Proceedings of the 1995 International Conference on Robotics and Automation.

Adaptive Path Planning: Algorithm and Analysis*

Pang C. Chen
Sandia National Laboratories
Albuquerque, NM 87185-0951, USA

Abstract

To address the need for a fast path planner, we present a learning algorithm that improves path planning by using past experience to enhance future performance. The algorithm relies on an existing path planner to provide solutions to difficult tasks. From these solutions, an evolving sparse network of useful robot configurations is learned to support faster planning. More generally, the algorithm provides a framework in which a slow but effective planner may be improved both cost-wise and capability-wise by a faster but less effective planner coupled with experience. We analyze the algorithm by formalizing the concept of improvability and deriving conditions under which a planner can be improved within the framework. The analysis is based on two stochastic models, one pessimistic (on task complexity), the other randomized (on experience utility). Using these models, we derive quantitative bounds to predict the learning behavior. We use these estimation tools to characterize the situations in which the algorithm is useful and to provide bounds on the training time. In particular, we show how to predict the maximum achievable speedup. Additionally, our analysis techniques are elementary and should be useful for studying other types of probabilistic learning as well.

1 Introduction

Path planning in known environments refers to finding a short, collision-free path from an initial robot configuration to a desired configuration. It has to be fast to support real-time task-level robot programming. Accordingly, it has received much attention [15, 12], and there are now a number of implemented path planners based on a variety of approaches. Unfortunately, current planning techniques are still too slow to be effective, as they often require several minutes, if not hours of computation.

To remedy this situation, we have developed a simple learning algorithm [5] that uses past experience to increase future performance. Thus, if there are more than one problem to be solved, the cost of each problem can be amortized and decreased through learning. The algorithm relies on an existing path planner to provide solutions to difficult tasks. From these solutions, it learns a sparse network of useful robot configurations that guides and supports fast planning. More generally, the algorithm is actually a framework in which a slow but effective planner may be improved both cost-wise and capability-wise by a faster but less effective planner coupled with experience. In this paper, we provide a deeper analysis by formalizing the concept of improvability, and deriving sharp conditions under which planners can be improved within the framework.

To achieve predictive power while preserving some generality, we study the algorithm under models with different simplifying assumptions and applications. The particular (as opposed to general) analysis is based on two stochastic models, one pessimistic (on task complexity), the other randomized (on experience utility). Using these models, we derive global quantitative bounds on planning cost and capability in terms of training time. We show that the reliance of the improved planner on the original slow planner is at most inversely proportional to the training time. We also characterize the situations in which learning is useful and prescribe the amount of training required. Finally, we use these analytic performance estimation tools to gain insight into some experimental results. Although our presentation is in the context of motion planning, the algorithm and the analysis are extensible to more general learning. In particular, they may be applied to higher-level task planning or other domains in which experience is useful. Our theoretical work should complement well with the experimental work of others.

2 Related Work

Our research builds on the results of [5], which presents the algorithm for stationary environments along with some general but weak analysis on the learning process. To cope with incrementally changing environments, the algorithm can be extended with an on-demand experience repair strategy and an object-attached experience abstraction scheme [1, 7]. In this paper, however, we deal only with the fundamental stationary case as in [5]. Our paper is self-contained except for two mathematical results from [5] which we use in the analysis section without repeating the proofs.

As mentioned in the introduction, a large amount of research has been done on robot path planning, most of which deals with solving one-time problems in stationary environments [2, 3, 6, 10, 14, 16, 21]. Most implemented path planners have been developed for mobile robots and manipulators with few degrees of freedom (dof). There are some that are designed for many dof manipulators based on random [2] (Brownian motion), sequential [10] (backtracking with virtual obstacles), or parallel [3] (genetic opti-

*This work has been performed at Sandia National Laboratories and supported by the U.S. Department of Energy under Contract DE-AC04-94AL85000.

DISCLAIMER

mization) search. All of these planners, however, typically require minutes of computation for mobile robots, and tens of minutes for 6 degrees of freedom manipulators.

For solving several problems in stationary environments, there are a few other path planners that incorporate learning: some [9, 20] take a higher-level, reasoning approach, while others [13, 19] take a lower-level, memory-based approach similar to ours. Learning can be done incrementally, or in phases which some consider as preprocessing [13]. To decrease the effective cost of solving each problem, all of these work maintain a network (roadmap) of useful robot configurations (landmarks) and employ some sort of a local planner for moving through the network. Algorithmically, there are some differences between ours and that of [13, 19]. First, we assume and use the same distribution of tasks (problems) for both training and subsequent problem-solving. In [13, 19], a uniform problem distribution is used for training. Second, we assume the existence of a fairly reliable, albeit slow, global planner, whereas they do not. Thus, while their algorithms may be more general, they may also require more training time to compensate for the lack of solutions when local planning fails. Overall, the most significant difference between all of the aforementioned work and ours is that we aim to provide a theoretical foundation for algorithm analysis to: 1) better understand and predict our experimental results; and 2) suggest similar analysis techniques so that others may apply to better understand their algorithms.

3 Algorithmic Framework

Given an arbitrary work environment and an arbitrary task (u, w) of moving the robot from configuration point u to w, we assume that there are initially two path planners available: fast and slow. Both return true (1) if successful, false (0) otherwise. The fast planner is required to be fast, symmetric, and only locally effective, i.e., it should have a good chance of success if u and w are close to each other. The slow planner, on the other hand, is required to be much more globally effective than fast, and hence can be very slow. It is the performance of this planner that we wish to improve. Note that this 'planner' can even be a human robot operator.

