3

Application of Streaming Algorithms and DFA Learning for Approximating Solutions to Problems in Robot Navigation

Carlos Rodríguez Lucatero
Universidad Autónoma Metropolitana Unidad Cuajimalpa
México

1. Introduction

The main subject of this chapter is the robot navigation what implies motion planning problems. With the purpose of giving context to this chapter, I will start making a general overview of what is robot motion planning. For this reason, I will start giving some abstract of the general definitions and notions that can be frequently found in many robot motion planning books as for example (Latombe (1990)). After that I will talk about some robot motion problems that can be found in many research articles published in the last fifteen years and that have been the subject of some of my own research in the robot navigation field.

1.1 Robot motion planning and configuration space of a rigid body

The purpose of this section is to define the notion of configuration space when a robot is a rigid object without cinematic and dynamic limitations. One of the main goals of robotics is to create autonomous robots that receive as input high level descriptions of the tasks to be performed without further human intervention. For high level description we mean to specify the *what* task moreover than the *how* to do a task. A robot can be defined as a flexible mechanical device equiped with sensors and controled by a computer. Among some domains of application of these devices it can be mentioned the following:

- Manufacturing
- Garbage recolection
- Help to inabilited people
- Space exploration
- Submarine exploration
- Surgery

The robotics field started up big challenges in Computer Science and tends to be a source of inspiration of many new concepts in this field.

1.2 Robot motion planning

The development of technologies for autonomous robots is in strong relationship with the achievements in computational learning, automatic reasoning systems, perception and control

research. Robotics give place to very interesting and important issues such as the *motion planning*. One of the concerns of *motion planning* is for example, what is the sequence of movements that have to be performed by a robot to achieve some given objects configuration. The less that can be hoped from an autonomous robot is that it has the hability to plan his own motions. At first sight it seems an easy job for a human because we normally do it all the time, but it is not so easy for the robots given that it has strong space and time computational constrains for performing it in an computational efficient way. The amount of mathematical as well as algorithmic that are needed for the implementation of a somehow general planner is overhelming. The first computer controlled robots appear in the 60's. However the biggest efforts have been lead during the 80's. Robotics and robot motion planning has been benefitted by the thoretical and practical knowledge produced by the research on Artificial Intelligence, Mathematics, Computer Science and Mechanical Engineering. As a consequence the computational complexity implications of the problems that arise in motion planning can be better grasped. This allow us to understand that robot motion planning is much more than to plan the movements of a robot avoiding to collide with obstacles.
The motion planning have to take into account geometrical as well as physical and temporal constrains of the robots. The motion planning under uncertainty need to interact with the environment and use the sensors information to take the best decision when the information about the world is partial. The concept of configuration space was coined by (Lozano-Perez (1986)) and is a mathematical tool for representing a robot as a point in an appropiate space. So, the geometry as well as the friction involved on a task can be mapped such configuration space. Many geometrical tool such as the geometrical topology and algebra are well adapted to such a representation. An alternative tool used frequently for motion planning is the *potential fields approach*. The figures 1 and 2 are an example of a motion planning simulation of a robot represented by a rectangular rod that moves in a 2D work space and with 3D configuration space ((x_i, y_i) position in the plane,(θ_i) orientation). This simulation uses a combination of configuration space planner and *potential method* planner.

Fig. 1. Robot motion planning simulation

1.3 Path planning

A robot is a flexible mechanical device that can be a manupulator, an articulated hand, a wheled vehicule, a legged mechanical device, a flying platform or some combination of all the mentioned possibilities. It has a *work space* and then it is subject to the nature laws. It is autonomous in the sense that it has capability to plan automatically their movements. It is almost impossible to preview all the possible movements for performing a task. The more complex is the robot more critcal becomes the motion planning process. The motion planning

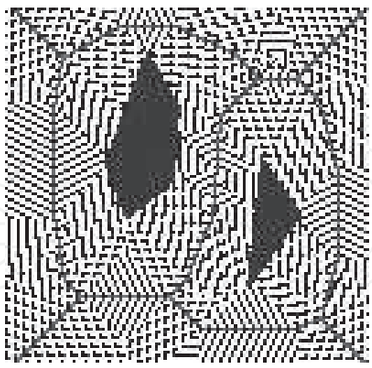


Fig. 2. Potential fields over a Voronoi diagram

is just one of the many aspects involved in the robot autonomy, the other could be for instance the real time control of the movement or the sensing aspects. It is clear that the motion planning is not a well defined problem. In fact it is a set of problems. These problems are variations of the robot motion planning problem whose computational complexity depend on the size of the dimension of the configration space where the robot is going to work, the presence of sensorial and/or control uncertainties and if the obstacles are fix or mobile. The robot motion navigation problems that I have treated in my own research are the following

- Robot motion planning under uncertainty
- Robot motion tracking
- Robot localization and map building

The methods and results obtained in my research are going to be explained in the following sections of this chapter.

2. Robot motion planning under uncertainty

As mentioned in the introduction section the robot motion planning become computionally more complex if the dimension of the configuration space grows. In the 80's many computationally efficient robot motion planning methods have been implemented for euclidean two dimensional workspace case, with plannar polygonal shaped obstacles and a robot having three degrees of freedom (Latombe et al. (1991)). The same methods worked quite well for the case of a 3D workspace with polyhedral obstacles and a manipulator robot with 6 articulations or degrees of freedom. In fact in this work (Latombe et al. (1991)) they proposed heuristically to reduce the manipulators degrees of freedom to 3 what gives a configuration space of dimension 3. By the same times it was proved in (Canny & Reif (1987); Schwartz & Sharir (1983)) that in the case of dealing with configuration spaces of dimension n or when obstacles in 2-dimensional work spaces move, the robot motion planning problem become computationally untractable ($NP-hard$, $NEXPTIME$, etc.). All those results were obtained under the hipothesis that the robot dont have to deal with sensorial uncertainties and that the robot actions were performed without deviations. The reality is not so nice and when those algorithms and methods were executed on real robots, many problems arised due to the uncertainties. The two most important sources of uncertainties were the sensors and

the actuators of the robot. The mobile robots are equiped with proximity sensors and cameras for trying to perform their actions without colliding on the walls or furniture that are placed on the offices or laboratories where the plans were to be executed. The proximity sensors are ultrasonic sensors that present sonar reflection problems and give unaccurate information about the presence or absence of obstacles. In figure 3 it is shown a simulation example, running over a simulator that we have implemented some years ago, of a mobile robot using a model of the sonar sensors. The planner used a *quadtree* for the division of the free space. It can be noticed in figure 3 that the information given by the sonar sensors is somehow noisy.

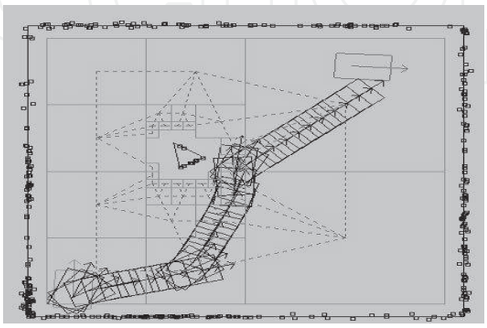


Fig. 3. Planner with sonar sensor simulation

The visual sensors present calibration problems and the treatment of 3D visual information some times can become very hard to deal with. If we take into account these uncertainties the motion planning problem become computationally complex even for the case of 2D robotic workspaces and configurations of low dimension (2D or 3D)(Papadimitriou (1985);Papadimitriou & Tsitsiklis (1987)). The motion planning problems that appear due to the sensorial uncertainies attracted many researches that proposed to make some abstractions of the sensors and use bayesian models to deal with it (Kirman et al. (1991); Dean & Wellman (1991); Marion et al. (1994)). In (Rodríguez-Lucatero (1997)) we study the three classic problems, evaluation, existence and optimization for the *reactive motion strategies* in the frame of a robot moving with uncertainty using various sensors, based on traversing colored graphs with a probabilistic transition model. We first show how to construct such graphs for geometrical scenes and various sensors. We then mention some complexity results obtained on evaluation, optimization and approximation to the optimal in the general case strategies, and at the end we give some hints about the approximability to the optimum for the case of reactive strategies. A planning problem can classically be seen as an optimum-path problem in a graph representing a geometrical environment, and can be solved in polynomial time as a function of the size of the graph. If we try to execute a plan π, given a starting point s and a terminating point t on a physical device such as a mobile robot, then the probability of success is extremely low simply because the mechanical device moves with uncertainty. If the environment is only partially known, then the probability of success is even lower. The robot needs to apply certain strategies to readjust itself using its sensors: in this paper, we define such strategies and a notion of *robustness* in order to compare various strategies. Concerning the research that we have done in (Rodríguez-Lucatero (1997)), the motion planning under uncertainty problem that interested us was the one that appears when there are deviations in execution of the commands given to the robot. These deviations produced robot postion uncertainties and the need to retrieve his real position by the use of landmarks in the robotic scene. For the sake of clarity in the exposition of the main ideas about motion planning under

uncertainty, we will define formally some of the problems mentioned. Following the seminal work of Schwartz and Sharir (Schwartz & Sharir (1991)), we look at the problem of *planning with uncertainty*, and study its computational complexity for graph-theoretical models, using the complexity classes BPP and IP. Given a graph with uncertainty, one looks at the complexity of a path problem in terms of the existence of a strategy of expectation greater than S (a threshold value). Such problems have been considered in (Valiant (1979);Papadimitriou (1985)) with slightly different probabilistic models, and the problem is #*P-complete* in Valiant's model, and *PSPACE-complete* in Papadimitriou's model.

