International Journal of Control
Vol. 82, No. 5, May 2009, 849–867

Taylor & Francis
Taylor & Francis Group

$v^{\star}$: a robot path planning algorithm based on renormalised measure of probabilistic regular languages

Ishanu Chattopadhyay, Goutham Mallapragada and Asok Ray*

Department of Mechanical Engineering, Pennsylvania State University, University Park, PA, USA

(*Received 30 November 2007; final version received 14 July 2008*)

This article introduces a novel path planning algorithm, called $v^{\star}$, that reduces the problem of robot path planning to optimisation of a probabilistic finite state automaton. The $v^{\star}$-algorithm makes use of renormalised measure v of regular languages to plan the optimal path for a specified goal. Although the underlying navigation model is probabilistic, the $v^{\star}$-algorithm yields path plans that can be executed in a deterministic setting with automated optimal trade-off between path length and robustness under dynamic uncertainties. The $v^{\star}$-algorithm has been experimentally validated on Segway Robotic Mobility Platforms in a laboratory environment.

Keywords: path planning; language measure; discrete event systems; supervisory control

1. Introduction and motivation

The field of trajectory and motion planning is enormous, with applications in diverse areas such as industrial robots, mobile robot navigation, spacecraft re-entry and video games. Many of the basic concepts are presented in Latombe's book (1991) and in a recent book by LaValle (2006). In the context of planning for mobile robots and manipulators much of the literature on path and motion planning is concerned with finding collision-free trajectories (Kondo 1991). A great deal of the complexity in these problems arises from the topology of the robot's configuration space, called the $\mathcal{C}$-Space. Various analytical techniques, such as wave-front expansion and cellular decomposition, have been reported in recent literature (Lozano-Perez 1987), which partition the $\mathcal{C}$-Space into a finite number of regions with the objective of reducing the motion planning problem as identification of a sequence of neighbouring cells between the initial and final (i.e. goal) regions. Graph-theoretic techniques have been successfully applied to path planning of wheeled ground robots and also used for solving radar evasion problems associated with unmanned aerial vehicles (UAVs) (Anisi, Hamberg and Hu 2003). In general, these path planning tools suffer from exponential growth in complexity with both graph size and dimension. To circumvent the complexity associated with graph-based planning, sampling based planning methods, such as probabilistic road maps (Barraquand, Langlois and Latombe 1990), have been widely used. These tools are probabilistically complete, i.e. a feasible solution could eventually be found if it exists. However, there is no guarantee of finding a solution within a specified time.

This article introduces and experimentally validates a novel path planning algorithm that formulates the navigation problem in the framework of probabilistic finite state automata (PFSA) from a control-theoretic perspective. The key contribution of the article is to be interpreted in the following light:

> A framework for robot path planning is reported, which is fundamentally different from the current state-of-the-art in the sense that it potentially leads to a new breed of high-performance planners rather than just formulating an algorithm designed to outperform the current planners for specific examples.

The proposed path planning algorithm, called $v^{\star}$, performs the path planning for a specified goal by optimisation of the renormalised language measure v of the PFSA model (Chattopadhyay and Ray 2006, 2007). Although the underlying navigation model is probabilistic, the $v^{\star}$-algorithm yields path plans that can be executed in a deterministic setting with automated optimal trade-off between the path length and robustness of computation under dynamic uncertainties. The advantages of the $v^{\star}$-algorithm are delineated below.

(1) *Computationally inexpensive pre-processing*: Conventional cellular approaches decompose the set of free configurations into simple non-overlapping regions (Lozano-Perez 1987;

*Corresponding author. Email: axr2@psu.edu

ISSN 0020–7179 print/ISSN 1366–5820 online

DOI: 10.1080/00207170802343196
http://www.informaworld.com

Kondo 1991) with the adjacency relations and reachability constraints, represented in a connectivity graph that is subsequently searched for a path. Conventional construction of the connectivity graph that faithfully reflects the continuous $\mathcal{C}$-Space constraints (e.g. obstacles) is expensive and often intractable for large-dimensional problems. In contrast, the decomposition in the $\boldsymbol{v}^\star$-algorithm is relatively simple and computationally inexpensive. The cells are mapped to PFSA states that are defined to have identical connectivity via symbolic inter-state transitions. Desirable or 'good' states are assigned positive weights relative to undesirable or 'bad' states that may have negative weights. Therefore, a change in the $\mathcal{C}$-Space constraints is realised by simply updating the one-dimensional state weight vector without recomputing the original decomposition.

(2) *Conceptually different from a search algorithm*: The $\boldsymbol{v}^\star$-algorithm reduces the 'search' problem into finding a solution to a sequence of linear algebraic systems (i.e. matrix operations). Upon completion of cellular decomposition, $\boldsymbol{v}^\star$ optimises the resultant PFSA via an iterative sequence of combinatorial operations that maximises the language measure vector elementwise (Chattopadhyay and Ray 2006, 2007). Although $\boldsymbol{v}^\star$ involves probabilistic reasoning, the final waypoint sequence is obtained in a deterministic setting. The intensive step in $\boldsymbol{v}^\star$ is generation of a unique solution of specialised matrix problems to compute the language measure $\mathbf{v}$. 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.

(3) *Global monotonicity:* The solution iterations are finite and globally monotonic. The final waypoint sequence is generated essentially by following the measure gradient which is maximised at the goal. The measure gradient, shown later in Figure 8 as colour variations, is reminiscent of potential field methods (Barraquand et al. 1990). However, $\boldsymbol{v}^\star$ automatically generates the measure gradient and no potential function is necessary. It is noteworthy that the potential-function-based path planning may become trapped in local minima, which is a mathematical impossibility for $\boldsymbol{v}^\star$.

The article is organised into nine sections including the present one. Section 2 succinctly presents the underlying principles of language-measure-theoretic modelling and optimal control of PFSA. Section 3 formulates the basic problem of path planning and §4 derives a decision-theoretic solution to this problem. Section 5 presents examples of planar robot path planning with different dynamic constraints. Section 6 analyses the computational complexity of the $\boldsymbol{v}^\star$-algorithm. Some critical issues pertaining to dynamic replanning and high-dimensional planning scenarios are discussed in §7. Experimental results on Segway Robotic Mobility Platforms (RMPs) are presented in §8. The article is summarised and concluded in §9 with recommendations for future work.

2. Brief review of language measure theory and optimisation

This section first summarises the signed real measure of regular languages; the details are reported in Ray (2005) and Chattopadhyay and Ray (2006). Then, it introduces the concept of optimal control of PFSA (Chattopadhyay and Ray 2007).

Let $G_i = (Q, \Sigma, \delta, q_i, Q_m)$ be a trim (i.e. accessible and co-accessible) finite-state automaton (FSA) 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 $\mathcal{I}_Q \equiv \{1, 2, \ldots, n\}$ 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 $\Sigma \cap Q = \emptyset$ and $\mathcal{I}_\Sigma \equiv \{1, 2, \ldots, \ell\}$ is the index set of events; $\delta : Q \times \Sigma \to Q$ is the (possibly partial) function of state transitions; and $Q_m \equiv \{q_{m_1}, q_{m_2}, \ldots, q_{m_l}\} \subseteq Q$ is the set of marked (i.e. accepted) states with $q_{m_k} = q_j$ for some $j \in \mathcal{I}_Q$. Let Σ^* be the Kleene closure of Σ, i.e. the set of all finite-length strings made of the events belonging to Σ as well as the empty string ϵ that is viewed as the identity of the monoid Σ^* under the operation of string concatenation, i.e. $\epsilon s = s = s\epsilon$. The state transition map δ is recursively extended to its reflexive and transitive closure $\delta : Q \times \Sigma^* \to Q$ by defining

$$\forall q_j \in Q, \quad \delta(q_j, \epsilon) = q_j \tag{1a}$$

$$\forall q_j \in Q, \sigma \in \Sigma, s \in \Sigma^\star, \quad \delta(q_i, \sigma s) = \delta(\delta(q_i, \sigma), s) \tag{1b}$$

Definition 2.1: The language $L(q_i)$ generated by a DFSA G initialised at the state $q_i \in Q$ is defined as:

$$L(q_i) = \{s \in \Sigma^* | \delta^*(q_i, s) \in Q\} \tag{2}$$

The language $L_m(q_i)$ marked by the DFSA G initialised at the state $q_i \in Q$ is defined as:

$$L_m(q_i) = \{s \in \Sigma^* | \delta^*(q_i, s) \in Q_m\} \tag{3}$$

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.

$$L_{i,j} = \{s \in \Sigma^* | \delta^*(q_i, s) = q_j \in Q\} \tag{4}$$

2.1 *Formal language measure for terminating plants*

The formal language measure is first defined for terminating plants (Garg 1992) 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 $\tilde{\pi}: Q \times \Sigma^\star \to [0, 1]$ such that $\forall q_j \in Q$, $\forall \sigma_k \in \Sigma$, $\forall s \in \Sigma^\star$,

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

The $n \times \ell$ event cost matrix $\widetilde{\mathbf{\Pi}}$ 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:

$$\forall q_i, q_j \in Q, \quad \pi_{ij} = \sum_{\sigma \in \Sigma \text{ s.t. } \delta(q_i, \sigma) = q_j} \tilde{\pi}(q_i, \sigma) \tag{5}$$

The $n \times n$ state transition probability matrix $\mathbf{\Pi}$ is defined as $\mathbf{\Pi}|_{ij} = \pi(q_i, q_j)$.

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 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 characterise 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:

$$\forall q \in Q, \quad \chi(q) \in \begin{cases} [-1, 0), & q \in Q_m^- \\ \{0\}, & q \notin Q_m \\ (0, 1], & q \in Q_m^+ \end{cases} \tag{6}$$

The state weighting vector, denoted by $\chi = [\chi_1\ \chi_2 \cdots \chi_n]^{\mathrm{T}}$, where $\chi_j \equiv \chi(q_j)$ $\forall j \in \mathcal{I}_Q$, is called the χ-vector. The j-th element χ_j of χ-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 $L^0(q_i) = L(q_i) - L_m(q_i)$ 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 (Hopcroft, Motwani, and Ullman 2001), the regular languages $L(q_i)$ and $L_m(q_i)$ can be expressed as:

$$L(q_i) = \bigcup_{q_k \in Q} L_{i,k} \tag{7}$$

$$L_m(q_i) = \bigcup_{q_k \in Q_m} L_{i,k} = L_m^+ \cup L_m^-, \tag{8}$$

where the sublanguage $L_{i,k} \subseteq L(q_i)$ having the initial state q_i is uniquely labelled by the terminal state q_k, $k \in \mathcal{I}_Q$ and $L_{i,j} \cap \mathrm{L}_{i,k} = \emptyset$ $\forall j \neq k$; and $L_m^+ \equiv \cup_{q_k \in Q_m^+} L_{i,k}$ and $L_m^- \equiv \cup_{q_k \in Q_m^-} L_{i,k}$ are good and bad sublanguages of $L_m(q_i)$, respectively. Then, $L^0 = \cup_{q_k \notin Q_m} L_{i,k}$ 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 σ-algebra $2^{L(q_i)}$ for any $i \in \mathcal{I}_Q$; interested readers are referred to Ray (2005) for the details of measure-theoretic definitions and results. With the choice of this σ-algebra, every singleton set made of an event string $s \in L(q_i)$ is a measurable set. By Hahn decomposition theorem (Rudin 1988); 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 μ^i of every singleton string set $\{\omega\}$ is defined as:

$$\mu^i(\{\omega\}) = \tilde{\pi}(q_i, \omega)\chi(q_j). \tag{9}$$

The signed real measure of a sublanguage $L_{i,j} \subseteq L(q_i)$ is defined as:

$$\mu_{i,j} = \mu^i(L(q_i, q_j)) = \left(\sum_{\omega \in L(q_i, q_j)} \tilde{\pi}(q_i, \omega) \right) \chi_j. \tag{10}$$