In our learning scheme, we retain the global effectiveness of slow by calling it whenever necessary, while reducing the overall time cost by calling fast whenever possible. To utilize fast fruitfully, we remember significant intermediate robot configurations learned from the solution paths of slow. These subgoals represent fully specified robot configurations and are stored in memory V, with connecting edges E (indicating successes of fast) maintained so that complete solution paths may be regenerated through applications of fast. The subgoals V can be thought of as 'trail-markers' in that each marker can be traced to one another through the trails E. We call the connected network of trail-markers the experience graph $G = (V, E)$.

Formally, the learning algorithm Adapt is shown in Figure 1. The algorithm is based on two planners: Fast and Slow, which are in turn based on fast and slow, respectively. Using V, planners Fast and Slow attempt to reach a desired configuration w from the current configuration. Since V forms a connected component, the planners only need to check the reachability of w from a known reachable trail-marker in V using a heuristic subgoal ordering function h. Planner Fast simply goes through the markers according to h, and finds the first marker v for which fast(v, w) succeeds. Planner Slow simply selects the best marker v according to h, and calls slow(v, w). Adapt is essentially Fast backed up by Slow. Learning occurs when Fast fails but Slow succeeds. In this case, we assume that the solution path of slow(v, w), denoted by slow$[v, w]$, is somehow 'abstracted' into a short chain ρ of trail-markers with each edge traversable by fast. (The abstraction can be implemented by locating the markers with binary search on a discretized solution path.) We achieve incremental learning by absorbing ρ into memory using a procedure, Learn, which is required to augment V with enough of ρ to ensure a solution path for Fast(w) if it were to be called again.

Algorithm Adapt(Fast, Slow)
  $V \leftarrow$ {current configuration};
  do forever
    $w \leftarrow$ goal configuration;
    if (**not** Fast$(w; V, h)$ **and** Slow$(w; V, h)$) **then**
      $\rho \leftarrow$ Abstract(Slow$[w; V, h]$);
      $V \leftarrow$ Learn(V, ρ);

Figure 1: An algorithm for improving path planning.

4 Performance Analysis

In this section, we present an approach to analyzing speedup learning [23], which is what the algorithm is doing – seeking to improve program efficiency through learning. We first formalize the concept of improvability, and derive general conditions for such improvements. Next, we introduce two models with additional simplifying assumptions and parameters. Using these models, we then derive sharp bounds on planning cost and capability in terms of training time. Finally, we characterize the improvable situations in terms of the model parameters, and prescribe the amount of training required.

Two performance measures are of interest: efficiency and capability. To quantify, we assume that the problems are drawn randomly and independently from a distribution (as in PAC-learning [18]) on some configuration space (C-space) S. We do not require slow to be complete; we do require that it have a success probability σ in solving a random task. We assume that only slow, fast, and Learn have costs, each being a constant. (The cost of Abstract can be absorbed into the cost of Learn.) To normalize, let 1, r, and c be the respective costs of slow, fast, and Learn. (Both r and c are typically $\ll 1$.) We use subscript n on a program variable to denote its value at the n^{th} loop. Thus, V_n denotes the memory V after Adapt has been trained

A_n	The probability that Adapt will need to call slow in solving problem $n+1$, i.e., the failure probability that problem $n+1$ will not be Fast-solvable with V_n.
K_n	The number of times that slow has been called by Adapt after n steps of training.
E_n	The cost of Fast in solving problem $(n+1)$.
F_n	The cumulative cost of Adapt after n steps of training.

Table 1: Variables of interest.

with n problems. We are interested in both the speedup that Adapt has over the plain iterations of slow, and the capability of Adapt as it increases with training. We are also interested in the performance of Fast, which is Adapt without the backup of Slow after some training. Thus, we use the following definition of improvability.

Definition 1 *Let $\mathcal{A}$ be a speedup learning algorithm designed to improve the efficiency of another algorithm $\mathcal{A}'$. We say that*

1. *$\mathcal{A}$ can (cost-wise)* improve *$\mathcal{A}'$ with average failure probability p iff $\mathcal{A}$ can perform the same task as $\mathcal{A}'$ with average probability at least $1-p$, while costing less on average.*
2. *$\mathcal{A}$ can* effectively improve *$\mathcal{A}'$ iff $\mathcal{A}$ can improve $\mathcal{A}'$ with failure probability no greater than that of $\mathcal{A}'$.*
3. *$\mathcal{A}$ can* effectively replace *$\mathcal{A}'$ iff $\mathcal{A}$ can effectively improve $\mathcal{A}'$ without relying on $\mathcal{A}'$.*

The random variables in Table 1 are important in characterizing the performance of Adapt. Their basic relationship is given by the following lemma:

Lemma 1 *The average planning cost of* Adapt *per problem after n steps of training is*

$$\mathbf{E}\Delta F_n \stackrel{\text{def}}{=} \mathbf{E}(F_{n+1} - F_n) = (1+\sigma c)\mathbf{E}A_n + \mathbf{E}E_n. \quad (1)$$

Consequently, the average cumulative cost of Adapt *after n steps of training is*

$$\mathbf{E}F_n = (1+\sigma c)\mathbf{E}K_n + \sum_{0 \le j < n} \mathbf{E}E_j. \quad (2)$$

Proof The cost for ΔF_n is obvious since in addition to E_n, a cost of $(1+\sigma c)A_n$ is required to call slow with probability A_n and Learn with probability σA_n. The second equation follows immediately from the fact that $\mathbf{E}K_n = \sum_{0 \le j < n} \mathbf{E}A_j$ [5]. ∎

Using these variables, we can immediately characterize the conditions under which improvements can be achieved. (Proof is clear.)

Lemma 2 *After n steps of training,*

1. Fast *can improve* slow *with failure probability* $\mathbf{E}A_n$ *iff* $\mathbf{E}E_n < 1$.
2. Adapt *can effectively improve* slow *iff* $\mathbf{E}\Delta F_n < 1$.

