSA $G$ initialized at the state $q_i \in Q$ is defined as: $L(q_i) = \{s \in \Sigma^* \mid \delta^*(q_i, s) \in Q\}$ The language $L_m(q_i)$ marked by the DFSA $G$ initialized at the state $q_i \in Q$ is defined as: $L_m(q_i) = \{s \in \Sigma^* \mid \delta^*(q_i, s) \in Q_m\}$*

***Definition 2.2:*** *For every $q_j \in Q$, let $L(q_i, q_j)$ denote the set of all strings that, starting from the state $q_i$, terminate at the state $q_j$, i.e., $L_{i,j} = \{s \in \Sigma^* \mid \delta^*(q_i, s) = q_j \in Q\}$*

The formal language measure is first defined for terminating plants [7] with sub-stochastic event generation probabilities *i.e.* the event generation probabilities at each state summing to strictly less than unity.

***Definition 2.3:*** *The event generation probabilities are specified by the function $\tilde{\pi} : \Sigma^\star \times Q \to [0, 1]$ such that $\forall q_j \in Q, \forall \sigma_k \in \Sigma, \forall s \in \Sigma^\star$,*

*(1) $\tilde{\pi}(\sigma_k, q_j) \triangleq \tilde{\pi}_{jk} \in [0, 1)$; $\sum_k \tilde{\pi}_{jk} = 1 - \theta$, with $\theta \in (0, 1)$;*
*(2) $\tilde{\pi}(\sigma, q_j) = 0$ if $\delta(q_j, \sigma)$ is undefined; $\tilde{\pi}(\epsilon, q_j) = 1$;*
*(3) $\tilde{\pi}(\sigma_k s, q_j) = \tilde{\pi}(\sigma_k, q_j)\ \tilde{\pi}(s, \delta(q_j, \sigma_k))$.*

*The $n \times \ell$ event cost matrix is defined as: $\widetilde{\mathbf{\Pi}}|_{ij} = \tilde{\pi}(q_i, \sigma_j)$*

***Definition 2.4:*** *The state transition probability $\pi : Q \times Q \to [0, 1)$, of the DFSA $G_i$ is defined as follows: $\forall q_i, q_j \in Q, \pi_{ij} = \sum_{\sigma \in \Sigma \text{ s.t. } \delta(q_i, \sigma) = q_j} \tilde{\pi}(\sigma, q_i)$ The $n \times n$ state transition probability matrix is defined as $\mathbf{\Pi}|_{jk} = \pi(q_i, q_j)$*

The set $Q_m$ of marked states is partitioned into $Q_m^+$ and $Q_m^-$, i.e., $Q_m = Q_m^+ \cup Q_m^-$ and $Q_m^+ \cap Q_m^- = \emptyset$, where $Q_m^+$ contains all *good* marked states

‡Department of Mechanical Engineering, The Pennsylvania State University, University Park, PA 16802

★ This work has been supported in part by the U.S. Army Research Office (ARO) under Grant No. W911NF-07-1-0376, and by the U.S. Office of Naval Research under Grant No. N00014-08-1-380.

that we desire to reach, and $Q_m^-$ contains all *bad* marked states that we want to avoid, although it may not always be possible to completely avoid the *bad* states while attempting to reach the *good* states. To characterize this, each marked state is assigned a real value based on the designer's perception of its impact on the system performance.

***Definition 2.5:*** *The characteristic function* $\chi : Q \to [-1,1]$ *that assigns a signed real weight to state-based sublanguages* $L(q_i,q)$ *is defined as:* $\forall q \in Q, \quad \chi(q) \in \begin{cases} [-1,0), & q \in Q_m^- \\ \{0\}, & q \notin Q_m \\ (0,1], & q \in Q_m^+ \end{cases}$ *The state weighting vector, denoted by* $\boldsymbol{\chi} = [\chi_1\ \chi_2\ \cdots\ \chi_n]^T$, *where* $\chi_j \equiv \chi(q_j)\ \forall j \in \mathcal{I}_Q$, *is called the* $\boldsymbol{\chi}$*-vector. The j-th element* $\chi_j$ *of* $\boldsymbol{\chi}$*-vector is the weight assigned to the corresponding terminal state* $q_j$.

In general, the marked language $L_m(q_i)$ consists of both good and bad event strings that, starting from the initial state $q_i$, lead to $Q_m^+$ and $Q_m^-$ respectively. Any event string belonging to the language $L^0 = L(q_i) - L_m(q_i)$ leads to one of the non-marked states belonging to $Q - Q_m$ and $L^0$ does not contain any one of the good or bad strings. Based on the equivalence classes defined in the Myhill-Nerode Theorem, the regular languages $L(q_i)$ and $L_m(q_i)$ can be expressed as: $L(q_i) = \bigcup_{q_k \in Q} L_{i,k}$ and $L_m(q_i) = \bigcup_{q_k \in Q_m} L_{i,k} = L_m^+ \cup L_m^-$ where the sublanguage $L_{i,k} \subseteq G_i$ having the initial state $q_i$ is uniquely labelled by the terminal state $q_k, k \in \mathcal{I}_Q$ and $L_{i,j} \cap L_{i,k} = \emptyset\ \forall j \neq k$; and $L_m^+ \equiv \bigcup_{q_k \in Q_m^+} L_{i,k}$ and $L_m^- \equiv \bigcup_{q_k \in Q_m^-} L_{i,k}$ are good and bad sublanguages of $L_m(q_i)$, respectively. Then, $L^0 = \bigcup_{q_k \notin Q_m} L_{i,k}$ and $L(q_i) = L^0 \cup L_m^+ \cup L_m^-$.

A signed real measure $\mu^i : 2^{L(q_i)} \to \mathbb{R} \equiv (-\infty, +\infty)$ is constructed on the $\sigma$-algebra $2^{L(q_i)}$ for any $i \in \mathcal{I}_Q$; interested readers are referred to [5] for the details of measure-theoretic definitions and results. With the choice of this $\sigma$-algebra, every singleton set made of an event string $s \in L(q_i)$ is a measurable set. By Hahn Decomposition Theorem [8], each of these measurable sets qualifies itself to have a numerical value based on the above state-based decomposition of $L(q_i)$ into $L^0$(null), $L^+$(positive), and $L^-$(negative) sublanguages.