Therefore, the signed real measure of the language of a DFSA G_i initialised at $q_i \in Q$, is defined as:

$$\mu_i = \mu^i(L(q_i)) = \sum_{j \in \mathcal{I}_Q} \mu^i(L_{i,j}). \tag{11}$$

It is shown in Ray (2005) that the language measure in Equation (11) can be expressed as:

$$\mu_i = \sum_{j \in \mathcal{I}_Q} \pi_{ij}\mu_j + \chi_i. \tag{12}$$

The language measure vector, denoted as $\boldsymbol{\mu} = [\mu_1 \; \mu_2 \cdots \mu_n]^{\mathrm{T}}$, is called the $\boldsymbol{\mu}$-vector. In vector form, Equation (12) becomes

$$\boldsymbol{\mu} = \boldsymbol{\Pi}\boldsymbol{\mu} + \boldsymbol{\chi} \tag{13}$$

whose solution is given by

$$\boldsymbol{\mu} = (\mathbf{I} - \boldsymbol{\Pi})^{-1}\boldsymbol{\chi}. \tag{14}$$

The inverse in Equation (14) exists for terminating plant models (Garg 1992a, b) because $\boldsymbol{\Pi}$ is a contraction operator (Ray 2005) 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 $q_i \in Q$. 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 renormalising the language measure (Chattopadhyay and Ray 2006) 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, respectively, 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 (Renormalised measure)**:** The renormalised measure $\nu_\theta^i : 2^{L(q_i(\theta))} \to [-1, 1]$ for the θ-parameterised terminating plant $G_i(\theta)$ is defined as:

$$\forall \omega \in L(q_i(\theta)), \;\; \nu_\theta^i(\{\omega\}) = \theta\mu^i(\{\omega\}). \tag{15}$$

The corresponding matrix form is given by

$$\boldsymbol{\nu}_\theta = \theta \;\; \boldsymbol{\mu} = \theta\,[\mathbf{I} - (1-\theta)\boldsymbol{\Pi}]^{-1}\boldsymbol{\chi} \;\; \text{with}\; \theta \in (0,1). \tag{16}$$

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

$$\nu_\theta^i(L(q_i(\theta))) = \boldsymbol{\nu}_\theta\big|_i. \tag{17}$$

The renormalised 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 (Chattopadhyay and Ray 2006).

Proposition 2.1: *The limiting measure vector* $\mathbf{v}_0 \triangleq \lim_{\theta \to 0^+} \boldsymbol{\nu}_\theta$ *exists and* $\|\mathbf{v}_0\|_\infty \leq 1$.

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*: $\mathbf{v}_0 = \mathscr{P}\boldsymbol{\chi}$ *where the matrix operator* $\mathscr{P} \triangleq \lim_{k \to \infty} (1/k) \sum_{j=0}^{k-1} \boldsymbol{\Pi}^j$.

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

$$\mathscr{P}\boldsymbol{\nu}_\theta = \mathscr{P}\boldsymbol{\chi}. \tag{18}$$

2.2 *Event-driven supervision of PFSA*

Plant models considered in this article are *deterministic* finite state automata (plant) with well-defined event occurrence *probabilities*. In other words, the occurrence of events is probabilistic, but the state at which the plant ends up, *given a particular event has occurred*, is deterministic. No emphasis is laid on the initial state of the plant and it is assumed that the plant may start from any state. Furthermore, having defined the characteristic state weight vector $\boldsymbol{\chi}$, it is not necessary to specify the set of marked states, because if $\chi_i = 0$, then q_i is not marked and if $\chi_i \neq 0$, then q_i is marked.

Definition 2.8 (Control philosophy)**:** If $q_i \xrightarrow{\sigma} q_k$, and the event σ is disabled at state q_i, then the supervisory action is to prevent the plant from making a transition to the state q_k, by forcing it to stay at the original state q_i. Thus disabling any transition σ at a given state q results in deletion of the original transition and appearance of the self-loop $\delta(q, \sigma) = q$ with the occurrence probability of σ from the state q remaining unchanged in the supervised and unsupervised plants.

Definition 2.9 (Controllable transitions)**:** For a given plant, transitions that can be disabled in the sense of Definition 2.8 are defined to be controllable transitions. The set of controllable transitions in a plant is denoted $\mathcal{C}$. Note controllability is state-based.

It follows from the above definitions that plant models can be completely specified by a sextuple as:

$$G = (Q, \Sigma, \delta, \widetilde{\boldsymbol{\Pi}}, \boldsymbol{\chi}, \mathcal{C}). \tag{19}$$

2.3 *The optimal supervision problem: formulation and solution*

A supervisor disables a subset of the set $\mathcal{C}$ of controllable transitions and hence there is a bijection between the set of all possible supervision policies and the power set $2^{\mathcal{C}}$. That is, there exists $2^{|\mathcal{C}|}$ possible supervisors and each supervisor is uniquely identifiable with a subset of $\mathcal{C}$ and the language measure $\boldsymbol{\nu}$ allows a quantitative comparison of different policies.

Definition 2.10: For an unsupervised plant $G = (Q, \Sigma, \delta, \widetilde{\boldsymbol{\Pi}}, \boldsymbol{\chi}, \mathcal{C})$, let $G^\dagger$ and $G^\ddagger$ be the supervised plants with sets of disabled transitions, $\mathscr{D}^\dagger \subseteq \mathcal{C}$ and $\mathscr{D}^\ddagger \subseteq \mathcal{C}$ whose measures are $\mathbf{v}^\dagger$ and $\mathbf{v}^\ddagger$, respectively.

Then, the supervisor that disables $\mathscr{D}^\dagger$ is defined to be superior to the supervisor that disables $\mathscr{D}^\ddagger$ if $\mathbf{v}^\dagger \geqq_{\text{(Elementwise)}} \mathbf{v}^\ddagger$ and strictly superior if $\mathbf{v}^\dagger >_{\text{(Elementwise)}} \mathbf{v}^\ddagger$.

Definition 2.11 (Optimal supervision problem)**:** Given a (non-terminating) plant $G = (Q, \Sigma, \delta, \widetilde{\mathbf{\Pi}}, \boldsymbol{\chi}, \mathcal{C})$, the problem is to compute a supervisor that disables a subset $\mathscr{D}^\star \subseteq \mathcal{C}$, such that $\mathbf{v}^\star \geqq_{\text{(Elementwise)}} \mathbf{v}^\dagger \ \forall \mathscr{D}^\dagger \subseteq \mathcal{C}$, where $\mathbf{v}^\star$ and $\mathbf{v}^\dagger$ are the measure vectors of the supervised plants $G^\star$ and $G^\dagger$ under $\mathscr{D}^\star$ and $\mathscr{D}^\dagger$, respectively.

Remark 2.1: The solution to the optimal supervision problem is obtained in Chattopadhyay and Ray (2007) by designing an optimal policy for a terminating plant (Garg 1992a, b) with a substochastic transition probability matrix $(1-\theta)\widetilde{\mathbf{\Pi}}$ with $\theta \in (0,1)$. To ensure that the computed optimal policy coincides with the one for $\theta = 0$, the suggested algorithm chooses a small value for θ in each iteration step of the design algorithm. However, choosing θ too small may cause numerical problems in convergence. Algorithm 1 computes the critical lower bound $\theta_\star$ (i.e. how small a θ is actually required). In conjunction with Algorithm 1, the optimal supervision problem is solved by use of Algorithm 2 for a generic PFSA as reported in Chattopadhyay and Ray (2007).

The following results in Propositions 2.3–2.6 are critical to development of the autonomous navigation algorithm in the sequel and hence are presented here without proof for the sake of brevity. The complete proofs are available in Chattopadhyay and Ray (2007).

Proposition 2.3: *Let* $\mathbf{v}^{[k]}$ *be the language measure vector computed in the k-th iteration of Algorithm* 2. *The measure vectors computed by the algorithm form an elementwise non-decreasing sequence*, i.e. $\mathbf{v}^{[k+1]} \geqq_{\text{(Elementwise)}} \mathbf{v}^{[k]} \ \forall k$.

Proposition 2.4 (Effectiveness)**:** *Algorithm* 2 *is an effective procedure* (Hopcroft et al. 2001), *i.e. it is guaranteed to terminate*.

Proposition 2.5 (Optimality)**:** *The supervision policy computed by Algorithm* 2 *is optimal in the sense of Definition* 2.11.

Proposition 2.6 (Uniqueness)**:** *Given an unsupervised plant G, the optimal supervisor* $G^\star$, *computed by Algorithm 2, is unique in the sense that it is maximally permissive among all possible supervision policies with optimal performance. That is, if* $\mathscr{D}^\star$ *and* $\mathscr{D}^\dagger$ *are the disabled transition sets, and* $\mathbf{v}^\star$ *and* $\mathbf{v}^\dagger$ *are the language measure vectors for* $G^\star$ *and an arbitrarily supervised plant* $G^\dagger$, *respectively, then* $\mathbf{v}^\star =_{\text{(Elementwise)}} \mathbf{v}^\dagger \Longrightarrow \mathscr{D}^\star \subset \mathscr{D}^\dagger \subseteq \mathcal{C}$.

Algorithm 1: Computation of the Critical Lower Bound $\theta_\star$

```
input : P, χ
output: θ*
begin
  Set θ* = 1;
  Set θ_curr = 0;
  Compute 𝒫 ;            /* Stable Prob. Dist. See [6] */
  Compute M0 = [I − P + 𝒫]^{-1} ;
  Compute M1 = [I − [I − P + 𝒫]^{-1}];
  Compute M2 = inf_{α≠0} ||[I − P + α𝒫]^{-1}||_∞ ;
  for j = 1 to n do
    for i = 1 to n do
      if (𝒫χ)_i − (𝒫χ)_j ≠ 0 then
        θ_curr = 1/(8M2) |(𝒫χ)_i − (𝒫χ)_j|
      else
        for r = 0 to n do
          if (M0χ)_i ≠ (M0χ)_j then
            Break;
          else
            if (M0 M1^r χ)_i ≠ (M0 M1^r χ)_j then
              Break;
            endif
          endif
        endfor
        if r == 0 then
          θ_curr = |{(M0 − 𝒫)χ}_i − {(M0 − 𝒫)χ}_j| / (8M2);
        else
          if r > 0 AND r ⩽ n then
            θ_curr = |(M0 M1 χ)_i − (M0 M1 χ)_j| / (2^{r+3} M2) ;
          else
            θ_curr = 1 ;
          endif
        endif
      endif
      θ* = min(θ*, θ_curr) ;
    endfor
  endfor
end
```

Definition 2.12: Following Remark 2.1, we note that Algorithm 1 computes a lower bound for the critical termination probability for each iteration of Algorithm 2 such that the disabling/enabling decisions for the terminating plant coincide with the given non-terminating model. We define

$$\theta_{\min} = \min_k \theta_\star^{[k]}, \tag{20}$$

where $\theta_\star^{[k]}$ is the termination probability computed by Algorithm 1 in the k-th iteration of Algorithm 2.

Definition 2.13: If G and $G^\star$ are the unsupervised and supervised PFSA, respectively, then we denote the renormalised measure of the terminating plant $G^\star(\theta_{\min})$ as $\nu_\#^i : 2^{L(q_i)} \to [-1, 1]$ (Definition 2.7). Hence, in vector notation we have:

$$\boldsymbol{\nu}_\# = \theta_{\min}[I - (1-\theta_{\min})\mathbf{\Pi}^\#]^{-1}\boldsymbol{\chi}, \tag{21}$$

where $\mathbf{\Pi}^\#$ is the transition probability matrix of the supervised plant $G^\star$.

Algorithm 2: Computation of Optimal Supervisor

input : P, χ, $\mathcal{C}$
output: Optimal set of disabled transitions $\mathscr{D}^*$

```
begin
    Set 𝒟^[0] = ∅ ;                          /* Initial disabling set */
    Set Π̃^[0] = Π̃ ;                         /* Initial event prob. matrix */
    Set θ*^[0] = 0.99;
    Set k = 1 ;
    Set Terminate = false;
    while (Terminate == false) do
        Compute θ*^[k];                      /* Algorithm 1 */
        Set Π̃^[k] = ((1−θ*^[k])/(1−θ*^[k−1])) Π̃^[k−1];
        Compute ν^[k] ;
        for j = 1 to n do
            for i = 1 to n do
                Disable all controllable transitions q_i --σ--> q_j
                such that ν_j^[k] < ν_i^[k] ;
                Enable all controllable transitions q_i --σ--> q_j such
                that ν_j^[k] ≧ ν_i^[k] ;
            endfor
        endfor
        Collect all disabled transitions in 𝒟^[k];
        if 𝒟^[k] == 𝒟^[k−1] then
            Terminate = true;
        else
            k = k + 1 ;
        endif
    endw
    𝒟* = 𝒟^[k] ;                             /* Optimal disabling set */
end
```

Remark 2.2: Referring to Algorithm 2, it is noted that $\boldsymbol{\nu}_{\#} = \boldsymbol{\nu}^{[K]}$, where K is the total number of iterations for Algorithm 2.