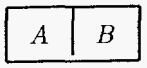

Figure 2: An environment with two traps.

3. Fast *can effectively replace* slow *iff* $\mathbf{E}E_n < 1$ *and* $\mathbf{E}A_n \le 1-\sigma$.

To express these conditions in more useful terms of training time, we need to have further information such as the specification of the vertex ordering function h and the incremental learning strategy Learn. In the following subsections, we introduce two models, one pessimistic, the other randomized, each with different applicability and additional simplifying assumptions. Using these models, we derive sharp bounds on the variables of Table 1, and explore the ramifications of Lemma 2.

4.1 Pessimistic Model

In the pessimistic model, we study the worst-case consequence of learning in environments in which the strategy of Learn is specified, and the connectivity of S under fast is characterized. To motivate, consider a point robot in a planar polygonal environment shown in Figure 2, and let fast be 'go-straight'. Since we are dealing with a point robot, the C-space and the work space are the same. Clearly, the C-space is well-connected locally in the sense that each feasible (configuration) point is connectable (visible) to at least half of the entire C-space under fast. However, this environment may be difficult for the algorithms of [13, 19] to handle in that the points randomly sampled will tend to form two disconnected components (traps) in A and B, and will not help in solving problems that require reaching B from A. In contrast, with the help of slow, our algorithm will adapt to this environment efficiently.

Definition 2 *Under pessimistic model $\mathcal{M}_p$,*

1. Learn *adopts the minimal memory strategy of adding all necessary edges to take in only the minimal subchain sufficient to* Fast*-solve the current problem.*
2. *C-space S is m-coverable for some m in that S can be covered by m components (not necessarily disjoint), with the initial configuration in S_1 and each component S_i $(1 \le i \le m)$ being connected under* fast*, i.e., every pair of points u, v in S_i satisfies* fast(u, v).

Thus, the environment in Figure 2 is 2-coverable under fast, with the two components being A and B. More generally, if the complexity of the C-space S relative to fast is measured by m in that S is m-coverable, then the following theorem basically says that the failure probability of Adapt is at most proportional to m and inversely proportional to the amount of training n, and that this bound is tight up to some constant factor.

Theorem 3 *Under $\mathcal{M}_p$, the expected* Fast*-failure probability of* Adapt *after n steps of training has a upper bound of*

$$\mathbf{E}A_n \le \frac{m-1}{\sigma n}, \quad (3)$$

and a lower bound of

$$\mathbf{E}A_n \geq \frac{m-1}{e\sigma(n+1)} \tag{4}$$

for $n \geq (m-1)/\sigma$ and some environment dependent on n.

Proof (3) Let w_j be the j^{th} random problem. Let $X_{i,j}$ be the 0-1 random variable indicating that $w_{j+1} \in S_i$ and w_{j+1} is not Fast-solvable using V_j. Since the S_i's cover S, we have $\mathbf{E}A_n \leq \mathbf{E}\sum_{i>1} X_{i,n}$. Also, since Adapt never forgets, we have $V_j \subseteq V_{j+1}$ for all j, which implies that $\mathbf{E}X_{i,j} \geq \mathbf{E}X_{i,j+1}$ because $\mathsf{Fast}(w; V_j, h) \implies \mathsf{Fast}(w; V_{j+1}, h)$ for all w. Finally, if $X_{i,j} = 1$ then Slow will be called. If it is successful, w_{j+1} will be remembered in V_{j+1}, causing $X_{i,j'} = 0$ for $j' > j$. Consequently, for any i, $\mathbf{E}\sum_j X_{i,j} \leq 1/\sigma$, which is the expected number of times that Slow will be called to reach some points in S_i before one is remembered. Combining all three inequalities, we have

$$\begin{aligned}\mathbf{E}A_n \leq \sum_{i>1}\mathbf{E}X_{i,n} &\leq \sum_{i>1}\sum_{0<j\leq n}\frac{\mathbf{E}X_{i,j}}{n} \\ &\leq \sum_{i>1}\frac{1}{\sigma n} = \frac{m-1}{\sigma n}.\end{aligned}$$

(4) Let S be composed of exactly m non-overlapping components, with every component disconnected from each other except S_1 under fast. Let the distribution be uniform within each component and have total probability $p = 1/(\sigma(n+1))$ on S_i for $i > 1$, and probability $1-(m-1)p$ on S_1. Then for $i > 1$, the probability that Adapt will be given j training problems from S_i after n steps and failed to learn from them is $\binom{n}{j}p^j(1-p)^{n-j}(1-\sigma)^j$. Thus, with this probability summed over j, Adapt will Fast-fail on S_i. Summing up each $i > 1$, we have

$$\begin{aligned}\mathbf{E}A_n &\geq (m-1)p\sum_j\binom{n}{j}(p(1-\sigma))^j(1-p)^{n-j} \\ &= (m-1)p(1-p\sigma)^n,\end{aligned}$$

yielding the desired lower bound. ∎

Using m as a complexity measure of the C-space, the following theorem says that the Fast-planning cost of Adapt is at most linear in r and m, and that this bound is tight up to a constant factor. Further, the number of times slow will be needed is at most proportional to m and inversely proportional to its capability σ, and that this bound is tight with sufficient amount of training.

Theorem 4 *Under $\mathcal{M}_p$, the expected* Fast*-planning cost of* Adapt *after n steps of training has an upper bound of*

$$\mathbf{E}E_n \leq r\left(3m-2-(m-1)\left(1-\tfrac{\sigma}{m-1}\right)^n\right). \tag{5}$$

The expected number of calls to Slow *has an upper bound of*

$$\mathbf{E}K_n \leq \left(\tfrac{m-1}{\sigma}\right)\left(1-\left(1-\tfrac{\sigma}{m-1}\right)^n\right). \tag{6}$$

Conversely, there exists an environment in which $\mathbf{E}E_n = r(2m-1)$, and an environment in which the equality of (6) is reached.