***Definition 2.6:*** *Let* $\omega \in L(q_i,q_j) \subseteq 2^{L(q_i)}$. *The signed real measure* $\mu^i$ *of every singleton string set* $\{\omega\}$ *is defined as:* $\mu^i(\{\omega\}) \equiv \tilde{\pi}(\omega,q_i)\chi(q_j)$. *The signed real measure of a sublanguage* $L_{i,j} \subseteq L(q_i)$ *is defined as:* $\mu_{i,j} \equiv \mu^i(L(q_i,q_j)) = \left(\sum_{\omega \in L(q_i,q_j)} \tilde{\pi}[\omega,q_i]\right)\chi_j$

Therefore, the signed real measure of the language of a DFSA $G_i$ initialized at $q_i \in Q$, is defined as $\mu_i \equiv \mu^i(L(q_i)) = \sum_{j \in \mathcal{I}_Q} \mu^i(L_{i,j})$. It is shown in [5] that the language measure can be expressed as $\mu_i = \sum_{j \in \mathcal{I}_Q} \pi_{ij}\mu_j + \chi_i$. The language measure vector, denoted as $\mu = [\mu_1\ \mu_2\ \cdots\ \mu_n]^T$, is called the $\mu$-vector. In vector form, we have $\mu = \boldsymbol{\Pi}\mu + \boldsymbol{\chi}$ whose solution is given by

$$\boldsymbol{\mu} = (\mathbb{I} - \boldsymbol{\Pi})^{-1}\boldsymbol{\chi} = \boldsymbol{\Lambda}\boldsymbol{\chi} \tag{1}$$

The inverse, i.e. $\boldsymbol{\Lambda}$ in Eq. (1) exists for terminating plant models [7][9] because $\boldsymbol{\Pi}$ is a contraction operator [5] due to the strict inequality $\sum_j \pi_{ij} < 1$. The residual $\theta_i = 1 - \sum_j \pi_{ij}$ is referred to as the termination probability for state $q_i \in Q$. It can be derived that the $ij^{th}$ element of $\boldsymbol{\Lambda}$ is the sum of the cost of all possible transitions from $q_i$ to $q_j$, i.e.

$$\Lambda_{i,j} = \sum_{\omega \in L(q_i,q_j)} \tilde{\pi}(q_i,\omega) \tag{2}$$

Therefore, $\boldsymbol{\Lambda}$ is called the *navigation cost matrix* in the sequel.

We extend the analysis to non-terminating plants with stochastic transition probability matrices (*i.e.* with $\theta_i = 0,\ \forall q_i \in Q$) by renormalizing the language measure [5] with respect to the uniform termination probability of a limiting terminating model as described next.

Let $\widetilde{\boldsymbol{\Pi}}$ and $\boldsymbol{\Pi}$ be the stochastic event generation and transition probability matrices for a non-terminating plant $G_i = (Q,\Sigma,\delta,q_i,Q_m)$. We consider the terminating plant $G_i(\theta)$ with the same DFSA structure $(Q,\Sigma,\delta,q_i,Q_m)$ such that the event generation probability matrix is given by $(1-\theta)\widetilde{\boldsymbol{\Pi}}$ with $\theta \in (0,1)$ implying that the state transition probability matrix is $(1-\theta)\boldsymbol{\Pi}$.

***Definition 2.7: (Renormalized Measure)*** *The renormalized measure* $\nu_\theta^i : 2^{L(q_i(\theta))} \to [-1,1]$ *for the* $\theta$*-parametrized terminating plant* $G_i(\theta)$ *is defined as:*

$$\forall \omega \in L(q_i(\theta)),\ \nu_\theta^i(\{\omega\}) = \theta\mu^i(\{\omega\}) \tag{3}$$

*The corresponding matrix form is given by*

$$\boldsymbol{\nu_\theta} = \theta\ \boldsymbol{\mu} = \theta\ [I - (1-\theta)\boldsymbol{\Pi}]^{-1}\boldsymbol{\chi} \text{ with } \theta \in (0,1) \tag{4}$$

*in terms of* $\boldsymbol{\Lambda}$:

$$\boldsymbol{\nu_\theta} = \theta\boldsymbol{\Lambda}\boldsymbol{\chi} \tag{5}$$

*We note that the vector representation allows for the following notational simplification*

$$\nu_\theta^i(L(q_i(\theta))) = \boldsymbol{\nu_\theta}\big|_i \tag{6}$$

*The renormalized measure for the non-terminating plant* $G_i$ *is defined to be* $\lim_{\theta \to 0^+} \nu_\theta^i$.

The following results are retained for the sake of completeness. Complete proofs can be found in [5].

***Proposition 2.1:*** *The limiting measure vector* $\nu_0 \triangleq \lim_{\theta \to 0^+} \nu_\theta$ *exists and* $\|\nu_0\|_\infty \leq 1$.

***Proposition 2.2:*** *Let* $\boldsymbol{\Pi}$ *be the stochastic transition matrix of a finite Markov chain (or equivalently a probabilistic regular language). Then, as the parameter* $\theta \to 0^+$, *the limiting measure vector is obtained as:* $\boldsymbol{\nu}_0 = \mathscr{P}\boldsymbol{\chi}$ *where the matrix operator* $\mathscr{P} \triangleq \lim_{k \to \infty} \frac{1}{k} \sum_{j=0}^{k-1} \boldsymbol{\Pi}^j$.