3. Formulation of the path planning problem

The path planning problem is formulated on a two-dimensional workspace for the mobile agents (e.g. wheeled ground robots). This restriction on workspace dimensionality serves to simplify the exposition and can be easily relaxed for other applications. To set up the problem, the workspace is first discretised into a finite grid and hence the approach developed in this article falls under the generic category of discrete planning. The underlying theory does not require the grid to be regular; however, for the sake of clarity, we shall present the problem formulation under the assumption of a regular grid. The obstacles are represented as blocked-off grid locations in the discretised workspace. A particular location is specified as the fixed goal and the problem of finding optimal and feasible paths from arbitrary initial grid locations in the workspace is considered. Figure 1(a) illustrates the basic problem setup. It is further assumed that, at any given time instant, the robot occupies one particular location (i.e. a particular square in Figure 1(a)). As shown in Figure 1(b), the robot has eight possible moves from any interior location. The possible moves are modelled as controllable transitions between grid locations since the robot can 'choose' to execute a particular move from the available set. The number of possible moves (in this case eight) depends on the chosen fidelity of discretisation of the robot motion and also on the intrinsic vehicle dynamics. The results on the problem complexity, presented in this article, are generated based on the assumption that the number of available moves is significantly smaller than the number of grid squares (i.e. the discretised position states). In this way, specification of inter-grid transitions allows generation of an FSA description of the navigation problem. That is, each square in the discretised workspace is modelled as an FSA state with the controllable transitions defining the corresponding state transition map. A formal description of the model is presented below.

3.1 *Modelling of autonomous navigation*

A key idea in autonomous path planning is that of the configuration space $\mathcal{C}$ introduced by Lozano-Pérez and Wesley (1979). Instead of planning directly in the workspace, using methods such as swept volumes to determine whether or not a path was collision-free, the notion of the configuration space allows one to work with a point abstraction of the robot with the motion planning problem reducing to that of finding a path from an initial point to a goal point which represents the initial and final desired configurations. Drawing upon standard terminology, $\mathcal{C}_{\text{obs}} \subseteqq \mathcal{C}$ denotes the set of invalid or in-collision configurations and the set of all admissible configurations is denoted as $\mathcal{C}_{\text{free}} = \mathcal{C} \backslash \mathcal{C}_{\text{obs}}$. Application of language-measure theoretic optimisation to path planning requires setting up the problem in a specific framework which we describe next. For simplicity of exposition, the basic formulation is first developed for the case of a circular planar robot that can rotate about its geometric centre. Generalisations thereof will be discussed subsequently.

Let $\mathbb{G}_{NAV} = (Q, \Sigma, \delta, \widetilde{\boldsymbol{\Pi}}, \boldsymbol{\chi}, \mathcal{C})$ be a PFSA in the notation of Equation 19. In the absence of dynamic uncertainties and state estimation errors, the alphabet Σ contains only one uncontrollable event σ_u, i.e. $\Sigma = \Sigma_C \cup \{\sigma_u\}$, where Σ_C is the set of controllable events corresponding to the possible moves of the robot. The uncontrollable event σ_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

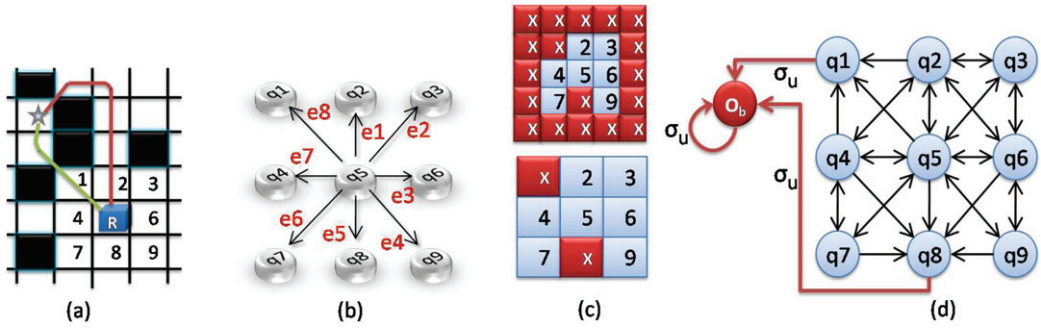

Figure 1. Illustration of the path planning problem formulation. (a) shows the vehicle (marked 'R') with the obstacle positions shown as black squares and the star identifies the goal. (b) shows the finite state representation of possible one-step moves from the current position q_5. (c) shows the the boundaries handled by surrounding the workspace with blocked position states of 'boundary obstacles' in the upper plate and a sample map with grid locations 1 and 8 blocked in the lower plate. (d) shows uncontrollable transitions 'σ_u' from states corresponding to blocked grid locations to the state 'O_b'.

numbered in a pre-determined scheme such that each $q_i \in Q \backslash \{q_\ominus\}$ denotes a specific square in the discretised 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 $Q_{\text{OBSTACLE}} \subseteq Q$.

Remark 3.1: Q_{OBSTACLE} thus represents an enumeration of the discretised $\mathcal{C}_{\text{obs}}$. For a circular planar robot, discretised configurations are the same as discretised positional coordinates. We note that it may be difficult to compute Q_{OBSTACLE} for complex planning scenarios from the physical problem specification. This issue is discussed in details later in § 7 where it is argued that accurate enumerations of $\mathcal{C}_{\text{obs}}$ is unnecessary and probabilistic estimations suffice.

Figure 1 illustrates the navigation automaton for a nine-state discretised workspace with two blocked squares. Note that the only outgoing transition from the blocked states q_1 and q_8 is σ_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 (Chattopadhyay and Ray 2007).

Definition 3.2: The characteristic weights are specified for the navigation automaton as follows:

$$\chi(q_i) = \begin{cases} -1 & \text{if } q_i \equiv q_\ominus \\ 1 & \text{if } q_i \text{ is the goal} \\ 0 & \text{otherwise} \end{cases}. \tag{22}$$

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 modelling 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: $\forall q_i \in Q$, $\sigma_j \in \Sigma$,

$$\tilde{\pi}(q_i, \sigma_j) = \begin{cases} \dfrac{1}{\text{No. of controllable events at } q_i}, & \text{if } \sigma_j \in \Sigma_C \\ 1, & \text{otherwise} \end{cases}. \tag{23}$$

It is shown in the sequel that computation of minimal length paths 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 Σ_C) is defined from each of the unblocked states.

The upper plate in Figure 1(c) shows that the boundaries are handled by 'surrounding' the workspace with blocked position states that are labelled as 'boundary obstacles' and the lower plate shows a scenario with grid locations 1 and 8 blocked. Figure 1(d) shows uncontrollable transitions σ_u from states corresponding to blocked grid locations to the state O_b.

3.2 *Generalised vehicle dynamics and operational constraints*

In general, the state set of the navigation automaton $\mathbb{G}_{\mathrm{NAV}}$ reflects an enumeration of the discretised configuration space of the robot. The alphabet denotes the different actions that the robot can execute at any configuration and hence the alphabet size corresponds to the number of reachable discrete configurations from each point in discretised $\mathcal{C}$. For multi-body robots, the resulting state space is rather large. However, construction of $\mathbb{G}_{\mathrm{NAV}}$ can be done algorithmically under the assumption that in the absence of obstacles the configuration space is a repetition of the local dynamical structure (except maybe at the workspace boundaries).

In the case of a 2D circular robot, the local structure is given by the ability to move to any of its eight neighbouring cells. The alphabet size is $8+1=9$ since the number of neighbouring cells correspond to the reachable configurations and we also have the uncontrollable collision event from the in-collision states. For a 2D rectangular robot with explicit specification of discretised heading, the alphabet size is $(8 \times \mathrm{H})+1$, where H is the number of discrete headings considered. For headings discretised at 15° intervals, the alphabet size is $192+1=193$. The constraints on the vehicle dynamics is reflected in the connectivity of $\mathbb{G}_{\mathrm{NAV}}$ and specifies the language of all possible action sequences that a particular robot can execute in absence of obstacles in the workspace. For a rectangular robot with a non-zero turn radius, the set of accessible configurations from any point of discretised $\mathcal{C}$ is restricted due to the constraint on the heading change that can be attained in a single action step. For 15° discretised headings and a maximum possible turn of $\pm 60°$ in each step, the locally accessible (discrete) configurations is illustrated in Figure 2. Note that in Figure 2, the robot is assumed to be unable to move back.

The handling of the resulting state space and determination of the characteristic weights χ is an issue for complex planning scenarios involving multi-body and non-holonomic robots. However, $\boldsymbol{v}^\star$ replaces expensive searches with a sequence of linear system solutions and thus is potentially applicable to the aforementioned cases. Unlike early deterministic planning approaches which required explicit and accurate enumeration of $\mathcal{C}_{\mathrm{obs}}$, $\boldsymbol{v}^\star$ works well with probabilistic specifications of the characteristic weights. These issues are discussed in more details in § 7.

4. Problem solution as decision-theoretic optimisation

The above-described PFSA-based navigation model facilitates computation of optimally feasible path plans via the language-measure-theoretic optimisation algorithm (Chattopadhyay and Ray 2007) that is succinctly described in § 2. Keeping in line with nomenclature in the path planning literature, the language-measure-theoretic algorithm is called $\boldsymbol{v}^\star$ in the sequel.

For the unsupervised model, the robot is free to execute any one of the defined controllable events from any given grid location as seen in Figure 2. The optimisation algorithm selectively disables controllable transitions to ensure that the formal measure vector of the navigation automaton is elementwise maximised. Physically, this implies that the supervised robot is constrained to choose among only the enabled moves at each state such that the probability of collision is minimised and simultaneously the probability of reaching the goal is maximised. Although $\boldsymbol{v}^\star$ is based on optimisation of probabilistic finite state machines, it is shown that an optimal and feasible path plan can be obtained that is executable in a deterministic sense. An example is shown in Figure 3 to illustrate the concept of decision-theoretic supervised navigation, where Figure 3(a) shows the available moves from a current state of an automaton under unsupervised navigation and Figure 3(b) shows

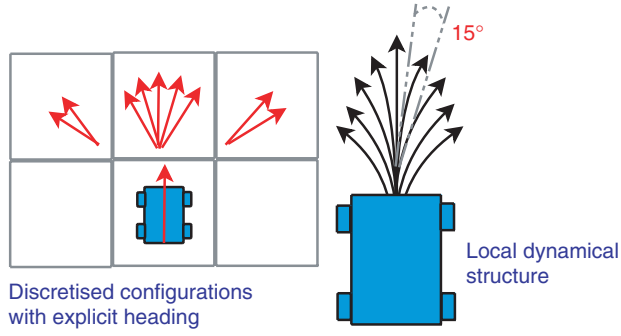

Figure 2. Modelling of rectangular robots with explicit heading.

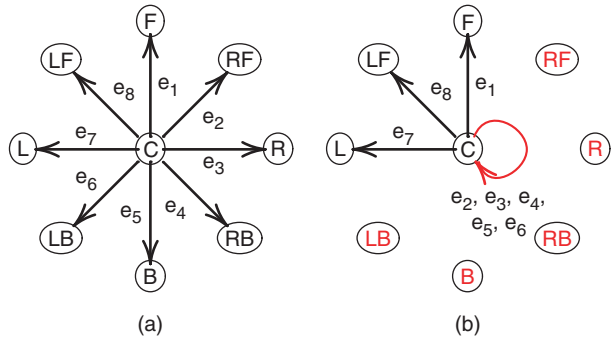

Figure 3. Illustration of navigation automaton. (a) shows available moves from the current state (C) in unsupervised navigation automaton. (b) shows enabled and disabled moves in the optimally supervised PFSA.

enabled and disabled moves under optimal supervision.