2.1 Valiant's and Papadimitriou's model

Let $G =< V, E >$ be an oriented graph of domain V with $E \subseteq V^2$ the set of edges, and let $s, t \in V$ be given. Let $p(e)$ be the probability that the edge e exists: $p : E \rightarrow [0, 1]$, and let S be a numerical value, used as a threshold. The problem is to decide if the expectation to reach s from t is greater than S.
In (Valiant (1979)) it is shown that this problem is #P-complete, i.e. can't be solved in polynomial time, unless some unlikely complexity conjectures were true.
This problem with uncertainty is $PSPACE$, although not $PSPACE-$complete. In (Papadimitriou (1985)) a different probabilistic model is considered where the probability of edge-existence is more complex. Let $p : E.V \rightarrow [0, 1]$. $p(e, v)$ is the probability that e exists, when we are on v.
The problem DGR or *Dynamic Graph Reliability* is the decision problem where given G, s, t, S, p, we look for the existence of a strategy whose probability of success is greater than S.
DGR is $PSPACE$-complete, and is the prototype of problems that can be approached as *games against nature*.
In (Rodríguez-Lucatero (1997)), I considered a global model of uncertainty, and defined the notion of a *robust strategy*. We then give simple examples of robust and non-robust strategies, by evaluating the probability of success. This task can be quite complex on a large scene with unknown obstacles, and hence we wanted to study strategies that are easy to be evaluated and try to keep its level of performance by using sensors.
Under this colored graph model I defined the *existence of one coloration and one Markovian strategy* denoted as EPU and obtained some complexity results.

2.2 The colored graph model

In our model, the free space is represented with a labeled hypergraph in which the vertices are associated with both the robot's state and the expected sensor's measures in this state, and the label edges indicates the recommended action for reaching a vertex from one another.

2.2.1 The coloration method

In (Kirman et al. (1991)) a model of sensors is used to relate the theoretical model with the physical model. It is used for the description of specific strategies, which would reduce the uncertainty in a planning system. Our approach is interested in the comparison of different strategies, that include the ones described in (Kirman et al. (1991)).
Rather than using strictly quantitative measures from the sensors, we model with colors some qualitative features in the environement. It can be the detection of a more or less close wall from US sensors, the localisation of a landmark with vision .
So we defined for a graph $G = (V, E)$:

- **COLOR**, a finite set of colors,

- $clr : V \rightarrow$ **COLOR**, the coloration of the vertices.

In the case clr is bijective, we talk about *public uncertainty* : after each move, though it is uncertain, we know the exact position of the robot. Otherwise, we talk about *private uncertainty*. This distinction is essential, because the complexity of the studied problems depends on it.
When we want to represent a *real* scene using our model, we first proceed in a simple cell decomposition of the free space. For simplicity, we are choosing right now to use a grid as accessibility graph (generally 4 or 8- connected). Then we mark those cells with the measure (i.e. the color) expected by the sensors, eventually using *sensor data fusion* as described later.

2.2.1.1 Ultrasonic sensors

As the few ultrasonic sensors of our robot are not very reliable, we first choose an extremly simple model (1 in figure 4) with three different colors; the only thing we expect to detect is a local information : we can observe either **NOTHING**, a **WALL** or a **CORNER**. Being more confident, we then introduce an orientation criterion which bring us to a model (2 in figure 4) with nine colors;

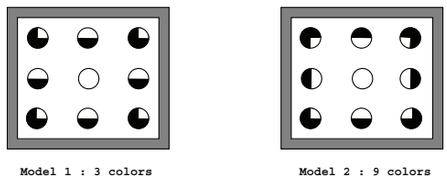


Fig. 4. Some simple models of US sensors

Many variations can be obtained by integrating some quantitative measures in qualitative concepts, like the colors previously described with a notion of *close* or *far*. For special types of graphs and related problems, many models have been introduced, one of them was presented in (Dean & Wellman (1991)),
Using the second model of coloration, we can obtain a scene such as figure 5. We first drew a grid on which we have suppressed the vertices occupied by obstacles. We then drew the color of the expected US measure on each vertex.

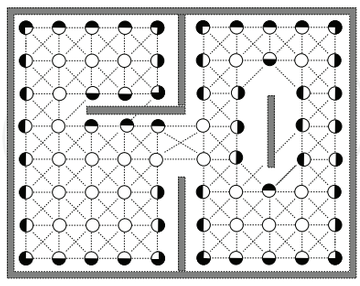


Fig. 5. A scene with two rooms and a door, using model 2 of US sensors

2.2.2 Moving with uncertainty

When executing an action, the new robot state can be different from the expected one. For this reason we use a hypergraph: each edge determines in fact a set of possible arriving vertices, with certain probabilities. *The uncertainty is then coded by a distribution probability over the labeled edges*.

In (Diaz-Frias (1991)) we can find a more formal definition of this model, in which two kinds of plans are considered: fixed plans and dynamical plans or strategies. In the rest of the section, we review the notion of *robust strategy* and we discuss the probability of robustness.
On a graph $G = (V, E)$, we define :

- $LABEL$, a finite set of basic command on G;
  on a 4-connected graph, for instance, we can have :
  $\mathbf{LABEL} = \{STOP, EAST, NORTH, WEST, SOUTH\}$,
- $lbl : V \times \mathbf{LABEL} \to V \vee \{FAIL\}$;

we then define the uncertainty on the moves by :

- $\delta : V \times \mathbf{LABEL} \times V \to [0,1]$; $\delta(v_i, l, v_j)$ is the probability beeing in $v_i \in V$, executing the command $l \in LABEL$, to arrive in $v_j \in V$. We assume δ is really a probability function, i.e. :

$$\forall v_i \in V, \forall l \in LABEL, \sum_{v_j \in V} \delta(v_i, l, v_j) = 1$$

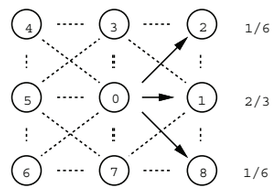


Fig. 6. An instance of the δ function : $\delta(0, EAST, 1) = \frac{2}{3}, \delta(0, EAST, 2) = \delta(0, EAST, 8) = \frac{1}{6}$ on an 8-connected grid.

2.2.3 Strategies

2.2.3.1 General case :

In the general case the strategies use his whole history for taking a decision. For more information see (Marion et al. (1994)) . We define the history, which is a record of all the actions and measures : $h \in H \subset (\mathbf{COLOR} \times LABEL)^*$. The history is first initialized with $clr(s)$; and then, during a realization, at the step t, the color of the current vertex is the last element of the history.

Definition 1. *A strategy with history, or* **H-strategy**, *is a function* $\sigma_H : H \to LABEL$.

2.2.3.2 Strategies without memory :

We then define two basic cases of strategies which are of interest because of their properties in the case of public uncertainty, and also because they are easy to evaluate.

Definition 2. *A Markov-strategy, or* **M-strategy**, *is a function* σ_M *:*

$$\sigma_M : \mathbf{COLOR} \to \mathbf{LABEL}$$

A Markov-strategy is a *time-independent* strategy. It depends only on the color of the current vertex. In the case of public uncertainty, it is a function of the current vertex.

Definition 3. *A* **T-strategy** *is a function* σ_T *:*

$$\sigma_T : \mathbf{COLOR} \times \mathbf{N} \rightarrow \mathbf{LABEL}$$

A T-strategy is a *time-dependent* strategy; It depends only of the color of the current vertex, and the number of steps. **remark :** A notion of **plan** is often defined. It is a particular kind of

T-strategy which depend only of the time : $\sigma_P : \mathbf{N} \rightarrow \mathbf{LABEL}$

For more details about other types of strategies more embeded that use a bounded memory see (Marion et al. (1994)).

2.2.3.3 Strategies using history :

If the two different strategies described above are efficient in some cases (as we will see in a further section, see section 4.2), other strategies using history can be more efficient. That is why we define a new kind of strategy named **D-strategy**.
For a graph $G = (V, E)$, a model of uncertainty δ, and a strategy σ, let's define at step k :

- $s_k \in V$ the position and h_k the history ; s_{k+1} is a function of $G, \delta, \sigma, h_k, s_k$, and $h_{k+1} = h_k \cup (col(s_{k+1}), \sigma(h_k))$.
- $\forall v \in V, f_{h_k}(v) = Pr(s_k = v \mid h_k)$, the probability of being in v at the step t, **knowing** the history.

At the step $k+1$, $f_{h_{k+1}}(v)$ is defined by :

$$[\xi_{h_{k+1}}(v) = \begin{cases} \sum_{u \in V} f_{h_k}(u)\delta(u, \sigma(h_k), v) & \text{if } col(v) = col(s_{k+1}) \\ 0 & \text{otherwise} \end{cases}] \; [f_{h_{k+1}}(v) = \frac{\xi_{h_{k+1}}(v)}{\sum_{u \in V} \xi_{h_{k+1}}(u)}]$$

Let be $\Phi : H \rightarrow [0,1]^{|V|}$, the function which associate for all h, $\Phi(h) = f_h$ the distribution over the vertices. We note $\mathcal{F} = \Phi(H)$.

Definition 4. *A* **D-strategy** *is a function :* $\sigma : \mathcal{F} \times \mathbf{N} \rightarrow \mathbf{LABEL}$

A D-strategy only depends on the time and the distribution over the vertices.