***Corollary 2.1:*** *(to Proposition 2.2) The expression* $\mathscr{P}\boldsymbol{\nu}_\theta$ *is independent of* $\theta$. *Specifically, the following identity holds for all* $\theta \in (0,1)$.

$$\mathscr{P}\boldsymbol{\nu}_\theta = \mathscr{P}\boldsymbol{\chi} \tag{7}$$

## 3. Formulation of the Path Planning Problem

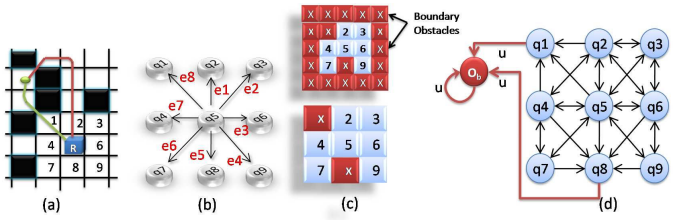


Fig. 1. Illustration of the path planning problem formulation. **(a)** shows the vehicle (marked "R") with the obstacle positions shown as black squares and the star identifies the goal. **(b)** shows the finite state representation of possible one-step moves from the current position $q_5$. **(c)** shows the "boundary obstacles" and a map with locations 1 and 8 blocked. **(d)** shows uncontrollable transitions "$\sigma_u$" from states corresponding to blocked states

Let $\mathcal{G}_{\mathsf{Nav}} = (Q,\Sigma,\delta,\widetilde{\boldsymbol{\Pi}},\boldsymbol{\chi},\mathscr{C})$ be a *PFSA* model. In the absence of dynamic uncertainties and state estimation errors, the alphabet $\Sigma$ contains only one uncontrollable event $\sigma_u$, i.e., $\Sigma = \Sigma_C \bigcup \sigma_u\}$, where $\Sigma_C$ is the set of controllable events corresponding to the possible moves of the robot. The uncontrollable event $\sigma_u$ is defined from each of the blocked states and leads to an additional state, called the obstacle state $q_\ominus$, which is a deadlock state and does not correspond to any physical grid location; all other transitions (i.e., moves) are removed from the blocked states. Thus, the state set $Q$ consists of states that correspond to grid locations and the additional deadlock state $q_\ominus$. The grid squares are numbered in a pre-determined scheme such that each $q_i \in Q \setminus \{q_\ominus\}$ denotes a specific square in the discretized workspace, where the particular numbering scheme chosen is irrelevant. Thus, if a robot moves into a blocked

state, it uncontrollably transitions to the deadlock state $q_\ominus$ which is physically interpreted to be a collision. It is also assumed that the robot fails to recover from collisions which is reflected by making $q_\ominus$ a deadlock state.

***Definition 3.1:*** *The set of blocked grid locations along with the obstacle state $q_\ominus$ is denoted as $Q_{\text{OBSTACLE}} \subsetneqq Q$.*

Figure 1 illustrates the navigation automaton for a nine-state discretized workspace with two blocked squares. Note that the only outgoing transition from the blocked states $q_1$ and $q_8$ is $\sigma_u$. The navigation $PFSA$ is augmented by specifying event generation probabilities defined by the map $\tilde{\pi} : Q \times \Sigma \to [0,1]$ and the characteristic state-weight vector specified as $\chi : Q \to [-1,1]$ that assigns scalar weights to the $PFSA$ states [6].

***Definition 3.2:*** *In the APDP setting, the characteristic weights are specified for the navigation automaton as follows:*

$$\chi(q_i) = \begin{cases} 1 & \text{if } q_i \text{ is the goal} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

In the absence of dynamic constraints and state estimation uncertainties, the robot can "choose" the particular controllable transition to execute at any grid location. Hence, the probability of generation of controllable events is assumed to be uniform over the set of moves defined at any particular state.

***Definition 3.3:*** *Under the modeling assumption that there are no uncontrollable events defined at any of the unblocked states and no controllable events defined at any of the blocked states, event generation probabilities are defined based on the following specification: $\forall q_i \in Q, \sigma_j \in \Sigma$,*

$$\tilde{\pi}(q_i,\sigma_j) = \begin{cases} \frac{1}{\text{No. of controllable events at } q_i}, & \text{if } \sigma_j \in \Sigma_C \\ 1, & \text{otherwise} \end{cases} \tag{9}$$

It is shown in the sequel that computation of path plans requires the navigation automaton to have a constant number of moves or controllable transitions from each of the unblocked states, as stated below in Definition 3.4.

***Definition 3.4:*** *The navigation model is defined to have identical connectivity as far as controllable transitions are concerned implying that every controllable transition or move (i.e., every element of $\Sigma_C$) is defined from each of the unblocked states.*

## 4. APDP Solution to the Path Planning Problem

The above-described $PFSA$-based navigation model facilitates the traversal of robot in the 2D grid. As Eq.(4) gives the language-measure of all states with regard to a goal state $q_{\text{GOAL}}$, the robot uses the following greedy strategy to move from current state to next state, until $q_{\text{GOAL}}$ is reached.

**Algorithm 1**: Computation of Next State

```
input : Current State q_CURRENT, ν_θ, δ
output: Next State q_NEXT
begin
    Set M = -INF;                          /* Large Negative Number */
    for j = 1 to CARD(Σ_C) do
        q_j = δ(q_CURRENT, σ_j);
        if ν_θ(q_j) > M then
            q_NEXT = q_j;
            M = ν_θ(q_j);
        endif
    endfor
end
```

Here we prove the above mentioned algorithm guarantees to find the $q_{\text{GOAL}}$.

***Notation 4.1:*** *For notational simplicity, we use*

$$\nu_\theta^i(L(q_i)) \triangleq \nu_(q_i) = \nu_i \tag{10}$$

*where $\boldsymbol{\nu} = \theta_{min}[I - (1-\theta_{min})\boldsymbol{\Pi}]^{-1}\boldsymbol{\chi}$*

***Definition 4.1:*** *(APDP-path:) A APDP-path $\rho(q_i,q_j)$ from state $q_i \in Q \setminus \{q_\theta\}$ to state $q_j \in Q \setminus \{q_\theta\}$ is defined to be an ordered set of PFSA states $\rho = \{q_{r_1}, \cdots, q_{r_M}\}$ with $q_{r_s} \in Q$, $\forall s \in \{1,\cdots,M\}, M \leq \text{CARD}(Q)$ such that*

$$q_{r_1} = q_i \tag{11a}$$

$$q_{r_M} = q_j \tag{11b}$$

$$\forall i,j \in \{1,\cdots,M\},\ q_{r_i} \neq q_{r_j} \tag{11c}$$

$$\forall s \in \{1,\cdots,M\}, \forall t < s,\ \nu(q_{r_t}) < \nu(q_{r_s}) \tag{11d}$$

***Lemma 4.1:*** ***(Absence of Local maxima)*** *If there exists a APDP-path from $q_i \in Q$ to $q_j \in Q$ and a APDP-path from $q_i$ to $q_{\text{GOAL}}$ then there exists a APDP-path from $q_j$ to $q_{\text{GOAL}}$, i.e.,*

$$\forall q_i,q_j \in Q\Big(\exists\rho_1(q_i,q_{\text{GOAL}}) \bigwedge \exists\rho_2(q_i,q_j) \Rightarrow \exists\rho(q_j,q_{\text{GOAL}})\Big)$$