Let $\mathbb{G}_{\text{NAV}}$ be the unsupervised navigation automaton and $\mathbb{G}^{\star}_{\text{NAV}}$ be the optimally supervised PFSA obtained by application of $\boldsymbol{\nu}^{\star}$, where $\nu^{i}_{\#}$ is the renormalised measure of the terminating plant $\mathbb{G}^{\star}_{\text{NAV}}(\theta_{\min})$ with substochastic event generation probability matrix $\widetilde{\boldsymbol{\Pi}}^{\theta_{\min}} = (1-\theta_{\min})\widetilde{\boldsymbol{\Pi}}$. Denoting the event generating function (Definition 2.3) for $\mathbb{G}^{\star}_{\text{NAV}}$ and $\mathbb{G}^{\star}_{\text{NAV}}(\theta_{\min})$ as $\tilde{\pi}: Q \times \Sigma \to Q$ and $\tilde{\pi}^{\theta_{\min}}: Q \times \Sigma \to Q$, respectively, we have

$$\tilde{\pi}^{\theta_{\min}}(q_i, \epsilon) = 1 \tag{24a}$$

$$\forall q_i \in Q, \sigma_j \in \Sigma,\ \tilde{\pi}^{\theta_{\min}}(q_i, \sigma_j) = (1-\theta_{\min})\tilde{\pi}(q_i, \sigma_j). \tag{24b}$$

Notation 4.1: *For notational simplicity, we use*

$$\nu^{i}_{\#}(L(q_i)) \triangleq \nu_{\#}(q_i) = \boldsymbol{\nu}_{\#}|_i, \tag{25}$$

where $\boldsymbol{\nu}_{\#} = \theta_{\min}\,[\mathrm{I} - (1-\theta_{\min})\boldsymbol{\Pi}^{\#}]^{-1}\boldsymbol{\chi}$.

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

$$q_{r_1} = q_i \tag{26a}$$

$$q_{r_M} = q_j \tag{26b}$$

$$\forall i,\ j \in \{1, \ldots, M\}, \quad q_{r_i} \neq q_{r_j} \tag{26c}$$

$$\forall s \in \{1, \ldots, M\}, \quad \forall t \leqq s,\ \nu_{\#}(q_{r_t}) \leqq \nu_{\#}(q_{r_s}). \tag{26d}$$

Lemma 4.1: *There exists an enabled sequence of transitions from state* $q_i \in Q \backslash Q_{\text{OBSTACLE}}$ *to* $q_j \in Q \backslash \{q_{\ominus}\}$ *in* $\mathbb{G}^{\star}_{\text{NAV}}$ *if and only if there exists a* $\boldsymbol{\nu}^{\star}$*-path* $\rho(q_i, q_j)$ *in* $\mathbb{G}^{\star}_{\text{NAV}}$.

Proof: Let $q_i \xrightarrow{\sigma_1} q_{k1} \xrightarrow{\sigma_2} q_{k2} \xrightarrow{\sigma_3} \cdots q_{k\ell} \to \cdots \to q_j$ be an enabled sequence of transitions in $\mathbb{G}^{\star}_{\text{NAV}}$ where $\ell \in \{1, \ldots, M\}$ for some $M \leqq \text{CARD}(Q)$. It follows from the defined structure of the navigation automaton that there is no transition $q_s \xrightarrow{\sigma} q_r$, where $q_s \in Q_{\text{OBSTACLE}}$, $q_r \neq q_{\ominus}$. Then, $\forall \ell \in \{1, \ldots, M\},\ q_{k\ell} \notin \mathrm{Q}_{\text{OBSTACLE}}$. Then, each individual transition in the above transition sequence is controllable. Algorithm 2 implies that $\boldsymbol{\nu}_{\#}(q_i) \leq \boldsymbol{\nu}_{\#}(q_{k1}) \leq \boldsymbol{\nu}_{\#}\ (q_{k2}) \leq \cdots \leq \boldsymbol{\nu}_{\#}(q_j)$. Consequently, $\{q_i, q_{k1}, q_{k2}, \ldots, q_j\}$ is a $\boldsymbol{\nu}^{\star}$-path. The converse follows by a similar argument. □

Proposition 4.1: *For the optimally supervised navigation automaton* $\mathbb{G}^{\star}_{\text{NAV}}$*, the following condition holds*:

$$\forall q_i \in Q \setminus Q_{\text{OBSTACLE}}, \quad L(q_i) \subseteqq \Sigma^{\star}_C.$$

Proof: The proposition is proved by contradiction. Let the terminating state $q_j \in Q_{\text{OBSTACLE}}$ be reached from the initial state $q_i \in Q \backslash Q_{\text{OBSTACLE}}$ by an event trace $s \in L(q_i)$ such that $s \notin \Sigma^{\star}_C$. Since σ_u is the only uncontrollable event, it suffices to assign $s = \tilde{s}\sigma_u^K$, where $\tilde{s} \in \Sigma^{\star}_C$ and $K \geqq 1$. Then, it follows from Lemma 4.1 and Notation 4.1 that

$$\nu_{\#}(q_i) \leqq \nu_{\#}(q_j). \tag{27}$$

Since $L(q_j) = \sigma_u^{\star}$ and $\theta_{\min} \in (0, 1)$ and $\chi_j = -1$, it follows that

$$\nu_{\#}(q_j) = \theta_{\min} \sum_{k=1}^{\infty} (1-\theta_{\min})^k \ \ \chi_j = (\theta_{\min} - 1) < 0. \tag{28}$$

Furthermore, the measure $\boldsymbol{\nu}_{\#}(q_i)$ can be decomposed as:

$$\nu_{\#}(q_i) = \nu_{\#}\big(\{\omega \in L(q_i) : \delta^{\#}(q_i, \omega) \notin Q_{\text{OBSTACLE}}\}\big) + \nu_{\#}\big(\{\omega \in L(q_i) : \delta^{\#}(q_i, \omega) \in Q_{\text{OBSTACLE}}\}\big). \tag{29}$$

Since each state, except $q_{\ominus}$, has a non-negative characteristic weight (Definition 3.2), the first term in the right-hand side of Equation (29) is non-negative, i.e.

$$\nu_{\#}\big(\{\omega \in L(q_i) : \delta^{\#}(q_i, \omega \notin Q_{\text{OBSTACLE}}\}\big) \geqslant 0. \tag{30}$$

The second term in the right-hand side of Equation (29) yields:

$$\nu_{\#}(\{\omega \in L(q_i) : \delta^{\#}(q_i, \omega) \in Q_{\text{OBSTACLE}}\}) = \alpha \nu_{\#}(q_j), \tag{31}$$

where $\alpha = \theta_{\min} \sum_{\omega : \delta^{\#}(q_i, \omega) \in Q_{\text{OBSTACLE}} \setminus \{q_{\theta}\}} \tilde{\pi}^{\theta_{\min}}(q_i, \omega)$.

Since, for every $\sigma \in \Sigma_C$, $\tilde{\pi}^{\theta_{\min}}(q_i, \sigma) = (1/\text{CARD})(\Sigma_C)$ (see Equation (23) in Definition 3.3), and $\theta_{\min} \in (0, 1)$, it follows that

$$\alpha = \theta_{\min} \frac{1}{\text{CARD}}(\Sigma_C)\,(1-\theta_{\min}) \times \text{CARD}(\Sigma_C) \in (0, 1). \tag{32}$$

It is concluded from Equations (28) to (32) that $\boldsymbol{\nu}_{\#}(q_i) > \boldsymbol{\nu}_{\#}(q_j)$ which contradicts Equation (27). □

Corollary 4.1 (Obstacle avoidance)**:** *There exists no* $\boldsymbol{\nu}^{\star}$*-path from any unblocked state to any blocked state in the optimally supervised navigation automaton* $\mathbb{G}^{\star}_{\text{NAV}}$.

Proof: Let $q_i \in Q \backslash Q_{\text{OBSTACLE}}$ and assume that there exists a $\boldsymbol{\nu}^{\star}$-path from q_i to some $q_j \in Q_{\text{OBSTACLE}} \backslash \{q_{\ominus}\}$. It follows from Lemma 4.1 that there exists an enabled sequence of transitions from q_i to q_j. Since the uncontrollable transition σ_u is defined from $q_j \in Q_{\text{OBSTACLE}}$ (Definition 3.1), it follows that $L(q_i) \nsubseteqq \Sigma^{\star}_C$ which contradicts Proposition 4.1. □

Proposition 4.2 (Existence of $\boldsymbol{\nu}^{\star}$-paths)**:** *There exists a* $\boldsymbol{\nu}^{\star}$*-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* $\boldsymbol{\nu}_{\#}(q_i) > 0$.

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

$$\nu_{\#}(q_i) = \nu_{\#}(\{\omega\colon \delta^{\#}(q_i,\omega) = q_{\text{GOAL}}\}) + \nu_{\#}(\{\omega\colon \delta^{\#}(q_i,\omega) \in Q \setminus (Q_{\text{OBSTACLE}} \bigcup \{q_{\text{GOAL}}\})\}) + \nu_{\#}(\{\omega\colon \delta^{\#}(q_i,\omega) \in Q_{\text{OBSTACLE}}\}). \quad (33)$$

Now, Corollary 4.1 implies $\{\omega\colon \delta^{\#}(q_i,\omega) \in Q_{\text{OBSTACLE}}\} = \emptyset \Longrightarrow \boldsymbol{\nu}_{\#}(\{\omega\colon \delta^{\#}(q_i,\omega) \in Q_{\text{OBSTACLE}}\}) = 0$. Also, since $\forall q_j \in Q\setminus(Q_{\text{OBSTACLE}} \cup \{q_{\text{GOAL}}\})$, $\chi(q_j) = 0$, we have $\boldsymbol{\nu}_{\#}(\{\omega\colon \delta^{\#}(q_i,\omega) \in Q\setminus(Q_{\text{OBSTACLE}} \cup \{q_{\text{GOAL}}\})\}) = 0$. Hence we conclude

$$\nu_{\#}(q_i) = \nu_{\#}(\{\omega\colon \delta^{\#}(q_i,\omega) = q_{\text{GOAL}}\}). \quad (34)$$

It follows from $\chi(q_{\text{GOAL}}) = 1$ that $\delta^{\#}(q_i,\omega) = q_{\text{GOAL}} \Longrightarrow \boldsymbol{\nu}_{\#}(\{\omega\}) > 0$ implying there exists an enabled sequence of controllable transitions from q_i to q_{GOAL} in $\mathbb{G}^{\star}_{\text{NAV}}$ if and only if $\boldsymbol{\nu}_{\#}(q_i) > 0$. The result then follows from Lemma 4.1. □

Corollary 4.2 (Absence of local maxima)**:** *If there exists a* $\boldsymbol{\nu}^{\star}$*-path from* $q_i \in Q$ *to* $q_j \in Q$ *and a* $\boldsymbol{\nu}^{\star}$*-path from* q_i *to* q_{GOAL} *then there exists a* $\boldsymbol{\nu}^{\star}$*-path from* q_j *to* q_{GOAL}, i.e.

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

Proof: Let $q_i, q_j \in Q$. We note

$$\exists \rho_1(q_i, q_{\text{GOAL}}) \Rightarrow \nu_{\#}(q_i) > 0 \quad \text{(Proposition 4.2)} \quad (35a)$$

$$\exists \rho_2(q_i, q_j) \Rightarrow \nu_{\#}(q_i) \leq \nu_{\#}(q_j) \quad \text{(Definition 4.1)}. \quad (35b)$$

It follows that $\boldsymbol{\nu}_{\#}(q_j) > 0$ which implies that there exists a $\boldsymbol{\nu}^{\star}$-path from q_j to q_{GOAL} (Proposition 4.2). □

Algorithm 3: Computation of Next State

```
input : Current State q_CURRENT, ν_#, δ
output: Next State q_NEXT
1  begin
2      Set M = -INF;                    /* Large Negative Number */
3      for j = 1 to CARD(Σ_C) do
4          q_j = δ(q_CURRENT, σ_j);
5          if ν_#(q_j) > M then
6              q_NEXT = q_j;
7              M = ν_#(q_j);
8          endif
9      endfor
10 end
```

Remark 4.1 (Smoothing computed $\boldsymbol{\nu}^{\star}$-paths)**:** We note that given an arbitrary current state q_i, Algorithm 3 computes the next state q_{NEXT} as the one which has maximal measure among all possible next states q_j satisfying $\boldsymbol{\nu}_{\#}(q_j) \geqq \boldsymbol{\nu}_{\#}(q_i)$. This precludes the possibility illustrated in Figure 4 implying that the computed plan is relatively smoother. However, such smoothing may result in a computed plan that is not necessarily the shortest path (SP) from the initial state to the goal. This is expected since $\boldsymbol{\nu}^{\star}$ is not meant to determine the SP, but compute a $\boldsymbol{\nu}$-optimal path to the goal. In the sequel, we expound this optimal tradeoff that $\boldsymbol{\nu}^{\star}$ makes between path length and robust planning.