2.2.4 Criteria of evaluation of a strategy

2.2.4.1 Reliability :

We are first interested in reaching t from s with the maximal possible probability, but in a limited time k :

$$\mathbf{R}(\sigma, k) = Prob(s \xrightarrow{\sigma} t \mid \; |h| \leq k)$$

We note, at the limit:

$$\mathbf{R}(\sigma) = \mathbf{R}(\sigma, \infty) = \lim_{k \rightarrow \infty} \mathbf{R}(\sigma, k)$$

This criterion is essentially uses for M-strategy, for which we have means to compute this value (see section 3.1).

Definition 5. *A strategy* σ_{opt} *is* **R-k***-optimal iff :*

$$\forall \sigma \; : \mathbf{R}(\sigma, k) \leq \mathbf{R}(\sigma_{opt}, k)$$

Definition 6. *A strategy* σ_{opt} *is* **R-optimal** *iff :*

$$\forall \sigma \; : \mathbf{R}(\sigma) \leq \mathbf{R}(\sigma_{opt})$$

2.3 Definition of the problems

Given $G = (V, E)$, an uncertainty function δ, a coloration function clr, a command function lbl, two points $s, t \in G$ (resp. source and target), and a criterion $\mathcal{C}$, let us consider the following problem :

- **PU**, the decision problem
  Output : 1, if there exists a strategy which satisfies the criterion, else 0.
- **PU**$_{opt}$, the optimization problem
  Output : the optimal strategy for the criterion.

And for a given strategy σ :

- **PU**$_{\sigma}$, the evaluation problem
  Output : the value $\mathcal{C}(\sigma)$.

3. M-strategies vs.T-strategies

3.1 The public uncertainty case

In that case T-strategy and M-strategy are interesting :

Theorem 1. *A T-strategy* **R***-optimal for a given k does exist and can be constructed in time polynomial in k and the size of the input graph with uncertain moves.*

Theorem 2. *For every graph with uncertain moves and a source/target pair of vertices there exists an* **R***-optimal M-strategy.*

Note that the first theorem consider finite time criterion, and the second one infinite time criterion. (The demonstration of these theorems and the methods to construct those optimal strategies can be found in (Burago et al. (1993)).
Using these criteria we can compare different types of strategies under an unified frame. In (Marion et al. (1994)) we can find un exemple of graph where for a given number of steps a T-strategy works better than an M-strategy. In this same paper we showed that we can construct more involved strategies that can be more performants but harder to evaluate, so we proposed the simulation as a tool for estimating the performances of this kind of strategies.

3.2 Example: Peg-in-hole

We assume that we have a robot manipulator with a tactil sensor in the end effector. This sensor allows us to move compliantly over a planar surface. We suppose too that we have a workspace limited by a rectangular frame, so we can detect with the fingers of the end effector if we are touching the frame or an obstacle, by interpretation of the efforts in the fingers of the end effector. The robotic scene is as follows:

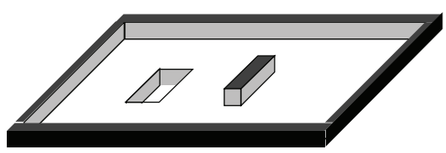


Fig. 7. A scene for the Peg-in-hole

If we use a sensorial model of the tactil sensor in a similar way as we used for the ultrasonic sensors we can obtain a colored graph representation like the next figure:

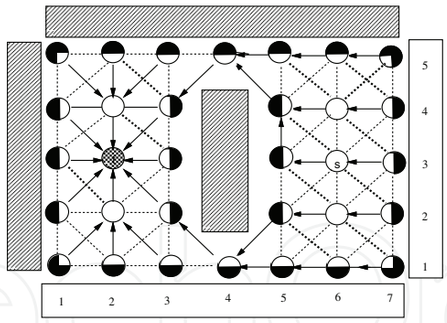


Fig. 8. Associated colored graph the arrows represents the M-strategy actions

If we evaluate this M-stretegy for a number of steps $k = 7$ we obtain: $R(\sigma_M, 20) = 0.16627$ as we can see this strategy is not performant even allowing a big number of steps. We can remark too that the coloration is not bijective so we can't distinguish between the limiting right frame and the right face of the rectangular obstacle. So we can propose a T-strategy (a little variation of the M-strategy) that for this color if $k \geq 5$ we execute the action E instead of making N The reliability for this new strategy is $R(\sigma_T, 20) = 0.8173582$ that makes a very big difference. In the case of finite polynomial time, there may not be an optimal M-strategy as shown on figure 9. In this example, the uncertainty is public. The command are RIGHT, LEFT and STRAIGHT, on the directed edges. The goal is to reach t from s in less than 6 steps. The moves are certain except for some edges :

- $\delta(s, RIGHT, 4) = \delta(s, RIGHT, 1) = \delta(s, LEFT, 4) = \delta(s, LEFT, 1) = \frac{1}{2}$,
- $\delta(8, LEFT, t) = \delta(8, LEFT, trap) = \frac{1}{2}$,

An *optimal* strategy first choose RIGHT to arrive at 4 in one step; but in case it fails, it will arrive there in four steps. An *optimal* M-strategy σ_M will **always** choose on vertex 4 :

- either LEFT, taking the risk not to arrive before the time is gone (6 steps maximum),
- either RIGHT, taking the risk to fall in the trap.

The optimal T-strategy σ_T will choose on vertex 4 :

- LEFT a (the safe way) if it arrive there in only one step,
- otherwise RIGHT (a more risky path).

Thus the probability of success of those two strategies are : $[\mathbf{R}(\sigma_M, 6) = \frac{1}{2} \leq \mathbf{R}(\sigma_T, 6) = \frac{3}{4}]$

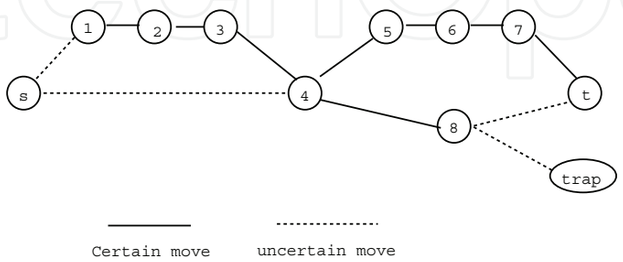


Fig. 9. σ_T is optimal though σ_M is not.

3.3 The private uncertainty case

In the case of total uncertainty (i.e. all the vertices have the same color),

Theorem 3. *It is $NP-hard$ to decide if there exists a strategy which succeeds with probability 1.*

Another result that we consider interesting concerns the approximability of this problem in the case of total uncertainty, that we can state as:

Theorem 4. *It is $NP-hard$ to decide if there exists an approximate solution.*

4. Complexity of the evalution problem

4.1 Evaluation in the general case

The computation of $R(\sigma, k)$ for some strategy σ may be very hard. This can be shown by a reduction of $3SAT$ to the evaluation problem. We represent F as a table. Let be:

$$F = \bigwedge_{1 \leq i \leq m} \bigvee_{1 \leq j \leq 3} z_{i,j}$$

a formula where $z_{i,j}$ are literals of the set

$$\{x_1, \ldots, x_n, \overline{x_1}, \ldots, \overline{x_n}\})$$

. His tabular representation is

$$\begin{array}{cccc} z_{1,1} & z_{2,1} & \cdots & z_{m,1} \\ z_{1,2} & z_{2,2} & \cdots & z_{m,2} \\ z_{1,3} & z_{2,3} & \cdots & z_{m,3} \end{array}$$

where the height is 3 and the length is m (the i-th column corresponds to the i-th clause in the formula). We say that two literals z_1 et z_2 are opposed iff $z_1 \Leftrightarrow \overline{z_2}$. We assume that there is not contradictory literals in a column. One path in F is an horizontal path P in the table build taking one literal by column (clause), we mean that P is a list of literals as $(z_{1,j_1}, z_{2,j_2}, \ldots, z_{m,j_m})$, $1 \leq j_i \leq 3$, $1 \leq i \leq m$. We interpret this paths as truth value assignments. In the case that a path have no pair of contradictory literals we say that is a model of the logic formula. The paths without contradictions are named as *opened* otherwise *closed*.
In this way we can construct a graph as:

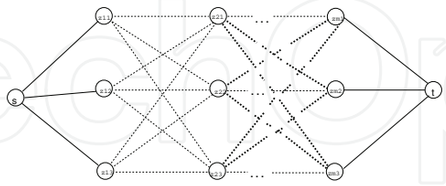


Fig. 10. Example multilayer graph for the reduction of 3SAT to PU

The triplets of vertex in the vertical sense represents the columns of the table. The dashed lines are probabilistic transitions of the strategy σ, that is, when the strategy takes a dashed line he arrives to one of the vertex in the next layer with a probability of $1/3$ it doesn't matter what edge has been taken.
In the case of continue lines, they are safe lines, that is if the strategy takes it, she follows it certainly. The strategy selects the edge seeing the walked path (i.e. a prefix of a path in F). If at this moment the path is an *opened* one the strategy takes a dashed line (i.e. makes a random mouvement), otherwise it takes a safe line going through the *trap*. If the strategy arrives to the

last layer by an *opened* path, then it transits to the goal by a safe line. We conclude that if F satisfaied then $R(\sigma, k) > 0$.
Before this result, the evaluation problem is a hard one in the general case. Even that we can try to work with strategies that are easy to evaluate as the M-strategies and T-strategies.