***Proof:*** For $q_i \neq q_{\text{GOAL}}$, $\chi_i = 0$ we have

$$\nu_i = \sum_{j \in \mathcal{I}_Q} (1-\theta_{min})\pi_{ij}\nu_j \tag{12}$$

Recall that in the $APDP$ setting, $\pi_{ij} = \tilde{\pi}(q_i,\sigma_j)$, where $\delta(q_i,\sigma_j) = q_j$. From Eq. (9), we get

$$\tilde{\pi}_{ij} = \frac{1}{\text{CARD}(\Sigma_C)} \tag{13a}$$

$$\Rightarrow \nu_i = \frac{1-\theta_{min}}{\text{CARD}(\Sigma_C)} \sum_{j,\sigma_j \in \Sigma_C} \nu_j \tag{13b}$$

Since $\theta_{min} > 0$, it follows that

$$\nu_i < \max_{j,\sigma_j \in \Sigma_C} (\nu_j), \forall q_i \neq q_{\text{GOAL}} \tag{14}$$

Eq.(14) implies the absence of local maxima in language measure. It also follows that measure of $q_{\text{GOAL}}$, i.e. $\nu_{\text{GOAL}}$ is the global maximum. ❐

***Proposition 4.1:*** ***(Existence of APDP-paths:)*** *There exists a APDP-path $\rho(q_i,q_{\text{GOAL}})$ from any state $q_i \in Q$ to the goal $q_{\text{GOAL}} \in Q$ if and only if $\nu(q_i) > 0$.*

***Proof:*** Partitioning $L(q_i)$ based on terminating states,

$$\begin{aligned} \nu(q_i) = \nu(\{\omega : \delta(q_i,\omega) = q_{\text{GOAL}}\}) \\ + \nu(\{\omega : \delta(q_i,\omega) \neq q_{\text{GOAL}}\}) \end{aligned} \tag{15}$$

Since $\forall q_j \neq q_{\text{GOAL}}, \chi(q_j) = 0$, we have $\nu(\{\omega : \delta(q_i,\omega) \neq q_{\text{GOAL}}\}) = 0$. Hence we conclude

$$\nu(q_i) = \nu(\{\omega : \delta(q_i,\omega) = q_{\text{GOAL}}\}) \tag{16}$$

It follows from $\chi(q_{\text{GOAL}}) = 1$ that $\delta(q_i,\omega) = q_{\text{GOAL}} \Rightarrow \nu(\{\omega\}) > 0$ implying there exists an enabled sequence of controllable transitions from $q_i$ to $q_{\text{GOAL}}$ in $\mathcal{G}_{\text{NAV}}$ if and only if $\nu(q_i) > 0$. The result then follows from Lemma 4.1. It is also obvious that the path computed by Algorithm 1 is a $APDP$-path. ❐

***Corollary 4.1:*** ***(Obstacle Avoidance:)*** *There exists no APDP-path leading to any blocked state in the navigation automaton $\mathcal{G}_{\text{NAV}}$.*

***Proof:*** $\forall q_i \in Q_{\text{OBSTACLE}}$, the only transition defined is $\delta(q_i,\sigma_u) = q_\theta$. Therefore, according to Eq. (16), the measure of a blocked state is always zero. And it follows from Definition 4.1 that there is no $APDP$-path leading to any blocked state. ❐

## 5. Dynamic Replanning Strategy

This section proposes a dynamic updating method under $APDP$ path planning scheme. The robot scans the environment with various sensors (sonar, laser range finder, etc) in real time and compare the signal with the original map stored in its memory. If a grid location is detected blocked while in the original map it is open, then the

corresponding state is marked as a *newly blocked state*; whereas if a newly opened grid location is detected, the corresponding state is marked as a *newly opened state*. The robot updates its plan according to *newly changed states*. Instead of recalculating the whole problem again, it is more efficient to dynamically update the measure of states based on new information. First we handle *newly blocked states*, and subsequently extend the analysis to *newly opened states*.

### A. Dynamic Updates for Newly Blocked States

To facilitate the formulation of updating strategy, we define $\Gamma$ as follows:

***Definition 5.1: (Contribution to Traversal)*** *The sum of cost of all strings starting from $q_i$, passing $q_k$, and ends at $q_j$ is defined as the contribution of traversal of $q_k$, which is specified by the function $\Gamma : Q\times Q\times Q \to [0, \frac{1}{\theta}]$ such that $\forall q_i, q_k, q_j \in Q$*

$$\Gamma(i,k,j) = \sum_{\omega\in L(q_i,q_k,q_j)} \tilde{\pi}(q_i,\omega) \tag{17}$$

$\Gamma(i,k,j)$ quantifies the contribution of state $q_k$ to the cost of traversing from $q_i$ to $q_j$. Therefore it can be used to update the changes in language measure of other states. Proposition 5.1 shows the calculation of $\Gamma$ for open states:

***Proposition 5.1: (Calculation of*** $\Gamma(i,k,j)$***)*** *$\forall q_i, q_j \in Q, q_k \in Q \setminus Q_{\text{Obstacle}}$:*

$$\Gamma(i,k,j) = \begin{cases} \frac{\Lambda_{i,k}\Lambda_{k,j}}{\Lambda_{k,k}}, & j \neq k \\ \frac{\Lambda_{i,k}}{\Lambda_{k,k}}, & j = k \end{cases} \tag{18}$$

***Proof:*** To facilitate the proof, we define set of substrings as follows:

***Definition 5.2: (Set of Substrings)*** *$\forall \omega \in \Sigma^*, \omega \neq \epsilon$ , $S(\omega)$ is the set of nonempty substrings of $\omega$, i.e. $S(\omega) = \{s \neq \epsilon \mid s \text{ is a substring of } \omega\}$.*