Proof Let J_n be the number of times slow is successful. Classify the markers of ρ into two types: type 1 sharing the same S_i for some i with the current V, and type 2 does not. According to the 'memory minimizing' strategy of Learn, at most one marker of the subchain can be of type 1, and at most two markers per component not 'occupied' by the current V can be of type 2. (Otherwise, an edge can be introduced to shorten the subchain.) Thus, counting all subchains, the total number of type 1 and 2 markers. are at most J_n and $2(m-1)$, respectively. Hence, $\|V_n\| \leq 1 + 2(m-1) + J_n$.

For (5), it suffices to show that $\mathbf{E}J_n \leq (m-1)\left(1-\left(1-\frac{\sigma}{m-1}\right)^n\right)$. Partition S into m disjoint components with the i^{th} component being $S_i' = S_i \setminus \bigcup_{j<i} S_j$. Let X_i be the 0-1 random variable indicating that one of the n training problems is both in S_i' and solvable by Slow. Then $J_n \leq \sum_{i>1} X_i$ because there can be at most one successful call of Slow for each $i > 1$. Let p_i be the probability that a random problem is both in S_i' and solvable by Slow. Then $\mathbf{E}X_i = 1-(1-p_i)^n$ and $\sum_{i>1} p_i = \sigma - p_1$. Using the fact that $(p_2, \ldots, p_m)$ majorizes [17] $(\frac{\sigma-p_1}{m-1}, \ldots, \frac{\sigma-p_1}{m-1})$, we have $\sum_{i>1}(1-p_i)^n \geq (m-1)(1-\frac{\sigma-p_1}{m-1})^n \geq (m-1)(1-\frac{\sigma}{m-1})^n$, as desired. For (6), simply notice that $\sigma\mathbf{E}K_n = \mathbf{E}J_n$.

For the lower bound on $\mathbf{E}E_n$, let S be composed of exactly m non-overlapping components, with the first $m-1$ components consisting of exactly 2 points, $s_{i,1}$ and $s_{i,2}$. Let the only inter-component connections under fast be between $s_{i,2}$ and $s_{i+1,1}$. Let $s_{1,1}$ be the initial configuration, and let the distribution be 0 on the first $m-1$ components. Let h select the markers in the increasing order of the component index. Then upon solving the first problem, a path of $2m-1$ markers connecting $s_{1,1}$ to a point in S_m will be incorporated into V. Consequently, a Fast-planning cost of $r(2m-1)$ is required for latter problems.

For the lower bound on $\mathbf{E}K_n$, let S be composed of exactly m non-overlapping components, with every component disconnected from each other except S_1 under fast. For each $i > 1$, let the distribution have equal total probability $1/(m-1)$ on S_i. Then the learning process becomes effectively a coupon collector's problem [8] with $m-1$ types of coupons. Thus, $\mathbf{E}J_n = \sum_{i>0}\mathbf{E}X_i$, where X_i is the 0-1 random variable indicating that one of the n training problems is both in S_i and solved by Slow. The bound now follows since $\mathbf{E}X_i = 1-(1-\frac{\sigma}{m-1})^n$. ∎

Finally, the following theorem discerns the situations in which Adapt is useful by weighing $1/r$, the speed of fast, against m, the complexity of S. For those situations in which Adapt can be useful, the theorem also prescribes the amount of training required.

Theorem 5 *Under $\mathcal{M}_p$, if $1/r > (3m-2)$, then* Adapt *can effectively improve* slow *with*

$$n \geq \frac{(m-1)(1+\sigma c)}{\sigma(1-r(3m-2))} \tag{7}$$

steps of training. If slow *is also not complete, then* Fast *can effectively replace* slow *with*

$$n \geq \frac{m-1}{\sigma(1-\sigma)} \tag{8}$$

steps of training. If $2m-1 < 1/r \leq 3m-2$, *then after*

$$n < \ln_{\left(\frac{m-1}{m-2}\right)} \left(\frac{m-1}{3m-2-1/r}\right) \tag{9}$$

steps of training, Fast *can still improve* slow *with average failure probability no greater than* $(m-1)/(\sigma n)$. *If* $1/r \leq 2m-1$, *then there exist environments in which neither* Adapt *nor* Fast *can improve* slow.

Proof From Theorem 4, we have $\mathbf{E}E_n < r(3m-2)$ for any finite n. If (7) holds, then from Theorem 3 and Lemma 1, we have $\mathbf{E}\Delta F_n \leq 1 - r(3m-2) + \mathbf{E}E_n < 1$, as prescribed by 2 of Lemma 2. Further, if $\sigma < 1$ and (8) holds, then $\mathbf{E}A_n \leq 1-\sigma$ and $\mathbf{E}E_n < 1$, as prescribed by 3 of Lemma 2. If $2m-1 < 1/r \leq 3m-2$ and (9) holds, then from Theorem 4, $\mathbf{E}E_n < r(2m-1+(m-1)(1-(3m-2-1/r)/(m-1))) = 1$, as prescribed by 1 of Lemma 2. If $2m-1 \geq 1/r$, then by Lemma 1 and Theorem 4, there exists an environment in which $\mathbf{E}F_n \geq \mathbf{E}E_n = r(2m-1) \geq 1$, contrary to what is required for improvement in Lemma 2. ∎

4.2 Randomized Model

In the randomized model, we study the average-case consequence of learning in environments in which the number of new trail-markers acquired by Learn and the power of fast are randomized. Thus, we are interested in the average behavior for a class of environments instead of a fixed environment.