4.2 Evaluation of M-strategy & T-strategy

Theorem 5. *Computing* $\mathbf{R}(\sigma, k)$ *if* σ *is a M-strategy, or a T-strategy which stops in polynomial time* k *, can be done in polynomial time (idem* $\mathbf{E}(\sigma, k)$ *).*

proof : It follows from Markov chain theory : Let us note μ_k the distribution over the vertices at step k. $\mu_0(s) = 1$.
A M-strategy σ_M can be seen as a transition matrix $\mathbf{M} = [m_{ij}]$. If the decision of the M-strategy in i is $l(i)$: from i, move to j.

$$m_{ik} = \begin{cases} 1 & \text{if } k = j, \\ 0 & \text{otherwise} \end{cases}$$

We compute $\mathbf{P} = [p_{ij}]$ with : $p_{ij} = \delta(i, l(i), j)$ Then, $\forall i \in \mathbf{N} : \mu_{i+1} = \mathbf{P}\mu_i$ and $\mathbf{R}(\sigma_M, k) = \mu_k$.

We can do the same thing with a T-strategy σ_T, except that in this case, the decision depends also of the time : $l(i, t)$. Then we define $\mathbf{P}(t)$ in the same way, and :
$\forall i \in \mathbf{N} : \mu_{i+1} = \mathbf{P}(i)\mu_i$ and $\mathbf{R}(\sigma_T, k) = \mu_k$. □

Theorem 6. *The problems of evaluating a M-strategy for a infinite time criterion can be solved in polynomial time.*

proof : An other result from Markov chain theory : we just compute the stationary distribution over the vertices π (ie. solve $P\pi = \pi$). □

4.3 Definition of EPU and complexity

One other approach for trying to deal with this problem is to explore the posibility of working with strategies that are easy to evaluate (M-strategies) and use some fixed amount of colours for keeping his performance. In this section we deal with this problem and give some complexity results about that.

Definition 7. *Problem* **KNAPSACK***: to find a subset of a set where each element is affected by two values, one given by a size function s and the other given by a weight function v. The addition of all the sizes in this sub-set must be lesser than a given value and the addition of the weights bigger than another given value.*
INPUT: $U = \{u_1, \ldots, u_n\}$ *a function* $s : u \in U \to Z^+$ *a function* $v : u \in U \to Z^+$ *two integers* B *et* K
OUTPUT: *1 if* $\exists U' \subset U$*, such that* $\sum_{u' \in U'} s(u') \leq B$ *and* $\sum_{u' \in U'} v(u') \geq K$*, and* 0 *otherwise.*

Definition 8. EPU: *problem of the existence of one coloration and one M-strategy given a fixed number of colors and a threshold.*
INPUT: $G(V, E), s, t \in V, k, q \in Q, T, \mu$
OUTPUT: *1 if* $\exists clr : v \in V \to \{1, \ldots, k\} : \exists \sigma_M$ *such that* $R(\sigma_M, T) \geq q$ *, and* 0 *otherwise.*

Theorem 7. EPU *is NP-complet*

proof : We show it by a reduction of **KNAPSACK** to **EPU**. It belongs to **NP** because if we want to obtain a **M-**strategy with T steps, given k colours and a threshold, we colour the graph randomly an associate an action to each color. Based on this, we calculate a Markov matrix an we evaluate the strategy in polynomial time. In this way we prouve that $EPU \in NP$.
The goal is to find a polynomial transformation of **KNAPSACK** to **EPU**. For this end we build 2 graphs having $n+2$ vertices as we show in the figure 11.

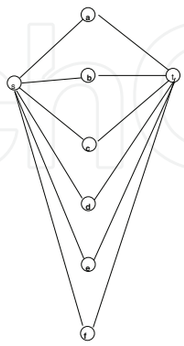


Fig. 11. Example of the graph for knapsack to EPU

We associate each graph to each restriction of **KNAPSACK**.
As it can be verified in the figure 11 we defined a graph that has one layer that contains n vertices that we name selection points.
This layer associates each selection vertex to an element of the **KNAPSACK** set U. We have two additional vertices s for the starting point and t for the goal. We draw n edges between s and each element of the selection layer, and we assign a uniform probability $1/n$ to each one.
Similarly we define an edge going from each element of the selection layer to the goal vertex t. Then we fix the number of steps $T = 2$, one for going from s to the selection layer an the other one for going from the selection layer to the goal vertex t
We fix the nomber of available colors k to 2 because we want to interpret the selection of elements in U as an assignement of colors and associate to each color one action. Next we define a probabilistic deviation model in function of the associated weights of elements for the second restriction of **KNAPSACK** as follows:

$$\forall 1 \leq i \leq B \ (u(i),t) \ \in E\mu_1(u,\sigma_M(clr(u(i))),t) = \begin{cases} p_1 = \frac{u(i)}{V} \\ 1-p_1 \quad \text{trap} \end{cases}$$

As we can see we introduced a parameter **V** that represents the total weight of the set U. This give us probability values between 0 and 1.
In the same way we define for the first restriction of **KNAPSACK** a probabilistic deviation model in function of the sizes associated to each element of the U as follows:

$$\forall 1 \leq i \leq B \ (u(i),t) \ \in E\mu_2(u,\sigma_M(clr(u(i))),t) = \begin{cases} p_2 = \frac{s(i)}{S} \ \text{trap} \\ 1-p_2 \end{cases}$$

As we can see we introduced a parameter **S** that represents the total size of the set U. This give us probability values between 0 and 1.
We have 2 label actions : one for **move to t** and the other for **stop**. Next we relate the color **selected** and the action **move to t** and the color not selected with the action **stop**.
For finishing the transformation we relate the thresholds q_1 et q_2 for each graph with the parameters of the **KNAPSACK** as follows:

$$q_1 = \frac{\mathbf{K}}{n \times V}$$

and

$$q_2 = 1 - \frac{\mathbf{B}}{n \times S}$$

Remark : In the definition of the distribution μ_1 we talked about a probability to get traped equal to $1 - p$, and a probability p to have succes in the mouvement. So we cant arrive to the goal since a vertex coloured with **non selected**. The same is valid for μ_2 too.
Before that we have for the first graph:
$R_1(\sigma_M, T) = \sum_{\pi \in PATHS} \mathbf{Prob}(\pi \text{ réalisation } de\ \sigma_M) =$
$\sum_{i=0}^{T} \sum_{j=1}^{n} \mu_1(i, \sigma_M, j)$
and for the second one:

$$R_2(\sigma_M, T) = \sum_{\pi \in PATHS} \mathbf{Prob}(\pi \text{ réalisation } de\ \sigma_M) = \sum_{i=0}^{T} \sum_{j=1}^{n} \mu_2(i, \sigma_M, j)$$

So we can proclaim that it exists a subset for the **KNAPSACK** iff it exists a colouring clr and an associated strategy σ_M that fullfils the performance q_1 for the first graph an simultaneously q_2 for the second one. That is that we have shown that **KNAPSACK** $\Rightarrow EPU$.
For showing the reduction in the oposite sense we say that if we have one colouring and an $M-$strategy associated that fullfils the performace q_1 and q_2 en each graph then it exist a subset $U' \subset U$ for **KNAPSACK**. For this we only need to label vertex of the selection layer with one element of the U and take the vertices that have a **selected** color. This gives a subset that works for **KNAPSACK**. In this way we prouved that **KNAPSACK** $\Leftarrow EPU$ and we can proclaim that:

$$\mathbf{KNAPSACK} \Leftrightarrow EPU$$

□

This result show that EPU is a hard one but what is intresting is that $KNAPSACK$ is one of the rare problems that are **NP-complet** and at the same time arbitrarly approximable. That is that $KNAPSACK$ has a fully polynomial approximation schema (**FPTAS**). So if we can make a special kind of transformation from EPU to $KNAPSACK$ called $L - reduction$ (for details see (Papadimitriou & Yannakakis (1991)) and (Papadimitriou & Steiglitz (1982))), we can find good approximations to the optimal for the EPU optimazing problem.

5. Robot motion tracking, DFA learning, sketching and streaming

Another robot navigation problem that has attracted my attention in the last six years has been the robot tracking problem. In this problem we are going to deal with another kind of uncertainty. The uncertainty on the setting of two robots that, one that plays de rôle of *observer* and the other that of *target*. This problem has to do with other kind of uncertainty that appears in robot navigation problems. The uncertainty on the knowledge of the other reactions in a given situation. This situation arise when there are two or more robots or agents in general that have to perform their tasks in the same time and to share the working space. Many everyday life situations can be seen as an interaction among agents, as can be to play football , to drive a car in Mexico city streets, or to play chess with your wife. The robot tracking is not the exception. In the examples given above as well as in the robot tracking case, the agents observe the actions taken by the other agents, and try to predict the behaviour of them and react in the best possible way. The fundamental difference between the examples that we have given and the robot tracking problem is that the agents in the last case are robots and as consequence they are limited in computational power. Because of