Consider a string $\omega \in L(q_i,q_j,q_k)$. Intuitively, one would consider $\omega$ consisting of two substrings: $\{\omega_{ik}\} = S(\omega)\cap L(q_i,q_k)$ and $\{\omega_{kj}\} = S(\omega)\cap L(q_k,q_j)$. However, because APDP allows self-loops (transitions from $q_k$to $q_k$), the substring

$$\{\omega_{kk}\} = \Big(S(\omega)\cap L(q_i,q_k)\Big)\bigcap\Big(S(\omega)\cap L(q_k,q_j)\Big) = S(\omega)\cap L(q_k,q_k)$$

is counted twice. Therefore to calculate the cost of $\omega$, according to Definition 2.3, $\forall q_i, q_j \in Q, q_k \in Q\setminus Q_{\text{Obstacle}}$, and $\forall \omega \in L(q_i,q_k,q_j)$

$$\tilde{\pi}(q_i,\omega) = \frac{\tilde{\pi}(q_i,\omega_{ik})\tilde{\pi}(q_k,\omega_{kj})}{\tilde{\pi}(q_k,\omega_{kk})} \tag{19}$$

By Eq.(2), summing up the cost of all singleton strings $\omega \in L(q_i,q_k,q_j)$, and $\Gamma(i,k,j)$ is obtained in Proposition 5.1. ❒

Each time when a *newly blocked states* $q_k$ is detected, the robot updates $\mathbf{\Lambda}$ to reflect the influence if $q_k$ in language measure.

For notational clarity, we use $\mathbf{\Lambda}^{[n]}$ to denote its $n^{th}$ version, i.e. the accurate value of $\mathbf{\Lambda}$ after *n newly changed states* are discovered; we use $\tilde{\mathbf{\Lambda}}^{[n]}$ to denote an estimate of $\mathbf{\Lambda}^{[n]}$. Other quantities used in the formulation of dynamic updating (such as $\Gamma$ and $\boldsymbol{\nu}$) follow the same rules.

***Proposition 5.2: (Updates for Newly Blocked States)*** *$\forall q_i, q_j \in Q, q_k \in Q\setminus Q_{\text{Obstacle}}$, the updating equation*

$$\Lambda_{i,j}^{[n]} = \Lambda_{i,j}^{[n-1]} - \Gamma^{[n-1]}(i,k,j), \quad n = 1,2,\ldots \tag{20}$$

*gives the accurate value of the new navigation cost $\Lambda_{i,j}^{[n]}$, as if the whole problem is calculated all over again.*

***Proof:*** Since $q_k$ is a now a blocked state, $L(q_i,q_k,q_j) = \emptyset$, i.e, $q_k$ has no contribution to the cost of traversal from $q_i$ to $q_j$. The proof then follows Definition 5.1. For multiple *newly blocked states*, Eq.(20) is used iteratively. Since each iteration using Eq.(20) is accurate, it returns the accurate $\Lambda_{i,j}^{[n]}$ at the end of iterations. ❒

***Remark 5.1:*** *(to Proposition 5.2) The language measure obtained by the following equation is accurate.*

$$\boldsymbol{\nu}^{[n]} = \theta_{min}\mathbf{\Lambda}^{[n]}\boldsymbol{\chi} \tag{21a}$$

$$\text{where } \chi(q_i) = \begin{cases} 1, & \text{if } q_i \text{is the goal} \\ 0, & \text{otherwise} \end{cases} \tag{21b}$$

### B. Dynamic Updates for Newly Opened States

The situation of *newly opened states* requires more explanation. *Newly opened states* were considered as blocked, and blocked states have no contribution for the traversal from $q_i$ to $q_j, \forall q_i, q_j \in Q\setminus\{q_\theta\}$. But we are interested in their contribution after they are detected as open states. Therefore, we define potential contribution for blocked states as below:

***Definition 5.3: (Potential Contribution to Traversal)*** *$\forall q_k \in Q_{\text{Obstacle}}, \forall q_i, q_j \neq q_\theta$, the potential contribution $\Gamma'(i,k,j)$ of $q_k$ is its contribution to the traversal from $q_i$ to $q_j$, as if $q_k$ is a open state.*

***Proposition 5.3: (Estimation of*** $\Gamma'(i,k,j)$***)*** *$\forall q_i, q_j \in Q, q_k \in Q_{\text{Obstacle}}$, the equation*

$$\tilde{\Gamma}'(i,k,j) = \begin{cases} \Lambda_{i,k}\Lambda_{j,k}, & j \neq k \\ \Lambda_{i,k}, & j = k \end{cases} \tag{22}$$

*gives an under-estimate of $\Gamma'(i,k,j)$, which satisfies:*

$$\tilde{\Gamma}'(i,k,j) = \alpha\Gamma'(i,k,j), \text{where } \alpha \in (0,1] \tag{23}$$

***Proof:*** For blocked states, there is no transition to open states. However, transitions from open states to blocked states are allowed. Let $\omega_{ik} \in L(q_i,q_k)$, $\omega_{jk} \in L(q_j,q_k)$. As $q_k$ becomes open, transitions from $q_k$ to $q_j$ also becomes allowed. And a corresponding string $\omega_{kj} \in L(q_k,q_j)$ can be constructed by reversing $\omega_{jk}$. Moreover, in the setting of *APDP*, $\tilde{\pi}(\omega_{jk}) = \tilde{\pi}(\omega_{kj})$. So we can construct a string $\omega_{ikj} \in L(q_i,q_k,q_j)$ by connecting $\omega_{ik}$ and $\omega_{kj}$, and the cost of the new string is given as:

$$\tilde{\pi}(\omega_{ikj}) = \tilde{\pi}(\omega_{ik})\tilde{\pi}(\omega_{jk}) \tag{24}$$

Summing up costs of all strings constructed in this way and an estimate of $\Gamma'(i,k,j)$ is obtained in Eq.(22)

Note that the strings $\omega_{ikj} \in L(q_i,q_k,q_j)$ constructed in the above mentioned way do not contain self-loops from $q_k$ to $q_k$, i.e. $S(\omega_{ikj})\cap L(q_k,q_k) = \emptyset$. Since $q_k$ was considered blocked, as $\omega_{ik}$ and $\omega_{jk}$ reach $q_k$, the very next transition is $\delta(q_k,\sigma_u) = q_\theta$. Now that $q_k$ is considered open, and self-loops are actually allowed. So Eq.(22) only takes into account a subset of all possible strings $\omega \in L(q_i,q_k,q_j)$.