Definition 3 *Under randomized model* $\mathcal{M}_r$,

1. *The number of new trail-markers acquired by* Learn, λ, *is an independent random variable.*
2. fast(u,w) *is 1 for any established edge* (u,w) *in* V, *and is 1 otherwise with independent probability* $\bar{\mu}$.

While $\mathcal{M}_r$ may not be physically realizable as opposed to $\mathcal{M}_p$, it does simplify the corresponding results for $\mathcal{M}_p$, and provide reasonable estimation tools as demonstrated in next section.

Theorem 6 *Under* $\mathcal{M}_r$,

$$\mathbf{E}A_n \leq \frac{1}{\sigma(n+1)\mathbf{E}\mu(\lambda)}, \tag{10}$$

where $\mathbf{E}\mu(\lambda) \stackrel{\text{def}}{=} 1 - \mathbf{E}(1-\bar{\mu})^{\lambda}$ *denotes the average utility of a learned chain* ρ.

Proof (sketch) Let $L_n = A_n - A_{n+1}$ be the additional probability of problems learned through the incorporation of ρ_n into V_n. We call $\mathbf{E}(L_n \mid A_n)$ the expected learning rate, which evaluates to $A_n \sigma \mathbf{E}\mu(\lambda)$.

It now suffices to show that for $n > 0$, $\mathbf{E}A_n \leq 1/(\alpha(n+1))$ for an expected learning of $\mathbf{E}(L_n \mid A_n) = \alpha A_n$, $\alpha < 1$. For $n = 0$, $\mathbf{E}A_0 \leq 1 \leq 1/\alpha$. For $n = 1$, $\mathbf{E}A_1 = \mathbf{E}A_0(1-\alpha A_0)$ has maximum value $1/(4\alpha)$, which is less than the desired upper bound of $1/(2\alpha)$. For $n \geq 2$, it is known [5, Theorem 8] that $\mathbf{E}A_n \leq (\alpha(n+1))^{-1} \exp\left((0.52 - \ln n)/(2n)\right)$, which implies the desired upper bound. ∎

Notice how the theorem above, which bounds the failure probability under $\mathcal{M}_r$, has the reciprocal of the average trail-marker utility essentially replacing the C-space complexity parameter for the corresponding Theorem 3 under $\mathcal{M}_p$. Notice also how the following theorem corresponding to Theorem 4 simplifies the planning cost of Fast in terms of its failure probability.

Theorem 7 *Under* $\mathcal{M}_r$, *the expected cost of* Fast *after* n *steps of training is*

$$\mathbf{E}E_n = \frac{r}{\bar{\mu}}(1 - \mathbf{E}A_n). \tag{11}$$

Consequently, the expected cost of Adapt *per problem after* n *steps of training is*

$$\mathbf{E}\Delta F_n = r/\bar{\mu} + (1 + \sigma c - r/\bar{\mu})\mathbf{E}A_n. \tag{12}$$

Proof Let N be the number of markers in V_n, and Q_i be the probability that problem $n+1$ cannot be reduced by fast to any of the first i markers. Then $\mathbf{E}E_n = r\mathbf{E}\sum_{0 \leq i < N} \mathbf{E}(Q_i \mid N) = r\sum_{0 \leq i < N}(1-\bar{\mu})^i = \frac{r}{\bar{\mu}}(1 - \mathbf{E}(Q_N \mid N)) = \frac{r}{\bar{\mu}}(1 - \mathbf{E}A_n)$. The formula for $\mathbf{E}\Delta F_n$ follows from Lemma 1. ∎

The following theorem, corresponding to Theorem 5, discerns the situations in which Adapt is useful and prescribes the necessary training period. Again, notice how under $\mathcal{M}_r$, the power of fast, $1/\bar{\mu}$, is playing the role of the C-space complexity parameter m under $\mathcal{M}_p$.

Theorem 8 *Under* $\mathcal{M}_r$, Adapt *can effectively improve* slow *with sufficient training iff* $r < \bar{\mu}$. *Sufficient training can be achieved with*

$$n \geq \frac{1}{\sigma \mathbf{E}\mu(\lambda)} \left(1 + \frac{\sigma c}{1 - r/\bar{\mu}}\right). \tag{13}$$

number of examples. If slow *is also not complete, then* Fast *can effectively replace* slow *with*

$$n \geq \frac{1}{\sigma(1-\sigma)\mathbf{E}\mu(\lambda)} \tag{14}$$

steps of training. If $r \geq \bar{\mu}$, *then* Fast *may still improve* slow, *but only with minimum failure probability* $\mathbf{E}A_n \geq 1 - \bar{\mu}/r$.

Proof From Lemma 1 and Theorem 7, we have $\mathbf{E}\Delta F_n = r/\bar{\mu} + (1 + \sigma c - r/\bar{\mu})\mathbf{E}A_n$. Combining Lemma 2, Theorem 6, and this formula yields the desired theorem. ∎

Finally, we have the following global performance bound of Adapt during training.