that we proposed in (Lucatero et al. (2004)) to pose the robot tracking problem as a repeated game. The main motivation of this proposal was that in many recent articles on the robot tracking problem (La Valle & Latombe (1997) La Valle & Motwani (1997)) and (Murrieta-Cid & Tovar (2002)) they make the assumption that the strategy of the target robot is to evade the *observer robot* and based on that they propose geometrical and probabilistic solutions of the tracking problem which consists on trying to maximize, by the *observer*, the minimal distance of escape of the *target*. We feel that this solution is limited at least in two aspects. First the target don't interact with the *observer* so there is no evidence that the strategy of the *target* will be to try to escape if it doesn't knows what are the actions taken by the observer. The second aspect is that even if it take place some sort of interaction between the *target* and the *observer*, the *target* is not necessarily following an evasion strategy so this may produce a failure on the tracking task. Because of that we proposed a DFA learning algorithm followed by each robot and obtained some performance improvements with respect to the results obtained by the methods used in (Murrieta-Cid & Tovar (2002)). In the last few years many research efforts have been done in the design and construction of efficient algorithms for reconstructing unknown robotic environments (Angluin & Zhu (1996);Rivest & Schapire (1993);Blum & Schieber (1991);Lumelsky & Stepanov (1987)) and apply learning algorithms for this end (Angluin & Zhu (1996);Rivest & Schapire (1993)). One computational complexity obstacle for obtaining efficient learning algorithms is related with the fact of being a passive or an active learner. In the first case it has been shown that it is impossible to obtain an efficient algorithm in the worst case (Kearns & Valiant (1989);Pitt & Warmuth (1993)). In the second case if we permit the learner to make some questions (i.e. to be an active learner) we can obtain efficient learning algorithms (Angluin (1981)) . This work done on the DFA learning area has given place to many excellent articles on learning models of intelligent agents as those elaborated by David Carmel and Shaul Markovitch (Carmel & Markovitch (1996);Carmel & Markovitch (1998)) and in the field of Multi-agent Systems those written about Markov games as a framework for multi-agent reinforcement learning by M.L. Littman (Littman (1994)). In (Lucatero et al. (2004)) we proposed to model the robot motion tracking problem as a repeated game. So, given that the agents involved have limited rationality, it can be assumed that they are following a behaviour controled by an automata. Because of that we can adapt the learning automata algorithm proposed in (Carmel & Markovitch (1996)) to the case of the robot motion tracking problem. In (Lucatero et al. (2004)) we assume that each robot is aware of the other robot actions, and that the strategies or preferences of decision of each agent are private. It is assumed too that each robot keeps a model of the behavior of the other robot. The strategy of each robot is adaptive in the sense that a robot modifies his model about the other robot such that the first should look for the best response strategy w.r.t. its utility function. Given that the search of optimal strategies in the strategy space is very complex when the agents have bounded rationality it has been proven in (Rubinstein (1986)) that this task can be simplified if we assume that each agent follow a Deterministic Finite Automate (DFA) behaviour. In (Papadimitriou & Tsitsiklis (1987)) it has been proven that given a DFA opponent model, there exist a best response DFA that can be calculated in polynomial time. In the field of computational learning theory it has been proven by E.M. Gold (Gold (1978)) that the problem of learning minimum state DFA equivalent to an unknown target is NP-hard. Nevertheless D. Angluin has proposed in (Angluin (1981)) a supervised learning algorithm called ID which learns a target DFA given a live-complete sample and a knowledgeable teacher to answer membership queries posed by the learner. Later Rajesh Parekh, Codrin Nichitiu and Vasant Honavar proposed in (Parekh & Honavar (1998)) a polynomial time incremental algorithm for

learning DFA. That algorithm seems to us well adapted for the tracking problem because the robots have to learn incrementally the other robot strategy by taking as source of examples the visibility information and the history of the actions performed by each agent. So, in (Lucatero et al. (2004)) we implemented a DFA learning that learned an aproximate DFA followed by the othe agent. For testing the performance of that algorithm it was necesary the creation of an automata for playing the role of target robot strategy, with a predefined behavior, and to watch the learning performance on the observer robot of the target robot mouvements. The proposed target robot behavior was a wall-follower. The purpose of this automata is to give an example that will help us to test the algorithm, because in fact the algorithm can learn other target automatas fixing the adequate constraints. The target automata strategy was simply to move freely to the North while the way was free, and at the detection of a wall to follow it in a clockwise sense. Besides the simplicity of the automata we need a discretization on the possible actions for being able to build the automata. For that reason we have to define some constraints. The first was the discretization of the directions to 8 possibilities (N, NW, W, SW, S, SE, E, NE). The second constraint is on the discretization of the possible situations that will become inputs to the automata of both robots. It must be clearly defined for each behavior what will be the input alphabet to which will react both robots. This can be done without modifying the algorithm The size of the input alphabet afect directly the learning algorithm performance, because it evaluates for each case all possible course of action. So, the table used for learning grows proportionaly to the number of elements of the input alphabet. It is worth mentioning that in the simulation we used, to compare with our method, an algorithm inspired on the geomety based methods proposed in (La Valle & Latombe (1997); La Valle & Motwani (1997)) and (Murrieta-Cid & Tovar (2002)). In this investigation, we have shown that the one-observer-robot/one-target-robot tracking problem can be solved satisfactorily using DFA learning algorithms inspired in the formulation of the robot motion tracking as a two-player repeated game and enable us to analyse it in a more general setting than the evader/pursuer case. The prediction of the target movements can be done for more general target behaviours than the evasion one, endowing the agents with learning DFA's abilities. So, roughly speaking we have shown that learning an approximate or non minimal DFA in this setting was factible in polynomial time. The question that arises is, *how near is the obtained DFA to the minimal one ?*. This problem can reduces to the problem of automata equivalece. For giving an answer to this question we have used the sketching and streaming algorihms. This will be developped in the following subsection.

5.1 DFA equivalence testing via sketch and stream algorithms

Many advances have been recently taking place in the approximation of several classical combinatorial problems on strings in the context of *Property Testing* (Magniez & de Rougemont (2004)) inspired on the notion of *Self-Testing* (Blum & Kannan S. (1995); Blum et al. (1993); Rubinfeld & Sudan (1993)). What has been shown in (Magniez & de Rougemont (2004)) is that, based on a statistical embedding of words, and constructing a tolerant tester for the equality of two words, it is possible to obtain an approximate normalized distance algorithm whose complexity don't depend on the size of the string. In the same paper (Magniez & de Rougemont (2004)) the embedding is extended to languages and get a geometrical approximate description of regular languages consisting in a finite union of polytopes. As an application of that its is obtained a new tester for regular languages whose complexity does not depend on the automaton. Based on the geometrical description just mentioned it is obtained an deterministic polynomial equivalent-tester for regular languages for a fixed

threshold distance. Computing *edit distance* between two words is an important subproblem of many applications like text-processing, genomics and web searching. Another field in Computer Science that where important advances recently have been taking place is that of embeddings of sequences (Graham (2003)). The sequences are fundamental objects in computer science because they can represent vectors, strings, sets and permutations. For beeing able to measure their similarity a distance among sequences is needed. Sometimes the sequences to be compared are very large so it is convenient to map or embed them in a different space where the distance in that space is an approximation of the distance in the original space. Many embeddings are computable under the streaming model where the data is too large to store in memory, and has to be processed as and when it arrives piece by piece. One fundamental notion introduced in the approximation of combinatorial objects context is the *edit-distance*. This concept can be defined as follows:

Definition 9. *The* **edit distance** *between two words is the minimal number of character substitutions to transform one word into the other. Then two words of size n are ϵ-far if they are at distance greater than ϵn.*

Another very important concept is the *property testing*. The *property testing* notion introduced in the context of program testing is one of the foundations of our research. If **K** is a class of finite structures and P is a property over **K**, we wish to find a **Tester**, in other words, given a structure U of **K**:

- It can be that U satisfy P.
- It can be that U is ϵ-far from P, that means, that the minimal distance between U and U' that satisfy P is greater than .
- The randomized algorithm runs in $O(\epsilon)$ time independently of n, where n represent, the size of the structure U.

Formally an ϵ-tester can be defined as follows.

Definition 10. *An ϵ-tester for a class $K_0 \subseteq K$ is randomized algorithm which takes a structure U_n of size n as input and decides if $U_n \in K_0$ or if the probability that U_n is ϵ-far from K_0 is large. A class K_0 is testable if for every sufficiently small ϵ there exists an ϵ-tester for K_0 whose time complexity is in $O(f(\epsilon))$, i.e. independent of n*

For instance, if **K** is the class of graphs and P is a property of being colorable, it is wanted to decide if a graph U of size n is 3-colorable or ϵ-far of being 3-colorable, i.e. the Hamming distance between U and U' is greater than $\epsilon \cdot n^2$. If **K** is the class of binary strings and P is a regular property (defined by an automata), it is wished to decide if a word U of size n is accepted by the automata or it is ϵ-far from being accepted, i.e., the *Edition distance* between U and U' is greater than $\epsilon \cdot n$. In both cases, it exists a **tester**, that is, an algorithm in that case take constant time, that depends only on and that decide the proximity of these properties. In the same way it can be obtained a **corrector** that in the case that U does not satisfy P and that U is not ϵ-far, finds a structure U' that satisfy P. The existence of **testers** allow us to approximate efficiently a big number of combinatorial problems for some privileged distances. As an example, we can estimate the distance of two words of size n by means of the *Edition distance* with shift, we mean, when it is authorized the shift of a sub-word of arbitrary size in one step. To obtain the distance it is enough to randomly sample the sub-words between two words, to observe the statistics of the random sub-words and to compare with the L_1 norm. In a general setting, it is possible to define distances between automata and to quickly test if two automata

are near knowing that the exact problem is NEXPTIME hard. By other side, an important concept that is very important in the context of sequence embeddings is the notion of *sketch*. A *sketch* algorithm for *edit distance* consit of two compression procedures, that produce a finger print or *sketch* from each input string, and a reconstrucction procedure that uses the sketches for approximating the *edit distance* between the to strings. A sketch model of computation can be described informally as a model where given an object x a shorter *sketch* x can be made so that compairing to *sketches* allow a function of the original objects to be approximated. Normally the function to be approximated is the distance. This allow efficient solutions of the next problems:

- Fast computation of short *sketches* in a variety of computing models, wich allow sequences to be comapred in constant time and spaces non depending on the size of the original sequences.
- Approximate nearest neighbor and clustering problems faster than the exact solutions.
- Algorithms to find approximate occurrences of pattern sequences in long text sequences in linear time.
- Efficient communication schemes to approximate the distance between, and exchange, sequences in close to the optimal amount of communication.