By similar arguments in the case of *newly blocked states*, the true value of $\Gamma'(i,k,j)$ is given by

$$\Gamma'(i,k,j) = \begin{cases} \Lambda_{i,k}\Lambda_{j,k}\Lambda'_{k,k}, & j \neq k \\ \Lambda_{i,k}\Lambda'_{k,k}, & j = k \end{cases} \tag{25}$$

where $\Lambda'_{k,k}$ is the sum of costs of all self-loop strings of $q_k$ if it is treated as a open state. Since $\tilde{\pi}(q_k,\epsilon) = 1$ (see Definition 2.3), $\forall q_k \in Q, \Lambda_{k,k} \geq 1$, and generally for $q_k \in Q\setminus Q_{\text{Obstacle}}, \Lambda_{k,k} > 1$.

The true value of $\Lambda'_{k,k}$ cannot be obtained in *APDP* method. So the result given by Eq.(22) is an under-estimate of $\Gamma'(i,k,j)$ as indicated by Eq.(23), where $\alpha = \frac{1}{\Lambda'_{k,k}} \in (0,1]$ (in reality, $\alpha$ is close to unity). ❒

The updates for *newly opened states* is similar to *newly blocked states*. As $q_k$ is detected as a *newly opened state*, $\mathbf{\Lambda}$ is updated by:

$$\tilde{\Lambda}_{i,j}^{[n]} = \tilde{\Lambda}_{i,j}^{[n-1]} + \tilde{\Gamma}^{[n-1]}(i,k,j), \quad n = 1,2,\ldots \tag{26}$$

and then the new language measure is given by:

$$\tilde{\boldsymbol{\nu}}^{[n]} = \theta_{min}\tilde{\mathbf{\Lambda}}^{[n]}\boldsymbol{\chi} \tag{27a}$$

$$\text{where } \chi(q_i) = \begin{cases} 1, & \text{if } q_i \text{is the goal} \\ 0, & \text{otherwise} \end{cases} \tag{27b}$$

### C. Effectiveness of Dynamic Updating

From the above analysis, we know if there only exist *newly blocked states*, the updated language is the same as recalculating the whole problem again. However, if there exist *newly opened states*, since the accurate value of $\Gamma'$ cannot be obtained for $q_k \in Q_{\text{OBSTACLE}}$, only an estimate of language measure can be calculated. Despite the possible inaccuracy in language measure, we are to show $\tilde{\boldsymbol{\nu}}^{[n]}$ obtained in the above described way preserves the key property of absence of local maxima in language measure, and hence it guarantees the robot to find *APDP* feasible path plans to the goal.

***Proposition 5.4:*** *$\forall q_i, q_j, q_k \in Q \setminus \{q_\theta\}$, $\tilde{\boldsymbol{\nu}}^{[n]}$ obtained by Eq.(20), (21), (26) and (27) preserves the property of absence of local maxima, described in Lemma 4.1.*

***Proof:*** Substituting Eq.(5) into Eq.(12), we get

$$\Lambda_{i,\text{GOAL}} = \sum_{j \in \mathcal{I}_Q} (1-\theta_{min})\pi_{ij}\Lambda_{j,\text{GOAL}}, \quad q_i \neq q_{\text{GOAL}} \tag{28}$$

Since the choice of $q_{\text{GOAL}}$ does not influence the relation described by Eq.(28), we can generalize this relation as:

$$\Lambda_{i,k} = \sum_{j \in \mathcal{I}_Q} (1-\theta_{min})\pi_{ij}\Lambda_{j,k}, \quad \forall q_i \neq q_k \tag{29}$$

Rewriting Eq.(22) using Eq.(29):

$$\tilde{\Gamma}'(i,k,l) = \Lambda_{i,k}\Lambda_{l,k} = \Lambda_{l,k} \sum_{j \in \mathcal{I}_Q} (1-\theta_{min})\pi_{ij}\Lambda_{j,k} \tag{30a}$$

$$\Rightarrow \tilde{\Gamma}'(i,k,l) = \sum_{j \in \mathcal{I}_Q} (1-\theta_{min})\pi_{ij}\tilde{\Gamma}'(j,k,l) \tag{30b}$$

Similar conclusion can be derived for the case of *newly blocked states* (Eq.20). We know that the original language measure $\boldsymbol{\nu}^{[0]}$ satisfies:

$$\nu_i^{[0]} = \sum_{j \in \mathcal{I}_Q} (1-\theta_{min})\pi_{ij}\nu_j^{[0]}, \quad q_i \neq q_{\text{GOAL}} \tag{31}$$

Therefore, it is easy to derive by induction that at each update using Eq.(20), (21), (26) and (27), the following relation is preserved:

$$\tilde{\nu}_i^{[n]} = \sum_{j \in \mathcal{I}_Q} (1-\theta_{min})\pi_{ij}\tilde{\nu}_j^{[n]}, \quad n = 1, 2, \ldots \tag{32}$$

Following the same arguments in Lemma 4.1 we know:

$$\tilde{\nu}_i^{[n]} < \max_{j, \sigma_j \in \Sigma_C} (\tilde{\nu}_j^{[n]}), \quad \forall q_i \neq q_{\text{GOAL}} \tag{33}$$