Theorem 9 *Under $\mathcal{M}_r$, the ratio of the average cost of* Adapt *to that of* slow *is bounded asymptotically by $r/\bar{\mu}$ as the number of training problems approaches infinity. More globally, the behavior is*

$$\frac{F_n}{n} \leq \frac{r}{\bar{\mu}} + \left(1 + \sigma c - \frac{r}{\bar{\mu}}\right) \begin{cases} \frac{\ln(eA_0\alpha n)}{(\alpha n)} & \text{if } A_0\alpha n > 1; \\ A_0 & \text{otherwise,} \end{cases} \tag{15}$$

where $\alpha = \sigma \mathbf{E}\mu(\lambda)$. Accordingly, the maximum value that the ratio can attain at any n is at most

$$F_n/n \leq r/\bar{\mu} + \left(1 + \sigma c - r/\bar{\mu}\right) A_0. \tag{16}$$

Proof From Lemma 1 and Theorem 7, it suffices to prove that $\mathbf{E}K_n \leq \ln(eA_0\alpha n)/\alpha$ if $A_0\alpha n > 1$; and $\mathbf{E}K_n \leq A_0 n$ otherwise. Since $\mathbf{E}K_n = \sum_{j<n} \mathbf{E}A_j$, and $\mathbf{E}A_n \leq \min(A_0, (\alpha(n+1))^{-1})$, we must have $\mathbf{E}K_n \leq A_0 x + (H_n - H_x)/\alpha$, for all positive integers $x \leq n$. Since $H_n - H_x \leq \ln(n/x)$, we may extend the domain of x to the reals and obtain $\mathbf{E}K_n \leq A_0 x + \ln(n/x)/\alpha$, which yields the theorem when minimized at $x = \min(n, 1/(\alpha A_0))$. ∎

5 Application and Verification

We now demonstrate the applicability and fidelity of the theory thus developed.

5.1 Pessimistic Model

Going back to the example of a point robot in a 2-coverable workspace of Figure 2, we see that from Theorem 3, the expected failure probability of Adapt can be no greater than $1/(\sigma n)$ with n being the number of training problems. Notice that this result does not depend on what the problem distribution is, as long as it is fixed for both training and subsequent problem-solving. More generally, we have the following theorem for a point robot in simple planar environments.

Theorem 10 *Consider a point robot in a planar simple polygonal workspace filled with b simple polygonal obstacles and having a total of c corners. Let* fast *be "go-straight". Then the workspace is $(2b + c - 2)$-coverable. Consequently, after n steps of training, the expected probability that* Adapt *will succeed in reaching the next random goal using only "go-straight" via the learned markers V_n is at least $1 - (2b + c - 3)/(\sigma n)$. Further, the expected cost of* Fast *is at most $3r(2b + c - 2)$.*

Proof It suffices to show that the feasible workspace (a simple polygon with holes) can be triangulated into $m = 2b + c - 2$ triangles, since each triangle is convex, hence connected under "go-straight". It is known that every simple polygon with holes has a valid constrained triangulation [22] whose edges are a superset of the input edges, and whose vertices are the input vertices (no added vertices are necessary). Let m and d be the number of triangles and edges in such a triangulation. Then by Euler's formula [22], we have $(m + b + 1) - d + c = 2$. Also, counting each edge of every triangle yields $3m = 2d - c$. Hence, $m + b + c - (3m + c)/2 = 1$ implies the desired result of $m = 2b + c - 2$. ∎

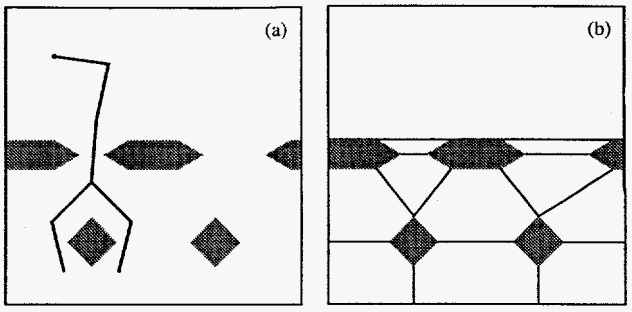

Figure 3: A 10-dof robot environment.

Beyond immediate applications to point robots, the theories thus developed can also help us make plausible performance predictions for more complicated robots. Shown in Figure 3a is a 10-dof robot in a planar environment studied in [13]. Let fast implement the following procedure:

1. move one end of the robot straight to the desired location with the rest of the robot complying;
2. with the first end point fixed, move the other end of the robot straight to the its desired location with the rest of the robot complying;
3. with both end points fixed, move the rest of the robot to their desired configuration using standard potential field approach.

Since the robot is snake-like with high dof, it is quite conceivable that fast will succeed if both end points are visible from their desired locations, and that there is indeed a solution. Under this plausible assumption, we can bound the number of fast-connected components necessary in covering the 10-dimensional C-space. From Figure 3b, we see that the workspace is 11-coverable for each end point under visibility. Also, from visual inspection, we see that there are at most 12 topologically distinct inverse-kinematic solutions for a given pair of end points. Hence, the C-space is at most $11 \cdot 11 \cdot 12 = 1452$-coverable. Consequently, if the teacher slow were a complete planner, it will take at most 145100 training problems for Adapt to attain a 99% expected capability.

5.2 Randomized Model

To demonstrate the randomized model, we explain results and make performance predictions on two separate experiments previously reported in [5]. Figure 4a shows a planar 2-link robot environment in which Adapt is applied. In this experiment, slow implements an incomplete but fairly effective planner [4], and fast implements a simple potential-field based hill-climb. There are 5 polygonal obstacles in the fixed workcell, and a goal set consisting of 9 preselected goal positions. Starting at home position 0, the robot is to go through a sequence of 100 goals randomly selected from the goal set. In Figure 4b, the ratio of the cumulative planning cost of Adapt to that of slow only is plotted against problem

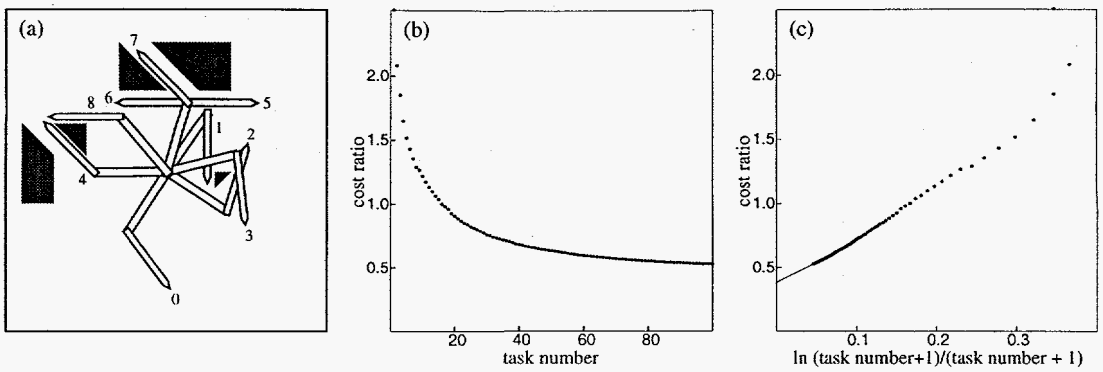

Figure 4: Time improvement on a planar 2-link robot environment.

number n. The planning costs are averaged over 100 runs and are measured by the number of robot-to-obstacle distance evaluations, which is the dominating factor in the computing cost of each planner. Figure 4c plots the ratio against $(\ln(n+1))/(n+1)$ to show their asymptotic linear relationship, hinted at by Theorem 9.

The experiment shows that Adapt is able to increase its performance relative to slow from 150% slower (ratio $\doteq 2.5$) to 50% faster at the end of 100 training examples. We can use Theorem 9 to predict the maximum speedup achievable. If we believe that (15) is also an asymptotic lower bound, then the plot implies that $r/\bar{\mu} \doteq 0.38$ is the minimum achievable cost ratio, equivalent to a maximum speedup of 62%. From other empirical observations, we estimate that $r \doteq 0.1$, $c \doteq 1$, $\sigma \doteq 1$, and $A_0 \doteq 0.9$. Hence, $\bar{\mu} \doteq 0.26$. Since $\lambda \doteq 2$, we also estimate $\mathbf{E}\mu(\lambda) \doteq 0.45$. To see how consistent these numbers are, we estimate the number of training problems required by Adapt to have its cumulative cost first become less than that of slow. Using (15), we have $\alpha \doteq 0.45$ and $\frac{\ln(eA_0\alpha n)}{eA_0\alpha n} \doteq \frac{1+\sigma c - r/\bar{\mu}}{eA_0(1-r/\bar{\mu})} \doteq 0.156$. giving us $eA_0\alpha n \doteq 19$, or $n \doteq 17.2$, which is very close to the observed $n = 17$ in the plot.

We use our theory to explain and predict another experiment performed previously in which Adapt is applied on a 3-dimensional 6-dof gantry robot environment [5]. The same slow and fast used for the planar case are also used here. In this environment (left side of Figure 5), there are 4 obstacles: a $(16+2)$-sided polyhedral approximation of a cylindrical cask, two cask stands, and a floor. Motivated by problems in radiation survey [11], the goal positions are chosen randomly, and correspond to the robot end effector touching the cask surface in a prescribed orientation. The tasks are sufficiently difficult that the original planner, slow, fails to reach 7 out of a sequence of 100 random goals. In contrast, Adapt is able to accomplish all but 1 task during the exercise, thereby increasing the capability of the original planner. Moreover, Adapt calls slow only 5 times, and stores only 11 trail-markers in addition to the initial robot position. Figure 5 plots the task number against the the ratio of the cumulative effort expended by Adapt to that expended by slow only. The 5 large points indicate Adapt's calling of slow, and the single white point indicates the only failure of Adapt.

Using the data, we estimate $\sigma \doteq 93\%$ because of the 7 failures; $\mathbf{E}\mu(\lambda) \doteq 1 - 0.01^{1/5} \doteq 80\%$ because only 5 chains are involved. Using (14) of Theorem 8, we then estimate $n \doteq 1/(0.93 \cdot 0.07 \cdot 0.8) \doteq 19.2$ to be the number of training tasks n required for Fast to improve both the speed and the capability of slow. This estimate means that fast is already very powerful, and that roughly only 2 calls (#17 and #18 in the plot) to slow are necessary for Fast to catch up with slow in task solving capability.

With Theorem 9, we can predict the maximum speedup achievable. We estimate $A_0 \doteq 1/17 \doteq 6\%$ because Adapt first failed at task #17. We also estimate $c \doteq 0.1$ from empirical observation. Again, if we believe that (15) is also a lower bound, then the maximum cost ratio is $r/\bar{\mu} + (1 + 0.1 - r/\bar{\mu})0.06 \doteq 0.32$ from the plot, which implies that $r/\bar{\mu} \doteq 0.27$, which is incidentally very close to the cost ratio at the end of task #100. Consequently, we do not anticipate Adapt to do much better with more training.

6 Conclusion

We have presented a learning algorithm that can improve path planning. The algorithm adapts to its working environment by maintaining an experience graph with vertices corresponding to useful robot configurations. It can both reduce time cost and increase task solving capability of existing planners.

To gain insight into this algorithm, we have presented some theoretical analysis based on two stochastic models: pessimistic $\mathcal{M}_p$ and randomized $\mathcal{M}_r$. The models have different assumptions and applications: $\mathcal{M}_p$ quantifies C-space complexity while $\mathcal{M}_r$ quantifies experience utility. Using these models, we characterize the situations in which speedup learning is useful, and provide global quantitative bounds on planning cost and capability in terms of training cost. We have also demonstrated the applicability and fidelity of our analysis on several robot path planning environments. In particular, we have illustrated a technique for predicting the maximum achievable speedup. Our theoretical results and techniques are elementary and should be useful for studying other types of probabilistic learning as well.

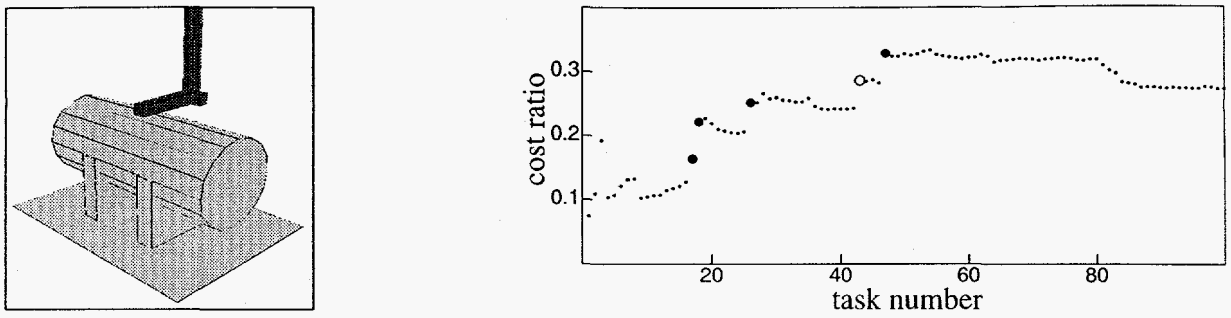

Figure 5: Time improvement on a 3-d, 6-dof robot problem.

References

[1] Barbehenn, M., Chen, P.C., Hutchinson, S., "An Efficient Hybrid Planner in Changing Environments," *Proc. of IEEE Int. Conf. on Robotics and Automation,* pp. 2755–2760, 1994.

[2] Barraquand, J. and Latombe, J., "A Monte-Carlo algorithm for path planning with many degrees of freedom," *Proc. of IEEE Int. Conf. on Robotics and Automation,* pp. 1712–1717, 1990.

[3] Bessiere, P., Ahuactzin, J.M., *et al.*, "The 'Ariadne's Clew' algorithm: Global planning with local methods," *Proc. of IEEE/RSJ Conf. on Intelligent Robots and Systems,* 1993.

[4] Chen, P.C., "Effective Path Planning through Task Restrictions," Sandia Report SAND91-1964, 1992.

[5] Chen, P.C., "Improving Path Planning with Learning," *Machine Learning: Proc. of the Ninth Int. Conf.,* pp. 55–61, 1992.

[6] Chen, P.C. and Hwang, Y.K., "SANDROS: A Motion Planner with Performance Proportional to Task Difficulty," *Proc. of IEEE Int. Conf. on Robotics and Automation,* pp. 2346–2353, 1992.

[7] Chen, P.C., "Adaptive Path Planning for Flexible Manufacturing," *Proc. of Fourth Int. Conf. on Computer Integrated Manufacturing and Automation Technology,* Oct., 1994.

[8] Feller, W., *An Introduction to Probability Theory and Its Application,* 3rd edition, v. 1, John Wiley & Sons, 1968.

[9] Goel, A., Callantine, T, Donnelian, M., Vazquez, N., "An intergrated experience-based approach to navigational path planning for autonomous mobile robot," *Proc. of IEEE Int. Conf. on Robotics and Automation,* pp. 818–825, 1993.

[10] Gupta, K.K., Zhu, X., "Practical Global Motion Planning for Many Degrees of Freedom: A Novel Approach within Sequential Framework," *Proc. of IEEE Int. Conf. on Robotics and Automation,* pp. 2038–2043, 1994.

[11] Harrigan, R.W., Sanders, T.L., "A Robotic System to Conduct Radiation and Contamination Surveys on Nuclear Waste Transport Casks," Sandia Report SAND89-0017, 1990.

[12] Hwang, Y.K. and Ahuja, N., "Gross Motion Planning – A Survey," *ACM Computing Surveys* vol 24, no 3, pp. 219–292, Sept. 1992.

[13] Kavraki, L. and Latombe, J.-C, "Randomized preprocessing of configuration space for fast path planning," *Proc. of IEEE Int. Conf. on Robotics and Automation,* pp. 2138–2145, 1994.

[14] Kondo, K., "Motion Planning with Six Degrees of Freedom by Multistrategic Bidirectional Heuristic Free-Space Enumeration", *IEEE Tran. on Robotics and Automation,* vol. 7, no. 3, pp. 267–277, June 1991.

[15] Latombe, J., *Robot Motion Planning,* Kluwer Academic Publishers, 1991.

[16] Lozano-Pérez, T., "A Simple Motion-Planning Algorithm for General Robot Manipulators," *IEEE J. of Robotics and Automation,* vol. RA-3, no. 3, pp. 224–238, June 1987.

[17] Marshall, A.W., Olkin, I., *Inequalities: Theory of Majorization and Its Applications,* Academic Press, 1979.

[18] Natarajan, B.K., *Machine Learning: A Theoretical Approach,* Morgan Kaufmann, 1991.

[19] Overmars, M.H., VSvestka, P., "Probabilistic Learning Approach to Motion Planning," *Workshop on the Algorithmic Foundations of Robotics,* Feb. 1994.

[20] Pandya, S., Hutchinson, S., "A Case-based Approach to Robot Motion Planning," *Proc. of IEEE Int. Conf. on Systems Man and Cybernetics,* pp. 492–497, 1992.

[21] Paden, B., Mees, A. and Fisher, M., "Path Planning Using a Jacobian-Based Freespace Generation Algorithm," *Proc. of IEEE Int. Conf. on Robotics and Automation,* pp. 1732–1737, 1989.

[22] Preparata, F.P., Shamos, M.I., *Computational Geometry: An Introduction,* Springer-Verlag, 1988.

[23] Tadepalli, P., "A Theory of Unsupervised Speedup Learning," *Proc. of AAAI,* pp. 229–234, 1992.