Notation 4.2: *We denote the* $\boldsymbol{\nu}^{\star}$*-path computed by Algorithm 3 as a* $\boldsymbol{\nu}^{\star}$*-plan in the sequel.*

4.1 *Optimal tradeoff: path length and robustness to uncertainties*

The majority of the reported path planning algorithms consider minimisation of the computed feasible path length as the sole optimisation objective. Mobile robotic platforms, however, suffer from varying degrees of dynamic and parametric uncertainties, implying that path length minimisation is of lesser practical importance to computing plans that are required to be robust under sensor noise, imperfect actuation and possibly accumulation of odometry errors that cannot be eliminated even with sophisticated signal processing techniques. The $\boldsymbol{\nu}^{\star}$-algorithm addresses this issue by an optimal trade-off between path lengths and availability of feasible alternate routes in the event of unforeseen dynamic uncertainties. Referring to Figure 5, if ω is the SP to goal from state q_k, then the SP from state q_i is given by $\sigma_2\omega$. However, a larger number of feasible paths is available

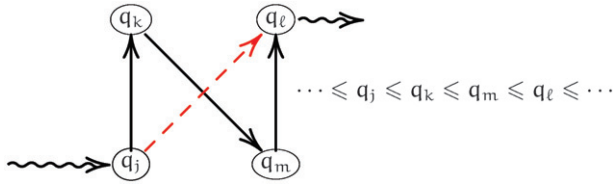

Figure 4. Situation involving four **neighbouring** position states q_j, q_k, q_m, q_ℓ with $q_j \leq q_k \leq q_m \leq q_\ell$ implying $\{q_j, q_k, q_m, q_\ell\}$ is a $\boldsymbol{\nu}^{*}$-path from q_j to q_ℓ. The smoothed $\boldsymbol{\nu}^{*}$-path is $\{q_j, q_\ell\}$.

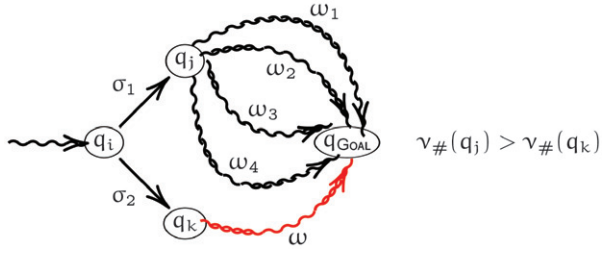

Figure 5. Tradeoff between path-length and robustness under dynamic uncertainty, where $\sigma_2\omega$ is the SP to q_{GOAL} from q_i. The $\boldsymbol{\nu}^{*}$ plan may be $\sigma_1\omega_1$ due to the availability of a larger number of feasible paths through q_j.

from state q_j which may result in the optimal $\boldsymbol{v}^\star$ plan to be $\sigma_1\omega_1$. Mathematically, each feasible path from state q_j has a positive measure which may sum to be greater than the measure of the single path ω from state q_k. The condition $\boldsymbol{v}_\#(q_j) > \boldsymbol{v}_\#(q_k)$ would then imply that the next state from q_i would be computed to be q_j and not q_k by Algorithm 3. Physically it can be interpreted that the mobile agent is better off going to q_j since the goal remains reachable even if one or more paths become unavailable. Next we estimate the *minimum number of alternate routes required for the $\boldsymbol{v}^\star$ plan to be different from the SP*. We need the following results:

Lemma 4.2: *For the optimally supervised navigation automaton $\mathbb{G}^\star_{\text{NAV}}$, we have $\forall q_i \in Q\backslash Q_{\text{OBSTACLE}}$,*

$$\forall \omega \in L(q_i),\ \nu^i_\#(\{\omega\}) = \theta_{\min}\left(\frac{1-\theta_{\min}}{\text{CARD}(\Sigma_C)}\right)^{|\omega|}\chi(\delta^\#(q_i,\omega)).$$

Proof: Following Proposition 4.1, it suffices to confine the range of event strings to $\omega \in \Sigma^\star_C$. It follows from Equation (23) in Definition 3.3 that $\forall q_i \in Q\backslash Q_{\text{OBSTACLE}}$,

$$\forall \omega \in L(q_i), \quad \tilde{\pi}^{\theta_{\min}}(q_i,\omega) = \left(\frac{1-\theta_{\min}}{\text{CARD}(\Sigma_C)}\right)^{|\omega|}. \tag{36}$$

Noting that Equation (36) is trivially satisfied for $|\omega| = 0$, the method of induction is used, starting with $|\omega| \geqq 1$. Since the control philosophy (Definition 2.8) does not alter the event generation probabilities, it follows that Equation (36) is satisfied for $|\omega| = 1$. Assuming that the result holds for all $\omega \in \Sigma^\star_C$ and for all $q_i \in Q\backslash Q_{\text{OBSTACLE}}$ such that $|\omega| = K$, we have

$$\tilde{\pi}^{\theta_{\min}}(q_i,\sigma\omega) = \tilde{\pi}^{\theta_{\min}}(q_i,\sigma)\tilde{\pi}^{\theta_{\min}}(\delta^\#(q_i,\sigma),\omega) \tag{37}$$

$$= \left(\frac{1-\theta_{\min}}{\text{CARD}(\Sigma_C)}\right)^{K+1}. \tag{38}$$

Finally, from Definition 2.13, $\nu^i_\#(\{\omega\}) = \theta_{\min}\tilde{\pi}^{\theta_{\min}}(q_i,\omega)\chi(q_j)$ where $q_j = \delta^\#(q_i,\omega)$. This completes the proof. □

Proposition 4.3: *For $q_i \in Q\backslash Q_{\text{OBSTACLE}}$, let $q_i \xrightarrow{\sigma_1} q_j \to \cdots \to q_{\text{GOAL}}$ be the SP to the goal. If there exists $q_k \in Q\backslash Q_{\text{OBSTACLE}}$ with $q_i \xrightarrow{\sigma_2} q_k$ for some $\sigma_2 \in \Sigma_C$ such that $\boldsymbol{v}_\#(q_k) > \boldsymbol{v}_\#(q_j)$, then the number of distinct paths to goal from state q_k is at least $\text{CARD}(\Sigma_C)+1$.*

Proof: Let the length of the SP $q_i \xrightarrow{\sigma_1} q_j \to \cdots \to q_{\text{GOAL}}$ be m. To compute a lower bound on the number of alternate routes, let us assume that there is only one path from state q_j to the goal. Further, we assume that every alternate path from state q_i through q_k has length $m+1$. If there are r such paths, then for the condition $\boldsymbol{v}_\#(q_k) > \boldsymbol{v}_\#(q_j)$ to be true, we need

$$\sum_{\omega_1 \in L(q_j)} \nu_\#(\{\omega_1\}) > \sum_{\omega_2 \in L(q_k)} \nu_\#(\{\omega_2\})$$

$$\Rightarrow r \times \theta_{\min}\left(\frac{1c-\theta_{\min}}{\text{CARD}(\Sigma_C)}\right)^m > \theta_{\min}\left(\frac{1-\theta_{\min}}{\text{CARD}(\Sigma_C)}\right)^{m-1}$$

(Lemma 4.2)

$$\Rightarrow r > \text{CARD}(\Sigma_C) \Rightarrow r\text{CARD}(\Sigma_C) + 1. \tag{39}$$

This completes the proof. □

The lower bound computed in Proposition 4.3 is not tight and if the alternate paths are longer or if there are multiple 'shortest' paths then the number of alternate routes required is significantly higher. In the detailed example presented in § 5, we illustrate a situation where $\boldsymbol{v}^\star$ opts for a longer but more robust plan.

4.2 *Summary of the $\boldsymbol{v}^\star$ algorithm for autonomous path planning*

The $\boldsymbol{v}^\star$ algorithm is summarised in Algorithm 4. There are two distinct approaches for executing the $\boldsymbol{v}^\star$ plan on a mobile platform (Figure 6) which can be enumerated as follows:

(1) Pass the computed $\boldsymbol{v}^\star$ plan (PLAN in Algorithm 4) to the mobile robot. The on board controller then executes the $\boldsymbol{v}^\star$ plan by sequentially moving to the states as defined in the ordered set PLAN . This involves less computation in real time on part of the on board controller.
(2) Terminate Algorithm 4 after Step 4 and pass the computed measure vector $\boldsymbol{v}_\#$ to the mobile robot. The on board controller then attempts to follow the measure gradient in the workspace defined by $\boldsymbol{v}_\#$. This approach is more robust since the mobile agent can choose to move to the next-best-state in the event of the best-state becoming unreachable due to unforeseen environmental dynamics.

5. An example of 2D path planning

Let a workspace be defined by a 9×9 grid as shown in Figure 7. The goal is labelled and prominently marked with a dot. The obstacles, i.e. states in the set Q_{OBSTACLE} (§ 3), are illustrated as blocked-off grid locations in black while the navigable space is shown in white. From the formulation presented in § 3, it follows that each grid location (i,j) is mapped to a state in the constructed navigation automaton $\mathbb{G}_{\text{NAV}}$ which has a total of $9\times9+1 = 82$ states. State numbers are

Algorithm 4: v^* Algorithm for Path Planning

```
input : Map, Goal Location
output: v*_Max-path to Goal Path
1  begin
2      Discretize map;
3      Generate G_NAV;
4      Compute v_#;
5      Get current state q_CURRENT;
6      if v_#(q_CURRENT) ≦ 0 then
7          Goal is unreachable ;
8          Abort;
9      else
10         count = 1;
11         Set Path(count) = q_CURRENT;
12         while q_CURRENT ≠ q_GOAL do
13             count = count + 1;
14             Get q_NEXT ;
15             Go to q_NEXT;
16             Set q_CURRENT = q_NEXT;
17             Save Path(count) = q_CURRENT;
18         endw
19     endif
20 end
```

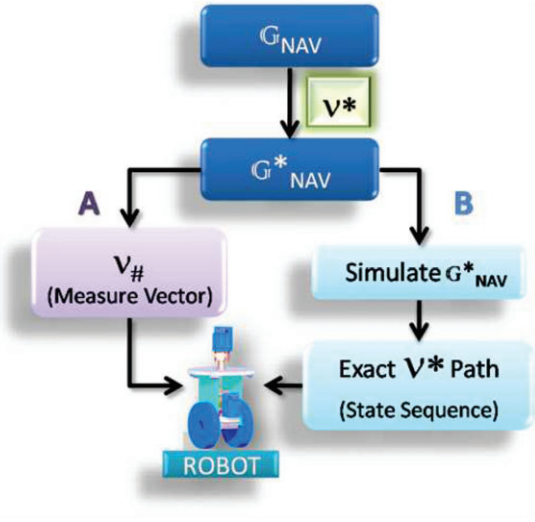

Figure 6. Alternative approaches to executing computed plan: The alternative A passes the measure vector expecting the on board controller to follow the measure gradient, while the alternative B computes the exact plan and passes it to the robot.

encoded by first moving along rows and then along columns, i.e.

$$\text{state\_num} = \text{column\_num} + (\text{row\_num} - 1) \times 9.$$

The 82nd state q_{82} is the special obstacle state $q_{\ominus}$ defined in § 3. We note that since (2, 7) is the goal (Figure 7), we have $\chi(7 + (2-1) \times 9) = \chi(16) = 1$. The measure vector $\boldsymbol{v}_{\#}$ is obtained by executing Algorithm 4 and the result is tabulated in Table 1. We note that the (i, j)th entry in Table 1 t_{ij} corresponds to $\boldsymbol{v}_{\#}(q_{j+(i-1)\times 9})$. The gradient

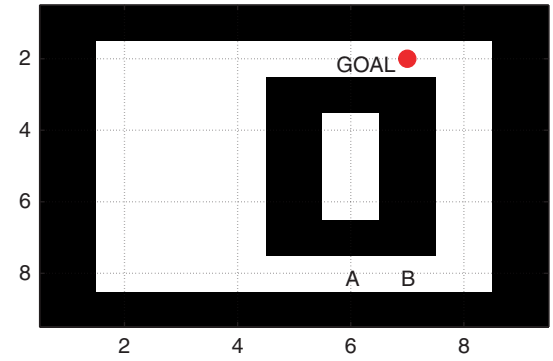

Figure 7. Illustration of 2D path planning. The work space shows a goal (GOAL) and two sample start locations (A and B).

induced in the workspace by $\boldsymbol{v}_{\#}$ is illustrated by the colour variation in Figure 8.