Hence the absence of local maxima. ❐

## 6. Algorithms for Implementation of Dynamic Replanning

This section introduces the implementation of dynamic replanning strategy described in Section 5. A fast online re-planning algorithm that is comparable with the current state of the art grid based 2D planning algorithms is derived. The key to dynamic updating is to keep $\boldsymbol{\Lambda}$ up to date. We can simply update every element of $\boldsymbol{\Lambda}$ each time a *newly changed state* is detected, but that takes $O(N^2)$ time, where $N$ is the number of states. Since the language measure with regard to $q_{\text{GOAL}}$ is derived from the $G^{th}$ column of $\boldsymbol{\Lambda}$ (see Eq.(21) and (27)), we only need to make sure $\Lambda_{i,\text{GOAL}}, i \in \mathcal{I}_Q$ is correctly updated each time. And we delay the updating for other elements in $\boldsymbol{\Lambda}$ until needed. Suppose $q_{km}$ is the newest *changed state*, and $(m-1)$ *newly changed states* $q_{k1}, q_{k2}, \ldots q_{k(m-1)}$ are detected prior to $q_{km}$. By Eq.(18), (22), (20) and (26), $\Lambda_{i,km}^{[m-1]}, i \in \mathcal{I}_Q$ is needed to update $\Lambda_{i,\text{GOAL}}^{[m]}$. Since we delayed the updates for $q_{km}$, only $\Lambda_{i,km}^{[0]}, i \in \mathcal{I}_Q$ are available. Therefore we update $\Lambda_{i,km}, i \in \mathcal{I}_Q$ with regard to the $(m-1)$ *newly changed states* detected prior to $q_{km}$. Algorithm 2 uses the above mentioned lazy approach to update $\boldsymbol{\Lambda}$. A list of all *newly changed states* is maintained. When a *newly changed state* $q_{km}$ is detected, it is added to the end of the list, and Algorithm 2 is called. Algorithm 2 "catches up with" the update for $q_{km}$ from Line 5 to Line 17. For the $m^{th}$ *changed state*, the 'For' loop from Line 5 to Line 17 run $(m-1)$ times, each loop updates $N$ elements of $\boldsymbol{\Lambda}$, and takes $O(N)$ time; the update for $q_{\text{GOAL}}$ from Line 18 to Line 27 runs once, and it takes $O(N)$ time. So for $m^{th}$ *changed state*, the running time of Algorithm 2 is $O(mN)$. Since generally $m \ll N$, Algorithm 2 is significantly faster than updating every element of $\boldsymbol{\Lambda}$. Algorithm 3 summarizes the execution of *APDP* method.

**Algorithm 2**: Dynamic Updating of $\boldsymbol{\Lambda}$

**input** : $\boldsymbol{\Lambda}_{\text{LAST}}$, list of newly changed states *qChange*()
**output**: $\boldsymbol{\Lambda}_{\text{NEW}}$

1 **begin**
2 $N = size(\Lambda_{\text{LAST}})$;
3 $n_1 = length(qChange)$;
4 $q_k = qChange(n_1)$ ; /* Newest changed state */ ; /* Lazy update for $q_k$ with regard to previously changed states */
5 **for** $n_2 \leftarrow 1$ **to** $(n_1 - 1)$ **do**
6 $q_l = qChange(n_2)$;
7 **if** $q_l$ *is a* newly blocked state **then**
8 **for** $i \leftarrow 1$ **to** $N$ **do**
9 $\Lambda_{\text{NEW}}(i,k) = \Lambda_{\text{LAST}}(i,k) - \frac{\Lambda_{\text{LAST}}(i,l)\Lambda_{\text{LAST}}(l,k)}{\Lambda_{\text{LAST}}(l,l)}$;
10 **endfor**
11 **else if** $q_l$ *is a* newly opened state **then**
12 **for** $i \leftarrow 1$ **to** $N$ **do**
13 $\Lambda_{\text{NEW}}(i,k) = \Lambda_{\text{LAST}}(i,k) + \Lambda_{\text{LAST}}(i,l)\Lambda_{\text{LAST}}(k,l)$;
14 **endfor**
15 **endif**
16 $\boldsymbol{\Lambda}_{\text{LAST}} = \boldsymbol{\Lambda}_{\text{NEW}}$;
17 **endfor** ; /* Update for $q_{\text{GOAL}}$, i.e. $\Lambda_{i,G}^{[n-1]}$ to $\Lambda_{i,G}^{[n]}, i \in \mathcal{I}_Q$ */
18 **if** $q_k$ *is a* newly blocked state **then**
19 **for** $i \leftarrow 1$ **to** $N$ **do**
20 $\Lambda_{\text{NEW}}(i,G) = \Lambda_{\text{LAST}}(i,G) - \frac{\Lambda_{\text{LAST}}(i,k)\Lambda_{\text{LAST}}(k,G)}{\Lambda_{\text{LAST}}(k,k)}$;
21 **endfor**
22 **else if** $q_k$ *is a* newly opened state **then**
23 **for** $i \leftarrow 1$ **to** $N$ **do**
24 $\Lambda_{\text{NEW}}(i,G) = \Lambda_{\text{LAST}}(i,G) + \Lambda_{\text{LAST}}(i,k)\Lambda_{\text{LAST}}(G,k)$;
25 **endfor**
26 **endif**
27 **end**

## 7. An Example of 2D Dynamic Path Planning

Let a workspace be defined by a $10 \times 10$ grid as shown in Figure 2(a). The goal is labeled and prominently marked with a dot. The obstacles, i.e., states in the set $Q_{\text{OBSTACLE}}$ (See Section 3), are illustrated as blocked-off grid locations in black while the navigable space is shown in white. From the formulation presented in Section 3, it follows that each grid location $(i, j)$ is mapped to a state in the constructed navigation automaton $\mathcal{G}_{\text{NAV}}$ which has a total of $10 \times 10 + 1 = 101$ states. State numbers are encoded by first moving along rows and then along columns, i.e.,

$$\text{state\_num} = \text{column\_num} + (\text{row\_num} - 1) \times 10$$

The $101^{st}$ state $q_{101}$ is the special obstacle state $q_\ominus$ defined in Section 3. We note that since (3,6) is the goal (See Figure 2(a)), we have $\chi(6 + (3-1) \times 10) = \chi(26) = 1$. The robot follows the procedure illustrated in Algorithm 3 to plan paths and execute the movements. In the example shown in Fig. 2 (b) to (d), color gradient indicates the measure of each state. Note in Section 4 it is shown the measure of blocked state is exactly zero. Here we set $\nu(q) = -2\theta_{min}, q \in Q_{\text{OBSTACLE}}$ to show the blocks more clearly in the figure (the true measures of $q \in Q_{\text{OBSTACLE}}$ are stored in $\boldsymbol{\Lambda}$ as $\Lambda(q, q_{\text{GOAL}})$, and are used in dynamic