Definition 11. *A distance sketch function $sk(a,r)$ with parameters ϵ, δ has the property that for a distance $d(\cdot,\cdot)$, a specified deterministic function f outputs a random variable $f(sk(a,r),sk(b,r))$ so that*

$$\begin{aligned}(1-\epsilon)d(f(sk(a,r),sk(b,r)))\\ \leq f(sk(a,r),sk(b,r))\\ \leq (1+\epsilon)d(f(sk(a,r),sk(b,r)))\end{aligned}$$

for any pairs of points a and b with probability $1-\delta$ taken over small choices of a small seed r chosen uniformly at random

The sketching model assumes complete access to a part of the input. An alternate model is the streaming model, in which the computation has limited access to the whole data. In that model the data arrive as a stream but the space for storage for keeping the information is limited.

6. Applications to robot navigation problems

As I mentioned in section 5 one automata model based aproach for solving the robot motion tracking has been proposed in (Lucatero & Espinosa (2005)). The problem consited in building a model in each robot of the navigation behaviour of the other robot under the assumption that both robots, target and observer, were following an automata behaviour. Once the approximate behaviour automata has been obtained the question that arises is, *how can be measured the compliance of this automata with automata followed by the target robot ?*. Stated otherwise *How can be tested that the automata of the target robot is equivalent to the one obtained by the observer robot ?* Is exactly in that context that the property testing algorithms can be applied for testing the equivalence automata in a computationally efficient way. It is well known that the problem of determining equivalence between automatas is hard computationally as was mentioned in section 5. The map learning can be as well formulated as an automata with stochastic output inferring problem (Dean et al. (1985)). It can be usefull to compare

the real automata describing the map of the environment and the information inferred by the sensorial information. This can be reduced to the equivalence automata problem, and for this reason, an approximate property testing algorithm can be applied. In (Goldreich et al. (1996)) can be found non obvious relations between property testing and learnability. As can be noted testing mimimics the standar frameworks of learning theory. In both cases one given access to an unknown *target* function. However there are important differences between testing and learning. In the case of a learning algorithm the goal is to find an approximation of a function $f \in K_0$, whereas in the case of testing the goal is to test that $f \in K_0$. Apparently it is harder to learn a property than to test it. (Goldreich et al. (1996)) it shown that there are some functions class which are harder to test than to learn provided that $NP \not\subset BPP$. In (Goldreich et al. (1996)) when they speak about the complexity of random testing algorithms they are talking about query complexity (number test over the input) as well as time complexity (number of steps) and hey show there that both types of complexities depend polynomially only on ϵ not on n for some properties on graphs as colorability, clique, cut and bisection. Their definition of property testing is inspired on the PAC-learning model (Valiant (1984)), so there it is considered de case of testers that take randomly chosen instances with arbitrarly distribution instead of querying. Taking into account the progress on property testing mentioned , the results that will be defined further can be applied to the problem of testing how well the automata inferred by the *observer robot* in the robot motion tracking problem solved in (Lucatero & Espinosa (2005)), fits the behaviour automata followed by the *target robot*. The same results can be applied to measure how much the automata obtained by explorations fits the automata that describes the space explored by the robot. Roughly speaking, the equivalence ϵ-tester for regular languages obtained in (Fisher et al. (2004)), makes a statistical embedding of regular languages to a vectorial statistical space which is an approximate geometrical description of regular languages as a finite union of polytopes. That embedding enables to approximate the *edit distance* of the original space by the ϵ-tester under a *sketch* calculation model. The automata is only required in a preprocessing step, so the ϵ-tester does not depend on the number of states of the automata. Before stating the results some specific notions must be defined

Definition 12. Block statistics. *Let w and w' two word in Σ each one of length n such that k dived n. Let $\epsilon = \frac{1}{k}$. The statitistics of block letters of w denoted as $b - stat(w)$ is a vector of dimension $|\Sigma|^k$ such that its u coordinate for $u \in \Sigma^k$ (Σ^k is called the block alphabet and its elements are the block letters) satisfies $b - stat(w)[u] \stackrel{def}{=} Pr_{j=1,\ldots,n/k}[w[j]_b = u]$ Then $b - sta(w)$ is called the block statistics of w*

A convenient way to define block statistics is to use the underlying distribution of word over Σ of size k that is on block letter on Σ^k. Then a uniform distribution on block letters $w[1]_b, w[2]_b, \ldots, w[\frac{n}{k}]_b$ of is the block distribution of w. Let X be a random vector of size $|\Sigma|^k$ where all the coordinates are 0 except its u-coordinate which is 1, where u is the index of the random word of size k that was chosen according to the block distribution of w. Then the expectation of X satisfies $E(X) = b - stat(w)$. The *edit distance* with moves between two word $w, w' \in \Sigma$ denoted as $dist(w, w')$ is the mininimal number of elementary operations on w to obtain w'. A class $K_0 \in K$ is testable if for every $\epsilon > 0$, there exists an ϵ-tester whose time complexity depends only on ϵ.

Definition 13. *Let $\epsilon \geq 0$. Let $K_1, K_2 \subseteq K$ two classes. K_1 is ϵ-contained in K_2 if every but finitely many structures of K_1 are ϵ-close to K_2. K_1 is ϵ-equivalent to K_2 if K_1 is ϵ-contained in K_2 and K_2 is ϵ-contained in K_1*

The following results that we are going to apply in the robotics field, are stated without demonstration but they can be consulted in (Fisher et al. (2004)).

Lemma 1. .-

$$dist(w, w') \leq \left(\frac{1}{2} \left| b - stat(w) - bstat(w') \right| + \epsilon \right) \times n$$

So we can embed a word w into its block statistics $b - stat(w) \in \Re^{|\Sigma|^{1/\epsilon}}$

Theorem 8. *For every real $\epsilon > 0$ and regular language L over a finite alphabet Σ there exists an ϵ-tester for L whose query complexity is $O(\frac{\lg |\Sigma|}{\epsilon^4})$ and time complexity $2^{|\Sigma|^{O(1/\epsilon)}}$*

Theorem 9. *There exists a deterministic algorithm T such that given two autimata A and B over a finite alphabet Σ with at most m states and a real $\epsilon > 0$, $T(A, B, \epsilon)$*

1. *accepts if A and B recognize the same language*
2. *rejects if A and B recognize languages that are not ϵ-equivalent. Moreover the time complexity of T is in $m^{|\Sigma|^{O(1/\epsilon)}}$*

Now based on 9 our main result can be stated as a theorem.

Theorem 10. *The level of approximability of the inferred behaviour automata of a* **target robot** *by an* **observer robot** *with respect to the real automata followed by the* **target robot** *in the motion tracking problem can be tested efficiently.*

Theorem 11. *The level of approximability of the sensorialy inferred automata of the* **environment** *by an* **explorator robot** *with respect to the real environment automata can be tested efficiently.*

7. Application of streaming algorithms on robot navigation problems

The starting premise of the sketching model is that we have complete access to one part of the input data. That is not the case when a robot is trying to build a map of the environemet based on the information gathered by their sensors. An alternative calculation model is the streaming model. Under this model the data arrives as a stream or predetermined sequence and the information can be stored in a limited amount of memory. Additionally we cannot backtrack over the information stream, but instead, each item must be processed in turn. Thus a stream is a sequence of n data items $z = (s_1, s_2, \ldots, s_n)$ that arrive sequentially and in that order. Sometimes, the nomber n of items is known in advance and some other times the last item s_{n+1} is used as an ending mark of the data stream. Data streams are fundamental to many other data processing applications as can be the atmospheric forecasting measurement, telecommunication network elements operation recording, stock market information updates, or emerging sensor networks as highway traffic conditions. Frequently the data streams are generated by geografically distributed information sources. Despite the increasing capacity of storage devices, it is not a good idea to store the data streams because even a simple processing operation, as can be to sort the incoming data, becomes very expensive in time terms. Then, the data streams are normally processed *on the fly* as they are produced. The stream model can be subdivided in various categories depending on the arrival order of the attributes and if they are aggregated or not. We assume that each element in the stream will be a pair $\langle i, j \rangle$ that indicates for a sequence a we have $a[i] = j$.