The following observations are made on the theoretical results derived in § 4.

(1) As derived in Proposition 4.2, the states or grid locations with strictly positive measure are precisely the ones from which the goal is reachable. We note from Table 1 that

$$t_{4,6} = t_{5,6} = t_{6,6} = 0 \tag{40}$$

and it is seen from Figure 7 that the grid locations (4, 6), (5, 6) and (6, 6) are completely surrounded with no existing feasible path to the goal.

(2) Each of the blocked-off grid locations has the minimal measure of −0.99 with every navigable state having a higher measure. It therefore follows that there exists no $\boldsymbol{v}^*$-path (Definition 4.1) from any navigable state to any blocked grid location (i.e. to any state in Q_{OBSTACLE}). Thus, as stated in Corollary 4.1, there is perfect obstacle avoidance as long as the robot follows a $\boldsymbol{v}^*$-path starting from any of the unblocked navigable states.

(3) There are no local maxima with the goal having the maximum measure value of 1.0. Thus true to Corollary 4.2, the robot is guaranteed to reach the goal by following any $\boldsymbol{v}^*$-path.

(4) The $\boldsymbol{v}^*$ plans are computed from the grid locations (8, 6) and (8, 7) (marked by the letters 'A' and 'B', respectively, in Figure 7). The respective plans are shown in Figure 8. The state sequences can be enumerated as follows:

$$\text{PLAN A: } 69 \to 68 \to 58 \to 48 \to 39 \to 30 \to 22 \to 14 \to 15 \to 16 \tag{41}$$

$$\text{PLAN B: } 70 \to 62 \to 53 \to 44 \to 35 \to 26 \to 16 \tag{42}$$

Table 1. The measure gradient.

−0.99	−0.99	−0.99	−0.99	−0.99	−0.99	−0.99	−0.99	−0.99
−0.99	0.969	0.972	0.976	0.984	0.992	1.00	0.992	−0.99
−0.99	0.969	0.972	0.976	−0.99	−0.99	−0.99	0.992	−0.99
−0.99	0.968	0.971	0.971	−0.99	0.00	−0.99	0.984	−0.99
−0.99	0.966	0.967	0.967	−0.99	0.00	−0.99	0.976	−0.99
−0.99	0.963	0.964	0.963	−0.99	0.00	−0.99	0.969	−0.99
−0.99	0.960	0.961	0.960	−0.99	−0.99	−0.99	0.961	−0.99
−0.99	0.957	0.958	0.957	0.955	0.950	0.953	0.953	−0.99
−0.99	−0.99	−0.99	−0.99	−0.99	−0.99	−0.99	−0.99	−0.99

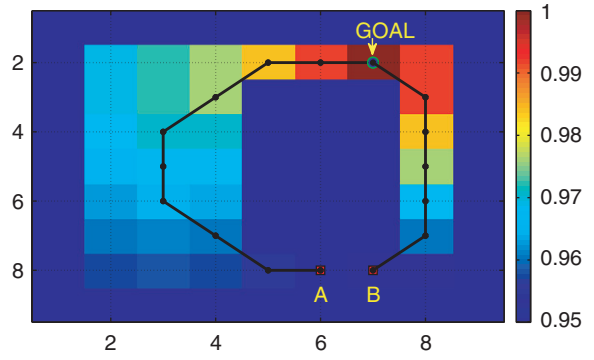

Figure 8. Gradient induced by $\nu_{\#}$ in the workspace and planning by the $\boldsymbol{\nu}^*$-algorithm from locations A and B.

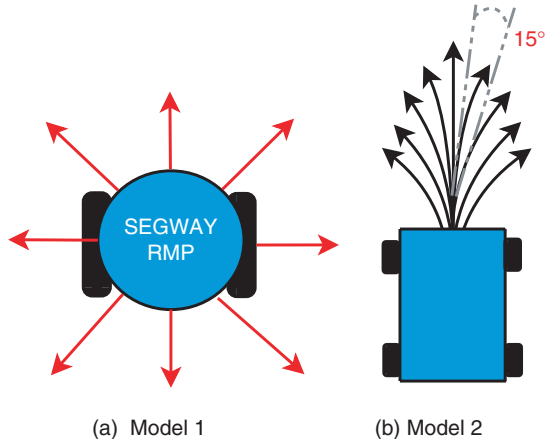

Figure 9. Two robot models. (a) A circular robot. (b) A rectangular robot with explicit heading and no provision for back-up.

The SP from location 'A' involves moving right to location 'B' and following the $\boldsymbol{\nu}^\star$ plan for 'B' thereafter. However, the $\boldsymbol{\nu}^\star$ algorithm opts for a more robust plan which avoids going through the narrow single-path corridor on the right. On the other hand, if the starting state is 'B', then the SP to the goal is just short enough to offset the effect of alternate (and longer) routes thus making the former optimal in this case.

5.1 *Examples of 2D planning with different dynamic constraints*

Two different models are considered pertaining to a circular robot e.g. Segway RMP (Model 1) and a rectangular robot e.g. Pioneer 2AT (Model 2) with non-zero turn radius that is incapable of moving backward (due to absence of rear sensors). The local dynamical structures of the two models is illustrated in Figure 9. We assume heading discretisation at 15° intervals resulting in 24 distinct headings. The maximum turn in a single action step is restricted to ±60° for the robot corresponding to Model 2. The simulated workspace is a 18×35 2D grid and is illustrated in Figure 10 along with obstacles (shaded). The navigation automata for each model is generated from the above described specification of the local dynamical structure as explained in §3. Table 2 enumerates the parameters of the respective navigation automata. Note that $\mathbb{G}_{\text{NAV}}$ corresponding to Model 1 has $(18 \times 35) + 1 = 631$ states; while the corresponding automaton for Model 2 has $(18 \times 35 \times 24) + 1 = 14282$ states. The alphabet size is also significantly greater for Model 2 due to the availability of larger number of possible actions at each state. While each state of Model 1 corresponds simply to a discretised positional coordinates, the Model 2 states also encode the discretised heading. The characteristic weights χ are assigned as explained in §3.1. For Model 2, a state is in collision with the obstacles if and only if the positional coordinates corresponding to the particular state matches an obstacle location in the workspace irrespective of the heading.

The dynamical constraints (e.g. turn restriction) on Model 2 imply that the number of feasible trajectories are significantly less as compared to Model 1. This is reflected in the increased average path length (e.g. 12.88 for Model 1; 19.3 for Model 2) computed over different initial configurations for the two models. Furthermore, the goal is no longer accessible from a number of initial configurations for Model 2 as enumerated in Table 2.

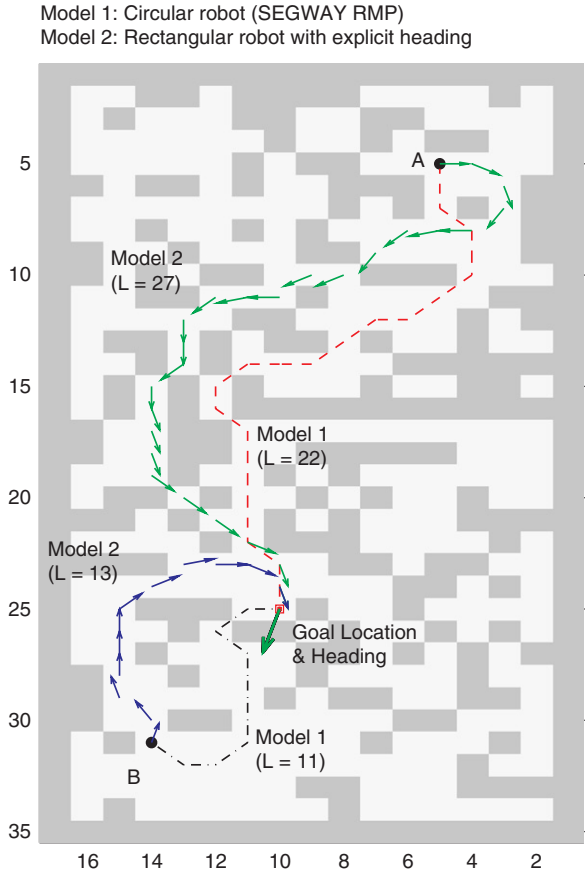

Figure 10. Sample runs for Models 1 and 2. Path length is denoted by L.

Table 2. Comparative results of two robot models.

	Model 1	Model 2
# States	631	14282
Alphabet size	9	193
Mean path length	12.88	19.3
Inaccessible states	0	4055

A comparison of the probability distribution of the path lengths for different initial configurations is further illustrative of the restricted dynamics for Model 2. Referring to Figure 11, we note that the distribution shifts towards longer path lengths. The first bin in the lower plate of Figure 11 (corresponding to path length -1) indicates the non-existence of a feasible path. Thus we note that for Model 2, the probability that no feasible path exists to the goal is $\sim$0.29 for randomly chosen initial configurations. Sample runs for the two models are shown in Figure 10. The initial configurations are indicated as A and B. The goal configuration (desired position and heading) is denoted by a bold arrow. The dotted paths are for Model 1 while the arrow

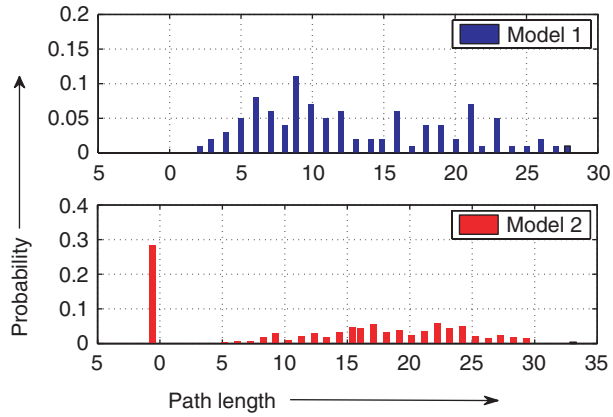

Figure 11. Path length distributions for Model 1 (upper plate) and Model 2 (lower plate). Non-existence of feasible paths in Model 2 is represented by assigning a path length of -1 which corresponds to the leftmost bin in the lower plate.

sequences denote the paths taken by Model 2. Note that the path lengths (denoted as L in Figure 10) are longer for Model 2. Also note that while the Model 1 paths have sharp turns; those for Model 2 are more rounded with gradual turns.