**Algorithm 3**: *APDP* Algorithm for Path Planning

**input** : Map, Goal Location
**output**: *APDP*-path to Goal

**begin**
  Discretize map;
  Generate $\mathcal{G}_{\text{NAV}}$;
  Compute $\mathbf{\Lambda}^{[0]}, \nu^{[0]}$;
  $\nu_{\text{NEW}} = \nu^{[0]}$;
  $qChange() \leftarrow \emptyset$;
  Get current state $q_{\text{CURRENT}}$;
  **if** $\nu(q_{\text{CURRENT}}) = 0$ **then**
    Goal is unreachable ;
    **Abort**;
  **else**
    count = 1;
    Set Path(count) = $q_{\text{CURRENT}}$;
    **while** $q_{\text{CURRENT}} \neq q_{\text{GOAL}}$ **do**
      count = count + 1;
      $q_{\text{NEXT}}$ = Algorithm 1($q_{\text{CURRENT}}, \nu_{\text{NEW}}$);
      Go to $q_{\text{NEXT}}$;
      Set $q_{\text{CURRENT}} = q_{\text{NEXT}}$;
      Scan environment for map changes;
      **if** *any state* $q_k$ *is changed* **then**
        Add $q_k$ to the end of $qChange()$;
        $\mathbf{\Lambda}_{\text{NEW}}$ = Algorithm 2($\mathbf{\Lambda}, qChange$);
        $\nu_{\text{NEW}} = \theta_{min}\mathbf{\Lambda}_{\text{NEW}}\chi$
      **endif**
      Save Path(count) = $q_{\text{CURRENT}}$;
    **endw**
  **endif**
**end**

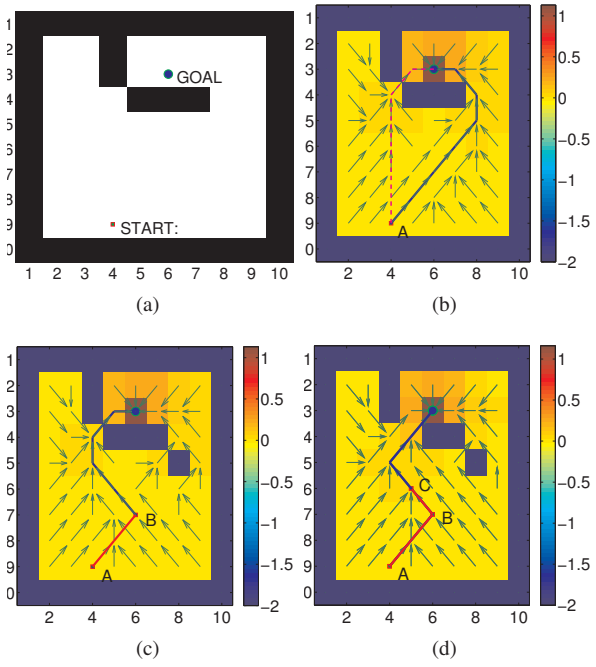


Fig. 2. Illustration of 2-D path planning. (a) shows a goal (GOAL) and the start location (Point A). (b) shows the gradient induced by *APDP* in the workspace. The initial path plan from A to GOAL given by the *APDP*-algorithm (solid blue line) is compared with the shortest path (dashed purple line) (c) When the robot reaches point B (grid location (6,7)), a new obstacle is detected at grid location (8,5). It updates the path plan with *APDP*-algorithm dynamically from B to GOAL (shown in solid blue line) (d)With the updated plan, the robot reaches point C (grid location (6,5)), where it detects grid location (4,5) is open. *APDP*-algorithm dynamically updates the path plan from C to GOAL to take advantage of this new opening (shown in solid blue line)

updating). The following observations are made on the theoretical results derived in Section 4 and Section 5.

1) **Absence of local maxima:** it is clearly shown by the vector field in Fig. 2 (b) to (d). The only local maximum (grid location with only incoming arrows) is the GOAL. And this property is held after dynamic updating, as shown in Fig. 2(c) and Fig. 2(d).
2) **Existence of *APDP* path and obstacle avoidance:** There exists no *APDP*-path (See Definition 4.1) from any navigable state to any blocked grid location (i.e., to any state in $Q_{\text{OBSTACLE}}$). And by following Algorithm 1, it is impossible to run into obstacles.
3) **Robustness of *APDP* path:** in Fig. 2(a), the robot avoids the shortest path (purple dashed line) because it passes through the narrow single-path corridor on the left, and is more risky. Instead, it takes a longer but more robust path through the open area on the right. Whereas in Fig. 2(c), after detecting a new obstacle at grid point (8,5), the robustness benefits of navigating through the right area no longer offset the benefits of shorter paths by navigating through the left area.
4) **Effectiveness of dynamic updating:** as shown in Fig. 2(c) and Fig. 2(d), the measure of all states are updated upon detection of *newly changed states*. The dynamically updated measure maintains all the properties mentioned above.

## 8. Summary & Future Research

A novel path planning algorithm *APDP* is introduced that adds dynamic updating feature to language measure-based path planning algorithms. The resulting algorithm calculates paths with automated trade-off between path length and robustness between all pairs of locations in the given map. The dynamic updating for changed environment is efficient, easy to understand and implement. The current dynamic updating updates the measure of every state in the map. The efficiency can be further improved by focusing the updating on states that are relevant with navigation. Proper heuristic is needed to focus the updates and an incremental updating strategy will be developed in the future.

## References


[1] A. T. Stentz, "Optimal and efficient path planning for partially-known environments," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '94)*, vol. 4, pp. 3310 – 3317, May 1994.

[2] S. Koenig and M. Likhachev, "D*lite," *Eighteenth national conference on Artificial intelligence*, pp. 476–483, 2002.

[3] D. Ferguson and A. T. Stentz, "The field d* algorithm for improved path planning and replanning in uniform and non-uniform cost environments," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-05-19, June 2005.

[4] I. Chattopadhyay, G. Mallapragada, and A. Ray, "$\nu^\star$ : a robot path planning algorithm based on renormalized measure of probabilistic regular languages," *International Journal of Control*, in press.

[5] I. Chattopadhyay and A. Ray, "Renormalized measure of regular languages," *Int. J. Control*, vol. 79, no. 9, pp. 1107–1117, 2006.

[6] ——, "Language-measure-theoretic optimal control of probabilistic finite-state systems," *Int. J. Control*, vol. 80, no. 8, pp. 1271–1290, 2007.

[7] V. Garg, "An algebraic approach to modeling probabilistic discrete event systems," *Proceedings of 1992 IEEE Conference on Decision and Control*, pp. 2348–2353, Tucson, AZ, December 1992.

[8] W. Rudin, *Real and Complex Analysis, 3rd ed.* McGraw Hill, New York, 1988.

[9] V. Garg, "Probabilistic languages for modeling of DEDs," *Proceedings of 1992 IEEE Conference on Information and Sciences*, pp. 198–203, Princeton, NJ, March 1992.