Definition 14. *A streaming algorithm accepts a data stream z and outpus a random variable $str(z, r)$ to approximate a function g so that*

$$(1-\epsilon)g(z) \leq str(z,r) \leq (1-\epsilon)g(z)$$

with probability $1-\delta$ *over all choices of the random seed* r*, for parameters* ϵ *and* δ

The streaming models can be adapted for some distances functions. Let suppose tha z consists of two interleaved sequences, a and b, and that $g(z) = d(a,b)$. Then the streaming algorithm to solve this proble approximates the distance between a and b. It is possible that the algorithm can can work in the sketching model as well as in the streaming model. Very frequently a streaming algorithm can be initially conceived as a sketching one, if it is supposed that the sketch is the contents of the storage memory for the streaming algorithm. However, a sketch algorithm is not necesarilly a streaming algorithm, and a streaming algorithm is not always a sketching algorithm. So, the goal of the use of this kind of algorithms, is to test equality between two object, approximately and in an efficient way.
Another advantage of using *fingerprints* is that they are integers that can be represented in $O(\log n)$ bits. In the commonly used RAM calculation model it is assumed that this kind of quantities can be worked with in $O(1)$ time. This quantities can be used for building has tables allowing fast access to them without the use of special complex data structures or sorting preprocessing. Approximation of L_p distances can be considered that fit well with sketching model as well as with the streaming model. Initially it can be suppossed that the vectors are formed of positive integers bounded by a constant, but it can be extended the results to the case of rational entries. An important property possesed by the sketches of vectors is the *composability*, that can be defined as follows:

Definition 15. *A sketch function is said to be composable if for any pair of sketches* $sk(a,r)$ *and* $sk(b,r)$ *we have that* $sk(a+b,r) = sk(a,r) + sk(b,r)$

One theoretical justification that enables us to embed an Euclidean vector space in a much smaller space with a small loss in accuracy is the Johnson-Lindenstrauss lema that can be stated as follows:

Lemma 2. *.- Let* a, b *be vectors of length* n*. Let* v *be a set of* k *different random vectors of length* n*. Each component* $v_{i,j}$ *is picked independently from de Gaussian distribution* $N(0,1)$*, then each vector* v_i *is normalised under the* L_2 *norm so that the magnitude of* v_i *is* 1*. Define the sketch of a to be a vector* $sk(a,r)$ *of length k so that* $sk(a,r)_i = \sum_{j=1}^{n} v_{i,j}a_j = v_i \cdot a$*. Given parameters* δ *and* ϵ*, we have with probability* $1-\delta$

$$\frac{(1-\epsilon)\|a-b\|_2^2}{n} \leq \frac{\|sk(a,r)-sk(b,r)\|_2^2}{k} \leq \frac{(1+\epsilon)\|a-b\|_2^2}{n}$$

where k is $O(1/\epsilon^2 \log 1/\delta)$

This lemma means that we can make a sketch of dimension smaller that $O(1/\epsilon^2 \log 1/\delta)$, from the convolution of each vector with a set of randomly created vectors drawn from the Normal distribution. So, this lemma enable us to map m vectors into a reduced dimension space. The sketching procedure cannot be assimilated directly to a streaming procedure, but it has been shown recently how to extend the sketching approach to the streaming environement for L_1 and L_2 distances. Concerning streaming algorithms, some of the first have been published in (**?**) for calculating the frequency moments. In this case, we have an unordered and unaggregated stream of n integers in the range of $1,\ldots,M$, such that $z = (s_1, s_2, \ldots, s_n)$ for

integers s_j. So, in (**?**) the authors focus on calculating the frequency moments F_k of the stream. Let it be, from the stream, $m_i = |\{j|s_j = i\}|$, the number of the occurrences of the integer i in the stream. So the frequency moments on the stream can be calculated as $F_k = \sum_{i+1}^{M}(m_i)^k$. Then F_0 is the number of different elements in the sequence, F_1 is the length of the sequence n, and F_2 is the repeat rate of the sequence. So, F_2 can be related with the distance L_2. Let us suppose that we build a vector v of length M with entries chosen at random, we process the stream $s_1, s_2, \ldots, s_n$ entry by entry, and initialise a variable $Z = 0$. So, after whole stream has been processed we have $Z = \sum_{i=1}^{M} v_i m_i$. Then F_2 can be estimated as

$$\begin{aligned} Z^2 &= \textstyle\sum_{i=1}^{M} v_i^2 m_i^2 \\ &+ \textstyle\sum_{i=1}^{M} \sum_{j \neq i} v_i m_i v_j m_j \\ &= \textstyle\sum_{i=1}^{M} m_i^2 \\ &+ \textstyle\sum_{i=1}^{M} \sum_{j \neq i} m_i m_j v_i v_j \end{aligned}$$

So, if the entries of the vector v are pairwise independent, then the expectation of the cross-terms $v_i v_j$ is zero and $\sum_{i=1}^{M} m_i^2 = F_2$. If this calculation is repeated $O(1/\epsilon^2)$ times, with a different random v each time, and the average is taken, then the calculation can be guaranteed to be an $(1 \pm \epsilon)$ approximation with a constant probability, and if additionallly, by finding the median of $O(1/\delta)$ averages, this constant probability can be amplified to $1 - \delta$. It has been observed in (Feigenbaum et al. (1999)) that the calculation for F_2 can be adapted to find the L_2 distance between two interleaved, unaggregated streams a and b. Let us suppose that the stream arrives as triples $s_j = (a_i, i, +1)$ if the element is from a and $s_j = (b_i, i, -1)$ if the item is from stream b. The goal is to find the square of the L_2 distance between a and b, $\sum_i (a_i - b_i)^2$. We initialise $Z = 0$. When a triple $(a_i, i, +1)$ arrives we add $a_i v_i$ to Z and when a triple $(b_i, i, -1)$ arrives we subtract $a_i v_i$ from Z. After the whole stream has been processed $Z = \sum(a_i v_i - b_i vi) = \sum_i (a_i - b_i) vi$. Again the expectation of the cross-terms is zero and, then the expectation of Z^2 is L_2 difference of a and b. THe procedure for L_2 has the nice property of being able to cope with case of unaggregated streams containing multiple triples of the form $(a_i, i, +1)$ with the same i due to the linearity of the addition. This streaming algorithm translates to the sketch model: given a vector a the values of Z can be computed. The sketch of a is then these values of Z formed into a vector $z(a)$. Then $z(a)_i = \sum_j (a_j - b_j) v_{i,j}$. This sketch vector has $O(1/\epsilon^2 \log 1/\delta)$ entries, requiring $O(\log Mn)$ bits each one. Two such sketches can be combined, due to the composability property of the sketches, for obtaining the sketch of the difference of the vectors $z(a - b) = (z(a) - z(b))$. The space of the streamin algorithm, and then the size of the sketch is a vector of length $O(1/\epsilon^2 \log 1/\delta)$ with entries of size $O(\log Mn)$. A natural question can be if it is possible to translate sketching algorithms to streaming ones for distances different from L_2 or L_1 and objects other than vectors of integers. In (Graham (2003)) it shown that it is possible the this translation for the Hamming distance. This can be found in the next theorem of (Graham (2003)).

Theorem 12. *The sketch for the Symmetric Difference (Hamming distance) between sets can be computed in the unordered, aggregated streaming model. Pairs of sketches can be used to make* $1 \pm \epsilon$ *approximmations of the Hamming distance between their sequences, which succeed with probability* $1 - \delta$. *The sketch is a vector of dimension* $O(1/\epsilon^2 \log 1/\delta)$ *and each entry is an integer in the range* $[-n \ldots n]$.

Given that, under some circumstances, streaming algorithms can be translated to sketch algorithms, then the theorem 10 can be applied for the robot motion tracking problem, under the streaming model as well.

In general, the mobile robotics framework is more complex because we should process data flows provided by the captors under a dynamic situation, where the robot is moving, taking into account two kind of uncertainty:

- The sensors have low precision
- The robot movements are subject to deviations as any mechanical object.

The data flow provided by the captors produce similar problems to those that can be found on the databases. The robot should make the fusion of the information sources to determine his motion strategy. Some sources, called bags, allow the robot to self locate geometrically or in his state graph. While the robot executes his strategy, it is subject to movement uncertainties and then should find robust strategies for such uncertainty source. The goal is to achieve the robustness integrating the data flow of the captors to the strategies. We consider the classical form of simple Markovian strategies. In the simplest version, a Markov chain, MDP, is a graph where all the states are distinguishable and the edges are labeled by actions $L_1, L_2, \ldots, L_p$. If the states are known only by his coloration in k colors $C_1, C_2, \ldots, C_k$. Two states having the same coloration are undistinguishable and in this case we are talking about POMDP (Partially Observed Markov Decision Process). A simple strategy σ is a function that associates an action simplex to a color among the possible actions. It is a probabilistic algorithm that allows to move inside the state graph with some probabilities. With the help of the strategies we look for reaching a given node of the graph from the starting node (the initial state) or to satisfy temporal properties, expressed in LTL formalism. For instance, the property C_1 Until C_2 that express the fact that we can reach a node with label C_2 preceded only by the node C_1. Given a property θ and a strategy σ, let $Prob_\sigma(\theta)$ be the probability that θ is true over the probability space associated to σ. Given a POMDP M two strategies σ and π can be compared by means of ther probabilities, that is, $Prob_\sigma(\theta) > Prob_\pi(\theta)$. If $Prob_\sigma(\theta) > b$, it is frequent to test such a property while b is not very small with the aid of the path sampling according to the distribution of the POMDP. In the case that $b < Prob_\sigma(\theta) < b - \epsilon$ it can be searched a corrector for σ, it means, a procedure that lightly modify σ in such a way that $Prob_\sigma(\theta) > b$. It can be modified too the graph associated and in that case, we look for comparing two POMDPs. Let be M_1 and M_2 two POMDPs, we want to compare this POMDPs provided with strategies σ and π in the same way as are compared two automata in the sense that they are approximately equivalent (refer to the section concerning distance between DTDs). How can we decide if they are approximately equivalent for a property class? Such a procedure is the base of the strategy learning. It starts with a low performance strategy that is modified in each step for improvement. The tester, corrector and learning algorithms notions find a natural application in this context. One of the specificities of mobile robotics is to conceive robust strategies for the movements of a robot. As every mechanical object, the robot deviates of any previewed trajectory and then it must recalculate his location. At the execution of an action L_i commanded by the robot, the realization will follow L_i with probability p, an action L_{i-1} with probability $(1-p)/2$ and an action L_{i+1} with probability $(1-p)/2$. This new probabilistic space induce robustness qualities for each strategy, in other words, the $Prob_\sigma(\theta)$ depends on the structure of the POMDP and on the error model. Then the same questions posed before can be formulated: how to evaluate the quality of the strategies, how to test properties of strategies, how to fix the strategies such that we can learn robust strategies. We can consider that the robots are playing a game against nature that is similar to a Bayesian game. The criteria of robust strategy are similar to those of the direct approach. Another problem that arise in robot motion is the relocalization of a robot in a map. As we mentioned in the initial part of the section 6, one method that has been used frequently in robot

exploration for reducing the uncertainty in the position of robot was the use of landmarks and triangulation. The search of a landmark in an unknown environment can be similar to searching a pattern in a sequence of characters or a string. In the present work we applied sketching and streaming algorithms for obtaining distance approximations between objects as vectors in a dimensional reduced, and in some sense, deformated space. If we want to apply sketching or streaming for serching patterns as landmarks in a scene we have to deal with distance between permutations.