6. Algorithmic complexity

Let n be the number of states in the navigation automaton $\mathbb{G}_{\text{NAV}}$. The key step in the $\boldsymbol{\nu}^\star$-algorithm is computation of the measure vector $\nu^{[k]} = \theta_\star^{[k]}[\mathbf{I} - (1-\theta_\star^{[k]})\mathbf{\Pi}^{[k]}]^{-1}$ in Step 8 of Algorithm 2 where the superscript $[k]$ denote the k-th iteration. It is noted that $[\mathbf{I} - (1-\theta_\star^{[k]})\mathbf{\Pi}^{[k]}]^{-1} = \sum_{j=1}^{\infty}(1-\theta_\star^{[k]})^j(\mathbf{\Pi}^{[k]})^j\boldsymbol{\chi}$. Noting that $\theta_\star^{[k]} \in (0,1)$, and having $\text{EPS} > 0$ to be the m/c precision, it follows that

$$\exists\, r(\text{EPS}) \in \mathbb{N} \text{ s.t. } (1-\theta_\star^{[k]})^r(\mathbf{\Pi}^{[k]})^r\boldsymbol{\chi} \leqq \text{EPS}. \tag{43}$$

Therefore, the following truncation is justified:

$$[\mathbf{I} - (1-\theta_\star^{[k]})\mathbf{\Pi}^{[k]}]^{-1} \approx \sum_{j=1}^{r}(1-\theta_\star^{[k]})^j(\mathbf{\Pi}^{[k]})^j\boldsymbol{\chi}. \tag{44}$$

It is observed that for the navigation model presented in §3, $\mathbf{\Pi}^{[k]}$ is sparse for all iteration steps of Algorithm 2 with the number of non-zero elements in each row bounded by $\text{CARD}(\Sigma_C)$. It follows that for any arbitrary vector y, $\mathbf{\Pi}^{[k]}y$ can be computed in $O(n \times \text{CARD}(\Sigma_C))$ time steps. Since $\mathbf{\Pi}^{[k]})^{j+1}\boldsymbol{\chi} = (\mathbf{\Pi}^{[k]})(\mathbf{\Pi}^{[k]})^j\boldsymbol{\chi}$, it follows that the computation in Equation (44) has a time complexity of $O(n \times \text{CARD}(\Sigma_C) \times r(\text{EPS}))$. It was argued in Chattopadhyay and Ray (2007) that the number of iterations for Algorithm 2 is strictly less than n. Therefore it implies that for a fixed machine

precision and a given number of controllable moves (i.e. a fixed $\text{CARD}(\Sigma_C)$), we have

$$\text{Time complexity of } \boldsymbol{\nu}^\star < O(n^2). \tag{45}$$

To put things in perspective, we compare experimental runs with Dijkstra's standard SP algorithm (Cormen, Leiserson, Rivest, and Stein 2001) in Figure 12 which was generated by choosing for each n, 10^2 random obstacle maps and goals solving each problem by the two algorithms consecutively. We note that $(\log \text{Run time}/\log n) \approx 2$ for Dijkstra's and is only about 1.4 for $\boldsymbol{\nu}^\star$ implying that the latter is significantly superior for large problem sizes.

The run times for Dijkstra's search in Figure 12 were generated without using priority queues resulting in asymptotic time complexity of $O(n^2)$. Implementations using prioritisation via Fibonacci heaps have complexity $O(|\text{E}|\log\ n)$, where $|\text{E}|$ is the number of graph edges. Therefore, for sparse connectivity, e.g. for $|\text{E}| = O(n)$, the asymptotic complexity is $O(n\ \log\ n)$. This is better than the $\boldsymbol{\nu}^\star$ time complexity bound of $O(n^2)$. The $O(n^2)$ bound on $\boldsymbol{\nu}^\star$, as shown above, is not tight and several speed-up possibilities exist. For example, simulation results suggest that it may be possible to pre-specify an upper bound on the number of iterations purely as a function of the desired precision independent of n, in which case the serial asymptotic complexity for $\boldsymbol{\nu}^\star$ (for sparse connectivity) will drop to $O(n)$. $\boldsymbol{\nu}^\star$ exclusively relies on sparse linear system solutions and hence the run-time speedup for parallel implementations is considerable being only a function of the state-of-the-art in the parallel algorithms for the solution of linear systems. To further highlight the computational advantage of using a linear-system-solution-based approach, we note that while parallelised general graph search complexity currently stands at $O(\log^2 n)$ (Meyer 2002), it is possible in theory to solve general linear systems in asymptotically constant time, i.e. in time asymptotically independent of problem size (Bojanczyk 1984) with $O(n^3)$ processors. Future work will focus on sharpening the complexity bounds on $\boldsymbol{\nu}^\star$ for both serial and parallel implementations.

Remark 6.1: The $\boldsymbol{\nu}^\star$ algorithm offers a fundamentally different and potentially faster solution to graph search problems if one is interested in finding 'robust' paths as opposed to 'shortest' ones. Dijkstra-like algorithms can be tuned to find robust paths by modifying the cost function, e.g. by including the costs due to presence of obstacles. However, construction of an algorithm that defines the costs by incorporating tradeoffs, such as shorter *versus* more robust paths, is non-trivial and it potentially increases complexity of the algorithm. In contrast, a tradeoff in the $\boldsymbol{\nu}^\star$ algorithm arises naturally due to the fact that existence of a number of feasible paths to the goal increases the measure of the state as discussed in §4.1.

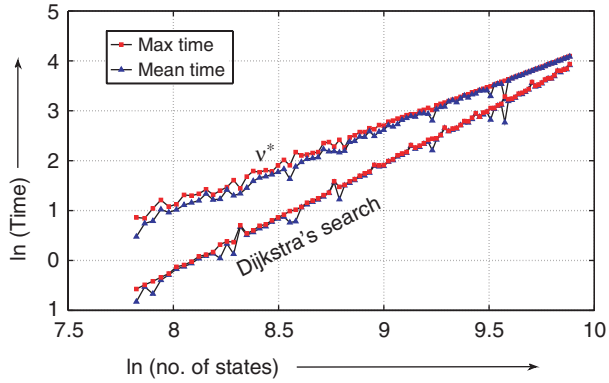

Figure 12. Run-time comparisons of $\boldsymbol{\nu}^*$ and Dijkstra's standard search based on extensive Monte Carlo simulation.

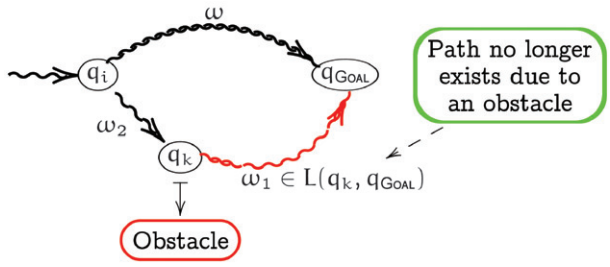

Figure 13. Illustration of the ν^*-based dynamic replanning concept.

7. Dynamic replanning and extension to higher dimensional problems

In §6, we compared $\boldsymbol{\nu}^\star$ with the standard Dijkstra's search. However, the state-of-the-art in grid-based 2D planning is D* (LaValle 2006), which is a modification of the basic Dijkstra's algorithm to enable fast re-computation of plans online in the face of newly learnt information. It turns out that, with regards to implementation, it is often critically important to be able to dynamically update plans based on new information. The analysis presented in this article primarily deals with offline plan computations. However, $\boldsymbol{\nu}^\star$ can be easily adapted for dynamic re-planning as discussed next. Let $\boldsymbol{\nu}_\#$ be the optimal language measure vector computed offline prior to mission execution. As the mission progresses, it is learnt that state q_k is in fact in collision with an obstacle i.e. $q_k \in Q_{\text{OBSTACLE}}$. It follows that paths from the current state q_i that reached the goal via q_k no longer exist. Figure 13 illustrates the situation. Note that if the measure of the current state $\nu^i_\# > \nu^k_\#$, then no re-planning is required since the $\boldsymbol{\nu}^\star$-path from q_i to

q_{GOAL} does not pass through q_k (Definition 4.1 and Proposition 4.2). On the other hand, if $v^i_{\#} \leqq v^k_{\#}$, then re-planning is necessary and $\boldsymbol{v}_{\#}$ may be updated in the following way.

Let $\Lambda = \theta_{\min}[\mathrm{I} - (1-\theta_{\min})\boldsymbol{\Pi}^{\#}]^{-1}$ and Λ_{ij} be its ij-th element.

Let G correspond to the index of state q_{GOAL}. Then,

$$\forall q_i \in Q, \quad \widehat{v}^i_{\#} = v^i_{\#} - \Lambda_{iG}\big(\Lambda_{kG}(1-\pi_{kk})\big)\chi_G \qquad (46)$$

which follows from the fact that $\Lambda_{iG}(\Lambda_{kG}(1-\pi_{kk}))\chi_G$ corresponds to the contribution from strings that reach goal via q_k.

Since $[\mathbf{I} - \Pi^{\#}]^{-1}$ is computed offline in course of computing the original plan, the update can be executed sufficiently fast. Furthermore, if one delays updating the measures of individual states until necessary (similar to the lazy approach in probabilistic planning where one delays collision checking until needed (Kondo 1991; Bohlin and Kavraki 2000)) then the update calculations can be carried out locally in constant time (for sparse connectivity) making the dynamic updates for $\boldsymbol{v}^{\star}$ potentially faster to D* in the sense of local asymptotic complexity. Rigorous development of a dynamically updating $\boldsymbol{v}^{\star}$ is a topic of future research.

Next we discuss potential extensions to high dimensional and complex planning scenarios. There are two key issues:

(1) Explicit enumeration of $\mathcal{C}_{obs}$ or equivalently specifying the state characteristic function χ *a priori* may become impractical for large problems.
(2) The state set Q for $\mathbb{G}_{\text{NAV}}$ becomes large.

To address the first issue, we recall from §4.1 that $\boldsymbol{v}^{\star}$ typically computes optimal plans as opposed to SP routes. As illustrated in the first example in §5, this amounts to avoiding *'narrow spaces'* if possible. Thus, $\boldsymbol{v}^{\star}$ plans are inherently robust and are insensitive to small perturbations of the obstacle map. Robustness of the computed path has at least two critical implications:

(1) Shortest path routes are often extremely difficult to navigate in practice due to noise corruption of on-board positional sensing. Optimal tradeoff between robustness and path length results in a plan that can be navigated with much more ease in the face of sensor noise and obstacle perturbations. Since a navigation algorithm typically slows down the robotic platform in obstacle vicinity, the performance gain is reflected in significant reduction of mission execution time as illustrated in the reported laboratory experiments (§8).
(2) Robustness to obstacle perturbations implies that exact enumeration of the characteristic function χ is unnecessary and a probabilistic estimation suffices. This property of the $\boldsymbol{v}^{\star}$-algorithm potentially remedies the problem with early bitmap-based planning approaches (Lengyel, Reichert, Donald and Greenberg 1990) that require exact enumeration of the $\mathcal{C}_{obs}$ and depend on classical AI-based search techniques to find paths. It could be possible to 'miss' obstacles in the sampled set; however, as shown in Equation (46), $\boldsymbol{v}^{\star}$-based dynamic re-planning can be implemented to run sufficiently fast to address this issue.

To address the issue of large state sets, we note that the arguments presented in §6 imply that $\boldsymbol{v}^{\star}$ can be implemented to run in $O(n)$ time under most cases and is highly parallelisable. The objective of this article is to present a planning philosophy that is fundamentally different from the ones reported in the literature and justify potential adaptations for complex planning scenarios. Critical issues such as the specifics of sampling strategy for χ and actual performance comparisons with the current state-of-the-art in high-dimensional and non-holonomic planning require further investigation and is a topic of future research.

8. Experimental results with Segway RMP

The proposed algorithm has been successfully implemented on different types of mobile robotic platforms in a laboratory environment. One such richly instrumented implementation platform (Segway RMP) is shown in the inset of Figure 14. The algorithm is coded on a fully open-source architecture (C++/gcc4.1/Fedora Core 7) and uses in-house developed hardware drivers in conjunction with the PLAYER robot server (Gerkey, Vaughan, and Howard 2003). The robot is equipped with a bar code that can be recognised via an

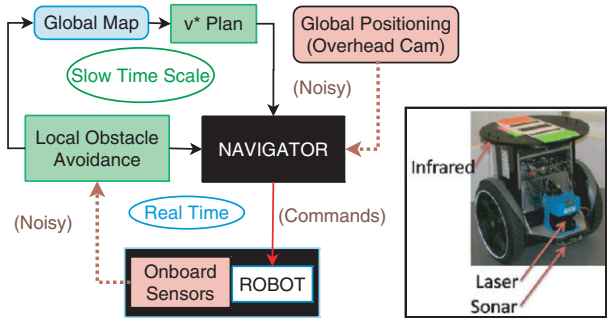

Figure 14. Autonomous navigation scheme with $\boldsymbol{v}^{*}$ implementation on a richly instrumented Segway RMP (shown in inset).

overhead multi-cam vision system to allow continuous global position and heading fix. However, the naturally varying illumination in the laboratory results in significant localisation noise. Moreover, even with an *a priori* known obstacle map, sensor noise causes observed perturbations in obstacle positions. The overall control scheme is illustrated in Figure 14. Navigation commands such as velocity and turn rate are written to the robotic hardware by the NAVIGATOR which decides upon the specific values in the real time based on the robot localisation, the $\boldsymbol{v}^{\star}$ plan and locally observed obstacles detected via on-board sensing. For mobile obstacles, the local information is integrated with the global map and the $\boldsymbol{v}^{\star}$ plan is updated in a slower time scale. In the reported experiment, we consider only static obstacles in the workspace. The obstacle positions are shown in Figures 15(a) and (b). The objective of the reported experiment is 2-fold:

(1) To validate the practical applicability of $\boldsymbol{v}^{\star}$.

(2) To validate that $\boldsymbol{v}^{\star}$ optimal routes are more reliable and navigable compared to shortest route planning especially under high-noise scenarios in cluttered environments.

The workspace dimensions are 9×7 m. Position discretisation resulted in $53 \times 29 = 1537$ states for the navigation automaton. Since the Segway RMP is a circular robot, we used the modelling scheme denoted as Model 1 in § 5 resulting in an alphabet size of eight. The mission consists of repeatedly visiting the pre-specified goals A, B, C, D and E (Figure 15a) in sequence starting from A. Noise filters were intentionally turned off to simulate a particularly high noise environment. Sensor and localisation noise coupled with the inverted-pendulum dynamics of the Segway RMP poses a challenging problem in autonomous navigation. The mission was executed using $\boldsymbol{v}^{\star}$ planning and SP routes separately for 32 laps each. Computation time in each case was comparable. The trajectories obtained from actual localisation

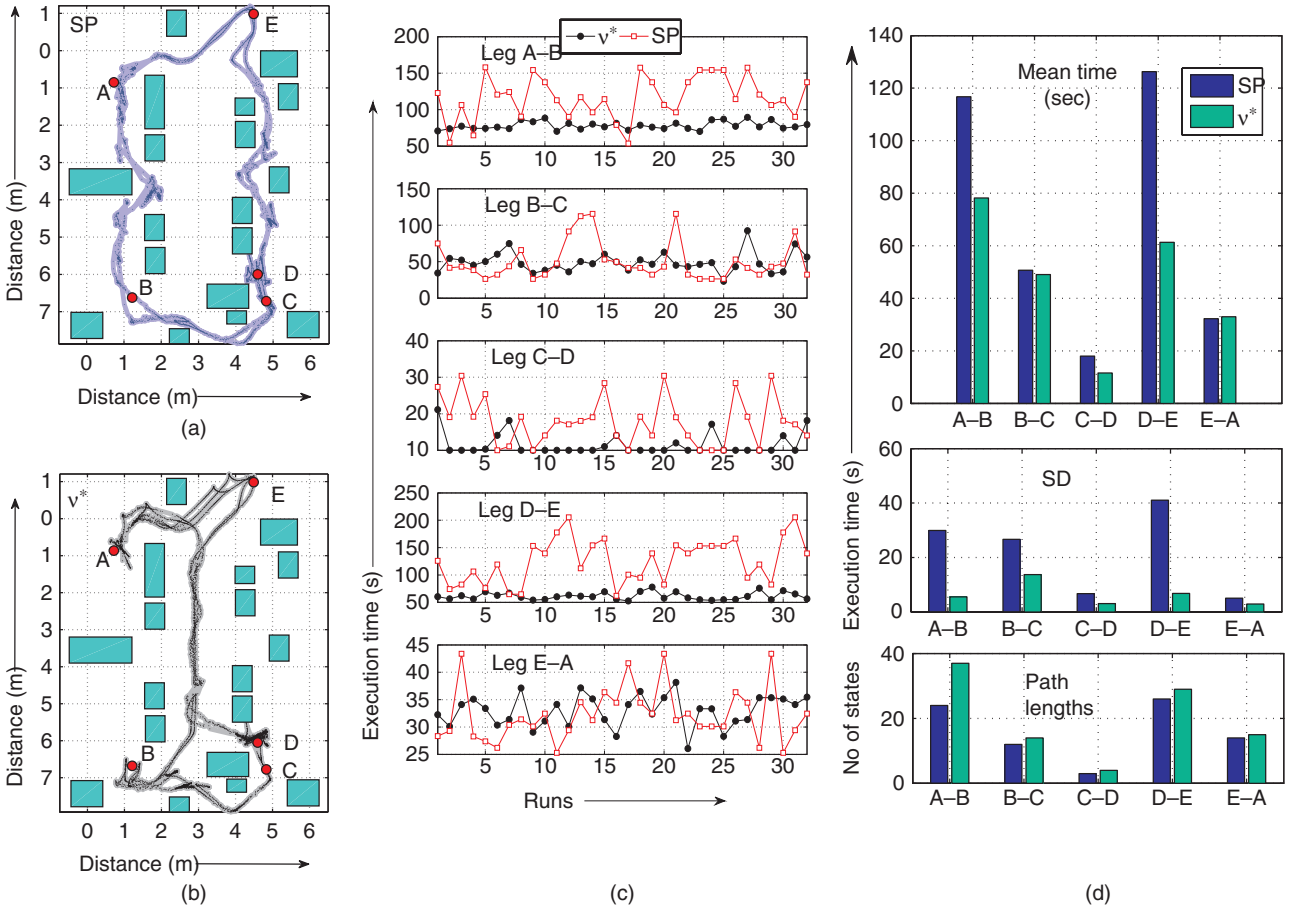

Figure 15. Experimental comparison of $\boldsymbol{v}^{*}$ and SP routes. Plots (a) and (b) illustrate the trajectories for SP and $\boldsymbol{v}^{*}$, respectively. The intermediate goals are denoted as A, B, C, D and E. Plot (c) compares execution times for each leg of the mission. The top and middle plates of plot (d) compare the mean and standard deviation (SD) of execution times for each leg, respectively. The lower plate of plot (d) compares the computed path lengths for each leg.

history for the SP planning is shown in Figure 15(a), while those for the $v^\star$ planning is shown in Figure 15(b). Note the localisation noise apparent in the trajectory histories. The SP routes typically take the robot through a significantly more convoluted path requiring precise positioning to avoid collisions which results in degraded overall mission performance as discussed next. The completion times for each leg of the mission are plotted in Figure 15(c). The mean execution times are compared in the top plate of Figure 15(d). Note that legs A–B and D–E take much longer on average for the SP routes. The mean times for the other legs are comparable. Referring to Figures 15(a) and (b), we note that the legs A–B and D–E are the ones at which the plans differ significantly. Moreover, the standard deviation (SD) of the execution times (middle plate of Figure 15d) illustrate that the $v^\star$ plans are significantly more reliable. The difference is best brought out for the data corresponding to leg D–E. Note that the path lengths as shown in the bottom plate of Figure 15(d) are comparable for this leg; however, the mean execution time for SP is more than double that of $v^\star$ with the SD for SP approximately five times that of $v^\star$. Thus, the experimental results validate the premise that SP routes may not be particularly desirable in uncertain environments; significant improvements can be obtained if one opts for a slightly longer paths using $v^\star$ that optimally avoid obstacle clutter.

9. Summary and future research

A novel path planning algorithm $v^\star$ is introduced that models the autonomous navigation as an optimisation problem for probabilistic finite state machines and applies the theory of language-measure-theoretic optimal control to compute v-optimal plan to the specified goal, with automated tradeoff between path length and robustness of the plan under dynamic uncertainty.

9.1 *Future work*

Future work will address the following issues:

(1) **Dynamic replanning:** Adapting $v^\star$ to enable efficient dynamic replanning and compare computational and executional effectiveness with the current state-of-the-art.
(2) **Probabilistic estimation of χ:** Sampling schemes for estimating the characteristic function χ in complex planning scenarios.
(3) **Performance comparison:** Performance comparisons with state-of-the-art planners for dynamic re-planning schemes and in high-dimensional scenarios with possibly non-holonomic constraints.
(4) **Multi-robot coordinated planning:** Run-time complexity grows exponentially with the number of agents if one attempts to solve the full Cartesian product problem. The $v^\star$-algorithm could be potentially used to plan individually followed by an intelligent assembly of the plans to take interaction into account.
(5) **Hierarchical implementation to handle very large work spaces:** Large work spaces could be solved more efficiently if path planning is done when needed rather than solving the whole problem at once; however, care must be taken to ensure that the computed solution is not too far from the optimal one.
(6) **Handling uncontrollable dynamic events:** In this article, the only uncontrollable event is the one modelling a collision when the robot attempts to move into a blocked state. Physical dynamic response of the robot may need to be modelled as uncontrolled transitions in highly uncertain environments.

Acknowledgements

This work has been supported in part by the US Army Research Office under Grant No. W911NF-07-1-0376 and the Office of Naval Research under Grant No. N00014-08-1-380.

References

Anisi, D.A., Hamberg, J., and Hu, X. (2003), 'Nearly Time-Optimal Paths for a Ground Vehicle,' *Journal of Control Theory and Applications*, 1, 2–8.

Barraquand, J., Langlois, B., and Latombe, J.-C. (1990), *Robot Motion Planning with Many Degrees of Freedom and Dynamic Constraints*, Cambridge, MA, USA: MIT Press.

Bohlin, R., and Kavraki, L.E. (2000), 'Path Planning Using Lazy PRM,' *International Conference on Robotics and Automation (ICRA)*, San Francisco, CA, pp. 521–528.

Bojanczyk, A. (1984), 'Complexity of Solving Linear Systems in Different Models of Computation,' *Siam Journal of Numerical Analysis*, 21, 591–603.

Chattopadhyay, I., and Ray, A. (2006), 'Renormalized Measure of Regular Languages,' *International Journal of Control*, 79, 1107–1117.

——— (2007), 'Language-Measure-Theoretic Optimal Control of Probabilistic Finite-State Systems,' *International Journal of Control*, 80, 1271–1290.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., and Stein, C. (2001), *Introduction to Algorithms* (2nd ed.), Cambridge, MA: MIT Press.

Garg, V., (December, 1992a), 'An Algebraic Approach to Modelling Probabilistic Discrete Event Systems,'

in *Proceedings of 1992 IEEE Conference on Decision and Control*, Tucson, AZ, pp. 2348–2353.

—— (March 1992b), 'Probabilistic Languages for Modelling of DEDs', in *Proceedings of 1992 IEEE Conference on Information and Sciences*, Princeton, NJ, pp. 198–203.

Gerkey, B., Vaughan, R., and Howard, A. (June 30–July 3, 2003), 'The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems,' in *Proceedings of the International Conference on Advanced Robotics (ICAR 2003)*, Coimbra, Portugal, pp. 317–323.

Hopcroft, J.E., Motwani, R., and Ullman, J.D. (2001), *Introduction to Automata Theory, Languages, and Computation* (2nd ed.), Boston, MA: Addison-Wesley.

Kondo, K. (1991), 'Motion Planning with Six Degrees of Freedom by Multi-strategic-bi-directional Heuristic Free-Space Enumeration,' *IEEE Transactions on Robotics and Automation*, 7, 267–277.

Latombe, J.-C. (1991), 'Robot Motion Planning,' in *International Series in Engineering and Computer Science; Robotics: Vision, Manipulation and Sensors*, Boston, MA, U.S.A.: Kluwer Academic Publishers.

LaValle, S. (2006), *Planning Algorithms*, Cambridge University Press, Cambridge, U.K., http://planning.cs.uiuc.edu/

Lengyel, J., Reichert, M., Donald, B.R., and Greenberg, D.P. (1990), 'Real-Time Robot Motion Planning Using Rasterizing Computer Graphics Hardware,' *Computer Graphics*, 24, 327–335.

Lozano-Perez, T. (1987), 'A Simple Motion-Planning Algorithm for General Robot Manipulators,' *IEEE Transactions on Robotics and Automation*, 3, 224–238.

Lozano-Perez, T., and Wesley, M.A. (1979), 'An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles,' *Communications of the ACM*, 22, 560–570.

Meyer, U. (2002), 'Buckets Strike Back: Improved Parallel Shortest-paths, in Parallel and Distributed Processing Symposium,' in *Proceedings International, IPDPS 2002*, Abstracts and CD-ROM, pp. 75–82.

Ray, A. (2005), 'Signed Real Measure of Regular Languages for Discrete-event Supervisory Control,' *International Journal of Control*, 78, 949–967.

Rudin, W. (1988), *Real and Complex Analysis* (3rd ed.), New York, NY: McGraw Hill.