8. Conclusion and future work

The property testing algorithms under the sketch and streaming calculation model for measuring the level of approximation of inferred automata with respect to the true automata in the case of robot motion tracking problem as well as the map construction problem in robot navigation context. The use of sketch algorithms allow us to approximate the distance between objects by the manipulation of sketches that are significantly smaller than the original objects. Another problem that arise in robot motion is the relocalization of a robot in a map. As we mentioned in the section 2, one method that has been frequently used in robot exploration for reducing the uncertainty in the position of robot was the use of landmarks and triangulation. The search of a landmark in an unknown environment can be similar to searching a pattern in a large sequence of characters or a big string. For doing this task in an approximated and efficient way, sketch and streaming algorithms can be usefull.

9. References

A. Blum, P. R. & Schieber, B. (1991). Navigating in unfamiliar geometric terrain, *ACM STOC 91*, pp. 494–504.

Angluin, D. (1981). A note on the number of queries needed to identify regular languages.

Blum, M., Luby M. & Rubinfeld R. (1993). Self-testing/correcting with application to numerical problems.

Blum, M. & Kannan S. (1995). Designing programs that check their work.

Burago, D., de Rougemont, M. & Slissenko, A. (1993). Planning of uncertain motion, *Technical report*, Université de Poitiers.

Canny, J. & Reif, J. (1987). New lower-bound techniques for robot motion planning problems, *Proc. 28st. FOCS*, pp. 49–60.

Carmel, D. & Markovitch, S. (1996). Learning models of intelligent agents, *Technical Report CIS9606*, Department of Computer Science, Technion University.

Carmel, D. & Markovitch, S. (1998). How to explore your oponentŠs strategy (almost) optimally, *Proceedings ICMAS98 Paris France*.

C.Rodríguez Lucatero, A. Albornoz. & R.Lozano (2004). A game theory approach to the robot tracking problem.

Dana Angluin, Westbrook J. & Zhu, W. (1996). Robot navigation with range queries, *ACM STOC 96*, pp. 469–478.

Dean T., Angluin D., Basye K., Engelson S., Kaelbling L., Kokkevis E. & Maron O. (1985). Inferring finite automata with stochastic output functions and an application to map learning.

Dean, T. & Wellman, M. (1991). *Planning and Control*, Morgan Kaufmann Publishers.

Diaz-Frias, J. (1991). About planning with uncertainty, *8e. Congres Reconnaisance des Formes et Intelligence Artificielle*, pp. 455–464.

Magniez F. & de Rougemont, M. (2004). Property testing of regular tree languages, *ICALP 2004*.

Feigenbaum J., Kannan S., Strauss M.& Viswanathan M. (1999). An approximmate l_1 difference algorithm for massive data streams., *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pp. 501–511.

Fisher E., Magniez F. & de Rougemont, M. (2004). Property and equivalence testing on strings, *ECCC 2004*.

Gold, E. (1978). Complexity of automaton identification from given data.

Goldreich, O., S.Goldwasser & D.Ron.Property (1996). Testing and its connection to learning and approximation, *IEEE Symposium on Foundations of Computer Science*.

Graham, C. (2003). Sequence distance embeddings, *Phd thesis in cs university of warmick*, University of Warmick.

Kirman, J., Bayse, K. & Dean, T. (1991). Sensor abstractions for control of navigation, *Proc. IEEE Robotics & Automation*, pp. 2812–2817.

Latombe, J. (1990). *Robot motion planning*, Kluwer.

Latombe, J., Lazanas, A. & Shekhar, S. (1991). Robot motion planning with uncertainty in control and sensing, *Artificial Intelligence* 52: 1–47.

Valiant, L.G. (1984). A theory of the learnable, *CACM*, pp. 1134–1142.

Littman, M. (1994). Markov games as a framework for multiagent reinforcement learning, *Proceedings of the eleventh International Conference on Machine Learning*, pp. 157–163.

Lozano-Perez, T. (1986). A simple motion planning algorithm for general robot manipulators.

Lucatero, C. R. & Espinosa, R. L. (2005). Application of automata learning algorithms to robot motion tracking.

Lumelsky, V. & Stepanov, A. (1987). Path planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape.

Marion, J., Rodriguez, C. & de Rougemont, M. (1994). The evaluation of strategies in motion planning with uncertainty, *Proc. IEEE Robotics & Automation*.

M.Kearns & Valiant, L. (1989). Cryptographic limitation on learning boolean formulae and finite automata, *Proc. 21th ACM Symposium on Theory of Computing*, pp. 433–444.

Papadimitriou, C. (1985). Games against nature, *Journal of Computer and System Sciences* (31): 288–301.

Papadimitriou, C. & Steiglitz, K. (1982). *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall Inc.

Papadimitriou, C. & Tsitsiklis, J. (1987). The complexity of markov decision processes, *Mathematics of operations research* (3): 441–450.

Papadimitriou, C. & Yannakakis, M. (1991). Optimization, approximation and complexity classes, *Journal of Computer and System Sciences* (43): 425–440.

Pitt, L. & Warmuth, M. (1993). The minimal consistent dfa problem cannot be approximated within any polynomial.

R. Murrieta-Cid, H. G.-B. & Tovar, B. (2002). A reactive motion planner to maintain visibility of unpredictable targets, *Proceedings of the IEEE International Conference on Robotics and Automation*.

Rajesh Parekh, C. N. & Honavar, V. (1998). A polynomial time incremental algorithm for learning dfa, *Proceedings of the Fourth International Colloquium on Grammatical Inference (ICGI'98), Ames, IA. Lecture Notes in Computer Science vol. 1433*, pp. 37–49.

Rivest, R. L. & Schapire, R. E. (1993). Inference of finite automata using homing sequences.

Rodríguez-Lucatero, C. (1997). Evaluation, existence and optimization of reactive strategies in motion with uncertainty.

Rubinfeld, R. & Sudan, M. (1993). Robust characterizations of polynomials with applications to program testing.

Rubinstein, A. (1986). Finite automata play the repeated prisionerŠs dilemma.

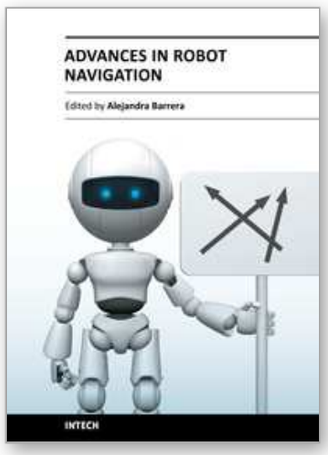
Schwartz, J. & Sharir, M. (1983). On the piano movers problem ii general techniques for computing topological properties of real algebraic manifolds, *Advances in Applied Mathematics Academic Press* .

Schwartz, J. & Sharir, M. (1991). Algorithmic motion planning in robotics., *Handbook of theoretical Computer Science* A: 391–425.

La Valle S.M., David Lin, L. J. G. J. L. & Motwani, R. (1997). Finding an unpredictable target in a workspace with obstacles, *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 731–736.

S.M. La Valle, H.H. González Baños, C. B. & Latombe, J. (1997). Motion strategies for maintaining visibility of a moving target, *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 731–736.

Valiant, L. (1979). The complexity of enumeration and reliability problems, *SIAM* 8(3).



Advances in Robot Navigation

Edited by Prof. Alejandra Barrera

ISBN 978-953-307-346-0
Hard cover, 238 pages
Publisher InTech
Published online 15, June, 2011
Published in print edition June, 2011

Robot navigation includes different interrelated activities such as perception - obtaining and interpreting sensory information; exploration - the strategy that guides the robot to select the next direction to go; mapping - the construction of a spatial representation by using the sensory information perceived; localization - the strategy to estimate the robot position within the spatial map; path planning - the strategy to find a path towards a goal location being optimal or not; and path execution, where motor actions are determined and adapted to environmental changes. This book integrates results from the research work of authors all over the world, addressing the abovementioned activities and analyzing the critical implications of dealing with dynamic environments. Different solutions providing adaptive navigation are taken from nature inspiration, and diverse applications are described in the context of an important field of study: social robotics.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Carlos Rodriguez Lucatero (2011). Application of Streaming Algorithms and DFA Learning for Approximating Solutions to Problems in Robot Navigation, Advances in Robot Navigation, Prof. Alejandra Barrera (Ed.), ISBN: 978-953-307-346-0, InTech, Available from: http://www.intechopen.com/books/advances-in-robot-navigation/application-of-streaming-algorithms-and-dfa-learning-for-approximating-solutions-to-problems-in-robo

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen